

ETH Lecture 401-4671-00L Advanced Numerical Methods for CSE

Advanced Numerical Methods for Computational Science and Engineering

Prof. R. Hiptmair, SAM, ETH Zurich

Autumn Term 2023
(C) Seminar für Angewandte Mathematik, ETH Zürich

[Link to the current version of this lecture document](#)



Always under construction!

The online version will always be work in progress and subject to change.

(Nevertheless, structure and main contents can be expected to be stable)

Contents

0	Introduction	7
0.1	Course Contents	12
0.1.1	Focus of this Course	12
0.1.2	Prerequisite Knowledge	12
0.1.3	Goals	12
0.1.4	Requests for Student Activity	13
0.1.5	Literature	13
0.2	Specific information	14
0.2.1	Assistants and exercise classes	14
0.2.2	Assignments	14
0.2.3	Information on Examinations	15
1	Boundary Element Methods (BEM)	17
1.0.1	Further Reading for this Chapter	19
1.1	Elliptic Model Boundary Value Problem: Electrostatics	19
1.1.1	The Electric Field	19
1.1.2	Electric Scalar Potential	21
1.1.3	Continuity of Fields and Boundary Conditions	24
1.1.4	Equilibrium Conditions	27
1.1.5	Variational Equations	29
1.1.6	Boundary Value Problems	30
1.1.7	Decay conditions on unbounded domains	33
1.1.8	Supplement: An energy norm for source charge distributions	35
1.2	Boundary Representation Formulas	36
1.2.1	Green's Formulas	36
1.2.2	Fundamental Solutions	38
1.2.2.1	Potential of a Point Charge	38
1.2.2.2	Potential of a Line Charge	39
1.2.2.3	Distributional View: $\mathbf{L}G = \delta_0$	40
1.2.3	Volume Potential Representation	44
1.2.4	Boundary Potential Representation	46
1.2.5	Layer Potentials	48
1.2.5.1	Single Layer Potential	49
1.2.5.2	Double Layer Potential	51
1.2.6	Green's Functions	52
1.3	Boundary Integral Equations (BIEs)	55
1.3.1	Trace Operators	55
1.3.1.1	Dirichlet Trace	56
1.3.1.2	Neumann Trace	59
1.3.2	Mapping Properties of Layer Potentials	63

1.3.3	Jump Relations for Layer Potentials	65
1.3.4	Boundary Integral Operators (BIOs)	68
1.3.4.1	Formal Definition	68
1.3.4.2	Integral Representations	69
1.3.4.3	Variational Form for Hypersingular BIO	71
1.3.5	Direct Boundary Integral Equations	74
1.3.5.1	First-kind BIEs	76
1.3.5.2	Second-kind BIEs	79
1.3.6	Indirect Boundary Integral Equations	80
1.4	Boundary Element Methods in Two Dimensions	82
1.4.1	Abstract Galerkin Discretization	83
1.4.2	Boundary Element Spaces on Curves	85
1.4.2.1	Curve Partitionings	86
1.4.2.2	Piecewise Polynomial Functions on Curves	87
1.4.2.3	Shape Functions	88
1.4.2.4	Solving Boundary Value Problems via Galerkin BEM	91
1.4.2.5	Approximation of Curves	93
1.4.3	Computation of BEM-Galerkin Matrix in 2D	95
1.4.3.1	Panel-oriented Assembly	95
1.4.3.2	Lowest-order BEM on Polygons: Analytic Formulas	101
1.4.3.3	Recapitulated [NumCSE Chapter 7]: Aspects of Numerical Quadrature	109
1.4.3.4	Matrix Entries by Quadrature	120
1.5	Boundary Element Methods on Closed Surfaces	129
1.5.1	Surface Meshes	129
1.5.2	Boundary Element Spaces on Triangulated Surfaces	131
1.5.2.1	Definitions	131
1.5.2.2	Shape Functions	133
1.5.3	Assembly of Galerkin Matrices	135
1.6	BEM: Various Aspects	141
1.6.1	Convergence	141
1.6.1.1	Abstract Galerkin Error Estimate	141
1.6.1.2	Approximation in Boundary Element spaces	142
1.6.1.3	Variational Crimes	147
1.6.1.4	Pointwise Recovery of Solutions	148
1.6.2	Mixed Boundary Value Problems	148
1.6.3	Transmission Problems	150
1.6.3.1	Two-Domain Setting	150
1.6.3.2	Multi-Domain Transmission Problem	154
1.6.4	BEM for Wave Propagation	154
2	Local Low-Rank Compression of Non-Local Operators	157
2.1	Examples: Non-Local Operators	158
2.1.1	(Discretized) Integral Operators	159
2.1.2	Long-Range Interactions in Discrete Models	162
2.1.3	Kernel Collocation Matrices	166
2.2	Approximation of Kernel Collocation Matrices	167
2.2.1	Separable (= Low-Rank) Kernel Approximation	169
2.2.1.1	Polynomial Expansions	171
2.2.1.2	Uni-directional Interpolation	173
2.2.1.3	Bi-directional interpolation	177
2.2.2	Error Estimates and Admissibility Condition for Singular Kernels	179

2.2.2.1	Truncation Error Estimates for Taylor Expansion	180
2.2.2.2	Interpolation Error Estimate for Chebychev Interpolation	182
2.2.2.3	Estimates for Bi-Directional Interpolation	186
2.3	Clustering Techniques	190
2.3.1	Local Separable Approximation	190
2.3.2	Cluster Trees	198
2.3.3	Building Near- and Far-Field Blocks	207
2.3.4	Storing Block-Partitioned Kernel Collocation Matrix	213
2.3.5	Matrix \times Vector: Efficient Implementation	221
2.3.6	Panel Clustering	223
2.4	Hierarchical Matrices	227
2.4.1	Definition	227
2.4.2	Low-Rank Matrices: Algorithms	234
2.4.3	\mathcal{H} -Addition of Hierarchical Matrices	240
2.4.4	\mathcal{H} -Multiplication of Hierarchical Matrices [Bör21, Sect. 5.6]	241
2.4.5	Hierarchical LU-Decomposition	253
2.4.6	\mathcal{H}^2 -Matrices	259
3	Convolution Quadrature	268
3.1	Basic Concepts and Tools	269
3.1.1	Convolution of Causal Functions	269
3.1.2	Discrete Convolutions	273
3.1.3	The Laplace Transform	276
3.1.4	Diagonalizing Convolutions	281
3.1.5	Toeplitz Matrix Numerical Linear Algebra	285
3.2	Convolution Equations: Examples	291
3.2.1	Tomography: Abel Integral Equation	291
3.2.2	Impedance Boundary Conditions	293
3.2.3	Time-Domain Boundary Integral Equations	296
3.3	Implicit-Euler Convolution Quadrature	299
3.3.1	Setting and Goal	299
3.3.2	Derivation of Implicit Euler CQ	301
3.3.3	Properties of implicit-Euler Convolution Quadrature	307
3.3.4	Convergence	310
3.4	Multistep Convolution Quadrature (MSCQ)	313
3.4.1	Linear Multi-Step Numerical Integrators	314
3.4.2	Multi-Step Convolution Quadrature: Weights	321
3.4.3	Multi-Step Convolution Quadrature: Algorithms	329
3.5	Runge-Kutta Convolution Quadrature (RKCCQ)	334
3.5.1	Implicit Runge-Kutta Single-Step Methods	335
3.5.2	Runge-Kutta CQ weights	336
3.6	Fast and Oblivious Convolution Quadrature	339
4	(Algebraic) Multigrid Methods	358
4.1	Solvers for Finite Element Linear Systems	358
4.1.1	Elliptic Model Boundary Value Problems	358
4.1.2	Sparse Elimination Solvers	365
4.1.3	Stationary Linear Iterations (SLIs)	366
4.1.4	Conjugate Gradient Method (CG)	371
4.2	Geometric Multigrid Method	373
4.2.1	Subspace Correction Methods	375
4.2.2	Convergence of SSC Methods	378

4.2.3	Coarse-Grid Correction (CGC)	385
4.2.4	Multigrid Iteration	389
4.2.5	Multigrid Preconditioning	393
4.2.6	Full Approximation Storage Multigrid (FAS)	396
4.3	Algebraic Multigrid (AMG): Matrix-Based Multigrid	400
4.3.1	AMG Framework	401
4.3.2	AMG Heuristics	405
4.3.3	\mathcal{C}/\mathcal{F} Splitting Algorithm	407
4.3.4	AMG Prolongation	410
5	Reduced Bases Methods (RBM)	415
5.1	Parameterized Boundary Value Problems	415
5.1.1	Coefficients as Parameters	415
5.1.2	Parameter-Dependent Domains	415
5.1.3	Abstract Framework	415
5.2	Reduced Bases Methods: Ideas and Algorithms	416
5.2.1	Prelude: Polynomial Interpolation	416
5.2.2	Projected Variational Problem	416
5.2.3	Generation of Reduced Bases	416
5.2.3.1	Proper Orthogonal Decomposition (POD)	416
5.2.4	Special Case: Separable Decomposition	416
5.3	Error Estimation	416
5.3.1	Residual-Based Estimator	416
5.3.2	Computation of Residual Norm	416
5.3.3	Lower Bound for $\gamma_h(\boldsymbol{\pi})$	416
5.4	Separable Approximation	416
5.4.1	Interpolation on Parameter Space	416
5.4.2	Adaptive Cross Approximation (ACA)	416
Index		417
	Acronyms	424
	Symbols	425
	Examples	427
	Codes	430

Chapter 0

Introduction

0.1 Course Contents

0.1.1 Focus of this Course

This course discusses *modern numerical methods* involving complex numerical techniques with an emphasis on *algorithms* and intricate data structures that render an efficient implementation non-trivial.

The course comprises various rather independent topics organized in chapters. Usually, not all chapters contained in this lecture document will be covered in the course.

Classroom instruction will be accompanied by *coding projects* in C++ assigned as homework.

0.1.2 Prerequisite Knowledge

This course has to take for granted a range of skills MSc students in CSE are expected to have:

- ◆ “Fluency” in C++,
- ◆ familiarity with basic numerical methods (as taught in the course “Numerical Methods for CSE”, [Link](#) to lecture documents), and
- ◆ knowledge about the finite element method for elliptic partial differential equations (as taught in the course “Numerical Methods for Partial Differential Equations”, [Link](#) to lecture documents).

0.1.3 Goals

- ◆ Appreciation of the interplay of functional analysis, advanced calculus, numerical linear algebra, and sophisticated data structures in modern computer simulation technology.
- ◆ Knowledge about the main ideas and mathematical foundations underlying boundary element methods, hierarchical matrix techniques, convolution quadrature, and reduced basis methods.
- ◆ Familiarity with the algorithmic challenges arising from these methods and the main ways on how to tackle them.
- ◆ Knowledge about the algorithms’ complexity and suitable data structures.
- ◆ Ability to understand details of given implementations.
- ◆ Skills concerning the implementation of algorithms and data structures in C++.

Indispensable:	Learning by doing (→ do exercises consistently)
----------------	---

0.1.4 Requests for Student Activity

The lecturers very much welcome and, putting it even more strongly, rather depend on feedback and suggestions of the students taking the course for continuous improvement of the course contents and presentation. Therefore all participants are strongly encouraged to get involved actively and contribute in the following ways:

§0.1.4.1 (DISCUNA communication platform)



Communication between lecturers and assistants on one side and students on the other side will rely on the [DISCUNA](#) online collaboration platforms that runs in any browser.

- ◆ [DISCUNA](#) allows to attach various types of *annotations* to shared PDF documents, see [documentation](#).
- ◆ [DISCUNA](#) offers *communication channels* that allow users to post messages (formatted text and images) and see and comment on other posts, see the [documentation](#).

In the beginning of the teaching period you receive a **join link** of the form `https://app.discuna.com/invite/<JOIN CODE>`. Open the link in a web browser and it will take you to the [DISCUNA](#) community page. ┘

§0.1.4.2 (Reporting errors) As the documents for this course will always be in a state of flux, they will inevitably and invariably be afflicted with small errors, mainly typos and omissions.

Error reporting through [DISCUNA](#)

For error reporting we use [DISCUNA](#): Errors should be reported by annotating the PDFs you can find in the `Error Reporting` folder on the [DISCUNA](#) AdvNumCSE community page.

To report an error,

1. select the corresponding PDF document (chapter of the lecture document of homework problem) in the left sidebar,
2. press the prominent white-on-blue **+**-button in the right sidebar,
3. click on the displayed PDF where the error is located,
4. then in the pop-up window choose the “Error” category,
5. and add a title and,
6. if the title does not tell everything, a short description.

In case you cannot or do not want to link an error to a particular point in the PDF, you may just click on the title page of the respective chapter. Then, please precisely specify the concerned section and the number of the paragraph, remark, equation etc. *Do not give page numbers* as they may change with updates to the documents.

Note that chapter PDFs and homework problem files will gradually be added to the [DISCUNA](#) AdvNumCSE community. Hence, the final chapters will not be accessible in the beginning of the course. ┘

0.1.5 Literature

Parts of the following monographs may be used as supplementary reading for this course. References to relevant sections will be provided in the course material.

Studying extra literature is not important for following this course!

- ◆ Chapter 2 M. BEBENDORF, *Hierarchical matrices: A means to efficiently solve elliptic boundary value problems*, Springer, 2008.
- ◆ Chapter 2 W. HACKBUSCH, *Hierarchical Matrices*, Springer, 2015.
- ◆ Chapter 2 S. BOERM, *Efficient Numerical Methods for Non-Local Operators: H2-Matrix Compression, Algorithms and Analysis*, EMS, 2010.
- ◆ Chapter 2 S. BOERM, *Numerical Methods for Non-Local Operators*, Lecture Notes Univ. Kiel, 2017.
- ◆ Chapter 3: M. HASSELL AND F.-J. SAYAS, *Convolution Quadrature for Wave Simulations*, Springer, 2016.
- ◆ Chapter 3: F.-J. SAYAS, *Retarded Potentials and Time-Domain Boundary Integral Equations*, Springer, 2016.
- ◆ Chapter 4: K. STÜBEN, *An Introduction to Algebraic Multigrid*, Appendix A of U. TROTTEBERG, C. OSTERLEE, AND A. SCHÜLLER, *Multigrid*, Academic Press, 2001.
- ◆ Chapter 4: J. XU AND L. ZIKATANOV, *Algebraic multigrid methods*, *Acta Numerica*, 26 (2017), pp. 591–721.

0.2 Specific information

0.2.1 Assistants and exercise classes

The course comprises both classroom lectures (four hours per week) and tutorial classes (two hours per week) taught by assistants, who are either PhD students and postdocs at the Seminar for Applied Mathematics of ETH Zürich or advanced MSc students in the CSE program.

Though the assistants email addresses are provided on the course website, their use should be restricted to cases of emergency:

In general **refrain from sending email messages** to the lecturer or the assistants. They will not be answered!

Questions should be asked in class (in public or during the break in private), during the tutorial sessions, or through the **DISCUNA** Q&A communication channels.

0.2.2 Assignments

You should expect to spend 6–10 hours per week on trying to solve the homework problems. Since many involve small coding projects, the time it will take an individual student to arrive at a solution is hard to predict.

§0.2.2.1 (Homeworks and tutors' corrections)

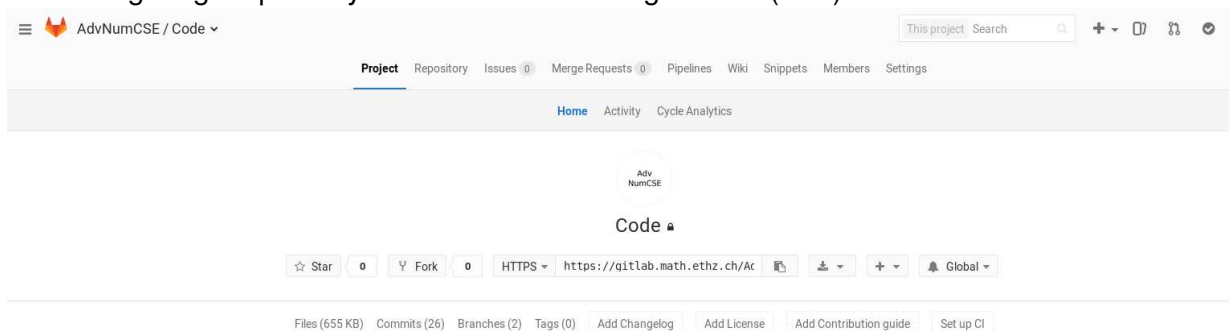
- ◆ The weekly assignments will be a few problems from the **ADVNCSE Problem Collection** available online as PDF. The particular problems to be solved will be communicated on **Wednesday** every week.

Please note that this problem collection is being compiled during this semester. Thus, make sure that you **obtain the most current version** every week.

- ◆ Some or all of the problems of an assignment sheet will be discussed in the tutorial classes on Thursday or Friday several days after the problems have been assigned.
- ◆ If you want your tutor to examine your solution of the current problem sheet, please hand it in to the tutor during the following exercise class, or use the dedicated online upload tool. This is voluntary, but feedback on your performance on homework problems can be important.
- ◆ Please clearly mark the homework sub-problems that you want your tutor to inspect.
- ◆ You are encouraged to hand-in incomplete and wrong solutions, because you can receive valuable feedback even on incomplete attempts.

┌

§0.2.2.2 (Git code repository) C++ codes for both the classroom and homework problems are made available through a git repository also accessible through Gitlab ([Link](#)):



The Gitlab toplevel page gives a short introduction into the repository for the course and provides a link to online sources of information about Git.

Download is possible via Git or as a zip archive. Which method you choose is up to you, but it should be noted that updating via git is more convenient.

➤ Shell command to download the git repository:

```
> git clone https://gitlab.math.ethz.ch/AdvNumCSE/Code
```

Updating the repository to fetch upstream changes is then possible by executing `> git pull` inside the Code folder.

Note that by default participants of the course will have **read access only**. However, if you want to *contribute corrections and enhancements* of lecture or homework codes you are invited to submit a **merge request**. Beforehand you have to inform your tutor so that a personal Gitlab account can be set up for you.

The Zip-archive download link is [here](#).

For instructions on how to compile assignments or lecture codes see the **README** file.

┌

0.2.3 Information on Examinations

§0.2.3.1 (Examination during the teaching period) From the ETH course directory:

During the teaching period students are expected to give a 10-15-minute oral code review and answer questions concerning selected homework programming assignments. The code review is regarded as a mandatory performance element that will contribute 20% of the final grade.

The **oral code review** is regarded as a **mandatory performance assessment** and is graded with an integer grade, which will contribute $\frac{1}{5}$ th of the final grade.

The code review will be held in December.

- ◆ The code review take 15 to 20 minutes and will center around the discussion of (parts of) codes the students have written as solutions of homework coding projects.
- ◆ The relevant homework coding projects will be communicated tow weeks before the first code reviews and will also be listed on the course web page.
- ◆ **Registration** through a Doodle poll is mandatory for taking parts in the code review. The link will be send by email and will be published on the course web page.

Non-registration or not showing up will incur a grade of 1.0 for the code review.

- ◆ Candidates for the code review are expected to send their codes until two days before their scheduled code review. Please upload all your files as a single `.zip` archive with different codes in different sub-directories;

Details on code upload will be published on the course web page and sent by email.

- ◆ No repetition of code reviews will be offered.

└

§0.2.3.2 (Main examination during the exam session)

- ◆ 30-minute **oral exam** in English
- ◆ Dates will be communicated by the ETH exam office and cannot be negotiated.
- ◆ Subjects of examination:

All topics, which have been addressed in class or in a homework problem or project

└

§0.2.3.3 (Repeating an exam)

Main exam.

- The main exam can be repeated once, conditional on failure.
- The grade earned in the code review will be taken into account again for the repeated exam.

└

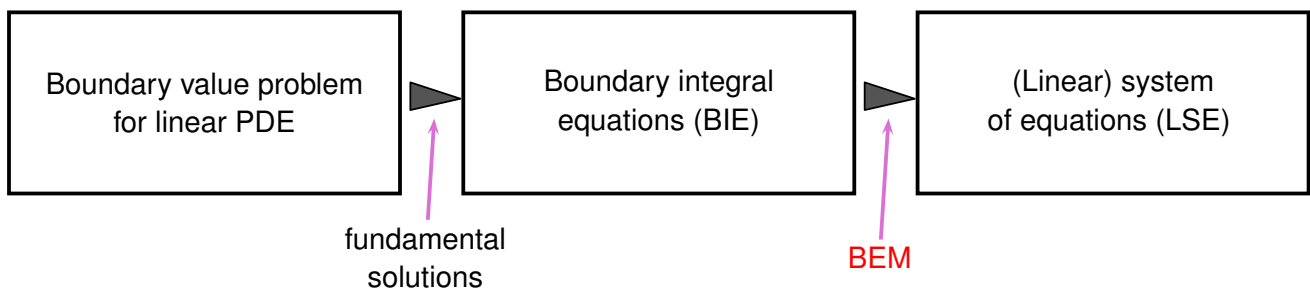
Chapter 1

Boundary Element Methods (BEM)

Preface

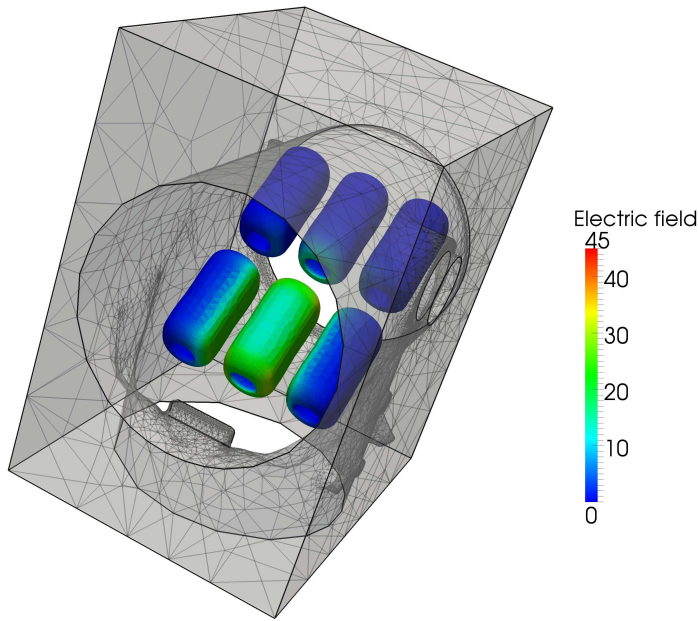
Boundary element methods (BEM) represent a class of numerical methods for the discretization of boundary integral equations (BIE) arising from boundary value problems (BVPs) for linear partial differential equations (PDEs) with constant coefficients.

In this chapter we focus on the derivation of various boundary integral equations, the study of their properties and on Galerkin discretization by means of boundary element methods, which can be regarded a finite element methods for BIE.



Boundary element methods play a significant role in computational engineering, in particular in the fields of computational electromagnetism and acoustics, and for simulations based on linear elasticity.

§1.0.0.1 (BEM in computational electromagnetics)



The plot shows the post-processed result of an **electrostatic field simulation** conducted by Lars Kielhorn for a test geometry provided by ABB Research, Baden/Dättwil (The strength of the electric field is given in units of $\frac{V}{m}$).

Computations were done by means of a low-order piecewise polynomials Galerkin boundary element method based on the boundary element library BETL [HK12].

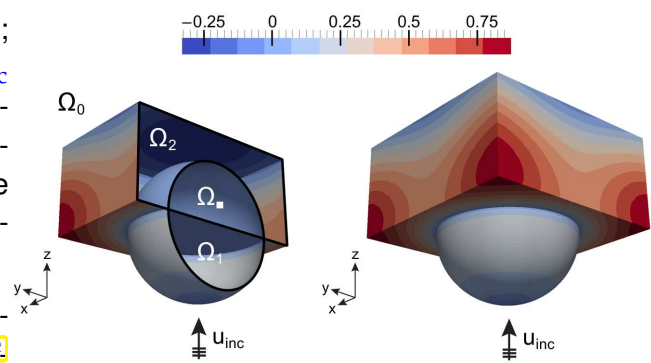
The mesh used for the computations is faintly drawn for the outer casing.

§1.0.0.2 (BEM for acoustic wave propagation)

A result from [CHS18]:

Acoustic wave propagation in frequency domain; scattering of an incident plane acoustic wave U_{inc} at a scatterer composed of three different homogeneous isotropic parts, of which Ω_{\blacksquare} is perfectly absorbing (sound soft). The color scale indicates the amplitude of the total acoustic pressure field on a surface.

Simulation was based on piecewise constant boundary element applied to a second-kind single trace of direct boundary integral equation formulation.



Contents

1.0.1	Further Reading for this Chapter	19
1.1	Elliptic Model Boundary Value Problem: Electrostatics	19
1.1.1	The Electric Field	19
1.1.2	Electric Scalar Potential	21
1.1.3	Continuity of Fields and Boundary Conditions	24
1.1.4	Equilibrium Conditions	27
1.1.5	Variational Equations	29
1.1.6	Boundary Value Problems	30
1.1.7	Decay conditions on unbounded domains	33
1.1.8	Supplement: An energy norm for source charge distributions	35
1.2	Boundary Representation Formulas	36
1.2.1	Green's Formulas	36
1.2.2	Fundamental Solutions	38
1.2.3	Volume Potential Representation	44
1.2.4	Boundary Potential Representation	46
1.2.5	Layer Potentials	48
1.2.6	Green's Functions	52
1.3	Boundary Integral Equations (BIEs)	55
1.3.1	Trace Operators	55

1.3.2	Mapping Properties of Layer Potentials	63
1.3.3	Jump Relations for Layer Potentials	65
1.3.4	Boundary Integral Operators (BIOs)	68
1.3.5	Direct Boundary Integral Equations	74
1.3.6	Indirect Boundary Integral Equations	80
1.4	Boundary Element Methods in Two Dimensions	82
1.4.1	Abstract Galerkin Discretization	83
1.4.2	Boundary Element Spaces on Curves	85
1.4.3	Computation of BEM-Galerkin Matrix in 2D	95
1.5	Boundary Element Methods on Closed Surfaces	129
1.5.1	Surface Meshes	129
1.5.2	Boundary Element Spaces on Triangulated Surfaces	131
1.5.3	Assembly of Galerkin Matrices	135
1.6	BEM: Various Aspects	141
1.6.1	Convergence	141
1.6.2	Mixed Boundary Value Problems	148
1.6.3	Transmission Problems	150
1.6.4	BEM for Wave Propagation	154

1.0.1 Further Reading for this Chapter

Note that the information given in class and contained in this lecture document is supposed to be self-contained. Studying additional literature should not be necessary. However, if you are interested in learning more about boundary element methods you may take a look at

- ◆ S. SAUTER AND CH. SCHWAB, *Boundary Element Methods*, Springer, 2010.
- ◆ O. STEINBACH, *Numerical approximation methods for elliptic boundary value problems*, Springer, 2008.

1.1 Elliptic Model Boundary Value Problem: Electrostatics

We consider electromagnetism in a *stationary* setting, that is none of the fields depends on time. In this case electric and magnetic fields become decoupled. In this section we focus on the electric field as we did in [NumPDE Section 1.2.2].

1.1.1 The Electric Field

§1.1.1.1 (Domains) We denote by $\Omega \subset \mathbb{R}^3$, called a *domain* in the sequel, an open subset of 3D Euclidean space with piecewise smooth Lipschitz boundary. For the intricate mathematical notion of a Lipschitz boundary we refer to [McL00, pp. 89] and [SS10, Def. 2.2.7]. If Ω is bounded, you may imagine a polyhedron with some curved faces, see § 1.2.1.4 below.

As a new aspect we will also consider boundary value problems for fields on *unbounded domains*, more precisely, the case when Ω is the (open) complement of a bounded Lipschitz domain $\subset \mathbb{R}^3$.

📎 Notation: $\Omega' := \mathbb{R}^d \setminus \overline{\Omega} \triangleq$ complement of a domain $\overline{\Omega} \subset \mathbb{R}^d$

┘

The simplest mathematical model for a stationary electric field is that of a *vectorfield* $\mathbf{E} : \Omega \rightarrow \mathbb{R}^3$, assigning a field vector $\mathbf{E}(\mathbf{x}) \in \mathbb{R}^3$ to each point $\mathbf{x} \in \Omega$.

✎ Notation: We write $\mathbf{a}, \dots, \mathbf{x}, \mathbf{y}, \mathbf{z}$ for small vectors and points in space.
 bold typeface for vector-valued quantities: $\mathbf{E}, \mathbf{u}, \mathbf{j}, \dots$

§1.1.1.2 (Energy (density) of electric field) Any non-zero electric field contains energy, which determined by both the strength of the electric field and the **dielectric medium** penetrated by the field. We restrict ourselves to simple **linear media**. In this case the we have the following expression for the energy:

Definition 1.1.1.3. Electrostatic field energy [NumPDE Eq. (1.2.2.6)]

The total **energy** of an electric field $\mathbf{E} : \Omega \rightarrow \mathbb{R}^3$ inside Ω is

$$J_{\text{el}}(\mathbf{E}) := \frac{1}{2} \int_{\Omega} (\epsilon(\mathbf{x})\mathbf{E}(\mathbf{x})) \cdot \mathbf{E}(\mathbf{x}) \, d\mathbf{x}, \tag{1.1.1.4}$$

where $\epsilon : \Omega \rightarrow \mathbb{R}^{3,3}$ is the **symmetric**, bounded, **uniformly positive definite dielectric tensor** field, see [NumPDE § 1.2.2.7].

We call a tensor field, that is, a matrix-valued function $\alpha : \Omega \rightarrow \mathbb{R}^{d,d}$, $d \in \mathbb{N}$, bounded and **uniformly positive definite** [NumPDE Def. 1.2.2.9], if

$$\exists \gamma^-, \gamma^+ > 0: \quad \gamma^- \|z\|^2 \leq z^T \alpha(\mathbf{x}) z \leq \gamma^+ \|z\|^2 \quad \forall z \in \mathbb{R}^d. \tag{1.1.1.5}$$

ϵ is a macroscopic material parameter taking into account complex microscopic interactions of electric fields and matter.

Energy norm

$\mathbf{E} \mapsto \sqrt{J_{\text{el}}(\mathbf{E})}$ defines a **norm** (\rightarrow [NumPDE Def. 0.3.1.10]) on the vector space of electric fields, the **energy norm**, cf. [NumPDE ??].

Remark 1.1.1.7 (Scaling of electromagnetic field problems, cf. [NumPDE Rem. 1.2.1.25])

Quantity	units
Electric field \mathbf{E}	$[\mathbf{E}] = 1 \frac{\text{V}}{\text{m}}$
Dielectric tensor ϵ	$[\epsilon] = 1 \frac{\text{As}}{\text{Vm}}$
Charge density ρ	$[\rho] = 1 \frac{\text{As}}{\text{m}^3}$
Field energy	$[J_{\text{el}}] = 1 \text{ VAs} = 1 \text{ J}$

The physical units of electrostatic quantities are given beside \triangleright

There are three “free units”, **1V** (unit of voltage), **1m** (unit of length), and **1As** (unit of charge), which can be fixed arbitrarily. For instance, one may set the unit of length to the diameter of Ω , if Ω is bounded, and set the units of voltage and charge to the “maximum expected values”.

Thus, one ends up with non-dimensional equations for electrostatics. In this course we will tacitly assume that equations have already been converted into non-dimensional form by suitable scaling. \lrcorner

1.1.2 Electric Scalar Potential

A point charge q in a (continuous) electric field $\mathbf{E} : \Omega \rightarrow \mathbb{R}^3$ at a point $\mathbf{x} \in \Omega$ experiences a *Coulomb force*

$$\mathbf{f}(\mathbf{x}) := q \mathbf{E}(\mathbf{x}). \tag{1.1.2.1}$$

(Note matching physical units $[\mathbf{f}] = \text{As} \frac{\text{V}}{\text{m}} = \frac{\text{J}}{\text{m}} = 1 \text{ N}$)

§1.1.2.2 (Vanishing circulation of electric fields) The integration of a force along a directed curve $\gamma : [0, 1] \mapsto \gamma(t) \in \Omega$ yields the *work* required to move the charge: $W = q \int_{\gamma} \mathbf{E} \cdot d\vec{s}$. Thus, in order to comply with the fundamental principle of energy conservation we have to demand

$$\int_{\gamma} \mathbf{E} \cdot d\vec{s} = \int_0^1 \mathbf{E}(\gamma(t)) \cdot \frac{d\gamma}{dt}(t) dt = 0 \quad \forall \text{ closed curves } \gamma \subset \Omega, \tag{1.1.2.3}$$

where the curve $\gamma : [0, 1] \rightarrow \Omega$ is called closed, if $\gamma(0) = \gamma(1)$. The *non-local* property (1.1.2.3) has an important *local* consequence, which can be stated by means of the **rotation operator** (also known as curl operator)

$$\mathbf{curl} \mathbf{v}(\mathbf{x}) := \begin{bmatrix} \frac{\partial v_3}{\partial x_2}(\mathbf{x}) - \frac{\partial v_2}{\partial x_3}(\mathbf{x}) \\ \frac{\partial v_1}{\partial x_3}(\mathbf{x}) - \frac{\partial v_3}{\partial x_1}(\mathbf{x}) \\ \frac{\partial v_2}{\partial x_1}(\mathbf{x}) - \frac{\partial v_1}{\partial x_2}(\mathbf{x}) \end{bmatrix}, \quad \text{for } \mathbf{v}(\mathbf{x}) = \begin{bmatrix} v_1(\mathbf{x}) \\ v_2(\mathbf{x}) \\ v_3(\mathbf{x}) \end{bmatrix} \text{ differentiable in } \mathbf{x}. \tag{1.1.2.4}$$

Theorem 1.1.2.5. Electric fields are irrotational/curl-free

Every differentiable stationary electric field $\mathbf{E} : \Omega \rightarrow \mathbb{R}^3$ satisfies $\mathbf{curl} \mathbf{E} = \mathbf{0}$ in Ω .

Proof. We assume $\mathbf{0} \in \Omega$ and show $\mathbf{curl} \mathbf{E}(\mathbf{0}) = \mathbf{0}$. Pick $i, j \in \{1, 2, 3\}, i \neq j$, and consider the closed curve describing an axes-aligned square of size $h > 0$:

$$\gamma = \{t \mapsto hte_i\} \cup \{t \mapsto he_i + the_j\} \cup \{t \mapsto he_i + he_j - hte_i\} \cup \{t \mapsto (1-t)he_j\}, \quad 0 \leq t \leq 1,$$

with \mathbf{e}_i standing for the i -th Cartesian basis vector. The path integral evaluates to

$$\int_{\gamma} \mathbf{E} \cdot d\vec{s} = \int_0^1 h\mathbf{E}(hte_i) \cdot \mathbf{e}_i + h\mathbf{E}(he_i + the_j) \cdot \mathbf{e}_j - \int_0^1 h\mathbf{E}(he_j + h(1-t) \cdot \mathbf{e}_i) \cdot \mathbf{e}_i - \int_0^1 h\mathbf{E}(h(1-t)\mathbf{e}_j) \cdot \mathbf{e}_j.$$

We plug in the first-order Taylor expansion of \mathbf{E} around $\mathbf{0}$:

$$\mathbf{E}(\mathbf{x}) = \mathbf{E}(\mathbf{0}) + \mathbf{DE}(\mathbf{0})\mathbf{x} + O(\|\mathbf{x}\|^2) \quad \text{for } \mathbf{x} \rightarrow \mathbf{0}. \tag{1.1.2.6}$$

 Notation: $\mathbf{DE} \hat{=}$ Jacobian of the (differentiable) vector field \mathbf{E} , see [NumPDE Eq. (0.3.2.16)].

$$\blacktriangleright \int_{\gamma} \mathbf{E} \cdot d\vec{s} = h^2 \left(\frac{1}{2} \mathbf{DE}(\mathbf{0}) \mathbf{e}_i \cdot \mathbf{e}_i + \mathbf{DE}(\mathbf{0}) \mathbf{e}_i \cdot \mathbf{e}_j + \frac{1}{2} \mathbf{DE}(\mathbf{0}) \mathbf{e}_j \cdot \mathbf{e}_j - \mathbf{DE}(\mathbf{0}) \mathbf{e}_j \cdot \mathbf{e}_i - \frac{1}{2} \mathbf{DE}(\mathbf{0}) \mathbf{e}_i \cdot \mathbf{e}_i - \frac{1}{2} \mathbf{DE}(\mathbf{0}) \mathbf{e}_j \cdot \mathbf{e}_j \right) + O(h^3) \quad \text{for } h \rightarrow 0.$$

Note that multiplication with unit vectors selects rows/columns of matrices and that (1.1.2.3) makes the path integral along γ vanish.

$$\blacktriangleright 0 = \int_{\gamma} \mathbf{E} \cdot d\vec{s} = h^2 \left((\mathbf{DE}(\mathbf{0}))_{ij} - (\mathbf{DE}(\mathbf{0}))_{ji} \right) + O(h^3) \quad \text{for } h \rightarrow 0.$$

This implies $(\mathbf{DE}(\mathbf{0}))_{i,j} = (\mathbf{DE}(\mathbf{0}))_{j,i}$, the Jacobian $\mathbf{DE}(\mathbf{0})$ is *symmetric*: $\mathbf{DE}(\mathbf{0}) = \mathbf{DE}(\mathbf{0})^T$. In light of the definition (1.1.2.4) of the rotation operator we see that this is equivalent to $\mathbf{curl} \mathbf{E}(\mathbf{0}) = \mathbf{0}$. □

§1.1.2.7 (Introducing the electrostatic potential) As another consequence of (1.1.2.3) we note that for an open curve κ the integral $\int_{\kappa} \mathbf{E} \cdot d\vec{s}$ will depend only on the endpoints of the curve (**path-independence**); connect both endpoints of κ by another curve of opposite orientation. Therefore, picking an arbitrary point $z \in \Omega$ we can define an **electric potential** through

$$u(x) = - \int_{\kappa_x} \mathbf{E} \cdot d\vec{s} \quad \text{for some curve } \kappa_x : [0, 1] \rightarrow \Omega, \kappa_x(0) = z, \kappa_x(1) = x, x \in \Omega. \quad (1.1.2.8)$$

Thanks to path-independence of the work integral this is a valid definition.

Now, let us assume $0 \in \Omega$ and that Ω is *star-shaped* with respect to $z := 0$, that is for every $x \in \Omega$ we have $[0, x] \subset \Omega$. Then, for every $x \in \Omega$ we can choose the straight line connecting 0 and x as curve κ_x in (1.1.2.8):

$$\kappa_x(t) = tx, \Rightarrow u(x) := - \int_0^1 \mathbf{E}(tx) \cdot x dt. \quad (1.1.2.9)$$

By differentiation under the integral we get by the chain rule and the product rule

$$\mathbf{grad} u(x) := \begin{bmatrix} \frac{\partial u}{\partial x_1}(x) \\ \frac{\partial u}{\partial x_2}(x) \\ \frac{\partial u}{\partial x_3}(x) \end{bmatrix} = - \int_0^1 t \mathbf{DE}(tx)^\top x + \mathbf{E}(tx) dt.$$

Applying the same differentiation rules, we also obtain

$$\frac{d}{d\tau} \{ \tau \mapsto \tau \mathbf{E}(\tau x) \}_{\tau=t} = t \mathbf{DE}(tx)x + \mathbf{E}(tx).$$

Combining both formulas leads to a recovery of the electric field:

$$\mathbf{grad} u(x) = - \int_0^1 \frac{d}{d\tau} \{ \tau \mapsto \tau \mathbf{E}(\tau x) \}_{\tau=t} + t \underbrace{(\mathbf{DE}(tx)^\top - \mathbf{DE}(tx)) \cdot x}_{=0 \text{ by Thm. 1.1.2.5 !}} dt = - \tau \mathbf{E}(\tau x) \Big|_{\tau=0}^{\tau=1} = -\mathbf{E}(x),$$

where we used the fundamental theorem of calculus and that the components of $\mathbf{curl} \mathbf{E}(x)$ agree with the off-diagonal entries of $\mathbf{DE}(tx)^\top - \mathbf{DE}(tx)$.

Theorem 1.1.2.10. Existence of electrostatic potential
 If a continuous vectorfield $\mathbf{E} : \Omega \rightarrow \mathbb{R}^3$ satisfies (1.1.2.3) ("circulation-free"), then (1.1.2.8) defines a differentiable function $u : \Omega \rightarrow \mathbb{R}$ such that $\mathbf{E} = -\mathbf{grad} u$.

Obviously, if Ω is connected, then a function $u : \Omega \rightarrow \mathbb{R}$ satisfying $\mathbf{grad} u = -\mathbf{E}$ for given \mathbf{E} is **unique up to a constant**.

The function u from Thm. 1.1.2.10 is called a **scalar potential** for \mathbf{E} . The $-$ in its definition is a convention.

Assumption 1.1.2.11. Connected domains
 The domain Ω is connected

┘

Remark 1.1.2.12 (Scalar potentials and work) By virtue of the very definition (1.1.2.8) of the scalar potential we conclude that

$-q(u(\mathbf{x}) - u(\mathbf{y}))$ is the work required to move a charge q from $\mathbf{y} \in \Omega$ to $\mathbf{x} \in \Omega$ against the force exerted by the electric field $-\mathbf{grad} u : \Omega \rightarrow \mathbb{R}^3$.

Note: positive work is “work done by the electric field” (we harvest energy), negative work amounts to “work done against the electric field” (we spend energy). ┘

We are still missing two things:

1. A mathematical description of the cause of electromagnetic fields, which are charges,
2. and a criterion for selecting the unique physical electric field induced by charges.

These issues will be tackled next and everything will center around the concept of field energy introduced in Def. 1.1.1.3.

§1.1.2.13 (Spaces for electric fields and scalar potentials) Physically admissible electric fields $\mathbf{E} : \Omega \rightarrow \mathbb{R}^3$ on $\Omega \subset \mathbb{R}^3$ (either bounded or unbounded) have to satisfy

- ◆ that their energy content $\int_{\Omega} \epsilon(\mathbf{x}) \mathbf{E}(\mathbf{x}) \cdot \mathbf{E}(\mathbf{x}) \, d\mathbf{x}$ is finite, cf. Def. 1.1.1.3,
- ◆ and that they are gradients of a scalar potential: $\mathbf{E} = -\mathbf{grad} u$ for a sufficiently smooth function $u : \Omega \rightarrow \mathbb{R}$.

Thus we can switch to a characterization by admissible scalar potentials, which form the set

$$\left\{ u : \Omega \rightarrow \mathbb{R} : \int_{\Omega} \epsilon(\mathbf{x}) \mathbf{grad} u(\mathbf{x}) \cdot \mathbf{grad} u(\mathbf{x}) \, d\mathbf{x} < \infty \right\}.$$

Since $\epsilon : \Omega \rightarrow \mathbb{R}^{3,3}$ is uniformly positive definite, this set can be endowed with the structure of a **Hilbert space** by completion, see [NumPDE § 1.3.3.2] and [NumPDE § 1.3.3.10]. On a *bounded* domain Ω this yields the **Sobolev space** $H^1(\Omega)$, recall [NumPDE Section 1.3.4].

Definition 1.1.2.14. Sobolev space $H^1(\Omega)$, [NumPDE Def. 1.3.4.8]

For a bounded domain $\Omega \subset \mathbb{R}^d, d \in \mathbb{N}$, we define the **Sobolev space**

$$H^1(\Omega) := \left\{ v \in L^2(\Omega) : \int_{\Omega} |\mathbf{grad} v(\mathbf{x})|^2 \, d\mathbf{x} < \infty \right\}$$

as a Hilbert space with norm field energy

$$\|v\|_{H^1(\Omega)}^2 := \|v\|_{L^2(\Omega)}^2 + \underbrace{\int_{\Omega} |\mathbf{grad} v(\mathbf{x})|^2 \, d\mathbf{x}}_{\text{field energy}}.$$

The above definition involves the Hilbert space $L^2(\Omega)$:

Definition 1.1.2.15. Hilbert space of square integrable functions [NumPDE Def. 1.3.2.3]

The function **space of square integrable functions** on $\Omega \subset \mathbb{R}^d$ is

$$L^2(\Omega) := \left\{ v : \Omega \rightarrow \mathbb{R} \text{ integrable} : \int_{\Omega} |v(\mathbf{x})|^2 \, d\mathbf{x} < \infty \right\},$$

a Hilbert space, when endowed with the norm

$$\|v\|_{L^2(\Omega)} := \left(\int_{\Omega} |v(\mathbf{x})|^2 \, d\mathbf{x} \right)^{1/2}.$$

Space for admissible scalar potentials

Meaningful electrostatic scalar potentials on a bounded domain $\Omega \subset \mathbb{R}^3$ belong to the Sobolev space $H^1(\Omega)$.

Remark 1.1.2.17 (Potentials on unbounded domains) It will turn out that some physically meaningful scalar potentials will not belong to $L^2(\Omega)$, if $\Omega \subset \mathbb{R}^3$ is an **exterior domain**, that is, the open complement of a bounded Lipschitz domain. In this case the proper space of admissible potentials is [SS10, Eq. (2.148)]

$$H^1(\Omega) := \left\{ u : \Omega \rightarrow \mathbb{R} : \|u\|_{H^1(\Omega)}^2 := \int_{\Omega} \|\mathbf{grad} u(x)\|^2 + \frac{|u(x)|^2}{1 + \|x\|^2} dx < \infty \right\}, \quad (1.1.2.18)$$

which is larger than the space on Ω we would get from Def. 1.1.2.14. When equipped with the norm defined in (1.1.2.18) also this space becomes a Hilbert space. Note that (??) still guarantees finite energy of the electric field.

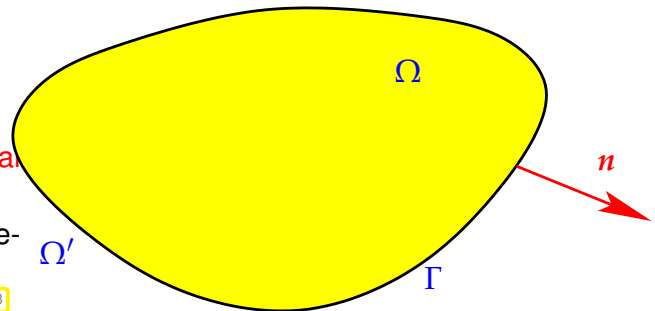


Many authors, also [SS10], use Def. 1.1.2.14 also for exterior domains and introduce special notation for the space defined in (1.1.2.18).

1.1.3 Continuity of Fields and Boundary Conditions

Maxwell's equations and their reduced version, the equations of electrostatics are generically posed on all of \mathbb{R}^3 . Often, one is interested in the behavior of the fields in a region $\Omega \neq \mathbb{R}^3$ only and the impact of the complement Ω' is taken into account by imposing boundary conditions on the boundary $\Gamma := \partial\Omega$.

The boundary $\Gamma := \partial\Omega$ is a two-dimensional orientable closed (that is, without a boundary itself) surface.



Notation: $n : \Gamma \rightarrow \mathbb{R}^3$ is the **exterior unit normal** vectorfield on Γ . (Defined only in the interior of faces for polyhedra)

Fig. 3

§1.1.3.1 (Jump conditions for electric field)

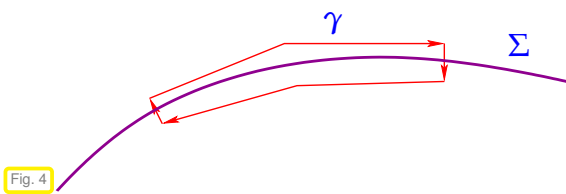


Fig. 4

$\Sigma \subset \mathbb{R}^3 \hat{=}$ smooth orientable surface ("interface"). Consider slender closed curve γ aligned with Σ , see Fig. 4. Letting the Σ -transversal width of γ shrink to zero, (1.1.2.3) $[\int_{\gamma} \mathbf{E} \cdot d\vec{s} = 0]$ can be satisfied for any such curve only if the tangential components of \mathbf{E} agree on both sides of Σ .

Continuity of electric fields

The **tangential** components of an electric field continuous on both sides of an orientable surface are **continuous** across that surface.

§1.1.3.3 (Continuity of scalar potentials) If a scalar potential $u : \Omega \rightarrow \mathbb{R}$ was only piecewise continuous with a jump across an interface, then pushing a charge by an "infinitesimally small" distance across the

interface could always release fixed finite amount of energy. This amounted to an infinitely large force acting on the charge, which does not make physical sense.

Continuity of scalar potentials

A scalar potential that is continuous on both sides of an orientable surface is also C^0 -continuous across it.

This finding very well matches our results about the appropriate function spaces for scalar potentials [NumPDE Thm. 1.3.4.23].

Theorem 1.1.3.5. Compatibility conditions for piecewise smooth functions in $H^1(\Omega)$

Let Ω be partitioned into sub-domains Ω_1 and Ω_2 . A function u that is continuously differentiable in both sub-domains and continuous up to their boundary, belongs to $H^1(\Omega)$, if and only if u is continuous on Ω .

We also recall from [NumPDE § 1.3.4.25] that continuous and piecewise continuously differentiable functions on Ω belong to $H^1(\Omega)$:


$$C_{pw}^1(\overline{\Omega}) \subset H^1(\Omega) . \quad (1.1.3.6)$$

We have to define C_{pw}^1 on the closed domain $\overline{\Omega}$ in (1.1.3.6) to make sure that the functions are continuous up to the boundary.

Be aware that the gradients of functions in $C_{pw}^1(\overline{\Omega})$ enjoy continuity of their tangential components across any interface inside Ω . They satisfy the natural jump conditions for electric fields, cf. § 1.1.3.1. \lrcorner

§1.1.3.7 (Normal and tangential components of a vectorfield on a surface) If $\mathbf{n}_\Sigma : \Sigma \rightarrow \mathbb{R}^3$ is a unit normal vector field on the orientable 2-surface Σ and \mathbf{v} a vectorfield continuous up to Σ , then

$$\begin{aligned} \text{the normal component of } \mathbf{v} \text{ in } \mathbf{x} \in \Sigma \text{ is} & \quad (\mathbb{T}_{\mathbf{n},\Sigma} \mathbf{v})(\mathbf{x}) := \mathbf{v}(\mathbf{x}) \cdot \mathbf{n}_\Sigma(\mathbf{x}), \\ \text{the tangential component of } \mathbf{v} \text{ in } \mathbf{x} \in \Sigma \text{ is} & \quad (\mathbb{T}_{t,\Sigma} \mathbf{v})(\mathbf{x}) := \mathbf{v}(\mathbf{x}) - (\mathbf{v}(\mathbf{x}) \cdot \mathbf{n}_\Sigma(\mathbf{x})) \mathbf{n}_\Sigma(\mathbf{x}), \end{aligned}$$

 Notations: $\mathbb{T}_{\mathbf{n},\Sigma} \hat{=}$ normal component (trace) of a vector field on Σ
 $\mathbb{T}_{t,\Sigma} \hat{=}$ tangential component (trace) of a vector field on Σ .
 (Subscript indicating the surface may be omitted when clear from the context.)

The mappings $\mathbb{T}_{\mathbf{n},\Sigma}$ and $\mathbb{T}_{t,\Sigma}$ are first examples of **trace operators**, linear mappings from function spaces on volume domains to function spaces on interfaces or boundaries. \lrcorner

§1.1.3.8 (Boundary conditions on the surface of conductors) A **conductor** is a region $\Omega_c \subset \mathbb{R}^3$ filled with (infinitely many) mobile charge carriers.

 The electric field vanishes inside a conductor.

Otherwise the field would cause permanent movement of charges, releasing an infinite amount of energy in the process.

If u is the electric potential (\rightarrow Thm. 1.1.2.10), then $\mathbf{E} = -\mathbf{grad} u = \mathbf{0}$ inside the conductor.

 The electric potential is constant inside each connected component of Ω_c .

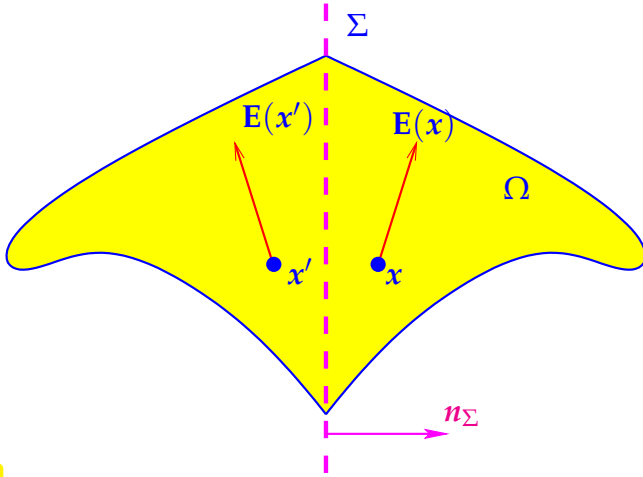
In light of the tangential continuity of the electric field \mathbf{E} , $\mathbf{E} = \mathbf{0}$ inside Ω_c means that

$$\mathbb{T}_{t,\partial\Omega_c} \mathbf{E} = \mathbf{0} \quad \text{on boundaries of conductors} . \quad (1.1.3.9)$$

Engineers refer to the boundary conditions (1.1.3.9) as **perfectly electrically conducting (PEC)**

In mathematics, these PEC boundary conditions belong to the class of **Dirichlet boundary conditions**, see [NumPDE Section 1.7].

§1.1.3.10 (Mirror symmetry boundary conditions)



Assume a situation mirror-symmetric with respect to a plane Σ (through $\mathbf{0}$) with unit normal \mathbf{n}_Σ , see Fig. 5:

$$\begin{aligned} \mathbf{x}' &= (\mathbf{I} - 2\mathbf{n}_\Sigma\mathbf{n}_\Sigma^\top)\mathbf{x}, \\ \mathbf{E}(\mathbf{x}') &= (\mathbf{I} - 2\mathbf{n}_\Sigma\mathbf{n}_\Sigma^\top)\mathbf{E}(\mathbf{x}). \end{aligned} \tag{1.1.3.11}$$

Note that the electric field \mathbf{E} is *completely continuous* across Σ , because Σ does not separate different physical domains (“artificial interface”).

Fig. 5

Thus, for $\mathbf{x} = \mathbf{x}' \in \Sigma$ we have

$$\mathbf{E}(\mathbf{x}) = (\mathbf{I} - 2\mathbf{n}_\Sigma\mathbf{n}_\Sigma^\top)\mathbf{E}(\mathbf{x}) \Rightarrow \mathbf{T}_{\mathbf{n},\Sigma}\mathbf{E} = \mathbf{E} \cdot \mathbf{n}_\Sigma = 0 \text{ on } \Sigma.$$

Boundary condition for electric fields as symmetry planes

At symmetry planes electric fields have vanishing normal components.
(= homogeneous Neumann boundary conditions [NumPDE Section 1.7])

§1.1.3.13 (Configuration space for electrostatic phenomena) Remember that the **configuration space** for a physical system is a subset of a vector space. Each element models a particular state of the system. In electrostatics states are characterized by functions on spatial domains, the fields.

In § 1.1.2.13 we saw that the configuration space can be a set of scalar potentials and should be a subspace of $H^1(\Omega)$. PEC boundary conditions as introduced in § 1.1.3.8 will enter the definition of the configuration space.

Configuration space for electrostatics

Let $\Gamma_1, \dots, \Gamma_m \subset \partial\Omega$ stand for the connected components of the part of $\partial\Omega$ corresponding to surfaces of conductors. Then the scalar potential is sought in the space

$$V := \{u \in H^1(\Omega): u|_{\Gamma_j} \equiv \text{const}, j = 1, \dots, m\}.$$

We can further restrict the configuration space, if the *scalar potential is imposed* on all or some connected components of the conducting part of $\partial\Omega$: Assume that $u|_{\Gamma_j} = U_j \in \mathbb{R}$ for $j = 1, \dots, k, k \leq m$. Then we can choose

$$V := \{u \in H^1(\Omega): u|_{\Gamma_j} = U_j, j = 1, \dots, k, u|_{\Gamma_j} \equiv \text{const}, j = k + 1, \dots, m\}. \tag{1.1.3.15}$$

This configuration space is an **affine space**. For any some $u_0 \in V$ it can be written as $V = u_0 + V_0$ with the Hilbert space

$$V_0 := \{u \in H^1(\Omega) : u|_{C_j} = 0, j = 1, \dots, k, u|_{C_j} \equiv \text{const}, j = k + 1, \dots, m\}. \tag{1.1.3.16}$$

Terminology: Connected components of the conducting boundary part of Ω where no potential is imposed are called **floating potentials**. ┘

Remark 1.1.3.17 (Fixing the potential) Since the potential is unique only up to constant, one can always set $u|_{\Gamma_1} = 0$ for one connected component of the conducting boundary without changing the outcome for the electric field. Then Γ_1 is called a grounded conductor. ┘

EXAMPLE 1.1.3.18 (Fixed potential boundary conditions)

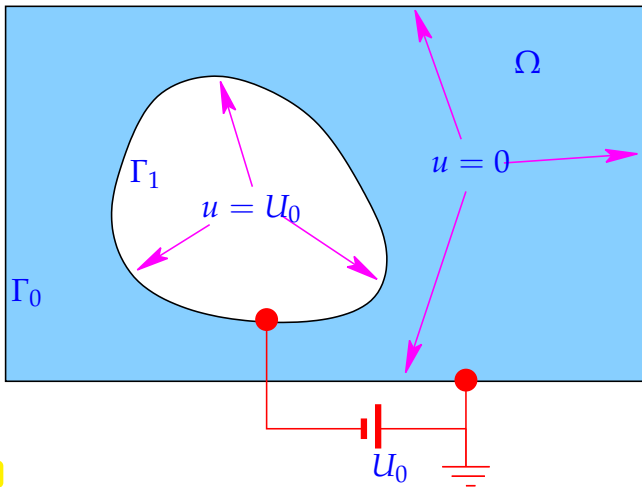


Fig. 6

◁ situation with imposed potentials

$$\begin{aligned} u &= 0 && \text{on } \Gamma_0, \\ u &= U_0 && \text{on } \Gamma_1. \end{aligned} \tag{1.1.3.19}$$

(Metal electrode inside a grounded metal box)

▶ configuration space

$$V = \left\{ u \in H^1(\Omega) : u \text{ satisfies (1.1.3.19)} \right\}.$$

1.1.4 Equilibrium Conditions

As the reader will know, the sources of electric fields are electric charges. Above we have already made of the construct of a point charge for measuring an electric field through the Coulomb force.

A large number of small “point charges” contained in a volume $\Omega \subset \mathbb{R}^3$ can be modeled by a **charge density** $\rho : \Omega \rightarrow \mathbb{R}$, physical units $[\rho] = 1 \frac{\text{As}}{\text{m}^3}$.

$$\blacktriangleright Q = \int_D \rho(x) dx \hat{=} \text{total charge in sub-volume } D \subset \Omega. \tag{1.1.4.1}$$

§1.1.4.2 (Energy of charges in a field) Assume $\Omega \subset \mathbb{R}^3$ to be bounded that the the scalar potential $u : \Omega \rightarrow \mathbb{R}$ satisfies $u|_{\partial\Omega} = 0$ (If Ω is the complement of a bounded set, we may just choose a normalization of the scalar potential that makes it vanish at large distance: $\lim_{\|x\| \rightarrow \infty} u(x) = 0$ uniformly, also written as “ $u(\infty) = 0$ ”). According to Rem. 1.1.2.12 it takes the work $-qu(x)$ to move a charge q to $x \in \Omega$ from $\partial\Omega$.

Now think of a charge density $\rho : \Omega \rightarrow \mathbb{R}_0^+$ as composed of many small point charges. The work it takes to assemble this arrangement of charges is the sum of the work units required for each individual charge, because we assume a fixed scalar potential not influenced by the presence of the charges. In the limit this summation becomes integration (\rightarrow Riemann integral), and the energy required for setting up the charge distribution ρ in the presence of a **fixed** potential is

$$J_\rho(u) = - \int_\Omega \rho(x) u(x) dx, \quad u \in H^1(\Omega). \tag{1.1.4.3}$$

The notation stresses the dependence of the energy on u , because this will play the role of the unknown. In the sequel the presence of charges modeled by $\rho(x)$ will engender the fields. Therefore we call ρ as **source charge distribution**. ┘

Remark 1.1.4.4 (Admissible source charge distributions) The energy $J_\rho(u)$ of a source charge distribution $\rho : \Omega \rightarrow \mathbb{R}$ should be finite for all admissible scalar potentials u . For bounded Ω , applying the Cauchy-Schwarz inequality in $L^2(\Omega)$ [NumPDE Eq. (1.3.4.15)] we get

$$|J_\rho(u)| = \left| \int_\Omega \rho(x) u(x) \, dx \right| \leq \left(\int_\Omega \rho(x)^2 \, dx \right)^{1/2} \left(\int_\Omega u(x)^2 \, dx \right)^{1/2} = \|\rho\|_{L^2(\Omega)} \|u\|_{L^2(\Omega)} .$$

Hence, $\rho \in L^2(\Omega)$ is a sufficient condition for a suitable source charge distribution, cf. [NumPDE Cor. 1.3.4.19].

If Ω is an exterior domain, we demand that ρ has bounded support in addition (“compactly supported”). ┘

The **total energy** in a electrostatic situation in a volume Ω is the sum of the energy (1.1.1.4) of the electric field and the energy content of the charges given by (1.1.4.3):

$$J(u) := J_{el}(u) + J_\rho(u) = \int_\Omega \frac{1}{2} \epsilon(x) \mathbf{grad} u(x) \cdot \mathbf{grad} u(x) - \rho(x) u(x) \, dx, \quad u \in V \subset H^1(\Omega) . \tag{1.1.4.5}$$

configuration space, see § 1.1.3.13

The selection of the scalar potential prevailing in a particular situation relies on a fundamental equilibrium principle also called virtual work principle, compare [NumPDE Eq. (1.2.2.16)]:

Equilibrium condition for electrostatic phenomena

Given a (compactly supported) source charge distribution $\rho \in L^2(\Omega)$ and a configuration space $V \subset H^1(\Omega)$ encoding boundary conditions, the scalar electrostatic potential u minimizes to total energy

$$u = \underset{v \in V}{\operatorname{argmin}} J(v) . \tag{1.1.4.7}$$

§1.1.4.8 (Total energy as quadratic functional) Let us introduce the following abbreviations:

$$a(u, v) := \int_\Omega \epsilon(x) \mathbf{grad} u(x) \cdot \mathbf{grad} v(x) \, dx, \quad u, v \in H^1(\Omega) , \tag{1.1.4.9}$$

$$\ell(v) := \int_\Omega \rho(x) v(x) \, dx, \quad v \in L^2(\Omega) . \tag{1.1.4.10}$$

Here, $a : H^1(\Omega) \times H^1(\Omega) \rightarrow \mathbb{R}$ is a **bilinear form** and $\ell : H^1(\Omega) \rightarrow \mathbb{R}$ is a **linear form**, see [NumPDE Def. 0.3.1.4]. Then the functional J from (1.1.4.5) can be written as

$$J(u) = \frac{1}{2} a(u, u) - \ell(u) . \tag{1.1.4.11}$$

Hence, J is a **quadratic functional**, see [NumPDE Def. 1.2.3.2] on $V \subset H^1(\Omega)$ and the scalar potential is defined as the solution of the **quadratic minimization problem** (1.1.4.7).

We remark the obvious fact that the bilinear form $a(\cdot, \cdot)$ from (1.1.4.9) is positive semi-definite [NumPDE Def. 1.2.3.27]. This connects to the fact that $\frac{1}{2} a(u, u)$ tells the energy (norm) (1.1.1.4) of the electric field

$$\mathbf{E} := -\operatorname{grad} u.$$

By the Cauchy-Schwarz inequality both \mathbf{a} and ℓ are **continuous** on $H^1(\Omega)$ in the sense of [NumPDE Def. 1.2.3.42]. \lrcorner

The next result answers the fundamental question about existence and uniqueness of solutions of the above quadratic minimization problem. Throughout, $V \subset H^1(\Omega)$ is the configuration space as described above.

Theorem 1.1.4.12. Existence and uniqueness of energy minimizing potentials

If

$\Omega \subset \mathbb{R}^3$ is bounded and u is fixed on some part of $\partial\Omega$

or

$\Omega \subset \mathbb{R}^3$ is the complement of a bounded domain

then (1.1.4.7) has a unique solution.

The proof of this theorem requires deep results from the theory of Sobolev spaces (a generalization of the first Poincaré-Friedrichs inequality [NumPDE Thm. 1.3.4.17]) and functional analysis (Riesz representation theorem [NumPDE Thm. 1.3.3.6]). If Ω is the complement of a bounded domain, then we have to appeal to [SS10, Prop. 2.10.8].

1.1.5 Variational Equations

Recall the notion of a linear variational problem from [NumPDE Def. 1.4.1.6]:

Definition 1.1.5.1. Linear variational problem

A variational problem posed on an affine space V and a vector space V_0 of the form

$$u \in V: \quad \mathbf{a}(u, v) = \ell(v) \quad \forall v \in V_0, \quad (1.1.5.2)$$

is called a **linear variational problem**, if

- $\mathbf{a} : V \times V_0 \mapsto \mathbb{R}$ is a **bilinear form** (BLF), that is, linear in both arguments (\rightarrow [NumPDE Def. 0.3.1.4]),
- and $\ell : V_0 \rightarrow \mathbb{R}$ is a linear form (LF).

We will also need fundamental abstract result from [NumPDE Section 1.4.1].

Theorem 1.1.5.3. Equivalence theorem for quadratic minimization problems

Let V_0 be a normed real vector space, V a related affine space and $\mathbf{a} : V \times V \rightarrow \mathbb{R}$, $\ell : V \rightarrow \mathbb{R}$ a continuous **symmetric positive semi-definite** (\rightarrow [NumPDE Def. 1.2.3.24]) bilinear form and continuous linear form, respectively. Then $u \in V$ is a minimizer of the quadratic functional $J(v) := \frac{1}{2}\mathbf{a}(v, v) - \ell(v)$, **if and only if** u solves the linear variational problem

$$u \in V: \quad \mathbf{a}(u, v) = \ell(v) \quad \forall v \in V_0.$$

The assertion of this theorem can concisely be stated as follows: for $u \in V$ holds the equivalence

$$u = \operatorname{argmin}_{v \in V} \frac{1}{2}\mathbf{a}(v, v) - \ell(v) \quad \iff \quad u \in V: \quad \mathbf{a}(u, v) = \ell(v) \quad \forall v \in V_0, \quad (1.1.5.4)$$

if \mathbf{a} is symmetric and positive semi-definite.

Proof of Thm. 1.1.5.3. (I) Assume that $u \in V$ is a minimizer of $J(v)$ over the affine space $V = u + V_0$. Then for any $v \in V_0$ the smooth auxiliary function

$$\varphi_v : \mathbb{R} \rightarrow \mathbb{R} \quad , \quad \varphi_v(t) := J(u + tv) \quad ,$$

has a global minimum in $t = 0$, which means

$$\frac{d\varphi_v}{dt}(0) = \{t \mapsto ta(v, v) + a(u, v) - \ell(v)\}_{|t=0} = a(u, v) - \ell(v) = 0 \quad .$$

Since $v \in V_0$ was arbitrary, (1.1.5.2) follows.

(II) Let $u \in V$ satisfy (1.1.5.2): $a(u, v) = \ell(v)$ for all $v \in V_0$. Then we can rewrite

$$J(v) = \frac{1}{2}a(v, v) - a(u, v) = \frac{1}{2} \underbrace{a(v - u, v - u)}_{\geq 0!} - \frac{1}{2}a(u, u) \quad .$$

Obviously, $v = u$ yields a global minimizer. □

Concretely, if the potential u is fixed to agree with a function $g : \Gamma_* \rightarrow \mathbb{R}$ on a part Γ_D (“Dirichlet part”) of the boundary $\partial\Omega$, it can be obtained as the solution of the following linear variational problem.

$$\begin{aligned} u \in H^1(\Omega), \\ u|_{\Gamma_D} = g \end{aligned} : \quad \int_{\Omega} \epsilon(x) \mathbf{grad} u(x) \cdot \mathbf{grad} v(x) \, dx = \int_{\Omega} \rho(x) v(x) \, dx \quad \forall \begin{aligned} v \in H^1(\Omega), \\ v|_{\Gamma_D} = 0 \end{aligned} \quad . \quad (1.1.5.5)$$

1.1.6 Boundary Value Problems

In [NumPDE Section 1.5] we learned that, under some assumptions on the smoothness of solutions, linear variational problems like (1.1.5.5) can be recast as **boundary value problems** for second-order linear partial differential equations in **strong form**. The main tool is Green’s first formula that we recall from [NumPDE Thm. 1.5.2.7].

Theorem 1.1.6.1. Green’s first formula

For all vector fields $\mathbf{j} \in (C^1_{pw}(\bar{\Omega}))^d$ and functions $v \in C^1_{pw}(\bar{\Omega})$ holds

$$\int_{\Omega} \mathbf{j} \cdot \mathbf{grad} v \, dx = - \int_{\Omega} \text{div} \mathbf{j} v \, dx + \int_{\partial\Omega} \mathbf{j} \cdot \mathbf{n} v \, dS \quad . \quad (1.1.6.2)$$

The **divergence** of a vector field $\mathbf{v}(x) = [v_1(x), \dots, v_d(x)]$ is

$$\text{div} \mathbf{v}(x) = \frac{\partial v_1}{\partial x_1} + \dots + \frac{\partial v_d}{\partial x_d} \quad .$$

As in [NumPDE Ex. 1.5.3.11] the derivation of the boundary value problem induced by (1.1.5.5) proceeds in two steps. Throughout we assume that $u \in C^2(\bar{\Omega})$ so that all manipulations are possible. The source charge distribution must have compact support in \mathbb{R}^3 .

❶ In (1.1.5.5) test with $v \in C_0^\infty(\Omega) \hat{=}$ smooth functions with compact support, vanishing on $\partial\Omega$

$$\int_{\Omega} \epsilon(x) \mathbf{grad} u(x) \cdot \mathbf{grad} v(x) \, dx = \int_{\Omega} \rho(x) v(x) \, dx \quad \forall v \in C_0^\infty(\Omega)$$

$$\begin{aligned}
& \Downarrow \leftarrow \text{by Thm. 1.1.6.1, } v|_{\partial\Omega} = 0! \\
& - \int_{\Omega} \operatorname{div}(\epsilon(x) \mathbf{grad} u(x)) v(x) \, dx = \int_{\Omega} \rho(x) v(x) \, dx \quad \forall v \in C_0^\infty(\Omega) \\
& \Downarrow \leftarrow \text{density of smooth functions in } L^2(\Omega) \\
& \boxed{- \operatorname{div}(\epsilon(x) \mathbf{grad} u(x)) = \rho \quad \text{in } \Omega} . \tag{1.1.6.3}
\end{aligned}$$

② In (1.1.5.5) test with $v \in C^\infty(\overline{\Omega})$ with bounded support in \mathbb{R}^3 , vanishing on Γ_D

$$\begin{aligned}
& \int_{\Omega} \epsilon(x) \mathbf{grad} u(x) \cdot \mathbf{grad} v(x) \, dx = \int_{\Omega} \rho(x) v(x) \, dx \quad \forall v \in C^\infty(\overline{\Omega}) \\
& \Downarrow \leftarrow \text{by Thm. 1.1.6.1} \\
& \int_{\Omega} \underbrace{- \operatorname{div}(\epsilon(x) \mathbf{grad} u(x))}_{=\rho(x) \text{ by (1.1.6.3)}} v(x) \, dx + \int_{\partial\Omega} \epsilon(x) \mathbf{grad} u \cdot \mathbf{n} v(x) \, dS(x) \\
& \qquad \qquad \qquad = \int_{\Omega} \rho(x) v(x) \, dx \quad \forall v \in C^\infty(\overline{\Omega}) \\
& \Downarrow \leftarrow \text{use (1.1.6.3)} \\
& \int_{\partial\Omega} \epsilon(x) \mathbf{grad} u \cdot \mathbf{n} v(x) \, dS(x) = 0 \quad \forall v \in C^\infty(\overline{\Omega}), v|_{\Gamma_D} = 0 \\
& \Downarrow \leftarrow \text{density of smooth functions} \\
& \boxed{\epsilon(x) \mathbf{grad} u \cdot \mathbf{n} = 0 \quad \text{on } \Gamma_N := \partial\Omega \setminus \Gamma_D} . \tag{1.1.6.4}
\end{aligned}$$

In fact, the boundary conditions on Γ_N agree with the symmetry boundary conditions derived in § 1.1.3.10.

Summing up, the strong form of the boundary value problem related to (1.1.5.5) is

$$- \operatorname{div}(\epsilon(x) \mathbf{grad} u(x)) = \rho \quad \text{in } \Omega , \tag{1.1.6.5a}$$

$$u = g \quad \text{on } \Gamma_D \tag{1.1.6.5b}$$

$$\epsilon(x) \mathbf{grad} u \cdot \mathbf{n} = 0 \quad \text{on } \Gamma_N . \tag{1.1.6.5c}$$

The boundary conditions (1.1.6.5b), which generalize the PEC boundary conditions from § 1.1.3.8, are **Dirichlet boundary conditions**, whereas (1.1.6.5c) is called (homogeneous) **Neumann boundary conditions** [NumPDE Section 1.9].

Remark 1.1.6.6 (Gauss' law) The partial differential equation $- \operatorname{div}(\epsilon(x) \mathbf{grad} u(x)) = \rho$ is known as **Gauss' law**. It holds beyond the stationary setting in electrodynamics (assuming a “suitable” definition of charge).

The field $\mathbf{D}(x) := -\epsilon(x) \mathbf{grad} u(x)$ is known as **displacement current** in electrodynamics (physical units $[\mathbf{D}] = 1 \frac{\text{As}}{\text{m}^2}$).

As a consequence of Gauss' law and **Gauss' theorem**, which is Green's first formula (1.1.6.2) with $v \equiv 1$, we get

$$\int_{\partial D} \epsilon(x) \mathbf{grad} u(x) \cdot \mathbf{n}(x) \, dS(x) = - \int_D \rho(x) \, dx \tag{1.1.6.7}$$

for all “control volumes” $D \subset \Omega$. ┘

Remark 1.1.6.8 (Electrostatics in homogeneous isotropic media) Homogeneous isotropic media feature a dielectric tensor that is a constant multiple of the identity matrix $\epsilon(\mathbf{x}) = \epsilon \mathbf{I}$ for some constant $\epsilon > 0$. In this case by scaling (\rightarrow Rem. 1.1.1.7) we can always obtain the non-dimensional **Poisson equation** from (1.1.6.3):

$$-\Delta u = \rho, \tag{1.1.6.9}$$

with the **Laplace operator**

$$\Delta = \text{div} \circ \text{grad} = \frac{\partial^2}{\partial x_1^2} + \frac{\partial^2}{\partial x_2^2} + \frac{\partial^2}{\partial x_3^2}.$$

┘

§1.1.6.10 (Transmission conditions) From Gauss law we conclude that

$$\text{div } \mathbf{D} = \text{div}(-\epsilon \text{grad } u) \in L^2(\Omega).$$

Let Ω be partitioned $\Omega = \Omega^l \cup \Sigma \cup \Omega^r$ with piecewise smooth interface Σ , see figure (\triangleleft) for cross-section.

Assume that both ϵ and u are smooth both in $\overline{\Omega^l}$ and $\overline{\Omega^r}$. Apply Gauss' theorem (1.1.6.7) in a small flat cylindrical box with "bottom" and "top" face locally aligned with Σ .

Let the height and width of the box tend to zero so that it shrinks to a point $\mathbf{x} \in \Sigma$. There we find

$$(\epsilon \text{grad } u|_{\Omega^r}(\mathbf{x}) - \epsilon \text{grad } u|_{\Omega^l}(\mathbf{x})) \cdot \mathbf{n}^-(\mathbf{x}) = \sigma(\mathbf{x}), \tag{1.1.6.11}$$

where $\sigma : \Sigma \rightarrow \mathbb{R}$ is a **surface charge**, that is, a layer of charge concentrated on Σ (which does not exist in $L^2(\Omega)$, however).

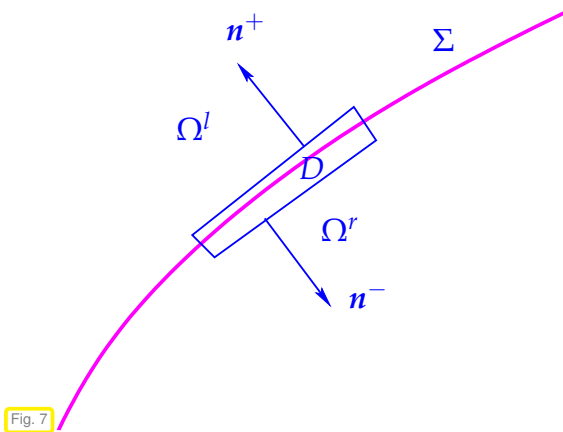


Fig. 7

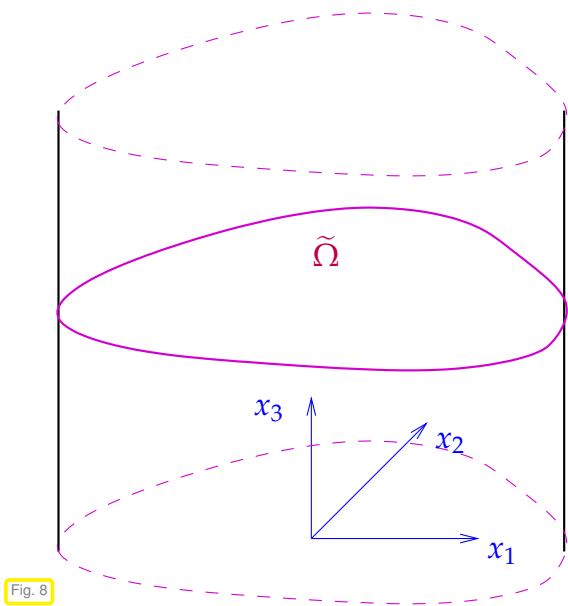
┘

Continuity of displacement current

If $\rho \in L^2(\Omega)$ and u solves (1.1.6.5a) and is piecewise smooth, then the **normal component** of $\mathbf{D} := -\epsilon \text{grad } u$ is **continuous** across any interface.

Note that surface charges cannot belong to $L^2(\Omega)$, because functions in $L^2(\Omega)$ cannot be restricted to some surface, cf. [NumPDE Ex. 1.3.2.5].

§1.1.6.13 (Electrostatics in two dimensions)



We say that a situation possesses **translational symmetry**, when

- ◆ there is a Cartesian coordinate system with coordinates (x_1, x_2, x_3) such that no quantity depends on the x_3 -coordinate,
- ◆ and it is posed on a **cylindrical** spatial domain of the tensor product form $\Omega = \tilde{\Omega} \times \mathbb{R}, \tilde{\Omega} \subset \mathbb{R}^2$

Then, (1.1.6.5) becomes a boundary value problem for $\tilde{u}(x_1, x_2) = u(x_1, x_2, 0)$ on $\tilde{\Omega}$:

$$-\text{div}(\tilde{\epsilon} \mathbf{grad} \tilde{u}) = \rho \text{ in } \tilde{\Omega}, \quad \tilde{u} = \tilde{g} \text{ on } \tilde{\Gamma}_D, \quad \tilde{\epsilon} \mathbf{grad} \tilde{u} \cdot \tilde{\mathbf{n}} = 0 \text{ on } \tilde{\Gamma}_N, \tag{1.1.6.14}$$

where, for instance, $\tilde{\epsilon}(x_1, x_2) = (\epsilon(x_1, x_2, 0))_{1:2,1:2}$, $\mathbf{grad} \tilde{u} = [\frac{\partial \tilde{u}}{\partial x_1}, \frac{\partial \tilde{u}}{\partial x_2}]^\top$.

Thus, we naturally arrive at a scalar elliptic boundary value problem in two dimensions. ┘

1.1.7 Decay conditions on unbounded domains

We are concerned with the electrostatic linear variational problem

$$u \in H^1(\Omega), \quad u|_{\Gamma_D} = g : \int_{\Omega} \epsilon(x) \mathbf{grad} u(x) \cdot \mathbf{grad} v(x) dx = \int_{\Omega} \rho(x) v(x) dx \quad \forall v \in H^1(\Omega), \quad v|_{\Gamma_D} = 0. \tag{1.1.5.5}$$

posed on the complement Ω of a bounded domain. We face a so-called **exterior BVP**. We also assume that $\rho(x) = 0$ and $\epsilon(x) = \mathbf{I}$ for $\|x\| \geq R$ and some $R \gg 1$.

Far away from $\partial\Omega$ and $\text{supp} \rho$ we expect the electric field to be “radial”:

$$\mathbf{E}(x) = \mathcal{E}(\|x\|)x/\|x\| \quad \text{with} \quad \mathcal{E} : \mathbb{R}^+ \rightarrow \mathbb{R} \quad \text{for} \quad \|x\| \rightarrow \infty.$$

Notation: $B_r(x) \hat{=}$ ball with center x and radius $r > 0$.

By Gauss’ law and theorem

$$4\pi r^2 \mathcal{E}(r) = \int_{\partial B_r(0)} \mathbf{E}(x) \cdot x/\|x\| dS(x) = \int_{B_r(0)} \rho(x) dx = \text{const} \quad \text{for} \quad r \rightarrow \infty,$$

▶ $\mathcal{E}(r) = O(r^{-2}) \quad \text{for} \quad r \rightarrow \infty.$

For large $\|x\|$ we also expect $u(x) = \mu(\|x\|)$, which means $\mathbf{grad} u(x) = \mu'(\|x\|)x/\|x\|$. Thus, from the asymptotic behavior of \mathbf{E} we conclude

$$|\mu(r)| \approx \left| \int_0^r \mathcal{E}(s) ds \right| \leq O(r^{-1}) \quad \text{for} \quad r \rightarrow \infty,$$

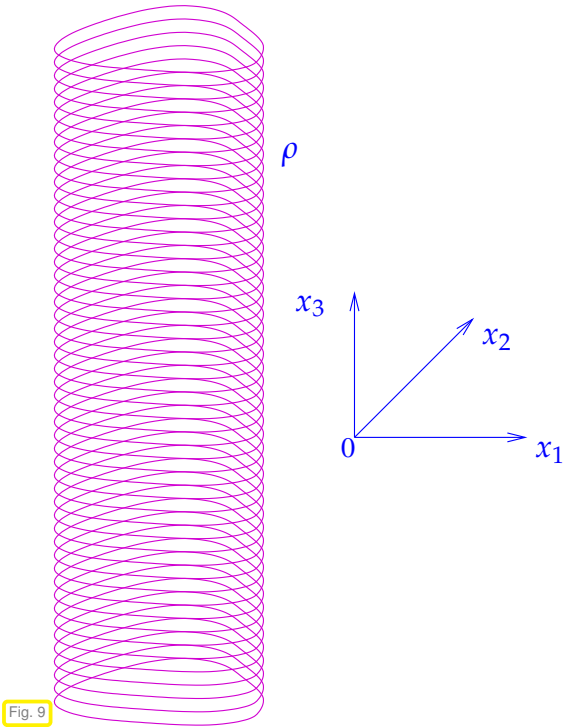
▶
 $|u(x)| = O(\|x\|^{-1}) \quad \text{and} \quad \|\mathbf{grad} u(x)\| = O(\|x\|^{-2}) \quad \text{for} \quad \|x\| \rightarrow \infty$
 . (1.1.7.1)

These decay conditions have to be imposed as “boundary conditions at ∞ ” for the exterior boundary value problems of 3D electrostatics.

Note that a smooth potential decaying according to (1.1.7.1) belongs to $H^1(\Omega)$ as defined in (1.1.2.18). To see this transform the integrals to polar coordinates (\rightarrow [NumPDE ??]).

Remark 1.1.7.2 (Necessity of decay conditions) Considering $\Omega = \mathbb{R}^3$ it is clear that without imposing decay conditions we cannot expect a unique solution of $-\Delta u = \rho$, because we could always add an unbounded harmonic function like $x \mapsto x_1^2 - x_2^2$ to u and would get a different solution. \lrcorner

§1.1.7.3 (Decay conditions in 2D electrostatics)



We consider an x_3 -translation-invariant setting in whole space \mathbb{R}^3 as in § 1.1.6.13 with a cylindrical source charge distribution $\rho(x) = \tilde{\rho}(x_1, x_2)$, $\tilde{\rho}$ compactly supported in the $x_1 - x_2$ plane and infinitely extended in x_3 -direction. \triangleright

Thought experiment: To compute the electric field in $x := [x_1, x_2, 0]^T$ we chop up ρ into many slices and obtain $\mathbf{E}(x)$ by **linear superposition** of the fields generated by the individual “charge slices”.

Then send $x_1^2 + x_2^2 \rightarrow \infty$ and take into account the decay condition (1.1.7.1) for the electric field: the field \mathbf{E}_ξ caused by the “charge slice” at $x_3 = \xi \in \mathbb{R}$ will behave like

$$\|\mathbf{E}_\xi(x_1, x_2, 0)\| = O((x_1^2 + x_2^2 + \xi^2)^{-1})$$

for $x_1, x_2, \xi \rightarrow \infty$ separately .

Now, letting the thickness of the slices tend to zero, summation can be replaced with integration (“Riemann summation”, see [NumPDE ??]). Writing $\tilde{x} := [x_1, x_2]^T$ we get with some constant $C > 0$

$$\|\mathbf{E}(x_1, x_2, 0)\| \leq C \int_{-\infty}^{\infty} \frac{1}{\|\tilde{x}\|^2 + \xi^2} d\xi = \frac{C}{\|\tilde{x}\|^2} \int_{-\infty}^{\infty} \frac{1}{1 + (\xi/\|\tilde{x}\|)^2} d\xi = \frac{C}{\|\tilde{x}\|} \int_{-\infty}^{\infty} \frac{1}{1 + \zeta^2} d\zeta = \frac{C\pi}{2\|\tilde{x}\|} .$$

For $\|\tilde{x}\| \rightarrow \infty$ we can again expect a merely radial dependence of the electric field and scalar potential

$$\mathbf{E}(x_1, x_2, 0) = \tilde{\mathcal{E}}(\|\tilde{x}\|) \frac{1}{\|\tilde{x}\|} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} , \quad u(x_1, x_2, 0) = \mu(\|\tilde{x}\|) .$$

By integrating the electric field in radial direction

$$|\mu(r)| \leq C \int_1^r \frac{1}{s} ds = O(\log r) \quad \text{for } r \rightarrow \infty ,$$

► $|\tilde{u}(\tilde{x})| = O(\log\|\tilde{x}\|) \quad , \quad \|\tilde{\mathbf{E}}(\tilde{x})\| = O(\|\tilde{x}\|^{-1}) \quad \text{for } \|\tilde{x}\| \rightarrow \infty$. (1.1.7.4)

1.1.8 Supplement: An energy norm for source charge distributions

In Rem. 1.1.4.4 we have seen that $\rho \in L^2(\Omega)$ makes $\rho : \Omega \rightarrow \mathbb{R}$ a valid source charge distribution on the domain $\Omega \subset \mathbb{R}^3$. Is $L^2(\Omega)$ the largest space of possible source charge distributions? In this section we will identify an even larger space of admissible source charge distributions by introducing a suitable norm on them.

Definition 1.1.8.1. Dual norm for source charge distributions

For $\rho \in L^2(\Omega)$ let $\tilde{\rho} \in L^2(\mathbb{R}^3)$ be its extension by zero to \mathbb{R}^3 and define

$$\|\rho\|_{\tilde{H}^{-1}(\Omega)} := \|u\|_{H^1(\mathbb{R}^3)} \quad \text{where } u \text{ solves } \begin{cases} -\Delta u = \tilde{\rho} & \text{in } \mathbb{R}^3, \\ u \text{ satisfies decay conditions} & (1.1.7.1). \end{cases} \quad (1.1.8.2)$$

The completion of $L^2(\Omega)$ w.r.t. $\|\cdot\|_{\tilde{H}^{-1}(\Omega)}$ yields the Hilbert space $\tilde{H}^{-1}(\Omega)$.

► The norm $\|\rho\|_{\tilde{H}^{-1}(\Omega)}$ can be read as the energy of the electric field on \mathbb{R}^3 engendered by the source charge distribution ρ (after extension by zero).

The solution u of the “exterior” boundary value problem in (1.1.8.2) can be obtained as the solution of the linear variational problem

$$u \in H^1(\mathbb{R}^3): \quad \int_{\mathbb{R}^3} \mathbf{grad} u(x) \cdot \mathbf{grad} v(x) \, dx = \int_{\mathbb{R}^3} \tilde{\rho}(x) v(x) \, dx \quad \forall v \in H^1(\mathbb{R}^3). \quad (1.1.8.3)$$

Remember that $H^1(\mathbb{R}^3)$ is defined through a weighted L^2 -norm in (1.1.2.18).

§1.1.8.4 (An embedding of $L^2(\Omega)$) From [SS10, Prop. 2.10.8] we learn that

$$\int_{\mathbb{R}^3} \frac{|v(x)|^2}{1 + \|x\|^2} \, dx \leq 4 \int_{\mathbb{R}^3} \|\mathbf{grad} v(x)\|^2 \, dx \quad \forall v \in H^1(\mathbb{R}^3). \quad (1.1.8.5)$$

Thus, setting $v = u$ in (1.1.8.3), we obtain for $\tilde{\rho} \in L^2(\Omega)$ that

$$\|\rho\|_{\tilde{H}^{-1}(\Omega)}^2 \leq \|\mathbf{grad} u\|_{L^2(\mathbb{R}^3)}^2 \leq \|\tilde{\rho}\|_{L^2(\mathbb{R}^3)} \|u\|_{L^2(\mathbb{R}^3)} \leq 4\sqrt{1 + R^2} \|\rho\|_{L^2(\Omega)} \|\mathbf{grad} u\|_{L^2(\mathbb{R}^3)},$$

► $\|\rho\|_{\tilde{H}^{-1}(\Omega)} \leq 4 \|\rho\|_{L^2(\Omega)}.$

Thus $\|\cdot\|_{\tilde{H}^{-1}(\Omega)}$ is a weaker norm than $\|\cdot\|_{L^2(\Omega)}$ and $\tilde{H}^{-1}(\Omega)$ is a **larger space** than $L^2(\Omega)$. ◻

§1.1.8.6 (Duality of $H^1(\Omega)$ and $\tilde{H}^{-1}(\Omega)$) Owing to (1.1.8.3) we can also characterize the space $\tilde{H}^{-1}(\Omega)$ and the norm $\|\cdot\|_{\tilde{H}^{-1}(\Omega)}$ in an equivalent way

$$\tilde{H}^{-1}(\Omega) := \left\{ \rho : \Omega \rightarrow \mathbb{R} : \left| \int_{\Omega} \rho(x) u(x) \, dx \right| < \infty \text{ for all finite-energy potentials } u \right\}, \quad (1.1.8.7)$$

$$\|\rho\|_{\tilde{H}^{-1}(\Omega)} = \sup_{u \in H^1(\Omega)} \frac{\int_{\Omega} \rho(x) u(x) \, dx}{\|u\|_{H^1(\Omega)}}, \quad \rho \in \tilde{H}^{-1}(\Omega). \quad (1.1.8.8)$$

duality means that the same characterization applies to $H^1(\Omega)$ and $\|\cdot\|_{H^1(\Omega)}$ in a reciprocal fashion:

$$H^1(\Omega) := \left\{ u \in L^2(\Omega) : \left| \int_{\Omega} \rho(x) u(x) \, dx \right| < \infty \quad \forall \rho \in \tilde{H}^{-1}(\Omega) \right\}, \quad (1.1.8.9)$$

$$\|u\|_{H^1(\Omega)} = \sup_{\rho \in \tilde{H}^{-1}(\Omega)} \frac{\int_{\Omega} \rho(x) u(x) dx}{\|\rho\|_{\tilde{H}^{-1}(\Omega)}}, \quad u \in H^1(\Omega). \quad (1.1.8.10)$$

┘

1.2 Boundary Representation Formulas

The focus will be on electrostatic problems in homogeneous isotropic dielectric media so that, after rescaling, we face boundary value problems (BVPs) for the Laplacian $-\Delta$ in 2D (\rightarrow § 1.1.6.13) and 3D, as explained in Rem. 1.1.6.8.

Most considerations apply to more general linear scalar second-order partial differential operators in **divergence form** in $d \in \mathbb{N}$ dimensions and with constant coefficients

$$Lu := -\operatorname{div}(\mathbf{A} \operatorname{grad} u) + cu, \quad \begin{array}{l} \mathbf{A} \in \mathbb{R}^{d,d} \text{ symmetric positive definite (s.p.d.)}, \\ c \in \mathbb{R}. \end{array} \quad (1.2.0.1)$$

1.2.1 Green's Formulas

Recall Green's first formula on $\Omega \subset \mathbb{R}^d$ from Thm. 1.1.6.1: for a vector field $\mathbf{j} \in (C_{\text{pw}}^1(\overline{\Omega}))^d$ and a function $v \in C_{\text{pw}}^1(\overline{\Omega})$,

$$\int_{\Omega} \mathbf{j} \cdot \operatorname{grad} v dx = - \int_{\Omega} \operatorname{div} \mathbf{j} v dx + \int_{\partial\Omega} \mathbf{j} \cdot \mathbf{n} v dS. \quad (1.1.6.2)$$

We may set $\mathbf{j} := \operatorname{grad} u$ for $u \in C^2(\overline{\Omega})$ and obtain from $\Delta = \operatorname{div} \operatorname{grad}$

$$\int_{\Omega} \operatorname{grad} u \cdot \operatorname{grad} v dx = - \int_{\Omega} \Delta u v dx + \int_{\partial\Omega} \operatorname{grad} u \cdot \mathbf{n} v dS. \quad (1.2.1.1)$$

Applying Green's first formula to the first integral in (1.2.1.1) yields Green's second formula [SS10, Thm. 2.7.4]

Theorem 1.2.1.2. Green's second formula

For $u, v \in C^2(\overline{\Omega})$ holds

$$\int_{\Omega} u \Delta v - v \Delta u dx = \int_{\partial\Omega} u \operatorname{grad} v \cdot \mathbf{n} - v \operatorname{grad} u \cdot \mathbf{n} dS(x). \quad (1.2.1.3)$$

These formulas are valid on any bounded Lipschitz domain $\Omega \subset \mathbb{R}^d$. If Ω is an exterior domain, a sufficiently fast decay of all functions for $\|x\| \rightarrow \infty$ has to be assumed.

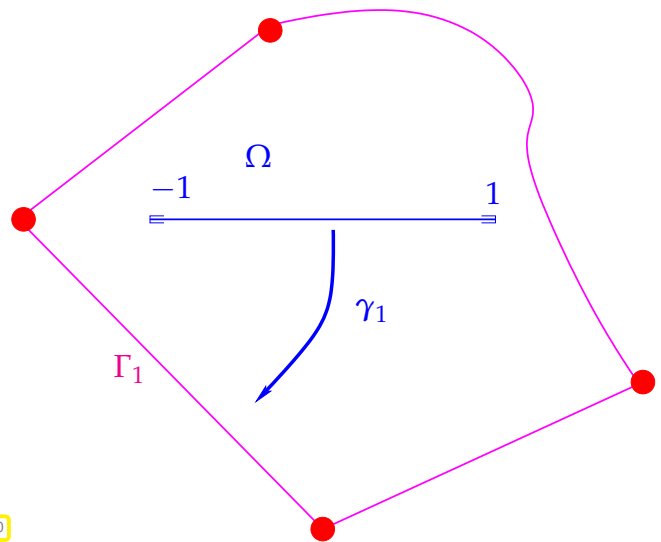
Since ultimately we are interested in discretization, we restrict our shapes to “engineering geometries” that can be described (in the context of a Bezier or NURBS model) by a few parameter.

§1.2.1.4 (2D: Curved Lipschitz polygons [NumPDE § 1.2.1.14]) Now we examine a relevant class of planar domains in computational engineering known as curved Lipschitz polygons:

Assumption 1.2.1.5.

The boundary Γ of Ω can be partitioned into finitely many open edges $\Gamma_1, \dots, \Gamma_M, M \in \mathbb{N}$, such that

- ◆ $\Gamma = \bar{\Gamma}_1 \cup \dots \cup \bar{\Gamma}_M$,
- ◆ $\Gamma_i \cap \Gamma_j = \emptyset$ for $i \neq j$,
- ◆ for every $j \in \{1, \dots, M\}$ there is a C^2 -function $\gamma_j : [-1, 1] \rightarrow \bar{\Gamma}_j \subset \mathbb{R}^2$ with $\frac{d}{dt}\gamma \neq 0$ (a smooth parameterization).



We can distinguish corners (●) and edges (—) of Γ Fig. 10

§1.2.1.6 (Curvilinear Lipschitz polyhedra) Eligible 3D domains $\Omega \subset \mathbb{R}^3$:

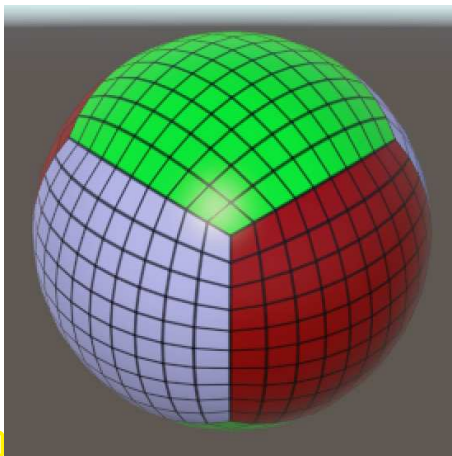


Fig. 11

Assumption 1.2.1.7.

The boundary Γ of Ω is Lipschitz and can be partitioned into finitely many open faces $\Gamma_1, \dots, \Gamma_M, M \in \mathbb{N}$, such that

- ◆ $\Gamma = \bar{\Gamma}_1 \cup \dots \cup \bar{\Gamma}_M$,
- ◆ $\Gamma_i \cap \Gamma_j = \emptyset$ for $i \neq j$,
- ◆ for every $j \in \{1, \dots, M\}$ there is an open planar polygon $\Pi_j \subset \mathbb{R}^2$ and a bijective C^2 -function $\gamma_j : \bar{\Pi}_j \rightarrow \bar{\Gamma}_j \subset \mathbb{R}^3$ (a smooth parameterization).

◁ Sphere composed of patches parameterized over squares (forums.tigsources.com)

§1.2.1.8 (Curve and surface integrals) The formulas (1.2.1), (1.2.1.1), (1.2.1.3) involve integrals of scalar integrands over $\Gamma := \partial\Omega$. Calculus supplies the following formulas:

- 2D ($d = 2$): Under Ass. 1.2.1.5 with the notations from there and for a piecewise continuous $f : \Gamma \rightarrow \mathbb{R}$ holds

$$\int_{\Gamma} f(x) dS(x) = \sum_{j=1}^M \int_{-1}^1 f(\gamma_j(t)) \|\dot{\gamma}_j(t)\| dt, \quad \dot{\gamma}_j(t) := \frac{d\gamma_j}{dt}(t) \in \mathbb{R}^2, \quad (1.2.1.9)$$

where $\|\cdot\|$ designates the Euclidean norm of a vector.

- 3D ($d = 3$): With Ass. 1.2.1.7 and its notations and $f : \Gamma \rightarrow \mathbb{R}$ integrable we have [Str09, Rem. 8.6.1]

$$\int_{\Gamma} f(x) dS(x) = \sum_{j=1}^M \int_{\Pi_j} f(\gamma_j(\hat{x})) g_j(\hat{x}) d\hat{x}, \quad g_j(\hat{x}) := \left(\det \underbrace{\left(D\gamma_j^T(\hat{x}) D\gamma_j(\hat{x}) \right)}_{\in \mathbb{R}^{2,2}} \right)^{1/2}. \quad (1.2.1.10)$$

The Jacobians $D\gamma_j$ map $\Pi_j \mapsto \mathbb{R}^{3,2}$ and the function g_j is the Gram determinant of γ_j .

┘

1.2.2 Fundamental Solutions

1.2.2.1 Potential of a Point Charge


We consider electrostatic in a homogeneous, isotropic, dielectric medium, that is, we assume $\epsilon \equiv 1$ after scaling, recall Rem. 1.1.6.8. Then the repulsive Coulomb force acting between two unit charges (in rescaled units) located at $x, y \in \mathbb{R}^3$ is

$$\mathbf{f} = \frac{1}{4\pi} \frac{\mathbf{y} - \mathbf{x}}{\|\mathbf{y} - \mathbf{x}\|^3}. \tag{1.2.2.1}$$

Recalling the link between Coulomb force on a point charge and the electric field expressed in (1.1.2.1), we conclude that

$$\mathbf{E}_x(\mathbf{y}) := \frac{1}{4\pi} \frac{\mathbf{y} - \mathbf{x}}{\|\mathbf{y} - \mathbf{x}\|^3}, \quad \mathbf{y} \neq \mathbf{x}, \tag{1.2.2.2}$$

describes is the electric field engendered by a unit charge at x . Now we are looking for the associated electric scalar potential (\rightarrow § 1.1.2.7), denoted by $\mathbf{y} \mapsto G_x(\mathbf{y})$ and satisfying $\mathbf{grad} G_x = -\mathbf{E}_x(\mathbf{y})$.

 Notation: We write $L_y F(x, y)$ to indicated that the differential operator L “acts on y ” and x is treated as a mere parameter; generalizes the concept of a partial derivative.

Gradients of functions depending on $\|\mathbf{x}\|$ are aligned with the radial direction:

$$\mathbf{grad}\{x \mapsto \|\mathbf{x}\|\} = \frac{\mathbf{x}}{\|\mathbf{x}\|} \Rightarrow \mathbf{grad}\{x \mapsto \frac{1}{\|\mathbf{x}\|}\} = -\frac{1}{\|\mathbf{x}\|^2} \cdot \frac{\mathbf{x}}{\|\mathbf{x}\|},$$

which reveals that \mathbf{E}_x can be expressed as a gradient:

$$\blacktriangleright \quad \mathbf{E}_x(\mathbf{y}) = \frac{1}{4\pi} \frac{1}{\|\mathbf{y} - \mathbf{x}\|^2} \frac{\mathbf{y} - \mathbf{x}}{\|\mathbf{y} - \mathbf{x}\|} = -\mathbf{grad}_y \left\{ \frac{1}{4\pi} \frac{1}{\|\mathbf{y} - \mathbf{x}\|} \right\}.$$

Potential due to a point charge

A point charge at $x \in \mathbb{R}^3$ generates the potential

$$\mathbf{y} \mapsto G_x(\mathbf{y}) := \frac{1}{4\pi} \frac{1}{\|\mathbf{x} - \mathbf{y}\|}, \quad \mathbf{y} \neq \mathbf{x} \tag{1.2.2.4}$$

Remark 1.2.2.5 (Properties of the potential due to a point charge) From (1.2.2.4) we read off that the potential $G_x x \in \mathbb{R}^3$,

- ◆ is a function of the the distance $\|\mathbf{x} - \mathbf{y}\|$ only,
- ◆ is smooth away from x : $G_x \in C^\infty(\mathbb{R}^3 \setminus \{x\})$,
- ◆ is harmonic: $\Delta G_x(\mathbf{y}) = 0$ for all $\mathbf{y} \in \mathbb{R}^3 \setminus \{x\}$,
- ◆ satisfies the decay conditions (1.1.7.1)

$$|G_x(\mathbf{y})| = O(\|\mathbf{y}\|^{-1}) \quad , \quad \|\mathbf{grad} G_x(\mathbf{y})\| = O(\|\mathbf{y}\|^{-2}) \quad \text{for} \quad \|\mathbf{y}\| \rightarrow \infty, \tag{1.2.2.6}$$

- ◆ has a **singularity** at x

$$|G_x(\mathbf{y})| = O(\|\mathbf{y} - \mathbf{x}\|^{-1}) \quad , \quad \|\mathbf{grad} G_x(\mathbf{y})\| = O(\|\mathbf{x} - \mathbf{y}\|^{-2}) \quad \text{for } \mathbf{y} \rightarrow \mathbf{x} \quad , \quad (1.2.2.7)$$

- ◆ and, owing to the singularity in x , $G_x \notin H^1(\mathbb{R}^3)$ (\rightarrow (1.1.2.18)): the field generated by a point charge **fails to have finite energy**, “point charge” is a non-physical concept (of great usefulness for formal considerations, however).

┘

1.2.2.2 Potential of a Line Charge

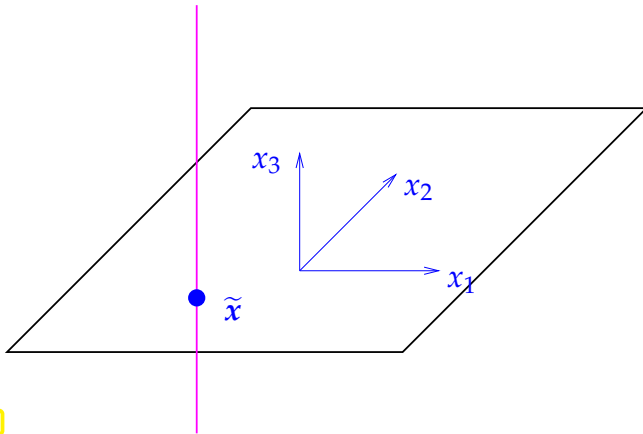


Fig. 12

We adopt the x_3 -translation invariant setting underlying 2D electrostatics, see § 1.1.6.13: a 2D point charge becomes a 3D infinite **line charge** concentrated on $\{x \in \mathbb{R}^3 : \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \tilde{x}\}$, $\tilde{x} \in \mathbb{R}^2$

By linear **superposition** in 3D we determine the electric field $\tilde{\mathbf{E}}_{\tilde{x}}(x_1, x_2) \in \mathbb{R}^2$ of the line charge in the $x_1 - x_2$ -plane.

By symmetry arguments (*) and suitable substitutions, we compute

$$\begin{aligned} \mathbf{E}_{\tilde{x}}(\tilde{\mathbf{y}}) &= \frac{1}{4\pi} \int_{-\infty}^{\infty} \frac{\begin{bmatrix} \tilde{\mathbf{y}} \\ 0 \end{bmatrix} - \begin{bmatrix} \tilde{x} \\ \zeta \end{bmatrix}}{\left(\|\tilde{x} - \tilde{\mathbf{y}}\|^2 + \zeta^2\right)^{3/2}} d\zeta \stackrel{(*)}{=} \frac{1}{4\pi} \begin{bmatrix} \tilde{\mathbf{y}} - \tilde{x} \\ 0 \end{bmatrix} \int_{-\infty}^{\infty} \frac{1}{\left(\|\tilde{x} - \tilde{\mathbf{y}}\|^2 + \zeta^2\right)^{3/2}} d\zeta \\ &= \frac{1}{4\pi} \begin{bmatrix} \tilde{\mathbf{y}} - \tilde{x} \\ 0 \end{bmatrix} \frac{1}{\|\tilde{x} - \tilde{\mathbf{y}}\|^2} \int_{-\infty}^{\infty} \frac{1}{(1 + \zeta^2)^{3/2}} d\zeta = \frac{1}{2\pi} \begin{bmatrix} \tilde{\mathbf{y}} - \tilde{x} \\ 0 \end{bmatrix} . \end{aligned}$$

As expected there is no x_3 -component and we have found

$$\tilde{\mathbf{E}}_{\tilde{x}}(\tilde{\mathbf{y}}) = \frac{1}{2\pi} \frac{\tilde{\mathbf{y}} - \tilde{x}}{\|\tilde{x} - \tilde{\mathbf{y}}\|^2} = -\mathbf{grad}_{\tilde{\mathbf{y}}} \left\{ -\frac{1}{2\pi} \log\|\tilde{x} - \tilde{\mathbf{y}}\| \right\} \quad , \quad \tilde{\mathbf{y}} \in \mathbb{R}^2 \setminus \{\tilde{x}\} . \quad (1.2.2.8)$$

Thus we have also identified the associated 2D potential.

Potential of a point charge in 2D

the scalar potential of a point charge at $\tilde{x} \in \mathbb{R}^2$ is

$$G_{\tilde{x}}(\tilde{\mathbf{y}}) := -\frac{1}{2\pi} \log\|\tilde{x} - \tilde{\mathbf{y}}\| \quad , \quad \tilde{\mathbf{y}} \in \mathbb{R}^2 \setminus \{\tilde{x}\} . \quad (1.2.2.10)$$

Remark 1.2.2.11 (Properties of the potential of a point charge in 2D) This echoes Rem. 1.2.2.11. The potential $G_{\tilde{x}}$ is a smooth harmonic function for $\mathbf{y} \neq \mathbf{x}$ and

- ◆ satisfies the decay conditions (1.1.7.4)

$$|G_{\tilde{x}}(\tilde{\mathbf{y}})| = O(\log\|\tilde{\mathbf{y}}\|) \quad , \quad \|\mathbf{grad} G_{\tilde{x}}(\tilde{\mathbf{y}})\| = O(\|\tilde{\mathbf{y}}\|^{-1}) \quad \text{for } \|\tilde{\mathbf{y}}\| \rightarrow \infty \quad , \quad (1.2.2.12)$$

◆ $G_{\tilde{x}}$ has a logarithmic singularity at \tilde{x}

$$|G_{\tilde{x}}(\tilde{y})| = O(\log\|\tilde{y} - \tilde{x}\|) \quad , \quad \|\mathbf{grad} G_{\tilde{x}}(\tilde{y})\| = O(\|\tilde{x} - \tilde{y}\|^{-1}) \quad \text{for } \tilde{y} \rightarrow \tilde{x} \quad , \quad (1.2.2.13)$$

◆ and the energy of the electric field of a 2D point charge is not bounded: $G_{\tilde{x}} \notin H^1(\mathbb{R}^3)$.

┘

1.2.2.3 Distributional View: $LG = \delta_0$

📎 Notation: For the potentials (1.2.2.4), (1.2.2.10) caused by point charges at $x \in \mathbb{R}^d$, $d = 2, 3$, we now indiscriminately write $G^\Delta(x, y)$ to emphasize the symmetric roles of both arguments.

In both 2D and 3D

$$\int_{\mathbb{R}^d} |G^\Delta(x, y)| dy < \infty \quad \forall x \in \mathbb{R}^d \quad ,$$

so that all the integrals below exist as **improper integrals** [Str09, Sect. 6.4]. For $x \in \mathbb{R}^d$ and a smooth *compactly supported* function $w \in C_0^\infty(\mathbb{R}^d)$ we find by $\Delta_y G^\Delta(x, y) = 0$ for $x \neq y$ and Green's second formula from Thm. 1.2.1.2,

$$\int_{\Omega} \mathbf{grad} u \cdot \mathbf{grad} v dx = - \int_{\Omega} \Delta u v dx + \int_{\partial\Omega} \mathbf{grad} u \cdot \mathbf{n} v dS \quad , \quad (1.2.1.1)$$

with $v \leftarrow w$ and $u \leftarrow \{y \mapsto G^\Delta(x, y)\}$ that

$$\begin{aligned} \int_{\mathbb{R}^3} G^\Delta(x, y) (-\Delta w)(y) dy &= \lim_{\epsilon \rightarrow 0} \int_{\|y\| > \epsilon} G^\Delta(x, y) (-\Delta w)(y) dy \\ &= \lim_{\epsilon \rightarrow 0} \int_{\|y\| > \epsilon} \underbrace{(-\Delta_y G^\Delta)}_{\rightarrow 0}(x, y) w(y) dy - \\ &\quad \lim_{\|y\| = \epsilon} \int G^\Delta(x, y) \mathbf{grad} w(y) \cdot \mathbf{n}(y) - \mathbf{grad}_y G^\Delta(x, y) \cdot \mathbf{n}(y) w(y) dS(y) \quad . \end{aligned}$$

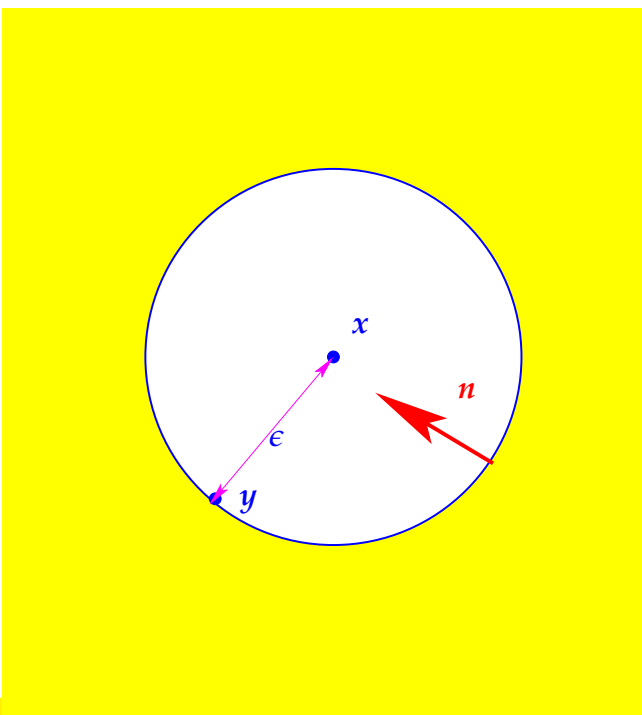


Fig. 13

Next, we examine the limit of the surface integral for $d = 3$:

In the case $d = 3$ we have concrete formulas at our disposal. For $y \in \partial B_\epsilon(x)$ we find

$$\begin{aligned} \mathbf{n}(y) &= \epsilon^{-1}(x - y) \quad , \\ G^\Delta(x, y) &= \frac{1}{4\pi\|x - y\|} = \frac{1}{4\pi\epsilon} \quad , \\ \mathbf{grad}_y G^\Delta(x, y) &= \frac{1}{4\pi} \frac{x - y}{\|x - y\|^3} = \frac{x - y}{4\pi\epsilon^3} \quad . \end{aligned}$$

We plug this into the surface integral over the ϵ -sphere:

$$\blacktriangleright \int_{\mathbb{R}^3} G^\Delta(x, \mathbf{y})(-\Delta w)(\mathbf{y}) \, d\mathbf{y} = \lim_{\epsilon \rightarrow 0} \int_{\|\mathbf{y}\|=\epsilon} -\frac{1}{4\pi\epsilon} \mathbf{grad} w(\mathbf{y}) \cdot \mathbf{n}(\mathbf{y}) + \frac{1}{4\pi\epsilon^2} w(\mathbf{y}) \, dS(\mathbf{y}) .$$

Since $w \in C_0^\infty(\mathbb{R}^3)$ is smooth and the area of the sphere shrinks like $O(\epsilon^2)$ for $\epsilon \rightarrow 0$, the contribution of the first term vanishes in the limit.

$$\blacktriangleright \int_{\mathbb{R}^3} G^\Delta(x, \mathbf{y})(-\Delta w)(\mathbf{y}) \, d\mathbf{y} = \lim_{\epsilon \rightarrow 0} \int_{\|\mathbf{y}\|=\epsilon} \frac{1}{4\pi\epsilon^2} w(\mathbf{y}) \, dS(\mathbf{y}) = w(x) . \tag{1.2.2.14}$$

The same result holds for $d = 2$.

Definition 1.2.2.15. Fundamental solution

A function $G^L : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ is a **fundamental solution** (FS) for a second-order scalar linear differential operator L , if

- (i) G^L is C^∞ -smooth on $\{(x, \mathbf{y}) \in \mathbb{R}^d \times \mathbb{R}^d : x \neq \mathbf{y}\}$,
- (ii) for all $x \in \mathbb{R}^d$: $L_y G^L(x, \mathbf{y}) = 0$ on $\mathbb{R}^d \setminus \{x\}$
- (iii) $\mathbf{y} \mapsto G^L(x, \mathbf{y})$ satisfies the appropriate decay conditions (1.1.7.1)/(1.1.7.4),
- (iv) $\mathbf{y} \mapsto G^L(x, \mathbf{y})$ is integrable on \mathbb{R}^d ,
- (v) for every $x \in \mathbb{R}^d, w \in C_0^\infty(\mathbb{R}^d)$

$$\int_{\mathbb{R}^d} G^L(x, \mathbf{y})(L^* w)(\mathbf{y}) \, d\mathbf{y} = w(x) . \tag{1.2.2.16}$$

Here L^* is the (formal) **adjoint differential operator** of L defined by

$$\int_{\Omega} (Lw)(x) v(x) \, dx = \int_{\Omega} w(x) (L^*v)(x) \, dx \quad \forall w, v \in C_0^\infty(\mathbb{R}^d) . \tag{1.2.2.17}$$


For all differential operators of the form $Lu := -\operatorname{div}(\mathbf{A} \operatorname{grad} u) + cu$ with $\mathbf{A} = \mathbf{A}^\top \in \mathbb{R}^{d,d}, c \in \mathbb{R}$, in particular $L = -\Delta$, we easily see from Green's formulas that $L^* = L$.

Remark 1.2.2.18 (" $L_y G^L = \delta_x$ ") "Testing equalities with smooth functions" is the idea underlying the **calculus of distributions** [RR04, Ch. 5]. Sloppily speaking, a distribution is a linear functional on $C_0^\infty(\mathbb{R}^d)$, continuous in a particular topology. In distributional calculus we can concisely rephrase

$$\int_{\mathbb{R}^d} G^L(x, \mathbf{y})(L^* w)(\mathbf{y}) \, d\mathbf{y} = w(x) \quad \forall w, x \iff L_y G^L(x, \mathbf{y}) = \delta_x \text{ in } \mathcal{D}(\mathbb{R}^d)' \quad \forall x \in \mathbb{R}^d ,$$

where δ_x is the so-called **δ -distribution** supported in $x \in \mathbb{R}^d$, that is, the point-evaluation functional:

$$\forall w \in C_0^\infty(\mathbb{R}^d), x \in \mathbb{R}^d: \int_{\mathbb{R}^d} \delta_x(\mathbf{y}) w(\mathbf{y}) \, d\mathbf{y} = w(x) . \tag{1.2.2.19}$$

 Notation: If an equation is supposed to hold in distributional sense, one often writes "in $\mathcal{D}(\Omega)'$ ".

┘

A mathematical discussion of fundamental solutions can be found in [McL00, pp. 191-197]. Existence and uniqueness are discussed there.

Theorem 1.2.2.20. Uniqueness of fundamental solutions

Fundamental solutions according to Def. 1.2.2.15 for differential operators (1.2.0.1) are unique.

§1.2.2.21 (Symmetries of fundamental solutions) If a differential operator L

- is symmetric in the sense that $L = L^*$, then $G^L(x, y) = G^L(y, x)$ for all $x, y \in \mathbb{R}^d, x \neq y$.
- has constant coefficients (L is **translation-invariant** in this case), then its fundamental solution depends only on $x - y$: $G^L(x, y) = G^L(x - y), x \neq y$.
- has constant coefficients and is **rotation-invariant**, then $G^L(x, y) = G^L(\|x - y\|)$.

Above, “abusing notations”, we used the same symbol G^L for different functions.

Definition 1.2.2.22. Rotation invariance

An operator $D : C^\infty(\mathbb{R}^d) \rightarrow C^\infty(\mathbb{R}^d)$ is **rotation-invariant**, if it “commutes with rotations”

$$(Dw)(Qx) = (D\{x \mapsto w(Qx)\})(x) \quad \forall w \in C^\infty(\mathbb{R}^d), \tag{1.2.2.23}$$

and for all **orthogonal** matrices $Q \in \mathbb{R}^{d,d}$.

┘

EXAMPLE 1.2.2.24 (Computing G^Δ in 3D) The rules from § 1.2.2.21 pave the way for easy computation of fundamental solutions for rotationally symmetric differential operators with constant coefficients by means of separation of variables.

We demonstrate the computation of the fundamental solution G^Δ for the Laplacian $L := -\Delta$ in 3D using **spherical coordinates**

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} r \cos \phi \sin \theta \\ r \sin \phi \sin \theta \\ r \cos \theta \end{bmatrix}, \quad r \geq 0, 0 \leq \phi < 2\pi, 0 \leq \theta < \pi. \tag{1.2.2.25}$$

Also recall the formula for the Laplacian in spherical coordinates

$$\Delta u = \frac{\partial^2 u}{\partial r^2} + \frac{2}{r} \frac{\partial u}{\partial r} + \frac{1}{r^2 \sin^2 \theta} \frac{\partial^2 u}{\partial \phi^2} + \frac{1}{r^2} \frac{\partial^2 u}{\partial \theta^2} + \frac{1}{r^2} \cot \theta \frac{\partial u}{\partial \theta}. \tag{1.2.2.26}$$

The Laplacian $-\Delta$ is the most prominent example of a linear differential operator that is both translation- and rotation-invariant. Thus, from § 1.2.2.21 we know that $G^\Delta(x, y) = G^\Delta(\|x - y\|)$. So we can set $G^\Delta(x, y) = f(\|x - y\|)$ and the requirement $\Delta_y G^\Delta(x, y) = 0$ leads to the linear second-order **ordinary differential equation**

$$\frac{\partial^2 f}{\partial r^2} + \frac{2}{r} \frac{\partial f}{\partial r} = f''(r) + \frac{2}{r} f'(r) = 0.$$

It has the family of solutions

$$f(r) = A + Br^{-1} \quad r \neq 0, \quad A, B \in \mathbb{R}.$$

By the decay conditions for fundamental solutions we know that $f(r) = O(r^{-1})$ for $r \rightarrow \infty$ has to be satisfied, which entails $A = 0$. The constant B must be chosen to satisfy (1.2.2.16). Eventually,

we recover the potential (1.2.2.4) of a point charge as fundamental solution.

┘

EXAMPLE 1.2.2.27 (Fundamental solution for 2nd-order partial differential operator) We consider the symmetric second-order scalar differential operator

$$Lu = -\operatorname{div}(\mathbf{A} \operatorname{grad} u), \quad \mathbf{A} = \mathbf{A}^\top \in \mathbb{R}^d \text{ s.p.d.} . \tag{1.2.2.28}$$

Its associate fundamental solution G^L will be symmetric and of the form $G^L(\mathbf{x}, \mathbf{y}) = G^L(\mathbf{x} - \mathbf{y})$, and must fulfill

$$\int_{\mathbb{R}^d} G^L(\mathbf{x} - \mathbf{y})(Lw)(\mathbf{y}) \, d\mathbf{y} = w(\mathbf{x}) \quad \forall w \in C_0^\infty(\mathbb{R}^d), \mathbf{x} \in \mathbb{R}^d . \tag{1.2.2.29}$$



Idea: Try to express G^L in terms of the fundamental solution for $-\Delta$.

To begin with recall the formulas

$$\operatorname{div} \mathbf{j} = \operatorname{Tr}(\mathbf{D}\mathbf{j}) \quad \text{for } \mathbf{j} : \mathbb{R}^d \rightarrow \mathbb{R}^d , \tag{1.2.2.30a}$$

$$\Delta w = \operatorname{Tr}(\underbrace{\mathbf{D} \operatorname{grad} w}_{\text{Hessian of } w}) \quad \text{for } w : \mathbb{R}^d \rightarrow \mathbb{R} , \tag{1.2.2.30b}$$

where $\operatorname{Tr} : \mathbb{R}^{d,d} \rightarrow \mathbb{R}$ is the trace operator for matrices,

$$\operatorname{Tr} \mathbf{M} = \sum_{j=1}^d (\mathbf{M})_{j,j} \quad \text{for } \mathbf{M} \in \mathbb{C}^{d,d} , \tag{1.2.2.31}$$

that satisfies $\operatorname{Tr}(\mathbf{X}\mathbf{Y}) = \operatorname{Tr}(\mathbf{Y}\mathbf{X})$.

We decompose $\mathbf{A} = \mathbf{C}\mathbf{C}^\top$, which can be achieved by means of a Cholesky-decomposition [NumCSE § 2.8.0.13]. For a function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ we define its pullback under the linear mapping induced by \mathbf{C} according to

$$\widehat{f}(\widehat{\mathbf{y}}) := f(\mathbf{C}\widehat{\mathbf{y}}) \quad \widehat{\mathbf{y}} \in \mathbb{R}^d .$$

Using the chain rule and (1.2.2.30b), we obtain for $\mathbf{x} \in \mathbb{R}^d$

$$\begin{aligned} \Delta \widehat{u}(\mathbf{x}) &= \operatorname{Tr}(\mathbf{D} \operatorname{grad} \{x \mapsto u(\mathbf{C}x)\}) = \operatorname{Tr} \mathbf{D}\{x \mapsto \mathbf{C}^\top(\operatorname{grad} u)(\mathbf{C}x)\} \\ &= \operatorname{Tr}(\mathbf{C}^\top(\mathbf{D} \operatorname{grad} u)(\mathbf{C}x)\mathbf{C}) = \operatorname{Tr}(\mathbf{C}\mathbf{C}^\top(\mathbf{D} \operatorname{grad} u)(\mathbf{C}x)) = \operatorname{Tr}(\mathbf{D}(\mathbf{C}\mathbf{C}^\top \operatorname{grad} u))(\mathbf{C}x) \\ &= (\operatorname{div}(\mathbf{C}\mathbf{C}^\top \operatorname{grad} u))(\mathbf{C}x) = (\operatorname{div}(\mathbf{A} \operatorname{grad} u))(\mathbf{C}x) = (Lu)(\mathbf{C}x) . \end{aligned}$$

We plug this identity into (1.2.2.29) and use the transformation formula for multi-dimensional integrals with $\widehat{\mathbf{y}} = \mathbf{C}^{-1}\mathbf{y}$.

$$\begin{aligned} \widehat{w}(\widehat{\mathbf{x}}) &= w(\mathbf{C}\widehat{\mathbf{x}}) = \int_{\mathbb{R}^d} G^L(\mathbf{C}\widehat{\mathbf{x}} - \mathbf{y})(Lw)(\mathbf{y}) \, d\mathbf{y} \\ &= \int_{\mathbb{R}^d} G^L(\mathbf{C}\widehat{\mathbf{x}} - \mathbf{C}\widehat{\mathbf{y}})(Lw)(\mathbf{C}\widehat{\mathbf{y}}) |\det \mathbf{C}| \, d\widehat{\mathbf{y}} \\ &= \int_{\mathbb{R}^d} \underbrace{\sqrt{\det \mathbf{A}} G^L(\mathbf{C}(\widehat{\mathbf{x}} - \widehat{\mathbf{y}}))}_{=G^\Delta(\mathbf{x},\mathbf{y})} (-\Delta \widehat{w})(\widehat{\mathbf{y}}) \, d\widehat{\mathbf{y}} , \end{aligned}$$

for any $w \in C_0^\infty(\mathbb{R}^d)$. By the uniqueness of the fundamental solution we conclude

$$G^L(x, y) = \frac{1}{\sqrt{\det \mathbf{A}}} G^\Delta(\mathbf{C}^{-1}(x - y)), \quad x \neq y, \tag{1.2.2.32}$$

where G_Δ is the fundamental solution for $-\Delta$

$$G^\Delta(x, y) = \begin{cases} -\frac{1}{2\pi} \log \|x - y\| & , \text{ if } d = 2, \\ \frac{1}{4\pi} \frac{1}{\|x - y\|} & , \text{ if } d = 3. \end{cases} \tag{1.2.2.33}$$

Eventually, as

$$\|\mathbf{C}^{-1}(x - y)\|^2 = (x - y)^\top \mathbf{C}^{-\top} \mathbf{C}^{-1}(x - y) = (x - y)^\top \mathbf{A}^{-1}(x - y),$$

we get

$$G^L(x, y) = \frac{1}{\sqrt{\det \mathbf{A}}} \cdot \begin{cases} -\frac{1}{2\pi} \log \sqrt{(x - y)^\top \mathbf{A}^{-1}(x - y)} & , \text{ if } d = 2, \\ \frac{1}{4\pi} \frac{1}{\sqrt{(x - y)^\top \mathbf{A}^{-1}(x - y)}} & , \text{ if } d = 3. \end{cases} \tag{1.2.2.34}$$

┘

1.2.3 Volume Potential Representation

We return to $L = -\Delta$ (electrostatics in a homogeneous, isotropic medium with $\epsilon = \mathbf{I}$), where we have the fundamental solutions

$$G^\Delta(x, y) = G^\Delta(x - y) = \begin{cases} -\frac{1}{2\pi} \log \|x - y\| & , \text{ if } d = 2, \\ \frac{1}{4\pi} \frac{1}{\|x - y\|} & , \text{ if } d = 3. \end{cases} \tag{1.2.2.33}$$

Since $\Delta^* = \Delta$, by the very property

$$\int_{\mathbb{R}^d} G^\Delta(x, y)(-\Delta w)(y) dy = w(x) \quad \forall x \in \mathbb{R}^d, \forall w \in C_0^\infty(\mathbb{R}^d), \tag{1.2.2.16}$$

of the fundamental solution, we conclude that for every smooth *compactly supported* source charge distribution $\rho \in C_0^\infty(\mathbb{R}^d)$, if u solves

$$-\Delta u = \rho \quad \text{in } \mathbb{R}^d, \quad u \text{ satisfies decay conditions for } \|x\| \rightarrow \infty,$$

then we have the **volume potential representation**

$$u(x) = \int_{\mathbb{R}^d} G^\Delta(x, y)\rho(y) dy, \quad x \in \mathbb{R}^3. \tag{1.2.3.1}$$

The operator on the right-hand side (rhs) of (1.2.3.1) is a **volume integral operator** with **kernel** G^Δ . It has been given a special name:

Definition 1.2.3.2. Newton potential

The linear operator

$$N_\Delta : \begin{cases} C_0^\infty(\mathbb{R}^d) & \rightarrow C_0^\infty(\mathbb{R}^d) \\ \rho & \mapsto \int_{\mathbb{R}^d} G^\Delta(x - y)\rho(y) dy \end{cases} \tag{1.2.3.3}$$

is the **Newton potential** for $-\Delta$.

Supplement 1.2.3.4. The Newton potential on \mathbb{R}^d is a volume integral operator of **convolution** type.

Definition 1.2.3.5. Convolution of functions in \mathbb{R}^d

The **convolution** of two functions $f, g \in L^1(\mathbb{R}^d)$ is the function

$$(f * g)(x) := \int_{\mathbb{R}^d} f(x - y)g(y) \, dy = \int_{\mathbb{R}^d} f(y)g(x - y) \, dy .$$

Using this notation, obviously,

$$N_\Delta(\rho) = G^\Delta * \rho , \tag{1.2.3.6}$$

because of the structure of the fundamental solution,

$$\int_{\mathbb{R}^d} G^\Delta(x, y)\rho(y) \, dy = \int_{\mathbb{R}^d} G^\Delta(x - y)\rho(y) \, dy = \int_{\mathbb{R}^d} G^\Delta(y)\rho(x - y) \, dy .$$

This last expression also reveals that for $\rho \in C_0^\infty(\mathbb{R}^d)$ also $u \in C^\infty(\mathbb{R}^d)$, since $y \mapsto G^\Delta(y)$ is integrable on \mathbb{R}^d . ┘

Theorem 1.2.3.7. Decay of Newton potential

For compactly supported ρ the function $N_\Delta(\rho)$ complies with the decay conditions (1.1.7.1),

$$|N_\Delta(\rho)(x)| = O(\|x\|^{-1}) \quad \text{and} \quad \|\mathbf{grad} N_\Delta(\rho)\| = O(\|x\|^{-2}) \quad \text{for} \quad \|x\| \rightarrow \infty ,$$

for $d = 3$ and (1.1.7.4)

$$|N_\Delta(\rho)(x)| = O(\log\|x\|) \quad , \quad \|\mathbf{grad} N_\Delta(\rho)(x)\| = O(\|x\|^{-1}) \quad \text{for} \quad \|x\| \rightarrow \infty ,$$

for $d = 2$, respectively.

We can immediately conclude this from the decay properties of the fundamental solutions. The next assertion is clear from the definition of the norm of $\tilde{H}^{-1}(\mathbb{R}^d)$ given in Def. 1.1.8.1.

Corollary 1.2.3.8. Mapping properties of the Newton potential

The Newton potential N_Δ as defined in (1.2.3.3) can be extended to a continuous mapping $\tilde{H}^{-1}(\mathbb{R}^d) \rightarrow H^1(\mathbb{R}^d)$ (\rightarrow 1.1.8).

The relationship (1.2.3.1) tells us that the Newton potential provides a solution operator for the Poisson problem $-\Delta u = \rho$ (+ decay conditions) in the whole space \mathbb{R}^d .

$$\Delta(N_\Delta \rho) = \rho \quad \forall \rho \in \tilde{H}^{-1}(\mathbb{R}^d) . \tag{1.2.3.9}$$

Remark 1.2.3.10 (The Newton potential from a physics perspective) We can imagine a source charge distribution ρ as being composed of (infinitely) many small point charges:

$$\rho = \sum_{j=1}^N q_j \delta_{x_j} , \quad x_j \in \mathbb{R}^d , \quad q_j \in \mathbb{R} .$$

The potentials generated by all these point charges can be added up and yield the potential

$$u(\mathbf{x}) = \sum_{j=1}^N q_j G^\Delta(\mathbf{x} - \mathbf{x}_j) .$$

Sending $N \rightarrow \infty$ and appealing to “intuitive Riemann integration” yields the Newton potential solution of $-\Delta u = \rho$ on \mathbb{R}^d . \lrcorner

1.2.4 Boundary Potential Representation

The manipulations in Section 1.2.2.3 that led to (1.2.2.14) and, in the sequel, to the volume potential representation for solutions of $-\Delta u = \rho$ on \mathbb{R}^d ,

$$u(\mathbf{x}) = \int_{\mathbb{R}^d} G^\Delta(\mathbf{x}, \mathbf{y}) \rho(\mathbf{y}) \, d\mathbf{y} , \quad \mathbf{x} \in \mathbb{R}^3 , \quad (1.2.3.1)$$

were carried out on the entire space. Now we move them to a *bounded* Lipschitz domain $\Omega \subset \mathbb{R}^d$, $d = 2, 3$.

Pick $\mathbf{x} \in \Omega$ and $w \in C^2(\overline{\Omega})$. Appealing to Green’s second formula

$$\int_{\partial\Omega} u \Delta v - v \Delta u \, d\mathbf{x} = \int_{\partial\Omega} u \mathbf{grad} v \cdot \mathbf{n} - v \mathbf{grad} u \cdot \mathbf{n} \, dS(\mathbf{x}) . \quad (1.2.1.3)$$

from Thm. 1.2.1.2 with $u \leftarrow w$ and $v \leftarrow G^\Delta$ we get

$$\begin{aligned} \int_{\Omega} G^\Delta(\mathbf{x}, \mathbf{y}) (-\Delta w) \, d\mathbf{y} &= \lim_{\epsilon \rightarrow 0} \int_{\|\mathbf{y}-\mathbf{x}\| > \epsilon} G^\Delta(\mathbf{x}, \mathbf{y}) (-\Delta w)(\mathbf{y}) \, d\mathbf{y} \\ &= - \lim_{\epsilon \rightarrow 0} \int_{\|\mathbf{y}-\mathbf{x}\| = \epsilon} G^\Delta(\mathbf{x}, \mathbf{y}) \mathbf{grad} w(\mathbf{y}) \cdot \mathbf{n}(\mathbf{y}) - \mathbf{grad}_{\mathbf{y}} G^\Delta(\mathbf{x}, \mathbf{y}) \cdot \mathbf{n}(\mathbf{y}) w(\mathbf{y}) \, dS(\mathbf{y}) \\ &\quad - \int_{\partial\Omega} G^\Delta(\mathbf{x}, \mathbf{y}) \mathbf{grad} w(\mathbf{y}) \cdot \mathbf{n}(\mathbf{y}) - \mathbf{grad}_{\mathbf{y}} G^\Delta(\mathbf{x}, \mathbf{y}) \cdot \mathbf{n}(\mathbf{y}) w(\mathbf{y}) \, dS(\mathbf{y}) \\ &= w(\mathbf{x}) - \int_{\partial\Omega} G^\Delta(\mathbf{x}, \mathbf{y}) \mathbf{grad} w(\mathbf{y}) \cdot \mathbf{n}(\mathbf{y}) - \mathbf{grad}_{\mathbf{y}} G^\Delta(\mathbf{x}, \mathbf{y}) \cdot \mathbf{n}(\mathbf{y}) \, dS(\mathbf{y}) w(\mathbf{y}) , \end{aligned}$$

and, based on the same limit arguments that yielded (1.2.2.14), we arrive at:

$$\begin{aligned} w(\mathbf{x}) &= \int_{\Omega} G^\Delta(\mathbf{x}, \mathbf{y}) (-\Delta w)(\mathbf{y}) \, d\mathbf{y} + \int_{\partial\Omega} G^\Delta(\mathbf{x}, \mathbf{y}) \mathbf{grad} w(\mathbf{y}) \cdot \mathbf{n}(\mathbf{y}) \, dS(\mathbf{y}) - \\ &\quad \int_{\partial\Omega} \mathbf{grad}_{\mathbf{y}} G^\Delta(\mathbf{x}, \mathbf{y}) \cdot \mathbf{n}(\mathbf{y}) w(\mathbf{y}) \, dS(\mathbf{y}) , \quad \mathbf{x} \in \Omega . \end{aligned} \quad (1.2.4.1)$$

The derivation was carried out for $-\Delta$ for the sake of simplicity, but all arguments carry over to the more general scalar linear differential operator $Lu = -\mathbf{div}(\mathbf{A} \mathbf{grad} u) + cu$ from (1.2.0.1), starting from Green’s first formula with (1.1.6.2) with $\mathbf{j} := \mathbf{A} \mathbf{grad} u$. Eventually this yields the following generalization of (1.2.4.1) [SS10, Thm. 3.1.6].

Theorem 1.2.4.2. Integral representation formula

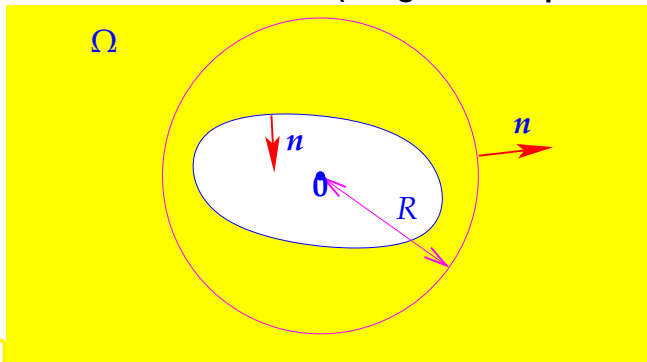
A solution $u \in C^2(\overline{\Omega})$ of $Lu := -\operatorname{div}(\mathbf{A} \operatorname{grad} u) - cu = \rho$ in Ω , \mathbf{A}, c as in (1.2.0.1), satisfies

$$u(x) = \int_{\Omega} G^L(x, \mathbf{y}) \rho(\mathbf{y}) \, d\mathbf{y} + \int_{\partial\Omega} G^L(x, \mathbf{y}) \mathbf{A} \operatorname{grad} u(\mathbf{y}) \cdot \mathbf{n}(\mathbf{y}) \, dS(\mathbf{y}) - \int_{\partial\Omega} \mathbf{A} \operatorname{grad}_{\mathbf{y}} G^L(x, \mathbf{y}) \cdot \mathbf{n}(\mathbf{y}) u(\mathbf{y}) \, dS(\mathbf{y}), \quad x \in \Omega, \tag{1.2.4.3}$$

where G^L is the fundamental solution for L , see Def. 1.2.2.15.

Note that the first term in (1.2.4.3) is the Newton potential from Def. 1.2.3.2.

Remark 1.2.4.4 (Integral representation formula for exterior domains)



Again, we elaborate the arguments for the Laplacian $L = -\Delta$.

If Ω is an exterior domain, that is, the open complement of a bounded Lipschitz domain, then we first apply Thm. 1.2.4.2 to $\widehat{\Omega} := \Omega \cap B_R(\mathbf{0})$ with $R > 0$ large enough such that $\partial\Omega \subset B_R(\mathbf{0})$, see Fig. 14.

Fig. 14

Note that $\partial\widehat{\Omega} = \partial\Omega \cup \partial B_R(\mathbf{0})$, so that (1.2.4.1) becomes

$$w(x) = \int_{\widehat{\Omega}} G^\Delta(x, \mathbf{y}) \rho(\mathbf{y}) \, d\mathbf{y} + \int_{\partial\Omega} G^\Delta(x, \mathbf{y}) \operatorname{grad} w(\mathbf{y}) \cdot \mathbf{n}(\mathbf{y}) \, dS(\mathbf{y}) - \int_{\partial\Omega} \operatorname{grad}_{\mathbf{y}} G^\Delta(x, \mathbf{y}) \cdot \mathbf{n}(\mathbf{y}) w(\mathbf{y}) \, dS(\mathbf{y}) + \int_{\|\mathbf{x}\|=R} G^\Delta(x, \mathbf{y}) \operatorname{grad} w(\mathbf{y}) \cdot \mathbf{n}(\mathbf{y}) \, dS(\mathbf{y}) - \int_{\|\mathbf{x}\|=R} \operatorname{grad}_{\mathbf{y}} G^\Delta(x, \mathbf{y}) \cdot \mathbf{n}(\mathbf{y}) w(\mathbf{y}) \, dS(\mathbf{y}).$$

Consider $d = 3$ and assume that w satisfies the decay conditions (1.1.7.1):

$$|w(x)| = O(\|x\|^{-1}) \quad \text{and} \quad \|\operatorname{grad} w(x)\| = O(\|x\|^{-2}) \quad \text{for} \quad \|x\| \rightarrow \infty.$$

In this case we have the following behavior of the integrands on $\partial B_R(\mathbf{0})$

$$\begin{aligned} G^\Delta(x, \mathbf{y}) \operatorname{grad} w(\mathbf{y}) \cdot \mathbf{n}(\mathbf{y}) &= O(\|\mathbf{y}\|^{-3}), \\ \operatorname{grad}_{\mathbf{y}} G^\Delta(x, \mathbf{y}) \cdot \mathbf{n}(\mathbf{y}) w(\mathbf{y}) &= O(\|\mathbf{y}\|^{-3}) \end{aligned} \quad \text{for} \quad \|\mathbf{y}\| \rightarrow \infty, \|\mathbf{x}\| = R.$$

Hence, in the limit $R \rightarrow \infty$, the contributions of $\partial B_R(\mathbf{0})$ vanish.

Theorem 1.2.4.5. Integral representation formula for 3D exterior domains

If $u \in C^2(\overline{\Omega})$ satisfies $-\Delta u = \rho$ in an exterior domain Ω plus the decay conditions (1.1.7.1), then for all $x \in \Omega$

$$u(x) = \int_{\Omega} G^\Delta(x, \mathbf{y}) \rho(\mathbf{y}) \, d\mathbf{y} + \int_{\partial\Omega} G^\Delta(x, \mathbf{y}) \operatorname{grad} u(\mathbf{y}) \cdot \mathbf{n}(\mathbf{y}) \, dS(\mathbf{y}) - \int_{\partial\Omega} \operatorname{grad}_{\mathbf{y}} G^\Delta(x, \mathbf{y}) \cdot \mathbf{n}(\mathbf{y}) u(\mathbf{y}) \, dS(\mathbf{y}), \tag{1.2.4.6}$$

where G^Δ is the fundamental solution for $-\Delta$, see (1.2.2.33).

For $d = 2$ a faster decay of u than stipulated by (1.1.7.4) has to be assumed in order to make (1.2.4.6) hold. ┘

1.2.5 Layer Potentials

Now we take a closer look at the building blocks of the integral representation formulas (1.2.4.3)/(1.2.4.6), in particular those terms mapping trace data (\rightarrow Notion 1.2.5.1) on the boundary $\partial\Omega$ back to the domain Ω .

Notion 1.2.5.1. Trace operator

A **trace operator** is a linear mapping from a function space on the volume domain Ω to a function space on (parts of) the boundary $\partial\Omega$.

The simplest trace operator is the plain restriction $C^0(\bar{\Omega}) \rightarrow C^0(\partial\Omega)$. We have also seen the tangential and normal component traces for vector fields in § 1.1.3.7.

Notion 1.2.5.2. (Layer) potentials

A (layer) potential is a linear mapping from a function space on $\partial\Omega$ into a function space on the volume domain Ω .

Remark 1.2.5.3 (Layer potentials and traces)

Obviously, (Layer) potentials (\rightarrow Notion 1.2.5.2) and trace operators (\rightarrow Notion 1.2.5.1) map into “opposite directions”

The integral representation formulas (1.2.4.3) contain two layer potentials acting on the traces

- ◆ $u|_{\partial\Omega} \hat{=}$ point-wise restriction of the potential u to the boundary, and
- ◆ $\text{grad } u \cdot \mathbf{n}|_{\partial\Omega}$ the normal component trace of the displacement current.

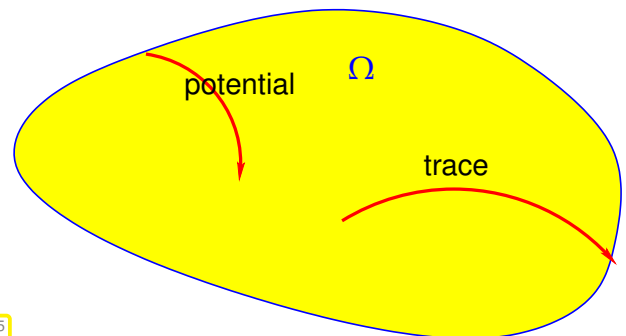


Fig. 15

Next, we examine the two layer potentials more closely. ┘

1.2.5.1 Single Layer Potential

The first layer potential occurring in (1.2.4.3), Thm. 1.2.4.2, involves the fundamental solution as **kernel**. Here, we call kernel a function $k : D_1 \times D_2 \rightarrow \mathbb{R}$ that defines an integral operator of the form

$$f \mapsto \left\{ x \mapsto \int_{D_2} k(x, y) f(y) dy \mid x \in D_1 \right\}, \tag{1.2.5.4}$$

mapping functions on D_2 to functions on D_1 . In this and in the next section we restrict ourselves to the differential operator $-\Delta$, but emphasize that all results carry over to operators L in general divergence form (1.2.0.1).

Definition 1.2.5.5. Single layer potential

The **single layer potential** for the Laplacian $-\Delta$ on $\partial\Omega$ is the mapping

$$\varphi \mapsto \{x \mapsto \Psi_{\text{SL}}^\Delta(\varphi)(x) := \int_{\partial\Omega} G^\Delta(x, y) \varphi(y) dS(y), x \notin \partial\Omega\} \quad (1.2.5.6)$$

We collect a few classical properties of Ψ_{SL} , see [Hac95, Sect. 8.1] for proofs using elementary calculus.


Theorem 1.2.5.7. Continuity of the single layer potential [Hac95, Sect. 8.1.2]

If $\varphi \in L^\infty(\partial\Omega)$, then $\Psi_{\text{SL}}^\Delta \in C^0(\mathbb{R}^d)$.

Proof. We recall the asymptotic behavior

$$G^\Delta(x, y) = \begin{cases} O(\log\|x - y\|) & , \text{ if } d = 2, \\ O(\|x - y\|^{-1}) & , \text{ if } d = 3, \end{cases} \quad \text{for } y \rightarrow x.$$

Note that $x \mapsto \log|x|$ is integrable on $[-1, 1]$ and $x \mapsto \|x\|^{-1}$ on $B_1(\mathbf{0}) \subset \mathbb{R}^2$. By means of piecewise smooth parameterizations we can reduce $\int_{\partial\Omega} \cdots dS(y)$ to integrals over domains in \mathbb{R}^{d-1} and the type of the singularities of the integrands will not change. Hence $\{y \mapsto G^\Delta(x, y) \varphi(y)\} \in L^1(\partial\Omega)$ with continuous dependence on x (as mapping $\mathbb{R}^d \rightarrow L^1(\partial\Omega)$). We conclude by appealing to general theorems about improper parameter dependent (Lebesgue) integrals. \square

 Notation: $L^\infty(D) \hat{=}$ space of (essentially) bounded functions on D , $C_{\text{pw}}^0(D) \subset L^\infty(D)$

$L^1(D) \hat{=}$ space of (improperly) integrable (in Lebesgue sense) functions on D

As a consequence, if Ψ_{SL}^Δ is evaluated for a piecewise polynomial function $\varphi : \partial\Omega \rightarrow \mathbb{R}$, it results in a globally continuous function.

Lemma 1.2.5.8. Smoothness of single layer potential

If $\varphi \in L^\infty(\partial\Omega)$ we have for every compact $D \subset \Omega$ or $D \subset \Omega' := \mathbb{R}^d \setminus \overline{\Omega}$ that

- (i) $\Psi_{\text{SL}}^\Delta(\varphi) \in C^\infty(D)$ (Ψ_{SL}^Δ is smooth away from $\partial\Omega$),
- (ii) $\Delta\Psi_{\text{SL}}^\Delta(\varphi) = 0$ on D (Ψ_{SL}^Δ is harmonic away from $\partial\Omega$).

Proof. This is a consequence that for any $x \notin \partial\Omega$ every derivative (w.r.t. x) of the integrand is integrable on $\partial\Omega$ (as a function of y). Thus, on D we can pull any derivative operator under the integral and the result will be a continuous function on D . \square

Finally, observe that Ψ_{SL}^Δ satisfies the **decay conditions** (1.1.7.1) and (1.1.7.4), respectively, e.g.,

$$\Psi_{\text{SL}}^\Delta(\varphi)(x) = \begin{cases} O(\log\|x\|) & , \text{ if } d = 2, \\ O(\|x\|^{-1}) & , \text{ if } d = 3. \end{cases} \quad \text{for } \|x\| \rightarrow \infty. \quad (1.2.5.9)$$

Remark 1.2.5.10 (Electrostatic interpretation of Ψ_{SL}) Comparing (1.2.5.6) and the formula (1.2.3.3) for the Newton potential

$$(\Psi_{\text{SL}}\varphi)(x) = \int_{\partial\Omega} G^\Delta(x, \mathbf{y}) \varphi(\mathbf{y}) \, dS(\mathbf{y}) \iff (N_{\Delta}\rho)(x) = \int_{\Omega} G^\Delta(x, \mathbf{y}) \rho(\mathbf{y}) \, d\mathbf{y},$$

we deduce that

the single layer potential is the Newton potential applied to a surface charge on $\partial\Omega$.

Recall the physical interpretation of the Newton potential $N_{\Delta}\rho$ as electrostatic scalar potential caused by the source charge distribution $\rho : \Omega \rightarrow \mathbb{R}$ on \mathbb{R}^d , This immediately suggests the following physical meaning of $\Psi_{\text{SL}}^\Delta\varphi$.

$\Psi_{\text{SL}}^\Delta\varphi$ is the electrostatic scalar potential induced by the **surface charge** φ on $\partial\Omega$.

┘

1.2.5.2 Double Layer Potential

Now we study the second potential (\rightarrow Notion 1.2.5.2) occurring in (1.2.4.3) and (1.2.4.6) (for the case of $L = -\Delta$). Refer to [Hac95, Sect. 8.2] for detailed proofs.

Definition 1.2.5.11. Double layer potential

For $u : \Gamma \rightarrow \mathbb{R}$ we define the **double layer potential** operator for the Laplacian $-\Delta$ by

$$\{x \mapsto \Psi_{\text{DL}}^\Delta(u)(x) := \int_{\partial\Omega} \mathbf{grad}_{\mathbf{y}} G^\Delta(x, \mathbf{y}) \cdot \mathbf{n}(\mathbf{y}) u(\mathbf{y}) \, dS(\mathbf{y}), x \notin \partial\Omega\} \quad (1.2.5.12)$$

For the kernel we can compute explicit formulas

$$d = 2: \quad G^\Delta(x, \mathbf{y}) = -\frac{1}{2\pi} \log\|x - \mathbf{y}\| \implies \mathbf{grad}_{\mathbf{y}} G^\Delta(x, \mathbf{y}) = \frac{1}{2\pi} \frac{x - \mathbf{y}}{\|x - \mathbf{y}\|^2}, \quad (1.2.5.13a)$$

$$\Psi_{\text{DL}}^\Delta(u)(x) = \int_{\partial\Omega} \frac{1}{2\pi} \frac{(x - \mathbf{y}) \cdot \mathbf{n}(\mathbf{y})}{\|x - \mathbf{y}\|^2} u(\mathbf{y}) \, dS(\mathbf{y}), \quad x \notin \partial\Omega,$$

$$d = 3: \quad G^\Delta(x, \mathbf{y}) = \frac{1}{4\pi\|x - \mathbf{y}\|} \implies \mathbf{grad}_{\mathbf{y}} G^\Delta(x, \mathbf{y}) = \frac{1}{4\pi} \frac{x - \mathbf{y}}{\|x - \mathbf{y}\|^3}, \quad (1.2.5.13b)$$

$$\Psi_{\text{DL}}^\Delta(u)(x) = \int_{\partial\Omega} \frac{1}{4\pi} \frac{(x - \mathbf{y}) \cdot \mathbf{n}(\mathbf{y})}{\|x - \mathbf{y}\|^3} u(\mathbf{y}) \, dS(\mathbf{y}), \quad x \notin \partial\Omega.$$

§1.2.5.14 (Continuity of double layer potential) Since $\mathbf{grad}_{\mathbf{y}} G^\Delta(x, \mathbf{y}) = O(\|x - \mathbf{y}\|^{-d+1})$ for $\mathbf{y} \rightarrow x$, which is a non-integrable singularity in dimension $d - 1$, the mapping $x \mapsto \{\mathbf{y} \mapsto \mathbf{grad}_{\mathbf{y}} G^\Delta(x, \mathbf{y}) \cdot \mathbf{n}(\mathbf{y})\}$ fails to be a continuous mapping into $L^1(\partial\Omega)$. So we cannot conclude global continuity of $\Psi_{\text{DL}}^\Delta u$ regardless of the smoothness of u .

In fact, if Ω is bounded, for $u \equiv \mathbb{1}$, the constant function $\mathbb{1} := \{\mathbf{y} \in \partial\Omega \rightarrow 1\}$, from Gauss theorem

$$\int_{\partial\Omega} \mathbf{grad} w(x) \cdot \mathbf{n}(x) \, dx = \int_{\Omega} \text{div} \mathbf{grad} w(x) \, dx, \quad w \in C_{\text{pw}}^1(\overline{\Omega}),$$

$$\begin{aligned}
 (\Psi_{\text{DL}}^\Delta \mathbb{1})(x) &= \int_{\partial\Omega} \mathbf{grad}_y G^\Delta(x, y) \cdot \mathbf{n}(y) \, dS(y) = \int_{\Omega} \Delta_y G^\Delta(x, y) \, dx \\
 &= \begin{cases} 0 & , \text{ if } x \notin \overline{\Omega} , \\ \text{“ } - \int_{\Omega} \delta_x \, dx \text{”} = -1 & , \text{ if } x \in \Omega . \end{cases}
 \end{aligned}$$

► $\Psi_{\text{DL}}^\Delta \mathbb{1} = \begin{cases} 0 & \text{ in } \Omega' := \mathbb{R}^d \setminus \overline{\Omega} , \\ -1 & \text{ in } \Omega \end{cases}$ is piecewise constant with a jump across $\partial\Omega$. ┘

Concerning smoothness away from $\partial\Omega$, the double layer potentials enjoy properties similar to those of the single layer potentials, with analogous proofs.

Lemma 1.2.5.15. Smoothness of double layer potential

If $\varphi \in L^1(\partial\Omega)$ we have for every compact $D \subset \Omega$ or $D \subset \Omega'$ that

- (i) $\Psi_{\text{DL}}^\Delta(\varphi) \in C^\infty(D)$ (Ψ_{DL}^Δ is smooth away from $\partial\Omega$),
- (ii) $\Delta \Psi_{\text{DL}}^\Delta(\varphi) = 0$ on D (Ψ_{DL}^Δ is harmonic away from $\partial\Omega$).

Remark 1.2.5.16 (Electrostatic meaning of Ψ_{DL}^Δ) Assume that $\Gamma := \partial\Omega$ is smooth with exterior unit normal vector field \mathbf{n} . Then, formally, for $\mathbf{y} \in \Gamma$

$$\mathbf{grad}_y G^\Delta(x, y) \cdot \mathbf{n}(y) = \lim_{\epsilon \rightarrow 0} \frac{G^\Delta(x, y + \epsilon \mathbf{n}(y)) - G^\Delta(x, y - \epsilon \mathbf{n}(y))}{2\epsilon} .$$

Hence, the double layer kernel models the potential of two unit charges of opposite sign at an infinitesimally small distance, an arrangement known as **electric dipole**. The double layer potential could also be called a dipole layer. ┘

1.2.6 Green’s Functions

We consider a bounded domain and a general scalar linear second-order differential operator L as in (1.2.0.1). We study generalized fundamental solutions that also satisfy boundary conditions.

Definition 1.2.6.1. Green’s function

A function $G_\Omega^L : \Omega \times \Omega \rightarrow \mathbb{R}$ is a **Green’s function** for a second-order scalar linear differential operator L on a bounded domain $\Omega \subset \mathbb{R}^d$, if

- (i) G_Ω^L is C^∞ -smooth on $\{(x, y) \in \Omega \times \Omega : x \neq y\}$,
- (ii) for all $x \in \Omega$: $L_y G_\Omega^L(x, y) = 0$ on $\Omega \setminus \{x\}$
- (iii) G_Ω^L satisfies *homogeneous Dirichlet boundary conditions*:

$$G_\Omega^L(x, y) = 0 \quad \text{for all } y \in \partial\Omega, x \in \Omega, \tag{1.2.6.2}$$

- (iv) $y \mapsto G_\Omega^L(x, y)$ is integrable on Ω for all $x \in \Omega$,
- (v) for every $x \in \Omega, w \in C^\infty(\overline{\Omega})$

$$\int_{\Omega} G_\Omega^L(x, y) (L^* w)(y) \, dy = w(x) . \tag{1.2.6.3}$$

We can rewrite Item (v) by means of distributional calculus as

$$L_y G_\Omega^L(x, y) = \delta_x \text{ in } \mathcal{D}(\Omega)', \tag{1.2.6.4}$$

see Rem. 1.2.2.18.

Now we can pursue the same manipulations as in Section 1.2.4 for the model case of $L = -\Delta$. We choose any $x \in \Omega$ and $w \in C^2(\bar{\Omega})$. Appealing to Green's first formula from Thm. 1.1.6.1 we get

$$\begin{aligned} \int_\Omega G_\Omega^\Delta(x, y)(-\Delta w) \, dy &= \lim_{\epsilon \rightarrow 0} \int_{\|y\| > \epsilon} G_\Omega^\Delta(x, y)(-\Delta w)(y) \, dy \\ &= - \lim_{\epsilon \rightarrow 0} \int_{\|y-x\|=\epsilon} G_\Omega^\Delta(x, y) \mathbf{grad} w(y) \cdot \mathbf{n}(y) - w(y) \mathbf{grad}_y G_\Omega^\Delta(x, y) \cdot \mathbf{n}(y) \, dS(y) + \\ &\quad - \int_{\partial\Omega} G_\Omega^\Delta(x, y) \mathbf{grad} w(y) \cdot \mathbf{n}(y) - w(y) \mathbf{grad}_y G_\Omega^\Delta(x, y) \cdot \mathbf{n}(y) \, dS(y) \\ &= w(x) - \int_{\partial\Omega} G_\Omega^\Delta(x, y) \mathbf{grad} w(y) \cdot \mathbf{n}(y) - w(y) \mathbf{grad}_y G_\Omega^\Delta(x, y) \cdot \mathbf{n}(y) \, dS(y), \end{aligned}$$

thanks to Item (iii). This yields a simplified integral representation formula compared to Thm. 1.2.4.2. For $w \in C^2(\bar{\Omega})$

$$w(x) = \int_\Omega G_\Omega^\Delta(x, y)(-\Delta w)(y) \, dy - \int_{\partial\Omega} \mathbf{grad}_y G_\Omega^\Delta(x, y) \cdot \mathbf{n}(y) w(y) \, dS(y). \tag{1.2.6.5}$$

Corollary 1.2.6.6. Green's function integral representations

◆ If $u \in C^2(\bar{\Omega})$ solves the boundary value problem

$$\begin{aligned} -\Delta u &= \rho \in C^0(\bar{\Omega}) \text{ in } \Omega, \quad u = 0 \text{ on } \partial\Omega, \\ \blacktriangleright \quad u(x) &= \int_\Omega G_\Omega^\Delta(x, y) \rho(y) \, dy, \quad x \in \Omega. \end{aligned} \tag{1.2.6.7}$$

◆ If $u \in C^2(\bar{\Omega})$ solves the Dirichlet boundary value problem

$$\begin{aligned} -\Delta u &= 0 \text{ in } \Omega, \quad u = g \in C^0(\partial\Omega) \text{ on } \partial\Omega, \\ \blacktriangleright \quad u(x) &= - \int_{\partial\Omega} \mathbf{grad}_y G_\Omega^\Delta(x, y) \cdot \mathbf{n}(y) g(y) \, dS(y), \quad x \in \Omega. \end{aligned} \tag{1.2.6.8}$$

Comparing with (1.2.3.9), we notice that the integral operator

$$\rho \mapsto \left\{ x \mapsto \int_\Omega G_\Omega^\Delta(x, y) \rho(y) \, dy \right\}$$

is the solution operator for the Dirichlet boundary value problem $-\Delta u = \rho$ in Ω , $u = 0$ on $\partial\Omega$.

Green's functions remain elusive for general domains Ω . Only for very special geometries and simple operators like $-\Delta$ they can be computed in closed form. Next we give an example.

EXAMPLE 1.2.6.9 (Green's function for $-\Delta$ on a disk) We compute the Green's function (\rightarrow Def. 1.2.6.1) for $-\Delta$, $d = 2$, and the unit disk domain $\Omega = \mathbb{D} := \{x \in \mathbb{R}^2 : \|x\| < 1\}$.

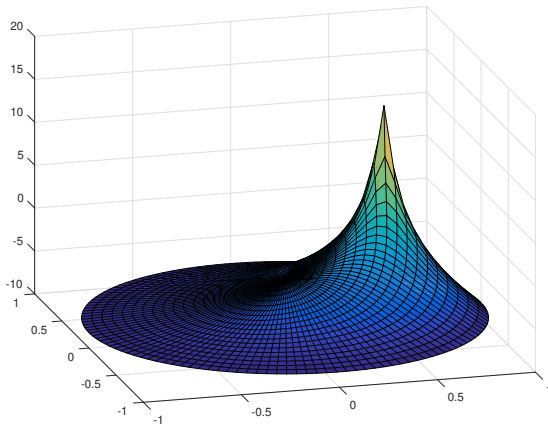


Fig. 16

The derivation is based on the **mirror charge approach** and reflection at the unit circle. For $x \in \mathbb{R}^2$ write $x^* := x/\|x\|^2$, that is $\|x^*\| = \frac{1}{\|x\|}$ and $x^* \notin \mathbb{D}$ for $x \in \mathbb{D}$. We fix $x \in \mathbb{D}$ and place a unit charge at x and a compensating charge at $x^* \notin \mathbb{D}$, which yields the total potential, cf (1.2.2.10),

$$G_{\mathbb{D}}^{\Delta}(x, y) = -\frac{1}{2\pi} \log\|x - y\| + \frac{1}{2\pi} \log\|x^* - y\| - \frac{1}{2\pi} \log\|x^*\|. \quad (1.2.6.10)$$

◁ Plot of $y \mapsto G_{\mathbb{D}}^{\Delta}$, $x = \begin{bmatrix} 0.5 \\ 0 \end{bmatrix}$

For $x \in \mathbb{D}$, thanks to $x^* \notin \mathbb{D}$, the properties Item (i), Item (ii), Item (iv), and Item (v) are all inherited from the first term, which is the fundamental solution for $-\Delta$ in 2D, see (1.2.2.10), and the only term with a singularity in \mathbb{D} . To see Item (iii) note that for $\|y\| = 1$, we have for all $x \in \mathbb{D}$

$$\begin{aligned} G_{\mathbb{D}}^{\Delta}(x, y) &= \frac{1}{4\pi} \log\left(\frac{\|y - x^*\|^2}{\|y - x\|^2 \|x^*\|^2}\right) = \frac{1}{4\pi} \log\left(\frac{\|x\|^2 \|y - x^*\|^2}{\|y - x\|^2}\right) \\ &= -\frac{1}{4\pi} \log\left(\frac{1 - 2x \cdot y + \|x\|^2}{\|x\|^2 (1 - 2\frac{1}{\|x\|^2} x \cdot y + \frac{1}{\|x\|^2})}\right) = -\frac{1}{4\pi} \log(1) = 0. \end{aligned}$$

┘

Remark 1.2.6.11 (Poisson integral formula [Hac92, Thm. 2.20]) The Green's function (1.2.6.10) combined with Cor. 1.2.6.6, Section 1.2.6, we get an explicit integral representation for solutions of

$$\begin{aligned} &-\Delta u = 0 \text{ in } \mathbb{D}, \quad u = g \text{ on } \partial\mathbb{D} := \{x \in \mathbb{R}^2 : \|x\| = 1\}, \\ \blacktriangleright \quad &u(x) = \frac{1 - \|x\|^2}{2\pi} \int_{\|y\|=1} \frac{1}{\|x - y\|^2} g(y) \, dS(y), \quad x \in \mathbb{D}. \end{aligned} \quad (1.2.6.12)$$

┘

EXAMPLE 1.2.6.13 (Green's function for a half space)

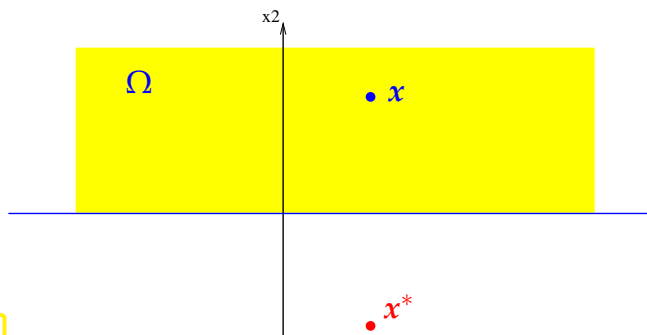


Fig. 17

In Def. 1.2.6.1 we assumed a bounded Ω , but Green's functions can easily be generalized to non-bounded domains by simply keeping all the requirements Item (i)–Item (v), demanding compact support of w in the latter. For instance, relying on another **mirror charge approach** for the **half space** $\Omega := \{x \in \mathbb{R}^2 : x_2 > 0\}$ we find

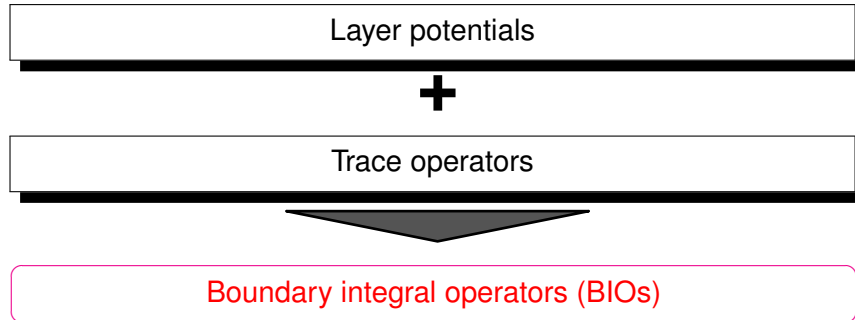
$$G_{\Omega}^{\Delta}(x, y) = -\frac{1}{2\pi} \log\|x - y\| + \frac{1}{2\pi} \log\|x^* - y\|, \quad x^* = \begin{bmatrix} x_1 \\ -x_2 \end{bmatrix}, \quad x, y \in \Omega. \quad (1.2.6.14)$$

┘

1.3 Boundary Integral Equations (BIEs)

Throughout this section we consider a Lipschitz domain $\Omega \subset \mathbb{R}^d$ satisfying Ass. 1.2.1.5 for $d = 2$ or Ass. 1.2.1.7 for $d = 3$. We write $\Gamma := \partial\Omega$ for its (compact) boundary and \mathbf{n} for the exterior unit normal vector field on Γ .

§1.3.0.1 (Outline) Trace operators (\rightarrow Notion 1.2.5.1) when applied potentials (\rightarrow Notion 1.2.5.2) yield linear mappings taking functions on Γ to other functions on Γ :



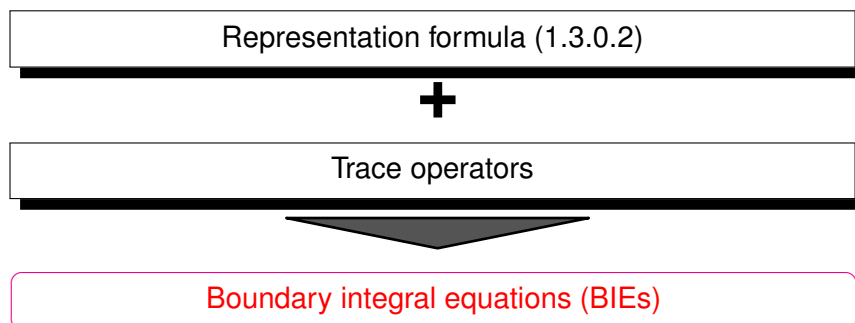
In particular, we may apply trace operators to layer potential representations formulas for solutions of second-order scalar PDEs with vanishing source terms, like those given in Thm. 1.2.4.2 and Thm. 1.2.4.5 for $\rho = 0$ (crucial traces of u highlighted, cf. Rem. 1.2.5.3):

$$u(x) = \int_{\Gamma} G^L(x, \mathbf{y}) \mathbf{A} \mathbf{grad} u(\mathbf{y}) \cdot \mathbf{n}(\mathbf{y}) \, dS(\mathbf{y}) - \int_{\Gamma} \mathbf{A} \mathbf{grad}_{\mathbf{y}} G^L(x, \mathbf{y}) \cdot \mathbf{n}(\mathbf{y}) u(\mathbf{y}) \, dS(\mathbf{y}), \quad (1.3.0.2)$$

for $x \in \Omega$, where $u \in C^2(\overline{\Omega})$ solves $Lu := -\operatorname{div}(\mathbf{A} \mathbf{grad} u) - cu = 0$ in Ω , \mathbf{A}, c as in (1.2.0.1), and G^L is the fundamental solution for L , see Def. 1.2.2.15. We point out that using our notations for the layer potentials, a compact way to write (1.2.4.3) is

$$u(x) = \Psi_{SL}^L(\mathbf{A} \mathbf{grad} u(\mathbf{y}) \cdot \mathbf{n}(\mathbf{y})|_{\Gamma})(x) - \Psi_{DL}^L(u|_{\Gamma})(x), \quad x \in \Omega. \quad (1.3.0.2)$$

Applying trace operators we should end up with equations linking the traces $u|_{\Gamma}$ and $\mathbf{A} \mathbf{grad} u(\mathbf{y}) \cdot \mathbf{n}(\mathbf{y})|_{\Gamma}$. One of these must be known in the case of well-defined boundary value problems, and we hope to determine the other through the obtained equations.



However, we have to ensure that trace operators can be applied to layer potentials. Adhering to an “energy-centric” approach, we investigate the continuity of the operators in energy norms. \lrcorner

1.3.1 Trace Operators

Notion 1.2.5.1 tells us that trace operators map functions on the volume domain Ω to functions on the boundary Γ . Now examine the continuity properties in energy norms of the two trace operators relevant for boundary value problems for the Laplacian $-\Delta$.

1.3.1.1 Dirichlet Trace

Definition 1.3.1.1. Dirichlet trace operator

The **Dirichlet trace (operator)** T_D boils down to pointwise restriction for smooth functions:

$$(T_D w)(x) := w(x) \quad \forall x \in \Gamma, \quad w \in C^\infty(\bar{\Omega}).$$

Though obvious, we stress the fact that T_D maps functions $\bar{\Omega} \mapsto \mathbb{R}$ to functions $\Gamma \mapsto \mathbb{R}$. Also not that, if Γ is merely piecewise smooth, even $w \in C^\infty(\bar{\Omega})$ does imply only $T_D w \in C^0(\Gamma)$!

§1.3.1.2 (An energy space for point traces of scalar potentials) Our goal is to extend the Dirichlet trace T_D to the energy space $H^1(\Omega)$ and to identify the strongest norm on $C^0(\Gamma)$ that will still render T_D continuous. Completion (\rightarrow [NumPDE § 1.3.3.10]) with respect to this norm will yield a suitable **trace space**, serving range space of $T_D|_{H^1(\Omega)}$.

Let $\|\cdot\|_X$ stand for a norm on $C^0(\Gamma)$. Recall that T_D is **continuous** with respect to this norm, if

$$\exists C > 0: \quad \|T_D u\|_X \leq C \|u\|_{H^1(\Omega)} \quad \forall u \in C^\infty(\bar{\Omega}). \quad (1.3.1.3)$$

A norm is dubbed “stronger” than another norm on the same space, if (up to a constant) it assigns larger norm values to every element of the space than this other norm. The *strongest possible norm* $\|\cdot\|_X$ on $C^0(\Gamma)$ for which we can still expect the continuity (1.3.1.3) can formally be *defined* as follows

$$\|u\|_X := \inf \left\{ \|v\|_{H^1(\Omega)} : v \in C^\infty(\bar{\Omega}), T_D v = u \right\}, \quad u \in C^\infty(\bar{\Omega})|_\Gamma. \quad (1.3.1.4)$$

The reader is encouraged to verify the norm axioms from [NumPDE Def. 0.3.1.10] for this $\|\cdot\|_X$.


Remark 1.3.1.5 (Density argument) A fundamental result in the theory of Sobolev spaces [McL00, Thm. 3.25] ensures the **density** of $C^\infty(\bar{\Omega})$ in $H^1(\Omega)$. Therefore, when studying T_D on $H^1(\Omega)$, it is sufficient to consider $T_D|_{C^\infty(\bar{\Omega})}$. Recall the advice [NumPDE Section 1.3.4] that one should focus on norms in the study of Sobolev spaces and not worry about the smoothness of the functions too much. \square

It is easy to establish that (1.3.1.4) defines a norm. In fact, $\|\cdot\|_X$ is derived from an inner product. Completion then yields the right trace space.

Definition 1.3.1.6. Dirichlet trace space

The **Dirichlet trace space** $H^{\frac{1}{2}}(\Gamma)$ is the Hilbert space obtained by completion of $C^\infty(\bar{\Omega})|_\Gamma$ with respect to the *energy norm*

$$\|u\|_{H^{\frac{1}{2}}(\Gamma)} := \inf \left\{ \|v\|_{H^1(\Omega)} : v \in C^\infty(\bar{\Omega}), T_D v = u \right\}, \quad u \in C^\infty(\bar{\Omega})|_\Gamma. \quad (1.3.1.7)$$

 Notation: We write u, v, w for functions in $H^{\frac{1}{2}}(\Gamma)$.

For mathematicians familiar with functional analysis the next result is an immediate consequence of Def. 1.3.1.6, thus labelled a corollary. A reader not well versed in functional analysis may just accept it as a fact.

Corollary 1.3.1.8. Mapping properties of Dirichlet trace [SS10, Sect. 2.6]

The Dirichlet trace T_D according to Def. 1.3.1.1 can be extended to a **continuous** and **surjective** linear operator $T_D : H^1(\Omega) \rightarrow H^{\frac{1}{2}}(\Gamma)$.

In the title of this § $H^{\frac{1}{2}}(\Gamma)$ was said to be an “energy space”. To see the connection look up the equilibrium condition (1.1.4.7) again to understand that the minimizer $w \in H^1(\Omega)$ of the expression in (1.3.1.7) agrees with the weak solution of the Dirichlet boundary value problem

$$-\Delta w = 0 \text{ in } \Omega, \quad w = u \text{ on } \Gamma.$$

In an electrostatic context this is the potential arising in the volume when imposing the potential values u on Γ . Hence, we arrive at the following “physical interpretation” of $\|\cdot\|_{H^{\frac{1}{2}}(\Gamma)}$

$\|u\|_{H^{\frac{1}{2}}(\Gamma)}$ is the electric field energy in Ω due to imposing the potential values u on Γ .

┘

§1.3.1.10 (Smoothness (“regularity”) of functions in $H^{\frac{1}{2}}(\Gamma)$) Def. 1.3.1.6 does not yield much insight into $H^{\frac{1}{2}}(\Gamma)$. To understand properties of functions in $H^{\frac{1}{2}}(\Gamma)$ we recall a first result on continuity properties of T_D [NumPDE Thm. 1.9.0.10].

Theorem 1.3.1.11. Multiplicative trace inequality [BS08, Thm. 1.6.6]

$$\exists C = C(\Omega) > 0: \quad \|u\|_{L^2(\Gamma)}^2 \leq C \|u\|_{L^2(\Omega)} \cdot \|u\|_{H^1(\Omega)} \quad \forall u \in H^1(\Omega).$$

Proof.

We demonstrate the proof only for domains Ω with $\text{diam } \Omega = 1$ that are **star-shaped** w.r.t. a ball $B_r(\mathbf{0})$, $0 < r < 1$, that is,

$$\forall y \in B_r(\mathbf{0}), x \in \Omega: \quad [y, x] \subset \Omega.$$

In this case

$$n(x) \cdot x \geq C_\Omega, \tag{1.3.1.12}$$

for a constant $C_\Omega > 0$.

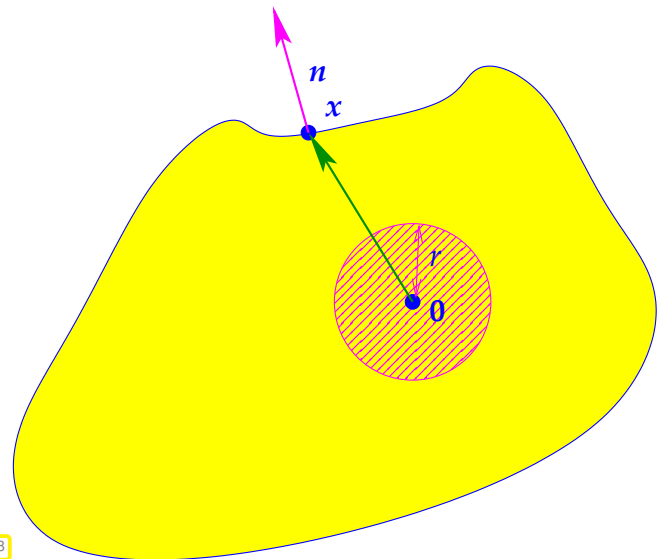


Fig. 18

Gauss’ theorem and the product rule show for $u \in C^2(\bar{\Omega})$

$$\begin{aligned} \int_\Gamma |u(x)|^2 dS(x) &\leq C_\Omega^{-1} \int_\Gamma n \cdot x |u(x)|^2 dS(x) = C_\Omega^{-1} \int_\Omega \text{div}(x|u(x)|^2) dx \\ &= C_\Omega^{-1} \int_\Omega d|u(x)|^2 + 2ux \cdot \text{grad } u \, dx \\ &\leq C_\Omega^{-1} \left(d\|u\|_{L^2(\Omega)}^2 + 4\|u\|_{L^2(\Omega)} \|u\|_{H^1(\Omega)}^2 \right), \end{aligned}$$

where we used $\|x\| \leq 2$. A density argument (\rightarrow Rem. 1.3.1.5) as in the proof of [NumPDE Thm. 1.3.4.17] establishes the claim. □

The next statement is labelled a corollary, that is, considered “obvious”. The reader should be able to conclude it from Def. 1.3.1.6 and Thm. 1.3.1.11 instantly.

Corollary 1.3.1.13. Embedding of $H^{\frac{1}{2}}(\Gamma)$

The space $H^{\frac{1}{2}}(\Gamma)$ is continuously embedded in $L^2(\Gamma)$: $\|u\|_{L^2(\Gamma)} \leq C \|u\|_{H^{\frac{1}{2}}(\Gamma)}$ for all $u \in H^{\frac{1}{2}}(\Gamma)$ and some $C > 0$ independent of u .

EXAMPLE 1.3.1.14 (“Continuity” of functions in $H^{\frac{1}{2}}(\Gamma)$) How smooth are functions in $H^{\frac{1}{2}}(\Gamma)$? For the Sobolev space $H^1(\Omega)$ we already asked this question in [NumPDE § 1.3.4.25].

We consider the unit disk domain $\Omega = \mathbb{D} := \{x \in \mathbb{R}^2 : \|x\| < 1\}$ and, in polar coordinates (r, φ) , the Fourier sums

$$g_n(\varphi) := \frac{4}{\pi} \sum_{k=1}^n \frac{1}{2k-1} \sin((2k-1)\varphi) \in L^2(\Gamma), \quad n \in \mathbb{N}.$$

The solutions of

$$-\Delta u_n = 0 \quad \text{in } \mathbb{D}, \quad \mathbb{T}_D u_n = g_n \quad \text{on } \Gamma,$$

are

$$u_n(r, \varphi) = \frac{4}{\pi} \sum_{k=1}^n \frac{1}{2k-1} r^{2k-1} \sin((2k-1)\varphi), \quad 0 \leq r < 1, \quad 0 \leq \varphi < 2\pi.$$

This is a consequence of the fact that $\Delta\{r^\ell \sin(\ell\varphi)\} = 0$.

By (1.3.1.7) the energy norm of u_n is equivalent to the trace norm of g_n :

$$\|u_n\|_{H^1(\Omega)} \approx \|g_n\|_{H^{\frac{1}{2}}(\Gamma)} \quad \text{with “universal constants”}.$$

From the theory of [Fourier series](#) we know

$$\lim_{n \rightarrow \infty} g_n = g \quad \text{in } L^2(\Gamma), \quad g(\varphi) = \begin{cases} -1 & \text{for } -\pi \leq \varphi \leq 0, \\ 1 & \text{for } 0 < \varphi \leq \pi, \end{cases}$$

that is the limit of the sequence $(g_n)_{n \in \mathbb{N}}$ is a piecewise constant, **discontinuous** function.

Termwise differentiation gives

$$\mathbf{grad} u_n(r, \varphi) = \sum_{k=1}^n r^{2k-2} (\sin((2k-1)\varphi) \mathbf{e}_r(r, \varphi) + \cos((2k-1)\varphi) \mathbf{e}_\varphi(r, \varphi)),$$

where $\{\mathbf{e}_r, \mathbf{e}_\varphi\}$ is the polar coordinate orthonormal basis, see [NumPDE ??].

$$\begin{aligned} \blacktriangleright \|\mathbf{grad} u_n\|_{L^2(\Omega)}^2 &= \sum_{k=1}^n \int_0^1 \int_0^{2\pi} r^{4k-4} \sin^2((2k-1)\varphi) + \cos^2((2k-1)\varphi) \, d\varphi \, r \, dr \\ &= \sum_{k=1}^n \frac{2\pi}{4k-3} \rightarrow \infty \quad \text{for } n \rightarrow \infty. \end{aligned}$$

As a consequence, “ $\|g\|_{H^{\frac{1}{2}}(\Gamma)} = \infty$ ”, $g \notin H^{\frac{1}{2}}(\Gamma)$.

$$C_{pw}^0(\Gamma) \not\subset H^{\frac{1}{2}}(\Gamma) !$$

EXAMPLE 1.3.1.15 (Unbounded functions in $H^{\frac{1}{2}}(\Gamma)$) According to [NumPDE Cor. 1.3.4.7] the point evaluation functional $u \mapsto u(\mathbf{y})$, $\mathbf{y} \in \Omega$, is **not** bounded on $H^1(\Omega)$ for $d \geq 2$; there are unbounded functions in $H^1(\Omega)$ and in [NumPDE Ex. 1.2.3.45] we found an example in 2D

$$v(x) = \log |\log \|x\||, \quad \|x\| < \frac{1}{2}, \quad v \in H^1(B_{\frac{1}{2}}(\mathbf{0})). \quad (1.3.1.16)$$

If $\mathbf{0} \in \Gamma$, $T_D v \in H^{\frac{1}{2}}(\Gamma)$ will **not** be **bounded**!

As a positive result we note that continuous, piecewise smooth functions belong to $H^{\frac{1}{2}}(\Gamma)$, because they are already contained in $H^1(\Omega)$. Compare with Ex. 1.3.1.14.

Corollary 1.3.1.17. Continuous, piecewise- C^1 functions in $H^{\frac{1}{2}}(\Gamma)$

$$C_{pw}^1(\Gamma) \subset H^{\frac{1}{2}}(\Gamma).$$

In words, piecewise smooth bounded functions $\Gamma \mapsto \mathbb{R}$ belong to $H^{\frac{1}{2}}(\Gamma)$, if and only if they are continuous: for them belonging to $H^{\frac{1}{2}}(\Gamma)$ entails the same compatibility conditions as for $H^1(\Omega)$, remember Thm. 1.1.3.5.

Remark 1.3.1.18 (Intrinsic norm of $H^{\frac{1}{2}}(\Gamma)$) As a consequence of *extension theorems* for $H^1(\Omega)$ [McL00, Appendix A], Def. 1.3.1.6 yields equivalent norms for $H^{\frac{1}{2}}(\Gamma)$, no matter whether we base the definition of $\|\cdot\|_{H^{\frac{1}{2}}(\Gamma)}$ on Ω or Ω' .

In fact, from [SS10, Def. 2.4.1] we learn, that there is an equivalent Γ -intrinsic definition

$$\|u\|_{H^{\frac{1}{2}}(\Gamma)}^2 \approx \|u\|_{L^2(\Gamma)}^2 + \int_{\Gamma} \int_{\Gamma} \frac{|u(\mathbf{x}) - u(\mathbf{y})|^2}{\|\mathbf{x} - \mathbf{y}\|^d} dS(\mathbf{y}) dS(\mathbf{x}), \quad u \in H^{\frac{1}{2}}(\Gamma). \quad (1.3.1.19)$$

This expression known as the **Sobolev-Slobodeckii** norm.

1.3.1.2 Neumann Trace

Now we take a closer look at the normal component trace of the displacement current, in non-dimensional form $\mathbf{grad} u \cdot \mathbf{n}|_{\Gamma}$.

Definition 1.3.1.20. Neumann trace operator

For smooth functions the **Neumann trace (operator)** T_N is defined by

$$(T_N w)(x) := \mathbf{grad} w \cdot \mathbf{n}(x) \quad \forall x \in \Gamma, w \in C^\infty(\bar{\Omega}).$$

Remark 1.3.1.21 (The Neumann trace is not defined on $H^1(\Omega)$) We consider $d = 2$, $\Omega = \mathbb{D} := \{x \in \mathbb{R}^2 : \|x\| < 1\}$, and (in polar coordinates (r, φ) , see [NumPDE ??]) the functions

$$u_n(r, \varphi) := n^{-1}r^n, \quad n \in \mathbb{N}.$$

Then $\mathbf{grad} u_n(r, \varphi) = r^{n-1}e_r$ and we find by simply computing the norms in polar coordinates

$$\|u_n\|_{H^1(\Omega)} \rightarrow 0 \quad \text{for } n \rightarrow \infty \quad \text{whereas } T_N u_n = \mathbf{1} \quad \text{on } \partial\mathbb{D}.$$

The message sent by this example is similar to the insight gained in [NumPDE Ex. 1.3.2.5]:

The Neumann trace T_N is not bounded on $H^1(\Omega)$.

In other words, Neumann boundary conditions cannot be imposed in $H^1(\Omega)$, analogous to the situation with the Dirichlet trace and $L^2(\Omega)$ as discussed in [NumPDE Ex. 1.3.2.5]. ┘

Remark 1.3.1.22 (Pairing of traces [SS10, Thm. 2.7.7]) It is a straightforward consequence of Green's first formula from Thm. 1.1.6.1 (with $\mathbf{j} := \mathbf{grad} u$) that

$$\int_{\Gamma} (T_N u)(x) (T_D v)(x) dS(x) = \int_{\Omega} \Delta u(x) v(x) + \mathbf{grad} u(x) \cdot \mathbf{grad} v(x) dx, \quad (1.3.1.23)$$

for all $u, v \in C^\infty(\bar{\Omega})$. If u is **harmonic**, that is $\Delta u = 0$, then

$$\int_{\Gamma} (T_N u)(x) (T_D v)(x) dS(x) = \int_{\Omega} \mathbf{grad} u(x) \cdot \mathbf{grad} v(x) dx. \quad (1.3.1.24)$$

In particular, we conclude that for any harmonic function $u \in H^1(\Omega)$ (solving $\Delta u = 0$), the paired Dirichlet and Neumann traces yield the function's energy:

$$\int_{\Gamma} (T_N u)(x) (T_D u)(x) dS(x) = \int_{\Omega} \|\mathbf{grad} u(x)\|^2 dx. \quad (1.3.1.25)$$

§1.3.1.26 (An energy norm for Neumann traces) From electrostatic theory we know that the normal component trace of the displacement current at a PEC boundary part corresponds to a **surface charge distribution**. ┘

► The range space of the Neumann trace operator T_N , the **Neumann trace space** is a space of surface charge distributions.



Define a norm on the Neumann trace space through the energy of the field induced by surface charge distribution.

Definition 1.3.1.27. Neumann trace space

The **Neumann trace space** $H^{-\frac{1}{2}}(\Gamma)$ is the Hilbert space obtained by the completion of $C^0(\Gamma)$ with respect to the norm

$$\|\phi\|_{H^{-\frac{1}{2}}(\Gamma)} := \|\tilde{\phi}\|_{\tilde{H}^{-1}(\Omega)}, \quad (1.3.1.28)$$

where $\|\cdot\|_{\tilde{H}^{-1}(\Omega)}$ is the norm on source charge distributions introduced in Def. 1.1.8.1 and $\tilde{\phi}$ is the "extension by zero to \mathbb{R}^d " of ϕ .

Temporarily, we restrict ourselves to $d = 3$. Given $\phi \in C^0(\Gamma)$ we define $u_\phi \in H^1(\mathbb{R}^3)$ through

$$\int_{\mathbb{R}^3} \mathbf{grad} u_\phi \cdot \mathbf{grad} v \, dx = \int_{\Gamma} \phi(\mathbf{x}) (\mathbb{T}_D v)(\mathbf{x}) \, dS(\mathbf{x}) \quad \forall v \in H^1(\mathbb{R}^3). \quad (1.3.1.29)$$

By virtue of (1.1.8.5) [SS10, Prop 2.10.8], the bilinear form of this variational problem is $H^1(\mathbb{R}^3)$ -elliptic and, thus, unique solvability is guaranteed. Then, from the definition of $\|\cdot\|_{\tilde{H}^{-1}(\Omega)}$ is immediate that

$$\|\phi\|_{H^{-\frac{1}{2}}(\Gamma)} = \|\tilde{\phi}\|_{\tilde{H}^{-1}(\Omega)} = \|\mathbf{grad} u_\phi\|_{L^2(\mathbb{R}^3)}. \quad (1.3.1.30)$$

With this in mind, in perfect analogy to § 1.3.1.2 we can also link the norm on $H^{-\frac{1}{2}}(\Gamma)$ to the energy norm of fields/potentials:

$\|\phi\|_{H^{-\frac{1}{2}}(\Gamma)}$ is the energy of the electric field engendered by the surface charge distribution ϕ .

┘

§1.3.1.31 (Continuity of Neumann trace) We have seen in Rem. 1.3.1.21 that the Neumann trace \mathbb{T}_N is not defined on $H^1(\Omega)$; we need a function space with a stronger norm, on which we can then define \mathbb{T}_N as a continuous linear operator.

Definition 1.3.1.32. Space of function with square-integrable Laplacian

We introduce the Hilbert space

$$H(\Delta, \Omega) := \{v \in H^1(\Omega) : \Delta v \in L^2(\Omega)\},$$

with norm

$$\|u\|_{H(\Delta, \Omega)}^2 := \|u\|_{H^1(\Omega)}^2 + \|\Delta u\|_{L^2(\Omega)}^2, \quad u \in H(\Delta, \Omega).$$

Theorem 1.3.1.33. Continuity of the Neumann trace on $H(\Delta, \Omega)$

The Neumann trace \mathbb{T}_N from Def. 1.3.1.20 can be extended to a continuous mapping

$$\mathbb{T}_N : H(\Delta, \Omega) \rightarrow H^{-\frac{1}{2}}(\Gamma).$$

Proof. Given $w \in C^\infty(\bar{\Omega})$ define $u_w \in H^1(\Omega)$ through

$$\int_{\mathbb{R}^3} \mathbf{grad} u_w \cdot \mathbf{grad} v \, dx = \int_{\Gamma} (\mathbb{T}_N w)(\mathbf{x}) (\mathbb{T}_D v)(\mathbf{x}) \, dS(\mathbf{x}) \quad \forall v \in H^1(\mathbb{R}^3). \quad (1.3.1.34)$$

Recall from § 1.3.1.26 that $\|\mathbb{T}_N w\|_{H^{-\frac{1}{2}}(\Gamma)} = \|\mathbf{grad} u_w\|_{L^2(\mathbb{R}^3)}$. Then use the pairing identity

$$\int_{\Gamma} (\mathbb{T}_N u)(\mathbf{x}) (\mathbb{T}_D v)(\mathbf{x}) \, dS(\mathbf{x}) = \int_{\Omega} \Delta u(\mathbf{x}) v(\mathbf{x}) + \mathbf{grad} u(\mathbf{x}) \cdot \mathbf{grad} v(\mathbf{x}) \, dx, \quad (1.3.1.23)$$

and obtain

$$\int_{\Gamma} (\mathbb{T}_N w)(\mathbf{x}) (\mathbb{T}_D u_w)(\mathbf{x}) \, dS(\mathbf{x}) = \int_{\Omega} \Delta w(\mathbf{x}) u_w(\mathbf{x}) + \mathbf{grad} w(\mathbf{x}) \cdot \mathbf{grad} u_w(\mathbf{x}) \, dx. \quad (1.3.1.35)$$

Combine (1.3.1.34) (with $v := u_w$) and (1.3.1.35) and conclude by means of the Cauchy-Schwarz inequality in $L^2(\Omega)$ [NumPDE Eq. (1.3.4.15)]

$$\|u_w\|_{H^1(\mathbb{R}^3)}^2 = \int_{\Omega} \Delta w(x) u_w(x) + \mathbf{grad} w(x) \cdot \mathbf{grad} u_w(x) \, dx \leq \|w\|_{H(\Delta, \Omega)} \|u_w\|_{H^1(\mathbb{R}^3)} .$$

We cancel $\|u_w\|_{H^1(\mathbb{R}^3)}$ in this inequality and the observation

$$\|\mathbb{T}_N w\|_{H^{-\frac{1}{2}}(\Gamma)} = \|\mathbf{grad} u_w\|_{L^2(\mathbb{R}^3)} \leq \|u_w\|_{H^1(\mathbb{R}^3)} \leq \|w\|_{H(\Delta, \Omega)} ,$$

clinches the proof. □ ┘

By Thm. 1.3.1.11 the Dirichlet trace \mathbb{T}_D is continuous as a mapping $H^1(\Omega) \rightarrow L^2(\Gamma)$. Then from (1.3.1.30) and (1.1.8.5) the following embedding can be inferred:

Theorem 1.3.1.36. Embedding of $H^{-\frac{1}{2}}(\Gamma)$

$$L^2(\Gamma) \text{ is continuously embedded in } H^{-\frac{1}{2}}(\Gamma): \quad L^2(\Gamma) \subset H^{-\frac{1}{2}}(\Gamma)$$

§1.3.1.37 (Duality) For $\phi \in H^{-\frac{1}{2}}(\Gamma)$ we also conclude from

$$\int_{\mathbb{R}^3} \mathbf{grad} u_{\phi} \cdot \mathbf{grad} v \, dx = \int_{\Gamma} \phi(x) (\mathbb{T}_D v)(x) \, dS(x) \quad \forall v \in H^1(\mathbb{R}^3) , \tag{1.3.1.29}$$

using the function u_{ϕ} defined thus, that

$$\int_{\Gamma} \phi(x) (\mathbb{T}_D u_{\phi})(x) \, dS(x) = \|\mathbf{grad} u_{\phi}\|_{L^2(\mathbb{R}^3)}^2 = \|\mathbf{grad} u_{\phi}\|_{L^2(\mathbb{R}^3)} \|\phi\|_{H^{-\frac{1}{2}}(\Gamma)} , \tag{1.3.1.38a}$$

$$\int_{\Gamma} \phi(x) \mathbf{v}(x) \, dS(x) = \int_{\mathbb{R}^3} \mathbf{grad} u_{\phi} \cdot \mathbf{grad} \tilde{\mathbf{v}} \, dx \leq \|\phi\|_{H^{-\frac{1}{2}}(\Gamma)} \|\mathbf{v}\|_{H^{\frac{1}{2}}(\Gamma)} \tag{1.3.1.38b}$$

for all $\mathbf{v} \in H^{\frac{1}{2}}(\Gamma)$, where $\tilde{\mathbf{v}} \in H^1(\mathbb{R}^3)$ is that extension of $\mathbf{v} \in H^{\frac{1}{2}}(\Gamma)$ for which $|\tilde{\mathbf{v}}|_{H^1(\Omega)} = \|\mathbf{v}\|_{H^{\frac{1}{2}}(\Gamma)}$. The estimates (1.3.1.38) can be translated into the following deep mathematical statement that holds for both $d = 2, 3$. The reader be reassured that grasping the full scope of the theorem is not necessary for applying it.

Theorem 1.3.1.39. $L^2(\Gamma)$ -duality between $H^{\frac{1}{2}}(\Gamma)$ and $H^{-\frac{1}{2}}(\Gamma)$

The bilinear form $(\psi, \mathbf{v}) \mapsto \int_{\Gamma} \psi(x) \mathbf{v}(x) \, dS(x)$, $\psi, \mathbf{v} \in L^2(\Gamma)$ induces isomorphisms between $H^{\frac{1}{2}}(\Gamma)$ and the dual space $(H^{-\frac{1}{2}}(\Gamma))'$, and between $H^{-\frac{1}{2}}(\Gamma)$ and the dual space $(H^{\frac{1}{2}}(\Gamma))'$. In particular,

$$\int_{\Gamma} \psi(x) \mathbf{v}(x) \, dS(x) \leq \|\psi\|_{H^{-\frac{1}{2}}(\Gamma)} \cdot \|\mathbf{v}\|_{H^{\frac{1}{2}}(\Gamma)} \quad \forall \psi \in H^{-\frac{1}{2}}(\Gamma), \mathbf{v} \in H^{\frac{1}{2}}(\Gamma) . \tag{1.3.1.40}$$

In fact, the duality asserted in Thm. 1.3.1.39 can be used to define $H^{-\frac{1}{2}}(\Gamma)$. Here, without further commenting on the theorem, we state an important consequence:

$$u, v \in H^{\frac{1}{2}}(\Gamma): u = v \Leftrightarrow \int_{\Gamma} (u - v)(x) \phi(x) dS(x) = 0 \quad \forall \phi \in H^{-\frac{1}{2}}(\Gamma), \quad (1.3.1.41a)$$

$$\phi, \psi \in H^{-\frac{1}{2}}(\Gamma): \psi = \phi \Leftrightarrow \int_{\Gamma} (\psi - \phi)(x) v(x) dS(x) = 0 \quad \forall v \in H^{\frac{1}{2}}(\Gamma). \quad (1.3.1.41b)$$

We will see several applications of these relationships below. \lrcorner

Remark 1.3.1.42 (Co-normal trace) If we deal with a general differential operator according to (1.2.0.1), $Lu := -\operatorname{div}(\mathbf{A} \operatorname{grad} u) + cu$, $\mathbf{A} \in \mathbb{R}^{d,d}$ s.p.d., $c \in \mathbb{R}$, then the Neumann trace T_N has to be replaced with the **co-normal trace** $u \mapsto T_N^L := \mathbf{A} \operatorname{grad} u \cdot \mathbf{n}|_{\Gamma}$. By and large, the results of this section carry over to T_N^L , see [SS10, Sect. 2.7]. \lrcorner

1.3.2 Mapping Properties of Layer Potentials

We recall the two layer potentials: the single layer potential Ψ_{SL} defined in Def. 1.2.5.5 and the double layer potential Ψ_{DL} defined in Def. 1.2.5.11. Above considered them for “sufficiently smooth” argument functions. Now we aim to study them as mappings between energy (trace) spaces, similar to what we have already done for the Newton potential in Cor. 1.2.3.8.

§1.3.2.1 (Single layer potential) We can relate the single layer potential operator for $-\Delta$ (\rightarrow Def. 1.2.5.5)

$$\Psi_{\text{SL}}^{\Delta}(\phi)(x) := \int_{\Gamma} G^{\Delta}(x - \mathbf{y}) \phi(\mathbf{y}) dS(\mathbf{y}), \quad x \notin \Gamma, \quad (1.2.5.6)$$

to the Newton potential (\rightarrow Def. 1.2.3.2)

$$(\mathbf{N}_{\Delta}\rho)(x) := \int_{\Omega} G^{\Delta}(x, \mathbf{y}) \rho(\mathbf{y}) d\mathbf{y}. \quad (1.2.3.3)$$

For smooth $\phi \in C^{\infty}(\overline{\Omega})|_{\Gamma}$, $\rho \in C^{\infty}(\overline{\Omega})$, interchanging integrals (Fubini's theorem), we get

$$\begin{aligned} \int_{\Omega} (\Psi_{\text{SL}}^{\Delta}\phi)(x) \rho(x) dx &= \int_{\Omega} \int_{\Gamma} G^{\Delta}(x, \mathbf{y}) \phi(\mathbf{y}) dS(\mathbf{y}) \rho(x) dx \\ &= \int_{\Gamma} \int_{\Omega} G^{\Delta}(x, \mathbf{y}) \phi(\mathbf{y}) \rho(x) dx dS(\mathbf{y}) \\ &= \int_{\Gamma} (T_D \mathbf{N}_{\Delta}\rho)(\mathbf{y}) \phi(\mathbf{y}) dS(\mathbf{y}) \\ &\stackrel{(1.3.1.40)}{\leq} \|T_D \mathbf{N}_{\Delta}\rho\|_{H^{\frac{1}{2}}(\Gamma)} \|\phi\|_{H^{-\frac{1}{2}}(\Gamma)} \\ &\leq |\mathbf{N}_{\Delta}\rho|_{H^1(\mathbb{R}^3)} \|\phi\|_{H^{-\frac{1}{2}}(\Gamma)} \leq \|\rho\|_{\tilde{H}^{-1}(\Omega)} \|\phi\|_{H^{-\frac{1}{2}}(\Gamma)}. \end{aligned}$$

We find that, if $\|\phi\|_{H^{-\frac{1}{2}}(\Gamma)} < \infty \Leftrightarrow \phi \in H^{-\frac{1}{2}}(\Gamma)$, then

$$\left| \int_{\Omega} (\Psi_{\text{SL}}^{\Delta}\phi)(x) \rho(x) dx \right| < \infty$$

for every admissible ($\|\rho\|_{\tilde{H}^{-1}(\Omega)} < \infty!$) source charge distribution ρ . Next, use the characterization (1.1.8.10).

Theorem 1.3.2.2. Continuity of single layer potential in energy (trace) spaces

The single layer potential operator Ψ_{SL}^Δ (\rightarrow Def. 1.2.5.5) can be extended to a continuous mapping

$$\Psi_{\text{SL}}^\Delta : H^{-\frac{1}{2}}(\Gamma) \rightarrow H^1(\mathbb{R}^d) \cap H(\Delta, \mathbb{R}^d \setminus \Gamma) .$$

The message of this theorem is that we can find a constant $C > 0$ depending only on Ω such that

$$\left\| \Psi_{\text{SL}}^\Delta \phi \right\|_{H^1(\mathbb{R}^d)} + \left\| \Psi_{\text{SL}}^\Delta \phi \right\|_{H(\Delta, \Omega)} + \left\| \Psi_{\text{SL}}^\Delta \phi \right\|_{H(\Delta, \Omega')} \leq C \|\phi\|_{H^{-\frac{1}{2}}(\Gamma)} .$$

┘

§1.3.2.3 (Double layer potential) To establish the continuity of the double layer potential operator Ψ_{DL}^Δ from Def. 1.2.5.11, we rely on the representation formulas (1.2.4.3) (for $L = -\Delta$) or (1.2.4.6). These can be written in a compact way as

$$u = N_\Delta(-\Delta u) + \Psi_{\text{SL}}^\Delta(\mathcal{T}_N u) - \Psi_{\text{DL}}^\Delta(\mathcal{T}_D u) \quad \forall u \in C^2(\bar{\Omega}) .$$

Pick $\mathbf{v} \in C^\infty(\bar{\Omega})|_\Gamma$ and define $u \in H^1(\Omega)$ as the solution of

$$-\Delta u = 0 \quad \text{in } \Omega \quad , \quad \mathcal{T}_D u = \mathbf{v} \quad \text{on } \Gamma .$$

By the continuity result for Ψ_{SL}^Δ from Thm. 1.3.2.2 we can plug this u into the representation formula

$$\blacktriangleright \quad u = \Psi_{\text{SL}}^\Delta(\mathcal{T}_N u) - \Psi_{\text{DL}}^\Delta(\mathbf{v}) \quad \text{in } \Omega . \quad (1.3.2.4)$$

Then, by Thm. 1.3.2.2 and Thm. 1.3.1.33 ($\Delta u = 0!$)

$$\left\| \Psi_{\text{SL}}^\Delta(\mathcal{T}_N u) \right\|_{H^1(\Omega)} \leq C \|\mathcal{T}_N u\|_{H^{-\frac{1}{2}}(\Gamma)} \leq C \|u\|_{H(\Delta, \Omega)} \leq C \|u\|_{H^1(\Omega)} \leq C \|\mathbf{v}\|_{H^{\frac{1}{2}}(\Gamma)} ,$$

with positive constants with different values at each stage but all independent of \mathbf{v} . The Δ -inequality combined with (1.3.2.4) yields

$$\left\| \Psi_{\text{DL}}^\Delta(\mathbf{v}) \right\|_{H^1(\Omega)} \leq \|u\|_{H^1(\Omega)} + \left\| \Psi_{\text{SL}}^\Delta(\mathcal{T}_N u) \right\|_{H^1(\Omega)} \leq C \|\mathbf{v}\|_{H^{\frac{1}{2}}(\Gamma)} .$$

This argument can also be employed on the complement domain Ω' .

Theorem 1.3.2.5. Continuity of the double layer potential in energy trace spaces

The double layer potential operator Ψ_{DL}^Δ (\rightarrow Def. 1.2.5.11) can be extended to a continuous mapping

$$\Psi_{\text{DL}}^\Delta : H^{\frac{1}{2}}(\Gamma) \rightarrow H(\Delta, \mathbb{R}^d \setminus \Gamma) .$$

┘

Remark 1.3.2.6 (General layer potentials) All the above arguments and results remain valid for layer potentials derived from fundamental solutions for general scalar second-order differential operators L in divergence form (1.2.0.1).

┘

1.3.3 Jump Relations for Layer Potentials

In § 1.2.5.14 we saw that the double layer potential may have a discontinuity, a jump, across Γ . In this section we will glean detailed information about jumps and kinks (ie, jumps of derivatives) of potentials.

§1.3.3.1 (Jumps and averages) Let $u \in L^2(\mathbb{R}^d)$ be smooth on both sides of $\Gamma := \partial\Omega$: $u|_{\Omega} \in C^\infty(\overline{\Omega})$ and $u|_{\Omega'} \in C^\infty(\overline{\Omega'})$, $\Omega' = \mathbb{R}^d \setminus \overline{\Omega}$. Then we can apply some trace operator \mathbf{T} on both sides and take the difference of the resulting functions, what we call a **jump** of $\mathbf{T}u$.

Concretely, for the jumps of Dirichlet and Neumann traces introduced in Def. 1.3.1.1 and Def. 1.3.1.20, respectively, we write

$$\text{Jumps:} \quad \begin{aligned} \llbracket \mathbf{T}_D u \rrbracket_\Gamma &:= \mathbf{T}_D(u|_{\Omega'}) - \mathbf{T}_D(u|_{\Omega}), \\ \llbracket \mathbf{T}_N u \rrbracket_\Gamma &:= \mathbf{T}_N(u|_{\Omega'}) - \mathbf{T}_N(u|_{\Omega}), \end{aligned}$$

where in the second difference \mathbf{T}_N is based on the exterior unit normal for Ω throughout. Jumps adhere to the convention “outside – inside” and they are functions on Γ . Note that the exterior unit normal for Ω enters the Neumann jump:

$$\llbracket \mathbf{T}_N u \rrbracket_\Gamma(x) = ((\mathbf{grad} u|_{\Omega'})(x) - (\mathbf{grad} u|_{\Omega})(x)) \cdot \mathbf{n}(x), \quad x \in \Gamma.$$

Similarly we can define **averages** of traces:

$$\text{Averages:} \quad \begin{aligned} \{ \mathbf{T}_D u \}_\Gamma &:= \frac{1}{2}(\mathbf{T}_D(u|_{\Omega'}) + \mathbf{T}_D(u|_{\Omega})), \\ \{ \mathbf{T}_N u \}_\Gamma &:= \frac{1}{2}(\mathbf{T}_N(u|_{\Omega'}) + \mathbf{T}_N(u|_{\Omega})). \end{aligned}$$

§1.3.3.2 (Jump representation formula) Pick $u \in C^\infty(\overline{\Omega})$, $\Delta u = 0$ in Ω , and $x \notin \overline{\Omega}$, that is, x is located in the interior of the complement domain Ω' . Then, by property (ii) of a fundamental solution (\rightarrow Def. 1.2.2.15), $\mathbf{y} \mapsto G^\Delta(x, \mathbf{y})$ is harmonic in Ω : $\Delta_{\mathbf{y}} G^\Delta(x, \mathbf{y}) = 0$. As a consequence of Green’s second formula (1.2.1.3)

$$\begin{aligned} \Psi_{\text{SL}}^\Delta(\mathbf{T}_N u)(x) - \Psi_{\text{DL}}^\Delta(\mathbf{T}_D u)(x) &= \int_\Gamma G^\Delta(x, \mathbf{y})(\mathbf{T}_N u)(\mathbf{y}) - \mathbf{grad}_{\mathbf{y}} G^\Delta(x, \mathbf{y}) \cdot \mathbf{n}(\mathbf{y})(\mathbf{T}_D u)(\mathbf{y}) \, dS(\mathbf{y}) \\ &= \int_\Omega G^\Delta(x, \mathbf{y}) \underbrace{(\Delta u)(\mathbf{y})}^0 - \underbrace{(\Delta_{\mathbf{y}} G^\Delta)(x, \mathbf{y})}_{\rightarrow 0} u(\mathbf{y}) \, d\mathbf{y} = 0. \end{aligned}$$

For bounded Ω combining this finding with the “interior” integral representation formula of Thm. 1.2.4.2 in the form (1.3.0.2), we get

$$\Psi_{\text{SL}}^\Delta(\mathbf{T}_N u) - \Psi_{\text{DL}}^\Delta(\mathbf{T}_D u) = \begin{cases} u(x) & , \text{ if } x \in \Omega, \\ 0 & , \text{ if } x \in \Omega'. \end{cases} \quad (1.3.3.3)$$

The same reasoning can be pursued for the “exterior” complement domain Ω' based on Thm. 1.2.4.5. Merging the resulting formulas gives a new version of the representation formula on $\mathbb{R}^d \setminus \Gamma$.

Theorem 1.3.3.4. Jump representation formula [SS10, Thm. 3.1.8]

For $u \in H(\Delta, \mathbb{R}^d \setminus \Gamma)$, $\Delta u = 0$ in $\Omega \cup \Omega'$, holds

$$u = -\Psi_{\text{SL}}^\Delta(\llbracket \mathbf{T}_N u \rrbracket_\Gamma) + \Psi_{\text{DL}}^\Delta(\llbracket \mathbf{T}_D u \rrbracket_\Gamma) \quad \text{in } H(\Delta, \mathbb{R}^d \setminus \Gamma). \quad (1.3.3.5)$$

We could state this theorem in terms of energy spaces, since from 1.3.2 we know that all traces and layer potentials are well defined. ┘

§1.3.3.6 (Jumps of single layer potential) According to Thm. 1.3.2.2, for $\phi \in H^{-\frac{1}{2}}(\Gamma)$ we know $\Psi_{\text{SL}}^\Delta \phi \in H^1(\mathbb{R}^d)$. Appealing to the fact that “functions in H^1 must not have discontinuities”, see Thm. 1.1.3.5, we conclude that

$$\llbracket \mathbb{T}_D(\Psi_{\text{SL}}^\Delta \phi) \rrbracket_\Gamma = 0 \quad \forall \phi \in H^{-\frac{1}{2}}(\Gamma). \tag{1.3.3.7}$$

We rely on “electrostatic heuristics” to elaborate the Neumann jump of $\Psi_{\text{SL}}^\Delta \phi$, recalling Rem. 1.2.5.10. From (1.2.3.9) we know that the Newton potential

$$(\mathbb{N}_\Delta \rho)(x) := \int_\Omega G^\Delta(x, \mathbf{y}) \rho(\mathbf{y}) \, d\mathbf{y}, \quad x \in \mathbb{R}^d,$$

generates the potential produced by the source charge distribution $\rho \in \tilde{H}^{-1}(\mathbb{R}^3)$. It solves the variational problem

$$\mathbb{N}_\Delta \rho \in H^1(\mathbb{R}^d): \int_\Omega (\mathbf{grad} \mathbb{N}_\Delta \rho)(x) \cdot \mathbf{grad} v(x) \, dx = \int_\Omega \rho(x) v(x) \, dx, \tag{1.3.3.8}$$

for all $v \in H^1(\mathbb{R}^d)$. Match this with the formula

$$(\Psi_{\text{SL}}^\Delta \phi)(x) = \int_\Gamma G^\Delta(x, \mathbf{y}) \phi(\mathbf{y}) \, dS(\mathbf{y}), \quad x \in \mathbb{R}^d,$$

defining the single layer potential, which gives the electrostatic potential due to the **surface charge** $\phi \in H^{-\frac{1}{2}}(\Gamma)$. Adapting (1.3.3.8), we find that

$$\int_\Omega (\mathbf{grad} \Psi_{\text{SL}}^\Delta \phi)(x) \cdot \mathbf{grad} v(x) \, dx = \int_\Gamma \phi(x) v(x) \, dS(x) \quad \forall v \in H^1(\mathbb{R}^3) \tag{1.3.3.9}$$

The policy demonstrated in 1.1.6 can be used to find the PDE form of the **transmission problem** encoded by (1.3.3.9). First test with smooth v compactly supported inside either Ω or Ω' , which shows

$$\Delta \Psi_{\text{SL}}^\Delta(\phi) = 0 \quad \text{in } \Omega \cup \Omega'. \tag{1.3.3.10}$$

Then test with $v \in C_0^\infty(\mathbb{R}^d)$, perform integration by parts (Green’s first formula (1.1.6.2)) both in Ω and Ω' and use (1.3.3.10) to remove all volume integrals. The remaining boundary terms on Γ lead to

$$\llbracket \mathbb{T}_N \Psi_{\text{SL}}^\Delta(\phi) \rrbracket_\Gamma = -\phi. \tag{1.3.3.11}$$

┘

§1.3.3.12 (Jumps of double layer potential)



For arbitrary $u \in H(\Delta, \mathbb{R}^d \setminus \Gamma)$ apply the jump operators $\llbracket \mathbb{T}_D \cdot \rrbracket_\Gamma$ and $\llbracket \mathbb{T}_N \cdot \rrbracket_\Gamma$ to the jump representation formula (\rightarrow Thm. 1.3.3.4)

$$u = -\Psi_{\text{SL}}^\Delta(\llbracket \mathbb{T}_N u \rrbracket_\Gamma) + \Psi_{\text{DL}}^\Delta(\llbracket \mathbb{T}_D u \rrbracket_\Gamma) \quad \text{in } H(\Delta, \mathbb{R}^d \setminus \Gamma). \tag{1.3.3.5}$$

◆ Apply $\llbracket \mathbb{T}_D \cdot \rrbracket_\Gamma$: In light of $\llbracket \Psi_{\text{SL}}^\Delta \rrbracket_\Gamma = 0$, see (1.3.3.7), we infer

$$\llbracket \mathbb{T}_D u \rrbracket_\Gamma = \llbracket \mathbb{T}_D \Psi_{\text{DL}}^\Delta(\llbracket \mathbb{T}_D u \rrbracket_\Gamma) \rrbracket_\Gamma \Leftrightarrow \llbracket \mathbb{T}_D \Psi_{\text{DL}}^\Delta \mathbf{v} \rrbracket_\Gamma = \mathbf{v} \quad \forall \mathbf{v} \in H^{\frac{1}{2}}(\Gamma), \tag{1.3.3.13}$$

because any jump $\llbracket \mathbb{T}_D u \rrbracket_\Gamma$ can be realized by choosing an appropriate u .

◆ Apply $[[\mathbb{T}_N \cdot]]_\Gamma$: By virtue of (1.3.3.11) we obtain from (1.3.3.5)

$$\begin{aligned} [[\mathbb{T}_N u]]_\Gamma &= - \underbrace{\left[\mathbb{T}_N \Psi_{\text{SL}}^\Delta ([[\mathbb{T}_N u]]_\Gamma) \right]_\Gamma}_{=-[[\mathbb{T}_N u]]_\Gamma} + \left[\mathbb{T}_N \Psi_{\text{DL}}^\Delta ([[\mathbb{T}_D u]]_\Gamma) \right]_\Gamma \\ &\quad \Downarrow \\ \left[\mathbb{T}_N \Psi_{\text{DL}}^\Delta \mathbf{v} \right]_\Gamma &= 0 \quad \forall \mathbf{v} \in H^{\frac{1}{2}}(\Gamma). \end{aligned} \quad (1.3.3.14)$$

┘

The following theorem summarizes our finding (1.3.3.7), (1.3.3.11), (1.3.3.13), (1.3.3.14).

Theorem 1.3.3.15. Jump relations for layer potentials [SS10, Thm. 3.3.1]

The single and double layer potentials Ψ_{SL}^Δ and Ψ_{DL}^Δ satisfy for all $\phi \in H^{-\frac{1}{2}}(\Gamma)$ and $\mathbf{v} \in H^{\frac{1}{2}}(\Gamma)$ the *jump relations*

$$\begin{aligned} \left[\mathbb{T}_D \Psi_{\text{SL}}^\Delta \phi \right]_\Gamma &= 0, & \left[\mathbb{T}_D \Psi_{\text{DL}}^\Delta \mathbf{v} \right]_\Gamma &= \mathbf{v} \text{ in } H^{\frac{1}{2}}(\Gamma), \\ \left[\mathbb{T}_N \Psi_{\text{SL}}^\Delta \phi \right]_\Gamma &= -\phi, & \left[\mathbb{T}_N \Psi_{\text{DL}}^\Delta \mathbf{v} \right]_\Gamma &= 0 \text{ in } H^{-\frac{1}{2}}(\Gamma). \end{aligned} \quad (1.3.3.16)$$

1.3.4 Boundary Integral Operators (BIOs)

Boundary integral operators (BIOs) arise from applying traces to layer potentials. By the results of Section 1.3.2 this is possible and the continuity properties in energy trace spaces are immediately clear. The challenge is to establish concrete integral formulas for the BIOs.

We exclusively focus on the Laplace operator, but point out that analogous considerations apply to all scalar second-order differential operators with constant coefficients.

1.3.4.1 Formal Definition

As explained in § 1.3.0.1:

$$\text{Two traces } \left\{ \begin{array}{c} \mathbb{T}_D \\ \mathbb{T}_N \end{array} \right\} + \text{two layer potentials } \left\{ \begin{array}{c} \Psi_{\text{SL}}^\Delta \\ \Psi_{\text{DL}}^\Delta \end{array} \right\} \blacktriangleright \text{four BIOs !}$$

Layer potentials are defined everywhere in $\mathbb{R}^d \setminus \Gamma$. The jump relations of Thm. 1.3.3.15 teach that traces of layer potentials may jump. Thus it makes a difference whether we take the trace from inside or outside Ω .

The convention adopted in the literature resorts to the average $\{\mathbb{T}\cdot\}_\Gamma$ of traces to resolve this ambiguity.

Definition 1.3.4.1. Boundary integral operators for $-\Delta$

The four **boundary integral operators** associated with the Laplacian $-\Delta$ are defined as follows:

$$\begin{aligned} \text{single layer BIO: } \mathbb{V}(\phi) &:= \left\{ \mathbb{T}_D \Psi_{\text{SL}}^\Delta(\phi) \right\}_\Gamma, \quad \phi \in H^{-\frac{1}{2}}(\Gamma), \\ \text{double layer BIO: } \mathbb{K}(\mathbf{v}) &:= \left\{ \mathbb{T}_D \Psi_{\text{DL}}^\Delta(\mathbf{v}) \right\}_\Gamma, \quad \mathbf{v} \in H^{\frac{1}{2}}(\Gamma), \\ \text{adjoint double layer BIO: } \mathbb{K}'(\phi) &:= \left\{ \mathbb{T}_N \Psi_{\text{SL}}^\Delta(\phi) \right\}_\Gamma, \quad \phi \in H^{-\frac{1}{2}}(\Gamma), \\ \text{hypersingular BIO: } \mathbb{W}(\mathbf{v}) &:= -\left\{ \mathbb{T}_N \Psi_{\text{DL}}^\Delta(\mathbf{v}) \right\}_\Gamma, \quad \mathbf{v} \in H^{\frac{1}{2}}(\Gamma). \end{aligned}$$

The mapping properties of BIOs in trace spaces follow immediately from what we know:



Continuity of
trace operators
(Cor. 1.3.1.8, Thm. 1.3.1.33)

+

Continuity of
layer potentials
(Thm. 1.3.2.2, Thm. 1.3.2.5)



Continuity
of BIOs

The next theorem gives summary.

Theorem 1.3.4.2. Continuity of boundary integral operators

The following linear operators are continuous:

$$\begin{aligned} \text{single layer BIO: } \mathbb{V} &: H^{-\frac{1}{2}}(\Gamma) \rightarrow H^{\frac{1}{2}}(\Gamma), \\ \text{double layer BIO: } \mathbb{K} &: H^{\frac{1}{2}}(\Gamma) \rightarrow H^{\frac{1}{2}}(\Gamma), \\ \text{adjoint double layer BIO: } \mathbb{K}' &: H^{-\frac{1}{2}}(\Gamma) \rightarrow H^{-\frac{1}{2}}(\Gamma), \\ \text{hypersingular BIO: } \mathbb{W} &: H^{\frac{1}{2}}(\Gamma) \rightarrow H^{-\frac{1}{2}}(\Gamma). \end{aligned}$$

Supplement 1.3.4.3 (Adjointness of double layer potentials) The reason, why \mathbb{K}' is called the **adjoint** double layer boundary integral operator is the formula

$$\int_\Gamma (\mathbb{K}u)(\mathbf{x}) \phi(\mathbf{x}) \, dS(\mathbf{x}) = \int_\Gamma u(\mathbf{x}) (\mathbb{K}'\phi)(\mathbf{x}) \, dS(\mathbf{x}), \quad (1.3.4.4)$$

which has to be seen from the perspective of the definition (1.2.2.17) of an adjoint operator. The proof of the formula makes use of the fact that, if u and v are harmonic in Ω , then

$$\int_\Gamma (\mathbb{T}_D u)(\mathbf{x}) (\mathbb{T}_N v)(\mathbf{x}) \, dS(\mathbf{x}) = \int_\Gamma (\mathbb{T}_N u)(\mathbf{x}) (\mathbb{T}_D v)(\mathbf{x}) \, dS(\mathbf{x}),$$

which is a consequence of (1.3.1.24). ┘

§1.3.4.5 (Continuity of BIOs in spaces of higher smoothness) Through Lipschitz parameterization of Γ we can define Sobolev spaces on Γ , see [SS10, Sect 2.4]: A function $f \in L^2(\Gamma)$ belongs to $H^1(\Gamma)$, if its pullback under the parameterization belongs to H^1 on the parameter domain.

As explained in [SS10, Sect. 3.1.2], the trace operators and the layer potentials also enjoy continuity in higher order Sobolev spaces. Hence, this is inherited by the boundary integral operators [SS10, Rem. 3.1.18], [Ste08, Sect. 6.6.5].

Theorem 1.3.4.6. “Higher” continuity of BIOs

The boundary integral operators from Def. 1.3.4.1 are continuous as operators mapping between the following spaces:

$$\begin{aligned}
 \text{single layer BIO:} & \quad \mathbb{V} : L^2(\Gamma) \rightarrow H^1(\Gamma) , \\
 \text{double layer BIO:} & \quad \mathbb{K} : L^2(\Gamma) \rightarrow L^2(\Gamma) , \\
 \text{adjoint double layer BIO:} & \quad \mathbb{K}' : L^2(\Gamma) \rightarrow L^2(\Gamma) , \\
 \text{hypersingular BIO:} & \quad \mathbb{W} : H^1(\Gamma) \rightarrow L^2(\Gamma) .
 \end{aligned}$$

┘

1.3.4.2 Integral Representations

“Integral representations” mean the possibility to write a boundary integral operator applied to a *sufficiently smooth* function $f : \Gamma \rightarrow \mathbb{R}$ in the form

$$f \mapsto \left\{ \mathbf{x} \mapsto \int_{\Gamma} k(\mathbf{x}, \mathbf{y}) f(\mathbf{y}) \, dS(\mathbf{y}) \mid \mathbf{x} \in \Gamma \right\} , \tag{1.3.4.7}$$

with a **kernel** $k : \Gamma \times \Gamma \rightarrow \mathbb{R}$. From Def. 1.3.4.1 it is not immediately clear that this is possible for the four BIOs. However, for numerical purposes it is essential that such integral representations are at our disposal.

§1.3.4.8 (Integral representation for single layer BIO) We have already noted

$$G^{\Delta}(\mathbf{x}, \mathbf{y}) = \begin{cases} -\frac{1}{2\pi} \log \|\mathbf{x} - \mathbf{y}\| & , \text{ if } d = 2 , \\ \frac{1}{4\pi} \frac{1}{\|\mathbf{x} - \mathbf{y}\|} & , \text{ if } d = 3 . \end{cases} \tag{1.2.2.33}$$

This implies that $\mathbf{y} \mapsto G^{\Delta}(\mathbf{x}, \mathbf{y})$ is integrable even on Γ : $\{\mathbf{y} \mapsto G^{\Delta}(\mathbf{x}, \mathbf{y})\} \in L^1(\Gamma)$ for any $\mathbf{x} \in \mathbb{R}^d$. Hence, for $\phi \in L^{\infty}(\Gamma)$ we have the integral representation as an improper (due to “ $G^{\Delta}(\mathbf{x}, \mathbf{x}) = \infty$ ”) integral

$$(\mathbb{V}\phi)(\mathbf{x}) = \int_{\Gamma} G^{\Delta}(\mathbf{x}, \mathbf{y}) \phi(\mathbf{y}) \, dS(\mathbf{y}) . \tag{1.3.4.9}$$

┘

The situation is more involved for the remaining BIOs, because their kernels feature stronger singularities and fail to be integrable on Γ .

§1.3.4.10 (Integral representation for double layer BIOs) The kernel of the double layer potential for $-\Delta$

$$(\Psi_{\text{DL}}^{\Delta} \mathbf{v})(\mathbf{x}) = \int_{\Gamma} \frac{\mathbf{x} - \mathbf{y}}{\omega_d \|\mathbf{x} - \mathbf{y}\|^d} \cdot \mathbf{n}(\mathbf{y}) \mathbf{v}(\mathbf{y}) \, dS(\mathbf{y}) , \quad \omega_d := \begin{cases} 2\pi & \text{for } d = 2 , \\ 4\pi & \text{for } d = 3 , \end{cases} \tag{1.3.4.11}$$

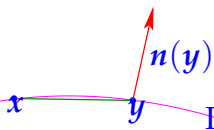
is not integrable a priori. On smooth parts of Γ , however, we make the following observation:

Lemma 1.3.4.12.
 If Γ is C^2 -smooth in a neighborhood of $x \in \Gamma$, then

$$|(x - y) \cdot n(y)| = O(\|x - y\|^2) \quad (1.3.4.13)$$

for $y \in \Gamma \rightarrow x$.

Fig. 19



◁ As $y \rightarrow x$ on Γ the normal $n(y)$ becomes “more and more orthogonal” to $x - y$, see [SS10, Lemma 2.2.14] for a rigorous proof.

► Under Ass. 1.2.1.5/Ass. 1.2.1.7 (Γ is a curved polygon/polyhedron with smooth faces) the kernel of the double layer potential behaves like

$$k(x, y) = \frac{x - y}{\omega_d \|x - y\|^d} \cdot n(y) = O(\|x - y\|^{2-d}) \quad \text{for } y \in \Gamma \rightarrow x,$$

for almost all $x \in \Gamma$.

Hence, for almost all $x \in \Gamma$ we can take for granted the integral representation formulas

$$K(v)(x) = \int_{\Gamma} \frac{x - y}{\omega_d \|x - y\|^d} \cdot n(y) v(y) dS(y), \quad x \in \text{smooth part of } \Gamma, \quad (1.3.4.14)$$

$$K'(\phi)(x) = \int_{\Gamma} \frac{y - x}{\omega_d \|x - y\|^d} \cdot n(x) \phi(y) dS(y), \quad x \in \text{smooth part of } \Gamma. \quad (1.3.4.15)$$

A rigorous treatment and a discussion of what happens at edges and corners can be found in [Hac95, Sect. 8.2] and [SS10, Sect. 3.3.3].

§1.3.4.16 (No integral representation for hypersingular BIO) Formally applying the Neumann trace T_N to the double layer potential Ψ_{DL} yields

$$(T_N \Psi_{DL} v)(x) = \int_{\Gamma} \left(\frac{n(y) \cdot n(x)}{\|x - y\|^d} - d \frac{(n(y) \cdot (x - y))(n(x) \cdot (x - y))}{\|x - y\|^{d+2}} \right) v(y) dS(y)$$

non-integrable for $x \in \Gamma$
integrable by Lemma 1.3.4.12

There is no useful surface integral representation for then hypersingular integral operator.

1.3.4.3 Variational Form for Hypersingular BIO

Fortunately, it has been discovered that the hypersingular operator W in weak form is amenable to a reformulation by integration by parts that curbs the strength of the singularity of the kernel.

Let us first examine that weak form: By Thm. 1.3.4.2 the hypersingular operator maps continuously $W : H^{\frac{1}{2}}(\Gamma) \rightarrow H^{-\frac{1}{2}}(\Gamma)$. Therefore, owing to Thm. 1.3.1.39, it gives rise to a continuous bilinear form

$$a_W : \begin{cases} H^{\frac{1}{2}}(\Gamma) \times H^{\frac{1}{2}}(\Gamma) & \rightarrow \mathbb{R} \\ (u, v) & \mapsto \int_{\Gamma} (Wu)(x) v(x) dS(x) \end{cases} \quad (1.3.4.17)$$

provided that \mathbf{u}, \mathbf{v} are “sufficiently smooth”, by “technical manipulations” equivalent expressions for $a_W(\mathbf{u}, \mathbf{v})$ can be derived that merely involve improper integrals on Γ .

§1.3.4.18 (Integration by parts on curves) Let $\gamma : [0, 1] \rightarrow \mathbb{R}^2$ be a C^2 -parameterization of a curve Σ with endpoints \mathbf{a}, \mathbf{b} . The **arclength derivative** of a function $f \in C^1(\Sigma)$ in $\mathbf{y} \in \Sigma$ is (tags the derivative of an univariate function)

$$\frac{df}{ds}(\mathbf{y}) = \dot{F}(t^*) \|\dot{\gamma}(t^*)\|^{-1}, \quad \mathbf{y} =: \gamma(t^*), \quad F(t) := f(\gamma(t)), \quad 0 \leq t, t^* \leq 1. \tag{1.3.4.19}$$

As a consequence of the chain rule, the arclength derivative is independent of the parameterization.

Given another function $g \in C^1(\Sigma)$, $G := g \circ \gamma$, we find the integration by parts formula for the arclength derivative:

$$\begin{aligned} \int_{\Sigma} \frac{df}{ds}(\mathbf{y}) g(\mathbf{y}) dS(\mathbf{y}) &= \int_0^1 \frac{\dot{F}(t)}{\|\dot{\gamma}(t)\|} G(t) \|\dot{\gamma}(t)\| dt = \int_0^1 \dot{F}(t) G(t) dt \\ &= F(1)G(1) - F(0)G(0) - \int_0^1 F(t) \dot{G}(t) dt \\ &= f(\mathbf{b})g(\mathbf{b}) - f(\mathbf{a})g(\mathbf{a}) - \int_{\Sigma} f(\mathbf{y}) \frac{dg}{ds}(\mathbf{y}) dS(\mathbf{y}). \end{aligned}$$

Let Γ be a **closed** curved Lipschitz polygon according to Ass. 1.2.1.5:

$$\Gamma = \bar{\Gamma}_1 \cup \dots \cup \bar{\Gamma}_M, \quad C^2\text{-parameterizations } \gamma_j : [0, 1] \rightarrow \bar{\Gamma}_j, \quad \begin{aligned} \gamma_{j-1}(1) &= \gamma_j(0), \\ \gamma_M(1) &= \gamma_1(0). \end{aligned}$$

Then we apply the integration by parts formula for the arclength derivative on each segment and observe that the endpoint contributions cancel:

Lemma 1.3.4.20. Arclength integration by parts

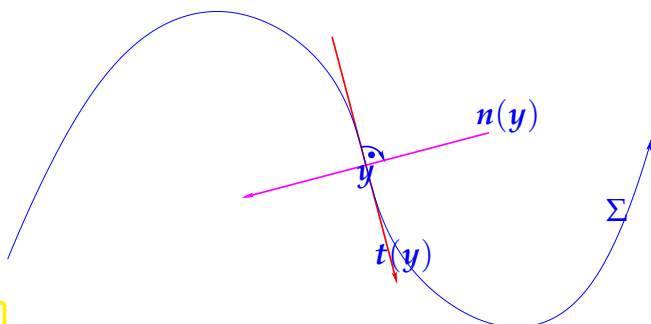
With Γ a closed Lipschitz curve satisfying Ass. 1.2.1.5 for $f, g \in C^1(\Gamma)$ we have

$$\int_{\Gamma} \frac{df}{ds}(\mathbf{y}) g(\mathbf{y}) dS(\mathbf{y}) = - \int_{\Gamma} f(\mathbf{y}) \frac{dg}{ds}(\mathbf{y}) dS(\mathbf{y}). \tag{1.3.4.21}$$

§1.3.4.22 (Arclength derivative of restrictions) With the notations of the previous §, if $f = \tilde{f}|_{\Sigma}$, where \tilde{f} is a C^1 -function defined in a neighborhood of Σ , then, by the chain rule,

$$\frac{df}{ds}(\mathbf{y}) = \mathbf{grad} \tilde{f}(\mathbf{y}) \cdot \mathbf{t}(\mathbf{y}), \quad \mathbf{y} \in \Sigma, \tag{1.3.4.23}$$

with \mathbf{t} standing for the unit tangent vector field at Σ :



$$\begin{aligned} \mathbf{t}(\mathbf{y}) &= \mathbf{n}(\mathbf{y})^\perp := \begin{bmatrix} -n_2(\mathbf{y}) \\ n_1(\mathbf{y}) \end{bmatrix}, \\ \mathbf{n}(\mathbf{y}) &= \begin{bmatrix} n_1(\mathbf{y}) \\ n_2(\mathbf{y}) \end{bmatrix} \hat{=} \text{unit normal vector at } \Sigma. \end{aligned}$$

Fig. 20

§1.3.4.24 (Integration by parts of a_{w} in 2D) We start from the formula for the double layer potential

$$(\Psi_{\text{DL}}^{\Delta} u)(x) = \int_{\Gamma} \mathbf{grad}_y G^{\Delta}(x, y) \cdot \mathbf{n}(y) u(y) dS(y),$$

for $u \in C_{\text{pw}}^1(\Gamma)$ smooth on all segments of Γ . By elementary computations

$$\frac{\partial G^{\Delta}}{\partial x_i}(x, y) = -\frac{1}{2\pi} \frac{\partial}{\partial x_i} \{\log \|x - y\|\} = -\frac{1}{2\pi} \frac{y_i - x_i}{\|x - y\|^2} = \frac{1}{2\pi} \frac{x_i - y_i}{\|x - y\|^2} = -\frac{\partial G^{\Delta}}{\partial y_i}(x, y), \quad x \neq y.$$

Hence, for $y \in \Gamma$,

$$\frac{\partial}{\partial x_i} \left(\mathbf{grad}_y G^{\Delta}(x, y) \cdot \mathbf{n}(y) \right) = -\mathbf{grad}_y \frac{\partial}{\partial y_i} G^{\Delta}(x, y) \cdot \mathbf{n}(y), \quad x \neq y.$$

$$\begin{aligned} \blacktriangleright \quad \frac{d}{ds} \left(\frac{\partial G^{\Delta}}{\partial y_1}(x, y) \right) &= n_1(y) \frac{\partial^2 G^{\Delta}}{\partial y_1 \partial y_2}(x, y) - n_2(y) \frac{\partial^2 G^{\Delta}}{\partial y_1^2}(x, y) \\ &\stackrel{(*)}{=} n_1(y) \frac{\partial^2 G^{\Delta}}{\partial y_1 \partial y_2}(x, y) + n_2(y) \frac{\partial^2 G^{\Delta}}{\partial y_2^2}(x, y) \\ &= n(y) \cdot \mathbf{grad}_y \left(\frac{\partial G^{\Delta}}{\partial y_2}(x, y) \right), \end{aligned}$$

$$\blacktriangleright \quad \frac{d}{ds} \left(\frac{\partial G^{\Delta}}{\partial y_2}(x, y) \right) = -n(y) \cdot \mathbf{grad}_y \left(\frac{\partial G^{\Delta}}{\partial y_1}(x, y) \right).$$

In step (*) we used that

$$\Delta_y G^{\Delta}(x, y) = \frac{\partial^2 G^{\Delta}}{\partial y_1^2}(x, y) + \frac{\partial^2 G^{\Delta}}{\partial y_2^2}(x, y) = 0.$$

Using all these auxiliary results, we obtain for the partial derivatives of the double layer potential

$$\begin{aligned} \frac{\partial \Psi_{\text{DL}}^{\Delta}(u)}{\partial x_1}(x) &= \int_{\Gamma} \frac{\partial}{\partial x_1} \left(\mathbf{grad}_y G^{\Delta}(x, y) \cdot \mathbf{n}(y) \right) u(y) dS(y) \\ &= - \int_{\Gamma} \mathbf{grad}_y \left\{ \frac{\partial G^{\Delta}}{\partial y_1}(x, y) \right\} \cdot \mathbf{n}(y) dS(y) \\ &= \int_{\Gamma} \frac{d}{ds} \left\{ \frac{\partial G^{\Delta}}{\partial y_2}(x, y) \right\} u(y) dS(y) \\ &= - \int_{\Gamma} \frac{\partial G^{\Delta}}{\partial y_2}(x, y) \frac{du}{ds}(y) dS(y), \\ \frac{\partial \Psi_{\text{DL}}^{\Delta}(u)}{\partial x_2}(x) &= \int_{\Gamma} \frac{\partial G^{\Delta}}{\partial y_1}(x, y) \frac{du}{ds}(y) dS(y). \end{aligned}$$

Now we attack the Neumann trace of the double layer potential in $x \in \Gamma$. We dodge issues of integrability and rely on formal manipulations (that can all be justified rigorously, of course). Using the above expressions for $\mathbf{grad} \Psi_{\text{DL}}^{\Delta}(u)$ we recover another arclength derivative:

$$\begin{aligned} (\mathbf{grad} \Psi_{\text{DL}}^{\Delta} u)(x) \cdot \mathbf{n}(x) &= \int_{\Gamma} \left\{ -n_1(x) \frac{\partial G^{\Delta}}{\partial y_2}(x, y) + n_2(x) \frac{\partial G^{\Delta}}{\partial y_1}(x, y) \right\} \frac{du}{ds}(y) dS(y) \\ &\quad + \int_{\Gamma} \left\{ n_1(x) \frac{\partial G^{\Delta}}{\partial x_2}(x, y) - n_2(x) \frac{\partial G^{\Delta}}{\partial x_1}(x, y) \right\} \frac{du}{ds}(y) dS(y) \\ &\quad + \int_{\Gamma} \frac{d}{ds(x)} \left\{ G^{\Delta}(x, y) \frac{du}{ds}(y) \right\} dS(y). \end{aligned}$$

This arclength derivative can be moved onto the second argument of the bilinear form a_W :

$$\begin{aligned} a_W(u, v) &= - \int_{\Gamma} (\mathbf{grad} \Psi_{DL} u)(x) \cdot \mathbf{n}(x) v(x) dS(x) \\ &= - \int_{\Gamma} \int_{\Gamma} \frac{d}{ds(x)} \left\{ G^{\Delta}(x, y) \frac{du}{ds}(y) \right\} dS(y) v(x) dS(x) \\ &= \int_{\Gamma} \int_{\Gamma} G^{\Delta}(x, y) \frac{du}{ds}(y) \frac{dv}{ds}(x) dS(y) dS(x) . \end{aligned}$$

Finally, we have arrived at an integral operator with the same **integrable kernel** as V . We traded this reduction of the singularity of the kernel for the need to differentiate the argument functions.

Theorem 1.3.4.25. Integral representation of a_W in 2D

If $d = 2$, $u, v \in C_{pw}^1(\Gamma)$, then the bilinear form a_W from (1.3.4.17) induced by the hypersingular operator $W : H^{\frac{1}{2}}(\Gamma) \rightarrow H^{-\frac{1}{2}}(\Gamma)$ can be expressed as

$$a_W(u, v) = -\frac{1}{2\pi} \int_{\Gamma} \int_{\Gamma} \log \|x - y\| \frac{du}{ds}(y) \frac{dv}{ds}(x) dS(y) dS(x) , \quad (1.3.4.26)$$

where $\frac{d}{ds}$ designates the arclength derivative, see (1.3.4.19).

┘

§1.3.4.27 (Surface gradient) For the statement of the 3D counterpart of Thm. 1.3.4.25 we need another tool: Let Σ be an orientable surface with a C^1 -parameterization $\gamma : \Pi \subset \mathbb{R}^2 \rightarrow \mathbb{R}^3$. For $f \in C^1(\Sigma)$ its **surface gradient** $\mathbf{grad}_{\Gamma} f$ is a **tangential vector field** defined as

$$(\mathbf{grad}_{\Gamma} f)(\gamma(\hat{x})) = D\gamma(\hat{x})(\mathbf{grad} F)(\hat{x}) , \quad \hat{x} \in \Pi , \quad F := f \circ \gamma . \quad (1.3.4.28)$$

The surface gradient does not depend on the parameterization.

┘

§1.3.4.29 (Integration by parts of a_W in 3D) Also the hypersingular operator in 3D is amenable to manipulations similar to those in § 1.3.4.24. Yet, technicalities are formidable and we refer to [Ste08, pp. 131-136] for the case of W , and to [SS10, Sect. 3.3.4] for the case of a general scalar second-order differential operator. [Ste08, Thm. 6.17] reads as follows:

Theorem 1.3.4.30. Integral representation of a_W in 3D

If $d = 2$, $u, v \in C_{pw}^1(\Gamma)$, then the bilinear form a_W from (1.3.4.17) induced by the hypersingular operator $W : H^{\frac{1}{2}}(\Gamma) \rightarrow H^{-\frac{1}{2}}(\Gamma)$ can be expressed as

$$a_W(u, v) = \frac{1}{4\pi} \int_{\Gamma} \int_{\Gamma} \frac{1}{\|x - y\|} (\mathbf{grad}_{\Gamma} u(y) \times \mathbf{n}(y)) \cdot (\mathbf{grad}_{\Gamma} v(x) \times \mathbf{n}(x)) dS(y) dS(x) , \quad (1.3.4.31)$$

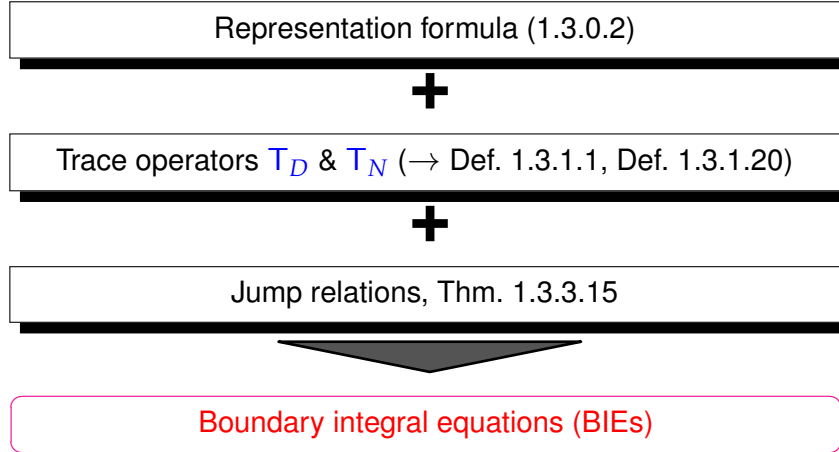
where \mathbf{grad}_{Γ} designates the surface gradient, see (1.3.4.28), and \times stands for the vector product.

┘

1.3.5 Direct Boundary Integral Equations

Now we have all the building blocks ready to devise boundary integral equations that permit us to solve boundary value problems.

The road to boundary integral equations (BIE):



In Def. 1.3.4.1 we defined boundary integral operators on $\Gamma := \partial\Omega$ by taking the average Dirichlet- and Neumann traces of the two layer potentials. To facilitate notations we now tag *traces from outside* Ω with “+”: generically T^+ , specifically T_N^+, T_D^+ . For traces from inside Ω we keep the notations T_D, T_N , and only occasionally write T_D^-, T_N^- to contrast them with exterior traces. For both T_N^-/T_N and T_D^+/T_D the normal vector n points from Ω into Ω' (exterior unit normal vector for Ω).

Using this new notation we can rewrite the definition of the boundary integral operators:

single layer BIO: $V(\phi) := \frac{1}{2} \left(T_D^+(\Psi_{SL}^\Delta(\phi)) + T_D^-(\Psi_{SL}^\Delta(\phi)) \right), \quad \phi \in H^{-\frac{1}{2}}(\Gamma),$

double layer BIO: $K(v) := \frac{1}{2} \left(T_D^+(\Psi_{DL}^\Delta(v)) + T_D^-(\Psi_{DL}^\Delta(v)) \right), \quad v \in H^{\frac{1}{2}}(\Gamma),$

adjoint double layer BIO: $K'(\phi) := \frac{1}{2} \left(T_N^+(\Psi_{SL}^\Delta(\phi)) + T_N^-(\Psi_{SL}^\Delta(\phi)) \right), \quad \phi \in H^{-\frac{1}{2}}(\Gamma),$

hypersingular BIO: $W(v) := -\frac{1}{2} \left(T_N^+(\Psi_{DL}^\Delta(v)) + T_N^-(\Psi_{DL}^\Delta(v)) \right), \quad v \in H^{\frac{1}{2}}(\Gamma).$

We combine this with the jump relations of Thm. 1.3.3.15

$$T_D^+(\Psi_{SL}^\Delta(\phi)) - T_D^-(\Psi_{SL}^\Delta(\phi)) = 0, \quad T_D^+(\Psi_{DL}^\Delta(v)) - T_D^-(\Psi_{DL}^\Delta(v)) = v.$$

$$T_N^+(\Psi_{SL}^\Delta(\phi)) - T_N^-(\Psi_{SL}^\Delta(\phi)) = -\phi, \quad T_N^+(\Psi_{DL}^\Delta(v)) - T_N^-(\Psi_{DL}^\Delta(v)) = 0.$$

Thus we can easily isolate interior and exterior traces of layer potentials:

$$T_D^-(\Psi_{SL}^\Delta(\phi)) = V(\phi), \quad T_D^+(\Psi_{SL}^\Delta(\phi)) = V(\phi), \tag{1.3.5.1a}$$

$$T_D^-(\Psi_{DL}^\Delta(v)) = -\frac{1}{2}v + K(v), \quad T_D^+(\Psi_{DL}^\Delta(v)) = \frac{1}{2}v + K(v), \tag{1.3.5.1b}$$

$$T_N^-(\Psi_{SL}^\Delta(\phi)) = \frac{1}{2}\phi + K'(\phi), \quad T_N^+(\Psi_{SL}^\Delta(\phi)) = -\frac{1}{2}\phi + K'(\phi), \tag{1.3.5.1c}$$

$$T_N^-(\Psi_{DL}^\Delta(v)) = -W(v), \quad T_N^+(\Psi_{DL}^\Delta(v)) = -W(v). \tag{1.3.5.1d}$$

Thus, when applying the trace operators to the representation formula for harmonic ($\Delta u = 0$) functions in Ω

$$u(x) = \Psi_{SL}^\Delta(T_N u) - \Psi_{DL}^\Delta(T_D u), \quad u \in H^1(\Omega), \Delta u = 0, \tag{1.3.5.2}$$

we obtain two boundary integral equations

Fundamental BIEs

[apply T_D :] $T_D u = V(T_N u) - (-\frac{1}{2}Id + K)(T_D u), \tag{1.3.5.4a}$

$$[\text{apply } T_N:] \quad T_N u = \left(\frac{1}{2}\text{Id} + K'\right)(T_N u) + W(T_D u). \quad (1.3.5.4b)$$

The boundary integral equations can be written in various **block operator forms** using the conventions of matrix×vector multiplication for operators on function spaces:

$$\begin{bmatrix} \frac{1}{2}\text{Id} - K & V \\ W & \frac{1}{2}\text{Id} + K' \end{bmatrix} \begin{bmatrix} T_D u \\ T_N u \end{bmatrix} = \begin{bmatrix} T_D u \\ T_N u \end{bmatrix} \Leftrightarrow \begin{bmatrix} \frac{1}{2}\text{Id} + K & -V \\ -W & \frac{1}{2}\text{Id} - K' \end{bmatrix} \begin{bmatrix} T_D u \\ T_N u \end{bmatrix} = 0. \quad (1.3.5.5)$$

The next result is the foundation of numerical methods relying on direct boundary integral equations, because it tells us that solutions of boundary integral equations are in one-to-one relationship to solutions of boundary value problems.

Theorem 1.3.5.6. Characterization of Cauchy data

A pair of functions $(u, \psi) \in H^{\frac{1}{2}}(\Gamma) \times H^{-\frac{1}{2}}(\Gamma)$ solves the boundary integral equations

$$\begin{bmatrix} \frac{1}{2}\text{Id} - K & V \\ W & \frac{1}{2}\text{Id} + K' \end{bmatrix} \begin{bmatrix} u \\ \psi \end{bmatrix} = \begin{bmatrix} u \\ \psi \end{bmatrix} \Leftrightarrow \begin{bmatrix} \frac{1}{2}\text{Id} + K & -V \\ -W & \frac{1}{2}\text{Id} - K' \end{bmatrix} \begin{bmatrix} u \\ \psi \end{bmatrix} = 0, \quad (1.3.5.7)$$

if and only if there is a function $u \in H^1(\Omega)$ with $\Delta u = 0$ in Ω such that

$$u = T_D u, \quad \psi = T_N u. \quad (1.3.5.8)$$

Proof. “ \Rightarrow ”: If $(u, \psi) \in H^{\frac{1}{2}}(\Gamma) \times H^{-\frac{1}{2}}(\Gamma)$ provide a solution of (1.3.5.7), then choose u according to

$$u(x) = \Psi_{\text{SL}}^\Delta(\psi)(x) - \Psi_{\text{DL}}^\Delta(u)(x), \quad x \in \Omega,$$

cf. (1.3.5.2). Lemma 1.2.5.8 and Lemma 1.2.5.15 confirm that we obtain a harmonic function. The trace matching is a direct consequence of the BIEs (1.3.5.7) and (1.3.5.1).

“ \Leftarrow ”: The BIE (1.3.5.7) are a direct consequence of the representation theorem Thm. 1.3.3.4 and (1.3.5.1). □

Remark 1.3.5.9 (BIEs for general second-order scalar differential operators) All of the above developments and results for $-\Delta$ carry over to scalar second-order differential operators with constant coefficients, cf. (1.2.0.1), with suitable fundamental solutions and an altered definition of T_N , of course, see Rem. 1.3.1.42. ┘

1.3.5.1 First-kind BIEs

§1.3.5.10 (Model boundary value problems) Our goal is to solve either of the following two “canonical” boundary value problems (BVPs) for the Laplacian $-\Delta$, which we give in strong form, though we usually consider weak (variational) solutions.

◆ **Dirichlet BVP:** given $g \in H^{\frac{1}{2}}(\Gamma)$ find $u \in H^1(\Omega)$ such that

$$-\Delta u = 0 \quad \text{in } \Omega, \quad T_D u = g \quad \text{on } \Gamma. \quad (1.3.5.11)$$

◆ **Neumann BVP:** given $\eta \in H_*^{-\frac{1}{2}}(\Gamma)$ determine $u \in H_*^1(\Omega)$ such that

$$-\Delta u = 0 \quad \text{in } \Omega, \quad \mathbb{T}_N u = \eta \quad \text{on } \Gamma. \quad (1.3.5.12)$$

The “*-spaces” are defined as spaces of functions with vanishing average:

$$H_*^{-\frac{1}{2}}(\Gamma) := \left\{ \phi \in H^{-\frac{1}{2}}(\Gamma) : \int_{\Gamma} \phi(x) \, dS(x) = 0 \right\},$$

$$H_*^1(\Omega) := \left\{ v \in H^1(\Omega) : \int_{\Omega} v \, dx = 0 \right\}.$$

This choice reflects

- the failure of the pure Neumann problem (1.3.5.12) to possess a unique solution (adding an arbitrary constant yields another solution),
- the corresponding compatibility condition on the Neumann data η [NumPDE Ex. 1.8.0.10].

┘

Now we formulate BIEs related to these boundary value problems for $-\Delta$. Of course, we cannot solve for the function $u \in H^1(\Omega)$, because BIEs are set in trace spaces. Rather, we consider a BVP solved in the sense of BIEs, if both the Dirichlet trace $\mathbb{T}_D u$ and the Neumann trace $\mathbb{T}_N u$ of the solution have been found. Then u can be recovered in a **post-processing step** based on evaluating the representation formula (1.3.5.2).

§1.3.5.13 (First-kind BIEs for the Dirichlet problem) In the case of (1.3.5.11) we have to find the unknown Neumann trace $\mathbb{T}_N u \in H^{-\frac{1}{2}}(\Gamma)$. Since $\mathbb{T}_D u = \mathfrak{g}$ is known, we can get it from the BIE (1.3.5.4a)

$$\mathbb{V}(\mathbb{T}_N u) = \left(\frac{1}{2}\text{Id} + \mathbb{K}\right)\mathfrak{g} \quad \text{in } H^{\frac{1}{2}}(\Gamma). \quad (1.3.5.14)$$

Due to the mapping property $\mathbb{V} : H^{-\frac{1}{2}}(\Gamma) \rightarrow H^{\frac{1}{2}}(\Gamma)$ and by the L^2 -duality of $H^{\frac{1}{2}}(\Gamma)$ and $H^{-\frac{1}{2}}(\Gamma)$, see Thm. 1.3.1.39 and (1.3.1.41a),

$$u, v \in H^{\frac{1}{2}}(\Gamma): \quad u = v \quad \Leftrightarrow \quad \int_{\Gamma} (u - v)(x) \phi(x) \, dS(x) = 0 \quad \forall \phi \in H^{-\frac{1}{2}}(\Gamma), \quad (1.3.1.41a)$$

this operator equation has a natural *equivalent* variational form (VF):

$$\psi \in H^{-\frac{1}{2}}(\Gamma): \quad a_V(\psi, \phi) = \int_{\Gamma} \left(\frac{1}{2}\text{Id} + \mathbb{K}\right)\mathfrak{g}(x) \phi(x) \, dS(x) \quad \forall \phi \in H^{-\frac{1}{2}}(\Gamma), \quad (1.3.5.15)$$

$$a_V(\psi, \phi) := \int_{\Gamma} \mathbb{V}(\psi)(x) \phi(x) \, dS(x). \quad (1.3.5.16)$$

The bilinear form $a_V : H^{-\frac{1}{2}}(\Gamma) \times H^{-\frac{1}{2}}(\Gamma) \rightarrow \mathbb{R}$ is clearly symmetric and bounded by Thm. 1.3.4.2 and Thm. 1.3.1.39. If we can show that it defines an **equivalent inner product** on $H^{-\frac{1}{2}}(\Gamma)$, also called $H^{-\frac{1}{2}}(\Gamma)$ -elliptic, then the Riesz representation theorem will guarantee unique solvability of (1.3.5.15). In 3D the next theorem confirms this. An in-depth discussion is given in [Ste08, Sect. 6.6.1].

Theorem 1.3.5.17. Ellipticity of a_V in 3D

For $d = 3$ the bilinear form a_V is $H^{-\frac{1}{2}}(\Gamma)$ -elliptic:

$$\exists C > 0: \quad |a_V(\phi, \phi)| \geq C \|\phi\|_{H^{-\frac{1}{2}}(\Gamma)}^2 \quad \forall \phi \in H^{-\frac{1}{2}}(\Gamma). \quad (1.3.5.18)$$

Proof. For $d = 3$ the decay conditions (1.1.7.1) satisfied by the single layer potential Ψ_{SL}^Δ ensure that the pairing identity (1.3.1.35) holds for both domains Ω and Ω' , no matter whether they are bounded or unbounded:

$$\int_{\Gamma} (\mathbb{T}_N \Psi_{\text{SL}}^\Delta(\phi))(x) (\mathbb{T}_D \Psi_{\text{SL}}^\Delta(\phi))(x) \, dS(x) = \int_{\Omega} \|\mathbf{grad} \Psi_{\text{SL}}^\Delta(\phi)(x)\|^2 \, dx. \quad (1.3.5.19)$$

Based on the jump relations for $\Psi_{\text{SL}}^\Delta(\phi)$ we deduce from (1.3.5.19)

$$\begin{aligned} \int_{\Gamma} \mathbb{V}(\phi) \phi(x) \, dS(x) &= - \int_{\Gamma} \mathbb{T}_D(\Psi_{\text{SL}}^\Delta(\phi))(x) \left[\mathbb{T}_N \Psi_{\text{SL}}^\Delta(\phi) \right]_{\Gamma}(x) \, dS(x) \\ &= \frac{1}{2} \int_{\Gamma} \mathbb{T}_D(\Psi_{\text{SL}}^\Delta(\phi))(x) \mathbb{T}_N(\Psi_{\text{SL}}^\Delta(\phi))(x) - \mathbb{T}_D^+(\Psi_{\text{SL}}^\Delta(\phi))(x) \mathbb{T}_N^+(\Psi_{\text{SL}}^\Delta(\phi))(x) \, dS(x) \\ &= \frac{1}{2} \left(\left| \Psi_{\text{SL}}^\Delta(\phi) \right|_{H^1(\Omega)}^2 + \left| \Psi_{\text{SL}}^\Delta(\phi) \right|_{H^1(\Omega')}^2 \right) = \|\phi\|_{H^{-\frac{1}{2}}(\Gamma)}^2, \end{aligned}$$

thanks to Def. 1.3.1.27, which means

$$\left| \Psi_{\text{SL}}^\Delta(\phi) \right|_{H^1(\Omega)}^2 + \left| \Psi_{\text{SL}}^\Delta(\phi) \right|_{H^1(\Omega')}^2 = \|\tilde{\phi}\|_{\tilde{H}^{-1}(\Omega)}^2 = \|\phi\|_{H^{-\frac{1}{2}}(\Gamma)}^2. \quad (1.3.5.20)$$

Note that \mathbb{T}_N^+ employs a normal vector field oriented opposite to the exterior normal vector field of Ω' . This explains the flipping of signs in the above manipulations. □

The poor decay properties of $\Psi_{\text{SL}}^\Delta(\phi)$ in 2D thwart (1.3.5.19). Nevertheless, the following result is available.

Theorem 1.3.5.21. Ellipticity of $a_{\mathbb{V}}$ in 2D

For $d = 2$ the bilinear form $a_{\mathbb{V}}$ is only $H_*^{-\frac{1}{2}}(\Gamma)$ -elliptic.

If $\text{diam} \Omega < 1$ then $a_{\mathbb{V}}$ is $H^{-\frac{1}{2}}(\Gamma)$ -elliptic also for $d = 2$.

► The variational problem

$$\begin{aligned} \psi \in H^{-\frac{1}{2}}(\Gamma): \quad a_{\mathbb{V}}(\psi, \phi) &= \int_{\Gamma} \left(\frac{1}{2}\text{Id} - \mathbb{K}\right) \mathbf{g}(x) \phi(x) \, dS(x) \quad \forall \phi \in H^{-\frac{1}{2}}(\Gamma), \quad (1.3.5.15) \\ a_{\mathbb{V}}(\psi, \phi) &:= \int_{\Gamma} \mathbb{V}(\psi)(x) \phi(x) \, dS(x). \end{aligned}$$

has a unique solution ψ for any $\mathbf{g} \in H^{\frac{1}{2}}(\Gamma)$, provided that for $d = 2$ we have $\text{diam} \Omega < 1$, because in this case $a_{\mathbb{V}}$ provides an inner product for $H^{-\frac{1}{2}}(\Gamma)$. ┘

§1.3.5.22 (First-kind BIEs for the Neumann problem) In (1.3.5.12) the Neumann trace $\eta \in H_*^{-\frac{1}{2}}(\Gamma)$ is given and we seek the unknown Dirichlet trace $\mathbb{T}_D u$ of the solution u . From (1.3.5.4b) we get

$$\mathbb{W}(\mathbb{T}_D u) = \left(\frac{1}{2}\text{Id} - \mathbb{K}'\right) \eta \quad \text{in} \quad H^{-\frac{1}{2}}(\Delta). \quad (1.3.5.23)$$

Invoking the duality relationship

$$\phi, \psi \in H^{-\frac{1}{2}}(\Gamma): \quad \psi = \phi \iff \int_{\Gamma} (\psi - \phi)(x) \mathbf{v}(x) \, dS(x) = 0 \quad \forall \mathbf{v} \in H^{\frac{1}{2}}(\Gamma), \quad (1.3.1.41b)$$

an equivalent variational formulation (VF) of (1.3.5.23) is

$$\begin{aligned} \mathbf{u} \in H_*^{\frac{1}{2}}(\Gamma): \quad a_W(\mathbf{u}, \mathbf{v}) &= \int_{\Gamma} \left(\frac{1}{2}\text{Id} - K'\right)\eta(\mathbf{x}) \mathbf{v}(\mathbf{x}) \, dS(\mathbf{x}) \quad \forall \mathbf{v} \in H_*^{\frac{1}{2}}(\Gamma), \\ a_W(\mathbf{u}, \mathbf{v}) &:= \int_{\Gamma} W(\mathbf{u})(\mathbf{x}) \mathbf{v}(\mathbf{x}) \, dS(\mathbf{x}), \end{aligned} \quad (1.3.5.24)$$

where $H_*^{\frac{1}{2}}(\Gamma) := \left\{ v \in H^{\frac{1}{2}}(\Gamma) : \int_{\Gamma} v(\mathbf{x}) \, dS(\mathbf{x}) = 0 \right\}$.

The need to restrict trial and test functions to the space $H_*^{\frac{1}{2}}(\Gamma)$ of functions with vanishing mean is clear from the representations (1.3.4.26) and (1.3.4.31). They imply

$$a_W(\mathbf{u}, \mathbf{v}) = 0 \quad \forall \mathbf{v} \in H_*^{\frac{1}{2}}(\Gamma) \quad \Leftrightarrow \quad \mathbf{u} \equiv \text{const.} \quad . \quad (1.3.5.25)$$

On the complement of its kernel a_W enjoys ellipticity, see [Ste08, Sect. 6.6.2] for details.

Theorem 1.3.5.26. Ellipticity of a_W

The bilinear form a_W induced by the hypersingular boundary integral operator $W : H_*^{\frac{1}{2}}(\Gamma) \rightarrow H_*^{-\frac{1}{2}}(\Gamma)$ is $H_*^{\frac{1}{2}}(\Gamma)$ -elliptic

$$\exists C > 0: \quad |a_W(\mathbf{v}, \mathbf{v})| \geq C \|\mathbf{v}\|_{H_*^{\frac{1}{2}}(\Gamma)}^2 \quad \forall \mathbf{v} \in H_*^{\frac{1}{2}}(\Gamma). \quad (1.3.5.27)$$

► The variational problem (1.3.5.24) has a unique solution $\mathbf{u} \in H_*^{\frac{1}{2}}(\Gamma)$ for any $\eta \in H_*^{-\frac{1}{2}}(\Gamma)$. ┘

Remark 1.3.5.28 (“First-kind”) Boundary integral equations are of the **first kind** if the mapping properties of the boundary integral operator on the left-hand side support a natural variational formulation in energy trace spaces via duality. Examples are (1.3.5.14) and (1.3.5.23). ┘

1.3.5.2 Second-kind BIEs

You might have been wondering why we simply ignored the second equation of (1.3.5.4) when treating the Dirichlet problem in § 1.3.5.13, and why we skipped the first equation in the case of the Neumann problem in § 1.3.5.22. The reason was that using these other equations will not result in a first-kind BIE. Now we study what we get from them.

§1.3.5.29 (Second-kind BIE for the Dirichlet problem) We consider the boundary value problem 1.3.5.11. Knowing $\mathbf{g} = T_D u$ we have to determine $\psi := T_N u$. From (1.3.5.4b) we extract the BIE

$$\left(\frac{1}{2}\text{Id} - K'\right)\psi = W(\mathbf{g}) \quad \text{in } H_*^{-\frac{1}{2}}(\Gamma). \quad (1.3.5.30)$$

In light of the duality of Thm. 1.3.1.39, (1.3.1.41), a natural variational formulation of 1.3.5.30 is

$$\psi \in H_*^{-\frac{1}{2}}(\Gamma): \quad \int_{\Gamma} \left(\left(\frac{1}{2}\text{Id} - K'\right)\psi\right)(\mathbf{x}) \mathbf{v}(\mathbf{x}) \, dS(\mathbf{x}) = \int_{\Gamma} (W\mathbf{g})(\mathbf{x}) \mathbf{v}(\mathbf{x}) \, dS(\mathbf{x}) [= a_W(\mathbf{g}, \mathbf{v})] \quad \forall \mathbf{v} \in H_*^{\frac{1}{2}}(\Gamma). \quad (1.3.5.31)$$

┘

§1.3.5.32 (Second-kind BIE for the Neumann problem) We want to solve the Neumann boundary value problem (1.3.5.12) by finding the unknown Dirichlet data $\mathbf{u} := \mathbf{T}_D \mathbf{u}$. We use (1.3.5.4a) and end up with the BIE

$$\left(\frac{1}{2}\text{Id} + \mathbf{K}\right)\mathbf{u} = \mathbf{V}\boldsymbol{\eta} \quad \text{in } H^{\frac{1}{2}}(\Gamma). \tag{1.3.5.33}$$

The duality (1.3.1.41a) yields the equivalent variational equation:

$$\mathbf{u} \in H_*^{\frac{1}{2}}(\Gamma): \int_{\Gamma} \left(\left(\frac{1}{2}\text{Id} + \mathbf{K}\right)\mathbf{u}\right)(\mathbf{x}) \phi(\mathbf{x}) \, dS(\mathbf{x}) = \int_{\Gamma} (\mathbf{V}\boldsymbol{\eta})(\mathbf{x}) \phi(\mathbf{x}) \, dS(\mathbf{x}) [\mathbf{a}_V(\boldsymbol{\eta}, \phi)] \quad \forall \phi \in H_*^{-\frac{1}{2}}(\Lambda). \tag{1.3.5.34}$$

Note the different trial and test spaces both in (??) and (1.3.5.34). ┘

§1.3.5.35 (Variational formulations in $L^2(\Gamma)$) Unfortunately the variational formulations (1.3.5.31) and (1.3.5.34) share the undesirable (from the point of view of Galerkin discretization) feature that *trial and test spaces do not coincide*.

This can be remedied by switching to variational formulations in $L^2(\Gamma)$. We multiply the BIEs (1.3.5.30) and (1.3.5.33) with a test function $w \in L^2(\Gamma)$ and integrate over Γ . When also using $L^2(\Gamma)$ as trial space, we end up with

$$\boldsymbol{\psi} \in L^2(\Gamma): \int_{\Gamma} \left(\left(\frac{1}{2}\text{Id} - \mathbf{K}'\right)\boldsymbol{\psi}\right)(\mathbf{x}) w(\mathbf{x}) \, dS(\mathbf{x}) = \int_{\Gamma} (\mathbf{W}\mathbf{g})(\mathbf{x}) w(\mathbf{x}) \, dS(\mathbf{x}) \quad \forall w \in L^2(\Gamma), \tag{1.3.5.36}$$

$$\mathbf{u} \in L_*^2(\Gamma): \int_{\Gamma} \left(\left(\frac{1}{2}\text{Id} + \mathbf{K}\right)\mathbf{u}\right)(\mathbf{x}) w(\mathbf{x}) \, dS(\mathbf{x}) = \int_{\Gamma} (\mathbf{V}\boldsymbol{\eta})(\mathbf{x}) w(\mathbf{x}) \, dS(\mathbf{x}) \quad \forall w \in L_*^2(\Gamma), \tag{1.3.5.37}$$

with $L_*^2(\Gamma) := \{v \in L^2(\Gamma) : \int_{\Gamma} v(\mathbf{x}) \, dS(\mathbf{x}) = 0\}$. Note that assuming $\mathbf{g} \in H^1(\Gamma)$, thanks to Thm. 1.3.4.6 these variational equations are meaningful (right-hand and left-hand sides are continuous on $L^2(\Gamma)$). Yet the bilinear forms occurring in (1.3.5.36) and (1.3.5.37) are neither symmetric nor elliptic. Results on existence and uniqueness of solutions of the BIEs (1.3.5.30) and (1.3.5.33), and the variational equations (1.3.5.36) and (1.3.5.37) required profound mathematical tools [Ste08, Sect. 6.6.4]. ┘

Remark 1.3.5.38 (“Second-kind”) Boundary integral equations of the **second kind** are distinguished by a left-hand side operator of the form $c\text{Id} + \mathbf{T}$, where $c \neq 0$ and \mathbf{T} is a continuous operator in L^2 . Obviously, the BIEs (1.3.5.33) and (1.3.5.34) are of this type. ┘

1.3.6 Indirect Boundary Integral Equations

In the previous sections we used the fundamental result of Thm. 1.3.5.6 to obtain (variational) boundary integral equations. Now we boldly “guess” a formula for the solutions of Dirichlet and Neumann boundary value problems (1.3.5.11) and (1.3.5.12) and justify it a posteriori.

We start by recalling from Lemma 1.2.5.8 and Lemma 1.2.5.15 that

- ✓ $\Delta \Psi_{\text{SL}}^{\Delta} = \Delta \Psi_{\text{DL}}^{\Delta} = 0 \quad \text{in } \mathbb{R}^d \setminus \Gamma,$
 - ✓ $\Psi_{\text{SL}}^{\Delta}$ and $\Psi_{\text{DL}}^{\Delta}$ satisfy “decay conditions at ∞ ”.
- (1.3.6.1)

► Idea: Use **trial expressions** based on layer potentials:



$$u = \Psi_{\text{SL}}^{\Delta}(\phi) \quad \text{or} \quad u = \Psi_{\text{DL}}^{\Delta}(f) \tag{1.3.6.2}$$

with unknown functions $\phi, f : \Gamma \rightarrow \mathbb{R}$ for the solution u of the boundary value problems (1.3.5.11) and (1.3.5.12).

Be aware that at this point we have no guarantee that the weak solution $u \in H^1(\Omega)$ of the boundary value problems allows any of the representations from (1.3.6.2). Strictly speaking, once we have proposed a way how to determine ϕ or f we have to proof that the trial expression really satisfies the boundary conditions.

§1.3.6.3 (Indirect first-kind BIE for the Dirichlet problem) For the Dirichlet problem: given $g \in H^{\frac{1}{2}}(\Gamma)$ find $u \in H^1(\Omega)$ such that

$$-\Delta u = 0 \text{ in } \Omega, \quad T_D u = g \text{ on } \Gamma, \tag{1.3.5.11}$$

we try

$$u = \Psi_{SL}^\Delta(\phi).$$

We impose the prescribed trace by applying T_D and use (1.3.5.1a), $T_D \Psi_{SL}^\Delta(\phi) = V(\phi)$:

$$\blacktriangleright \text{ BIE: } V(\phi) = g \text{ in } H^{\frac{1}{2}}(\Gamma). \tag{1.3.6.4}$$

By duality we obtain the natural variational formulation of this BIE in energy trace space:

$$\phi \in H^{-\frac{1}{2}}(\Gamma): \quad a_V(\phi, \psi) = \int_\Gamma g(x) \psi(x) dS(x) \quad \forall \psi \in H^{-\frac{1}{2}}(\Gamma). \tag{1.3.6.5}$$

Notice that this variational problem is based on the same bilinear form a_V as the first-kind variational formulation (1.3.5.15).

Theorem 1.3.6.6. Validity of 1st-kind indirect BIE for Dirichlet problem

In the case $d = 2$ assume $\text{diam}(\Omega) < 1$. Then $u = \Psi_{SL}^\Delta(\phi)$ solves (1.3.5.11) for the unique solution $\phi \in H^{-\frac{1}{2}}(\Gamma)$ of (1.3.6.5).

Proof. Existence and uniqueness of a solution $\phi \in H^{-\frac{1}{2}}(\Gamma)$ of (1.3.6.5) follows from Thm. 1.3.5.17 and Thm. 1.3.5.21. That u complies with the boundary conditions is built into the BIE (1.3.6.4). □

§1.3.6.7 (Indirect first-kind BIE for the Neumann problem) We consider the Neumann problem: given $\eta \in H_*^{-\frac{1}{2}}(\Gamma)$ determine $u \in H_*^1(\Omega)$ such that

$$-\Delta u = 0 \text{ in } \Omega, \quad T_N u = \eta \text{ on } \Gamma, \tag{1.3.5.12}$$

we try

$$u = \Psi_{DL}^\Delta(f).$$

To enforce the prescribed Neumann trace on u apply T_N and use (1.3.5.1d):

$$\blacktriangleright \text{ BIE: } W(f) = \eta \text{ in } H_*^{-\frac{1}{2}}(\Gamma). \tag{1.3.6.8}$$

Duality yields the natural variational formulation in energy trace spaces

$$f \in H_*^{\frac{1}{2}}(\Gamma): \quad a_W(f, v) = \int_\Gamma \eta(x) v(x) dS(x) \quad \forall v \in H_*^{\frac{1}{2}}(\Gamma). \tag{1.3.6.9}$$

Again, we have arrived at a variational formulation involving the same bilinear form a_W and trace spaces as the first-kind variational problem (1.3.5.24).

Theorem 1.3.6.10. Validity of 1st-kind indirect BIE for Neumann problem

If $f \in H_^{\frac{1}{2}}(\Gamma)$ is the unique solution of (1.3.6.9), then $u := \Psi_{DL}^\Delta(f)$ solves the Neumann problem (1.3.5.12).*

Proof. The assertion is immediate from Thm. 1.3.5.26 and the construction of the BIE (1.3.6.8). \square ┘

Remark 1.3.6.11 (Meaning of “density unknowns” ϕ and \mathbf{v}) The unknown functions $\phi \in H^{-\frac{1}{2}}(\Gamma)$ in (1.3.6.4) and $\mathbf{v} \in H_*^{\frac{1}{2}}(\Gamma)$ in (1.3.6.8) do **not** agree with any **trace** of the solution u of the related BVP; they are called **densities**.

However, there is a relationship with traces that we elaborate for (1.3.6.4). By the jump relations of Thm. 1.3.3.15 we have for $u = \Psi_{\text{SL}}^{\Delta}(\phi)$.

$$\begin{aligned} \textcircled{1} \quad & \llbracket T_N u \rrbracket_{\Gamma} = \llbracket T_N \Psi_{\text{SL}}^{\Delta}(\phi) \rrbracket_{\Gamma} = -\phi \quad \text{on } \Gamma, \\ \textcircled{2} \quad & T_D^- u = T_D^+ u = \mathbf{v}(\phi) = \mathbf{g} \quad \text{on } \Gamma. \end{aligned}$$

- ▶ The solution $\phi \in H^{-\frac{1}{2}}(\Gamma)$ of the indirect 1st-kind BIE (1.3.6.4) coincides with the **jump across Γ of the Neumann trace** of the solutions of the Dirichlet BVPs (with data \mathbf{g}) on Ω and Ω' .
- ▶ The solution $\mathbf{f} \in H_*^{\frac{1}{2}}(\Gamma)$ of the indirect 1st-kind BIE (1.3.6.8) coincides with the **jump across Γ of the Dirichlet trace** of the solutions of the Neumann BVPs (with data η) on Ω and Ω' . ┘

1.4 Boundary Element Methods in Two Dimensions

§1.4.0.1 (A C++ 2D BEM code →GITLAB) To demonstrate principles of implementation of 2D BEM we rely on a C++ port by C. Urzua (formerly, SAM, ETH Zurich, now University of Graz, Austria) of the MATLAB BEM code **HILBERT** [Aur+14] developed in the group of **D. Praetorius** at TU Wien.

The C++ library provides functions for the assembly of boundary element Galerkin matrices that will be used for homework coding projects. Meshes (\rightarrow Def. 1.4.2.5) of a closed connected curve $\Gamma := \partial\Omega$, $\Omega \subset \mathbb{R}^2$ are stored in **BoundaryMesh** objects, see Code 1.4.2.57. In the sequel let $n_V \in \mathbb{N}$ and n_E denote the number of vertices and panels of the current mesh \mathcal{G} .

- **void** computeV(**Eigen::MatrixXd**& V, **const** BoundaryMesh& mesh, **double** eta)

This function constructs the Galerkin matrix $\mathbf{V} \in \mathbb{R}^{n_E, n_E}$ for the bilinear form a_V induced by the *single layer BIO V*, using $\mathcal{S}_0^{-1}(\mathcal{G})$ as test and trial space, equipped with the characteristic functions $\beta_N^i \in \mathcal{S}_0^{-1}(\mathcal{G})$, $i = 1, \dots, n_E$, of panels as basis, see Ex. 1.4.2.17.

$$(\mathbf{V})_{ij} = a_V(\beta_N^j, \beta_N^i) = -\frac{1}{2\pi} \int_{\pi_i} \int_{\pi_j} \log \|x - y\| dS(y) dS(x), \quad i, j = 1, \dots, n_E. \quad (1.4.0.2)$$

Here and below, the input argument `eta` is the so-called admissibility parameter and defines which entries are to be computed analytically (as in Section 1.4.3.2) or semi-analytically using numerical quadrature for some of the integrals. Specifying `eta=0.0` selects analytic formulas throughout.

- **void** computeW(**Eigen::MatrixXd**& W, **const** BoundaryMesh& mesh, **double** eta)

This function builds the Galerkin matrix $\mathbf{W} \in \mathbb{R}^{n_V, n_V}$ of the bilinear form a_W induced by the *hypersingular BIO W*, using $\mathcal{S}_1^0(\mathcal{G})$ as test and trial spaces, endowed with the “tent function” basis $\{b_N^1, \dots, b_N^{n_V}\}$, see Ex. 1.4.2.19. The matrix entries are

$$(\mathbf{W})_{ij} = a_W(b_N^j, b_N^i) = -\frac{1}{2\pi} \int_{\Gamma} \int_{\Gamma} \log \|x - y\| \frac{db_N^j}{ds}(y) \frac{db_N^i}{ds}(x) dS(y) dS(x), \quad (1.4.0.3)$$

for $i, j = 1, \dots, n_V$.

- **void** computeK(Eigen::MatrixXd& K, **const** BoundaryMesh& mesh, **double** eta)

This function assembles the Galerking matrix $\mathbf{K} \in \mathbb{R}^{n_E n_V}$ of the bilinear form induced by the *double layer BIO K*, using $\mathcal{S}_0^{-1}(\mathcal{G})$ and $\mathcal{S}_1^0(\mathcal{G})$ as test and trial spaces, respectively. The standard nodal bases from Ex. 1.4.2.17 and Ex. 1.4.2.19 are employed and we get for the matrix entries

$$(\mathbf{K})_{ij} = a_K(b_N^j, \beta_N^i) = -\frac{1}{2\pi} \int_{\pi_i} \int_{\text{supp } b_N^j} \frac{\mathbf{x} - \mathbf{y}}{\|\mathbf{x} - \mathbf{y}\|^2} \cdot \mathbf{n}(\mathbf{y}) b_N^j(\mathbf{y}) dS(\mathbf{y}) dS(\mathbf{x}), \quad (1.4.0.4)$$

for $i \in \{1, \dots, n_E\}, j \in \{1, \dots, n_V\}$.

- **void** computeK00(Eigen::MatrixXd& K, **const** BoundaryMesh& mesh, **double** eta)

This function assembles the Galerking matrix $\mathbf{K} \in \mathbb{R}^{n_E n_E}$ of the bilinear form induced by the *double layer BIO K*, using $\mathcal{S}_0^{-1}(\mathcal{G})$ as test and trial space, equipped with the characteristic functions of panels as basis.

$$(\mathbf{K0})_{ij} = a_K(\beta_N^j, \beta_N^i) = -\frac{1}{2\pi} \int_{\pi_i} \int_{\pi_j} \frac{\mathbf{x} - \mathbf{y}}{\|\mathbf{x} - \mathbf{y}\|^2} \cdot \mathbf{n}(\mathbf{y}) dS(\mathbf{y}) dS(\mathbf{x}), \quad (1.4.0.5)$$

for $i, j = 1, \dots, n_E$.

- **void** computeM01(Eigen::SparseMatrix<double> &M, **const** BoundaryMesh& mesh)

This function creates the so-called mass matrix $\mathbf{M} \in \mathbb{R}^{n_E n_V}$ as defined in (1.4.2.39c). (Note that for this case you need to initialize the matrix passed in M *with its size* before calling this function). Please consult [NumCSE Section 2.7.2] to learn about data structures for sparse matrices in EIGEN.

- **void** computeM00(Eigen::SparseMatrix<double> &M, **const** BoundaryMesh& mesh)

This function creates another mass matrix $\mathbf{M}_0 \in \mathbb{R}^{n_E n_E}$, a Galerkin matrix for the $L^2(\Gamma)$ -inner product using $\mathcal{S}_0^{-1}(\mathcal{G})$ as trial and test space (with the standard nodal basis consisting of characteristic functions of panels). As before, you must initialize the matrix M *with its size* $n_E \times n_E$.

We refer to the Doxygen documentation of the library for further details on the implementation of these methods. ┘

1.4.1 Abstract Galerkin Discretization

Regardless of whether we tackle the first-kind variational boundary integral equations (1.3.5.15)/(1.3.5.24) set in energy trace space or the second-kind versions (1.3.5.36)/(1.3.5.37), we face **linear variational problems** (\rightarrow Def. 1.1.5.1)

$$u \in V: \quad a(u, v) = \ell(v) \quad \forall v \in V_0, \quad (1.1.5.2)$$

posed on function spaces $V = V_0$ on Γ in each case. In this section we recall from [NumPDE Section 2.2] the policy of **Galerkin discretization** as an abstract approach for the approximate solution of linear variational problems on infinite-dimensional spaces.

Galerkin approximation

Idea of **Galerkin approximation**:

Replace V_0 in (1.1.5.2) with a **finite dimensional subspace** V_N .
 ($V_N \subset V_0$ called Galerkin (or discrete) trial space/test space)

Notation: Twofold nature of symbol “ N ”, cf. [NumPDE ??]:

- ◆ N = formal index, tagging “discrete entities” (\rightarrow “finite amount of information”)
- ◆ $N = \dim V_N \in \mathbb{N} \hat{=}$ dimension of Galerkin trial/test space



Discrete variational problem (DVP), cf. [NumPDE ??],

$$u_N \in V_N: a(u_N, v_N) = \ell(v_N) \quad \forall v_N \in V_N. \tag{1.4.1.2}$$

Galerkin solution

The discrete variational problem is “discrete” in the sense that it involves only a finite number N of degrees of freedom, but it is still not amenable to direct implementation. To that end, it has to be recast as a linear system of equations (LSE), which can be accomplished as follows:

Second step of Galerkin discretization

Recall from [NumPDE ??]: 2nd step of Galerkin discretization:

Introduce (ordered) **basis** \mathfrak{B}_N of V_N :

$$\mathfrak{B}_N := \{b_N^1, \dots, b_N^N\} \subset V_N, \quad V_N = \text{Span}\{\mathfrak{B}_N\}, \quad N := \dim(V_N).$$

► **Unique** basis expansions:

$$\begin{aligned} u_N &= \mu_1 b_N^1 + \dots + \mu_N b_N^N, \quad \mu_i \in \mathbb{R} \\ v_N &= \nu_1 b_N^1 + \dots + \nu_N b_N^N, \quad \nu_i \in \mathbb{R} \end{aligned} \quad : \quad \text{plug into (1.4.1.2).}$$

Remark 1.4.1.4 (Affine space V) In Section 1.1.5 we saw the use of an affine space $V = g + V_0$ with a so-called **offset function** g (\rightarrow [NumPDE Def. 5.2.2.20]) in order to impose essential boundary conditions in (1.1.5.5). Since the boundary integral equations that we have encountered so far do not involve any “essential conditions” to be taken into account in the trial trace, we will have $V = V_0$ in the sequel. \lrcorner

The derivation of a linear system of equations **equivalent** to (1.4.1.2) boils down to inserting the unique basis expansions into (1.4.1.2) and exploiting the linearity of both a and ℓ .

$$u_N \in V_{0,N}: a(u_N, v_N) = \ell(v_N) \quad \forall v_N \in V_N. \tag{1.4.1.2}$$

$$\Leftrightarrow \left[\begin{aligned} u_N &= \mu_1 b_N^1 + \dots + \mu_N b_N^N, \mu_i \in \mathbb{R} \\ v_N &= \nu_1 b_N^1 + \dots + \nu_N b_N^N, \nu_i \in \mathbb{R} \end{aligned} \right]$$

$$\sum_{k=1}^N \sum_{j=1}^N \mu_k \nu_j a(b_N^k, b_N^j) = \sum_{j=1}^N \nu_j \ell(b_N^j) \quad \forall \nu_1, \dots, \nu_N \in \mathbb{R},$$

$$\begin{aligned} & \Leftrightarrow \\ & \sum_{j=1}^N v_j \left(\sum_{k=1}^N \mu_k a(b_N^k, b_N^j) - \ell(b_N^j) \right) = 0 \quad \forall v_1, \dots, v_N \in \mathbb{R}, \\ & \Leftrightarrow (*) \\ & \sum_{k=1}^N \mu_k a(b_N^k, b_N^j) = \ell(b_N^j) \quad \text{for } j = 1, \dots, N. \\ & \Leftrightarrow [\vec{\mu} = (\mu_1, \dots, \mu_N)^T \in \mathbb{R}^N] \\ & \boxed{\mathbf{A} \vec{\mu} = \vec{\varphi}}, \text{ with } \mathbf{A} = \left(a(b_N^k, b_N^j) \right)_{j,k=1}^N \in \mathbb{R}^{N,N}, \\ & \vec{\varphi} = \left(\ell(b_N^j) \right)_{j=1}^N. \end{aligned}$$

A linear system of equations

Summary: notions connected with Galerkin discretization

Linear discrete variational problem $u_N \in V_N: a(u_N, v_N) = \ell(v_N) \quad \forall v_N \in V_N$
Choosing basis \mathfrak{B}_N
Linear system of equations $\mathbf{A} \vec{\mu} = \vec{\varphi}$

Galerkin matrix (GalM): $\mathbf{A} = \left(a(b_N^k, b_N^j) \right)_{j,k=1}^N \in \mathbb{R}^{N,N},$

Right hand side vector: $\vec{\varphi} = \left(\ell(b_N^j) \right)_{j=1}^N \in \mathbb{R}^N,$

Coefficient vector: $\vec{\mu} = (\mu_1, \dots, \mu_N)^T \in \mathbb{R}^N,$

Recovery of solution: $u_N = \sum_{k=1}^N \mu_k b_N^k.$

Assuming exact arithmetic, the second step of Galerkin discretization is a “mere aspect of implementation” and will not affect the quality of the Galerkin solution.

Theorem 1.4.1.6. Independence of Galerkin solution of choice of basis [NumPDE Thm. 2.2.2.6]

The choice of the basis \mathfrak{B} has no impact on the (set of) Galerkin solutions u_N of (1.4.1.2).

1.4.2 Boundary Element Spaces on Curves

Now we are concerned with defining suitable trial and test spaces for the Galerkin discretization of the variational BIEs. We seek “simple” finite-dimensional subspaces of the energy trace spaces $H^{\frac{1}{2}}(\Gamma)$ (\rightarrow Def. 1.3.1.6), $H^{-\frac{1}{2}}(\Gamma)$ (\rightarrow Def. 1.3.1.27) for the first-kind BIEs (1.3.5.15) and (1.3.5.24), and of $L^2(\Gamma)$ for the second-kind BIEs (1.3.5.36) and (1.3.5.37).

The new Galerkin trial and test spaces will be called **boundary element (BE) spaces** and will be of a “piecewise polynomial type”. The construction of these spaces will rely on many of the principles underlying the design of finite element spaces in 1D, see [NumPDE ??]. This reflects a rather general relationship.

Boundary element methods (**BEM**)
= Finite element methods (**FEM**) for variational BIEs on curves and surfaces

§1.4.2.1 (Main ingredients of FEM) In light of the above relationships it is useful to recall the building blocks of FEM from [NumPDE Section 2.5]:

- ◆ A **mesh/triangulation** of the computational domain, see [NumPDE Section 2.5.1], in particular [NumPDE Def. 2.5.1.1],
- ◆ local **polynomial spaces** defined on the cells of the mesh, see [NumPDE Section 2.5.2],
- ◆ and local and global **shape functions** (→ [NumPDE Section 2.5.3], [NumPDE Def. 2.5.3.4]) providing bases \mathfrak{B}_N of the finite element space V_N .

Another fundamental paradigm in the field of finite element methods is the **parametric construction** of finite element spaces based on the pullback under suitable transformations of shape functions defined on **reference elements**, see [NumPDE Section 2.8], [NumPDE Def. 2.8.3.1], and § 1.4.2.24 below. ┘

1.4.2.1 Curve Partitionings

Now we introduce the counterparts of the building blocks of finite element methods for boundary element methods on closed curves $\Gamma := \partial\Omega$, $\Omega \subset \mathbb{R}^2$.

§1.4.2.2 (Curved closed polygons) We assume that Γ is a **connected curved closed Lipschitz polygon** according to Ass. 1.2.1.5. There is a (small) number $M \in \mathbb{N}$

$$\Gamma = \bar{\Gamma}_1 \cup \dots \cup \bar{\Gamma}_M, \quad \Gamma_i \cap \Gamma_j = \emptyset, \quad (1.4.2.3)$$

where the Γ_j , $j = 1, \dots, M$, are the edges of Γ with C^2 **parameterizations**

$$\begin{aligned} \gamma_j : [-1, 1] &\rightarrow \bar{\Gamma}_j, \quad j = 1, \dots, M, \\ \gamma_j(1) &= \gamma_{j+1}(-1), \quad j = 1, \dots, M-1, \quad \underbrace{\gamma_1(-1) = \gamma_M(1)}_{\Rightarrow \text{close curve}}. \end{aligned} \quad (1.4.2.4)$$

We assume that point evaluations of γ and its derivative $\dot{\gamma}$ are cheap and, inside a code, provided by simple function calls. ┘

Definition 1.4.2.5. Mesh/partitioning of a curve

A **mesh/partitioning** of a closed curved polygon according to Ass. 1.2.1.5 is a decomposition

$$\Gamma = \bigcup_{j=1}^M \bigcup_{i=1}^{N_j} \bar{\pi}_i^{(j)}, \quad \pi_i^{(j)} = \gamma_j([\xi_{i-1}^{(j)}, \xi_i^{(j)}]), \quad i = 1, \dots, N_j, \quad N_j \in \mathbb{N}, \quad j = 1, \dots, M, \quad (1.4.2.6)$$

induced by grids of the parameter intervals $[-1, 1]$:

$$-1 =: \xi_0^{(j)} < \xi_1^{(j)} < \dots < \xi_{N_j-1}^{(j)} < \xi_{N_j}^{(j)} := 1. \quad (1.4.2.7)$$

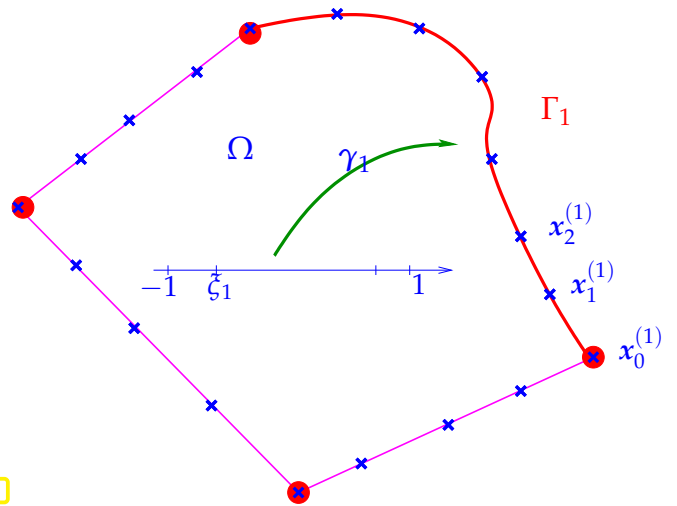
Mesh/partition of Γ induced by partitions of parameter intervals ▷

Terminology:

- **vertices:** $x_i^{(j)} = \gamma(\xi_i^{(j)})$, $i = 0, \dots, N_j$,
- **panels:** $\pi_i^{(j)} = \gamma_j([\xi_{i-1}^{(j)}, \xi_i^{(j)}])$, $i = 1, \dots, N_j$.

(In the context of FE methods we use the terms “cells” or “elements” instead of “panels” to denote the (open) sets forming the mesh partition.)

Fig. 21



Notation: We write \mathcal{G}_Γ (or simply \mathcal{G} if Γ is clear from the context) to denote a mesh/partitioning of Γ and also the set of its panels.

We define the **size** h_π of the panel $\pi \in \mathcal{G}$ as its diameter: $h_\pi := \text{diam } \pi = \|a - b\|$, where a, b are the endpoints of π . Since the parameterizations γ_j are fixed C^2 -diffeomorphisms (twice continuously differentiable, invertible, with also γ^{-1} twice continuously differentiable), for any mesh \mathcal{G}_Γ of a given closed curved polygon Γ we have bi-Lipschitz continuity

$$\exists \underline{c}, \bar{c} > 0: \quad \underline{c}|\xi - \eta| \leq \|\gamma_j(\xi) - \gamma_j(\eta)\| \leq \text{length}(\gamma([\xi, \eta])) \leq \bar{c}|\xi - \eta| \quad \forall \xi, \eta \in [-1, 1],$$

for some constants $0 < \underline{c} < \bar{c}$. So the size of a panel is “about the same” as the length of its associated parameter interval.

1.4.2.2 Piecewise Polynomial Functions on Curves

We write $\mathcal{P}_p = \mathcal{P}_p(\mathbb{R}^1)$ for the space of **univariate polynomials** of degree $\leq p$, $p \in \mathbb{N}$. This is a vector space of dimension $p + 1$.

The construction of boundary element spaces will be parametric from the beginning, relying on the local parameterization of Γ . The reader is advised to refresh his knowledge of parametric finite elements [NumPDE Section 2.8].

Definition 1.4.2.8. Pullback from a curve

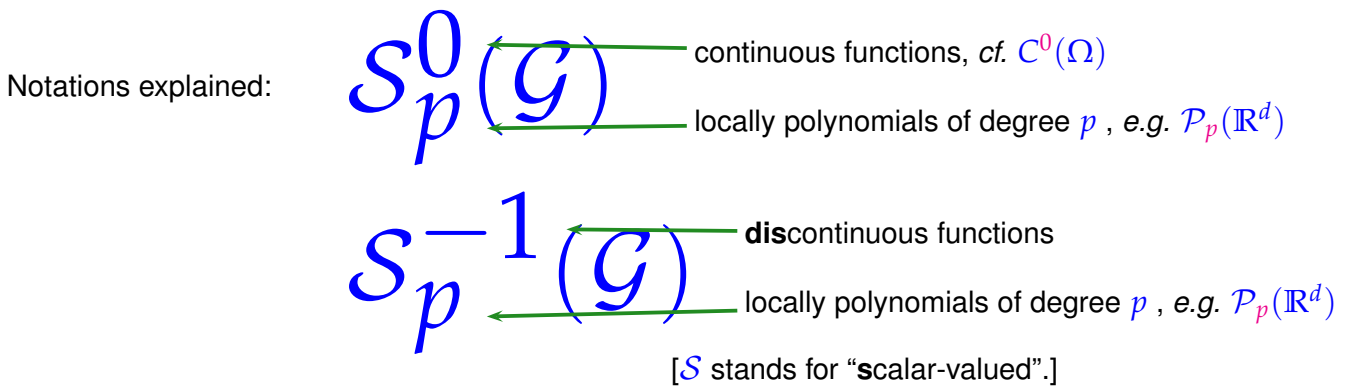
The **pullback** $\gamma_j^* f$ of a function $f : \Gamma_j \rightarrow \mathbb{R}$, Γ_j on an edge of the parameterized curved polygon Γ according to Ass. 1.2.1.5, is defined as

$$\gamma_j^* f :]-1, 1[\rightarrow \mathbb{R} \quad , \quad \gamma_j^* f(\xi) := f(\gamma_j(\xi)) \quad , \quad -1 < \xi < 1. \quad (1.4.2.9)$$

Adapting the notations for Lagrangian finite element spaces from [NumPDE Section 2.6] we write:

$$\mathcal{S}_p^0(\mathcal{G}) := \left\{ v \in C^0(\Gamma) : \gamma_j^*(v|_\pi) \in \mathcal{P}_p, \forall \pi \in \mathcal{G}, \pi \subset \Gamma_j, j = 1, \dots, M \right\}, \quad p \geq 1, \quad (1.4.2.10)$$

$$\mathcal{S}_p^{-1}(\mathcal{G}) := \left\{ v \in L^2(\Gamma) : \gamma_j^*(v|_\pi) \in \mathcal{P}_p, \forall \pi \in \mathcal{G}, \pi \subset \Gamma_j, j = 1, \dots, M \right\}, \quad p \geq 0. \quad (1.4.2.11)$$



As a consequence of Cor. 1.3.1.17 and Thm. 1.3.1.36 we conclude the following embeddings:

Corollary 1.4.2.12. Embeddings of boundary element spaces

The boundary element spaces defined in (1.4.2.10) and (1.4.2.11) satisfy

$$S_p^0(\mathcal{G}) \subset C_{pw}^1(\Gamma) \subset H^{\frac{1}{2}}(\Gamma),$$

$$S_p^{-1}(\mathcal{G}) \subset C_{pw}^0(\Gamma) \subset L^2(\Gamma) \subset H^{-\frac{1}{2}}(\Gamma),$$

where “pw” refers to the mesh \mathcal{G} .

However note that $S_p^{-1}(\mathcal{G}) \not\subset H^{\frac{1}{2}}(\Gamma)$, as we saw in Ex. 1.3.1.14.

§1.4.2.13 (Dimensions of boundary element spaces on curves) From $\dim \mathcal{P}_p = p + 1$ and the fact that the condition $S_p^0(\mathcal{G}) \subset C^0(\Gamma)$ “removes one degree of freedom per vertex of \mathcal{G} ”, we deduce the dimensions of boundary element spaces by a counting argument.

Theorem 1.4.2.14. Dimensions of BE spaces on curves

$$\dim S_p^0(\mathcal{G}) = p \cdot \#\mathcal{G}, p \geq 1 \quad \text{and} \quad \dim S_p^{-1}(\mathcal{G}) = (p + 1) \cdot \#\mathcal{G}, p \geq 0.$$

Notation: $\#\mathcal{G} \hat{=}$ no. of panels contained in \mathcal{G}

1.4.2.3 Shape Functions

Following the terminology for finite element methods from [NumPDE Section 2.5.3], the elements of an (ordered) basis $\mathfrak{B}_N := \{b_N^1, \dots, b_N^N\}$ of a boundary element space are called (global) shape functions (GSF).

Notation: We write $\mathfrak{B}_N := \{b_N^1, \dots, b_N^N\}$ for some basis of the boundary element space V_N , $N := \dim V_N$.

The shape functions for boundary element methods have to meet the same requirements as those for finite element methods:

Properties of global shape functions (GSF)

Basis functions b_N^1, \dots, b_N^N for a boundary element trial/test space V_N built on a mesh \mathcal{G} must satisfy:

- (a) $\mathfrak{B}_N := \{b_N^1, \dots, b_N^N\}$ is a basis of $V_N \quad \triangleright \quad N = \dim V_N,$

- (b) each b_N^i is **associated** with a single geometric entity (panel/edge/vertex) of \mathcal{G} ,
- (c) $\text{supp}(b_N^i) = \bigcup \{\bar{\pi} : \pi \in \mathcal{G}, p \in \bar{\pi}\}$, if b_N^i is associated with the panel/edge/vertex p .

§1.4.2.16 (Local supports of global shape functions) Condition 1.4.2.15 means that global shape functions have **small local supports**. Concretely, for a mesh of a closed curve (\rightarrow Def. 1.4.2.5), which comprises the geometric entities “vertices” and “panels”, we have that

- ◆ if b_N^i is associated with a vertex x , its support $\text{supp } b_N^i$ is the union of the panels adjacent to x ,
- ◆ if b_N^i is associated with a panel π , then $\text{supp } b_N^i = \bar{\pi}$.

EXAMPLE 1.4.2.17 (A basis for $\mathcal{S}_0^{-1}(\mathcal{G})$) $\mathcal{S}_0^{-1}(\mathcal{G})$ is the space of piecewise constant (pwc) functions on the mesh \mathcal{G} . As natural global shape functions we choose the **characteristic functions** of the panels

$$\beta_N^\pi(x) := \begin{cases} 1 & , \text{ if } x \in \pi , \\ 0 & \text{ elsewhere on } \Gamma . \end{cases}$$

which results in the basis (a “nodal basis”)

$$\mathfrak{B}_N = \{\beta_N^\pi, \pi \in \mathcal{G}\} \subset \mathcal{S}_0^{-1}(\mathcal{G}) , \tag{1.4.2.18}$$

with $\#\mathfrak{B}_N = \#\mathcal{G}$, matching Thm. 1.4.2.14.

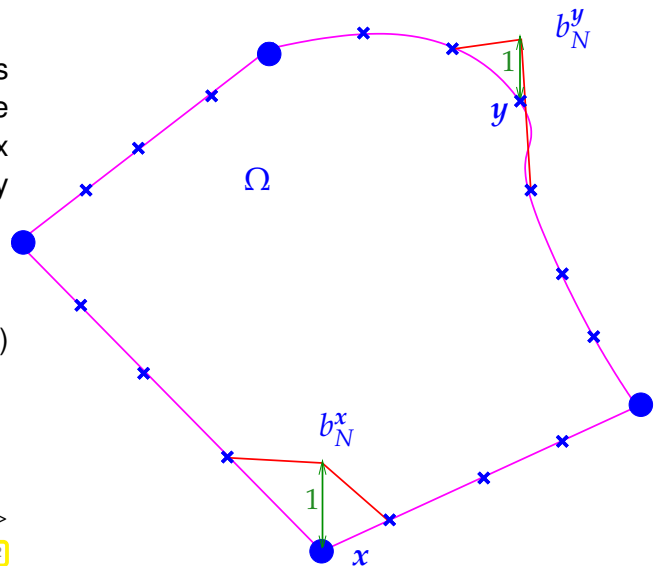
EXAMPLE 1.4.2.19 (Nodal basis for $\mathcal{S}_1^0(\mathcal{G})$)

For a mesh \mathcal{G} of a closed curve with vertices $\mathcal{V}(\mathcal{G}) := \{x_1, \dots, x_N\}$, $N \in \mathbb{N}$, we define the **tent function** (hat function) associated with a vertex $p \in \mathcal{V}(\mathcal{G})$, cf. 1.4.2.15, as in [NumPDE § 2.3.1.4] by

$$b_N^p \in \mathcal{S}_1^0(\mathcal{G}) , \tag{1.4.2.20}$$

$$b_N^p(x) = \begin{cases} 1 & , \text{ if } x = p , \\ 0 & . \text{ if } x \in \mathcal{V}(\mathcal{G}) \setminus \{p\} . \end{cases}$$

► $\text{supp } b_N^p = \bigcup \{\pi \in \mathcal{G} : p \in \bar{\pi}\} .$



Two tent functions drawn over a surface mesh

Fig. 22

§1.4.2.21 (Local shape functions (LSF)) Local shape functions for boundary element spaces are defined in exactly the same way as for finite element spaces [NumPDE Def. 2.5.3.4]. Given a panel \mathcal{G} of a mesh \mathcal{G} of Γ and a boundary element space V_N with basis $\mathfrak{B}_N = \{b_N^1, \dots, b_N^N\}$, $N := \dim V_N$, we define the set of **local shape functions** (LSF) of V_N associated with the panel π as

$$\{b_{\pi}^1, \dots, b_{\pi}^Q\} = \{b_N|_{\pi} : b_N \in \mathfrak{B}_N\} \setminus \{0\} \quad \text{for some } Q = Q(\pi) \in \mathbb{N} . \tag{1.4.2.22}$$

In words, the set of local shape functions for a panel π is the set of non-zero restrictions of global shape functions to that element. By the very definition of $\mathcal{S}_p^0(\mathcal{G})$ and $\mathcal{S}_p^{-1}(\mathcal{G})$ through pullback, see (1.4.2.10) and (1.4.2.11), we have

$$\forall \pi \in \mathcal{G}, \quad \pi \subset \Gamma_j : \quad \gamma_j^*(\text{Span}\{b_{\pi}^1, \dots, b_{\pi}^Q\}) = \mathcal{P}_p , \tag{1.4.2.23}$$

if $\{b_\pi^1, \dots, b_\pi^Q\}$ is the set of local shape functions for $S_p^0(\mathcal{G})$ or $S_p^{-1}(\mathcal{G})$ on π . The local shape functions span full polynomial spaces in this case. \lrcorner

§1.4.2.24 (Parametric construction of local shape functions) For every panel $\pi := \gamma_j([\eta_1, \eta_2]) \subset \Gamma_j$, $-1 \leq \eta_1, \eta_2 \leq 1$, of the mesh \mathcal{G} of a closed curve Γ we denote by

$$\gamma_\pi(\xi) := \gamma\left(\frac{1}{2}((1-\xi)\eta_1 + (\xi+1)\eta_2)\right), \quad \xi \in]-1, 1[, \quad (1.4.2.25)$$

a parameterization of π over the **reference interval** $\hat{I} :=]-1, 1[$: $\pi = \gamma_\pi(\hat{I})$. For instance, if the panel is a straight oriented line segment

$$\pi = [a, b], \quad a, b \in \mathbb{R}^2 \quad \blacktriangleright \quad \gamma_\pi(\xi) = \frac{1}{2}(1-\xi)a + \frac{1}{2}(\xi+1)b, \quad -1 \leq \xi \leq 1. \quad (1.4.2.26)$$

In the parametric approach the set of local shape functions $\{b_\pi^1, \dots, b_\pi^Q\}$ on π is defined through a given set of **reference shape functions** $\{\hat{b}^1, \dots, \hat{b}^Q\} \subset C^0(\hat{I})$ on \hat{I} according to [NumPDE Eq. (2.8.2.4)]

$$\hat{b}^j = \gamma_\pi^*(b_\pi^j), \quad j = 1, \dots, Q. \quad (1.4.2.27)$$

Be aware that the choice of \hat{b}^j has to make sure that the resulting local shape functions can be “glued” into global shape functions satisfying potential continuity constraints [NumPDE § 2.8.2.6]. Of course, the \hat{b}^j may depend on π , which was ignored in (1.4.2.27).

For the boundary element spaces $S_p^0(\mathcal{G})$ and $S_p^{-1}(\mathcal{G})$ the reference shape functions do not depend on the panel and are of the form

$$S_p^{-1}(\mathcal{G}): \quad \{\hat{\beta}^1, \dots, \hat{\beta}^{p+1}\} = \text{any basis of } \mathcal{P}_p, \quad p \geq 0, \quad (1.4.2.28)$$

$$S_p^0(\mathcal{G}): \quad \hat{b}^1(\xi) = \frac{1}{2}(1-\xi), \quad \hat{b}^2(\xi) = \frac{1}{2}(\xi+1), \quad (1.4.2.29)$$

$$\hat{b}^j(\xi) = (1-\xi^2)q_{j-3}(\xi), \quad j = 3, \dots, p+1, \quad \{q_0, \dots, q_{p-2}\} \text{ a basis of } \mathcal{P}_{p-2}. \quad (1.4.2.30)$$

In (1.4.2.29) the choice of \hat{b}^1 and \hat{b}^2 and the fact that $\hat{b}^j(-1) = \hat{b}^j(1) = 0$, $j = 3, \dots, p+1$, makes possible a gluing that respects the constraint $b_N^i \in C^0(\Gamma)$. \lrcorner

Supplement 1.4.2.31 (Stability of local shape functions) For larger values of the polynomial degree p stability of the reference shape functions $\hat{b}^1, \dots, \hat{b}^Q$ becomes an issue. Following the recommendation of [NumPDE Rem. 4.3.0.5] a good choice is basis functions derived from **orthogonal polynomials**:

$$\text{for } S_p^{-1}(\mathcal{G}): \quad \hat{\beta}^j = P_{j-1}, \quad j = 1, \dots, p+1, \quad (1.4.2.32)$$

$$\text{for } S_p^0(\mathcal{G}): \quad \hat{b}^1(\xi) = \frac{1}{2}(1-\xi), \quad \hat{b}^2(\xi) = \frac{1}{2}(\xi+1), \quad (1.4.2.33)$$

$$\hat{b}^j(\xi) = \int_{-1}^{\xi} P_{j-2}(\tau) d\tau, \quad j = 3, \dots, p+1.$$

Here, P_n is the n -th **Legendre polynomial** [NumPDE Def. 4.3.0.9]. The higher degree reference shape functions for $S_p^0(\mathcal{G})$ are called integrated Legendre polynomials; $\hat{b}^j(\pm 1) = 0$ for $j \geq 3$ follows from the $L^2(\hat{I})$ -orthogonality of the Legendre polynomials. \lrcorner

1.4.2.4 Solving Boundary Value Problems via Galerkin BEM

This section discusses the use of Galerkin boundary element methods to solve the boundary value problems introduced in § 1.3.5.10, the

◆ **Dirichlet BVP**: given $\mathfrak{g} \in H^{\frac{1}{2}}(\Gamma)$ find $u \in H^1(\Omega)$ such that

$$-\Delta u = 0 \quad \text{in } \Omega, \quad \mathbb{T}_D u = \mathfrak{g} \quad \text{on } \Gamma, \quad (1.3.5.11)$$

◆ and the **Neumann BVP**: given $\eta \in H_*^{-\frac{1}{2}}(\Gamma)$ determine $u \in H_*^1(\Omega)$ such that

$$-\Delta u = 0 \quad \text{in } \Omega, \quad \mathbb{T}_N u = \eta \quad \text{on } \Gamma. \quad (1.3.5.12)$$

For standard Galerkin discretization we need finite dimensional subspaces of the trace space on which the variational BIEs are posed:

function space	Eligible BE space(s)
$H^{-\frac{1}{2}}(\Gamma)$	$\mathcal{S}_p^{-1}(\mathcal{G}), p \geq 0$ and $\mathcal{S}_p^0(\mathcal{G}), p \geq 1$
$L^2(\Gamma)$	$\mathcal{S}_p^{-1}(\mathcal{G}), p \geq 0$ and $\mathcal{S}_p^0(\mathcal{G}), p \geq 1$
$H^{\frac{1}{2}}(\Gamma)$	$\mathcal{S}_p^0(\mathcal{G}), p \geq 1$, only

$\mathcal{G} \triangleq \text{mesh of } \Gamma$

§1.4.2.34 (Approximation of data) For implementation we also need a discrete representation of the data, of $\mathfrak{g} \in H^{\frac{1}{2}}(\Gamma)$ for (1.3.5.11), and of $\eta \in H^{-\frac{1}{2}}(\Gamma)$ for (1.3.5.12).

Assumption 1.4.2.35. Data in procedural form

The data functions $\mathfrak{y} \mapsto \mathfrak{g}(\mathfrak{y})$ and $\mathfrak{y} \mapsto \eta(\mathfrak{y})$ can be evaluated at any point $\mathfrak{y} \in \Gamma$.

For instance, the data functions may be supplied through a function of the signature \triangleright CppRef

```
std::function<double(double)> .
```

Then we can replace

\mathfrak{g} with $\mathfrak{g}_N \in \mathcal{S}_q^0(\mathcal{G}), q \in \mathbb{N}$, obtained by \mathcal{G} -piecewise polynomial interpolation of \mathfrak{g} (always including the vertices of the mesh into the sets of interpolation nodes),

η with $\eta_N \in \mathcal{S}_q^{-1}(\mathcal{G}), q \in \mathbb{N}_0$, obtained by \mathcal{G} -piecewise local polynomial interpolation of η on each panel. ┘

§1.4.2.36 (Galerkin BEM for 1st-kind direct BIE for Dirichlet BVP) As explained in § 1.3.5.13, the Dirichlet BVP (1.3.5.11) can be solved through the variational BIE

$$\begin{aligned} \psi \in H^{-\frac{1}{2}}(\Gamma): \quad a_V(\psi, \phi) &= \int_{\Gamma} \left(\frac{1}{2} \text{Id} + \mathbb{K} \right) \mathfrak{g}(x) \phi(x) \, dS(x) \quad \forall \phi \in H^{-\frac{1}{2}}(\Gamma), \\ a_V(\psi, \phi) &:= \int_{\Gamma} \mathbb{V}(\psi)(x) \phi(x) \, dS(x). \end{aligned} \quad (1.3.5.15)$$

Using $\mathcal{S}_p^{-1}(\mathcal{G}) \subset H^{-\frac{1}{2}}(\Gamma)$ as Galerkin trial and test space we arrive at the discrete variational problem

$$\psi_N \in \mathcal{S}_p^{-1}(\mathcal{G}): \quad a_V(\psi_N, \phi_N) = \int_{\Gamma} \left(\frac{1}{2} \text{Id} + \mathbb{K} \right) \mathfrak{g}_N(x) \phi_N(x) \, dS(x) \quad \forall \phi_N \in \mathcal{S}_0^{-1}(\Gamma), \quad (1.4.2.37)$$

where the data \mathfrak{g} have already been approximated by $\mathfrak{g}_N \in \mathcal{S}_q^0(\mathcal{G}), q \geq 1$. In order to balance accuracy, the choice $q = p + 1$ is recommended.

Choosing bases

$$\begin{aligned}\mathfrak{B}^{-1} &= \{\beta_N^1, \dots, \beta_N^N\}, & N &:= \dim \mathcal{S}_p^{-1}(\mathcal{G}), & \text{for } \mathcal{S}_p^{-1}(\mathcal{G}), & \text{and} \\ \mathfrak{B}^0 &= \{b_N^1, \dots, b_N^K\}, & K &:= \dim \mathcal{S}_q^0(\mathcal{G}), & \text{for } \mathcal{S}_q^0(\mathcal{G}),\end{aligned}$$

and writing

$$\begin{aligned}\vec{\gamma} &\in \mathbb{R}^K, & K &= \dim \mathcal{S}_q^0(\mathcal{G}), & \text{for the coefficient vector of } \mathbf{g}_N & \text{with respect to } \mathfrak{B}^0, & \text{and} \\ \vec{\psi} &\in \mathbb{R}^N, & N &:= \dim \mathcal{S}_p^{-1}(\mathcal{G}), & \text{for the coefficient vector of } \psi_N & \text{with respect to } \mathfrak{B}^{-1},\end{aligned}$$

we obtain the linear systems of equations

$$\mathbf{V}\vec{\psi} = \left(\frac{1}{2}\mathbf{M} + \mathbf{K}\right)\vec{\gamma}, \quad (1.4.2.38)$$

with the Galerkin matrices

$$\begin{aligned}\mathbf{V} &= \left(a_V(\beta_N^j, \beta_N^i)\right)_{i,j=1}^N \\ &= \left(-\frac{1}{2\pi} \int_{\Gamma} \int_{\Gamma} \log\|x - \mathbf{y}\| \beta_N^j(\mathbf{y}) \beta_N^i(x) \, dS(\mathbf{y}) \, dS(x)\right)_{i,j=1}^N \in \mathbb{R}^{N,N},\end{aligned} \quad (1.4.2.39a)$$

$$\mathbf{K} = \left(\int_{\Gamma} (\mathbf{K}b_N^j)(x) \beta_N^i(x) \, dS(x)\right)_{\substack{i=1,\dots,N \\ j=1,\dots,K}} \in \mathbb{R}^{N,K}, \quad (1.4.2.39b)$$

$$\mathbf{M} = \left(\int_{\Gamma} \beta_N^i(x) b_N^j(x) \, dS(x)\right)_{\substack{i=1,\dots,N \\ j=1,\dots,K}} \in \mathbb{R}^{N,K}. \quad (1.4.2.39c)$$

┘

§1.4.2.40 (Galerkin BEM for 1st-kind direct BIE for Neumann BVP) To solve the Neumann BVP (1.3.5.12) by Galerkin BEM we can start from the variational BIE

$$\begin{aligned}\mathbf{u} \in H_*^{\frac{1}{2}}(\Gamma): & \quad a_W(\mathbf{u}, \mathbf{v}) = \int_{\Gamma} \left(\frac{1}{2}\text{Id} - \mathbf{K}'\right) \boldsymbol{\eta}(x) \mathbf{v}(x) \, dS(x) \quad \forall \mathbf{v} \in H_*^{\frac{1}{2}}(\Gamma), \\ & \quad a_W(\mathbf{u}, \mathbf{v}) := \int_{\Gamma} \mathbf{W}(\mathbf{u})(x) \mathbf{v}(x) \, dS(x),\end{aligned} \quad (1.3.5.24)$$

posed on spaces of functions with vanishing mean. Unfortunately, there is no way to reconcile the zero mean condition and the advantages of locally supported bases for boundary element spaces. Therefore, we switch to an **augmented variational formulation** by explicitly adding the zero mean constraint: We seek the Dirichlet trace $\mathbf{u} \in H^{\frac{1}{2}}(\Gamma)$, $\alpha \in \mathbb{R}$, such that

$$\begin{aligned}a_W(\mathbf{u}, \mathbf{v}) &+ \alpha \int_{\Gamma} \mathbf{v}(x) \, dS(x) = \int_{\Gamma} \left(\frac{1}{2}\text{Id} - \mathbf{K}'\right) \boldsymbol{\eta}(x) \mathbf{v}(x) \, dS(x) \quad \forall \mathbf{v} \in H^{\frac{1}{2}}(\Gamma), \\ \int_{\Gamma} \mathbf{u}(x) \, dS(x) &= 0.\end{aligned} \quad (1.4.2.41)$$

Obviously, a vanishing mean value for the solution \mathbf{u} is enforced through the second equation. The unknown α is a so-called **Lagrangian multiplier** for the scalar zero mean constraint imposed in the second line of the augmented variational formulation.

As Galerkin trial and test we must use $\mathcal{S}_p^0(\mathcal{G}) \subset H^{\frac{1}{2}}(\Gamma)$. After replacing $\boldsymbol{\eta}$ with an approximation $\boldsymbol{\eta}_N \in \mathcal{S}_q^{-1}(\mathcal{G})$ (\rightarrow § 1.4.2.34), we thus get the discrete variational problem: Seek $\mathbf{u}_N \in \mathcal{S}_p^0(\mathcal{G})$, $\alpha \in \mathbb{R}$:

$$\begin{aligned}a_W(\mathbf{u}_N, \mathbf{v}_N) &+ \alpha \int_{\Gamma} \mathbf{v}_N(x) \, dS(x) = \int_{\Gamma} \left(\frac{1}{2}\text{Id} - \mathbf{K}'\right) \boldsymbol{\eta}_N(x) \mathbf{v}_N(x) \, dS(x) \quad \forall \mathbf{v}_N \in \mathcal{S}_p^0(\mathcal{G}), \\ \int_{\Gamma} \mathbf{u}_N(x) \, dS(x) &= 0.\end{aligned} \quad (1.4.2.42)$$

As above choosing bases

$$\mathfrak{B}^0 = \{b_N^1, \dots, b_N^N\}, N := \dim \mathcal{S}_p^0(\mathcal{G}), \text{ for } \mathcal{S}_p^0(\mathcal{G}), \text{ and}$$

$$\mathfrak{B}^{-1} = \{\beta_N^1, \dots, \beta_N^K\}, K := \dim \mathcal{S}_p^{-1}(\mathcal{G}), \text{ for } \mathcal{S}_p^{-1}(\mathcal{G}),$$

and writing

$$\vec{\eta} \in \mathbb{R}^K, K = \dim \mathcal{S}_q^{-1}(\mathcal{G}), \text{ for the coefficient vector of } \eta_N \text{ with respect to } \mathfrak{B}^{-1}, \text{ and}$$

$$\vec{\mu} \in \mathbb{R}^N, N = \dim \mathcal{S}_p^0(\mathcal{G}), \text{ for the coefficient vector of } \mathbf{u}_N \text{ with respect to } \mathfrak{B}^0,$$

we end up with the linear systems of equations

$$\begin{bmatrix} \mathbf{W} & \mathbf{c} \\ \mathbf{c}^\top & 0 \end{bmatrix} \begin{bmatrix} \vec{\mu} \\ \alpha \end{bmatrix} = \begin{bmatrix} \left(\frac{1}{2}\mathbf{M}^\top + \mathbf{K}^\top\right) \vec{\eta} \\ 0 \end{bmatrix}, \quad (1.4.2.43)$$

with the Galerkin matrices \mathbf{M} and \mathbf{K} from (1.4.2.39), and

$$\mathbf{W} = \left(a_W(b_N^j, b_N^i) \right)_{i,j=1}^N = \left(-\frac{1}{2\pi} \int_\Gamma \int_\Gamma \log \| \mathbf{x} - \mathbf{y} \| \frac{db_N^j}{ds}(\mathbf{y}) \frac{db_N^i}{ds}(\mathbf{x}) dS(\mathbf{y}) dS(\mathbf{x}) \right)_{i,j=1}^N, \quad (1.4.2.44)$$

$$\mathbf{c} = \left(\int_\Gamma b_N^j(\mathbf{x}) dS(\mathbf{x}) \right)_{j=1}^N \in \mathbb{R}^N. \quad (1.4.2.45)$$

Note that we also used that the double layer BIOs are adjoint to each other, see Suppl. 1.3.4.3,

$$\int_\Gamma (\mathbf{K}\mathbf{u})(\mathbf{x}) \phi(\mathbf{x}) dS(\mathbf{x}) = \int_\Gamma \mathbf{u}(\mathbf{x}) (\mathbf{K}'\phi)(\mathbf{x}) dS(\mathbf{y}), \quad \forall \mathbf{u} \in H^{\frac{1}{2}}(\Gamma), \phi \in H^{-\frac{1}{2}}(\Gamma). \quad (1.4.2.46)$$

Therefore, we can reuse the Galerkin matrix \mathbf{K} of the double layer boundary integral operator and simply transpose it to discretize \mathbf{K}' . \lrcorner

1.4.2.5 Approximation of Curves

In most BEM codes the curve Φ is represented by a **piecewise polynomials model**: Instead of relying on the “exact” parameterization γ_j of the edge Γ_j , one uses a piecewise polynomial approximate parameterization. Here, “piecewise” refers to the partitioning

$$[-1, 1] = [\xi_0^{(j)}, \xi_1^{(j)}] \cup [\xi_1^{(j)}, \xi_2^{(j)}] \cup \dots \cup [\xi_{N_j-1}^{(j)}, \xi_{N_j}^{(j)}], \quad (1.4.2.47)$$

of the parameter interval $[-1, 1]$ induced by the grid

$$-1 =: \xi_0^{(j)} < \xi_1^{(j)} < \dots < \xi_{N_j-1}^{(j)} = \xi_{N_j}^{(j)} := 1. \quad (1.4.2.7)$$

On each parameter grid interval one considers the vector-valued polynomial

$$\tilde{\gamma}_i^{(j)} : [\xi_{i-1}^{(j)}, \xi_i^{(j)}] \rightarrow \mathbb{R}^2, \quad \tilde{\gamma}_i^{(j)} \in (\mathcal{P}_p)^2 \quad p \in \mathbb{N}, \quad (1.4.2.48)$$

interpolating γ at the endpoints

$$\tilde{\gamma}_i^{(j)}(\xi_k^{(j)}) = \gamma(\xi_k^{(j)}) \quad \text{for } k = i-1, i. \quad (1.4.2.49)$$

§1.4.2.50 (Approximation by a polygon)

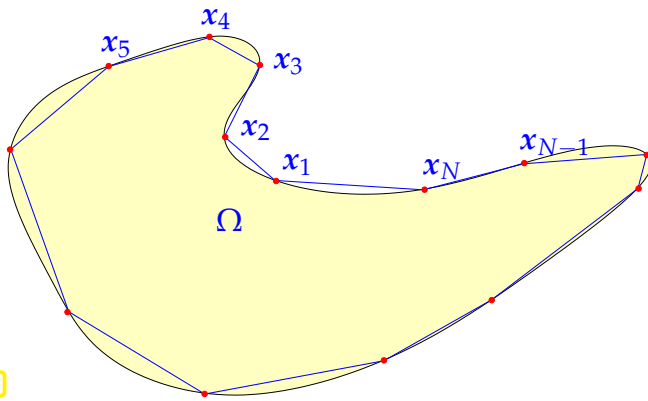


Fig. 23

The simplest case $p = 1$ amounts to an approximation of Γ by a polygon.

◁ polygon interpolating a closed curve described by a single parameterization γ_1

Setting $x_i^{(j)} := \gamma(\xi_i^{(j)})$, we get the affine formula

$$\tilde{\gamma}_i^{(j)}(\xi) = x_{i-1}^{(j)} \frac{\xi_i^{(j)} - \xi}{\xi_i^{(j)} - \xi_{i-1}^{(j)}} + x_i^{(j)} \frac{\xi - \xi_{i-1}^{(j)}}{\xi_i^{(j)} - \xi_{i-1}^{(j)}}, \tag{1.4.2.51}$$

$$\xi_{i-1} \leq \xi \leq \xi_i, \quad i = 1, \dots, N_j.$$

The derivative with respect to the parameter is

$$\frac{d}{d\xi} \tilde{\gamma}_i^{(j)}(\xi) = x_i^{(j)} - x_{i-1}^{(j)}. \tag{1.4.2.52}$$

┘

§1.4.2.53 (Curve approximation by interpolation) The approximate polynomial parameterizations $\tilde{\gamma}_i^{(j)} \in (\mathcal{P}_p)^2$ can be constructed by means of **polynomial interpolation** of γ on $[\xi_{i-1}^{(j)}, \xi_i^{(j)}]$: fixing $p + 1$ **interpolation nodes**

$$\xi_{i-1}^{(j)} \leq v_0 < v_1 < \dots < v_p \leq \xi_i^{(j)}$$

by [NumCSE Thm. 5.2.2.7] we can find a unique interpolating polynomial $\tilde{\gamma}_i^{(j)} \in (\mathcal{P}_p)^2$ satisfying the interpolation conditions

$$\tilde{\gamma}_i^{(j)}(v_k) = \gamma(v_k), \quad k = 0, \dots, p. \tag{1.4.2.54}$$

Stability of the interpolation procedure is a major concern, cf. [NumCSE Section 5.2.4], and the use of **Chebyshev interpolation** is recommended, see [NumCSE Section 6.2.3], in particular [NumCSE Rem. 6.2.3.19]. It is based on the nodes [NumCSE Eq. (6.2.3.12)]

$$v_k = \xi_{i-1}^{(j)} + \frac{1}{2}(\xi_i^{(j)} - \xi_{i-1}^{(j)}) \left(\cos\left(\frac{2k+1}{2(p+1)}\pi\right) + 1 \right), \quad k = 0, \dots, p. \tag{1.4.2.55}$$

We remark that interpolation need not be carried out on the grid intervals (1.4.2.47) of the parameter domain. Instead global polynomial interpolation of γ_j on $[-1, 1]$ is another option. ┘

§1.4.2.56 (Data structure for closed polygon) In the C++ code presented in § 1.4.0.1 a closed polygon with N vertices is represented by

- a $N \times 2$ -matrix whose rows store the coordinates of the corners.
- another $N \times 2$ -matrix containing the indices of the endpoints of the panels.

C++11 code 1.4.2.57: Class for closed polygon (incomplete listing) →GITLAB

```

1 class BoundaryMesh
2 {
3     private:
4         /// The two coordinates for vertices are stored in the rows of a
5         matrix
6         typedef Eigen::Matrix<double, Eigen::Dynamic, 2> coord_matrix_t;

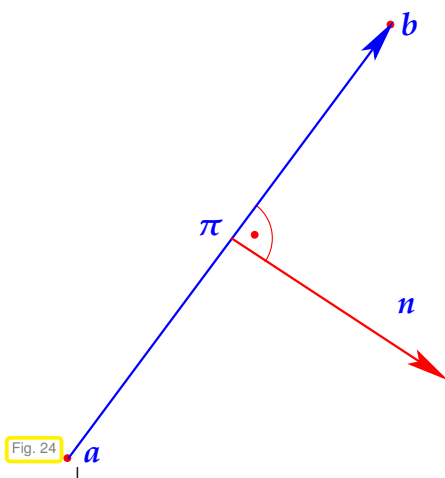
```

```

6 | // The indices of endpoints of flat panels are stored in the rows of
   | a matrix
7 | typedef Eigen::Matrix<int, Eigen::Dynamic, 2> elem_matrix_t;
8 |
9 | // data container for geometric and topological information
10 | coord_matrix_t coordinates_;
11 | elem_matrix_t elements_;
12 | bool isInitialized_;
13 |
14 | public:
15 | // Constructor from raw data
16 | BoundaryMesh(const coord_matrix_t& coords,
17 |             const elem_matrix_t& elems);
18 | // Constructor reading the data from file
19 | BoundaryMesh(const std::string& filename);
20 | // Straightforward access methods
21 | int numVertices() const; // No. of vertices
22 | int numElements() const; // No. of panels
23 | const coord_matrix_t &getMeshVertices() const;
24 | const elem_matrix_t &getMeshElements() const;
25 | // Coordinates of i-th vertex
26 | Eigen::Vector2d getVertex(int i) const;
27 | // Coordinates of vertices of i-th element
28 | std::pair<Eigen::Vector2d, Eigen::Vector2d> getElementVertices(int i) const;
29 | // Coordinates of j-th vertex, j=0,1 of i-th element
30 | int getElementVertex(int i, int j) const;
31 | };

```

The code adopts the following *convention* about the orientation of the normal vector



$$\pi = [a, b], \quad a = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

$$\Rightarrow n(x) = (b - a)^\perp = \begin{bmatrix} b_2 - a_2 \\ -(b_1 - a_1) \end{bmatrix}, \quad x \in \pi.$$

1.4.3 Computation of BEM-Galerkin Matrix in 2D

1.4.3.1 Panel-oriented Assembly

As setting we consider a boundary element discretization of a linear variational problem (\rightarrow Def. 1.1.5.1)

$$u \in V_0: \quad a(u, v) = \ell(v) \quad \forall v \in V_0, \quad (1.1.5.2)$$

that arises from a variational formulation of a first-kind or second kind variational boundary integral equation like

$$\psi \in H^{-\frac{1}{2}}(\Gamma): \quad a_V(\psi, \phi) = \int_\Gamma (\frac{1}{2}\text{Id} + K)g(x) \phi(x) dS(x) \quad \forall \phi \in H^{-\frac{1}{2}}(\Gamma), \quad (1.3.5.15)$$

$$a_V(\psi, \phi) := \int_\Gamma V(\psi)(x) \phi(x) dS(x), \quad (1.3.5.16)$$

for which $V_0 = H^{-\frac{1}{2}}(\Gamma)$, or

$$\begin{aligned} \mathbf{u} \in H_*^{\frac{1}{2}}(\Gamma): \quad a_W(\mathbf{u}, \mathbf{v}) &= \int_{\Gamma} \left(\frac{1}{2} \text{Id} - K' \right) \boldsymbol{\eta}(\mathbf{x}) \mathbf{v}(\mathbf{x}) \, dS(\mathbf{x}) \quad \forall \mathbf{v} \in H_*^{\frac{1}{2}}(\Gamma), \\ a_W(\mathbf{u}, \mathbf{v}) &:= \int_{\Gamma} W(\mathbf{u})(\mathbf{x}) \mathbf{v}(\mathbf{x}) \, dS(\mathbf{x}), \end{aligned} \quad (1.3.5.24)$$

where $V_0 = H^{\frac{1}{2}}(\Gamma)$. We remark that we could also start from the 2nd-kind variational BIE (1.3.5.31) and (1.3.5.34), for which $V_0 = L^2(\Gamma)$.

We equip the curve Γ with a mesh \mathcal{G} as in Section 1.4.2.1. For Galerkin discretization (\rightarrow Section 1.4.1) we employ a boundary element space $V_N \subset V_0$, $\dim V_N = N$, concretely

- $V_N = \mathcal{S}_p^{-1}(\mathcal{G})$ for $V_0 = H^{-\frac{1}{2}}(\Gamma)$ and $V_0 = L^2(\Gamma)$,
- and $V_N = \mathcal{S}_p^0(\mathcal{G})$ for $V_0 = H^{\frac{1}{2}}(\Gamma)$ (\rightarrow Section 1.4.2.2).

We endow V_N with a basis $\mathfrak{B}_N = \{b_N^1, \dots, b_N^N\}$ as in Section 1.4.2.3. As elaborated in Section 1.4.2.3 the basis functions, also called global shape functions (GSF), are **locally supported** and parametric piecewise polynomials composed of contributions of local shape functions (LSF), see § 1.4.2.21. The standard choice of global shape functions for $\mathcal{S}_0^{-1}(\mathcal{G})$ and $\mathcal{S}_1^0(\mathcal{G})$ is presented in Ex. 1.4.2.17 and Ex. 1.4.2.19.

We end up with a linear system of equations

$$\mathbf{A} \vec{\mu} = \vec{\varphi}, \quad \mathbf{A} = \left(a(b_N^j, b_N^i) \right)_{i,j=1}^N \in \mathbb{R}^{N,N}, \quad \vec{\varphi} := \left(\ell(b_N^i) \right)_{i=1}^N \in \mathbb{R}^N. \quad (1.4.3.1)$$

As in the field of finite element methods [NumPDE Section 2.7.4], also for boundary element methods **assembly** means the initialization of the Galerkin matrix $\mathbf{A} \in \mathbb{R}^{N,N}$, and right hand side vector $\vec{\varphi} \in \mathbb{R}^n$. We start by writing $a(\mathbf{u}, \mathbf{v})$ as a sum of contributions of pairs of panels, e.g., in the case of the bilinear form induced by the single layer boundary integral operator

$$a_V(\psi, \phi) = \sum_{i=1}^{\#\mathcal{G}} \sum_{j=1}^{\#\mathcal{G}} \int_{\pi_i} \int_{\pi_j} G^\Delta(\mathbf{x}, \mathbf{y}) \psi(\mathbf{y}) \phi(\mathbf{x}) \, dS(\mathbf{y}) \, dS(\mathbf{x}). \quad (1.4.3.2)$$

This is also possible for all other BIE-related bilinear forms occurring in the variational problems of Section 1.3.5:

$$\int_{\Gamma} (K\mathbf{v})(\mathbf{x}) \phi(\mathbf{x}) \, dS(\mathbf{x}) = \sum_{i=1}^{\#\mathcal{G}} \sum_{j=1}^{\#\mathcal{G}} \int_{\pi_i} \int_{\pi_j} \mathbf{grad}_y G^\Delta(\mathbf{x}, \mathbf{y}) \cdot \mathbf{n}(\mathbf{y}) \mathbf{v}(\mathbf{y}) \phi(\mathbf{x}) \, dS(\mathbf{y}) \, dS(\mathbf{x}), \quad (1.4.3.3)$$

$$\int_{\Gamma} (K'\phi)(\mathbf{x}) \mathbf{v}(\mathbf{x}) \, dS(\mathbf{x}) = \sum_{i=1}^{\#\mathcal{G}} \sum_{j=1}^{\#\mathcal{G}} \int_{\pi_i} \int_{\pi_j} \mathbf{grad}_x G^\Delta(\mathbf{x}, \mathbf{y}) \cdot \mathbf{n}(\mathbf{x}) \phi(\mathbf{y}) \mathbf{v}(\mathbf{x}) \, dS(\mathbf{y}) \, dS(\mathbf{x}), \quad (1.4.3.4)$$

$$\int_{\Gamma} (W\mathbf{u})(\mathbf{x}) \mathbf{v}(\mathbf{x}) \, dS(\mathbf{x}) = \sum_{i=1}^{\#\mathcal{G}} \sum_{j=1}^{\#\mathcal{G}} \int_{\pi_i} \int_{\pi_j} G^\Delta(\mathbf{x}, \mathbf{y}) \frac{d\mathbf{u}}{ds}(\mathbf{y}) \frac{d\mathbf{v}}{ds}(\mathbf{x}) \, dS(\mathbf{y}) \, dS(\mathbf{x}), \quad (1.4.3.5)$$

where $\frac{d}{ds}$ denotes the arclength derivative (1.3.4.19). More explicit formulas for the integrands are given in § 1.3.4.8 and § 1.3.4.10.

§1.4.3.6 (Non-locality of variational BIEs) The bilinear forms \mathbf{b} occurring in variational formulations of partial differential equations (PDEs), for instance in (1.1.5.5), are **local** in the sense that

$$\text{[for PDEs]:} \quad \text{vol}_d(\text{supp}(u) \cap \text{supp}(v)) = 0 \implies \mathbf{b}(u, v) = 0. \quad (1.4.3.7)$$

This usually makes it possible to evaluate $b(u_N, v_N)$ for finite element trial and test functions u_N, v_N by summing **once** over the elements of the finite element mesh, see [NumPDE Section 2.7.4.1]. The property (1.4.3.8) also makes locally supported basis functions spawn **sparse Galerkin matrices** in the **finite element method** see [NumPDE Section 2.4.4].

The situation is fundamentally different in the case of the bilinear forms spawned by boundary integral operators. The presence of globally supported kernels thwarts any locality of the kind (1.4.3.8):

$$\text{[for BIEs]: } \text{vol}_{d-1}(\text{supp}(u) \cap \text{supp}(v)) = 0 \not\Rightarrow b(u, v) = 0 . \tag{1.4.3.8}$$

This has profound consequences for **boundary element methods**, particular for data structures and algorithms:

- boundary element Galerkin matrices will be **densely populated**,
- the bilinear forms arising from BIE require a **double summation** of local contributions as in (1.4.3.2).

┘

§1.4.3.9 (Local → global index mapping) The formulas (1.4.3.2)–(1.4.3.5) are the starting point for developing algorithms for the assembly of the Galerkin matrix \mathbf{A} and the right-hand side vector $\vec{\phi}$ from (1.4.3.1). A key issue will be the algorithmic representation of the relationship between global shape functions (GSF) and local shape functions (LSF, → § 1.4.2.21). To see why, note that, with $\beta_N^i, i = 1, \dots, N$, denoting the global and $\beta_\pi^k, k = 1, \dots, Q$, the local shape functions of a boundary element space $\subset H^{-\frac{1}{2}}(\Gamma)$, for every pair $(j, i) \in \{1, \dots, N\}^2$

$$a_V(\beta_N^j, \beta_N^i) = \sum_{\pi \in \mathcal{G}} \sum_{\pi' \in \mathcal{G}} \int_\pi \int_{\pi'} G^\Delta(x, y) \beta_{\pi'}^k(y) \beta_\pi^\ell(x) dS(y) dS(x) ,$$

for **uniquely defined** $\ell, k \in \{1, \dots, Q\}$. Now we formalize these considerations.

- ❶ The global shape functions of the basis $\mathfrak{B}_N = \{b_N^1, \dots, b_N^N\}$ of a boundary element space V_N are supposed to be **ordered** and, thus, can be identified through a unique index $\in \{1, \dots, N\}$ (as already insinuated by the notation b_N^i).
- ❷ We also assume an ordering of the local shape functions b_π^i for every panel $\pi \in \mathcal{G}$, also indicated by indices $\in \{1, \dots, Q\}$.
- ❸ Observe that for each $\pi \in \mathcal{G}$ and its local shape function $b_\pi^i, i = 1, \dots, Q$, there is a **unique** global shape function $b_N^j, j \in \{1, \dots, N\}$ such that $b_N^j|_\pi = b_\pi^i$

► We can define a **local→global index map** (“d.o.f. mapper”) as [NumPDE Eq. (2.7.4.8)]

$$\begin{aligned} \text{locglobmap} : \mathcal{G} \times \mathbb{N} &\rightarrow \mathbb{N} , \\ \text{locglobmap}(\pi, i) &= j , \text{ if } \underset{\substack{\text{global shape function} \\ \uparrow}}{b_N^j|_\pi} = \underset{\substack{\text{local shape function} \\ \leftarrow}}{b_\pi^i} \text{ } \leftarrow i \in \{1, \dots, Q(\pi)\} . \end{aligned} \tag{1.4.3.10}$$

┘

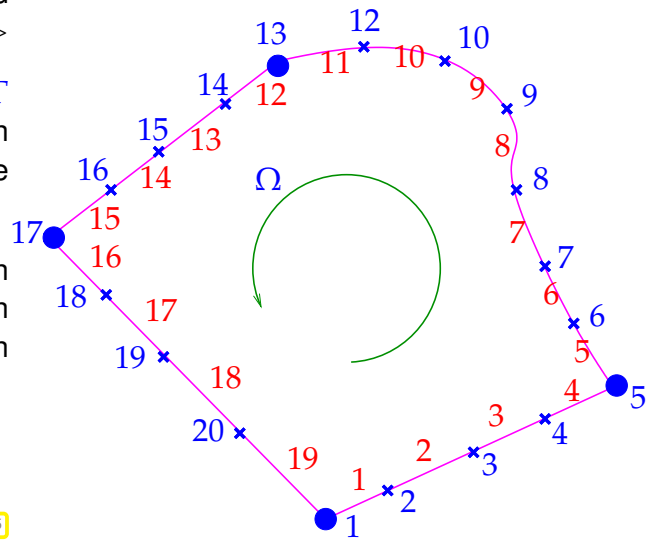
EXAMPLE 1.4.3.11 (local→global index map)

We assume that the curve $\Gamma := \partial\Omega$ is connected and oriented counterclockwise \triangleright

Vertices (blue) and panels (red) of the mesh \mathcal{G} of Γ are numbered consecutively as in Fig. 25: the i -th panel, $i = 1, \dots, N-1$, has vertices i and $i+1$, the N -th panel vertices N and 1 .

We consider $V_N = \mathcal{S}_1^0(\mathcal{G})$, tent function basis as in Ex. 1.4.2.19, and assume that the i -th basis function is associated with the i -th vertex, $i = 1, \dots, N$. Then the local \rightarrow global index map from (1.4.3.10) reads:

$$\text{locglobmap}(k, i) = \begin{cases} k & , \text{ if } i = 1, \\ k + 1 & , \text{ if } i = 2. \end{cases} \quad \text{Fig. 25} \quad (1.4.3.12)$$



Using the local \rightarrow global index mapping, we can now write in a rigorous way

$$a_{\nu}(\beta_N^j, \beta_N^i) = \sum_{\substack{\pi \in \mathcal{G} \\ \text{locglobmap}(\pi, k) = j}} \sum_{\substack{\pi' \in \mathcal{G} \\ \text{locglobmap}(\pi', \ell) = i}} \int_{\pi} \int_{\pi'} G^{\Delta}(x, y) \beta_{\pi'}^k(y) \beta_{\pi}^{\ell}(x) dS(y) dS(x) . \quad (1.4.3.13)$$

An efficient implementation of this formula takes into account the constraints $\text{locglobmap}(\pi, k) = j$ and $\text{locglobmap}(\pi', \ell) = i$ by inverting them, thus *distributing* the numbers obtained from evaluating the double integrals to suitable entries of the Galerkin matrix. This leads to **panel-oriented assembly**.

The following pseudocode demonstrates the implementation of panel-oriented assembly by means of two nested loops over all panels.

Pseudocode 1.4.3.14: Outline of panel-oriented assembly of BE matrices for BIOs (same trial and test space)

```

Matrix A(N,N); A = 0;      {Initialize dense matrix with zero}
forall panels  $\pi \in \mathcal{G}$  do {outer loop}
  Q := no_of_loc_shape_fns( $\pi$ );
  forall panels  $\pi' \in \mathcal{G}$  do {inner loop}
    Q' := no_of_loc_shape_fns( $\pi'$ );
    Matrix A1 := get_interaction_matrix( $\pi, \pi'$ ); {get "local" matrix}
    for k=1 to Q do
      i = locglobmap( $\pi, k$ );
      for l=1 to Q' do
        j = locglobmap( $\pi', l$ );
        A(i, j) += A1(k, l); {update of Galerkin matrix}
      endfor
    endfor
  endfor
endfor
endfor
endfor
    
```

As auxiliary functions we need

- (I) the local \rightarrow global index mapping function `locglobmap` as introduced in § 1.4.3.9,

- (II) a function `get_interaction_matrix` computing the contribution of a pair of panels π, π' to the Galerkin matrix. If Q, Q' are the number of local shape functions (\rightarrow § 1.4.2.21) on π and π' , respectively, then this function returns a $Q \times Q'$ -matrix \mathbf{A}_{loc} :

$$(\mathbf{A}_{\text{loc}})_{kl} = \int_{\pi} \int_{\pi'} k(\mathbf{x}, \mathbf{y}) b_{\pi}^k(\mathbf{y}) b_{\pi'}^l(\mathbf{x}) dS(\mathbf{y}) dS(\mathbf{x}), \quad (1.4.3.15)$$

which we wrote for a general boundary integral operator with kernel k . For instance, in the case of $\mathbf{a} = \mathbf{a}_V$, we face the singular kernel $k(\mathbf{x}, \mathbf{y}) = G^{\Delta}(\mathbf{x}, \mathbf{y})$.

EXAMPLE 1.4.3.16 (Assembly of Galerkin matrix for double layer BIO K) We consider the Galerkin discretization of the bilinear form induced by the double layer boundary integral operator \mathbf{K}

$$(\mathbf{v}, \phi) \mapsto \int_{\Gamma} (\mathbf{K}(\mathbf{v}))(\mathbf{x}) \phi(\mathbf{x}) dS(\mathbf{x}), \quad \mathbf{v} \in H^{\frac{1}{2}}(\Gamma), \phi \in H^{-\frac{1}{2}}(\Gamma).$$

We rely on lowest order/degree piecewise polynomial boundary element spaces

$$H^{\frac{1}{2}}(\Gamma) \rightarrow \mathcal{S}_1^0(\mathcal{G}) \subset H^{\frac{1}{2}}(\Gamma), \quad H^{-\frac{1}{2}}(\Gamma) \rightarrow \mathcal{S}_0^{-1}(\mathcal{G}) \subset H^{-\frac{1}{2}}(\Gamma),$$

where \mathcal{G} is a mesh of the closed polygon Γ , see Def. 1.4.2.5.

As bases we use

for $\mathcal{S}_1^0(\mathcal{G})$: tent function basis $\mathfrak{B}^0 := \{b_N^1, \dots, b_N^N\}$ see (1.4.2.19),

for $\mathcal{S}_0^{-1}(\mathcal{G})$: characteristic function basis $\mathfrak{B}^{-1} := \{\beta_N^1, \dots, \beta_N^N\}$ see (1.4.2.17).

- ▶ $\text{supp } b_N^i$ is the union of two adjacent panels, $\text{supp } \beta_N^j$ covers only a single panel.
- ▶ For $\mathcal{S}_1^0(\mathcal{G})$, $Q = 2$, for $\mathcal{S}_0^{-1}(\mathcal{G})$, $Q = 1$, where Q designates the number of local shape functions per panel, see § 1.4.2.21

$$\Rightarrow \mathbf{A}_{\text{loc}} \in \mathbb{R}^{1,2} \quad (\text{see (1.4.3.15) for the definition of } \mathbf{A}_{\text{loc}}).$$

Since a fully populated matrix has to be initialized we face the following computational cost of assembly:

Cost of assembling a BIO Galerkin matrix

The asymptotic computational effort for assembling the Galerkin matrix discretizing a BIO based on trial and test spaces with dimensions N and M , respectively, is at least $O(MN)$ for $M, n \rightarrow \infty$.

The following C++ function performs the assembly of the Galerkin matrix for \mathbf{K} . Refer to Code 1.4.2.57 for explanations on the class **BoundaryMesh**.

C++11 code 1.4.3.18: Assembly of Galerkin matrix for double layer BIO K [→GITLAB](#)

```

2 void computeK(Eigen::MatrixXd &K, const BoundaryMesh &mesh, double eta) {
3     int nE = mesh.numElements();
4     int nC = mesh.numVertices();
5     // Matrix returned through reference: resize and initialize matrix
6     K.resize(nE, nC);
7     K.setZero();
8     double l0 = 0.0, l1 = 0.0;
9
10    // outer loop: traverse the panels
11    for (int j = 0; j < nE; ++j) {

```

```

12 // get vertices indices and coordinates for panel  $\pi_j = [a, b]$ 
13 int aidx = mesh.getElementVertex(j, 0);
14 int bidx = mesh.getElementVertex(j, 1);
15 const Eigen::Vector2d &a = mesh.getVertex(aidx);
16 const Eigen::Vector2d &b = mesh.getVertex(bidx);
17
18 // inner loop: traverse the panels
19 for (int i = 0; i < nE; ++i) {
20 // get vertices indices and coordinates for panel  $\pi_i = [c, d]$ 
21 int cidx = mesh.getElementVertex(i, 0);
22 int didx = mesh.getElementVertex(i, 1);
23 const Eigen::Vector2d &c = mesh.getVertex(cidx);
24 const Eigen::Vector2d &d = mesh.getVertex(didx);
25 // Zero contribution for parallel panels !
26 double lindep1 = fabs((a - c)[0] * (b - a)[1] - (a - c)[1] * (b - a)[0]);
27 double lindep2 = fabs((a - d)[0] * (b - a)[1] - (a - d)[1] * (b - a)[0]);
28
29 if (lindep1 > EPS * (a - c).norm() ||
30     lindep2 > EPS * (a - d).norm()) //
31 {
32 // compute entries of  $1 \times 2$  interaction matrix
33 // double I0=0.0, I1=0.0;
34 computeKij(&I0, &I1, eta, a, b, c, d);
35 // distribute values to matrix entries
36 K(j, cidx) += I0 - I1; //
37 K(j, didx) += I0 + I1; //
38 } // endif
39 } // endfor
40 } // endfor
41 }

```

Remarks on Code 1.4.3.18

- The function `computeKij` adopts an unusual convention for the reference shape functions (1.4.2.27) for the $\mathcal{S}_1^0(\mathcal{G})$:

$$\widehat{b}^1(\widehat{\xi}) = \frac{1}{2}, \quad \widehat{b}^2(\widehat{\xi}) = \frac{1}{2}\xi, \quad \xi \in \widehat{I} :=]-1, 1[.$$

This accounts for the linear combinations used in Line 36 and Line 37.

- Note that, if $\pi \parallel \pi'$ (parallel panels), then

$$\int_{\pi} (K(\mathbf{v}|_{\pi'}))(x) \phi(x) dS(x) = \int_{\pi} \int_{\pi'} \frac{(\mathbf{y} - x) \cdot \mathbf{n}(\mathbf{y})}{\|\mathbf{y} - x\|^2} \mathbf{v}(\mathbf{y}) \phi(x) dS(\mathbf{y}) dS(x) = 0 \quad \forall \mathbf{v}, \phi,$$

because of the orthogonality $(\mathbf{y} - x) \cdot \mathbf{n}(\mathbf{y}) = 0$. This is tested in a numerically sound way in Line 30.

1.4.3.2 Lowest-order BEM on Polygons: Analytic Formulas

We consider the case that Γ is or is approximated by a closed connected polygon (with straight edges!), see § 1.4.2.50. In this case all panels of a mesh \mathcal{G} are line segments.

$$\mathcal{G} = \{\pi_1, \dots, \pi_N\}, \quad \Gamma = \overline{\pi}_1 \cup \dots \cup \overline{\pi}_N, \quad \pi_i = [p_i, q_i], \quad p_i, q_i \in \mathbb{R}^2.$$

A data structure modeling such meshes is presented in Code 1.4.2.57.

We restrict ourselves to Galerkin discretization based on lowest degree boundary element spaces $\mathcal{S}_1^0(\mathcal{G})$ and $\mathcal{S}_0^{-1}(\mathcal{G})$, which are implemented in the 2D BEM C++ code introduced in § 1.4.0.1. We use the standard bases of “tent functions” and characteristic functions, respectively, for these spaces, see Ex. 1.4.2.17 and Ex. 1.4.2.19.

§1.4.3.19 (Panel interaction matrix for the single layer BIO) The bilinear form

$$a_V(\psi, \phi) = -\frac{1}{2\pi} \int_{\Gamma} \int_{\Gamma} \log\|x - y\| \psi(y) \phi(x) dS(y) dS(x), \quad \psi, \phi \in H^{-\frac{1}{2}}(\Gamma),$$

is discretized on $\mathcal{S}_0^{-1}(\mathcal{G}) \times \mathcal{S}_0^{-1}(\mathcal{G})$, the local shape functions have constant value = 1 on each panel. Therefore, we just have to compute **1 × 1 interaction matrices** for pairs of panels:

$$I := a_V(\beta_N^\pi, \beta_N^{\pi'}) = -\frac{1}{2\pi} \int_{\pi} \int_{\pi'} \log\|x - y\| \beta_N^\pi(y) \beta_N^{\pi'}(x) dS(y) dS(x), \quad \pi, \pi' \in \mathcal{G}. \quad (1.4.3.20)$$

❶ If $\pi = [p_1, q_1]$, $\pi' = [p_2, q_2]$, then we can **transform** the line integrals **to the reference interval** $\hat{I} := [-1, 1]$ through the parameterizations

$$\begin{aligned} \text{[for } \pi]: \quad \gamma(t) &:= p_1 + (t+1)\frac{1}{2}(q_1 - p_1) = \frac{1}{2}(p_1 + q_1) + \frac{1}{2}t(q_1 - p_1), \\ \text{[for } \pi']: \quad \gamma'(t) &:= p_2 + (t+1)\frac{1}{2}(q_2 - p_2) = \frac{1}{2}(p_2 + q_2) + \frac{1}{2}t(q_2 - p_2), \end{aligned} \quad -1 \leq t \leq 1,$$

which, by the defining formula (1.2.1.9) for curve integrals

$$\int_{\pi} f(x) dS(x) = \int_{-1}^1 f(\gamma(t)) \|\dot{\gamma}(t)\| dt, \quad \dot{\gamma} := \frac{d\gamma}{dt},$$

results in

$$I = -\frac{1}{2\pi} \int_{-1}^1 \int_{-1}^1 \log\|su - tv + z\| \frac{1}{2} \|q_1 - p_1\| \frac{1}{2} \|q_2 - p_2\| dt ds. \quad (1.4.3.21)$$

$$\text{with } u = \frac{1}{2}(q_1 - p_1), \quad v = \frac{1}{2}(q_2 - p_2), \quad z := \frac{1}{2}(p_1 + q_1 - p_2 - q_2). \quad (1.4.3.22)$$

The following manipulations mainly rely on the identity

$$(\hat{x} - c) \cdot \mathbf{grad}_{\hat{x}}(\log\|\mathbf{M}(\hat{x} - c)\|) = (\hat{x} - c) \cdot \mathbf{M}^\top \frac{\mathbf{M}(\hat{x} - c)}{\|\mathbf{M}(\hat{x} - c)\|^2} = 1. \quad (1.4.3.23)$$

The reader is encouraged to derive this formula by applying the chain rule twice.

❷ By **Green's first formula** Thm. 1.1.6.1 we conclude for any domain $D \subset \mathbb{R}^2$

$$\begin{aligned} \int_D \log\|\mathbf{M}(\hat{x} - c)\| d\hat{x} &= \int_D \frac{1}{2} \operatorname{div}_{\hat{x}} \{\hat{x} \mapsto (\hat{x} - c)\} \log\|\mathbf{M}(\hat{x} - c)\| d\hat{x} \\ &= -\frac{1}{2} \underbrace{\int_D (\hat{x} - c) \cdot \mathbf{grad}_{\hat{x}} \log\|\mathbf{M}(\hat{x} - c)\| d\hat{x}}_{=\operatorname{vol}_2(D) \text{ by (1.4.3.23)}} \\ &\quad + \frac{1}{2} \int_{\partial D} (\hat{x} - c) \cdot \mathbf{n}(\hat{x}) \log\|\mathbf{M}(\hat{x} - c)\| dS(\hat{x}). \end{aligned}$$

The boundary integral $\int_{\partial D}$ is a one-dimensional line integral. Moreover, if D is a **polygon** the exterior unit normal $\mathbf{n}(\hat{x})$ is piecewise constant and $\hat{x} \mapsto (\hat{x} - c) \cdot \mathbf{n}(\hat{x})$ will be **constant** on all edges, cf. Hesse normal form of a line in \mathbb{R}^2 .

③ In concrete terms we apply this trick to the integral (1.4.3.21) with $\hat{\mathbf{x}} = \begin{bmatrix} s \\ t \end{bmatrix}$, $\mathbf{M} = [\mathbf{u}, -\mathbf{v}]$, $\mathbf{c} = \begin{bmatrix} \alpha \\ -\beta \end{bmatrix}$. Then, for any $\alpha, \beta \in \mathbb{R}$, $\mathbf{u}, \mathbf{v} \in \mathbb{R}^2$ (1.4.3.23) implies

$$\left\{ (s + \alpha) \frac{\partial}{\partial s} + (t - \beta) \frac{\partial}{\partial t} \right\} \log \|\mathbf{u}(s + \alpha) - \mathbf{v}(t - \beta)\| = 1. \quad (1.4.3.24)$$

$$\begin{aligned} \blacktriangleright \quad 4 &= \int_{-1}^1 \int_{-1}^1 (s + \alpha) \frac{\partial}{\partial s} \log \|\mathbf{u}(s + \alpha) - \mathbf{v}(t - \beta)\| dt ds + \\ &\quad \int_{-1}^1 \int_{-1}^1 (t - \beta) \frac{\partial}{\partial t} \log \|\mathbf{u}(s + \alpha) - \mathbf{v}(t - \beta)\| dt ds \\ &= \int_{-1}^1 \left\{ [(s + \alpha) \log \|\dots\|]_{s=-1}^{s=1} - \int_{-1}^1 \log \|\dots\| ds \right\} dt + \\ &\quad \int_{-1}^1 \left\{ [(t - \beta) \log \|\dots\|]_{t=-1}^{t=1} - \int_{-1}^1 \log \|\dots\| dt \right\} ds, \end{aligned}$$

where $\|\dots\| = \|\mathbf{u}(s + \alpha) - \mathbf{v}(t - \beta)\|$ and one-dimensional integration by parts has been employed in a straightforward way. This reduces the integral over $\log \|\dots\|$ to four one-dimensional integrals

$$\blacktriangleright \quad \int_{-1}^1 \int_{-1}^1 \log \|\dots\| dt ds = -2 + \int_{-1}^1 (1 + \alpha) \log \|(1 + \alpha)\mathbf{u} - (t - \beta)\mathbf{v}\| dt \quad (11)$$

$$+ \int_{-1}^1 (1 - \alpha) \log \|(-1 + \alpha)\mathbf{u} - (t - \beta)\mathbf{v}\| dt \quad (12)$$

$$+ \int_{-1}^1 (1 - \beta) \log \|(s + \alpha)\mathbf{u} - (1 - \beta)\mathbf{v}\| ds \quad (13)$$

$$+ \int_{-1}^1 (1 + \beta) \log \|(s + \alpha)\mathbf{u} - (-1 - \beta)\mathbf{v}\| ds. \quad (14)$$

④ Now we return to the computation of

$$\int_{-1}^1 \int_{-1}^1 \log \|s\mathbf{u} - t\mathbf{v} + \mathbf{z}\| dt ds, \quad (1.4.3.25)$$

where we have to distinguish two cases:

Case I: \mathbf{u}, \mathbf{v} from (1.4.3.22) are linearly independent. Then there are $\alpha, \beta \in \mathbb{R}$ such that $\mathbf{z} = \alpha\mathbf{u} + \beta\mathbf{v}$.

$$\blacktriangleright \quad \int_{-1}^1 \int_{-1}^1 \log \|s\mathbf{u} - t\mathbf{v} + \mathbf{z}\| dt ds = \int_{-1}^1 \int_{-1}^1 \log \|\mathbf{u}(s + \alpha) - \mathbf{v}(t - \beta)\| dt ds,$$

and we can apply the above formulas, see Code 1.4.3.30, Line 29–Line 37.

Case II: $\mathbf{v} = \zeta\mathbf{u}$ for some $\zeta \neq 0$ (parallel panels). The previous formulas cannot be used, but we can resort to the identity

$$\zeta \frac{\partial}{\partial s} \log \|su - tv + z\| = \zeta \frac{(su - tv + z) \cdot u}{\|su - tv + z\|^2} = -\frac{\partial}{\partial t} \log \|su - tv + z\|. \quad (1.4.3.26)$$

Straightforward integration by parts gives

$$\begin{aligned} & \int_{-1}^1 \int_{-1}^1 \log \|su - tv + z\| dt ds \\ &= \int_{-1}^1 \left\{ [t \log \|su - tv + z\|]_{t=-1}^{t=1} - \int_{-1}^1 t \frac{\partial}{\partial t} \log \|su - tv + z\| dt \right\} ds \\ &= \int_{-1}^1 \left\{ \log \|su - v + z\| + \log \|su + v + z\| + \int_{-1}^1 \zeta t \frac{\partial}{\partial s} \log \|su - tv + z\| \right\} ds \\ &= \int_{-1}^1 \log \|su - v + z\| + \log \|su + v + z\| ds + \zeta \int_{-1}^1 [\log \|su - tv + z\|]_{s=-1}^{s=1} dt \quad (1.4.3.27) \\ &= \int_{-1}^1 \log \|su - v + z\| ds + \int_{-1}^1 \log \|su + v + z\| ds + \\ & \quad \zeta \int_{-1}^1 t \log \|u - tv + z\| dt + \zeta \int_{-1}^1 t \log \|-u - tv + z\| dt. \end{aligned}$$

These formulas are implemented in Code 1.4.3.30, Line 20-Line 28.

⑤ Thus, the computations are reduced to evaluating integrals of the form

$$\int_{-1}^1 t^k \log \|tu + v\| dt \quad u, v \in \mathbb{R}^2, \quad t = 0, 1. \quad (1.4.3.28)$$

We elaborate the expressions for $k = 0$ and point out that the case $k > 0$ can be reduced to $k = 0$ by repeated integration by parts. With

$$\begin{aligned} & \|tu + v\|^2 = \alpha t^2 + \beta t + \gamma, \quad \alpha := \|u\|^2, \quad \beta := 2u \cdot v, \quad \gamma := \|v\|^2. \\ \blacktriangleright \quad & L := \int_{-1}^1 \log \|tu + v\| dt = \frac{1}{2} \int_{-1}^1 \log \|tu + v\|^2 dt = \frac{1}{2} \int_{-1}^1 \log(\alpha t^2 + \beta t + \gamma) dt. \end{aligned}$$

We have to proceed differently, depending on whether the argument of the logarithm has a zero or not. For the quadratic polynomial in t we examine the discriminant.

Case I: $4\alpha\gamma - \beta^2 = 0 \iff \beta = 2\sqrt{\alpha\gamma}$: argument of logarithm can vanish

$$L = \int_{-1}^1 \log \left((\sqrt{\alpha}t + \sqrt{\gamma})^2 \right) dt = 2 \int_{-1}^1 \log |\sqrt{\alpha}t + \sqrt{\gamma}| dt,$$

then distinguish cases, $\alpha \geq \gamma$ (split interval) and $\alpha < \gamma$, and use explicit principal, see Code 1.4.3.29, Line 26-Line 33.

Case I: $4\alpha\gamma - \beta^2 > 0$: $t \mapsto \alpha t^2 + \beta t + \gamma$ has no real zero and an explicit principal can be used, Code 1.4.3.29, Line 34-Line 41.

C++11 code 1.4.3.29: Evaluating integrals of the form (1.4.3.28) →GITLAB, [Mai08, Sect. 2]

```

2 Eigen::VectorXd splterative(int k, const Eigen::Vector2d& u,
3                             const Eigen::Vector2d& v)
4 {
5     double a = u.squaredNorm(); //  $\alpha = \|u\|^2$ 
6     double b = 2. * u.dot(v);   //  $\beta = 2u \cdot v$ 
7     double c = v.squaredNorm(); //  $\gamma = \|v\|^2$ 
8     double D = 0.;             // discriminant
9     Eigen::VectorXd val(k+1);   // return values
10
11     // Ensure one non-zero argument vector
12     double tmp = 4*a*c - b*b;
13     assert(fabs(u[0]) > EPS || fabs(u[1]) > EPS
14            || fabs(v[0]) > EPS || fabs(v[1]) > EPS);
15     // By Cauchy-Schwarz inequality tmp >= 0
16     assert(tmp >= -fabs(EPS*4*a*c));
17
18     // Numerically sound way of testing if discriminant = 0
19     if (tmp > EPS*4*a*c) D = sqrt(tmp);
20     else D = 0.;
21
22     // The case k=0: pure logarithmic integrand
23     if (fabs(u[0]) < EPS && fabs(u[1]) < EPS) { // constant integrand
24         val[0] = 2*log(c);
25     }
26     else if (D == 0.) { // Integrand is logarithm of a pure square
27         tmp = b + 2*a;
28         if (fabs(tmp) > EPS*a) val[0] = tmp * log( 0.25*tmp*tmp / a );
29         else val[0] = 0;
30         tmp = b - 2*a;
31         if (fabs(tmp) > EPS*a) val[0] -= tmp * log( 0.25*tmp*tmp / a );
32         val[0] = 0.5*val[0] / a - 4.0;
33     } //
34     else { // case D > 0: argument of logarithm has no zeros
35         tmp = c - a;
36         if (fabs(tmp) < EPS*c) val[0] = 0.5*M_PI;
37         else if (a < c) val[0] = atan( D /tmp );
38         else val[0] = atan( D /tmp ) + M_PI;
39
40         val[0] = (0.5*((b+2*a)*log(a+b+c)-(b-2*a)*log(a-b+c))+ D*val[0])/a-4.0;
41     } //
42     if (k == 0) return val;

```

C++11 code 1.4.3.30: Evaluating integrals of the form (1.4.3.21) →GITLAB, [Mai08, Sect. 3]

```

2 double computeWijAnalytic(const Eigen::Vector2d& a,
3                           const Eigen::Vector2d& b,
4                           const Eigen::Vector2d& c,
5                           const Eigen::Vector2d& d)
6 {
7     double hi = (b-a).squaredNorm(); // length2 of first panel [a,b]
8     double hj = (d-c).squaredNorm(); // length2 of second panel [c,d]
9     double val = 0.;
10    double lambda, mu;
11    // Vectors defined in (1.4.3.22)

```



```

12 Eigen::Vector2d x = (b-a)/2.;
13 Eigen::Vector2d y = (c-d)/2.;
14 Eigen::Vector2d z = (a+b-c-d)/2.;
15
16 // There hold different recursion formulae when the panels
17 // are parallel (det = 0) or not
18 double det = CrossProd2d(x,y);
19
20 if ( fabs(det) <= EPS*sqrt(hi*hj) ) { // parallel panels, Case II
21     if ( fabs(x[0]) < fabs(x[1]) )
22         lambda = y[1] / x[1];
23     else
24         lambda = y[0] / x[0];
25     // Evaluate the four integrals from (1.4.3.27)
26     val = 0.5*( lambda * ( slp(1, y, z-x) - slp(1, y, z+x) )
27                 + slp(0, x, z+y) + slp(0, x, z-y) );
28 } //
29 else { // x and y linearly independent, Case I
30     lambda = (z[0]*y[1] - z[1]*y[0]) / det;
31     mu = (x[0]*z[1] - x[1]*z[0]) / det;
32     // Integrals (I1)-(I4)
33     val = 0.25 * (-8 + (lambda+1)*slp(0, y, z+x) -
34                 (lambda-1)*slp(0, y, z-x)+
35                 (mu+1)*slp(0, x, z+y) -
36                 (mu-1)*slp(0, x, z-y));
37 } //
38 return -0.125*val/M_PI; // = -1/8π * val
39 }

```

Note that the test whether x and y are parallel in Line 20 takes into account the presence of roundoff errors. ┘

§1.4.3.31 (Local analytic formulas for double layer BIO) We consider the bilinear form

$$(\mathbf{v}, \phi) \mapsto \frac{1}{2\pi} \int_{\Gamma} \int_{\Gamma} \frac{(\mathbf{x} - \mathbf{y}) \cdot \mathbf{n}(\mathbf{y})}{\|\mathbf{x} - \mathbf{y}\|^2} \mathbf{v}(\mathbf{y}) \phi(\mathbf{x}) \, dS(\mathbf{y}) \, dS(\mathbf{x}), \quad \mathbf{v} \in H^{\frac{1}{2}}(\Gamma), \phi \in H^{-\frac{1}{2}}(\Gamma),$$

and its Galerkin discretization based on $S_1^0(\mathcal{G}) \times S_0^{-1}(\mathcal{G})$, that is \mathbf{v} is \mathcal{G} -piecewise linear and ϕ \mathcal{G} -piecewise constant. For a pair $(\pi, \pi') \in \mathcal{G} \times \mathcal{G}$ of panels the entries of the 2×1 interaction matrix can be computed from the two integrals

$$I_0 := \frac{1}{2\pi} \int_{\pi} \int_{\pi'} \frac{(\mathbf{x} - \mathbf{y}) \cdot \mathbf{n}(\mathbf{y})}{\|\mathbf{x} - \mathbf{y}\|^2} \, dS(\mathbf{y}) \, dS(\mathbf{x}), \quad (1.4.3.32)$$

$$I_1 := \frac{1}{2\pi} \int_{\pi} \int_{\pi'} \frac{(\mathbf{x} - \mathbf{y}) \cdot \mathbf{n}(\mathbf{y})}{\|\mathbf{x} - \mathbf{y}\|^2} \ell(\mathbf{y}) \, dS(\mathbf{y}) \, dS(\mathbf{x}), \quad (1.4.3.33)$$

where ℓ is (parametric) linear on π with vanishing mean. In the case of line segments $\pi = [p_1, q_1]$, $\pi' = [p_2, q_2]$, the unit normal vector field $\mathbf{n}(\mathbf{y})$ is constant on π' and a transformation to the reference interval $\tilde{I} =]-1, 1[$ yields ($\mathbf{n} \hat{=}$ normal to π')

$$I_0 = \frac{1}{8\pi} \|\mathbf{q}_1 - \mathbf{p}_1\| \|\mathbf{q}_2 - \mathbf{p}_2\| \int_{-1}^1 \frac{(s\mathbf{u} - t\mathbf{v} + \mathbf{z}) \cdot \mathbf{n}}{\|s\mathbf{u} - t\mathbf{v} + \mathbf{z}\|^2} \, dt \, ds, \quad (1.4.3.34)$$

$$I_1 = \frac{1}{8\pi} \|\mathbf{q}_1 - \mathbf{p}_1\| \|\mathbf{q}_2 - \mathbf{p}_2\| \int_{-1}^1 \frac{(s\mathbf{u} - t\mathbf{v} + \mathbf{z}) \cdot \mathbf{n}}{\|s\mathbf{u} - t\mathbf{v} + \mathbf{z}\|^2} t \, dt \, ds, \quad (1.4.3.35)$$

$$\text{with } \mathbf{u} = \frac{1}{2}(\mathbf{q}_1 - \mathbf{p}_1), \mathbf{v} = \frac{1}{2}(\mathbf{q}_2 - \mathbf{p}_2), \mathbf{z} := \frac{1}{2}(\mathbf{p}_1 + \mathbf{q}_1 - \mathbf{p}_2 - \mathbf{q}_2). \quad (1.4.3.22)$$

Also note that both integrals vanish in the case $\boldsymbol{\pi} = \boldsymbol{\pi}'$.

We exploit an identity similar to (1.4.3.23). For $\mathbf{M} \in \mathbb{R}^{2,2}, \mathbf{c} \in \mathbb{R}^2, \mathbf{n} \in \mathbb{R}^2$,

$$F(\mathbf{x}) := \frac{\mathbf{M}(\mathbf{x} - \mathbf{c}) \cdot \mathbf{n}}{\|\mathbf{M}(\mathbf{x} - \mathbf{c})\|^2} \Rightarrow (\mathbf{grad} F)(\mathbf{x}) = -\frac{2(\mathbf{M}(\mathbf{x} - \mathbf{c}) \cdot \mathbf{n})\mathbf{M}^\top \mathbf{M}(\mathbf{x} - \mathbf{c})}{\|\mathbf{M}(\mathbf{x} - \mathbf{c})\|^4} + \frac{\mathbf{M}^\top \mathbf{n}}{\|\mathbf{M}(\mathbf{x} - \mathbf{c})\|^2},$$

$$\blacktriangleright (\mathbf{x} - \mathbf{c}) \cdot \mathbf{grad} F(\mathbf{x}) = -F(\mathbf{x}). \quad (1.4.3.36)$$

As above we apply **Green's formula** from Thm. 1.1.6.1 on a domain $D \subset \mathbb{R}^2$

$$\int_D G(\mathbf{x})F(\mathbf{x}) \, d\mathbf{x} = -\int_D G(\mathbf{x}) (\mathbf{x} - \mathbf{c}) \cdot \mathbf{grad} F(\mathbf{x}) \, d\mathbf{x}$$

$$= \int_D \text{div}(G(\mathbf{x})(\mathbf{x} - \mathbf{c})) F(\mathbf{x}) \, d\mathbf{x} - \int_{\partial D} G(\mathbf{x}) (\mathbf{x} - \mathbf{c}) \cdot \mathbf{n}(\mathbf{x}) F(\mathbf{x}) \, dS(\mathbf{x}),$$

$$\blacktriangleright \int_D (G(\mathbf{x}) + \mathbf{grad} G(\mathbf{x}) \cdot (\mathbf{x} - \mathbf{c})) F(\mathbf{x}) \, d\mathbf{x} = \int_{\partial D} G(\mathbf{x}) (\mathbf{x} - \mathbf{c}) \cdot \mathbf{n}(\mathbf{x}) F(\mathbf{x}) \, dS(\mathbf{x}). \quad (1.4.3.37)$$

for any smooth function $G \in C^1(\overline{D})$. In particular, if G is linear, $G(\mathbf{x}) = \mathbf{d} \cdot \mathbf{x}$ for some $\mathbf{d} \in \mathbb{R}^2$, then the computation of $\int_D G(\mathbf{x})F(\mathbf{x}) \, d\mathbf{x}$ can be reduced to the case $G \equiv 1$ up to integrals on ∂D . Note that for **regular M** the term under the integral $\int_{\partial D}$ in (1.4.3.37) is bounded, since in this case

$$\exists c^-, c^+ > 0: c^- \|\mathbf{x} - \mathbf{c}\| \leq \|\mathbf{M}(\mathbf{x} - \mathbf{c})\| \leq c^+ \|\mathbf{x} - \mathbf{c}\| \quad \forall \mathbf{x}, \mathbf{c} \in \mathbb{R}^2.$$

As before we observe that if D is a **polygon** $\mathbf{x} \mapsto (\mathbf{x} - \mathbf{c}) \cdot \mathbf{n}(\mathbf{x})$ will be **constant** on all edges of D .

This formula can be applied, if \mathbf{u} and \mathbf{v} are linearly independent, compare (1.4.3.24). Conversely, if $\mathbf{u} \parallel \mathbf{v}$, $\mathbf{v} = \zeta \mathbf{u}$ for $\zeta \neq 0$, we can use

$$\zeta \frac{\partial}{\partial s} f(s, t) = -\frac{\partial}{\partial t} f(s, t), \quad f(s, t) := \frac{(\mathbf{s}\mathbf{u} - t\mathbf{v} + \mathbf{z}) \cdot \mathbf{n}}{\|\mathbf{s}\mathbf{u} - t\mathbf{v} + \mathbf{z}\|^2},$$

analogously to (1.4.3.26), e.g. [Mai08, p. 7],

$$\int_{-1}^1 \int_{-1}^1 t f(s, t) \, dt ds = \int_{-1}^1 \left\{ \left[t^2 f(s, t) \right]_{t=-1}^{t=1} - \int_{-1}^1 t^2 \frac{\partial}{\partial t} f(s, t) \, dt \right\} ds$$

$$= \int_{-1}^1 \left\{ \left[t^2 f(s, t) \right]_{t=-1}^{t=1} - \int_{-1}^1 \zeta t^2 \frac{\partial}{\partial s} f(s, t) \, dt \right\} ds$$

$$= \int_{-1}^1 (f(s, 1) - f(s, -1)) \, ds + \zeta \int_{-1}^1 t^2 (f(1, t) - f(-1, t)) \, dt$$

All these formulas are implemented in Code 1.4.3.40

Eventually, all two-dimensional integrals are reduced to integrals of rational functions of the form

$$\int_{-1}^1 \frac{t^k}{\|\mathbf{t}\mathbf{p} + \mathbf{q}\|^2} \, dt, \quad \mathbf{p}, \mathbf{q} \in \mathbb{R}^2, \quad (1.4.3.38)$$

whose evaluation is done in Code 1.4.3.39 based on [Mai08, Lemma 2.1].

C++11 code 1.4.3.39: Evaluating integrals of the form (1.4.3.38) →GITLAB, [Mai08, Sect. 2]

```

2  double dlp(int k, const Eigen::Vector2d& p, const Eigen::Vector2d& q)
3  {
4      // The full recursion is not implemented
5      assert(k<=2 && (k>=0));
6
7      double a = p.squaredNorm(); // a = <p,p>
8      double b = 2 * p.dot(q);    // b = 2 <p,q>
9      double c = q.squaredNorm(); // c = <q,q>
10     double D = 4*a*c-b*b;        // Discriminant
11     double root_D = 0.;
12     double G0 = 0., G1 = 0.;
13
14     assert(D>=-EPS*4*a*c); // In exact arithmetic, D >= 0
15     if (D > EPS*4*a*c){ root_D = sqrt(D);} else{ D = 0.0;}
16     if (D == 0.0){ G0 = 2./(c-a);} // linearly dependent vectors, [Mai08, (5)]
17     else // Denominator cannot vanish, integrate rational function
18     {
19         if (fabs(c-a) < EPS*fabs(c)){ G0 = M_PI/root_D;}
20         else if (a < c){ G0 = 2.*atan( root_D/(c-a) )/root_D;}
21         else
22             { G0 = 2.*(atan( root_D/(c-a) )+M_PI)/root_D;}
23     }
24
25     if (k >= 1) // First step of recursion for k=1
26     {
27         // g1-1 in [Mai08, Lemma 2.1]
28         G1 = -b*G0;
29         if (a+b+c > EPS*a){ G1 += log(a+b+c);}
30         if (a-b+c > EPS*a){ G1 -= log(a-b+c);}
31         G1 /= (2.*a);
32
33         // g2-1 in [Mai08, Lemma 2.1]
34         if (k == 2){return (2.-b*G1-c*G0)/a;}
35
36         return G1;
37     }
38     return G0;
39 }

```

C++11 code 1.4.3.40: Evaluating integrals (1.4.3.34) and (1.4.3.35) →GITLAB

```

2  void computeKijAnalytic(double* I0, double* I1,
3                          const Eigen::Vector2d& a, const Eigen::Vector2d& b,
4                          const Eigen::Vector2d& c, const Eigen::Vector2d& d)
5  {
6      double hi = (b-a).squaredNorm(); // hi = norm(b-a) squared
7      double hj = (d-c).squaredNorm(); // hj = norm(d-c) squared
8      Eigen::Vector2d n = unitNormal(c,d); // normal vector
9
10     Eigen::Vector2d u = a-b, v = d-c, w = c+d-a-b;
11     Eigen::Vector2d wpu = w+u, wmu = w-u;
12     Eigen::Vector2d wpv = w+v, wmv = w-v;
13
14     double dot_u_n = u.dot(n), dot_w_n = w.dot(n);
15     double dot_wpu_n = wpu.dot(n), dot_wmu_n = wmu.dot(n);
16     double det = CrossProd2d(u,v);
17
18     double lambda=0.0, mu=0.0;

```

```

19  if (fabs(det) <= EPS*sqrt(hi*hj)) { // u,v linearly dependent
20      if (fabs(u[0]) > fabs(u[1])) mu = v[0]/u[0];
21      else mu = v[1]/u[1];
22
23      *I0 = dot_w_n*( dlp(0,u,wpv)+dlp(0,u,wmv)+ mu*( dlp(1,v,wmu)-dlp(1,v,wpu) ) );
24      *I1 = dot_w_n*( dlp(0,u,wpv)-dlp(0,u,wmv)+ mu*( dlp(2,v,wmu)-dlp(2,v,wpu) ) ) *0.5;
25  }
26  else { // u,v linearly independent
27      if (a[0] == d[0] && a[1] == d[1]) {
28          *I0 = 2*( dot_wpu_n*dlp(0,v,wpu)+dot_u_n*dlp(1,u,wmv)+dot_w_n*dlp(0,u,wmv) );
29          *I1 = dot_wpu_n*dlp(1,v,wpu)-dot_u_n*dlp(1,u,wmv)-dot_w_n*dlp(0,u,wmv)
30              + 0.5*( *I0 );
31      }
32      else if (b[0] == c[0] && b[1] == c[1]) {
33          *I0 = 2*( dot_wmu_n*dlp(0,v,wmu)+dot_u_n*dlp(1,u,wpv)+dot_w_n*dlp(0,u,wpv) );
34          *I1 = dot_wmu_n*dlp(1,v,wmu)+dot_u_n*dlp(1,u,wpv)+dot_w_n*dlp(0,u,wpv)
35              - 0.5*( *I0 );
36      }
37      else {
38          mu = CrossProd2d(w,v)/det;
39          lambda = CrossProd2d(u,w)/det;
40
41          *I0 = (mu+1)*dot_wpu_n*dlp(0,v,wpu) - (mu-1)*dot_wmu_n*dlp(0,v,wmu)
42              + (lambda+1)*( dot_u_n*dlp(1,u,wpv) + dot_w_n*dlp(0,u,wpv) )
43              - (lambda-1)*( dot_u_n*dlp(1,u,wmv) + dot_w_n*dlp(0,u,wmv) );
44          *I1 = 0.5*( (mu+1)*dot_wpu_n*dlp(1,v,wpu) - (mu-1)*dot_wmu_n*dlp(1,v,wmu)
45              + (lambda+1)*( dot_u_n*dlp(1,u,wpv) + dot_w_n*dlp(0,u,wpv) )
46              + (lambda-1)*( dot_u_n*dlp(1,u,wmv) + dot_w_n*dlp(0,u,wmv) )
47              - lambda*( *I0 ) );
48      }
49  }
50  *I0 *= -0.125*sqrt(hi*hj)/M_PI;
51  *I1 *= -0.125*sqrt(hi*hj)/M_PI;
52 }

```

1.4.3.3 Recapitulated [NumCSE Chapter 7]: Aspects of Numerical Quadrature

Numerical quadrature studies the approximate evaluation of integrals $\int_D f(x) dx$ for a given domain $D \subset \mathbb{R}^d$, $d \in \mathbb{N}$, and a function $f : D \rightarrow \mathbb{R}$, for which at least a routine for point evaluation must be available (ensured, if f given in procedural form [NumCSE § 7.1.0.2]).

The simplest approach is the approximation of a one-dimensional integral by a weighted sum of function values.

Definition 1.4.3.41. 1D Quadrature formula (QF)/quadrature rule (QR) [NumCSE Def. 7.2.0.1]

An n -point (one-dimensional) **quadrature formula** (QF)/**quadrature rule** (QR) on $[a, b]$ provides an approximation of the value of an integral through a **weighted sum** of point values of the integrand:

$$\text{for } f : [a, b] \rightarrow \mathbb{R}: \quad \int_a^b f(t) dt \approx Q_n(f) := \sum_{j=1}^n w_j^n f(c_j^n). \quad (1.4.3.42)$$

Terminology: $w_j^n \hat{=}$ quadrature weights $\in \mathbb{R}$
 $c_j^n \hat{=}$ quadrature nodes $\in [a, b]$

Definition 1.4.3.43. Order of a quadrature rule [NumCSE Def. 7.4.1.1]

The **order** of quadrature rule $Q_n : C^0([a, b]) \rightarrow \mathbb{R}$ is defined as

$$\text{order}(Q_n) := \max\{m \in \mathbb{N}_0 : Q_n(p) = \int_a^b p(t) dt \quad \forall p \in \mathcal{P}_m\} + 1, \quad (1.4.3.44)$$

that is, as the **maximal** degree +1 of polynomials for which the quadrature rule is guaranteed to be exact.

Given a quadrature formula $(\hat{c}_j, \hat{w}_j)_{j=1}^n$ on, e.g., the reference interval $[-1, 1]$, a quadrature formula of the **same order** on $[a, b]$ is spawned by affine transformation:

$$\int_a^b f(t) dt \approx \frac{1}{2}(b-a) \sum_{j=1}^n \hat{w}_j \hat{f}(\hat{c}_j) = \sum_{j=1}^n w_j f(c_j). \quad (1.4.3.45)$$

with

$$\begin{aligned} \text{quadrature nodes} \quad c_j &= \frac{1}{2}(1 - \hat{c}_j)a + \frac{1}{2}(1 + \hat{c}_j)b, \\ \text{quadrature weights} \quad w_j &= \frac{1}{2}(b-a)\hat{w}_j. \end{aligned}$$

In words, the nodes are just mapped through the affine transformation $c_j = \Phi(\hat{c}_j)$, $\Phi(\tau) := \frac{1}{2}(1 - \tau)a + \frac{1}{2}(\tau + 1)b$, the weights are scaled by the ratio of lengths of $[a, b]$ and $[-1, 1]$.

EXAMPLE 1.4.3.46 (Trapezoidal rule [NumCSE Ex. 7.5.0.3]) A simple **composite quadrature formula** of (low) order 2 is the equidistant trapezoidal rule:

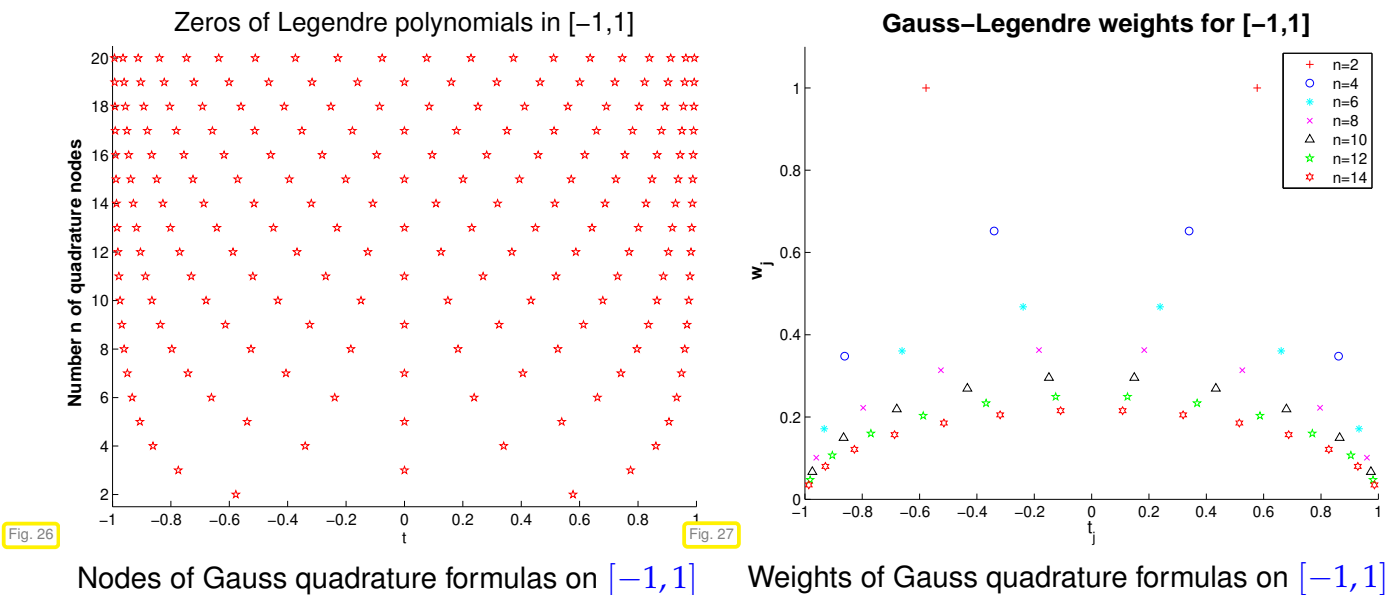
$$\int_a^b f(t) dt \approx \frac{1}{2n}f(a) + \frac{1}{n} \sum_{k=1}^{n-1} \left(a + \frac{b-a}{n}k\right) f\left(a + \frac{b-a}{n}k\right) + \frac{1}{2n}f(b), \quad n \in \mathbb{N}. \quad (1.4.3.47)$$

However, in the context of boundary element methods it is mainly global quadrature rules of high order that are relevant. ┘

§1.4.3.48 (Gauss(-Legendre) quadrature rules [NumCSE Section 7.4])**Theorem 1.4.3.49. Gauss(-Legendre) quadrature**

For every $n \in \mathbb{N}$ there is a **unique** n -point quadrature rule on $[-1, 1]$ of **maximal order** $2n$, the **Gauss(-Legendre) quadrature rule**.

It has **positive weights** and its nodes coincide with the zeros of the n -th **Legendre polynomial** $P_n \in \mathcal{P}_n$.



Nodes and weights of n -point Gauss(-Legendre) quadrature rules on $[-1, 1]$ can be computed efficiently by

- ◆ solving an $n \times n$ dense eigenvalue problem: **Golub-Welsch algorithm** [NumCSE Rem. 7.4.2.23],
- ◆ using Newton's method for finding the zeros of the Legendre polynomials (with initial guesses from asymptotic closed-form formulas) and then solving an $n \times n$ linear system to determine the weights [NumCSE Rem. 7.4.1.7].

In codes nodes and weights are often accessed by simple table look-up. ┘

§1.4.3.50 ("Practical" Clenshaw-Curtis quadrature rules [Tre08]) This is a family of quadrature rules on $[-1, 1]$ based on the quadrature nodes

$$c_j^n := \cos\left(\frac{(j-1)\pi}{n-1}\right), \quad j = 1, \dots, n. \tag{1.4.3.51}$$

These nodes form a set of dilated Chebychev nodes (1.4.2.55), which are known to be "optimal" for global polynomial interpolation [NumCSE Section 6.2.3]. The so-called **Clenshaw-Curtis quadrature rules** use the nodes (1.4.3.51) also for numerical quadrature and fix the weights in order to achieve **order $\geq n$** for the corresponding n -point quadrature formula.

Theorem 1.4.3.52. Positivity of Clenshaw-Curtis weights

For all $n \in \mathbb{N}$ the weights of the n -point Clenshaw-Curtis are positive.

Clenshaw-Curtis nodes in [-1,1]

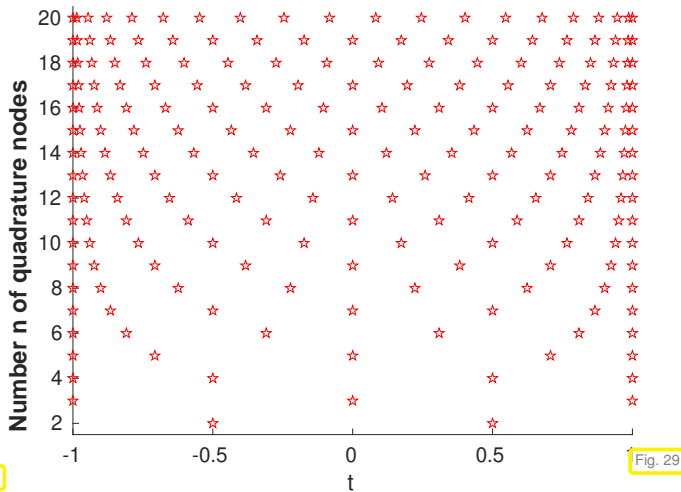


Fig. 28

Clenshaw-Curtis weights for [-1,1]

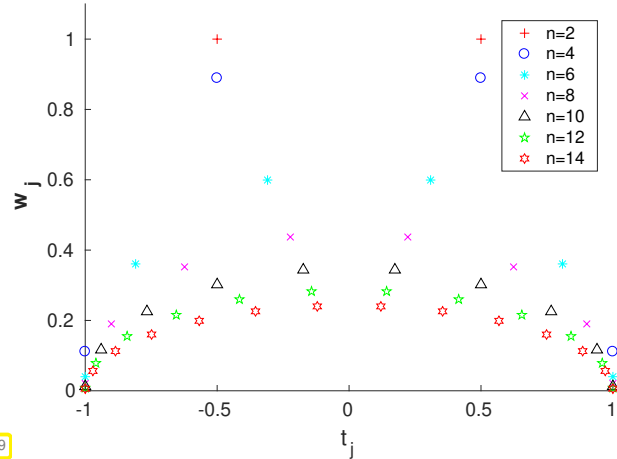


Fig. 29

Clenshaw-Curtis nodes (1.4.2.55) on $[-1, 1]$

Weights for Clenshaw-Curtis rule on $[-1, 1]$

The weights of any n -point Clenshaw-Curtis rule can be computed with a computational effort of $O(n \log n)$ using FFT. ┘

§1.4.3.53 (Generalized Gauss quadrature rules) The theory for Gauss(-Legendre) quadrature developed in [NumCSE Section 7.4] heavily relies on orthogonality with respect to the $L^2([-1, 1])$ inner product $(u, v) \mapsto \int_{-1}^1 u(t)v(t) dt$. A closer scrutiny reveals that the considerations remain valid for a large class of **weighted L^2 -inner products**.

We fix a **weight function** $w \in C^0([-1, 1])$ satisfying

$$w(t) > 0 \quad \forall t \in]-1, 1[\quad \text{and} \quad \int_{-1}^1 w(t) dt < \infty. \tag{1.4.3.54}$$

▶ The weight function w defines an **inner product** on $C^0([-1, 1])$ through $(u, v) \mapsto \int_{-1}^1 w(t)u(t)v(t) dt$.

Thus we can orthogonalize the monomials $\{t \mapsto t^k\}$, $k \in \mathbb{N}_0$, by means of the **Gram-Schmidt** algorithm as in [NumCSE Rem. 7.4.2.8].

Lemma 1.4.3.55. Generalized orthogonal polynomials [Han02, Sect. 33]

There exists a unique sequence of polynomials $(U_n)_{n \in \mathbb{N}_0}$ that fulfills

- (i) U_n is a polynomial of degree $\leq n$: $U_n \in \mathcal{P}_n$,
- (ii) U_n has leading coefficient 1: $U_n(t) = t^n + \dots$,
- (iii) U_n is "**w-orthogonal**" to all polynomials of smaller degree

$$\int_{-1}^1 w(t)U_n(t)p(t) dt = 0 \quad \forall p \in \mathcal{P}_{n-1}.$$

We used the Legendre polynomials to define the nodes for the Gauss-Legendre quadrature rules, and in the same vein we can harness the polynomials U_n , thus generalizing Thm. 1.4.3.49.

Theorem 1.4.3.56. Generalized Gauss quadrature

For every $n \in \mathbb{N}$ there exists an n -point (generalized) Gauss quadrature formula with nodes/weights c_j^n/w_j^n , $j = 1, \dots, n$, such that

$$\sum_{j=1}^n w_j^n p(c_j^n) = \int_{-1}^1 w(t) p(t) dt \quad \forall p \in \mathcal{P}_{2n-1}.$$

The nodes c_j^n are the zeros of the generalized orthogonal polynomials U_n and the weights are positive.

The generalized orthogonal polynomials satisfy a 3-term recurrence

$$U_{n+1}(t) = (t + \alpha_n)U_n(t) + \beta_n U_{n-1}(t), \quad \alpha_n, \beta_n \in \mathbb{R}.$$

Explicit formulas for α_n and β_n are known only for very few special weight functions w , of course for $w \equiv 1$ (Legendre polynomials, see [NumCSE Eq. (7.4.2.21)]). The accurate and stable computation of these recursion coefficients for general w is a challenging numerical problem [Gau18; Gau04].

§1.4.3.57 (Quadrature error [NumCSE § 7.2.0.12]) A natural concept for a quadrature rule Q_n is the

$$\text{quadrature error} \quad E_n(f) := \left| \int_a^b f(t) dt - Q_n(f) \right|$$

It is all but impossible to estimate the quadrature error for complicated integrands that may be given only implicitly. Therefore we have to be content with understanding the asymptotic behavior of the quadrature error for large numbers of quadrature nodes.

Definition 1.4.3.58. Asymptotic convergence of quadrature rules, cf. [NumCSE Def. 6.2.2.7]

Let $(Q_n)_{n \in \mathbb{N}}$ be a family of n -point quadrature rules for approximating $\int_a^b f(t) dt$. For a given function $f : [a, b] \rightarrow \mathbb{R}$ the quadrature errors $E_n(f)$ are said to

- converge algebraically with rate p , if $E_n(f) = O(n^{-p})$ for some $p \in \mathbb{N}$,
- converge exponentially, if $E_n(f) = O(q^n)$ for some $0 \leq q < 1$,

for $n \rightarrow \infty$.

Asymptotically, exponential convergence always beats algebraic convergence

We refer to [NumCSE § 6.2.2.9] on how to glean qualitative and quantitative information about the asymptotic behavior of the quadrature error from errors measured in numerical experiments. We may examine plots of the quadrature error versus the number of quadrature points:

- Exponential convergence manifests itself through points tracing out lines in semi-logarithmic plots.
- Algebraic convergence leads to points approximately lying on lines in a doubly logarithmic plot.

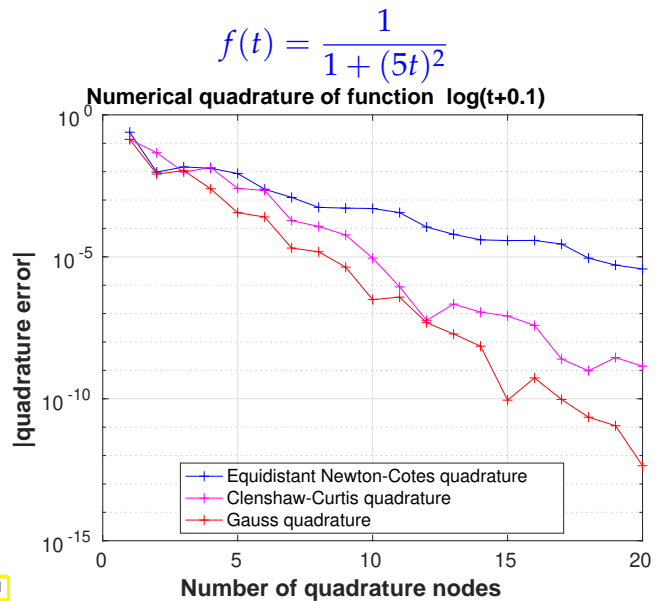
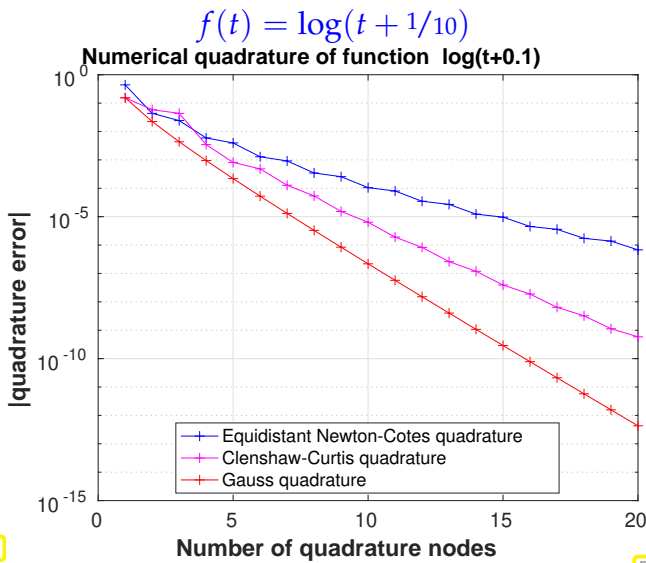
EXPERIMENT 1.4.3.59 (Behavior of quadrature errors for global quadrature rules) We monitor the error of global n -point quadrature rules on $[0, 1]$, $n \in \mathbb{N}$

- Newton-Cotes rule with equidistant nodes $c_k^n = \frac{k-1}{n-1}$, $k = 1, \dots, n$,
- n -point Gauss(-Legendre) rules according to Thm. 1.4.3.49,

- n -point Clenshaw-Curtis rule, nodes according to (1.4.3.51).

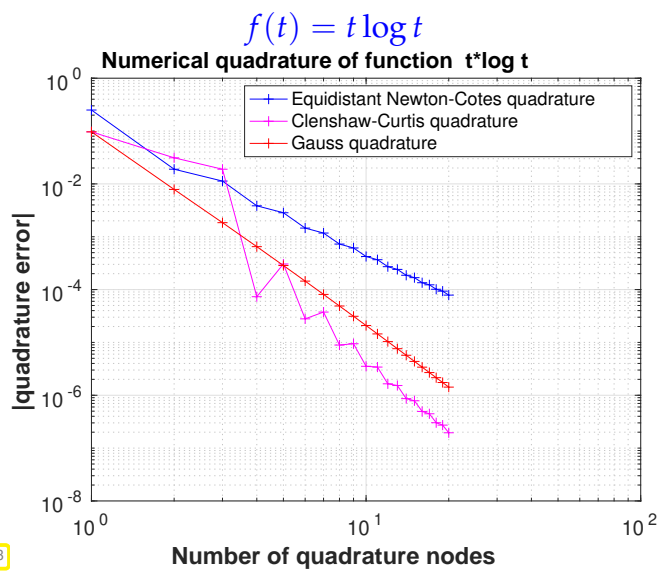
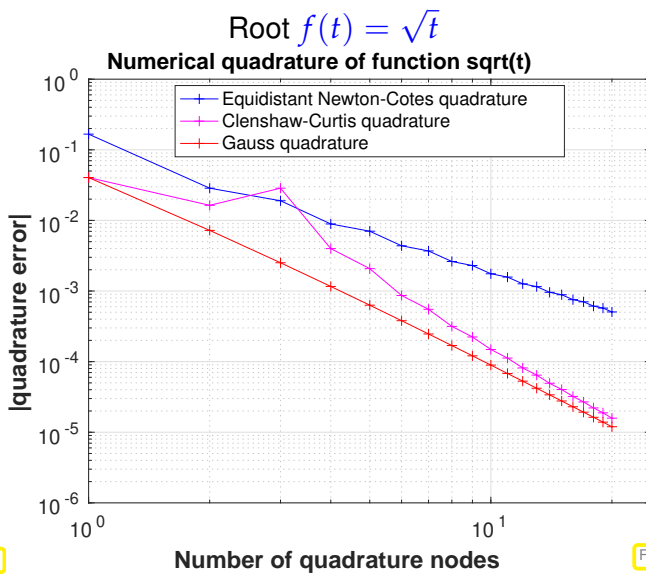
We apply these rules to different integrands $f \in C^0([0,1])$ and plot the quadrature errors for $n = 1, \dots, 20$.

❶ Smooth functions:



Observation: Exponential convergence for all quadrature rule, Gauss-Legendre rule fastest.

❷ functions with a (higher order) singularity:



Observation: Merely algebraic convergence for all quadrature rules, Gauss-Legendre rule again fastest.

❸ functions with (higher-order) kinks:

bump $f(t) = \begin{cases} \cos^2(4t - 2) & \text{for } |t - \frac{1}{2}| < \frac{1}{4}, \\ 0 & \text{elsewhere.} \end{cases}$

tent $f(t) = \begin{cases} 1 - |4t - 2| & \text{for } |t - \frac{1}{2}| < \frac{1}{4}, \\ 0 & \text{elsewhere.} \end{cases}$

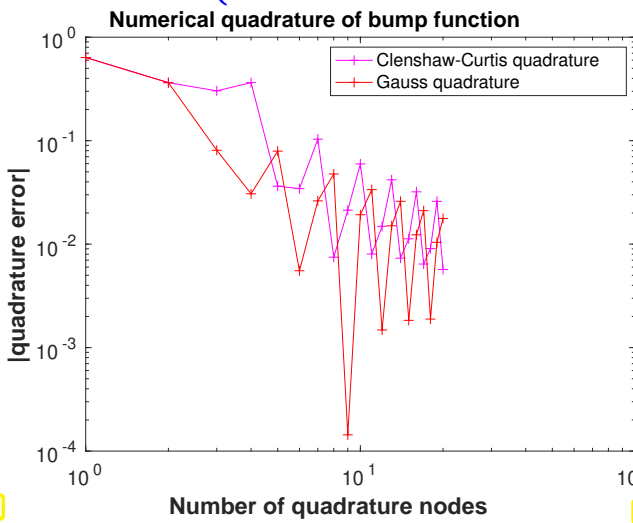


Fig. 35

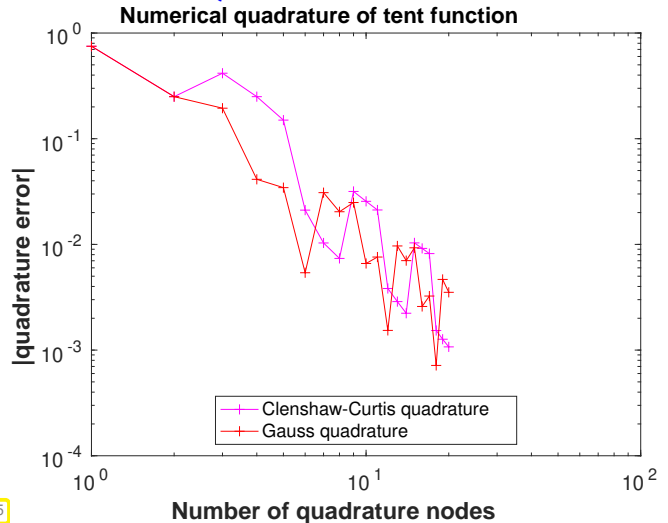


Fig. 34

Observation: We vaguely see algebraic convergence, big impact of presence of kinks.

§1.4.3.60 (Finite smoothness quadrature error estimates) If a quadrature rule is of order q , then the quadrature error does not change when adding a polynomial of degree $< q$ to the integrand:

$$E_n(f) = E_n(f - q) \quad \forall q \in \mathcal{P}_{q-1}.$$

In addition, the weights of a quadrature rule have to add up to the length of the interval. These two ideas plus the Δ -inequality yield the following result.

Lemma 1.4.3.61. Quadrature error and best-approximation error [NumCSE Thm. 7.4.3.3]

If Q is a quadrature formula on $[a, b]$ of order $q \in \mathbb{N}$ with positive weights, then the quadrature error can be estimated by

$$\left| \int_a^b f(t) dt - Q(f) \right| \leq 2|b - a| \inf\{\|f - p\|_{L^\infty([a,b])}, p \in \mathcal{P}_{q-1}\}. \quad (1.4.3.62)$$

► The quadrature error can be estimated by error (in maximum norm) of the polynomial best approximation.

Therefore polynomial best approximation estimates like [NumCSE Thm. 6.2.1.11] immediately translate into quadrature error estimates:

Theorem 1.4.3.63. Quadrature error estimate for integrands with finite smoothness

If $f \in C^m([a, b])$, $m \in \mathbb{N}_0$, and the quadrature rule Q is of order $q > m$ with positive weights, then

$$\begin{aligned} \left| \int_a^b f(t) dt - Q(f) \right| &\leq (4 + 2\pi^2) \left| \frac{b - a}{2} \right|^{m+1} \frac{(q - 1 - m)!}{(q - 1)!} \|f^{(m)}\|_{L^\infty([a,b])} \\ &\leq C(m) |b - a|^{m+1} \frac{1}{(q - 1)^m} \|f^{(m)}\|_{L^\infty([a,b])}, \end{aligned} \quad (1.4.3.64)$$

with an increasing function $C : \mathbb{N} \rightarrow \mathbb{R}^+$.

► Let $(Q_q)_{q \in \mathbb{N}}$ be a family of quadrature rules on $[a, b]$ with positive weights and Q_q have order q . If $f \in C^m([a, b])$ **at most**, then we expect asymptotic **algebraic convergence** of the quadrature error with rate m for $q \rightarrow \infty$:

$$\left| \int_a^b f(t) dt - Q_q(f) \right| = O(q^{-m}) \quad \text{for } q \rightarrow \infty. \tag{1.4.3.65}$$

┘

§1.4.3.66 (Quadrature error estimate for analytic integrands) What does Thm. 1.4.3.63 mean for $f \in C^\infty([a, b])$? If its derivatives do not grow “too fast” a very fast decay of the quadratur error can be predicted as the quadrature order $q \rightarrow \infty$.

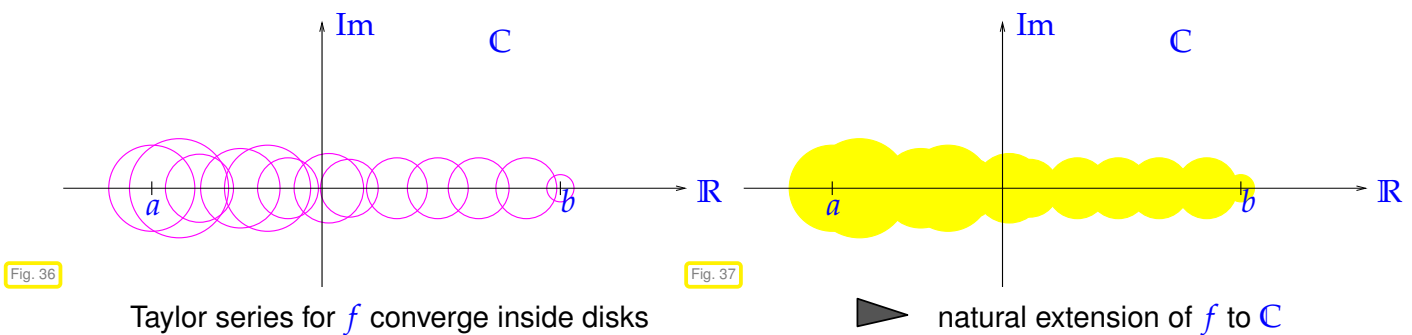
Now we meet functions whose derivatives do not grow “too fast” and we call them analytic. Analytic functions are locally “polynomials of infinite degree”, the class of general functions closest to polynomials:

Definition 1.4.3.67. Real analytic functions

A function $f \in C^\infty([a, b])$ is **analytic**, if for every $t \in [a, b]$ its Taylor series converges in a neighborhood of t :

$$\forall t \in [a, b]: \exists r_t > 0: f(\tau) = \sum_{k=0}^{\infty} \frac{(\tau - t)^k}{k!} f^{(k)}(t) \quad \forall \tau: |\tau - t| < r_t.$$

Since power series make perfect sense for complex arguments, we can replace $t \in \mathbb{R}$ with $z \in \mathbb{C}$ and obtain a complex-valued function defined on a neighborhood of $[a, b]$ in the complex plane \mathbb{C} , an **analytic extension** of f .



The analytic extension of f will also have locally convergent Taylor series:

Definition 1.4.3.68. Analyticity of a function in \mathbb{C}

Let $D \subset \mathbb{C}$ be an *open* set in the complex plane. A function $f : D \rightarrow \mathbb{C}$ is called **analytic/holomorphic** in D , if f has a representation as a convergent power series in a neighborhood of every $z \in D$:

$$\forall z \in D: \exists r_z > 0, (a_k)_{k \in \mathbb{N}_0}, a_k \in \mathbb{C}: f(w) = \sum_{k=0}^{\infty} a_k (w - z)^k \quad \forall w: |z - w| < r_z.$$

Functions $f \in [a, b]$ that possess an analytic extension into a sufficiently large \mathbb{C} -neighborhood of $[a, b]$ allow excellent approximation by polynomials, for instance, by their Chebychev interpolants, see [NumCSE Rem. 6.2.3.26].

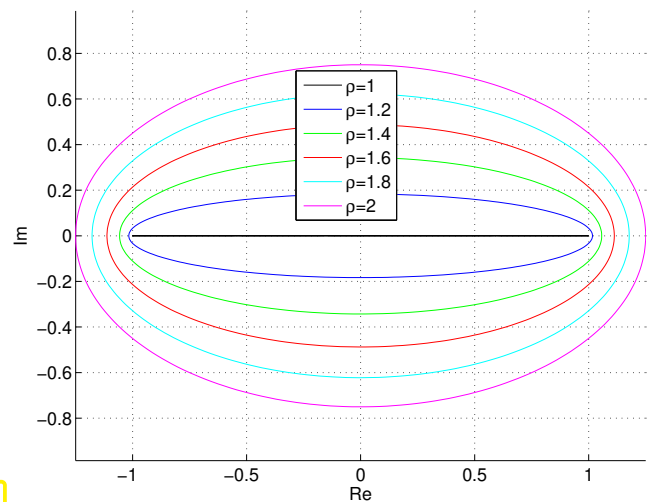
For the reference interval $[-1, 1]$ these particular \mathbb{C} -neighborhoods can be identified as ellipses with foci -1 and 1 :

$$\begin{aligned} \mathcal{E}_\rho &:= \{z \in \mathbb{C} : |z - 1| + |z + 1| = \rho + \rho^{-1}\} \\ &= \left\{ \begin{array}{l} z = \frac{1}{2}(\rho + \rho^{-1}) \cos \theta + \\ \quad i \frac{1}{2}(\rho - \rho^{-1}) \sin \theta, \\ 0 \leq \theta \leq 2\pi \end{array} \right\}, \end{aligned} \tag{1.4.3.69}$$

with a parameter $\rho > 0$ controlling the size of the ellipse. ▷

\mathcal{E}_ρ is often called **Bernstein ellipse**.

Fig. 38



Theorem 1.4.3.70. Polynomial approximation of analytic functions, [NumCSE Eq. (6.2.3.28)]

If $f : [-1, 1] \rightarrow \mathbb{C}$ possesses an analytic extension \tilde{f} to \mathbb{C} beyond the ellipse \mathcal{E}_ρ for a $\rho > 0$, then

$$\inf_{p \in \mathcal{P}_m} \|f - p\|_{L^\infty([-1,1])} \leq \frac{2|\mathcal{E}_\rho|}{\pi} \frac{1}{(\rho^{m+1} - 1)(\rho + \rho^{-1} - 2)} \cdot \max_{z \in \mathcal{E}_\rho} |f(z)|, \tag{1.4.3.71}$$

for all polynomial degrees $m \in \mathbb{N}_0$.

Obviously, the bound in (1.4.3.71) decays exponentially like $O(\rho^m)$ for $m \rightarrow \infty$. By virtue of Lemma 1.4.3.61 the same bound holds for the quadrature error of a quadrature rule with positive weights and order $q = m + 1$.

Asymptotics of quadrature error for analytic functions

If $f : [a, b] \rightarrow \mathbb{R}$ has an analytic extension to a neighborhood of an ellipse in \mathbb{C} with foci a and b , then the **quadrature errors** for both Gauss(-Legendre) quadrature and Clenshaw-Curtis quadrature will **decrease exponentially** in the number of quadrature points.

┘

EXPERIMENT 1.4.3.73 (Global quadrature of analytic integrand) We use Gauss-Legendre quadrature (\rightarrow Thm. 1.4.3.49) and Clenshaw-Curtis rules for the numerical quadrature of

$$t \mapsto \log(t + \alpha), \quad \alpha \in \{1.05, 1.01, 1.2, 1.4\} \text{ on } [-1, 1].$$

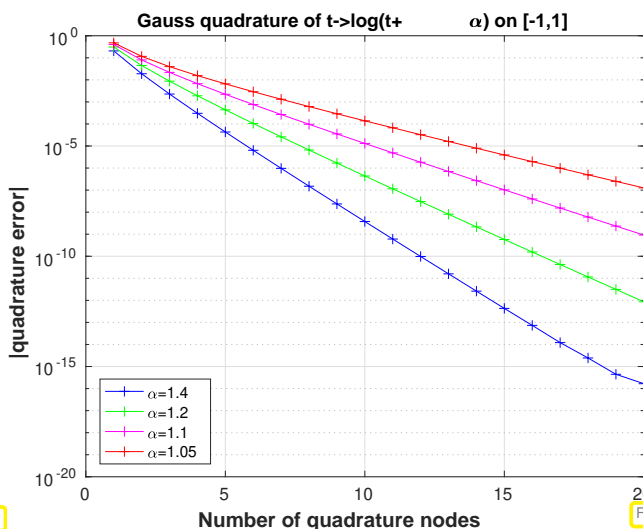


Fig. 39

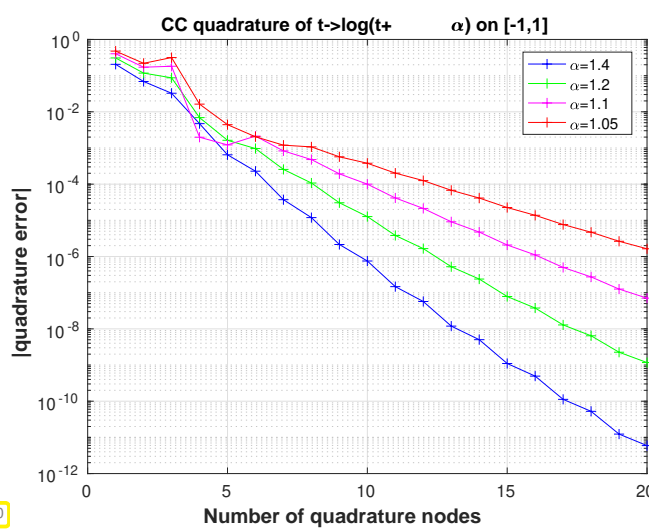


Fig. 40

Observation: The smaller $\alpha - 1$, the slower the exponential convergence of the quadrature error

The (main branch of the) logarithm $z \mapsto \log(z)$ is analytic on $\mathbb{C} \setminus \mathbb{R}_0^-$. Hence the domain of analyticity of $z \mapsto \log(z + \alpha)$ is $D_\alpha := \mathbb{C} \setminus]-\infty, -\alpha]$ and the range of ρ for which the ellipse $\mathcal{E}_\rho \subset D_\alpha$ shrinks for $\alpha \rightarrow 1$. The bound $O(\rho^{-n})$ from (1.4.3.71) will predict “flatter” exponential convergence as $\alpha \rightarrow 1$. \lrcorner

Summary: Significance of smoothness of integrand

The *maximal smoothness* of the integrand determines the quantitative asymptotic behavior of quadrature errors for *increasing quadrature order*:

- Integrand of class C^m only \triangleright algebraic convergence.
- Integrand has analytic extension \triangleright exponential convergence.

§1.4.3.75 (Adaptive global quadrature) The numerical quadrature of analytic integrands by means of Clenshaw-Curtis rules usually results in (slightly) larger errors than the use of Gauss(-Legendre) quadrature with the same number of nodes. Nevertheless, the Clenshaw-Curtis nodes (1.4.3.51) feature an obvious, but interesting nesting property:

$$\text{for } c_j^n \text{ from (1.4.3.51): } \quad c_{2j}^{2n} = c_j^n, \quad j = 1 \dots, n.$$

Thus, successively, using Clenshaw-Curtis rules with $n = 2, 4, 8, 16, \dots, s^L$ nodes, $L \in \mathbb{N}$, to approximate $\int_a^b f(t) dt$ requires only 2^L point evaluations of the integrand.

The following pseudo-code implements an adaptive Clenshaw-Curtis quadrature. It assumes that the corresponding nodes and weights (c_j^n, w_j^n) are available already in a table. The quadrature error is estimated by comparing results obtained for different numbers of quadrature points. Refer to [NumCSE Section 7.6] for a detailed discussion of ideas underlying adaptive quadrature controlled by specifying a relative tolerance $\text{rtol} > 0$ and and absolute tolerance $\text{atol} > 0$.

Pseudocode 1.4.3.76: Adaptive Clenshaw-Curtis quadrature

```

n := 3; {Start with 3 nodes}
y[1] = f(c_1^3); y[2] = f(c_2^3); y[3] = f(c_3^3);
I := w_1^3 y[1] + w_2^3 y[2] + w_3^3 y[3]; {evaluate quadrature formula}
repeat {main adaptive loop}
  I_old := I;
  n := 2 * (n - 1) + 1; {next number of nodes}
  y[n] := y[(n - 1) / 2 + 1];
  for j := (n - 1) / 2 downto 1 do
    y[2 * j - 1] = y[j]; {reuse previous function values}
    y[2 * j] = f(c_{2j}^n); {additional f-evaluations}
  endfor
  I := \sum_{k=1}^n w_k^n y[k]; {evaluate quadrature formula}
  \epsilon := |I - I_old|; {estimate for quadrature error}
{Check termination criterion based on absolute and relative tolerance}
until (\epsilon < \text{rtol} \cdot I or \epsilon < \text{atol} or n \geq n_{\max});
return(I);

```

\lrcorner

EXPERIMENT 1.4.3.77 (Adaptive Clenshaw-Curtis quadrature) We test the algorithm of ?? for a family

of quadrature problems with a “nearly singular” integrand:

$$\int_{-1}^1 f(t) dt, \quad f(t) = \log(t + \alpha) \quad \text{on } [-1, 1], \quad \alpha > 1.$$

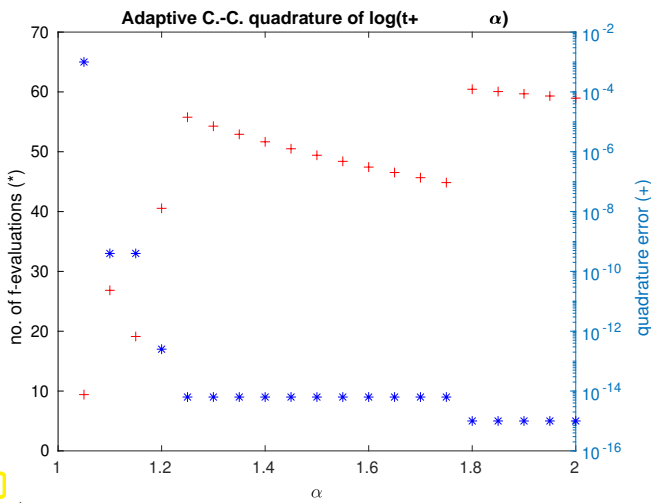


Fig. 41

◁ For $rtol = 10^{-2}$, $rtol = 10^{-6}$, total number of f -evaluations using the algorithm of Code 1.4.3.76.

The adaptive strategy triggers the use of a higher-order quadrature rule, whenever the proximity of the singularity might affect accuracy.

§1.4.3.78 (Tensor-product quadrature, cf. [NumPDE Ex. 2.7.5.38]) Nested quadrature formulas can be used to integrate bi-variate functions over tensor-product domains. Given a quadrature formula

$$Q_n(f) := \sum_{j=1}^n w_j^n f(c_j^n) \approx \int_{-1}^1 f(t) dt, \quad f \in C^0([-1, 1]),$$

we can use it to approximate integrals over $[-1, 1]^2$: for $F \in C^0([-1, 1]^2)$

$$\int_{-1}^1 \int_{-1}^1 F(s, t) dt ds \approx \sum_{j=1}^n w_j^n \int_{-1}^1 F(c_j^n, t) dt \approx \sum_{j=1}^n w_j^n \sum_{k=1}^n w_k^n F(c_j^n, c_k^n).$$

Thus we have found the derived two-dimensional **tensor-product quadrature formula**

$$\int_{-1}^1 \int_{-1}^1 F(s, t) dt ds \approx \sum_{j=1}^n \sum_{k=1}^n w_j^n w_k^n F(c_j^n, c_k^n) =: Q_{n \times n}(F), \tag{1.4.3.79}$$

with nodes $(c_j^n, c_k^n) \in \mathbb{R}^2$, and weights $w_j^n w_k^n, j, k = 1, \dots, n$.

This approach can easily be generalized to even higher dimensions and the combination of different quadrature formulas with different numbers of points in different directions.

If the underlying one-dimensional quadrature rule has order q , then $Q_{n \times n}$ will be exact for **tensor product polynomials** of degree $\leq q - 1$.

Definition 1.4.3.80. Tensor-product polynomials

The space of **tensor product polynomials** of (separate) degree $p \in \mathbb{N}$ in d dimensions is

$$\mathcal{TP}_p(\mathbb{R}^d) := \{x \mapsto q_1(x_1) \cdots q_d(x_d), q_i \in \mathcal{P}_p, i = 1, \dots, d\}.$$

For tensor-product quadrature formulas we define their order relying on exactness on spaces of tensor-product polynomials:

$$\text{order of } Q_n = m \implies Q_{n \times n}(q) = \int_{-1}^1 \int_{-1}^1 q(s, t) \, dt ds \quad \forall q \in \mathcal{TP}_{m-1}(\mathbb{R}^2).$$

As in one dimension, see Lemma 1.4.3.61, quadrature error and best approximation error in $\mathcal{TP}_m(\mathbb{R}^2)$ are closely related: If the one-dimensional quadrature rule Q_n is of order m , then

$$\begin{aligned} Q_{n \times n}(F) - \int_{-1}^1 \int_{-1}^1 F(s, t) \, dt ds \\ \leq \left(1 + \left(\sum_{k=1}^n |w_j^n| \right)^2 \right) \inf \{ \|F - P\|_{L^\infty([-1,1]^2)}, P \in \mathcal{TP}_{m-1}(\mathbb{R}^2) \} \\ \leq 5 \inf \{ \|F - P\|_{L^\infty([-1,1]^2)}, P \in \mathcal{TP}_{m-1}(\mathbb{R}^2) \}, \quad (1.4.3.81) \end{aligned}$$

if Q_n has positive weights.

Without going into details we point out that nested interpolation and approximation estimates make it possible to exploit Thm. 1.4.3.70 also in higher dimensions:

If both $\{t \mapsto F(s, t)\}$ and $\{s \mapsto F(s, t)\}$ allow an analytic extension to an ellipse neighborhood of $[-1, 1]$ in \mathbb{C} independent of the other variable, then the quadrature error of $Q_{n \times n}(F)$ will decay exponentially for $n \rightarrow \infty$, provided that Q_n has positive weights and order $\approx n$.

┘

1.4.3.4 Matrix Entries by Quadrature

We admit a general closed connected curve complying with Ass. 1.2.1.5: It can be split into $M \in \mathbb{N}$ edges Γ_j , $j = 1, \dots, M$, each available through a parameterization $\gamma_j : [-1, 1] \rightarrow \bar{\Gamma}_j$, see also (1.4.2.4). Every parameterization fulfills

$$\exists c > 0: \quad \|\dot{\gamma}_j(t)\| \geq c \quad \forall t \in [-1, 1], j = 1, \dots, M. \quad (1.4.3.82)$$

§1.4.3.83 (Data structure for general parameterization) When the use of a parameterization of an edge or of a single panel in a code is mentioned, one should read this as the availability of an object of the following type.

C++11 code 1.4.3.84: Model class representing a smooth parameterization (incomplete listing), →GITLAB

```

1 class CurveParam
2 {
3     public:
4         // :
5         // Querying the parameter interval
6         std::pair<double, double> ParameterRange(void) const;
7         // Accessing a point  $\gamma(t)$  on the edge/panel
8         Eigen::Vector2d operator()(double t) const;
9         // Retrieving the derivative  $\dot{\gamma}(t)$ , a tangent vector
10        Eigen::Vector2d Derivative(double t) const;
11        // :
12    };

```

The parameterizations of edges are supposed to be “maximally smooth”:

Assumption 1.4.3.85. Analytic parameterization

All parameterizations γ_j possess an analytic extension (\rightarrow § 1.4.3.66) beyond $[-1, 1]$.

Parlance: When, in the sequel, using the term “analytic” for a function of one or two variables on a bounded interval, we actually mean the possibility of analytic extension to an ellipse neighborhood of that interval, *cf.* Thm. 1.4.3.70.

Ass. 1.4.3.85 is obviously satisfied, if γ is a polynomial and for many function systems (NURBS) used in CAD modeling.

We endow Γ with a mesh $\mathcal{G} = \{\pi_1, \dots, \pi_N\}$ according to Def. 1.4.2.5. For each panel $\pi \in \mathcal{G}$ the relevant parameterization induces a local parameterization $\gamma_\pi : [-1, 1] \rightarrow \bar{\pi}$ as defined in (1.4.2.25).

Writing $k(x, y)$ for the kernel of some boundary integral operator (single layer BIO V or double layer BIO K, K'), this section is devoted to the approximate computation of the entries of the **interaction matrix**

$$\int_{\pi} \int_{\pi'} k(x, y) b_{\pi'}^j(y) b_{\pi}^i(x) dS(y) dS(x), \quad i, j \in \{1, \dots, Q\}, \quad (1.4.3.86)$$

where $b_{\pi}^1, \dots, b_{\pi}^Q$ are the local shape functions (\rightarrow § 1.4.2.21) associated with the panel π .

§1.4.3.87 (Transformation to reference interval) The first step in the computation of (1.4.3.86) employs transformation to the reference interval $\hat{I} =]-1, 1[$.

$$\begin{aligned} & \int_{\pi} \int_{\pi'} k(x, y) b_{\pi'}^j(y) b_{\pi}^i(x) dS(y) dS(x) \\ &= \int_{-1}^1 \int_{-1}^1 k(\gamma_{\pi}(s), \gamma_{\pi'}(t)) \hat{b}^j(t) \hat{b}^i(s) \|\dot{\gamma}_{\pi'}(t)\| \|\dot{\gamma}_{\pi}(s)\| dt ds, \end{aligned} \quad (1.4.3.88)$$

with **reference shape functions** \hat{b}^j, \hat{b}^i as defined in (1.4.2.27).

We have assumed that γ_{π} and $\gamma_{\pi'}$ are **analytic** with $\|\dot{\gamma}_{\pi}\|$ and $\|\dot{\gamma}_{\pi'}\|$ bounded away from zero on $[-1, 1]$. Moreover, for customary boundary element spaces like $S_p^0(\mathcal{G})$ or $S_p^{-1}(\mathcal{G})$ the reference shape functions are simple polynomials, *cf.* (1.4.2.28) and (1.4.2.29). Thus the task amounts to computing integrals

$$\int_{-1}^1 \int_{-1}^1 \hat{k}(s, t) F(t) G(s) dt ds, \quad \hat{k}(s, t) := k(\gamma_{\pi}(s), \gamma_{\pi'}(t)), \quad (1.4.3.89)$$

for **analytic** functions $F, G : [-1, 1] \rightarrow \mathbb{R}$. Note that the kernel \hat{k} might inherit the singularities of k , if $\bar{\pi} \cap \bar{\pi}' \neq \emptyset$ (touching/overlapping panels). \lrcorner

§1.4.3.90 (Single layer BIO: Identical panels) We consider $k = G^{\Delta}$ and $\pi = \pi'$, in which case (1.4.3.89) becomes

$$I := \int_{-1}^1 \int_{-1}^1 \log \|\gamma_{\pi}(s) - \gamma_{\pi}(t)\| F(t) G(s) dt ds. \quad (1.4.3.91)$$

Using calculus for log we rewrite the kernel:



$$2 \log \|\gamma_\pi(s) - \gamma_\pi(t)\| = \log \left(\frac{\|\gamma_\pi(s) - \gamma_\pi(t)\|^2}{(s-t)^2} \right) + 2 \log |s-t|. \quad (1.4.3.92)$$

We examine the first term and, in particular, the “difference quotient” in the argument of the logarithm,

$$S(s,t) := \begin{cases} \frac{\|\gamma_\pi(s) - \gamma_\pi(t)\|^2}{(s-t)^2} & \text{for } s \neq t, \\ \|\dot{\gamma}_\pi(t)\|^2 & \text{for } s = t. \end{cases} \quad (1.4.3.93)$$

In this formula we have already filled the gap at $s = t$ with the norm of the derivative $\dot{\gamma}$.

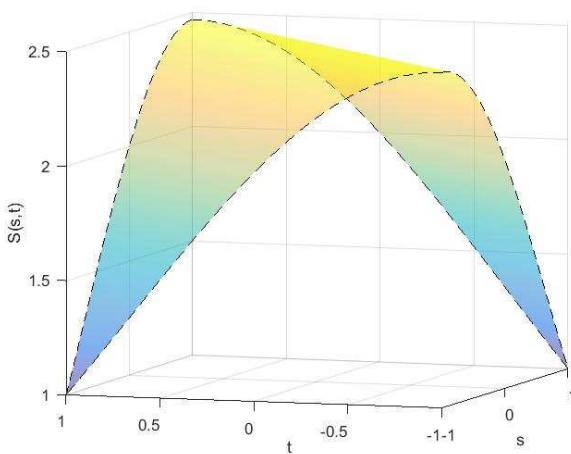


Fig. 42

◁ Plot of $S(s,t)$ for the semi-circle

$$\gamma(t) = \begin{bmatrix} \cos(\frac{\pi t}{2}) \\ \sin(\frac{\pi t}{2}) \end{bmatrix}, \quad -1 \leq t \leq 1.$$

The plot shows a perfectly smooth function nicely bounded away from zero.

Actually, we find by means of Taylor expansion that for the analytic function γ_π the difference quotient $S(s,t)$ is still analytic in both variables $s,t \in [-1,1]$. Hence, since $\|\dot{\gamma}_\pi(t)\| \geq c > 0$ on $[-1,1]$, also $(s,t) \mapsto \log D(s,t)$ is analytic, and

$$I = \int_{-1}^1 \int_{-1}^1 \underbrace{\frac{1}{2} \log(S(s,t)) F(t)G(s)}_{\text{analytic}} dt ds + \int_{-1}^1 \int_{-1}^1 \underbrace{\log |t-s| F(t)G(s)}_{\text{singular}} dt ds =: I_1 + I_2, \quad (1.4.3.94)$$

splits into an integral with an analytic integrand and one with a singular. Thus,

an exponentially convergent approximation of I_1 is provided by **tensor-product Gaussian quadrature** (\rightarrow § 1.4.3.78).



Idea: Move location of singularities of integrands to a coordinate axis by an affine transformation of the integration domain.

In the second integral in (1.4.3.94) the singularities of the integrand are located at the diagonal $\{s = t\}$ of the square. In the spirit of the policy just described, we tackle I_2 by the linear transformation

$$\begin{bmatrix} z \\ w \end{bmatrix} := \Phi^{-1} \begin{bmatrix} s \\ t \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} s \\ t \end{bmatrix} \Leftrightarrow \begin{bmatrix} s \\ t \end{bmatrix} = \Phi \begin{bmatrix} z \\ w \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} z \\ w \end{bmatrix}. \quad (1.4.3.95)$$

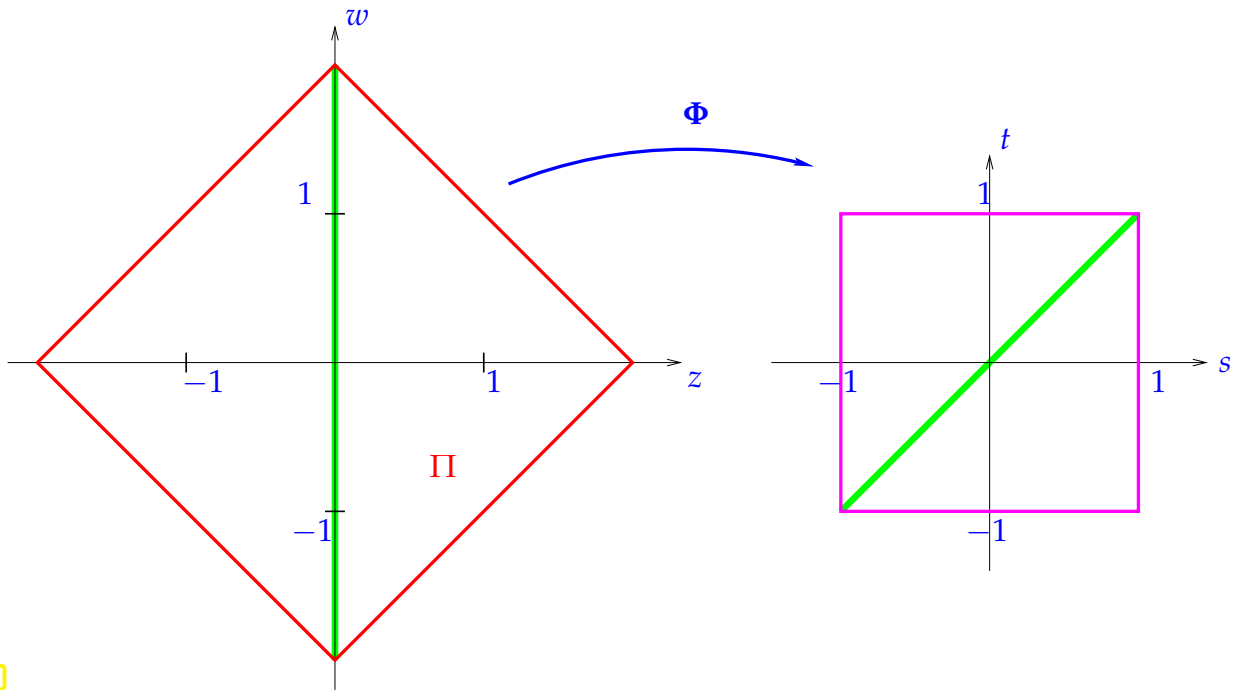


Fig. 43

Then apply the transformation formula for integrals over $D \subset \mathbb{R}^2$ [Str09, Sect. 8.5]

$$\int_D f(x) dx = \int_{\Phi^{-1}(D)} f(\Phi(\hat{x})) |\det D\Phi(\hat{x})| d\hat{x}, \quad f \text{ integrable on } D. \tag{1.4.3.96}$$

$$\blacktriangleright \quad I_2 = 2 \int_{\Phi^{-1}([-1,1]^2)} \log |z| F\left(\frac{1}{2}(w-z)\right) G\left(\frac{1}{2}(w+z)\right) dzdw \tag{1.4.3.97}$$

The integral over the square $\Pi := \Phi^{-1}([-1,1]^2)$ (left in Fig. 43) is split into the left and right half and then we add the contributions

$$I_2 = 2 \int_0^2 \log(z) \underbrace{\int_{-2+z}^{2-z} \overbrace{F\left(\frac{1}{2}(w-z)\right)G\left(\frac{1}{2}(w+z)\right) - F\left(\frac{1}{2}(w+z)\right)G\left(\frac{1}{2}(w-z)\right)}^{\text{analytic in } (z,w)} dw}_{\text{analytic as a function of } z} dz. \tag{1.4.3.98}$$

The inner integral is amenable to standard Gaussian quadrature. Then we face an integral of the form $\int_0^2 \log z f(z) dz$ with an analytic function $f : [0,2] \rightarrow \mathbb{R}$.

Generalized Gaussian quadrature (\rightarrow § 1.4.3.53) with *weight* $\log(z)$ can approximate I_2 with exponential accuracy.

┘

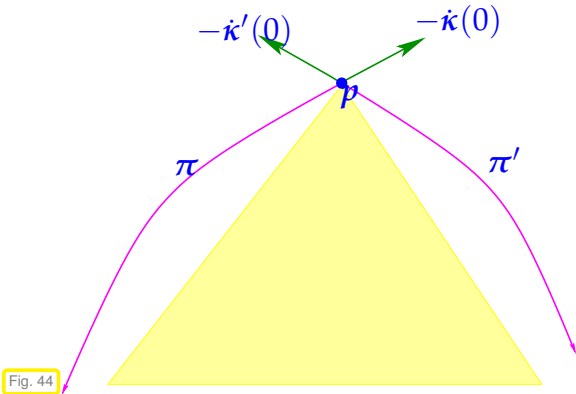
§1.4.3.99 (Single layer BIO: Adjacent panels) We assume $\pi, \pi' \in \mathcal{G}$, $\pi \neq \pi'$, $\overline{\pi} \cap \overline{\pi'} = \{p\}$. Writing $|\pi|, |\pi'|$ for the length of π and π' , respectively, in this § we will make use of a local **arclength parameterization**

$$\begin{aligned} \text{[for } \pi]: \quad & \kappa : [0, |\pi|] \rightarrow \overline{\pi}, \quad \|\dot{\kappa}(t)\| = 1 \quad \forall t \in [0, |\pi|], \\ \text{[for } \pi']: \quad & \kappa' : [0, |\pi'|] \rightarrow \overline{\pi'}, \quad \|\dot{\kappa}'(t)\| = 1 \quad \forall t \in [0, |\pi'|]. \end{aligned} \tag{1.4.3.100}$$

Thus, after transformation to the parameter domain, the entries of the interaction matrix for (π, π') are given by integrals

$$J := \int_0^{|\pi|} \int_0^{|\pi'|} \log \|\kappa(s) - \kappa'(t)\| F(t)G(s) dt ds, \tag{1.4.3.101}$$

with suitable univariate analytic functions F and G .



Uniform cone condition:

Lipschitz property of Γ entails lower bound on angle enclosed by π and π' :

$$\dot{\kappa}(0) \cdot \dot{\kappa}'(0) \leq c_{\angle} < 1. \tag{1.4.3.102}$$

◁ The panels cannot invade the yellow cone.



Taking the cue from (1.4.3.92) we split the kernel according to

$$\log \|\kappa(s) - \kappa'(t)\| = \frac{1}{2} \left(\log \frac{\|\kappa(s) - \kappa'(t)\|^2}{s^2 + t^2} + \log(s^2 + t^2) \right). \tag{1.4.3.103}$$

By Taylor expansion around $s = t = 0$:

$$\kappa(s) - \kappa'(t) = \dot{\kappa}(0)s - \dot{\kappa}'(0)t + O(s^2 + t^2) \text{ for } s, t \approx 0, \tag{1.4.3.104}$$



$$\|\kappa(s) - \kappa'(t)\|^2 = s^2 + t^2 - 2st\dot{\kappa}(0) \cdot \dot{\kappa}'(0) + O(s^4 + t^4) \text{ for } s, t \approx 0. \tag{1.4.3.105}$$

The prominent presence of $s^2 + t^2$ suggests that we introduce **polar coordinates** (r, φ) , see [NumPDE ??], according to



$$s = r \cos \varphi, \quad t = r \sin \varphi,$$

with r, φ in a suitable range that makes (s, t) cover $D := [0, |\pi|] \times [0, |\pi'|]$.

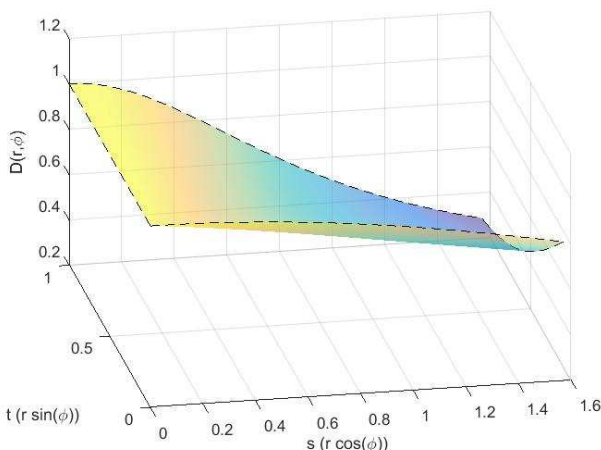
In polar coordinates the result of the above Taylor expansion reads

$$\|\kappa(s) - \kappa'(t)\|^2 = r^2(1 - \sin(2\varphi)\dot{\kappa}(0) \cdot \dot{\kappa}'(0) + O(r^2)) \text{ for } r \rightarrow 0.$$

▶ Due to (1.4.3.102) we can take for granted that the logarithm of

$$D(r, \varphi) := \begin{cases} \frac{\|\kappa(r \cos \varphi) - \kappa'(r \sin \varphi)\|^2}{r^2} & , \text{ if } (s, t) \neq (0, 0), \\ 1 - \sin(2\varphi)\dot{\kappa}(0) \cdot \dot{\kappa}'(0) \geq 1 - c_{\angle} & , \text{ if } r = 0, \end{cases} \tag{1.4.3.106}$$

is **analytic** on $D := [0, |\pi|] \times [0, |\pi'|]$.



◁ Plot of $D(r, \varphi)$ for the

$$\begin{aligned} \kappa(s) &= \begin{bmatrix} \cos(t) \\ \sin(t) \end{bmatrix}, & 0 \leq t \leq \frac{\pi}{2}, \\ \kappa'(t) &= \begin{bmatrix} 1-t \\ 0 \end{bmatrix}, & 0 \leq t \leq 1. \end{aligned}$$

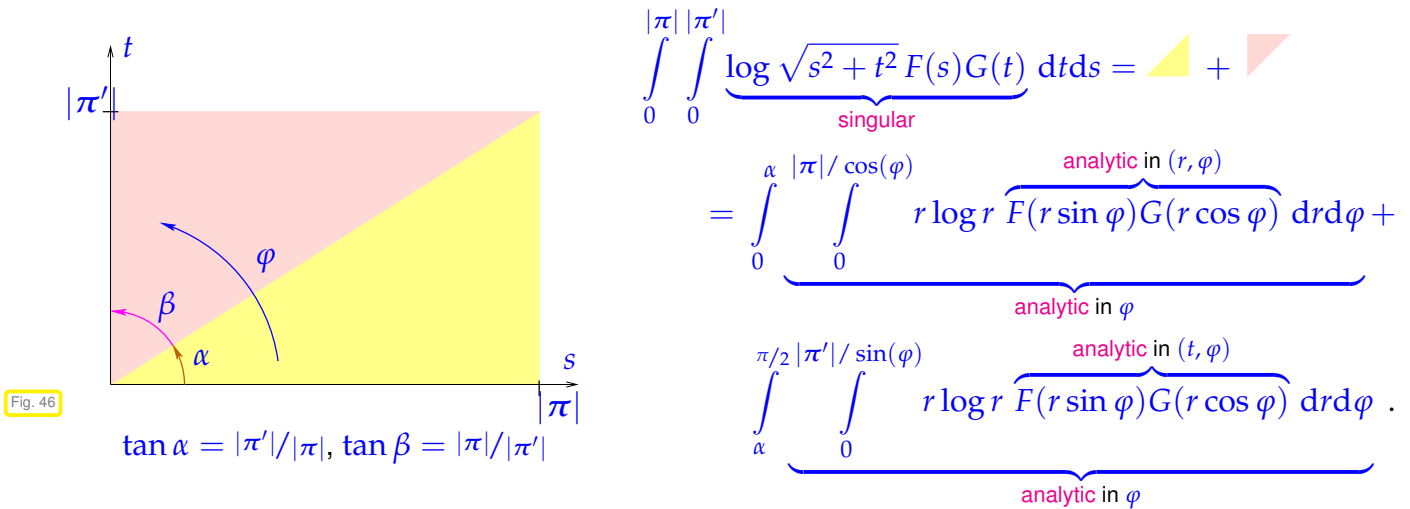
The graph looks perfectly smooth, hinting at a benign dependence of D on the polar coordinates (r, φ) .

Fig. 45

Analogously to (1.4.3.94) the integral can be split into two summands with “nice” and “nasty” integrands, respectively.

$$J = \frac{1}{2} \int_D \underbrace{r \log(D(r, \varphi)) F(t \cos \varphi) G(r \sin \varphi)}_{\text{analytic}} dr d\varphi + \int_D \underbrace{r \log r F(s) G(t)}_{\text{singular}} dr d\varphi \quad (1.4.3.107)$$

The first summand is amenable to tensor-product Gauss quadrature. The domain D of integration has to be decomposed in two triangles for integration in polar coordinates.



This suggest that we use

- ◆ generalized Gaussian quadrature formulas (\rightarrow § 1.4.3.53) with weight $r \rightarrow r \log r$ for the inner integral,
- ◆ standard Gaussian quadrature for the outer integral.

┘

§1.4.3.108 (Double layer BIO: Coinciding panels) In the case $\pi = \pi'$ (local analytic parameterization $\gamma_\pi : [-1, 1] \rightarrow \overline{\pi}$), for the double layer BIO K we have to approximate integrals of the form

$$K := \int_{-1}^1 \int_{-1}^1 \frac{(\gamma_\pi(s) - \gamma_\pi(t)) \cdot \mathbf{n}(\gamma_\pi(t))}{\|\gamma_\pi(s) - \gamma_\pi(t)\|^2} F(t) G(s) dt ds , \quad (1.4.3.109)$$

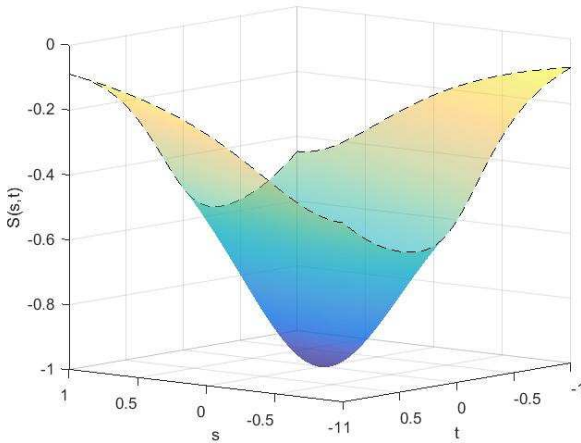


Fig. 47

◁ plot of

$$(s, t) \mapsto \frac{(\gamma_\pi(s) - \gamma_\pi(t)) \cdot \mathbf{n}(\gamma_\pi(t))}{\|\gamma_\pi(s) - \gamma_\pi(t)\|^2}$$

for π a semi-circle of radius 1.

We see the graph of a perfectly smooth function!

To understand, why the integrand in (1.4.3.109) is smooth, note that $\mathbf{n}(\gamma_\pi(t)) \cdot \dot{\gamma}_\pi(t) = 0$ for all $-1 \leq t \leq 1$, because $\dot{\gamma}_\pi(t)$ is tangential to Γ in the point $\gamma_\pi(t)$. Thus, by power series expansion of the analytic function γ_π for $|s - t|$ sufficiently small

$$\begin{aligned} (\gamma_\pi(s) - \gamma_\pi(t)) \cdot \mathbf{n}(\gamma_\pi(t)) &= \left(\sum_{j=1}^{\infty} \frac{(s-t)^j}{j!} \gamma_\pi^{(j)}(t) \right) \cdot \mathbf{n}(\gamma_\pi(t)) \\ &= (s-t)^2 \cdot \underbrace{\sum_{j=0}^{\infty} \frac{(s-t)^j}{(j+2)!} \gamma_\pi^{(j)} \cdot \mathbf{n}(\gamma_\pi(t))}_{\text{analytic function of } (s,t)}. \end{aligned}$$

By the same arguments

$$\begin{aligned} \|\gamma_\pi(s) - \gamma_\pi(t)\|^2 &= (s-t)^2 \cdot \{ \text{smooth function } > 0 \text{ of } (s,t) \}. \\ \blacktriangleright (s, t) \mapsto \frac{(\gamma_\pi(s) - \gamma_\pi(t)) \cdot \mathbf{n}(\gamma_\pi(t))}{\|\gamma_\pi(s) - \gamma_\pi(t)\|^2} &\text{ is analytic in } (s, t) \quad \mathbf{!} \end{aligned}$$

Hence, the integrand in (1.4.3.109) is analytic and we can achieve

exponential convergence of the quadrature error by standard tensor-product Gaussian (\rightarrow § 1.4.3.78) quadrature of (1.4.3.109).

┘

§1.4.3.110 (Double layer BIO: Abutting panels) We discuss the situation of § 1.4.3.99 for the double layer boundary integral operator K . As earlier, we assume that the panels $\pi, \pi' \in \mathcal{G}$ have in common exactly one point $\overline{\pi} \cap \overline{\pi'} = \{p\}$ and we make use of the arclength parameterization (1.4.3.100).

$$\begin{aligned} \text{[for } \pi\text{]:} \quad \kappa &: [0, |\pi|] \rightarrow \overline{\pi}, \quad \|\dot{\kappa}(t)\| = 1 \quad \forall t \in [0, |\pi|], \\ \text{[for } \pi'\text{]:} \quad \kappa' &: [0, |\pi'|] \rightarrow \overline{\pi'}, \quad \|\dot{\kappa}'(t)\| = 1 \quad \forall t \in [0, |\pi'|]. \end{aligned} \tag{1.4.3.100}$$

We are concerned with the numerical evaluation of integrals in the parameter domain of the form

$$J := \int_0^{|\pi|} \int_0^{|\pi'|} \frac{(\kappa(s) - \kappa'(t)) \cdot \mathbf{n}(\kappa'(t))}{\|\kappa(s) - \kappa'(t)\|^2} F(t)G(s) dt ds. \tag{1.4.3.111}$$

We can no longer count on the regularizing effect of orthogonality as in § 1.4.3.108.



Inspired by the success in § 1.4.3.99, we switch to polar coordinates (r, φ) for the domain in integration: $s = r \cos \varphi, t = r \sin \varphi$.

Then, since $\kappa(0) = \kappa'(0) = \mathbf{p}$, Taylor expansion around $s = t = 0$ yields for $0 \leq s, t$ sufficiently small

$$\kappa(s) - \kappa'(t) = r \cdot \sum_{j=0}^{\infty} \frac{r^j}{(j+1)!} \left(\kappa^{(j+1)}(0) \cos^{j+1} \varphi - \kappa'^{(j+1)}(0) \sin^{j+1} \varphi \right) = r \mathbf{b}(r, \varphi), \quad (1.4.3.112)$$

with a (componentwise) **analytic** function $\mathbf{b} : \mathbb{R}_0^+ \times [0, 2\pi] \rightarrow \mathbb{R}^2$ that satisfies $\mathbf{b}(0, \varphi) \neq \mathbf{0}$ on the domain of integration, compare (1.4.3.105). Thus, in polar coordinates

$$\blacktriangleright \frac{(\kappa(s) - \kappa'(t)) \cdot \mathbf{n}(\kappa'(t))}{\|\kappa(s) - \kappa'(t)\|^2} = \frac{1}{r} \cdot \underbrace{\frac{\mathbf{b}(r, \varphi) \cdot \mathbf{n}(\kappa'(r \sin \varphi))}{\|\mathbf{b}(r, \varphi)\|^2}}_{\text{analytic in } (r, \varphi)}. \quad (1.4.3.113)$$

Thus we can achieve a **cancellation of the singular term** $r \mapsto r^{-1}$ by the metric factor ($dt ds \rightarrow r dr d\varphi$) when integrating in polar coordinates, see Fig. 46 for the meaning of α, β ,

$$\int_0^{|\pi|} \int_0^{|\pi'|} \frac{(\kappa(s) - \kappa'(t)) \cdot \mathbf{n}(\kappa'(t))}{\|\kappa(s) - \kappa'(t)\|^2} F(t)G(s) dt ds = \int_0^\alpha \int_0^{\cos(\varphi)} \frac{\mathbf{b}(r, \varphi) \cdot \mathbf{n}(\kappa'(r \sin \varphi))}{\|\mathbf{b}(r, \varphi)\|^2} dr d\varphi + \int_{\alpha}^{\pi/2} \int_0^{|\pi'|/\sin(\varphi)} \frac{\mathbf{b}(r, \varphi) \cdot \mathbf{n}(\kappa'(r \sin \varphi))}{\|\mathbf{b}(r, \varphi)\|^2} dr d\varphi.$$

For the resulting two integrals

standard tensor-product Gaussian quadrature yields an exponentially convergent numerical approximation.

Remark 1.4.3.114 (Stable evaluation of integrands) The functions $(s, t) \mapsto S(s, t)/D(s, t)$ introduced in (1.4.3.93)/(1.4.3.106) and $(r, \varphi) \mapsto \mathbf{b}(r, \varphi)$ are defined as

$$S(s, t) = \frac{\|\gamma_\pi(s) - \gamma_\pi(t)\|^2}{(s - t)^2} \quad \text{for } s \neq t, \quad (1.4.3.115)$$

$$D(s, t) = \frac{\|\kappa(s) - \kappa'(t)\|^2}{s^2 + t^2} \quad \text{for } s^2 + t^2 > 0, \quad (1.4.3.116)$$

$$\mathbf{b}(r, \varphi) = \frac{\kappa(r \cos \varphi) - \kappa'(r \sin \varphi)}{r} \quad \text{for } r > 0. \quad (1.4.3.117)$$



Evaluating these expressions for $s \approx t, s^2 + t^2 \approx 0$, or $r \approx 0$, respectively, incurs **cancellation**.

As explained in [NumCSE Section 1.5.4], cancellation is a massive amplification of roundoff errors due to subtracting numbers of almost the same value. We have to follow the recommendation of [NumCSE Ex. 1.5.4.26] and

use truncated Taylor expansions of κ, κ' to avoid cancellation !



$$\begin{aligned} \gamma_\pi(s) - \gamma_\pi(t) &\approx (s - t) \dot{\gamma}_\pi\left(\frac{1}{2}(s + t)\right) \quad \text{for } |s - t| < \sqrt{\text{EPS}}, \\ \kappa(s) - \kappa'(t) &\approx \dot{\kappa}(0)s - \dot{\kappa}'(0)t \quad \text{for } r^2 = s^2 + t^2 \leq \text{EPS}. \end{aligned}$$

(EPS $\hat{=}$ machine precision, see [NumCSE Ass. 1.5.3.11])

► Stable evaluation by means of the expressions

$$S(s, t) \approx \left\| \dot{\gamma}_\pi\left(\frac{1}{2}(s+t)\right) \right\|^2 \quad \text{for } |s-t| < \sqrt{\text{EPS}}, \quad (1.4.3.118)$$

$$D(s, t) \approx 1 - \dot{\kappa}(0) \cdot \dot{\kappa}'(0) \frac{2st}{s^2 + t^2} \quad \text{for } s^2 + t^2 \leq \text{EPS}, \quad (1.4.3.119)$$

$$\mathbf{b}(r, \varphi) \approx \dot{\kappa}(0) \cos \varphi - \dot{\kappa}'(0) \sin \varphi \quad \text{for } r < \sqrt{\text{EPS}}. \quad (1.4.3.120)$$

┘

§1.4.3.121 (Treatment of disjoint panels) Now we discuss the situation $\overline{\pi} \cap \overline{\pi'} = \emptyset$. We use the standard local parameterizations of π, π' over $]-1, 1[$ from (1.4.2.25). In principle we face only integrals

$$\int_{-1}^1 \int_{-1}^1 \widehat{k}(s, t) F(t) G(s) dt ds, \quad \widehat{k}(s, t) := k(\gamma_\pi(s), \gamma_{\pi'}(t)), \quad (1.4.3.89)$$

with **analytic** integrands, because the singularity of the fundamental solution is avoided. However, if π and π' are very close,

the proximity of a singularity will be “felt” by Gaussian quadrature and (exponential) convergence (in terms of the number of quadrature points) will deteriorate, see Exp. 1.4.3.73.

Thus we have to link the number of quadrature points to the inverse **relative distance of panels**

$$\rho(\pi, \pi') := \frac{\max\{|\pi|, |\pi'|\}}{\text{dist}(\pi; \pi')}, \quad \text{dist}(\pi; \pi') := \inf\{\|x - y\|, x \in \pi, y \in \pi'\}. \quad (1.4.3.122)$$

The following **heuristic** (supported by the analysis of [SS10, Sect. 5.3.2]) may be implemented:

For (1.4.3.89) use $n \times n$ -point tensor-product Gaussian quadrature on $[-1, 1]^2$ with

$$n = n_0 \cdot \max\left\{1, 1 + C \log\left(\frac{\rho(\pi, \pi')}{\eta}\right)\right\}, \quad (1.4.3.123)$$

where n_0 is a small fixed number, $n_0 \in \{3, 4, 5\}$, and $C, \eta > 0$ are constants, $\frac{1}{2} \leq \eta < 1$, $C \approx ???$.

Thus, in particular,

if $\eta \text{ dist}(\pi; \pi') \geq \max\{|\pi|, |\pi'|\}$ then use fixed $n_0 \times n_0$ -point quadrature .

┘

1.5 Boundary Element Methods on Closed Surfaces

The first-kind and second-kind boundary integral equations stated in variational form in Section 1.3.5.1/Section 1.3.5.2 and Section 1.3.6 hold for both $d = 2, 3$, if based on the respective fundamental solutions. The previous section gave a detailed introduction into the building blocks and algorithmic details of Galerkin boundary element methods in 2D. It is not surprising that for $d = 3$ similar principles, constructions and algorithms will apply. of course, the paradigm of Galerkin discretization elaborated in § 1.4.1.1 remains unchanged.

Also the other ingredients of boundary element methods remain relevant for surfaces, with slight adaptations to the additional dimension:

- ◆ **meshes**, see Section 1.4.2.1 for the 2D case, attain much greater flexibility and will be discussed in Section 1.5.1,
- ◆ **boundary element spaces**, for 2D introduced in Section 1.4.2.2 will again turn out to be “2D finite element spaces on surfaces”, see Section 1.5.2 below,
- ◆ **shape functions**, both global and local will become more complicated than those presented in Section 1.4.2.3, but still comply with the same design pattern, see Section 1.5.2.2,
- ◆ **parametric construction** as presented in § 1.4.2.24 will remain a crucial tool for defining and handling shape functions.
- ◆ panel-oriented **assembly** will exactly agree with its 2D counterpart from Section 1.4.3.1.

Because of the similarities some aspects of boundary element methods in 3D will be treated only briefly with reference to further explanation given in Section 1.4. Also many concepts will be borrowed from Lagrangian finite element methods in 2D, see [NumPDE Section 2.5] and [NumPDE Section 2.6].

1.5.1 Surface Meshes

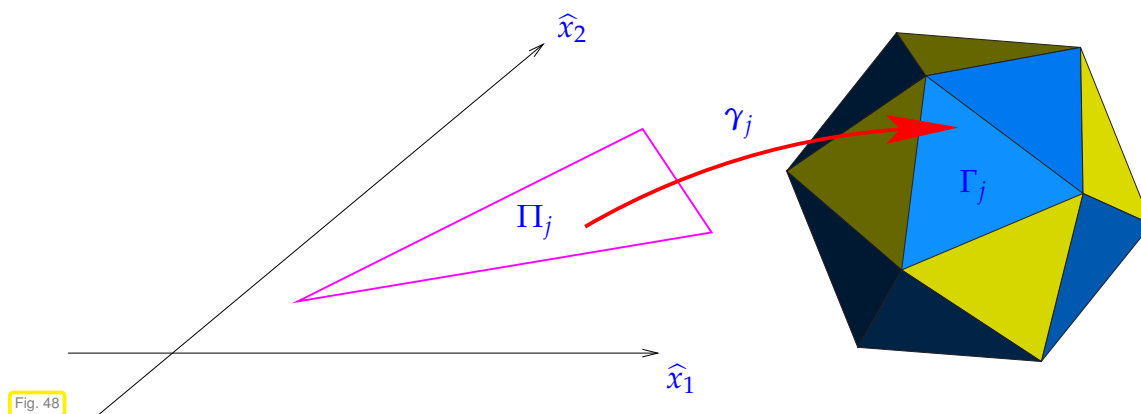
If $\Omega \subset \mathbb{R}^3$, then Γ is an orientable two-dimensional manifold, a surface embedded into three-dimensional Euclidean space \mathbb{R}^3 .

§1.5.1.1 (Γ with smooth faces) Ass. 1.2.1.7 should still apply: Γ is a curved Lipschitz polyhedron and can be partitioned into $M \in \mathbb{N}$ faces

$$\Gamma = \bar{\Gamma}_1 \cup \dots \cup \bar{\Gamma}_M, \quad \Gamma_i \cap \Gamma_j = \emptyset \quad \text{for } i \neq j,$$

of which each has a C^2 -parameterization

$$\gamma_j : \bar{\Pi}_j \rightarrow \bar{\Gamma}_j, \quad \bar{\Pi}_j \subset \mathbb{R}^2 \quad \text{a planar polygon}.$$



§1.5.1.2 (Planar triangulations [NumPDE Section 2.5.1])

Definition 1.5.1.3. Triangular planar mesh/triangulation, cf. [NumPDE Def. 2.5.1.1]

A **triangular mesh/triangulation** \mathcal{M} of a polygon $\Pi \subset \mathbb{R}^2$ is a finite collection $\{K_i\}_{i=1}^N$, $N \in \mathbb{N}$, of *open* non-degenerate **triangles**

- (A) $\bar{\Pi} = \bigcup \{\bar{K}_i, i = 1, \dots, M\}$ (covering property),
- (B) $K_i \cap K_j = \emptyset \Leftrightarrow i \neq j$ (partition property)
- (C) for all $i, j \in \{1, \dots, M\}$, $i \neq j$, the intersection $\bar{K}_i \cap \bar{K}_j$ is either empty or a vertex or edge of both K_i and K_j .

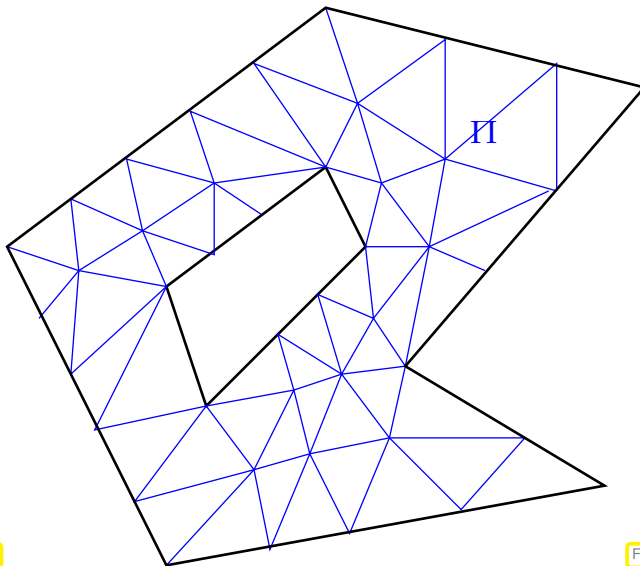


Fig. 49

A triangular mesh/triangulation

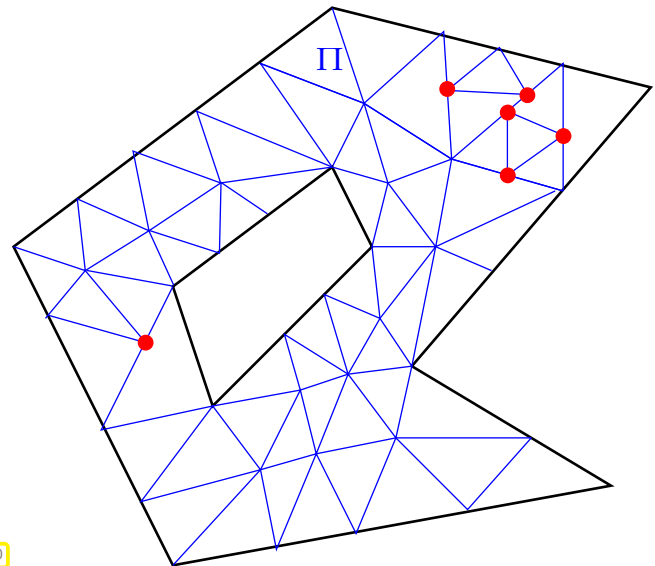


Fig. 50

Inadmissible "hanging nodes"

For the notions of triangles, edges, and vertices as basic constituent parts of a triangulation we appeal to geometric intuition. The triangles of a mesh may also be called **cells**. \lrcorner

Putting it simply, surface mesh is the image of *compatible triangulations* of the parameter domains $\Pi_j \subset \mathbb{R}^2$ under the parameterizations γ_j .

Definition 1.5.1.4. Triangular surface mesh/surface triangulation

A **triangular surface mesh/surface triangulation** \mathcal{G} is a partitioning

$$\Gamma = \overline{\pi}_1 \cup \dots \cup \overline{\pi}_N, \quad \pi_i \cap \pi_j = \emptyset \quad \text{for } i \neq j,$$

such that

- (i) every **panel** π_i is contained in exactly one face,
- (ii) the pre-images of the panels contained in Γ_j under the parameterization γ_j form a triangulation \mathcal{M}_j of Π_j according to Def. 1.5.1.3,
- (iii) for all $\pi_i, \pi_j \in \mathcal{G}$ the intersections $\overline{\pi}_i \cap \overline{\pi}_j$ are either empty, a common vertex, or a face of **both** panels.

As usual, we identify a surface triangulation \mathcal{G} with its set of panels. It should be evident what is meant by edges and vertices of a surface mesh. The vertices may also be called the nodes of the mesh.

\pencil Notation: $\mathcal{V}(\mathcal{G}) \hat{=}$ set of vertices (nodes) of \mathcal{G}
 $\mathcal{E}(\mathcal{G}) \hat{=}$ set of edges of \mathcal{G}

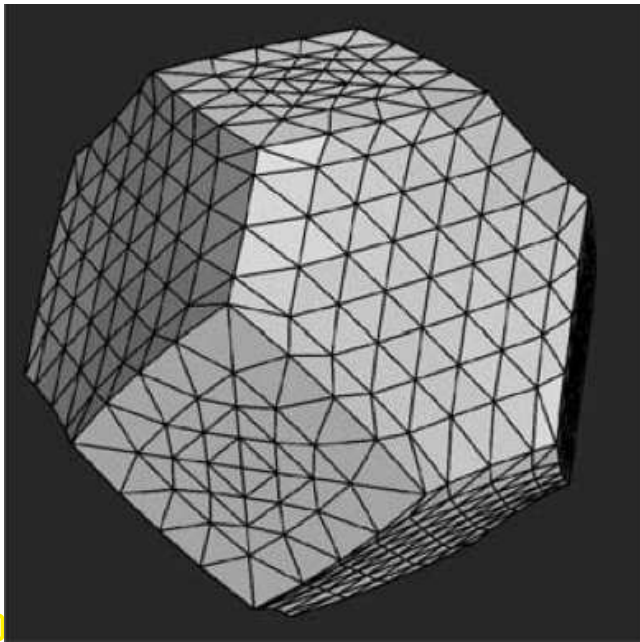


Fig. 51

Item (iii) ensures that the surface triangulation \mathcal{G} is compatible across the edges separating the faces of Γ : also there hanging nodes cannot occur.

◁ Surface triangulation covering a polyhedron

Remark 1.5.1.5 (Surface meshes as traces of volume meshes) We could also have introduced surface meshes as restrictions of tetrahedral finite element meshes of the volume domain Ω to the boundary Γ .

Conversely, we may assume that for every surface mesh \mathcal{G} there is a generalized tetrahedral mesh \mathcal{M} of Ω according to [NumPDE Def. 2.5.1.1], possibly with curved faces and edges, such that $\mathcal{G} = \mathcal{M}|_{\Gamma}$. ▽

Remark 1.5.1.6 (More general surface meshes) Of course, we could have also relied on more general meshes of the parameter domains in our definition of surface meshes, like quadrilateral meshes or hybrid meshes, see [NumPDE § 2.5.1.3]. We restrict ourselves to triangular surface meshes just to simplify the presentation. ▽

1.5.2 Boundary Element Spaces on Triangulated Surfaces

1.5.2.1 Definitions

§1.5.2.1 (Polynomials in \mathbb{R}^2) Polynomials on Γ are again defined via their pullbacks γ^* to parameter domains $\Pi_j \subset \mathbb{C}$. The definition of the pullback Def. 1.4.2.8 carries over and what is a polynomial on Π_j is clear from the following definition for $d = 2$:

Definition 1.5.2.2. Multivariate polynomials

The space of d -variate polynomials of (total) degree $p \in \mathbb{N}_0$ is

$$\mathcal{P}_p(\mathbb{R}^d) := \{x \in \mathbb{R}^d \mapsto \sum_{\alpha \in \mathbb{N}_0^d, |\alpha| \leq p} c_{\alpha} x^{\alpha}, c_{\alpha} \in \mathbb{R}\}.$$

▶
$$d = 2: \mathcal{P}_p(\mathbb{R}^2) = \left\{ \sum_{\substack{\alpha_1, \alpha_2 \geq 0 \\ \alpha_1 + \alpha_2 \leq p}} c_{\alpha_1, \alpha_2} x_1^{\alpha_1} x_2^{\alpha_2}, c_{\alpha_1, \alpha_2} \in \mathbb{R} \right\},$$

for instance $\mathcal{P}_2(\mathbb{R}^2) = \text{Span}\{1, x_1, x_2, x_1^2, x_2^2, x_1 x_2\}$.

From [NumPDE Lemma 2.5.2.5] we learn that

$$\dim \mathcal{P}_p(\mathbb{R}^d) = \binom{d+p}{p} \quad \blacktriangleright \quad \dim \mathcal{P}_p(\mathbb{R}^2) = \frac{1}{2}(p+2)(p+1). \tag{1.5.2.3}$$

┘

§1.5.2.4 (Piecewise polynomials on triangulated surfaces) A function $f : \Gamma_j \rightarrow \mathbb{R}$ is called a polynomial of degree $\leq p$ on the face Γ_j , if its pullback $\gamma_j^* f$ is a 2-variate polynomial of degree $\leq p$ on $\Pi_j \subset \mathbb{R}^2$. Thus, the definitions of piecewise polynomial spaces for Section 1.4.2.2 remain unchanged.

$$\mathcal{S}_p^0(\mathcal{G}) := \left\{ v \in C^0(\Gamma) : \gamma_j^*(v|_\pi) \in \mathcal{P}_p(\mathbb{R}^2), \forall \pi \in \mathcal{G}, \pi \subset \Gamma_j, j = 1, \dots, M \right\}, \quad p \geq 1, \quad (1.5.2.5)$$

$$\mathcal{S}_p^{-1}(\mathcal{G}) := \left\{ v \in L^2(\Gamma) : \gamma_j^*(v|_\pi) \in \mathcal{P}_p(\mathbb{R}^2), \forall \pi \in \mathcal{G}, \pi \subset \Gamma_j, j = 1, \dots, M \right\}, \quad p \geq 0. \quad (1.5.2.6)$$

The embeddings $\mathcal{S}_p^0(\mathcal{G}) \subset C_{\text{pw}}^1(\Gamma) \subset H^{\frac{1}{2}}(\Gamma)$, $\mathcal{S}_p^{-1}(\mathcal{G}) \subset C_{\text{pw}}^0(\Gamma) \subset L^2(\Gamma) \subset H^{-\frac{1}{2}}(\Gamma)$, stated in Cor. 1.4.2.12 remain true. ┘

Theorem 1.5.2.7. Dimensions of BE spaces on triangulated surfaces

- ◆ $\dim \mathcal{S}_p^0(\mathcal{G}) = \#\mathcal{V}(\mathcal{G}) + (p-1) \cdot \#\mathcal{E}(\mathcal{G}) + \frac{1}{2}(p-1)(p-2) \cdot \#\mathcal{G}, \quad p \geq 1,$
 - ◆ $\dim \mathcal{S}_p^{-1}(\mathcal{G}) = \#\mathcal{G} \frac{1}{2}(p+1)(p+2), \quad p \geq 0.$
- (negative terms to be set to zero!)

BE spaces from FE spaces

Let \mathcal{M}_j be the triangular mesh of Π_j inducing $\mathcal{G}|_{\Gamma_j}$. Then

$$\begin{aligned} \mathcal{S}_p^0(\mathcal{M}_j) &= \gamma_j^* \mathcal{S}_p^0(\mathcal{G})|_{\Gamma_j}, \\ \mathcal{S}_p^{-1}(\mathcal{M}_j) &= \gamma_j^* \mathcal{S}_p^{-1}(\mathcal{G})|_{\Gamma_j}, \end{aligned} \quad j = 1, \dots, M, \quad (1.5.2.9)$$

where $\mathcal{S}_p^0(\mathcal{M}_j)$ is the p -th degree Lagrangian finite element space on \mathcal{M}_j as defined in [NumPDE Def. 2.6.1.1], and $\mathcal{S}_p^{-1}(\mathcal{M})$ the space of \mathcal{M}_j -piecewise polynomials of degree $\leq p$.

The relationship (1.5.2.9) permits us to *transfer most concepts from finite element spaces in 2D to surface boundary element spaces*. In particular, this will be done in the next section.

Remark 1.5.2.10 (Nodal interpolation operators) The relationship expressed in (1.5.2.9) permits us to transfer most tools from the world of finite elements to boundary elements.

Let \mathcal{M}_j be the preimage of $\mathcal{G}|_{\Gamma_j}$ under γ_j . For the Lagrangian finite element space $\mathcal{S}_p^0(\mathcal{M}_j)$ there are **nodal interpolation operators** $l_p^{0,j} : C^0(\overline{\Pi}_j) \rightarrow \mathcal{S}_p^0(\mathcal{M}_j)$ defined through interpolation in special interpolation points. Their locations for different p in 2D are described in [NumPDE Ex. 2.6.1.2] and [NumPDE ??]. Then nodal interpolation operators $l_p^0 : C^0(\Gamma) \rightarrow \mathcal{S}_p^0(\mathcal{G})$ can be defined by

$$l_p^0|_{\Gamma_j} := (\gamma_j^{-1})^* \circ l_p^{0,j} \circ \gamma_j^*. \quad (1.5.2.11)$$

This amounts to “piecewise polynomial interpolation in the mapped interpolation nodes”. For $p = 1$ the interpolation nodes coincide with the vertices of \mathcal{G} . ┘

Remark 1.5.2.12 (Approximation of surfaces) 3D boundary element codes often resort to **piecewise polynomial** approximation of Γ , analogous to what was done for curves in Section 1.4.2.5.

Given a triangular surface mesh according to Def. 1.5.1.4, we define approximate piecewise polynomial parameterizations by

$$\tilde{\gamma}_j := l_p^{0,j} \circ \gamma_j : \Pi_j \rightarrow \mathbb{R}^3, \tag{1.5.2.13}$$

where the nodal interpolation operator $l_p^{0,j}$ acts on the three components of γ_j . Then we obtain the approximate surface

$$\tilde{\Gamma} = \tilde{\Gamma}_1 \cup \dots \cup \tilde{\Gamma}_M, \quad \tilde{\Gamma}_j := \gamma_j(\Pi_j).$$

┘

1.5.2.2 Shape Functions

§1.5.2.14 (Global shape functions) Everything from Section 1.4.2.3 can be adapted to triangulated surfaces and the associated boundary element spaces $\mathcal{S}_p^0(\mathcal{G})$ of continuous, and $\mathcal{S}_p^{-1}(\mathcal{G})$ of discontinuous piecewise polynomials. Again, we can find bases of the boundary element spaces consisting of **locally supported basis functions** associated with geometric entities of the surface mesh \mathcal{G} ; they satisfy the properties § 1.4.2.15, § 1.4.2.15, and § 1.4.2.15 from 89 and are called **global shape functions (GSF)**. From [NumPDE Ex. 2.5.3.2] we recall

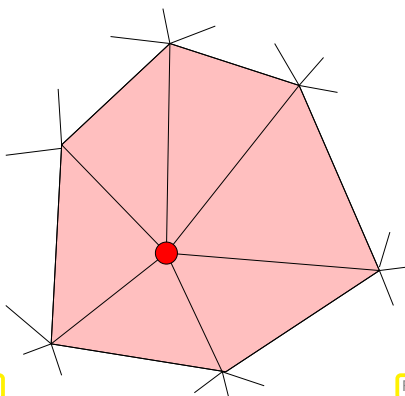


Fig. 52

Support of vertex-associated basis function

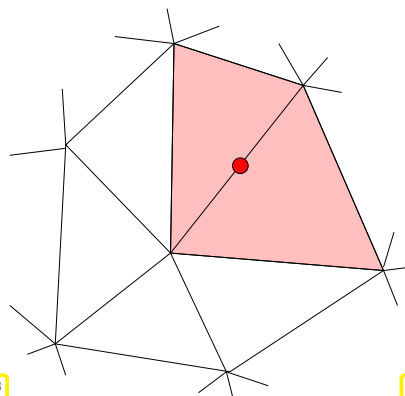


Fig. 53

Support of edge-associated basis function

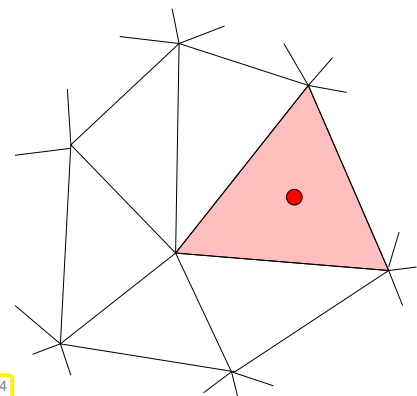


Fig. 54

Support of panel-associated basis function

§1.5.2.15 (Local shape functions) Restricting global shape functions to individual panels we obtain **local shape functions (LSF)**:

$$\{b_\pi^1, \dots, b_\pi^Q\} = \{b_N|_\pi : b_N \in \mathfrak{B}_N\} \setminus \{0\} \quad \text{for some } Q = Q(\pi) \in \mathbb{N}. \tag{1.4.2.22}$$

Also (1.4.2.23) remains true: If $\{b_\pi^1, \dots, b_\pi^Q\}$ is the set of local shape functions of $\mathcal{S}_p^0(\mathcal{G})$ or $\mathcal{S}_p^{-1}(\mathcal{G})$ for a panel $\pi \subset \Gamma_j$ then

$$\forall \pi \in \mathcal{G}, \pi \subset \Gamma_j: \gamma_j^*(\text{Span}\{b_\pi^1, \dots, b_\pi^Q\}) = \mathcal{P}_p(\mathbb{R}^2). \tag{1.4.2.23}$$

┘

§1.5.2.16 (Reference shape functions) The role of the reference interval $\hat{I} :=]-1, 1[$ in 2D is now played by the “unit triangle” $\hat{K} := \langle [0, 1], [0, 0], [1, 1] \rangle$, see [NumPDE ??].

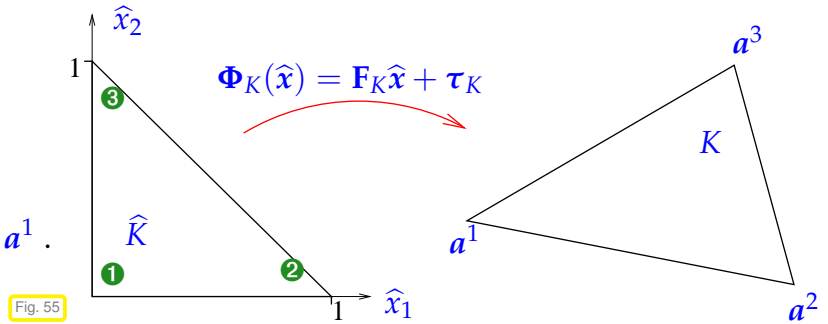
For a panel $\pi \subset \Gamma_j$ the **local parameterization** γ_π is built by a two-stage procedure:

- 1 Find the unique affine mapping from \hat{K} to the triangle $K := \gamma_j^{-1}(\pi)$:

“Unit triangle”: $\hat{K} = \langle [0, 0], [1, 0], [0, 1] \rangle$

For $K = \langle a^1, a^2, a^3 \rangle$:

$$F_K = \begin{bmatrix} a_1^2 - a_1^1 & a_1^3 - a_1^1 \\ a_2^2 - a_2^1 & a_2^3 - a_2^1 \end{bmatrix}, \quad \tau_K = a^1.$$



- 2 Map from K to π through the parameterization $\gamma_j|_K$.

These two mappings can be concatenated into a local parameterization of the panel π :

$$[\hat{K} \xrightarrow{\Phi_K} K \xrightarrow{\gamma_j} \pi] , \quad \gamma_\pi := \gamma_j \circ \Phi_K : \hat{K} \rightarrow \pi . \tag{1.5.2.17}$$

The pullback of shape functions to \hat{K} yields **reference shape functions**:

$$\hat{b}^j = \gamma_\pi^*(b_\pi^j), \quad j = 1, \dots, Q . \tag{1.4.2.27}$$

For the standard boundary element spaces $S_p^0(\mathcal{G})$ and $S_p^{-1}(\mathcal{G})$ on a triangulated surface the reference shape functions can be chosen independent of the panel π :

- For $S_p^{-1}(\mathcal{G})$, $p \geq 0$:

Any basis of $\mathcal{P}_p(\mathbb{R}^2)$ can supply valid reference shape functions.

- For $S_p^0(\mathcal{G})$, $p \geq 1$:

Reference shape functions $\in \mathcal{P}_p(\mathbb{R}^2)$ as Lagrange polynomials for suitable **interpolation nodes** on \hat{K} , see [NumPDE Ex. 2.6.1.2] and [NumPDE Ex. 2.6.1.7].

In both cases $Q = \dim \mathcal{P}_p(\mathbb{R}^2) = \frac{1}{2}(p+1)(p+2)$. ┘

The reference shape functions in the lowest-degree cases are straightforward:

EXAMPLE 1.5.2.18 (Reference shape functions for $S_0^{-1}(\mathcal{G})$) The space $\mathcal{P}_0(\mathbb{R}^2)$ spanned by the reference shape functions has dimension 1 and, therefore, for $S_0^{-1}(\mathcal{G})$

$$\hat{\beta}^1 \equiv 1 \quad \text{on } \hat{K} .$$
┘

EXAMPLE 1.5.2.19 (Reference shape functions for $S_1^0(\mathcal{G})$) The reference shape functions space the space $\mathcal{P}_1(\mathbb{R}^2)$ of dimension 3. The reference shape functions are the **barycentric coordinate functions** $\lambda_1, \lambda_2, \lambda_3$ on \hat{K}

$$\begin{aligned} \hat{b}^1(\mathbf{t}) = \lambda_1(\mathbf{t}) &:= 1 - t_1 - t_2 && \text{[associated with vertex } \begin{bmatrix} 0 \\ 0 \end{bmatrix} \text{]}, \\ \hat{b}^2(\mathbf{t}) = \lambda_2(\mathbf{t}) &:= t_1 && \text{[associated with vertex } \begin{bmatrix} 1 \\ 0 \end{bmatrix} \text{]}, \\ \hat{b}^3(\mathbf{t}) = \lambda_3(\mathbf{t}) &:= t_2 && \text{[associated with vertex } \begin{bmatrix} 0 \\ 1 \end{bmatrix} \text{]}, . \end{aligned}$$
┘

1.5.3 Assembly of Galerkin Matrices

The entire discussion in Section 1.4.3.1 including Code 1.4.3.14 carries over to surface boundary elements. Therefore we completely focus on the computation of entries of the interaction matrices of two panels $\pi, \pi' \in \mathcal{G}$ by means of **quadrature-based techniques**, that is, we present the subject of Section 1.4.3.4 for $d = 3$. As in Section 1.4.3.4 we assume maximally smooth parameterizations, compare Ass. 1.4.3.85.

Assumption 1.5.3.1. Analyticity of local parameterizations

We assume that the local parameterizations γ_π according to (1.5.2.17) can be extended analytically (\rightarrow Def. 1.4.3.68) to an ellipse neighborhood of $[0, 1]$ in both variables and independently of the panel $\pi \in \mathcal{G}$.

Also due to the different nature of singularities in the fundamental solutions

$$G^\Delta(\mathbf{x}, \mathbf{y}) = \begin{cases} -\frac{1}{2\pi} \log \|\mathbf{x} - \mathbf{y}\| & , \text{ if } d = 2, \\ \frac{1}{4\pi} \frac{1}{\|\mathbf{x} - \mathbf{y}\|} & , \text{ if } d = 3, \end{cases} \quad (1.2.2.33)$$

the technical details of the computations will be very different for the different dimensions. In this section we exclusively focus on the **single layer BIO V**, that is, the evaluation of integrals of the form

$$I := \int_\pi \int_{\pi'} \frac{1}{\|\mathbf{x} - \mathbf{y}\|} b_{\pi'}^j(\mathbf{y}) b_\pi^i(\mathbf{x}) dS(\mathbf{y}) dS(\mathbf{x}),$$

for pairs of panels $\pi, \pi' \in \mathcal{G}$, where b_π^i are local shape functions, see § 1.5.2.15.

§1.5.3.2 (Transformation to reference triangle, cf. § 1.4.3.87) By pullback to the reference triangle $\widehat{K} := \left\langle \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right\rangle$ we obtain

$$I = \int_{\widehat{K}} \int_{\widehat{K}} \frac{1}{\|\gamma_\pi(\mathbf{s}) - \gamma_{\pi'}(\mathbf{t})\|} F(\mathbf{t}) G(\mathbf{s}) dt ds, \quad (1.5.3.3)$$

with smooth functions $F, G \in C^\infty(\widehat{K})$ that possess an analytic extension beyond \widehat{K} in each variable. The domain of integration in (1.5.3.3) is **four-dimensional**, a tensor-product of two triangles, the convex polyhedron.

$$\widehat{K} \times \widehat{K} = \left\{ [s_1, s_2, t_1, t_2]^\top \in \mathbb{R}^4 : t_1, t_2, s_1, s_2 > 0, t_1 + t_2 < 1, s_1 + s_2 < 1 \right\}.$$

┘

§1.5.3.4 (Coinciding panels, compare § 1.4.3.90, [SS10, Sect. 5.2.1]) We deal with the situation $\pi = \pi', \gamma_\pi = \gamma_{\pi'} =: \gamma : \widehat{K} \rightarrow \pi$.



We observe that in (1.5.3.3) the integrand has a singularity for $\mathbf{s} = \mathbf{t}$, which suggests the following change of coordinates [SS10, Sect. 5.2.1].

$$\begin{bmatrix} \widehat{\mathbf{s}} \\ \widehat{\mathbf{z}} \end{bmatrix} = \begin{bmatrix} \mathbf{s} \\ \mathbf{s} - \mathbf{t} \end{bmatrix} \Leftrightarrow \begin{bmatrix} \mathbf{s} \\ \mathbf{t} \end{bmatrix} = \begin{bmatrix} \widehat{\mathbf{s}} \\ \widehat{\mathbf{s}} - \widehat{\mathbf{z}} \end{bmatrix}. \quad (1.5.3.5)$$

This is a volume preserving ($\det = 1$) linear transformation and it converts (1.5.3.3) into

$$I = \int_D \int_D \frac{1}{\|\gamma(\widehat{\mathbf{s}}) - \gamma(\widehat{\mathbf{s}} - \widehat{\mathbf{z}})\|} \underbrace{F(\widehat{\mathbf{s}} - \widehat{\mathbf{z}}) G(\widehat{\mathbf{s}})}_{\text{analytic in } (\widehat{\mathbf{s}}, \widehat{\mathbf{z}})} d\widehat{\mathbf{z}} d\widehat{\mathbf{s}}, \quad (1.5.3.6)$$

where $D \subset \mathbb{R}^4$ is the transformed convex polyhedron

$$D = \left\{ [\hat{s}_1, \hat{s}_2, \hat{z}_1, \hat{z}_2]^\top \in \mathbb{R}^4 : \begin{array}{l} \hat{s}_1, \hat{s}_2 > 0, \hat{s}_1 - \hat{z}_1 > 0, \hat{s}_2 - \hat{z}_2 > 0, \\ \hat{s}_1 + \hat{s}_2 - (\hat{z}_1 + \hat{z}_2) < 1 \end{array} \right\}. \quad (1.5.3.7)$$

Now the singularity has been isolated at $\hat{z} = \mathbf{0}$, where the integrand behaves like $O(\|\hat{z}\|^1)$ for $\hat{z} \rightarrow \mathbf{0}$.



As in § 1.4.3.110 for the treatment of $O(\|\hat{z}\|^{-1})$ -type singularities switch to **polar coordinates**: $\hat{z}_1 = r \cos \varphi, \hat{z}_2 = r \sin \varphi, r \geq 0$.

To understand the behavior of the integrand we perform two-dimensional Taylor expansion around $\hat{z} = \mathbf{0}$:

$$\gamma(\hat{s}) - \gamma(\hat{s} - \hat{z}) = -D\gamma(\hat{s})\hat{z} + \sum_{k=1}^{\infty} \frac{1}{k!} \sum_{\ell_1 + \ell_2 = k} D^{\ell} \gamma(\hat{s}) (-\hat{z})^{\ell}.$$

Then we plug in polar coordinates and get

$$B(\hat{s}, \hat{z}) := \frac{\|\gamma(\hat{s}) - \gamma(\hat{s} - \hat{z})\|^2}{\|\hat{z}\|^2} = \begin{bmatrix} \cos \varphi \\ \sin \varphi \end{bmatrix}^\top D\gamma(\hat{s})^\top D\gamma(\hat{s}) \begin{bmatrix} \cos \varphi \\ \sin \varphi \end{bmatrix} + r \cdot \{\text{analytic in } (\hat{s}, r, \varphi)\}.$$

Since γ is a parameterization, the smallest eigenvalue of the Gram matrix $D\gamma^\top D\gamma$ must be uniformly positive on \bar{K} . Therefore, for sufficiently small $r := \|\hat{z}\|$, $B(\hat{s}, \hat{z})$ will be positive, and

$$(\hat{s}, \hat{z}) \in D \mapsto \sqrt{B(\hat{s}, \hat{z})} \in \mathbb{R}^+$$

will possess an analytic extension beyond D . Hence, we have

$$I = \int \int_D \frac{1}{\|\hat{z}\|} \underbrace{\frac{F(\hat{s} - \hat{z}) G(\hat{s})}{\sqrt{B(\hat{s}, \hat{z})}}}_{\text{analytic in } \bar{D}} d\hat{z} d\hat{s} = \int \int_D \frac{F(\hat{s} - \hat{z}) G(\hat{s})}{\sqrt{B(\hat{s}, \hat{z})}} d\hat{s} r d\varphi, \quad (1.5.3.8)$$

because the volume element $d\hat{z} d\hat{s} = r dr d\varphi$ cancels the denominator $r = \|\hat{z}\|$. We have achieved an integral with an **analytic integrand**, on a complicated domain, however.

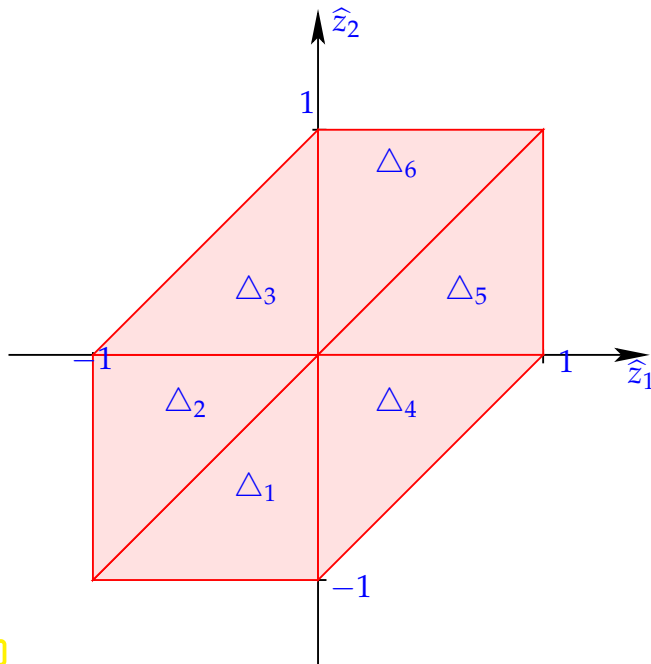


Split D into six four-dimensional simplices with a vertex in $\mathbf{0}$ [SS10, p. 309]!

$$\begin{aligned} D &= \{ -1 < \hat{z}_1 < 0, -1 < \hat{z}_2 < \hat{z}_1, -\hat{z}_2 < \hat{s}_1 < 1, -\hat{z}_2 < \hat{s}_2 < \hat{s}_1 \} && \cup \\ &\{ -1 < \hat{z}_1 < 0, \hat{z}_1 < \hat{z}_2 < 0, \hat{z}_1 < \hat{s}_1 < 1, -\hat{z}_2 < \hat{s}_2 < \hat{s}_1 + \hat{z}_1 - \hat{z}_2 \} && \cup \\ &\{ -1 < \hat{z}_1 < 0, 0 < \hat{z}_2 < 1 + \hat{z}_1, \hat{z}_2 - \hat{z}_1 < \hat{s}_1 < 1, 0 < \hat{s}_2 < \hat{s}_1 + \hat{z}_1 - \hat{z}_2 \} && \cup \\ &\{ 0 < \hat{z}_1 < 1, -1 + \hat{z}_1 < \hat{z}_2 < 0, -\hat{z}_2 < \hat{s}_1 < 1 - \hat{z}_1, -\hat{z}_2 < \hat{s}_2 < \hat{s}_1 \} && \cup \\ &\{ 0 < \hat{z}_1 < 1, 0 < \hat{z}_2 < \hat{z}_1, 0 < \hat{s}_1 < 1 - \hat{z}_1, 0 < \hat{s}_2 < \hat{s}_1 \} && \cup \\ &\{ 0 < \hat{z}_1 < 1, \hat{z}_1 < \hat{z}_2 < 1, \hat{z}_2 - \hat{z}_1 < \hat{s}_2 < 1 - \hat{z}_1, 0, \hat{s}_2 < \hat{z}_1 - \hat{z}_2 + \hat{s}_1 \} && \cup \\ &=: D_1 \cup D_2 \cup D_3 \cup D_4 \cup D_5 \cup D_6. \end{aligned} \quad (1.5.3.9)$$

We have arranged the inequalities defining the sets D_i in a way that removes the dependence of the \hat{z} -coordinate from \hat{s} . So we can rewrite

$$\begin{aligned}
 D = \{ & \hat{z} \in \Delta_1, -\hat{z}_2 < \hat{s}_1 < 1, -\hat{z}_2 < \hat{s}_2 < \hat{s}_1 \} & \cup \\
 & \{ \hat{z} \in \Delta_2, \hat{z}_1 < \hat{s}_1 < 1, -\hat{z}_2 < \hat{s}_2 < \hat{s}_1 + \hat{z}_1 - \hat{z}_2 \} & \cup \\
 & \{ \hat{z} \in \Delta_3, \hat{z}_2 - \hat{z}_1 < \hat{s}_1 < 1, 0 < \hat{s}_2 < \hat{s}_1 + \hat{z}_1 - \hat{z}_2 \} & \cup \\
 & \{ \hat{z} \in \Delta_4, -\hat{z}_2 < \hat{s}_1 < 1 - \hat{z}_1, -\hat{z}_2 < \hat{s}_2 < \hat{s}_1 \} & \cup \\
 & \{ \hat{z} \in \Delta_5, 0 < \hat{s}_1 < 1 - \hat{z}_1, 0 < \hat{s}_2 < \hat{s}_1 \} & \cup \\
 & \{ \hat{z} \in \Delta_6, \hat{z}_2 - \hat{z}_1 < \hat{s}_2 < 1 - \hat{z}_1, 0, \hat{s}_2 < \hat{z}_1 - \hat{z}_2 + \hat{s}_1 \} , &
 \end{aligned}
 \tag{1.5.3.10}$$



◁ with suitably defined triangles Δ_i in the $\hat{z}_1 - \hat{z}_2$ -plane.

Refer to Fig. 46 for the representation of triangles in polar coordinates.

Fig. 56

Thus we can express the integral through contributions from simpler domains:

$$\begin{aligned}
 \iint_D \dots d\hat{s} dr d\varphi = & \int_{\Delta_1} \int_{-\hat{z}_2}^1 \int_{-\hat{z}_2}^{\hat{s}_1} \dots d\hat{s}_2 d\hat{s}_1 dr d\varphi + \int_{\Delta_2} \int_{-\hat{z}_1}^1 \int_{-\hat{z}_2}^{\hat{s}_1 + \hat{z}_1 - \hat{z}_2} \dots d\hat{s}_2 d\hat{s}_1 dr d\varphi + \\
 & \int_{\Delta_3} \int_{\hat{z}_2 - \hat{z}_1}^1 \int_0^{\hat{s}_1 + \hat{z}_1 - \hat{z}_2} \dots d\hat{s}_2 d\hat{s}_1 dr d\varphi + \int_{\Delta_4} \int_{-\hat{z}_2}^{1 - \hat{z}_1} \int_{-\hat{z}_2}^{\hat{s}_1} \dots d\hat{s}_2 d\hat{s}_1 dr d\varphi \\
 & \int_{\Delta_5} \int_0^{1 - \hat{z}_1} \int_0^{\hat{s}_1} \dots d\hat{s}_2 d\hat{s}_1 dr d\varphi + \int_{\Delta_6} \int_{\hat{z}_2 - \hat{z}_1}^{1 - \hat{z}_1} \int_0^{\hat{z}_1 - \hat{z}_2 + \hat{s}_1} \dots d\hat{s}_2 d\hat{s}_1 dr d\varphi .
 \end{aligned}
 \tag{1.5.3.11}$$

For the triangles it is easy to determine the corresponding integration bounds in the (r, φ) -domain: make the radius dependent of the angle as we did in § 1.4.3.99.

Four-nested Gauss(-Legendre) quadrature rules applied to every integral in (1.5.3.11) yield an exponentially convergent quadrature approximation.



Remember Rem. 1.4.3.114 and be wary of cancellation that may affect the evaluation of $B(\hat{s}, \hat{z})$.

┘

Remark 1.5.3.12 (Precomputing complex quadrature formula) The domain of integration in (1.5.3.8) does not depend on π , only the smooth integrand does. Hence, for a fixed order of the four-nested Gauss-Legendre rule used to evaluate the integrals in (1.5.3.11), all points at which we have to evaluate the integrand are known in advance and will be independent of π . Thus we can simply **precompute** the resulting family of complex quadrature formula on D (in polar coordinates) and tabulate them. \lrcorner

§1.5.3.13 (Adjacent panels [SS10, Sect. 5.2.2], cf. § 1.4.3.99) We face the situation $\pi \neq \pi'$, $\overline{\pi} \cap \overline{\pi'} = E$, E an edge of \mathcal{G} . Given local parameterizations $\gamma := \gamma_\pi : \widehat{K} \rightarrow \pi$ and $\gamma' := \gamma_{\pi'} : \widehat{K} \rightarrow \pi'$ we assume that they agree for E :

$$\bar{E} = \gamma([0,1] \times \{0\}) = \gamma'([0,1] \times \{0\}) \quad , \quad \gamma\left(\begin{bmatrix} t \\ 0 \end{bmatrix}\right) = \gamma'\left(\begin{bmatrix} t \\ 0 \end{bmatrix}\right) \quad , \quad 0 \leq t \leq 1. \quad (1.5.3.14)$$

Thus, the integrand in the transformed integral

$$I = \int_{\widehat{K}} \int_{\widehat{K}} \frac{1}{\|\gamma_\pi(s) - \gamma_{\pi'}(t)\|} F(t) G(s) dt ds \quad , \quad (1.5.3.3)$$

has a **singularity** for $t_1 = s_1$!

To deal with the singularity at $t_1 = s_1$ we employ the following **change of integration variables** [SS10, p. 313]



$$\begin{bmatrix} \widehat{s}_1 \\ \widehat{z} \\ \widehat{s}_2 \\ \widehat{t}_2 \end{bmatrix} = \begin{bmatrix} s_1 \\ s_1 - t_1 \\ s_2 \\ t_2 \end{bmatrix} \Leftrightarrow \begin{bmatrix} s_1 \\ s_2 \\ t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} \widehat{s}_1 \\ \widehat{s}_2 \\ \widehat{s}_1 - \widehat{z} \\ \widehat{t}_2 \end{bmatrix} \quad . \quad (1.5.3.15)$$

This yields the transformed integral over the pre-image D of $\widehat{K} \times \widehat{K}$ under this transformation:

$$I = \int_D \int_D \frac{1}{\|\gamma(\begin{bmatrix} \widehat{s}_1 \\ \widehat{s}_2 \end{bmatrix}) - \gamma'(\begin{bmatrix} \widehat{s}_1 - \widehat{z} \\ \widehat{t}_2 \end{bmatrix})\|} F\left(\begin{bmatrix} \widehat{s}_1 - \widehat{z} \\ \widehat{t}_2 \end{bmatrix}\right) G\left(\begin{bmatrix} \widehat{s}_1 \\ \widehat{s}_2 \end{bmatrix}\right) d\widehat{s}_1 d\widehat{z} d\widehat{s}_2 d\widehat{t}_2 \quad , \quad (1.5.3.16)$$

with the four-dimensional convex polyhedron

$$D := \left\{ \begin{bmatrix} \widehat{s}_1 \\ \widehat{z} \\ \widehat{s}_2 \\ \widehat{t}_2 \end{bmatrix} \in \mathbb{R}^4 : \begin{array}{l} 0 < \widehat{s}_1 < 1, \widehat{s}_1 - 1 < \widehat{z} < \widehat{s}_1, \\ 0 < \widehat{s}_2 < 1 - \widehat{s}_1, 0 < \widehat{t}_2 < 1 - \widehat{s}_1 + \widehat{z} \end{array} \right\} \quad . \quad (1.5.3.17)$$

To motivate the next transformation, we temporarily focus on the case that both π and π' are *flat triangles*:

$$\pi = \langle a, b, c \rangle \quad , \quad \pi' = \langle a, b, c' \rangle \quad , \quad a, b, c, c' \in \mathbb{R}^3 \quad ,$$

that is $E = [a, b]$.

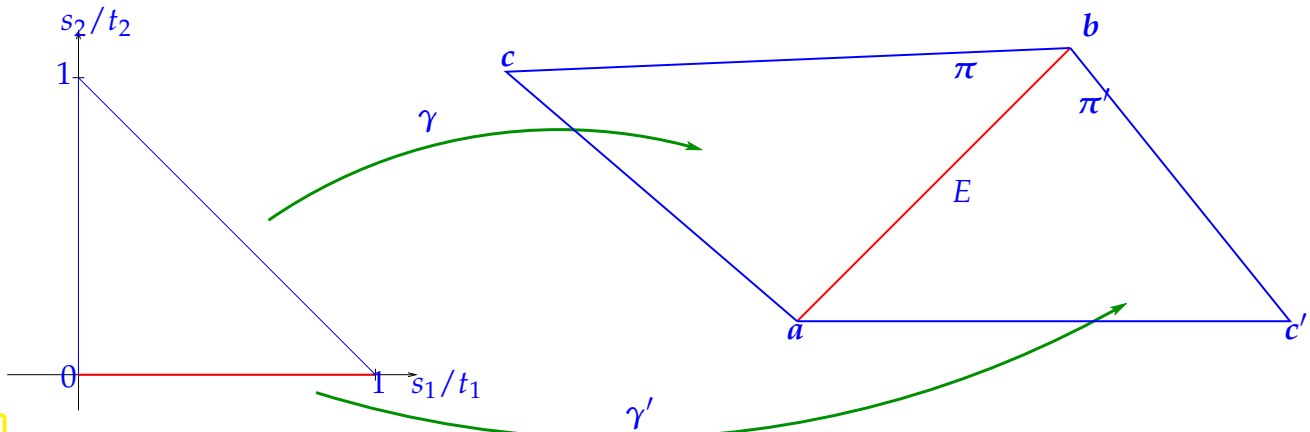


Fig. 57

In this special case the local parameterizations can be chosen as, see Fig. 57:

$$\gamma(s) = a + s_1 u + s_2 v, \quad \gamma'(t) = a + t_1 u + t_2 v', \quad \begin{aligned} u &:= b - a, \\ v &:= c - a, \\ v' &:= c' - a. \end{aligned} \tag{1.5.3.18}$$

We point out the geometric minimal angle conditions for triangles of the mesh and at edges of Γ : with α_0 independent of π, π'

$$\angle(v, v'), \angle(u, v), \angle(u, v') > \alpha_0. \tag{1.5.3.19}$$

Then we find

$$\begin{aligned} \|\gamma(s) - \gamma'(t)\|^2 &= \|\hat{s}_1 u + \hat{s}_2 v - (\hat{s}_1 - \hat{z})u + \hat{t}_2 v'\|^2 \\ &= \|\hat{s}_2 v - \hat{z}u + \hat{t}_2 v'\|^2 \\ &= \hat{s}_2^2 \|v\|^2 + \hat{z}^2 \|u\|^2 + \hat{t}_2^2 \|v'\|^2 - 2\hat{s}_2 \hat{z} (u \cdot v) - 2\hat{t}_2 \hat{z} (u \cdot v') + 2\hat{s}_2 \hat{t}_2 (v \cdot v'). \end{aligned}$$

This suggests that we use **spherical coordinates** (r, θ, φ) in $\hat{z} - \hat{s}_2 - \hat{t}_2$ -space:



$$\hat{z} = r \sin \theta \cos \varphi, \quad \hat{s}_2 = r \sin \theta \sin \varphi, \quad \hat{t}_2 = r \cos \theta, \tag{1.5.3.20}$$

$$r \geq 0, \quad 0 \leq \varphi < 2\pi, \quad 0 < \theta < \pi, \text{ for which the volume element is } d\hat{z}d\hat{s}_2d\hat{t} = r^2 \sin \theta d\theta d\varphi dr.$$

In these new coordinates we obviously have for flat panels

$$\|\gamma(s) - \gamma'(t)\|^2 = r^2 \cdot p(\theta, \varphi),$$

where p is a polynomial in $\sin \theta, \sin \varphi, \cos \theta, \cos \varphi$, **uniformly positive** in $[0, \pi] \times [0, 2\pi]$ due to the angle condition (1.5.3.19).

In the general case Taylor expansion arguments confirm that for small $r \geq 0$

$$(\hat{s}_1, r, \theta, \varphi) \mapsto \frac{r}{\left\| \gamma \left(\begin{bmatrix} \hat{s}_1 \\ r \sin \theta \sin \varphi \end{bmatrix} \right) - \gamma' \left(\begin{bmatrix} \hat{s}_1 - r \sin \theta \cos \varphi \\ r \cos \theta \end{bmatrix} \right) \right\|}$$

is **analytic** on the pre-image \bar{D}_o of \bar{D} under the spherical coordinate transformation. To write I as nested integrals $\int \int \int \int \dots d\hat{s}_1 dr d\theta d\varphi$, in analogy to (1.5.3.9), we split D into five simplices with a single vertex in $\mathbf{0}$ each. For details refer to [SS10, pp. 313].

Four-nested Gauss(-Legendre) quadrature applied (pieces of) to the $(\hat{s}_1, r, \theta, \varphi)$ -transformed integral I converges exponentially in the number of quadrature nodes.

Pre-computation of the corresponding complex quadrature rule is possible, of course. ┘

§1.5.3.21 (Common vertex [SS10, Sect. 5.2.3]) To deal with the case $\bar{\pi} \cap \bar{\pi}' = \{p\}$, $p \in \mathbb{R}^3$ a point, we assume local parameterizations $\gamma := \gamma_\pi : \hat{K} \rightarrow \pi$ and $\gamma' := \gamma_{\pi'} : \hat{K} \rightarrow \pi'$ that satisfy $\gamma(\mathbf{0}) = \gamma'(\mathbf{0}) = p$. Thus the integrand of the transformed integral

$$I = \int_{\hat{K}} \int_{\hat{K}} \frac{1}{\|\gamma_\pi(s) - \gamma_{\pi'}(t)\|} F(t) G(s) dt ds, \tag{1.5.3.3}$$

has a singularity in $s = t = \mathbf{0}$ only. This can be removed by switching to **four-dimensional spherical coordinates**. The arguments are similar to those elaborated in § 1.5.3.13. ┘

§1.5.3.22 (Panels at a positive distance) We follow heuristic rules put forth in § 1.4.3.121 and use Gauss quadrature formulas on $\hat{K} \times \hat{K}$ with orders adjusted to the relative distance of the panels according to (1.4.3.123). ┘

1.6 BEM: Various Aspects

1.6.1 Convergence

As in convergence theory for finite elements we can make statements only about **asymptotic convergence** considering families of boundary element trial/test spaces. The reason is that it is usually impossible to predict the size of the discretization error for general boundary value problems. So we have to settle for results merely telling the behavior of discretization error under variation of discretization parameters, read [NumPDE ??], [NumPDE § 3.3.5.12].

The focus will be on **h-refinement**, increasing the resolution of the boundary element spaces by using finer meshes, see [NumPDE Ex. 3.1.4.3].

1.6.1.1 Abstract Galerkin Error Estimate

We recall a fundamental result of [NumPDE Section 3.1] for the Galerkin discretization of linear variational problems (\rightarrow Def. 1.1.5.1)

$$u \in V_0: \quad a(u, v) = \ell(v) \quad \forall v \in V_0, \quad (1.1.5.2)$$

where V_0 is a Hilbert space with norm $\|\cdot\|_V$. Galerkin discretization based on the trial and test space $V_N \subset V_0$, $N := \dim V_N < \infty$, leads to the discrete variational problem

$$u_N \in V_N: \quad a(u_N, v_N) = \ell(v_N) \quad \forall v_N \in V_N, \quad (1.4.1.2)$$

with Galerkin solution $u_N \in V_N$.

Theorem 1.6.1.1. Cea's lemma [NumPDE Thm. 3.1.3.7]

Assume that the bilinear form $a : V_0 \times V_0 \rightarrow \mathbb{R}$ is continuous and **elliptic**, that is

$$\exists C_a > 0: \quad |a(u, v)| \leq C_a \|u\|_V \|v\|_V \quad \forall u, v \in V_0, \quad (1.6.1.2)$$

$$\exists c > 0: \quad |a(v, v)| \geq c \|v\|_V^2 \quad \forall v \in V_0. \quad (1.6.1.3)$$

Then both (1.1.5.2) and (1.4.1.2) have unique solutions $u \in V_0$ and $u_N \in V_N$, respectively, that satisfy

$$\|u - u_N\|_V \leq \frac{C_a}{c} \inf_{v_N \in V_N} \|u - v_N\|_V. \quad (1.6.1.4)$$

The theorem tells us that the norm of the Galerkin discretization error $u - u_N$ is bounded by the **best-approximation error** times a constant that is independent of V_N .

Elliptic first-kind variational BIEs

The assumptions of Thm. 1.6.1.1 are satisfied for most **first-kind** variational BIEs

- ◆ for (1.3.5.15) with $a = a_V$, $V_0 = H^{-\frac{1}{2}}(\Gamma)$ by Thm. 1.3.5.17/Thm. 1.3.5.21 (when $\text{diam}(\Omega) < 1$ for $d = 2$),
- ◆ for (1.3.5.24) with a_W , $V_0 = H_*^{\frac{1}{2}}(\Gamma)$ by Thm. 1.3.5.26.

Remark 1.6.1.6 (Galerkin error estimates for 2nd-kind BIE) Estimates for the Galerkin discretization error for the second-kind variational BIEs (1.3.5.36) and (1.3.5.37) on general curved polyhedra have remained elusive up to date. \lrcorner

1.6.1.2 Approximation in Boundary Element spaces

Thanks to Thm. 1.6.1.1 we can obtain full information about (“energy” trace norms of) the Galerkin discretization error for direct first-kind BIEs by studying how well traces of solutions of boundary value problems can be approximated (in “energy” trace space norms) in boundary element spaces.

§1.6.1.7 (Spaces for functions of higher smoothness on Γ) Approximation error estimates require smoothness of the traces, and this smoothness is conveniently measured in a **Sobolev scale**, recall [NumPDE Section 3.3.3]. Sobolev spaces of functions on smooth faces of Γ are defined via pullback (\rightarrow Def. 1.4.2.8).

As before we make Ass. 1.2.1.5 ($d = 2$) or Ass. 1.2.1.7 ($d = 3$), that is Γ consists of (smooth) faces Γ_j , $j = 1, \dots, M$, with individual parameterizations $\gamma_j : \Pi_j \subset \mathbb{R}^{d-1} \rightarrow \Gamma_j$.

Definition 1.6.1.8. Piecewise Sobolev spaces on Γ

For $m \in \mathbb{N}_0$ and assuming C^m -parameterizations γ_j we define the **piecewise Sobolev space** of order $m \in \mathbb{N}$ on Γ as

$$H_{\text{pw}}^m(\Gamma) := \{v \in L^2(\Gamma), \gamma_j^*(v|_{\Gamma_j}) \in H^m(\Pi_j)\},$$

with (Sobolev) norm

$$\|v\|_{H_{\text{pw}}^m(\Gamma)}^2 := \sum_{j=1}^M \left\| \gamma_j^*(v|_{\Gamma_j}) \right\|_{H^m(\Pi_j)}^2 = \sum_{j=1}^M \int_{\Pi_j} \sum_{\substack{\alpha \in \mathbb{N}_0^{d-1} \\ |\alpha| \leq m}} |D^\alpha \gamma_j^*(v|_{\Gamma_j})(x)|^2 dx, \quad v \in H_{\text{pw}}^m(\Gamma).$$

The definition of $H^m(D)$ for domains $D \subset \mathbb{R}^d$ is also given in [NumPDE Def. 3.3.3.1]. \lrcorner

§1.6.1.9 (Mesh parameters) Approximation estimates for the boundary element spaces $S_p^0(\mathcal{G})$ and $S_p^{-1}(\mathcal{G})$ will hinge on properties of the mesh expressed through fundamental mesh parameters.

Definition 1.6.1.10. Meshwidth

The meshwidth of \mathcal{G} is the size of its largest panel

$$h_{\mathcal{G}} := \max_{\pi \in \mathcal{G}} \text{diam}(\pi).$$

Definition 1.6.1.11. Minimal angle

For $d = 3$ we call the **minimal angle** $\alpha_{\min}(\mathcal{G})$ of \mathcal{G} the minimal angle occurring in all triangles of the 2D meshes \mathcal{M}_j in Def. 1.5.1.4.

For planar triangulations the minimal angle measures the **shape regularity** of a mesh [NumPDE § 3.3.2.19].

§1.6.1.12 (Summary: approximation estimates) The following results from [SS10, Sects. 4.3.4 & 4.3.5] mirror [NumPDE Thm. 3.3.5.6]. In fact, via the pullbacks γ_j^* they can immediately be inferred from interpolation error estimates for finite element spaces in dimension $d - 1$.

Theorem 1.6.1.13. Main approximation theorem for $\mathcal{S}_p^{-1}(\mathcal{G})$

With a constant $C > 0$ depending only on $m \in \mathbb{N}_0$, the C^m -parameterizations γ_j , and the minimal angle $\alpha_{\min}(\mathcal{G})$, for any $p \in \mathbb{N}_0$ we have the best-approximation estimate

$$\inf_{\varphi_N \in \mathcal{S}_p^{-1}(\mathcal{G})} \|u - \varphi_N\|_{H^{-\frac{1}{2}}(\Gamma)} \leq C \left(\frac{h_{\mathcal{G}}}{p+1} \right)^{\min\{p+1, m\} + \frac{1}{2}} \|u\|_{H_{pw}^m(\Gamma)} \quad \forall u \in H_{pw}^m(\Gamma). \tag{1.6.1.14}$$

rate of alg. cvg. smoothness requirement

Theorem 1.6.1.15. Main approximation theorem for $\mathcal{S}_p^0(\mathcal{G})$

With a constant $C > 0$ depending only on $m \geq 2$, the C^m -parameterizations γ_j , and the minimal angle $\alpha_{\min}(\mathcal{G})$, for any $p \in \mathbb{N}$ we have the best-approximation estimate

$$\inf_{v_N \in \mathcal{S}_p^0(\mathcal{G})} \|u - v_N\|_{H^{\frac{1}{2}}(\Gamma)} \leq C \left(\frac{h_{\mathcal{G}}}{p} \right)^{\min\{p+1, m\} - \frac{1}{2}} \|u\|_{H_{pw}^m(\Gamma)} \quad \forall u \in H_{pw}^m(\Gamma) \cap C^0(\Gamma). \tag{1.6.1.16}$$

rate of alg. cvg. smoothness requirement

Algebraic convergence of best approximation errors

The energy norm of the best approximation error for $\mathcal{S}_p^{-1}(\mathcal{G})$, $\mathcal{S}_p^0(\mathcal{G})$ for fixed polynomial degree p converges algebraically (\rightarrow Def. 1.4.3.58) for $h_{\mathcal{G}} \rightarrow 0$, if a uniform minimal angle condition is satisfied for $d = 3$.

We can even read off the **rates** of algebraic convergence in $h_{\mathcal{G}} \rightarrow 0$:

- $\mathcal{S}_p^{-1}(\mathcal{G})$, $p \in \mathbb{N}_0$ for $u \in H_{pw}^m(\Gamma)$, $m \in \mathbb{N}_0$ \triangleright rate $\min\{p+1, m\} + \frac{1}{2}$ in $H^{-\frac{1}{2}}(\Gamma)$ -norm,
- $\mathcal{S}_p^0(\mathcal{G})$, $p \in \mathbb{N}$ for $u \in H_{pw}^m(\Gamma)$, $m \geq 2$ \triangleright rate $\min\{p+1, m\} - \frac{1}{2}$ in $H^{\frac{1}{2}}(\Gamma)$ -norm.

┘

Combined with Thm. 1.6.1.1 we immediately conclude **asymptotic algebraic convergence** of Galerkin boundary element solutions of variational first-kind BIEs in terms of the meshwidth $h_{\mathcal{G}} \rightarrow 0$. ┘

§1.6.1.18 (Smoothness of solution traces) The smoothness of the unknown trace of the solution of the related boundary value problem imposes a limit on the achievable rate of algebraic convergence in the meshwidth $h_{\mathcal{G}}$. In turns, this smoothness is determined by the smoothness of the solution of the boundary value problem.

Theorem 1.6.1.19. Higher order trace theorem

Let $\Gamma := \partial\Omega$ satisfy Ass. 1.2.1.5 ($d = 2$) or Ass. 1.2.1.7 ($d = 3$) with C^∞ -parameterizations γ_j . Then,

$$u \in H^m(\Omega) \quad \text{for } m \geq 1 \quad \Rightarrow \quad T_D(u) \in H_{pw}^{m-1}(\Gamma) \cap C^0(\Gamma), \quad (1.6.1.20)$$

$$u \in H^m(\Omega) \quad \text{for } m \geq 2 \quad \Rightarrow \quad T_N(u) \in H_{pw}^{m-2}(\Gamma). \quad (1.6.1.21)$$

So, when one uses *manufactured solutions* $u \in C^\infty(\mathbb{R}^d)$ to test a boundary element code, the maximal rate of convergence as limited by the polynomial degree p should be observed.

However, in actual computations, the inevitable emergence of **singularities** of the solutions of BVPs on Ω at **corners/edges** of Γ will curtail their smoothness, see [NumPDE Section 3.4]. Thus, for non-smooth Γ only reduced rates of h_G -convergence of fixed-degree BEM will be observed. \perp

§1.6.1.22 (Validation of BEM Galerkin matrices for BIOs) If $u \in H(\Delta, \Omega)$ satisfies $\Delta u = 0$ in Ω , then Thm. 1.3.5.6 provides the fundamental relationships between Dirichlet trace $T_D u$ and Neumann trace $T_N u$ of u :

$$\begin{bmatrix} \frac{1}{2}\text{Id} - K & V \\ W & \frac{1}{2}\text{Id} + K' \end{bmatrix} \begin{bmatrix} T_D u \\ T_N u \end{bmatrix} = \begin{bmatrix} T_D u \\ T_N u \end{bmatrix} \Leftrightarrow \begin{bmatrix} \frac{1}{2}\text{Id} + K & -V \\ -W & \frac{1}{2}\text{Id} - K' \end{bmatrix} \begin{bmatrix} T_D u \\ T_N u \end{bmatrix} = 0. \quad (1.6.1.23)$$

Let us assume that we have a code, allegedly capable of computing the Galerkin matrices

- $V \in \mathbb{R}^{K,K}$, $K := \dim \mathcal{S}_{p-1}^{-1}(\mathcal{G})$, of a_V on $\mathcal{S}_{p-1}^{-1}(\mathcal{G}) \times \mathcal{S}_{p-1}^{-1}(\mathcal{G}) \subset H^{-\frac{1}{2}}(\Gamma) \times H^{-\frac{1}{2}}(\Gamma)$, (single layer BIO)
- $W \in \mathbb{R}^{N,N}$, $N := \dim \mathcal{S}_p^0(\mathcal{G})$, of a_W on $\mathcal{S}_p^0(\mathcal{G}) \times \mathcal{S}_p^0(\mathcal{G}) \subset H^{\frac{1}{2}}(\Gamma) \times H^{\frac{1}{2}}(\Gamma)$, (hypersingular BIO)
- $K \in \mathbb{R}^{K,N}$ of $(v, \psi) \mapsto \int_\Gamma (Kv)(x) \psi(x) dS(x)$ on $\mathcal{S}_p^0(\mathcal{G}) \times \mathcal{S}_{p-1}^{-1}(\mathcal{G}) \subset H^{\frac{1}{2}}(\Gamma) \times H^{-\frac{1}{2}}(\Gamma)$, (double layer BIO)

for some fixed degree $p \in \mathbb{N}$. We want to exploit on (1.6.1.23) to validate the implementation.



Use smooth “manufactured” solution $u \in C^\infty(\bar{\Omega})$ of $\Delta u = 0$ and check, if its **BE interpolants** satisfy (1.6.1.23) “up to higher order errors”.

We consider a sequence of meshes $(\mathcal{G}_h)_{h \in \mathbb{H}}$, $\mathbb{H} := \{h_0, h_1, h_2, \dots\}$, where h is the meshwidth of \mathcal{G}_h and \mathcal{G}_{h_k} arises from $\mathcal{G}_{h_{k-1}}$ by means of **uniform dyadic refinement** (in the parameter domain), which implies $h_k \approx \frac{1}{2}h_{k-1}$ and that $\alpha_{\min}(\mathcal{G}_h) \geq \alpha_0$ for all $h \in \mathbb{H}$, see Def. 1.6.1.11. Uniform dyadic refinement amounts to

- splitting each grid cell $[\zeta_{i-1}^{(j)}, \zeta_i^{(j)}]$ (\rightarrow Def. 1.4.2.5) into two equal intervals for $d = 2$,
- subdividing each triangle of \mathcal{M}_j (\rightarrow Def. 1.5.1.4) into four congruent ones [NumPDE Fig. 173] for $d = 3$.

To define “boundary element interpolants” we use

- for $\mathcal{S}_p^0(\mathcal{G})$ the **nodal interpolation operators** $I_p^0 : C^0(\Gamma) \rightarrow \mathcal{S}_p^0(\mathcal{G})$ from Rem. 1.5.2.10,
- for $\mathcal{S}_{p-1}^{-1}(\mathcal{G})$ the **local L^2 -projections** $Q_{p-1}^{-1} : L^2(\Gamma) \rightarrow \mathcal{S}_{p-1}^{-1}(\mathcal{G})$ defined by

$$\int_\Gamma (Q_{p-1}^{-1} f)(x) \psi_N(x) dS(x) = \int_\Gamma f(x) \psi_N(x) dS(x) \quad \forall \psi_N \in \mathcal{S}_{p-1}^{-1}(\mathcal{G}). \quad (1.6.1.24)$$

Of course, the actual implementation of Q_{p-1}^{-1} has to rely on numerical quadrature on the panels using high-order quadrature formulas.

For smooth u we have $T_D u, T_N u \in H_{pw}^m(\Gamma)$ for every $m \in \mathbb{N}$. Then we can use the following interpolation error estimates.

Theorem 1.6.1.25. Asymptotic interpolation/projection error estimates

With constants depending only on $m > p$, $p \in \mathbb{N}$, the C^m -parameterizations γ (and the minimal angle α_0 for $d = 3$)

$$\left\| u - I_p^0 u \right\|_{H^{\frac{1}{2}}(\Gamma)} \leq C h_{\mathcal{G}}^{p+\frac{1}{2}} \|u\|_{H_{pw}^m(\Gamma)} \quad \forall u \in H_{pw}^m(\Gamma), \quad (1.6.1.26)$$

$$\left\| u - Q_{p-1}^{-1} u \right\|_{H^{-\frac{1}{2}}(\Gamma)} \leq C h_{\mathcal{G}}^{p+\frac{1}{2}} \|u\|_{H_{pw}^m(\Gamma)} \quad \forall u \in H_{pw}^m(\Gamma). \quad (1.6.1.27)$$

Appealing to (1.6.1.23) we expect the **residual functionals** (they depend on p and the mesh \mathcal{G}_h)

$$\begin{aligned} r_D(\psi) &:= \int_{\Gamma} \left(\left(\frac{1}{2} \text{Id} + K \right) I_p(T_D u) - V(Q_{p-1}^{-1} T_N u) \right)(x) \psi(x) dS(x) \\ &= \int_{\Gamma} \left(\left(\frac{1}{2} \text{Id} + K \right) (I_p - \text{Id})(T_D u) - V((Q_{p-1}^{-1} - \text{Id}) T_N u) \right)(x) \psi(x) dS(x), \quad \psi \in H^{-\frac{1}{2}}(\Gamma), \\ r_N(\mathbf{v}) &:= \int_{\Gamma} \left(-W I_p(T_D u) + \left(\frac{1}{2} \text{Id} - K' \right) Q_{p-1}^{-1} T_N u \right)(x) \mathbf{v}(x) dS(x) \\ &= \int_{\Gamma} \left(-W (I_p - \text{Id})(T_D u) + \left(\frac{1}{2} \text{Id} - K' \right) (Q_{p-1}^{-1} - \text{Id}) T_N u \right)(x) \mathbf{v}(x) dS(x), \quad \mathbf{v} \in H^{\frac{1}{2}}(\Gamma), \end{aligned}$$

to become “small” as $h \rightarrow 0$. To quantify this, observe that owing to the continuity of the boundary integral operators (\rightarrow Def. 1.3.4.1) we can conclude from Thm. 1.6.1.25 that on the mesh \mathcal{G}_h

$$|r_D(\psi)| \leq C h^{p+\frac{1}{2}} \cdot \|\psi\|_{H^{-\frac{1}{2}}(\Gamma)}, \quad \psi \in H^{-\frac{1}{2}}(\Gamma), \quad (1.6.1.28)$$

$$|r_N(\mathbf{v})| \leq C h^{p+\frac{1}{2}} \cdot \|\mathbf{v}\|_{H^{\frac{1}{2}}(\Gamma)}, \quad \mathbf{v} \in H^{\frac{1}{2}}(\Gamma), \quad (1.6.1.29)$$

with constants independent of h . We still have to deal with the presence of the general functions ψ and \mathbf{v} .



Replace them with nodal basis functions b_N^i , $i = 1, \dots, N$, and β_N^j , $j = 1, \dots, K$ of $\mathcal{S}_p^0(\mathcal{G})$ and $\mathcal{S}_{p-1}^{-1}(\mathcal{G})$, respectively (\rightarrow § 1.4.2.24, § 1.5.2.16), for which we have rather precise information about their energy trace norms.

We elaborate this for $d = 3$, resorting to a heuristic **scaling argument**. If π is a panel of diameter h and β_N^i a global shape function associated with it, we get from Thm. 1.3.5.17

$$\begin{aligned} \left\| \beta_N^i \right\|_{H^{-\frac{1}{2}}(\Gamma)}^2 &\approx a_V(\beta_N^i, \beta_N^i) \approx \int_{\pi} \int_{\pi} \frac{1}{4\pi \|x - y\|} \beta_N^i(y) \beta_N^i(x) dS(x) dS(y) \\ &\approx h^{2 \cdot 2} \int_{\hat{K}} \int_{\hat{K}} \frac{1}{4\pi h \|s - t\|} \hat{\beta}(s) \hat{\beta}(t) dt ds, \end{aligned}$$

for some *fixed* reference shape functions $\hat{\beta}$, see (1.4.2.27). The last step is justified by thinking of the transformation $\hat{K} \rightarrow \pi$, \hat{K} the reference triangle, as simply a scaling by h . Then the powers of h arise from the transformation formula for $d - 1$ -dimensional integrals. We conclude the asymptotic two-sided estimate (similar arguments for $d = 2$)

$$\left\| \beta_N^i \right\|_{H^{-\frac{1}{2}}(\Gamma)} \approx h^{d/2} \quad \text{on } \mathcal{G}_h, \quad (1.6.1.30)$$

with constants independent of h .

To determine $\|b_N^i\|_{H^{\frac{1}{2}}(\Gamma)}$ recall from Thm. 1.3.4.30 and Thm. 1.3.5.26

$$\|b_N^i\|_{H^{\frac{1}{2}}(\Gamma)}^2 \approx a_W(b_N^i, b_N^i) = a_V(\mathbf{grad}_\Gamma b_N^i \times \mathbf{n}, \mathbf{grad}_\Gamma b_N^i \times \mathbf{n}).$$

Under scaling pullback to \widehat{K} the surface gradient behaves like $\sim h$, Then the same argument as above confirms

$$\|b_N^i\|_{H^{\frac{1}{2}}(\Gamma)} \approx h^{d/2-1} \quad \text{on } \mathcal{G}_h, \quad (1.6.1.31)$$

with constants independent of h .

Thus, setting $\psi := \beta_N^i$ in (1.6.1.28) and $\mathbf{v} := b_N^j$ in (1.6.1.29), the estimates (1.6.1.30) and (1.6.1.31) imply

$$|r_D(\beta_N^i)| \leq C h^{p+\frac{1}{2}+d/2}, \quad |r_D(b_N^j)| \leq C h^{p-\frac{1}{2}+d/2}, \quad \begin{array}{l} i = 1, \dots, K, \\ j = 1, \dots, N. \end{array} \quad (1.6.1.32)$$

Thus, defining the residual coefficient vectors

$$\begin{aligned} \vec{\rho}_D &:= \left[r_D(\beta_N^i) \right]_{i=1}^K = (\tfrac{1}{2}\mathbf{M} + \mathbf{K})\vec{\delta} - \mathbf{V}\vec{v} \in \mathbb{R}^K, \\ \vec{\rho}_N &:= \left[r_N(b_N^j) \right]_{j=1}^N = -\mathbf{W}\vec{\delta} + (\tfrac{1}{2}\mathbf{M}^\top - \mathbf{K}^\top)\vec{v} \in \mathbb{R}^N, \end{aligned} \quad (1.6.1.33)$$

based on the basis expansions

$$\vec{\delta} \in \mathbb{R}^N \leftrightarrow l_p^0(\mathbb{T}_D u) \in \mathcal{S}_p^0(\mathcal{G}) \quad , \quad \vec{v} \in \mathbb{R}^K \leftrightarrow Q_{p-1}^{-1}(\mathbb{T}_N u) \in \mathcal{S}_{p-1}^{-1}(\mathcal{G}) \quad , \quad ,$$

we can predict the algebraic decay of the components:

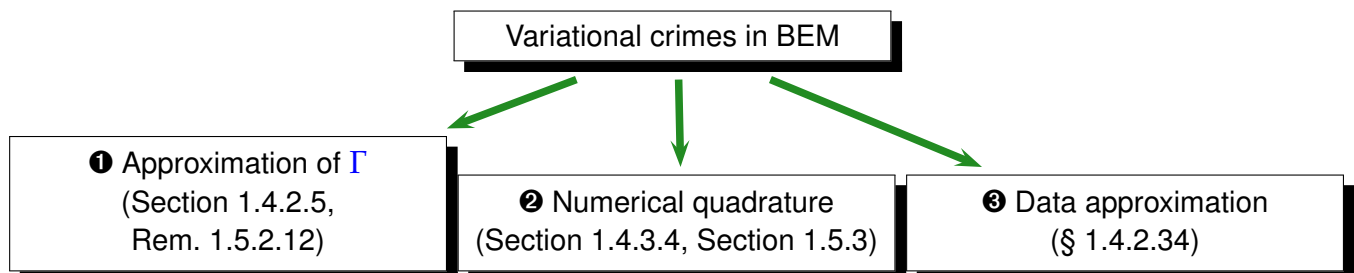
$$\|\vec{\rho}_D\|_\infty = O(h^{p+\frac{1}{2}+d/2}) \quad , \quad \|\vec{\rho}_N\|_\infty = O(h^{p-\frac{1}{2}+d/2}). \quad (1.6.1.34)$$

As we learn from (1.6.1.33), the vectors $\vec{\rho}_D$ and $\vec{\rho}_N$ can be computed. Then tabulate the norms in (1.6.1.34) for sequences of dyadically refined meshes and check whether they exhibit a decay as $h \rightarrow 0$ matching (1.6.1.34). If this is observed, the Galerkin matrices have passed the test. \square

1.6.1.3 Variational Crimes

As in the context of finite element methods [NumPDE Section 3.5], also for boundary element methods the term **variational crime** also for boundary element methods means that Galerkin discretization is based on a perturbed variational problem or even a trial/test space not contained in the function space, on which the original variational problem is posed.

We can distinguish three main categories of variational crimes in BEM:



We recall from [NumPDE Section 3.5]:

Guideline for acceptable variational crimes

Variational crimes must not interfere with (type and rate) of asymptotic convergence!

► For Galerkin boundary element methods based on the piecewise polynomial boundary element spaces $\mathcal{S}_{p-1}^{-1}(\mathcal{G}) \subset L^2(\Gamma) \subset H^{-\frac{1}{2}}(\Gamma)$, $\mathcal{S}_p^0(\mathcal{G}) \subset H^{\frac{1}{2}}(\Gamma)$, $p \in \mathbb{N}$,

- ◆ the degree of polynomial boundary approximation must be linked to p ,
- ◆ the boundary element spaces for data approximation must depend on p ,
- ◆ the order of numerical quadrature must be larger for larger p .

§1.6.1.36 (Quantitative recipes) A very detailed quantitative analysis of variational crimes of type ❶ is conducted in [SS10, Ch. 8] and of type ❷ in [SS10, Sect. 5.3]. These results and practical experience inspire the following **rules of thumb**:

If the following trial/test spaces are used for variational BIE in energy trace space

$$\mathcal{S}_{p-1}^{-1}(\mathcal{G}) \subset L^2(\Gamma) \subset H^{-\frac{1}{2}}(\Gamma) \quad , \quad \mathcal{S}_p^0(\mathcal{G}) \subset H^{\frac{1}{2}}(\Gamma) \quad , \quad p \in \mathbb{N} \quad ,$$

then do the following:

- ❶ for the approximation of Γ :

use piecewise polynomial interpolants of degree p ,

- ❷ for computation of entries of Galerkin matrices by means of numerical quadrature

follow (1.4.3.123), but no clear rule for selecting order of Gauss quadrature rules in general,

- ❸ for data approximation:

interpolate Dirichlet in $\mathcal{S}_p^0(\mathcal{G})$, Neumann data in $\mathcal{S}_{p-1}^{-1}(\mathcal{G})$.

┘

1.6.1.4 Pointwise Recovery of Solutions

1.6.2 Mixed Boundary Value Problems

In mixed second-order elliptic boundary value problems both Dirichlet and Neumann boundary conditions are imposed on different parts Γ_D and Γ_N of the boundary $\Gamma := \partial\Omega$ of the computational domain $\Omega \subset \mathbb{R}^d$ [NumPDE Section 1.7], which satisfy

$$\Gamma = \bar{\Gamma}_D \cup \bar{\Gamma}_N \quad , \quad \Gamma_D \cap \Gamma_N = \emptyset \quad , \quad \text{vol}_{d-1}(\Gamma_N), \text{vol}_{d-1}(\Gamma_D) > 0 . \quad (1.6.2.1)$$

The associated mixed BVP for $-\Delta$ reads

$$-\Delta u = 0 \quad \text{in } \Omega \quad , \quad \begin{aligned} \mathbb{T}_D u &= \mathbf{g} \quad \text{on } \Gamma_D \quad , \\ \mathbb{T}_N u &= \boldsymbol{\eta} \quad \text{on } \Gamma_N \quad , \end{aligned} \quad (1.6.2.2)$$

where $\mathbf{g} : \Gamma_D \rightarrow \mathbb{R}$ and $\boldsymbol{\eta} : \Gamma_N \rightarrow \mathbb{R}$ are given data. If Ω is an exterior unbounded domain, we have to impose additional decay conditions (1.1.7.1)/(1.1.7.4).

§1.6.2.3 (Offset technique for BIE) By Thm. 1.3.5.6 the traces of the solution u satisfy the fundamental boundary integral equations

$$\mathbb{T}_D u = \mathbb{V}(\mathbb{T}_N u) - \left(-\frac{1}{2}\text{Id} + \mathbb{K}\right)(\mathbb{T}_D u) \quad \text{in } H^{\frac{1}{2}}(\Gamma) \quad , \quad (1.6.2.4)$$

$$\mathbb{T}_N u = \left(\frac{1}{2}\text{Id} + \mathbb{K}'\right)(\mathbb{T}_N u) + \mathbb{W}(\mathbb{T}_D u) \quad \text{in } H^{-\frac{1}{2}}(\Gamma). \quad (1.6.2.5)$$

To take into account the fact that both traces are known on some parts of the boundary we introduce **extensions of the data** to all of Γ :

$$\begin{aligned} \tilde{\mathbf{g}} &\in H^{\frac{1}{2}}(\Gamma): & \tilde{\mathbf{g}}|_{\Gamma_D} &= \mathbf{g}, \\ \tilde{\eta} &\in H^{-\frac{1}{2}}(\Gamma): & \tilde{\eta}|_{\Gamma_N} &= \eta. \end{aligned} \quad (1.6.2.6)$$



Offset function technique: We seek the unknown traces as additive corrections of these extended data, the corrections of course supported on either Γ_D or Γ_N [SS10, Sect. 3.5.2].

$$\begin{aligned} \mathbb{T}_D u &= \tilde{\mathbf{g}} + \mathbf{u}, \quad \mathbf{u} \in H_{\Gamma_D}^{\frac{1}{2}}(\Gamma) := \{\mathbf{v} \in H^{\frac{1}{2}}(\Gamma) : \mathbf{v}|_{\Gamma_D} = 0\}, \\ \mathbb{T}_N u &= \tilde{\eta} + \psi, \quad \psi \in H_{\Gamma_N}^{-\frac{1}{2}}(\Gamma) := \{\phi \in H^{-\frac{1}{2}}(\Gamma) : \phi|_{\Gamma_N} = 0\}. \end{aligned} \quad (1.6.2.7)$$

The functions $\tilde{\mathbf{g}}$ and $\tilde{\eta}$ serve as **offset functions** in a context similar to the use of offset functions for imposing essential boundary conditions in variational formulations of boundary value problems for PDEs as discussed in [NumPDE § 1.2.3.12].

Next, we insert (1.6.2.7) into (1.6.2.4) and 1.6.2.5 and get

$$0 = \mathbb{V}(\tilde{\eta} + \psi) - \left(\frac{1}{2}\text{Id} + \mathbb{K}\right)(\tilde{\mathbf{g}} + \mathbf{u}) \quad \text{in } H^{\frac{1}{2}}(\Gamma), \quad (1.6.2.8)$$

$$0 = \left(-\frac{1}{2}\text{Id} + \mathbb{K}'\right)(\tilde{\eta} + \psi) + \mathbb{W}(\tilde{\mathbf{g}} + \mathbf{u}) \quad \text{in } H^{-\frac{1}{2}}(\Gamma). \quad (1.6.2.9)$$

The unknowns are $\mathbf{u} \in H_{\Gamma_D}^{\frac{1}{2}}(\Gamma)$, $\psi \in H_{\Gamma_N}^{-\frac{1}{2}}(\Gamma)$. ┘

§1.6.2.10 (Variational BIE for correction trace functions) Collecting known and unknown quantities in 1.6.2.8 and 1.6.2.9 leads to

$$-\mathbb{V}(\psi) + \left(\frac{1}{2}\text{Id} + \mathbb{K}\right)(\mathbf{u}) = \mathbb{V}(\tilde{\eta}) - \left(\frac{1}{2}\text{Id} + \mathbb{K}\right)(\tilde{\mathbf{g}}) \quad \text{in } H^{\frac{1}{2}}(\Gamma), \quad (1.6.2.11)$$

$$\left(\frac{1}{2}\text{Id} - \mathbb{K}'\right)(\psi) - \mathbb{W}(\mathbf{u}) = \left(-\frac{1}{2}\text{Id} + \mathbb{K}'\right)(\tilde{\eta}) + \mathbb{W}(\tilde{\mathbf{g}}) \quad \text{in } H^{-\frac{1}{2}}(\Gamma). \quad (1.6.2.12)$$

As usual, a variational formulation arises from invoking duality (1.3.1.41). Yet, we have to ensure that trial and test spaces are the same. The trial spaces are the trace spaces for the unknowns \mathbf{u} and ψ and those have to be chosen from $H_{\Gamma_D}^{\frac{1}{2}}(\Gamma)$ and $H_{\Gamma_N}^{-\frac{1}{2}}(\Gamma)$, respectively. Thus, we

- do not test (1.6.2.8) with $H^{-\frac{1}{2}}(\Gamma)$, but with $\mathbf{v} \in H_{\Gamma_N}^{-\frac{1}{2}}(\Gamma)$,
("Test (1.6.2.8) only where $\mathbb{T}_N u$ is not known")
- do not test 1.6.2.9 with $H^{\frac{1}{2}}(\Gamma)$, but with $\mathbf{v} \in H_{\Gamma_D}^{\frac{1}{2}}(\Gamma)$.
("Test 1.6.2.9 only where $\mathbb{T}_D u$ is not known")

This leads to a linear variational problem in $H_{\Gamma_N}^{-\frac{1}{2}}(\Gamma) \times H_{\Gamma_D}^{\frac{1}{2}}(\Gamma)$:

$$\begin{aligned} \psi \in H_{\Gamma_N}^{-\frac{1}{2}}(\Gamma): \quad & -a_{\mathbb{V}}(\psi, \mathbf{v}) + \int_{\Gamma} \left(\left(\frac{1}{2}\text{Id} + \mathbb{K}\right)(\mathbf{u})\right)(\mathbf{x}) \mathbf{v}(\mathbf{x}) \, dS(\mathbf{x}) \\ & = a_{\mathbb{V}}(\tilde{\eta}, \mathbf{v}) - \int_{\Gamma} \left(\left(\frac{1}{2}\text{Id} + \mathbb{K}\right)(\tilde{\mathbf{g}})\right)(\mathbf{x}) \mathbf{v}(\mathbf{x}) \, dS(\mathbf{x}) \quad \forall \mathbf{v} \in H_{\Gamma_N}^{-\frac{1}{2}}(\Gamma), \end{aligned} \quad (1.6.2.13a)$$

$$\mathbf{u} \in H_{\Gamma_D}^{\frac{1}{2}}(\Gamma): \quad \int_{\Gamma} \left(\left(\frac{1}{2}\text{Id} - \mathbb{K}'\right)(\psi)\right)(\mathbf{x}) \mathbf{v}(\mathbf{x}) \, dS(\mathbf{x}) - a_{\mathbb{W}}(\mathbf{u}, \mathbf{v}) \quad (1.6.2.13b)$$

$$= \int_{\Gamma} ((-\frac{1}{2}\text{Id} + K')(\tilde{\eta}))(x) \mathbf{v}(x) dS(x) + a_W(\tilde{\mathbf{g}}, \mathbf{v}) \quad \forall \mathbf{v} \in H_{\Gamma_D}^{\frac{1}{2}}(\Gamma).$$

§1.6.2.14 (Boundary element discretization of variational BIE for mixed BVP) We suppose that we are given a mesh \mathcal{G} of Γ according to Def. 1.4.2.5 ($d = 2$) or Def. 1.5.1.4 ($d = 3$) that resolves the parts Γ_D and Γ_N of the boundary in the following sense.

Assumption 1.6.2.15. Mesh compatible with partition

Both $\bar{\Gamma}_D$ and $\bar{\Gamma}_N$ are the union of closed panels of the mesh \mathcal{G} .

We have to adapt the boundary element spaces $\mathcal{S}_p^0(\mathcal{G}) \subset H^{\frac{1}{2}}(\Gamma)$ and $\mathcal{S}_{p-1}^{-1}(\mathcal{G}) \subset H^{-\frac{1}{2}}(\Gamma)$, degree $p \in \mathbb{N}$, in order to obtain subspaces of $H_{\Gamma_D}^{\frac{1}{2}}(\Gamma)$ and $H_{\Gamma_N}^{-\frac{1}{2}}(\Gamma)$. On the formal level this is straightforward

$$\mathcal{S}_{p,\Gamma_D}^0(\mathcal{G}) := H_{\Gamma_D}^{\frac{1}{2}}(\Gamma) \cap \mathcal{S}_p^0(\mathcal{G}) \quad , \quad \mathcal{S}_{p-1,\Gamma_N}^{-1}(\mathcal{G}) := H_{\Gamma_N}^{-\frac{1}{2}}(\Gamma) \cap \mathcal{S}_{p-1}^{-1}(\mathcal{G}). \quad (1.6.2.16)$$

In practice,

$\mathcal{S}_{p,\Gamma_D}^0(\mathcal{G})$ and $\mathcal{S}_{p-1,\Gamma_N}^{-1}(\mathcal{G})$ are obtained by dropping all global shape functions of $\mathcal{S}_p^0(\mathcal{G})/\mathcal{S}_{p-1}^{-1}(\mathcal{G})$ whose supports intersect Γ_D or Γ_N , respectively.

The construction runs utterly parallel to that of finite element subspaces of $H_0^1(\Omega)$ from finite element subspaces of $H^1(\Omega)$, see [NumPDE § 2.4.3.7].

Note that the Galerkin matrices for the variational boundary integral operators arising from using the boundary element spaces $\mathcal{S}_{p,\Gamma_D}^0(\mathcal{G})$ and $\mathcal{S}_{p-1,\Gamma_N}^{-1}(\mathcal{G})$ are sub-matrices of the Galerkin matrices we get when using the unconstrained boundary element spaces.

As explained in § 1.4.2.34, in boundary element computations the data \mathbf{g} and η are usually replaced with approximations. In the case of (1.6.2.13) this approximation also takes care of extension of the data to all of Γ :

- ◆ $\tilde{\mathbf{g}}$ is replaced with $\mathbf{g}_N \in \mathcal{S}_p^0(\mathcal{G})$ obtained by
 1. interpolating \mathbf{g} in $\mathcal{S}_p^0(\mathcal{G})|_{\Gamma_D}$
(e.g., piecewise linear interpolation in the case of $p = 1$),
 2. and then setting the contribution of all shape function supported outside Γ_D to zero.
- ◆ $\tilde{\eta}$ is replaced with $\eta_N \in \mathcal{S}_{p-1}^{-1}(\mathcal{G})$, obtained by
 1. interpolating η in $\mathcal{S}_{p-1}^{-1}(\mathcal{G})|_{\Gamma_N}$
(e.g., midpoint interpolation onto piecewise constants for $p = 1$),
 2. and then setting the contribution of all shape function supported outside Γ_N to zero.

┘

1.6.3 Transmission Problems

So far we have discussed BEM for scalar elliptic boundary value problems with constant coefficients. This section will present boundary integral equations related to problems with **piecewise constant coefficients** posed on \mathbb{R}^d , so-called **transmission problems**.

1.6.3.1 Two-Domain Setting

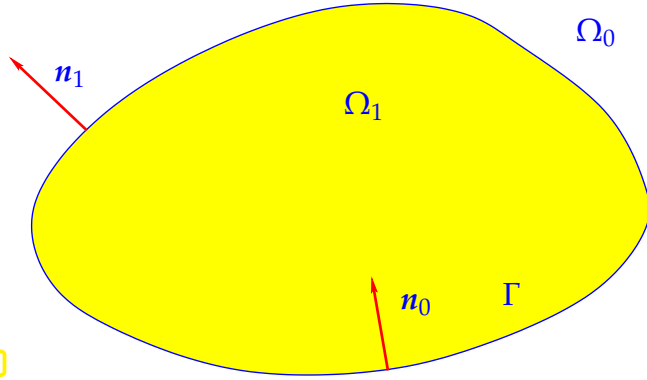


Fig. 58

◁ We consider a partition

$$\begin{aligned} \mathbb{R}^d &= \Omega_0 \cup \Gamma \cup \Omega_1, \\ \Gamma &= \partial\Omega_0 = \partial\Omega_1, \end{aligned} \tag{1.6.3.1}$$

where Γ is a curved Lipschitz polygon ($d = 2$) or polyhedron ($d = 3$), Ω_1 is bounded.

Note the opposite orientation of the two normals n_0 and n_1 .

We seek a solution of

$$-\operatorname{div}(\mathbf{A}(x) \operatorname{grad} u) = 0 \quad \text{in } \mathbb{R}^3, \tag{1.6.3.2a}$$

$$\text{with } \mathbf{A}(x) = \begin{cases} \mathbf{A}_1 \in \mathbb{R}^{d,d} \text{ s.p.d.} & \text{for } x \in \Omega_1, \\ \mathbf{I} & \text{for } x \in \Omega_0, \end{cases} \tag{1.6.3.2b}$$

$$u - u_{\text{inc}} \text{ satisfies decay conditions (1.1.7.1)/(1.1.7.4).} \tag{1.6.3.2c}$$

Here u_{inc} is a given exciting incident field satisfying $\Delta u_{\text{inc}} = 0$ in \mathbb{R}^d . For instance, it may represent an applied external electric field; $u_{\text{inc}}(x) = \mathbf{E}_0 \cdot x$.

§1.6.3.3 (Reformulation as transmission problem) We can restrict solution u of (1.6.3.2) to both domains and define

$$u_0 := u|_{\Omega_0} - u_{\text{inc}} \in H(\Delta, \Omega_0) \quad , \quad u_1 := u|_{\Omega_1} \in H(\Delta, \Omega_1), \tag{1.6.3.4}$$

where $H(\Delta, \Omega)D$ has been introduced in Def. 1.3.1.32. These functions solve

$$-\Delta(\mathbf{A}_1 \operatorname{grad} u_1) = 0 \quad \text{in } \Omega_1 \quad , \quad -\Delta u_0 = 0 \quad \text{in } \Omega_0. \tag{1.6.3.5}$$

In § 1.1.6.10 we learned that u_0 and u_1 are connected by transmission conditions reflecting the continuity of scalar potentials and the normal continuity of displacement currents. We can state them concisely by means of Dirichlet and Neumann traces:

$$\mathbb{T}_D^0 u_0 + \mathbb{T}_D^0 u_{\text{inc}} = \mathbb{T}_D^1 u_1 \quad , \quad \mathbb{T}_N^0 u_0 + \mathbb{T}_N^0 u_{\text{inc}} = -\mathbb{T}_N^1 u_1, \tag{1.6.3.6}$$

where we remind that the coefficients and the normal vectors (responsible for the $-$ -sign) enter the definition of the Neumann trace

$$(\mathbb{T}_N^0 u)(x) = \operatorname{grad} u|_{\Omega_0}(x) \cdot n_0(x) \quad , \quad (\mathbb{T}_N^1 u)(x) = \mathbf{A}_1 \operatorname{grad} u|_{\Omega_1}(x) \cdot n_1(x) \quad , \quad x \in \Gamma. \tag{1.6.3.7}$$

The partial differential equations (1.6.3.5) together with the transmission conditions (1.6.3.6) and decay conditions for u_0 represent a transmission problem. ┘

§1.6.3.8 (First-kind boundary integral equations) In Ex. 1.2.2.27, (1.2.2.34) we found the fundamental solution for the general linear, translation-invariant second-order differential operator $\mathbf{L}u := -\Delta(\mathbf{A} \operatorname{grad} u)$ with symmetric positive definite (s.p.d.) matrix $\mathbf{A} \in \mathbb{R}^{d,d}$. Drawing on (1.2.2.34) we set

$$G^0(x, y) = \begin{cases} -\frac{1}{2\pi} \log \|x - y\| & , \text{ if } d = 2, \\ \frac{1}{4\pi \|x - y\|} & , \text{ if } d = 3, \end{cases} \quad x \neq y,$$

$$G^1(x, y) = \frac{1}{\sqrt{\det \mathbf{A}_1}} \cdot \begin{cases} -\frac{1}{4\pi} \log \left((x - y)^\top \mathbf{A}_1^{-1} (x - y) \right) & , \text{ if } d = 2, \\ \frac{1}{4\pi \sqrt{(x - y)^\top \mathbf{A}_1^{-1} (x - y)}} & , \text{ if } d = 3, \end{cases} \quad x \neq y.$$

for the fundamental solutions associated with the PDE in Ω_0 and Ω_1 , respectively. Based on these the fundamental solutions G^0 and G^1 we can introduce boundary integral operators V_0, K_0, K'_0 , and W_0 , and V_1, K_1, K'_1 , and W_1 . The subscript indicates, which fundamental solution and which Neumann trace operator is used in their definition, for instance, cf. (1.3.4.14),

$$\begin{aligned} (K_0 v)(x) &= \int_{\Gamma} \mathbf{grad}_y G^0(x, y) \cdot \mathbf{n}_0(y) v(y) dS(y), \\ (K_1 v)(x) &= \int_{\Gamma} (\mathbf{A}_1 \mathbf{grad}_y G^1(x, y)) \cdot \mathbf{n}_1(y) v(y) dS(y), \end{aligned} \quad x \in \Gamma.$$



- Idea:
- ❶ Use the fundamental boundary integral identities of Thm. 1.3.5.6 both in Ω_0 and Ω_1 .
 - ❷ Combine them with the transmission conditions (1.6.3.6).

❷: (1.3.5.7) gives us

$$\begin{bmatrix} \frac{1}{2}\text{Id} + K_0 & -V_0 \\ -W_0 & \frac{1}{2}\text{Id} - K'_0 \end{bmatrix} \begin{bmatrix} T_D^0 u_0 \\ T_N^0 u_0 \end{bmatrix} = 0, \tag{1.6.3.9a}$$

$$\begin{bmatrix} \frac{1}{2}\text{Id} + K_1 & -V_1 \\ -W_1 & \frac{1}{2}\text{Id} - K'_1 \end{bmatrix} \begin{bmatrix} T_D^1 u_1 \\ T_N^1 u_1 \end{bmatrix} = 0, \tag{1.6.3.9b}$$

❷: Eliminate $\begin{bmatrix} T_D^0 u_0 \\ T_N^0 u_0 \end{bmatrix}$ by means of the transmission conditions (1.6.3.6):

$$\begin{aligned} \begin{bmatrix} T_D^0 u_0 \\ T_N^0 u_0 \end{bmatrix} &= \begin{bmatrix} T_D^1 u_1 \\ -T_N^1 u_1 \end{bmatrix} - \begin{bmatrix} T_D^0 u_{\text{inc}} \\ T_N^0 u_{\text{inc}} \end{bmatrix}. \\ \blacktriangleright \begin{bmatrix} \frac{1}{2}\text{Id} + K_0 & -V_0 \\ -W_0 & \frac{1}{2}\text{Id} - K'_0 \end{bmatrix} \begin{bmatrix} T_D^1 u_1 \\ -T_N^1 u_1 \end{bmatrix} &= \begin{bmatrix} \frac{1}{2}\text{Id} + K_0 & -V_0 \\ -W_0 & \frac{1}{2}\text{Id} - K'_0 \end{bmatrix} \begin{bmatrix} T_D^0 u_{\text{inc}} \\ T_N^0 u_{\text{inc}} \end{bmatrix} =: \begin{bmatrix} f \\ \varphi \end{bmatrix}. \end{aligned} \tag{1.6.3.10}$$

Then subtract the two boundary integral equations:

$$(1.6.3.9b) \quad \blacktriangleright \quad \begin{bmatrix} \frac{1}{2}\text{Id} + K_1 & -V_1 \\ -W_1 & \frac{1}{2}\text{Id} - K'_1 \end{bmatrix} \begin{bmatrix} T_D^1 u_1 \\ T_N^1 u_1 \end{bmatrix} = 0,$$

$$(1.6.3.10) \quad \blacktriangleright \quad \begin{bmatrix} \frac{1}{2}\text{Id} + K_0 & V_0 \\ W_0 & \frac{1}{2}\text{Id} - K'_0 \end{bmatrix} \begin{bmatrix} T_D^1 u_1 \\ T_N^1 u_1 \end{bmatrix} = \begin{bmatrix} f \\ -\varphi \end{bmatrix}$$

$$\begin{bmatrix} K_1 - K_0 & -V_1 - V_0 \\ -W_1 - W_0 & -K'_1 + K'_0 \end{bmatrix} \begin{bmatrix} T_D^1 u_1 \\ T_N^1 u_1 \end{bmatrix} = \begin{bmatrix} -f \\ \varphi \end{bmatrix}.$$

Writing $\mathbf{u} := T_D^1 u_1$ and $\psi := T_N^1 u_1$ for the unknown traces we get the following boundary integral equations for the transmission problem

$$\begin{bmatrix} K_1 - K_0 & -V_1 - V_0 \\ -W_1 - W_0 & -K'_1 + K'_0 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \psi \end{bmatrix} = \begin{bmatrix} -f \\ \varphi \end{bmatrix}. \tag{1.6.3.11}$$

If this system of boundary integral equations has a unique solution, then \mathbf{u} and ψ will furnish traces on Γ of the solution u of (1.6.3.2), see Cor. 1.6.3.22 below. Thus, (1.6.3.11) qualifies as a *direct* BIE formulation. ┘

Remark 1.6.3.12 (Simplification of right-hand side) As in Section 1.3.4.1 let $V, K, K',$ and W denote the four fundamental boundary integral operators for $-\Delta$ on Ω_1 . Since we have assumed $\Delta u_{\text{inc}} = 0$ on \mathbb{R}^d , we know that u_{inc} is harmonic in Ω_1 . Hence, Thm. 1.3.5.6 yields the identity

$$\begin{bmatrix} \frac{1}{2}\text{Id} - K & V \\ W & \frac{1}{2}\text{Id} + K' \end{bmatrix} \begin{bmatrix} T_D u_{\text{inc}} \\ T_N u_{\text{inc}} \end{bmatrix} = \begin{bmatrix} T_D u_{\text{inc}} \\ T_N u_{\text{inc}} \end{bmatrix}. \tag{1.6.3.13}$$

Here T_N is the “standard” Neumann trace (\rightarrow Def. 1.3.1.20) from within Ω_1 : $T_N u_{\text{inc}} := \text{grad } u_{\text{inc}} \cdot n_1$. Also note that $V, K, K',$ and W are based on the same fundamental solution G^0 as $V_0, K_0, K'_0,$ and W_0 , but on a normal vector with opposite orientation. Therefore, a scrutiny of Def. 1.3.4.1 reveals that

$$V = V_0, \quad K = -K_0, \quad K' = -K'_0, \quad W = W_0. \tag{1.6.3.14}$$

in addition $T_D u_{\text{inc}} = T_D^0 u_{\text{inc}}$ and $T_N u_{\text{inc}} = -T_N^0 u_{\text{inc}}$ (change of the orientation of normals!), so that we can rewrite (1.6.3.13) as

$$\begin{aligned} \begin{bmatrix} \frac{1}{2}\text{Id} + K_0 & V_0 \\ W_0 & \frac{1}{2}\text{Id} - K'_0 \end{bmatrix} \begin{bmatrix} T_D^0 u_{\text{inc}} \\ -T_N^0 u_{\text{inc}} \end{bmatrix} &= \begin{bmatrix} T_D^0 u_{\text{inc}} \\ -T_N^0 u_{\text{inc}} \end{bmatrix} \\ \Updownarrow & \\ \begin{bmatrix} \frac{1}{2}\text{Id} + K_0 & -V_0 \\ -W_0 & \frac{1}{2}\text{Id} - K'_0 \end{bmatrix} \begin{bmatrix} T_D^0 u_{\text{inc}} \\ T_N^0 u_{\text{inc}} \end{bmatrix} &= \begin{bmatrix} T_D^0 u_{\text{inc}} \\ T_N^0 u_{\text{inc}} \end{bmatrix}. \end{aligned} \tag{1.6.3.15}$$

Compare this with the definition

$$\begin{bmatrix} f \\ \varphi \end{bmatrix} := \begin{bmatrix} \frac{1}{2}\text{Id} + K_0 & -V_0 \\ -W_0 & \frac{1}{2}\text{Id} - K'_0 \end{bmatrix} \begin{bmatrix} T_D^0 u_{\text{inc}} \\ T_N^0 u_{\text{inc}} \end{bmatrix} \quad \blacktriangleright \quad \begin{bmatrix} f \\ \varphi \end{bmatrix} = \begin{bmatrix} T_D^0 u_{\text{inc}} \\ T_N^0 u_{\text{inc}} \end{bmatrix}.$$

The right hand side of (1.6.3.11) boils down to simple Dirichlet and Neumann traces of the exciting harmonic function u_{inc} ! ┘

§1.6.3.16 (Variational BIE for transmission problem) We can rewrite (1.6.3.11) as

$$\begin{aligned} (K_1 - K_0)u &- (V_1 + V_0)\psi &= -f &\text{ in } H^{\frac{1}{2}}(\Gamma), \\ -(W_1 + W_0)u &+ (-K'_1 + K'_0)\psi &= \varphi &\text{ in } H^{-\frac{1}{2}}(\Gamma). \end{aligned}$$

The customary approach via **duality** (1.3.1.41) gives us an equivalent variational first-kind (\rightarrow Rem. 1.3.5.28) BIE:

$$\begin{aligned} u \in H^{\frac{1}{2}}(\Gamma), \quad \psi \in H^{-\frac{1}{2}}(\Gamma) : \quad & a_{K,1}(u, \eta) - a_{K,0}(u, \eta) - a_{V,1}(\psi, \eta) + a_{V,0}(\psi, \eta) = \\ & - \int_{\Gamma} f(x) \eta(x) \, dS(x) \quad \forall \eta \in H^{-\frac{1}{2}}(\Gamma), \\ & -a_{W,1}(u, v) - a_{W,0}(u, v) - a_{K,1}(v, \psi) + a_{K,0}(v, \psi) = \\ & \int_{\Gamma} \varphi(x) v(x) \, dS(x) \quad \forall v \in H^{\frac{1}{2}}(\Gamma), \end{aligned} \tag{1.6.3.17}$$

where we have used the “adjointness” (1.4.2.46) of K_i and $K'_i, i = 0, 1$. The bilinear forms in (1.6.3.17) are defined as, $i = 0, 1$,

$$a_{V,i}(\psi, \eta) := \int_{\Gamma} (V_i \psi)(x) \eta(x) \, dS(x), \quad \psi, \eta \in H^{-\frac{1}{2}}(\Gamma) \quad , \text{ cf. (1.3.5.15)}$$

$$a_{W,i}(u, v) := \int_{\Gamma} (W_i u)(x) v(x) \, dS(x), \quad u, v \in H^{\frac{1}{2}}(\Gamma) \quad , \text{ cf. (1.3.5.24)}$$

$$a_{K,i}(v, \eta) := \int_{\Gamma} (K_i v)(x) \eta(x) \, dS(x), \quad v \in H^{\frac{1}{2}}(\Gamma), \eta \in H^{-\frac{1}{2}}(\Gamma).$$

The variational problem (1.6.3.17) is posed on $H^{\frac{1}{2}}(\Gamma) \times H^{-\frac{1}{2}}(\Gamma)$ and can be expressed as

$$\begin{aligned} \begin{bmatrix} u \\ \psi \end{bmatrix} &\in H^{\frac{1}{2}}(\Gamma) \times H^{-\frac{1}{2}}(\Gamma) : \\ c\left(\begin{bmatrix} u \\ \psi \end{bmatrix}, \begin{bmatrix} v \\ \eta \end{bmatrix}\right) &= \int_{\Gamma} \varphi(x) v(x) - f(x) \eta(x) \, dS(x) \\ \forall \begin{bmatrix} v \\ \eta \end{bmatrix} &\in H^{\frac{1}{2}}(\Gamma) \times H^{-\frac{1}{2}}(\Gamma) . \end{aligned} \quad (1.6.3.18)$$

with the bilinear form

$$c\left(\begin{bmatrix} u \\ \psi \end{bmatrix}, \begin{bmatrix} v \\ \eta \end{bmatrix}\right) := a_{K,1}(u, \eta) - a_{K,0}(u, \eta) - a_{V,1}(\psi, \eta) - a_{V,0}(\psi, \eta) - a_{W,1}(u, v) - a_{W,0}(u, v) - a_{K,1}(v, \psi) + a_{K,0}(v, \psi) . \quad (1.6.3.19)$$

Lemma 1.6.3.20. Ellipticity of c

(Assuming $\text{diam}(\Omega_1) < 1$ for $d = 2$,) the bilinear form c from (1.6.3.19) of the first-kind variational boundary integral equations for the transmission problem is $H^{\frac{1}{2}}(\Gamma) \times H^{-\frac{1}{2}}(\Gamma)$ -elliptic:

$$\left| c\left(\begin{bmatrix} v \\ \eta \end{bmatrix}, \begin{bmatrix} v \\ \eta \end{bmatrix}\right) \right| \geq c(\|v\|_{H^{\frac{1}{2}}(\Gamma)}^2 + \|\eta\|_{H^{-\frac{1}{2}}(\Gamma)}^2) \quad \forall v \in H^{\frac{1}{2}}(\Gamma), \eta \in H^{-\frac{1}{2}}(\Gamma) , \quad (1.6.3.21)$$

with $c > 0$ depending on Γ and \mathbf{A}_0 .

Proof. Observing the **cancellation** of all terms contributed by double layer BIODs, the result is an immediate consequence of Thm. 1.3.5.17, Thm. 1.3.5.21, and Thm. 1.3.5.26. □

We immediately conclude *uniqueness and existence* of a solution $\begin{bmatrix} u \\ \psi \end{bmatrix}$ of (1.6.3.18). By its derivation these are the traces of the solution of the transmission problem on Γ .

Corollary 1.6.3.22. Direct 1st-kind variational BIE for transmission problem

If u solves the transmission problem (1.6.3.2) and $\begin{bmatrix} u \\ \psi \end{bmatrix}$ solves (1.6.3.18), then

$$u = \mathcal{T}_D^1 u \quad , \quad \psi = \mathcal{T}_N^1 u .$$

┘

§1.6.3.23 (Direct BEM for transmission two-domain problem) Galerkin boundary element discretization of (1.6.3.18) is straightforward: Given a standard mesh \mathcal{G} of Γ we opt for the natural trial/test spaces from Section 1.4.2/Section 1.5.2

$$\mathcal{S}_{p-1}^{-1}(\mathcal{G}) \text{ for } H^{-\frac{1}{2}}(\Gamma) \quad , \quad \mathcal{S}_p^0(\mathcal{G}) \text{ for } H^{\frac{1}{2}}(\Gamma) . \quad (1.6.3.24)$$

The resulting discrete version of (1.6.3.18) will also enjoy existence and uniqueness of solutions. Based on nodal bases we arrive at the following linear system of equations written in block form

$$\begin{bmatrix} \mathbf{W}_0 + \mathbf{W}_1 & \mathbf{K}_1^T - \mathbf{K}_0^T \\ -\mathbf{K}_1 + \mathbf{K}_0 & \mathbf{V}_0 + \mathbf{V}_1 \end{bmatrix} \begin{bmatrix} \vec{\mu} \\ \vec{\psi} \end{bmatrix} = \begin{bmatrix} -\mathbf{M}^T \vec{\kappa} \\ \mathbf{M} \vec{\phi} \end{bmatrix} . \quad (1.6.3.25)$$

with boundary element Galerkin matrices

- $\mathbf{W}_i \in \mathbb{R}^{N,N}$, $N := \dim \mathcal{S}_p^0(\mathcal{G})$ for $\mathbf{a}_{\mathbf{W},i}$ on $\mathcal{S}_p^0(\mathcal{G}) \times \mathcal{S}_p^0(\mathcal{G})$,
- $\mathbf{V}_i \in \mathbb{R}^{K,K}$, $K := \dim \mathcal{S}_{p-1}^{-1}(\mathcal{G})$ for $\mathbf{a}_{\mathbf{V},i}$ on $\mathcal{S}_{p-1}^{-1}(\mathcal{G}) \times \mathcal{S}_{p-1}^{-1}(\mathcal{G})$,
- $\mathbf{K}_i \in \mathbb{R}^{K,N}$ for $\mathbf{a}_{\mathbf{K},i}$ on $\mathcal{S}_p^0(\mathcal{G}) \times \mathcal{S}_{p-1}^{-1}(\mathcal{G})$,
- $\mathbf{M} \in \mathbb{R}^{K,N}$ for $(\mathbf{v}, \eta) \mapsto \int_{\Gamma} \mathbf{v}(\mathbf{x}) \eta(\mathbf{x}) \, dS(\mathbf{x})$ on $\mathcal{S}_p^0(\mathcal{G}) \times \mathcal{S}_{p-1}^{-1}(\mathcal{G})$,

and right hand side vectors $\vec{\mathbf{k}}$ and $\vec{\mathbf{\phi}}$ containing the basis expansion coefficients of interpolants (\rightarrow “data approximation”, § 1.4.2.34) of φ and \mathbf{f} in $\mathcal{S}_{p-1}^{-1}(\mathcal{G})$ and $\mathcal{S}_p^0(\mathcal{G})$, respectively.

BEM for direct first-kind BIE for two-domain transmission problems requires only the assembly of the usual boundary element Galerkin matrices.

┘

1.6.3.2 Multi-Domain Transmission Problem

1.6.4 BEM for Wave Propagation

Bibliography

- [Aur+14] Markus Aurada, Michael Ebner, Michael Feischl, Samuel Ferraz-Leite, Thomas Führer, Petra Goldenits, Michael Karkulik, Markus Mayr, and Dirk Praetorius. “HILBERT—a MATLAB implementation of adaptive 2D-BEM”. In: *Numer. Algorithms* 67.1 (2014), pp. 1–32. DOI: [10.1007/s11075-013-9771-2](https://doi.org/10.1007/s11075-013-9771-2) (cit. on p. 82).
- [BS08] S. Brenner and R. Scott. *Mathematical theory of finite element methods*. 3rd. Texts in Applied Mathematics. Springer–Verlag, New York, 2008 (cit. on p. 57).
- [CHS18] X. Claeys, R. Hiptmair, and E. Spindler. “Second-Kind Boundary Integral Equations for Scattering at Composite Partly Impenetrable Objects”. In: *Comm. Computational Physics* 23.1 (2018), pp. 264–295. DOI: [doi:10.4208/cicp.OA-2016-0171](https://doi.org/10.4208/cicp.OA-2016-0171) (cit. on p. 18).
- [Gau18] W. Gautschi. *A Software Repository for Orthogonal Polynomials*. Philadelphia: SIAM, 2018 (cit. on p. 112).
- [Gau04] Walter Gautschi. *Orthogonal polynomials: computation and approximation*. Numerical Mathematics and Scientific Computation. Oxford University Press, New York, 2004, pp. x+301 (cit. on p. 112).
- [Hac92] W. Hackbusch. *Elliptic Differential Equations. Theory and Numerical Treatment*. Vol. 18. Springer Series in Computational Mathematics. Berlin: Springer, 1992 (cit. on p. 54).
- [Hac95] W. Hackbusch. *Integral equations. Theory and numerical treatment*. Vol. 120. International Series of Numerical Mathematics. Basel: Birkhäuser, 1995 (cit. on pp. 49, 51, 70).
- [Han02] M. Hanke-Bourgeois. *Grundlagen der Numerischen Mathematik und des Wissenschaftlichen Rechnens*. Mathematische Leitfäden. Stuttgart: B.G. Teubner, 2002 (cit. on p. 112).
- [HK12] R. Hiptmair and L. Kielhorn. *BETL – A generic boundary element template library*. Report 2012-36. Switzerland: SAM, ETH Zürich, 2012 (cit. on p. 18).
- [Mai08] M. Maischak. *The analytical computation of the Galerkin elements for the Laplace, Lamé and Helmholtz equation in 2D-BEM*. Preprint. Germany: IFAM, Universität Hannover, 2008 (cit. on pp. 104, 105, 107, 108).
- [McL00] W. McLean. *Strongly Elliptic Systems and Boundary Integral Equations*. Cambridge, UK: Cambridge University Press, 2000 (cit. on pp. 19, 42, 56, 59).
- [Rei18] M. T. Homer Reid. “Taylor–Duffy Method for Singular Tetrahedron-Product Integrals: Efficient Evaluation of Galerkin Integrals for VIE Solvers”. In: *IEEE Journal on Multiscale and Multiphysics Computational Techniques* 3 (2018), pp. 121–128. DOI: [10.1109/JMMCT.2018.2833873](https://doi.org/10.1109/JMMCT.2018.2833873) (cit. on p. 135).
- [RR04] Michael Renardy and Robert C. Rogers. *An introduction to partial differential equations*. Second. Vol. 13. Texts in Applied Mathematics. Springer-Verlag, New York, 2004, pp. xiv+434 (cit. on p. 42).
- [SS10] S. Sauter and C. Schwab. *Boundary Element Methods*. Vol. 39. Springer Series in Computational Mathematics. Heidelberg: Springer, 2010 (cit. on pp. 19, 24, 29, 35, 36, 47, 57, 59–61, 63, 66, 67, 69, 70, 74, 128, 136–140, 142, 147, 148).
- [Ste08] Olaf Steinbach. *Numerical approximation methods for elliptic boundary value problems*. New York: Springer, 2008, pp. xii+386. DOI: [10.1007/978-0-387-68805-3](https://doi.org/10.1007/978-0-387-68805-3) (cit. on pp. 69, 74, 77, 79, 80).
- [Str09] M. Struwe. *Analysis für Informatiker*. Lecture notes, ETH Zürich. 2009 (cit. on pp. 38, 40, 123).

- [Tre08] Lloyd N. Trefethen. “Is Gauss quadrature better than Clenshaw-Curtis?” In: *SIAM Rev.* 50.1 (2008), pp. 67–87. DOI: [10.1137/060659831](https://doi.org/10.1137/060659831) (cit. on p. 111).

Chapter 2

Local Low-Rank Compression of Non-Local Operators

Contents

2.1	Examples: Non-Local Operators	158
2.1.1	(Discretized) Integral Operators	159
2.1.2	Long-Range Interactions in Discrete Models	162
2.1.3	Kernel Collocation Matrices	166
2.2	Approximation of Kernel Collocation Matrices	167
2.2.1	Separable (= Low-Rank) Kernel Approximation	169
2.2.2	Error Estimates and Admissibility Condition for Singular Kernels	179
2.3	Clustering Techniques	190
2.3.1	Local Separable Approximation	190
2.3.2	Cluster Trees	198
2.3.3	Building Near- and Far-Field Blocks	207
2.3.4	Storing Block-Partitioned Kernel Collocation Matrix	213
2.3.5	Matrix \times Vector: Efficient Implementation	221
2.3.6	Panel Clustering	223
2.4	Hierarchical Matrices	227
2.4.1	Definition	227
2.4.2	Low-Rank Matrices: Algorithms	234
2.4.3	\mathcal{H} -Addition of Hierarchical Matrices	240
2.4.4	\mathcal{H} -Multiplication of Hierarchical Matrices [Bör21, Sect. 5.6]	241
2.4.5	Hierarchical LU-Decomposition	253
2.4.6	\mathcal{H}^2 -Matrices	259

§2.0.0.1 (The need for matrix compression for BEM) The boundary element Galerkin discretizations of boundary integral operators presented in Chapter 1 lead to **densely populated matrices** as explained in § 1.4.3.6.

We consider an (interior) boundary value problem on a bounded domain $\Omega \subset \mathbb{R}^d$, $d = 2, 3$, equipped with a “uniform” finite element mesh \mathcal{M} with a global meshwidth h .

We *assume* that a low-order **finite element Galerkin discretization** on \mathcal{M} provides a solution with an accuracy similar to that achieved by a low-order **boundary element Galerkin discretization** on $\mathcal{G} := \mathcal{M}|_{\Gamma}$, $\Gamma := \partial\Omega$.

	Finite element method (FEM)	↔	Boundary element method (BEM)
No. of degrees of freedom (unknowns):	h_M^{-d}	↔	h_G^{-d+1}
No. of nonzero entries of Galerkin matrices:	h_M^{-d}	↔	h_G^{-2d+2}

Hence, asymptotically for $h_M, h_G \rightarrow 0$ and $d = 3$, the BEM will require much more memory for storing the linear system of equations than FEM, $O(h_G^{-4})$ vs. $O(h_M^{-3})$. The lower number of unknowns for BEM becomes irrelevant!

Without matrix compression BEM cannot compete with FEM!

Further Reading on Local Low-Rank Compression

If you want to obtain information beyond what is covered in the course, please refer to

- ◆ M. BEBENDORF, *Hierarchical matrices: A means to efficiently solve elliptic boundary value problems*, Springer, 2008.
- ◆ W. HACKBUSCH, *Hierarchical Matrices*, Springer, 2015.
- ◆ S. BOERM, *Efficient Numerical Methods for Non-Local Operators: H2-Matrix Compression, Algorithms and Analysis*, EMS Publishing House, 2010.
- ◆ S. BOERM, *Numerical Methods for Non-Local Operators*, Lecture Notes Univ. Kiel, 2021.

2.1 Examples: Non-Local Operators

To understand the title of this section first Recall the notion of the support of a function:

Definition [NumPDE Def. 2.3.2.5]. Support of a function

The **support** of a function $f : D \mapsto \mathbb{R}$ is defined as

$$\text{supp}(f) := \overline{\{x \in D : f(x) \neq 0\}}.$$

Notion 2.1.0.1. Non-local operators on function spaces

An operator L defined on a space of functions on the domain $D \subset \mathbb{R}^d$ is said to be **non-local**, if for any two subsets $A, B \subset D$ there is a function f with $\text{supp}(f) \subset A$ such that $\text{supp}(L(f)) \cap B \neq \emptyset$.

EXAMPLE 2.1.0.2 (Differential operators are strictly local) It is instructive to recall a class of linear operators for $D = \mathbb{R}$ that are certainly *not non-local*. It is **differential operators** of form

$$Lf := \sum_{k=0}^m \alpha_k \frac{d^k f}{dx^k}, \quad \alpha_k \in \mathbb{R}, \quad (2.1.0.3)$$

acting on $C^m([a, b])$, $a < b$. In fact, these operators can be classified as strictly local in the sense that $\text{supp}(Lf) \subset \text{supp}(f)$.

Locality of operators can also be given meaning in finite dimensions.

Notion 2.1.0.4. Non-local operators on \mathbb{R}^N

A mapping $L: \mathbb{R}^N \rightarrow \mathbb{R}^N$, $N \in \mathbb{N}$, is said to be **non-local**, if for any two subsets $A, B \subset \{1, \dots, N\}$ there is a vector $\vec{\mu} \in \mathbb{R}^N$ with $j \notin A \Rightarrow (\vec{\mu})_j = 0$ such that $(L\vec{\mu})_\ell \neq 0$ for some $\ell \in B$.

► Linear non-local operators in \mathbb{R}^N can usually be represented only by *fully populated matrices*.

In mathematical models of physical phenomena, non-locality of operators is often caused by **long-range interactions** of spatial components.

2.1.1 (Discretized) Integral Operators

EXAMPLE 2.1.1.1 (Nyström-discretized boundary integral equations of the second kind) Given a bounded domain $\Omega \subset \mathbb{R}^d$, $d = 2, 3$, with C^1 -smooth boundary $\Gamma := \partial\Omega$, and Dirichlet data $g \in C^0(\Gamma)$, we want compute an (approximate) solution u of the **exterior Dirichlet problem** (EDP), cf. Section 1.1.7,

$$-\Delta u = 0 \quad \text{in } D := \mathbb{R}^d \setminus \bar{\Omega}, \quad u = g \quad \text{on } \Gamma := \partial\Omega, \quad (2.1.1.2a)$$

$$+ \text{decay condition } |u(x)| = O(\|x\|^{-1}) \quad \text{uniformly for } \|x\| \rightarrow \infty. \quad (2.1.1.2b)$$

This is a model for an electrostatic potential in the exterior of an object, cf. Rem. 1.1.6.8 and § 1.1.6.13.



We borrow an idea from potential theory and write u as a **double layer potential**:

$$u(x) = \int_{\Gamma} \mathbf{grad}_y G^\Delta(x, y) \cdot \mathbf{n}(y) \rho(y) \, dS(y), \quad x \in D, \quad (2.1.1.3)$$

where

- G^Δ is the fundamental solution (\rightarrow Def. 1.2.2.15) for the Laplacian,

$$G^\Delta(x, y) = \begin{cases} -\frac{1}{2\pi} \log \|x - y\| & , \text{ if } d = 2, \\ \frac{1}{4\pi} \frac{1}{\|x - y\|} & , \text{ if } d = 3, \end{cases} \quad x, y \in \mathbb{R}^d, \quad x \neq y, \quad (1.2.2.33)$$

- $\mathbf{n}: \Gamma \rightarrow \mathbb{R}^d$ is the unit normal vectorfield pointing from Ω to D ,
- and $\rho \in C^0(\Gamma)$ is an *unknown density*.

We can regard (2.1.1.3) as an “infinite superposition” of singular potentials generated by dipoles in direction $\mathbf{n}(x)$ located at $x \in \Gamma$,

$$y \in \mathbb{R}^d \setminus \{x\} \mapsto \mathbf{grad}_y G^\Delta(x, y) \cdot \mathbf{n}(y) = \begin{cases} -\frac{1}{2\pi} \frac{(x - y) \cdot \mathbf{n}(y)}{\|x - y\|^2} & , \text{ if } d = 2, \\ -\frac{1}{4\pi} \frac{(x - y) \cdot \mathbf{n}(y)}{\|x - y\|^3} & , \text{ if } d = 3, \end{cases} \quad (2.1.1.4)$$

each of which solves $-\Delta u = 0$ away from x and satisfies the decay condition (2.1.1.2b). A rigorous justification for the trial expression (2.1.1.3) can be drawn from Thm. 1.2.4.5.

We write $T_D : H^1(D) \rightarrow H^{\frac{1}{2}}(\partial\Omega)$ for the *exterior* Dirichlet trace operator (\rightarrow Section 1.3.1.1) and apply it to both sides of (2.1.1.3), that is, we let $x \in D$ tend to $x^* \in \Gamma$. Why not simply plug $x \in \Gamma$ into on both sides of (2.1.1.3). This may not be possible, because G^Δ and its derivatives are not defined for $x = y$. Yet, closer scrutiny reveals that the double layer integral still makes sense as an improper integral, the singularity of the integrand is integrable, the integral has a finite value though the integrand $= \infty$ at a point.

However, another unforeseen complication arises: The double layer formula (2.1.1.3) is clearly well-defined for $x \notin \Gamma$, but that extended function $u : \mathbb{R}^d \setminus \Gamma \rightarrow \mathbb{R}$ has a *jump across Γ* .

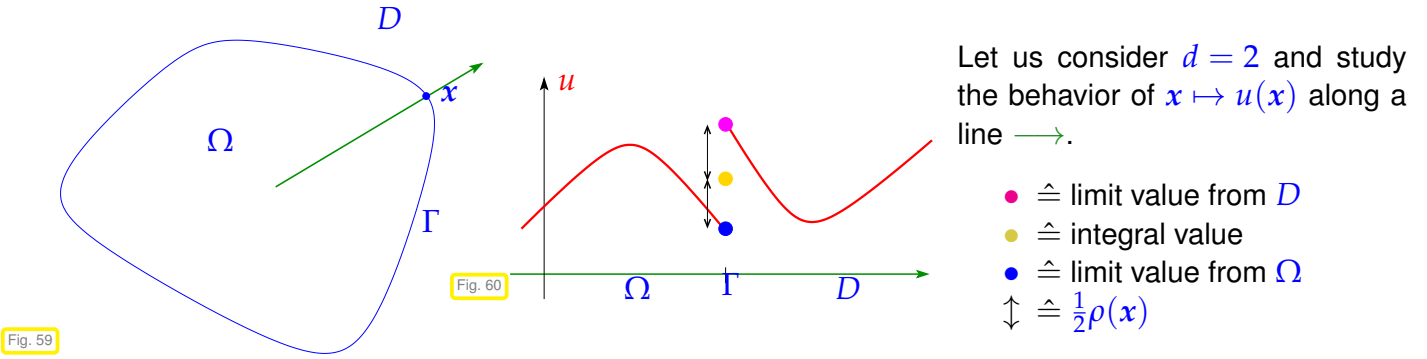


Fig. 59

Fig. 60

Tedious computations, some of which are discussed in Section 1.2.5.2, § 1.3.3.12, and § 1.3.4.10, reveal that

- the height of the jump of $x \rightarrow u(x)$ at $x \in \Gamma$ is equal to $\rho(x)$, and that
- the integral value is half-way between the limits from “inside” (Ω) and “outside” (D):

So for $x \in \Gamma$:

$$\underbrace{\lim_{x' \in D \rightarrow x^*} u(x')}_{\text{outside limit}} - \frac{1}{2}\rho(x) = \int_{\Gamma} \mathbf{grad}_y G^\Delta(x, y) \cdot \mathbf{n}(y)\rho(y) dS(y) = \underbrace{\lim_{x' \in \Omega \rightarrow x^*} u(x')}_{\text{inside limit}} + \frac{1}{2}\rho(x). \quad (2.1.1.5)$$

In other symbols, this jump relation gives for $x \in \Gamma$

$$(T_D u)(x) = \frac{1}{2}\rho(x) + \int_{\Gamma} \mathbf{grad}_y G^\Delta(x, y) \cdot \mathbf{n}(y)\rho(y) dS(y). \quad (2.1.1.6)$$

The reader may simply accept this formula. Taking it for granted and taking into account the Dirichlet boundary condition $u = g$ on Γ we end up with the **boundary integral equation** (BIE)

$$\boxed{\frac{1}{2}\rho(x) + \int_{\Gamma} \mathbf{grad}_y G^\Delta(x, y) \cdot \mathbf{n}(y)\rho(y) dS(y) = g(x) \quad x \in \Gamma}, \quad (2.1.1.7)$$

for the unknown density $\rho \in C^0(\Gamma)$. The behavior of the integrand for $y \rightarrow x$ is discussed in § 1.3.4.10, but it need not worry us any further, because next we *regularize* the integral. We start from the so-called solid-angle formula¹

$$\int_{\Gamma} \mathbf{grad}_y G^\Delta(x, y) \cdot \mathbf{n}(y) dS(y) = \begin{cases} -1 & \text{for } x \in \Omega, \\ -\frac{1}{2} & \text{for } x \in \Gamma, \\ 0 & \text{for } x \in D. \end{cases} \quad (2.1.1.8)$$

This converts (2.1.1.7) into the equivalent BIE

$$\boxed{\int_{\Gamma} k(x, y)(\rho(y) - \rho(x)) dS(y) = g(x) \quad x \in \Gamma}, \quad (2.1.1.9)$$

¹The solid-angle formula can be proved using the techniques of Section 1.2.2.3.

with **kernel** (function)

$$k(x, y) := \mathbf{grad}_y G^\Delta(x, y) \cdot \mathbf{n}(y) = \begin{cases} -\frac{1}{2\pi} \frac{(x - y) \cdot \mathbf{n}(y)}{\|x - y\|^2} & , \text{ if } d = 2, \\ -\frac{1}{4\pi} \frac{(x - y) \cdot \mathbf{n}(y)}{\|x - y\|^3} & , \text{ if } d = 3, \end{cases} \quad x, y \in \mathbb{R}^d, x \neq y. \tag{2.1.1.10}$$

If ρ is Lipschitz continuous, the singularity of the integrand for $y = x$ will cancel even for $d = 3$, a fact that accounts for the term “regularization”.

The **Nyström discretization** of (2.1.1.9) consists of two steps

- 1 We approximate the integral by means of an n -point **quadrature formula** Def. 1.4.3.41, $n \in \mathbb{N}$:

$$\int_\Gamma \varphi(y) dS(y) \approx \sum_{j=1}^n w_j \varphi(c_j) \tag{2.1.1.11}$$

with weights $w_j \in \mathbb{R}$ and nodes $c_j \in \Gamma, j = 1, \dots, n$. This results in the approximate boundary integral equation

$$\sum_{j=1}^n w_j k(x, c_j) (\rho(c_j) - \rho(x)) = g(x), \quad x \in \Gamma. \tag{2.1.1.12}$$

- 2 We apply **collocation** to (2.1.1.12). We demand that the equation only holds in the n quadrature nodes c_i and arrive at

$$\sum_{\substack{j=1 \\ j \neq i}}^n w_j k(c_i, c_j) (\rho(c_j) - \rho(c_i)) = g(c_i), \quad i = 1, \dots, n. \tag{2.1.1.13}$$

This is an $n \times n$ linear system of equations (LSE) for the unknown point values $\rho(x_i), i = 1, \dots, n$. Collecting the unknowns in the vector $\vec{\rho} := (\rho(c_i))_{i=1}^n \in \mathbb{R}^n$ it can be written as

$$(\mathbf{D} + \mathbf{M})\vec{\rho} = \vec{\gamma} := (g(c_j))_{j=1}^n, \tag{2.1.1.14}$$

with the matrices

$$\mathbf{M} \in \mathbb{R}^{n,n}, \quad (\mathbf{M})_{ij} := \begin{cases} w_j k(c_i, c_j) & \text{for } i \neq j, \\ 0 & \text{else,} \end{cases} \quad i, j \in \{1, \dots, n\}, \tag{2.1.1.15}$$

$$\mathbf{D} \in \mathbb{R}^{n,n}, \quad (\mathbf{D})_{ij} := \begin{cases} -\sum_{\substack{j=1 \\ j \neq i}}^n w_j k(c_i, c_j) & \text{for } i = j, \\ 0 & \text{else,} \end{cases} \quad i, j \in \{1, \dots, n\}. \tag{2.1.1.16}$$

┘

The key constituent part of the boundary integral equation (2.1.1.9) is an integral operator on the left-hand side. In general an **integral operator** on a space $X(D)$ of functions $D \rightarrow \mathbb{R}, D \subset \mathbb{R}^d$ a domain of integration (which can also be a lower-dimensional manifold like a boundary), is a linear mapping $L : X(D) \rightarrow Y(D), Y(D)$ another function space, defined by

$$(Lf)(x) := \int_D k(x, y) f(y) dy, \quad x \in D, \quad f \in X(D), \tag{2.1.1.17}$$

with a **kernel** function $k : D \times D \rightarrow \mathbb{R}$. If the support of k is *global*, then \mathbf{T} will be a archetypical non-local operator according to Notion 2.1.0.1.

EXAMPLE 2.1.1.18 (Boundary integral equations related to scalar 2nd-order elliptic BVPs) This topic is presented in Chapter 1. That part of the lecture derives and discussed a host of non-local integral operators, which occur in integral equation reformulations of 2nd-order scalar elliptic boundary value problems with constant coefficients. Important specimens of those non-local integral operators are

- the Newton potential (\rightarrow Def. 1.2.3.2)

$$(\mathbf{N}\rho)(x) := \int_{\Omega} G^{\Delta}(x, \mathbf{y}) \rho(\mathbf{y}) d\mathbf{y}, \quad \rho \in \tilde{H}^{-1}(\mathbb{R}^d), \quad (2.1.1.19)$$

with the fundamental solution (\rightarrow Def. 1.2.2.15) for the Laplacian

$$G^{\Delta}(x, \mathbf{y}) = \begin{cases} -\frac{1}{2\pi} \log\|\mathbf{x} - \mathbf{y}\| & , \text{ if } d = 2, \\ \frac{1}{4\pi} \frac{1}{\|\mathbf{x} - \mathbf{y}\|} & , \text{ if } d = 3, \end{cases} \quad \mathbf{x} \neq \mathbf{y}, \quad (1.2.2.33)$$

whose support is $\mathbb{R}^d \times \mathbb{R}^d$ and obviously unbounded.

- and the fundamental boundary integral operators of Def. 1.3.4.1, for instance the single layer boundary integral operator on $\Gamma := \partial\Omega$ for the Laplacian $-\Delta$ (\rightarrow § 1.3.4.8)

$$(\mathbf{V}\phi)(x) = \int_{\Gamma} G^{\Delta}(x, \mathbf{y}) \phi(\mathbf{y}) dS(\mathbf{y}), \quad \phi \in H^{-\frac{1}{2}}(\partial\Omega). \quad (1.3.4.9)$$

The Galerkin discretization (\rightarrow Section 1.4.1) of an integral operator of the form (2.1.1.17) based on a basis $\{b_N^1, \dots, b_N^N\} \subset X(\Omega)$ leads to Galerkin matrices $\mathbf{T} \in \mathbb{R}^{N,N}$ with entries

$$\mathbf{T} = \left[\int_D \int_D k(\mathbf{x}, \mathbf{y}) b_N^j(\mathbf{y}) b_N^i(\mathbf{x}) d\mathbf{y} d\mathbf{x} \right]_{i,j=1}^N. \quad (2.1.1.20)$$

If \mathbf{T} is non-local then the matrix \mathbf{T} will be **densely populated** even if the basis functions are locally supported, recall § 1.4.3.6. ┘

2.1.2 Long-Range Interactions in Discrete Models

In computational physics interactions are classified as short-range, if for each component of a system (star, particle, molecule, etc.) only the interaction with a fixed small number of “neighbors” matters.

§2.1.2.1 (Gravitational forces in astrophysics)

The goal is to simulate the dynamics of the n stars in a galaxy; usually $n \approx 10^9$. This can be done by treating the stars as “point masses” and solving Newton’s equations of motion by numerical integration, which entails computing the gravitational attraction between every of the 10^{18} pairs of stars.

Let $\mathbf{x}^i = [x_1^i, \dots, x_d^i]^\top \in \mathbb{R}^d, i = 1, \dots, n, d = 2, 3$, stand for the position of the i -th star with mass $m_i > 0$. Then the force on the j -th star is

$$\mathbf{f}^j = \frac{g}{4\pi} \sum_{\substack{i=1 \\ i \neq j}}^n \frac{\mathbf{x}^i - \mathbf{x}^j}{\|\mathbf{x}^j - \mathbf{x}^i\|^3} m_i m_j, \quad j = 1, \dots, n, \tag{2.1.2.2}$$



where g is the gravitational constant, $g = 6.674 \cdot 10^{-11} \text{m}^3 \text{kg}^{-1} \text{s}^{-2}$.

In terms of vector components (2.1.2.2) reads

$$f_\ell^j = \frac{g}{4\pi} \sum_{\substack{i=1 \\ i \neq j}}^n \frac{x_\ell^i - x_\ell^j}{\|\mathbf{x}^j - \mathbf{x}^i\|^3} m_i m_j, \quad \ell = 1, 2, 3, \quad j = 1, \dots, n. \tag{2.1.2.3}$$

Collecting all force components in long vectors $\mathbf{F}_\ell := [f_\ell^1, \dots, f_\ell^n]^\top$ permits us to express (2.1.2.3) as **matrix×vector-product**: for $\ell = 1, 2, 3$

$$\mathbf{F}_\ell := \begin{bmatrix} f_\ell^1 \\ \vdots \\ f_\ell^n \end{bmatrix} = \frac{g}{4\pi} \begin{bmatrix} m_1 & & & \\ & \ddots & & \\ & & \ddots & \\ & & & m_n \end{bmatrix} \mathbf{M}_\ell \begin{bmatrix} m_1 \\ \vdots \\ m_n \end{bmatrix}, \tag{2.1.2.4}$$

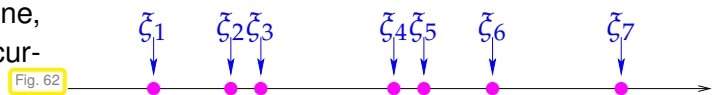
with $(\mathbf{M}_\ell)_{j,i} = \begin{cases} \frac{x_\ell^i - x_\ell^j}{\|\mathbf{x}^j - \mathbf{x}^i\|^3} & \text{for } i \neq j, \\ 0 & \text{for } i = j, \end{cases} \quad i, j = 1, \dots, n.$

Thus the complete vector of force components \mathbf{F}_ℓ in every timestep can be obtained from multiplying the vector of masses with the matrix \mathbf{M}_ℓ . However, the evaluation of the force components \mathbf{F}_ℓ for many timesteps is way beyond the capabilities of even the largest supercomputers, because \mathbf{M} is a fully populated matrix with $\approx 10^{18}$ entries!

Fortunately, the matrices \mathbf{M}_ℓ possess a very special structure, they are so-called kernel collocation matrices (\rightarrow Def. 2.1.3.1 below) associated with a singular, asymptotically smooth kernel function (\rightarrow Rem. 2.2.2.1 below). In this chapter you will learn how to realize an approximate matrix×vector product with a computational effort way smaller than the number of non-zero matrix entries. \lrcorner

§2.1.2.5 (Forces on parallel wires)

We consider n long straight parallel wires in a plane, with the j -th wire at location $\zeta_j \in \mathbb{R}$ carrying the current $c_j \in \mathbb{R}$.



The (scaled) magnetic force on the j -th wire is

$$f^j = \sum_{\substack{i=1 \\ i \neq j}}^n \frac{c_i c_j}{|\xi_i - \xi_j|}, \quad j = 1, \dots, n. \tag{2.1.2.6}$$

Again, we can collect all forces in one long vector $F := [f^1, \dots, f^n]^\top$ and rewrite (2.1.2.6) as a **matrix × vector-product**:

$$F = \begin{bmatrix} c_1 & & & \\ & \ddots & & \\ & & \ddots & \\ & & & c_n \end{bmatrix} M \begin{bmatrix} c_1 \\ \vdots \\ \vdots \\ c_n \end{bmatrix}, \quad \text{with } (M)_{i,j} = \begin{cases} \frac{1}{|\xi_j - \xi_i|} & \text{for } i \neq j, \\ 0 & \text{for } i = j \end{cases}, \quad i, j = 1, \dots, n. \tag{2.1.2.7}$$

In a sense, comparing (2.1.2.7) and (2.1.2.4), the task to compute the magnetic force on the wires can be regarded as a one-dimensional counterpart of the challenge to compute gravitational forces in galaxies. ◻

§2.1.2.8 (A glimpse of clustering approximation) We continue § 2.1.2.1 and describe a heuristic for the efficient **approximate** evaluation of gravitational interactions. We assume $x^i \in [0, 1]^d$ for all star positions $x^i \in \mathbb{R}^d, d = 2, 3$.

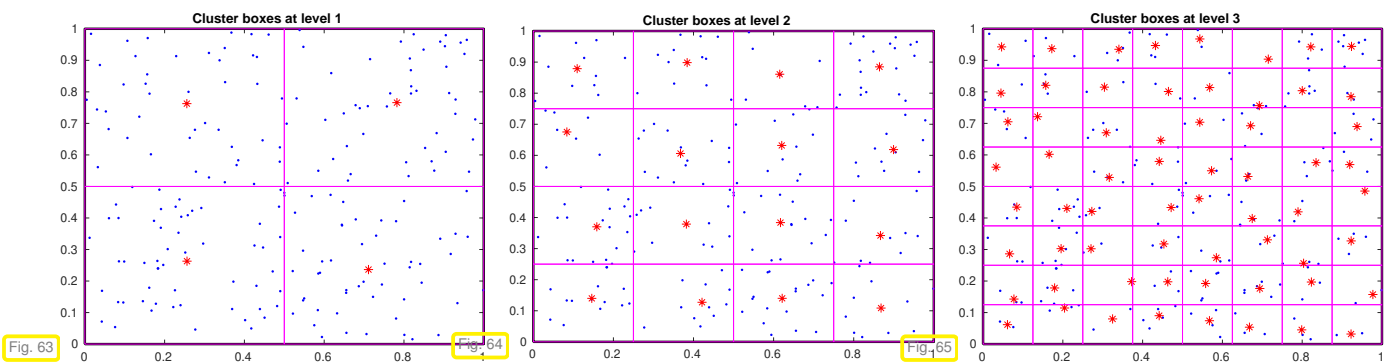


To evaluate the force f^j replace “remote” clusters of stars with a single “equivalent” massive star in the center of gravity.



Define **clusters** through **quadtree** ($d = 2$)/ **octree** ($d = 3$) decomposition of the spatial box containing the galaxy.
(Choose depth $L \in \mathbb{N}$ of octree such that a leaf contains a single star at most)

Example in 2D ($d = 2$): **quadtree** decomposition of $[0, 1]^2, \cdot \hat{=} \text{stars}, * \hat{=} \text{equivalent stars}$.



The clusters on level $\ell \in \{0, \dots, L\}$ are $(\alpha = (\alpha_1, \alpha_2))$

$$\left\{ i \in \{1, \dots, n\} : x^i \in C_\alpha^\ell := h_\ell \cdot ([\alpha_1, \alpha_1 + 1] \times [\alpha_2, \alpha_2 + 1]), \alpha_i \in \{0, \dots, 2^\ell - 1\} \right\},$$

$h_\ell := 2^{-\ell}$. Each cluster of stars is uniquely characterized by its **bounding box** C_α^ℓ .



In the case of a given threshold for the approximation error it is clear that lumping together stars will introduce smaller errors, if those stars are farther away from x^j :

Heuristics: The larger the *distance* of a cluster from x^j , the larger can be the *size* of the bounding box of the cluster.

In quantitative terms this can be expressed by requiring that the **admissibility condition**

$$\text{dist}(C_\alpha^\ell; x^j) \geq \eta \text{diam}(C_\alpha^\ell), \quad \alpha \in \{0, \dots, 2^\ell - 1\}^2, \quad \eta > 0, \quad (2.1.2.9)$$

where $\text{dist}(C_\alpha^\ell; x^j) := \min\{\|z - x^j\| : z \in C_\alpha^\ell\}$, $\text{diam}(C_\alpha^\ell) = 2^{-\ell}$,

has to be satisfied for the cluster C_α^ℓ , if its stars are to be replaced with a single equivalent star. Here, $\eta > 0$ is a control parameter, for whom suitable values have to be found by numerical tests.

Assumption 2.1.2.10. Uniform distribution

The stars are uniformly distributed in $]0, 1[^2$ (Constant asymptotic density of stars).

❶ The algorithm starts with a **Preprocessing step**: For each cluster (stars in a box of the quadtree/octree decomposition) with associated bounding box C_α^ℓ determine the center of gravity c_α^ℓ and total mass m_α^ℓ , that is, find an “equivalent star”.



$$\text{cost} = O(n \log n), \quad \text{for no. } n \text{ of stars} \rightarrow \infty$$

Then we want to compute the force on the star located at x^j ,

$$f_\ell^j = \frac{g}{4\pi} \sum_{\substack{i=1 \\ i \neq j}}^n \frac{x_\ell^i - x_\ell^j}{\|x^j - x^i\|^3} m_i m_j, \quad \ell = 1, 2, 3, \quad j = 1, \dots, n. \quad (2.1.2.3)$$

❷ Find a set \mathcal{A} of bounding boxes C_α^ℓ (corresponding to star clusters) such that (for $d = 2$)

(I) The bounding boxes cover $]0, 1[^2$: $]0, 1[^2 \subset \bigcup_{C \in \mathcal{A}} C$.

(II) Different bounding boxes are disjoint: $C, C' \in \mathcal{A} \Rightarrow C \cap C' = \emptyset$.

(III) $C \in \mathcal{A}$ **either** satisfies the admissibility condition (2.1.2.9) with respect to x^j **or** belongs to the finest level L of the quadtree (C is leaf).

(IV) If $C \in \mathcal{A}$, then no bounding box/cluster on a lower level ℓ that contains C is in \mathcal{A} .

The requirements of Item (I) and Item (II) mean that the bounding boxes in \mathcal{A} form a **partition** of $]0, 1[^2$, whereas Item (IV) ensures that the admissible bounding boxes in \mathcal{A} are as large as possible.

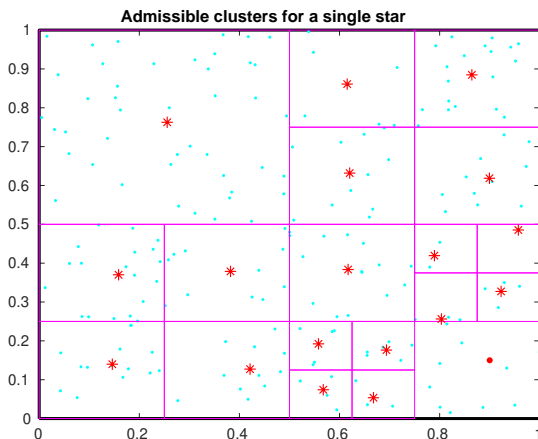


Fig. 66

Example in 2D ($d = 2$):

“Stars” randomly and uniformly distributed in $]0, 1[^2$.

Admissibility condition (2.1.2.9) with $\eta \approx 0.6$

* $\hat{=}$ “equivalent stars”

◁ Admissible clusters w.r.t. star •, level $\ell \geq 3$.

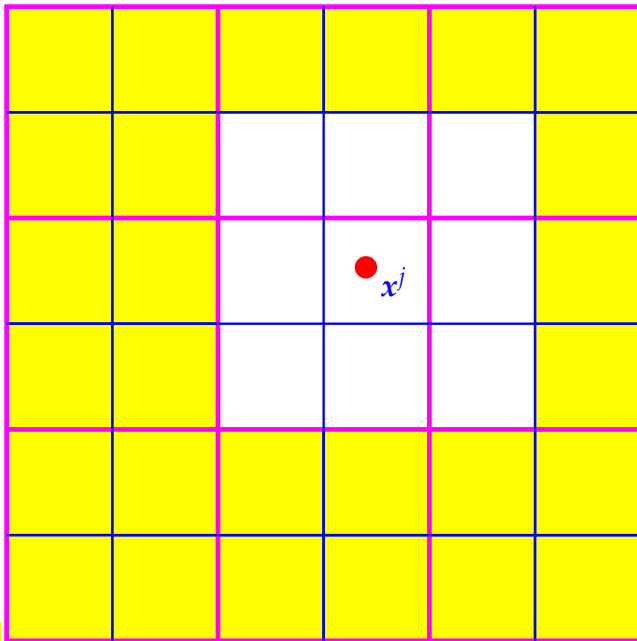


Fig. 67

◁ Star at x^j is surrounded by at most 9 inadmissible clusters on level ℓ (magenta lines)

At most 9 clusters on level $\ell + 1$ (blue lines) will be inadmissible.

■ $\hat{=}$ admissible clusters on level $\ell + 1$.

▶ There are at most 36 relevant admissible clusters on level $\ell + 1$.

The number of contributing clusters on each level is bounded: $O(1)$ for $n \rightarrow \infty$, of course, dependent on η .

③ Approximate f_ℓ^j as

$$f_\ell^j \approx \frac{gm_j}{4\pi} \sum_{C \in \mathcal{A}} m_C \frac{x_\ell^j - c_\ell^C}{\|x^j - c^C\|^3}, \tag{2.1.2.11}$$

where c^C and m_C are the position and mass of the equivalent star for a cluster/bounding box.

▶ cost = $O(\log n)$ for computing f^j in the limit $n \rightarrow \infty$

However, except for choosing different parameters $\eta > 0$ in the admissibility condition (2.1.2.9), there is *no way to control the accuracy* of the approximation inherent in this approach. \lrcorner

2.1.3 Kernel Collocation Matrices

As a model problem for the treatment of non-local operators we study the approximation of densely populated matrices of a particular form.

Definition 2.1.3.1. Kernel collocation matrix

We are given

- two bounded domains $D_x, D_y \subset \mathbb{R}^d, d \in \mathbb{N}$,
- a **kernel function** $G : D_x \times D_y \rightarrow \mathbb{R}$,
- and **collocation points** $x^i \in D_x, y^j \in D_y$. The matrix $\mathbf{M} \in \mathbb{R}^{n,m}$ with entries

$$(\mathbf{M})_{i,j} := G(x^i, y^j), \quad i \in \{1, \dots, n\} \quad j \in \{1, \dots, m\}, \tag{2.1.3.2}$$

is a **kernel collocation matrix**.

📎 Notation: If $d = 1$, we write $\xi_i, i = 1, \dots, n$, and $\eta_j, j = 1, \dots, m$, for the collocation points and assume that they are *sorted*:

$$\xi_1 < \xi_2 < \dots < \xi_n, \quad \xi_i \in D_x \subset \mathbb{R}, \quad \eta_1 < \eta_2 < \dots < \eta_m, \quad \eta_j \in D_y \subset \mathbb{R}, \quad m, n \in \mathbb{N}.$$

EXAMPLE 2.1.3.3 (Globally supported singular kernel functions) We are mainly interested in globally supported kernels $(x, y) \mapsto G(x, y), x \in D_x, y \in D_y$ that are *singular* for $x = y$.

Examples are kernels related to **fundamental solutions** (→ Def. 1.2.2.15) of scalar linear partial differential operators with constant coefficients.

$$G(x, y) = \begin{cases} \log\|x - y\| & , \text{ if } x \neq y, \\ 0 & \text{ else,} \end{cases} \quad \text{or} \quad G(x, y) = \begin{cases} \frac{1}{\|x-y\|} & , \text{ if } x \neq y, \\ 0 & \text{ else.} \end{cases} \quad (2.1.3.4)$$

Note that these kernel functions are C^∞ -smooth even **analytic** (Def. 1.4.3.67) in every variable on $D_x \times D_y$, provided that $\overline{D_x} \cap \overline{D_y} = \emptyset$.

We also saw another relevant class of kernel functions in § 2.1.2.1, see (2.1.2.3):

$$G(x, y) \sim \frac{x_\ell - y_\ell}{\|x - y\|^3}, \quad x, y \in \mathbb{R}^d, \quad d = 2, 3, \quad x \neq y. \quad (2.1.3.5)$$

┘

Review question(s) 2.1.3.6 (Non-local operators)

(Q2.1.3.6.A) Given a sequence $(c_j)_{j \in \mathbb{Z}} \in \ell^1(\mathbb{Z})$, when is the **convolution operator**

$$L : \ell^\infty \rightarrow \ell^\infty, \quad (L(x_k))_k := \sum_{j \in \mathbb{Z}} c_{k-j} x_j, \quad k \in \mathbb{Z},$$

a non-local operator?

(Q2.1.3.6.B) [Gauss kernel] One of the most important functions in mathematical modeling is the Gaussian (normal distribution)

$$t \mapsto f(t) := \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{t}{\sigma}\right)^2}, \quad t \in \mathbb{R}.$$

Do kernel collocation matrices based on the kernel function

$$G : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}, \quad G(x, y) = f(\|x - y\|_2), \quad x, y \in \mathbb{R}^d,$$

have any special properties?

△

2.2 Approximation of Kernel Collocation Matrices

§2.2.0.1 (Data-sparse approximate representation) Obviously, kernel collocation matrices $M \in \mathbb{R}^{n,m}$ (→ Def. 2.1.3.1) based on kernel functions like those in (2.1.3.4) are **densely populated**.

► $O(nm)$ memory/effort for straightforward storage/initialization, for $n, m \rightarrow \infty$.

$O(nm)$ computational cost for $M \times$ vector

Goal: Given a tolerance $\epsilon > 0$, find an **approximation** $\widetilde{M} \in \mathbb{R}^{n,m}$ of M such that we can guarantee a prescribed accuracy

$$\|M - \widetilde{M}\| \leq \epsilon \quad [\|\cdot\| \text{ some matrix norm}], \quad (2.2.0.2)$$

with both

cost of storage/initialization of \widetilde{M}

cost($\widetilde{M} \times$ vector)

$$= O((m+n) \log^q(m+n) |\log^p \epsilon|) \quad \text{for } m, n \rightarrow \infty, \epsilon \rightarrow 0,$$

for some exponents $p, q \in \mathbb{N}_0$.

Alluding to the efficiency of algorithms for large sparse matrices, data structures with which we can achieve the above goal are called **data sparse**. ┘

Remark 2.2.0.3 (Families of sparse matrices) The name is telling: families of **sparse matrices** with a fixed maximal number of non-zero entries per column or row, as they arise, for instance, from the finite-element discretization of boundary-value problems for partial differential equations on shape-regular families of meshes, allow data-sparse representation without any approximation, see [NumCSE Section 2.7.1]. ┘

§2.2.0.4 (Recalled: Matrix norms [NumCSE § 1.5.5.3]) The $\|\cdot\|$ in (2.2.0.2) denotes a matrix norm. Remember that matrix norms can be **induced by vector norms** as norms of the linear mapping described by the matrix. If $\|\cdot\|_1$ is a norm on \mathbb{R}^m and $\|\cdot\|_2$ a norm on \mathbb{R}^n , then the associated matrix norm $\|\cdot\|$ is [NumCSE Def. 1.5.5.10]

$$\mathbf{M} \in \mathbb{R}^{n,m}: \quad \|\mathbf{M}\| := \sup_{\vec{\xi} \in \mathbb{R}^m \setminus \{0\}} \frac{\|\mathbf{M}\vec{\xi}\|_1}{\|\vec{\xi}\|_2}. \quad (2.2.0.5)$$

📎 Notation: Matrix norms for *quadratic matrices* associated with standard vector norms:

$$\|\mathbf{x}\|_2 \rightarrow \|\mathbf{M}\|_2, \quad \|\mathbf{x}\|_1 \rightarrow \|\mathbf{M}\|_1, \quad \|\mathbf{x}\|_\infty \rightarrow \|\mathbf{M}\|_\infty$$

For the matrix norms $\|\cdot\|_1$ and $\|\cdot\|_2$ there are simple formulas [NumCSE Ex. 1.5.5.12]:

$$\text{➤ matrix norm } \leftrightarrow \|\cdot\|_\infty = \text{row sum norm} \quad \|\mathbf{M}\|_\infty := \max_{i=1,\dots,n} \sum_{j=1}^m |(\mathbf{M})_{ij}|, \quad (2.2.0.6)$$

$$\text{➤ matrix norm } \leftrightarrow \|\cdot\|_1 = \text{column sum norm} \quad \|\mathbf{M}\|_1 := \max_{j=1,\dots,m} \sum_{i=1}^n |(\mathbf{M})_{ij}|. \quad (2.2.0.7)$$

There is no corresponding simple formula for the Euclidean matrix norm $\|\cdot\|_2$, see [NumCSE Lemma 1.5.5.15], [NumCSE Cor. 1.5.5.16].

Not induced by a vector norm is the **Frobeniusnorm** [NumCSE Def. 3.4.4.17]

$$\|\mathbf{M}\|_F^2 := \sum_{i=1}^n \sum_{j=1}^m (\mathbf{M})_{ij}^2, \quad \mathbf{M} \in \mathbb{R}^{n,m}. \quad (2.2.0.8)$$

Note that $\|\cdot\|_F$ provides an upper bound for $\|\cdot\|_2$. ┘

2.2.1 Separable (= Low-Rank) Kernel Approximation

§2.2.1.1 (Low-rank matrices) There is an important class of fully populated matrices for which **exact** data-sparse representation is possible. These are matrices whose rank is much lower than their maximal rank.

Definition 2.2.1.2. Rank of a matrix [NS02, Sect. 2.4]

The **rank** of matrix $\mathbf{M} \in \mathbb{R}^{n,m}$ is the dimension of its image space:

$$\text{rank}(\mathbf{M}) := \dim \mathcal{R}(\mathbf{M}).$$

We have $\text{rank}(\mathbf{M}) \leq \min\{m, n\}$ for every $\mathbf{M} \in \mathbb{R}^{n,m}$. A matrix is called **low-rank**, if $\text{rank}(\mathbf{M}) \ll \min\{m, n\}$.

Lemma 2.2.1.3. Representation of low-rank matrices

If $\mathbf{M} \in \mathbb{R}^{n,m}$ satisfies $\text{rank}(\mathbf{M}) = q$, then there are matrices $\mathbf{U} \in \mathbb{R}^{n,q}$ and $\mathbf{V} \in \mathbb{R}^{m,q}$ such that $\mathbf{M} = \mathbf{UV}^\top$.

Proof. The lemma is an immediate consequence of the singular value decomposition theorem [NumCSE Thm. 3.4.1.1] and the fact that $\text{rank}(\mathbf{M})$ is equal to the number of non-zero singular values. \square

The message of Lemma 2.2.1.3 can be visualized as follows:

$$\left[\begin{array}{c} \mathbf{M} \end{array} \right] = \left[\begin{array}{c} \mathbf{U} \end{array} \right] \left[\begin{array}{c} \mathbf{V}^\top \end{array} \right].$$

► $\text{storage}(\mathbf{M}) = O(q(n + m))$ for $n, m \rightarrow \infty$ (2.2.1.4)

Recall from [NumCSE Ex. 1.4.3.1] the possibilities offered by associative multiplication:

$$\text{rank}(\mathbf{M}) = q \implies \text{Cost}(\mathbf{M} \times \text{vector}) = O(q(n + m)) \text{ for } n, m \rightarrow \infty. \tag{2.2.1.5}$$

$$\left[\begin{array}{c} \mathbf{M} \end{array} \right] \left[\begin{array}{c} \vec{\zeta} \end{array} \right] = \left[\begin{array}{c} \mathbf{U} \end{array} \right] \underbrace{\left(\left[\begin{array}{c} \mathbf{V}^\top \end{array} \right] \left[\begin{array}{c} \vec{\zeta} \end{array} \right] \right)}_{q \text{ scalar products of length } m}, \quad \vec{\zeta} \in \mathbb{R}^m. \tag{2.2.1.6}$$

Of course, the promise of (2.2.1.5) can only be realized, if the low-rank matrix \mathbf{M} is available in factorized form $\mathbf{M} = \mathbf{UV}^\top$ according to Lemma 2.2.1.3. \lrcorner

§2.2.1.7 (Separable kernel functions) Let us consider a kernel collocation matrix $\mathbf{M} \in \mathbb{R}^{n,m}$ (\rightarrow Def. 2.1.3.1) based on a **separable** kernel function $G(x, y)$, that is, a kernel function that can be written as a product of a function of the first argument x and another function of the second argument y :

$$G : D_x \times D_y \rightarrow \mathbb{R} \quad , \quad G(x, y) := g(x)h(y) \quad \text{with} \quad \begin{array}{l} g : D_x \rightarrow \mathbb{R} \\ h : D_y \rightarrow \mathbb{R} \end{array} \tag{2.2.1.8}$$

Writing $\mathbf{x}^i \in D_x$, $\mathbf{y}^j \in D_y$, $i \in \{1, \dots, n\}$, $j \in \{1, \dots, m\}$, for the collocation points and using the notations of Def. 2.1.3.1 we observe that \mathbf{M} can be written as the **tensor-product** [NumCSE Section 1.3.1] of two vectors of length n and m , respectively:

$$\mathbf{M} = [G(\mathbf{x}^i, \mathbf{y}^j)]_{i,j} = [g(\mathbf{x}^i)]_{i=1,\dots,n} [h(\mathbf{y}^j)]_{j=1,\dots,m}^\top \quad \blacktriangleright \quad \text{rank}(\mathbf{M}) = 1. \tag{2.2.1.9}$$

Hence, \mathbf{M} is a rank-1 matrix whose factorized form according to Lemma 2.2.1.3 is immediately available: According to § 2.2.1.1, \mathbf{M} needs $O(m + n)$ storage and the evaluation of $\mathbf{M}\vec{\zeta}$, $\vec{\zeta} \in \mathbb{R}^m$ incurs computational cost $O(m + n)$ for $m, n \rightarrow \infty$. \lrcorner

§2.2.1.10 (More general separable kernel functions) The considerations of § 2.2.1.7 can be generalized to so-called **rank- q separable** kernel functions, which are the sum of q separable functions, $q \in \mathbb{N}$:

$$G : D_x \times D_y \rightarrow \mathbb{R} \quad , \quad G(x, y) := \sum_{\ell=1}^q g_\ell(x)h_\ell(y) \quad , \quad \begin{matrix} g_\ell : D_x \rightarrow \mathbb{R} \\ h_\ell : D_y \rightarrow \mathbb{R} \end{matrix} \quad , \quad \ell = 1, \dots, q. \quad (2.2.1.11)$$

In this case we end up with a rank- q kernel collocation matrix (based on collocation points $x^i \in D_x$, $y^j \in D_y$, $i \in \{1, \dots, n\}$, $j \in \{1, \dots, m\}$)

$$\mathbf{M} = [G(x^i, y^j)]_{i,j} = \begin{bmatrix} g_1(x^1) \\ \vdots \\ g_1(x^n) \end{bmatrix} [h_1(y^1), \dots, h_1(y^m)] + \dots + \begin{bmatrix} g_q(x^1) \\ \vdots \\ g_q(x^n) \end{bmatrix} [h_q(y^1), \dots, h_q(y^m)] \quad ,$$

whose factorized form according to Lemma 2.2.1.3 is

$$\mathbf{M} = \mathbf{A}\mathbf{B}^\top \quad , \quad \begin{matrix} \mathbf{A} \in \mathbb{R}^{n,q} \quad , \quad (\mathbf{A})_{i,\ell} := g_\ell(x^i) \quad , \quad i = 1, \dots, n \quad , \\ \mathbf{B} \in \mathbb{R}^{m,q} \quad , \quad (\mathbf{B})_{j,\ell} := h_\ell(y^j) \quad , \quad j = 1, \dots, m \quad , \end{matrix} \quad \ell = 1, \dots, q. \quad (2.2.1.12)$$

Evidently, the asymptotic computational effort for computing $\mathbf{M}\vec{\zeta}$, $\vec{\zeta} \in \mathbb{R}^m$, is $O(q(m+n))$ for $m, n \rightarrow \infty$, q fixed. ┘



Idea: Obtain a data-sparse approximation of a kernel collocation matrix $\mathbf{M} = [G(x^i, y^j)]_{\substack{i=1,\dots,n \\ j=1,\dots,m}}$ (\rightarrow Def. 2.1.3.1) by a **separable approximation** of G :

$$G(x, y) \approx \tilde{G}(x, y) := \sum_{\ell=1}^q g_\ell(x)h_\ell(y) \quad \blacktriangleright \quad \tilde{\mathbf{M}} = [\tilde{G}(x^i, y^j)]_{\substack{i=1,\dots,n \\ j=1,\dots,m}} \quad , \quad \text{rank}(\tilde{\mathbf{M}}) = q.$$

The challenge is to find a minimal $q \in \mathbb{N}$ and construct suitable functions g_ℓ, h_ℓ so that $\tilde{\mathbf{M}}$ is an acceptable approximation of \mathbf{M} .

Remark 2.2.1.13 (Impact of kernel approximation on kernel matrix) Replacing the kernel function G with an approximation \tilde{G} amounts to perturbing the kernel collocation matrix \mathbf{M} . This can be quantified by estimating $\|\mathbf{M} - \tilde{\mathbf{M}}\|$, $\|\cdot\|$ a relevant matrix norm as introduced in § 2.2.0.4.

Let \mathbf{M} be a kernel collocation matrix according to Def. 2.1.3.1 based on the kernel function $G : D_x \times D_y \rightarrow \mathbb{R}$ and collocation points $x^i \in D_x$, $i = 1, \dots, n$, $y^j \in D_y$, $j = 1, \dots, m$. From the definition of the matrix norms we (immediately) conclude how a bound on the pointwise deviation of \tilde{G} from G implies estimates for $\|\mathbf{M} - \tilde{\mathbf{M}}\|$:

$$\begin{aligned} \|G - \tilde{G}\|_{L^\infty(D_x \times D_y)} &\leq \delta \\ \Downarrow & \\ \|\mathbf{M} - \tilde{\mathbf{M}}\|_\infty &\leq m\delta \quad , \quad \|\mathbf{M} - \tilde{\mathbf{M}}\|_1 \leq n\delta \quad , \quad \|\mathbf{M} - \tilde{\mathbf{M}}\|_{F'} \quad , \quad \|\mathbf{M} - \tilde{\mathbf{M}}\|_2 \leq \sqrt{mn}\delta. \end{aligned} \quad (2.2.1.14)$$

Only the bound for $\|\mathbf{M} - \tilde{\mathbf{M}}\|_2$ is not straightforward. In fact it is a consequence of the so-called **Riesz-Thorin theorem** for linear operators $\mathbb{R}^m \rightarrow \mathbb{R}^n$. ┘

The next three sections present different ways how to obtain promising separable approximations with rather explicit formulas for g_ℓ and h_ℓ .

2.2.1.1 Polynomial Expansions

For the sake of clarity we restrict ourselves to one dimension $d = 1$, $D_x, D_y \subset \mathbb{R}$. To understand the following, recall the Taylor formula in 1D for $f \in C^{m+1}([a, b])$, $a < b$, and expansion point $x^* \in [a, b]$:

$$f(x) = f(x^*) + (x - x^*)f'(x^*) + \frac{1}{2}(x - x^*)^2 f''(x^*) + \dots \\ \dots + \frac{1}{(q-1)!}(x - x^*)^{q-1} f^{(q-1)}(x^*) + \int_{x^*}^x \frac{1}{(q-1)!}(x - \tau)^{q-1} f^{(q)}(\tau) d\tau. \quad (2.2.1.15)$$

Dropping the remainder term $\int_{x^*}^x \dots d\tau$ we obtain an approximation of f in a neighborhood of x^* by its **Taylor polynomial** of degree $q - 1$,

$$f(x) \approx \sum_{\ell=0}^{q-1} \frac{1}{\ell!} (x - x^*)^\ell f^{(\ell)}(x^*). \quad (2.2.1.16)$$

We can apply this approximation to the “1D function” $x \mapsto G(x, y)$ and simply regard y as a parameter.

Idea: Approximate $G(x, y)$ by a *truncated Taylor expansion* in the x -variable:



$$\tilde{G}(x, y) \approx \sum_{\ell=0}^{q-1} \underbrace{\frac{1}{\ell!} (x - x^*)^\ell}_{=:g_\ell(x)} \underbrace{\frac{\partial^\ell G}{\partial x^\ell}(x^*, y)}_{=:h_\ell(y)}, \quad x, x^* \in D_x, y \in D_y, \quad (2.2.1.17)$$

for a “sufficiently” large truncation parameter $q \in \mathbb{N}$.

As indicated in (2.2.1.17), this provides a rank- q separable approximation of G . The number q of terms in the polynomial expansion can be used to control the accuracy, because we expect $\tilde{G} \rightarrow G$ for $q \rightarrow \infty$. This will be examined in Section 2.2.2.1 below.

EXAMPLE 2.2.1.18 (Separable approximation by truncated power series) We consider the **globally C^∞ -smooth** kernel function

$$G(x, y) = \frac{1}{1 + (x - y)^2} \quad \text{on } I \times I, \quad I := [-a, a], \quad a \in \mathbb{R}^+.$$

We want to approximate it **globally** by truncated power series expansions around $x^* = 0$, which is a natural choice for symmetry reasons.

The geometric series summation formula $(1 + \zeta)^{-1} = \sum_{k=0}^{\infty} (-\zeta)^k$ gives the Taylor series expansion at $x^* = 0$, valid for $|x - y| < 1$:

$$G(x, y) = \sum_{k=0}^{\infty} \left(-(x - y)^2 \right)^k = \sum_{k=0}^{\infty} (-1)^k (x - y)^{2k} = \sum_{k=0}^{\infty} (-1)^k \sum_{\ell=0}^{2k} \binom{2k}{\ell} x^\ell (-y)^{2k-\ell} \\ = \sum_{\ell=0}^{\infty} x^\ell \cdot \sum_{k=\lceil \ell/2 \rceil}^{\infty} (-1)^k \binom{2k}{\ell} (-y)^{2k-\ell},$$

where we also used the binomial formula. However, this is a double series. We truncate the geometric sum to the first q summands to obtain a rank- $2q$ separable approximation:

$$\tilde{G}(x, y) = \sum_{k=0}^{q-1} (-1)^k \sum_{\ell=0}^{2k} \binom{2k}{\ell} x^\ell (-y)^{2k-\ell} = \sum_{\ell=0}^{2(q-1)} \underbrace{x^\ell}_{=:g_\ell(x)} \cdot \underbrace{\sum_{k=\lceil \ell/2 \rceil}^{q-1} (-1)^k \binom{2k}{\ell} (-y)^{2k-\ell}}_{=:h_\ell(y)}.$$

In a numerical experiment we monitor $\|G - \tilde{G}\|_{L^\infty(I \times I)}$, approximated on a very fine grid on $I \times I$, as a function of the interval size a and expansion degree q .

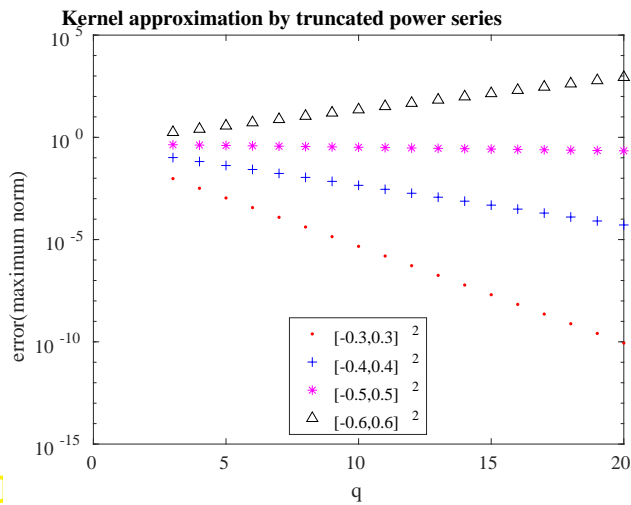


Fig. 68

$\triangleleft \|G - \tilde{G}\|_{L^\infty(I \times I)}$ for different values of truncation parameter q and on different intervals I .

Exponential convergence in q for small intervals.

“Exponential divergence” on large intervals.

Global separable kernel approximation based on Taylor expansion/power series is usually possible *only locally* (on small domains).

┘

EXPERIMENT 2.2.1.19 (Logarithmic kernel in 1D: Separable approximation by Taylor expansion)

We consider the *singular* kernel function

$$G(x, y) = -\log|x - y|, \quad x \in D_x, y \in D_y, \quad D_x, D_y \subset \mathbb{R} \text{ intervals, } \overline{D_x} \cap \overline{D_y} = \emptyset.$$

The condition $\overline{D_x} \cap \overline{D_y} = \emptyset$ avoids the singularity of the kernel. Thus, on $D_x \times D_y$ the kernel function G is C^∞ -smooth and amenable to Taylor expansion.

Without loss of generality we assume $y > x$ on $D_x \times D_y$ (D_x to the left of D_y).

$$\blacktriangleright \frac{\partial^\ell G}{\partial x^\ell}(x, y) = (\ell - 1)!(y - x)^{-\ell} \quad \text{for } (x, y) \in D_x \times D_y, \quad \ell \geq 1. \tag{2.2.1.20}$$

This yields the Taylor polynomial with expansion point $x^* \in D_x$:

$$\begin{aligned} -\log(y - x) &\approx \tilde{G}(x, y) = \sum_{\ell=0}^{q-1} \frac{1}{\ell!} (x - x^*)^\ell \frac{\partial^\ell G}{\partial x^\ell}(x^*, y) \\ &= -\log(y - x^*) + \sum_{\ell=1}^{q-1} \underbrace{\frac{1}{\ell} (x - x^*)^\ell}_{=: g_\ell(x)} \underbrace{(y - x^*)^{-\ell}}_{=: h_\ell(y)}. \end{aligned} \tag{2.2.1.21}$$

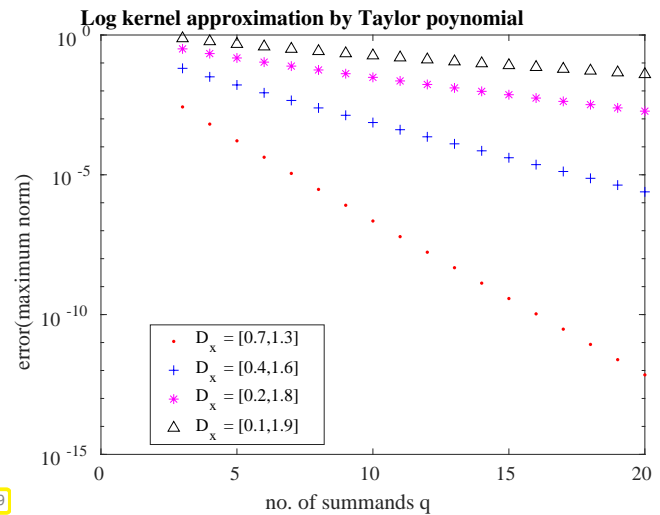
$$D_Y := [2, 3]$$

$\|G - \tilde{G}\|_{L^\infty(D_x \times D_y)}$, \tilde{G} as in (2.2.1.21), sampling approximation on fine grid ▷

We observe **exponential convergence** in truncation parameter q .

(Would observe “exponential divergence” on larger intervals)

Fig. 69



▶ Same bottom line as in Ex. 2.2.1.18 applies.

2.2.1.2 Uni-directional Interpolation

Since the Taylor expansion of the kernel function G is fixed, we have little options to remedy potentially small domains of convergence. Moreover, the Taylor expansion and power series techniques from Section 2.2.1.1 require knowledge of higher-order partial derivatives of G . Conversely, the interpolation techniques presented in this section are more flexible and rely on point evaluations of G alone.

§2.2.1.22 ((Abstract) Linear interpolation operators) Let $D \subset \mathbb{R}^d$ be a closed bounded domain and $V \subset C^0(D)$ a q -dimensional space of continuous functions.

Given are q distinct interpolation nodes $t^j \in D, j = 1, \dots, q$.

Assumption 2.2.1.23. Unisolvence of interpolation nodes

We assume that for any numbers $\varphi_1, \dots, \varphi_q \in \mathbb{R}$ there is a **unique** $f \in V$ satisfying the **interpolation conditions**

$$f(t^j) = \varphi_j \quad \text{for all } j = 1, \dots, q. \tag{2.2.1.24}$$

In approximation theory this particular property of the space V and the set $\{t^j\}_j$ of interpolation nodes is known as **unisolvence**.

Definition 2.2.1.25. Linear interpolation operator

For a **unisolvence** set of interpolation nodes $\{t^j\}_{j=1}^q, q \in \mathbb{N}$, w.r.t. $V \subset C^0(D)$, $\dim V = q$, define the associated **linear interpolation operator** by

$$I : C^0(D) \rightarrow V, \quad If \in V: (If)(t^j) = f(t^j) \quad \forall j = 1, \dots, q. \tag{2.2.1.26}$$

Lemma 2.2.1.27. Properties of I

The mapping I according to (2.2.1.26) is linear, continuous, and surjective.

We can write

$$If = \sum_{\ell=1}^q f(t^\ell) b_\ell \quad \forall f \in C^0(D), \tag{2.2.1.28}$$

where the **cardinal functions** $b_\ell \in V$, $\ell = 1, \dots, q$, are defined (possible thanks to Ass. 2.2.1.23!) by

$$b_\ell(\mathbf{t}^j) = \delta_{\ell,j} := \begin{cases} 1 & \text{for } \ell = j, \\ 0 & \text{else,} \end{cases} \quad \ell, j \in \{1, \dots, q\}. \quad (2.2.1.29)$$

┘

§2.2.1.30 (Separable approximation by interpolation) We examine the following setting: we are given

- ◆ a continuous kernel function $G : D_x \times D_y \rightarrow \mathbb{R}$, $G \in C^0(\overline{D}_x \times \overline{D}_y)$,
- ◆ and a linear interpolation operator $I : C^0(D_x) \rightarrow V$ according to Def. 2.2.1.25, based on interpolation nodes $\{\mathbf{t}^j\}_{j=1}^q$ and a q -dimensional function space $V \subset C^0(D_x)$ (satisfying the unisolvence Assumption 2.2.1.23, of course)

Performing interpolation in the x -variable, we build a rank- q separable “approximation” of the kernel function G :

$$\tilde{G}(x, y) := \sum_{\ell=1}^q \underbrace{b_\ell(x)}_{=:g_\ell(x)} \underbrace{G(\mathbf{t}^\ell, y)}_{=:h_\ell(y)}, \quad (x, y) \in D_x \times D_y, \quad (2.2.1.31)$$

where the b_ℓ are the cardinal functions for the interpolation into V with nodes \mathbf{t}^j as defined by the property (2.2.1.29). The variable y is treated as a mere parameter. Appealing to the considerations of § 2.2.1.10 we immediately get a special version of the factorization (2.2.1.12) of the kernel collocation matrix based on \tilde{G} :

$$\tilde{\mathbf{M}} := [\tilde{G}(x^i, y^j)]_{\substack{i=1,\dots,n \\ j=1,\dots,m}} = \mathbf{A} \cdot \mathbf{B}^\top, \quad \begin{aligned} \mathbf{A} &= [b_\ell(x^i)]_{\substack{i=1,\dots,n \\ \ell=1,\dots,q}} \in \mathbb{R}^{n,q}, \\ \mathbf{B} &= [G(\mathbf{t}^\ell, y^j)]_{\substack{j=1,\dots,m \\ \ell=1,\dots,q}} \in \mathbb{R}^{m,q}. \end{aligned} \quad (2.2.1.32)$$

┘

§2.2.1.33 (Polynomial interpolation in 1D [NumCSE Section 5.2]) The most important class of interpolation schemes is global **polynomial interpolation**. For $d = 1$ and if $D \subset \mathbb{R}$ is an interval, it relies on the space of uni-variate polynomials

$$V := \mathcal{P}_q := \text{Span}\{x \mapsto x^\ell, \ell = 0, \dots, q-1\}, \quad q \in \mathbb{N},$$

of degree $\leq q-1$. By [NumCSE Thm. 5.2.2.7], any set of q distinct points $\mathbf{t}^j \in D$ enjoys unisolvence with respect to V , which guarantees Ass. 2.2.1.23.

As explained in [NumCSE § 5.2.2.3], the cardinal functions of uni-variate polynomial interpolation in the nodes $\mathbf{t}^1, \dots, \mathbf{t}^q$ are the **Lagrange polynomials** [NumCSE Eq. (5.2.2.4)]

$$L_\ell(x) := \prod_{\substack{j=1 \\ j \neq \ell}}^q \frac{x - \mathbf{t}^j}{\mathbf{t}^\ell - \mathbf{t}^j}, \quad x \in \mathbb{R}, \quad \ell = 1, \dots, q \Rightarrow L_\ell(\mathbf{t}^j) = \delta_{\ell,j}. \quad (2.2.1.34)$$

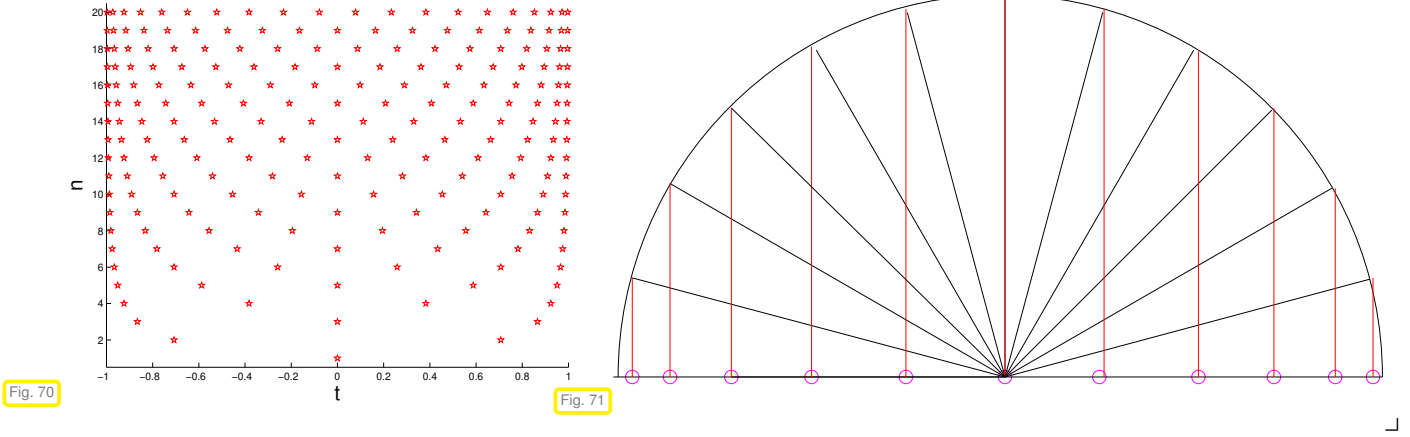
Then for $G : D_x \times D_y \rightarrow \mathbb{R}$, $D_x, D_y \subset \mathbb{R}$, the separable approximation according to (2.2.1.31) reads

$$\tilde{G}(x, y) := \sum_{\ell=1}^q \underbrace{L_\ell(x)}_{=:g_\ell(x)} \underbrace{G(\mathbf{t}^\ell, y)}_{=:h_\ell(y)}, \quad (x, y) \in D_x \times D_y. \quad (2.2.1.35)$$

How to chose the t^j ? From [NumCSE Rem. 5.2.4.13] we know that equidistant interpolation nodes may lead to massive oscillations in the interpolants for larger polynomial degree. As a remedy we learned that “optimal” nodes for polynomial interpolation are the **Chebyshev nodes**, see [NumCSE Section 6.2.3], [NumCSE Eq. (6.2.3.12)]. If $D_x = [a, b]$ those are defined as

$$t^j := a + \frac{1}{2}(b - a) \left(\cos \left(\frac{2j-1}{2q} \pi \right) + 1 \right), \quad j = 1, \dots, q. \tag{2.2.1.36}$$

Below: Chebyshev nodes on $[-1, 1]$; they conspicuously cluster close to the endpoints of the interval.



§2.2.1.37 (Tensor-product polynomial interpolation) The generic case we study is kernel functions defined on subsets of $\mathbb{R}^d \times \mathbb{R}^d$. Therefore we have to extend polynomial interpolation to higher dimensions, which can be done by means of a *tensor-product construction*.

If $D \subset \mathbb{R}^d$ is a tensor-product domain

$$D = [a_1, b_1] \times \dots \times [a_d, b_d], \quad a_i < b_i, \quad i = 1, \dots, d,$$

then we can define a d -dimensional polynomial interpolation into the space of **tensor-product polynomials** (\rightarrow Def. 1.4.3.80 or [NumPDE Def. 2.5.2.7])

$$\mathcal{TP}_q(\mathbb{R}^d) := \{x \mapsto p_1(x_1) \cdots p_d(x_d), p_i \in \mathcal{P}_q, i = 1, \dots, d\}, \quad \dim \mathcal{TP}_q(\mathbb{R}^d) = q^d.$$

Of course, the scheme is based on uni-variate polynomial interpolation as introduced in § 2.2.1.33.

Let t_i^1, \dots, t_i^q be nodes for uni-variate polynomial interpolation on $[a_i, b_i]$ into \mathcal{P}_q . Denote by $L_{i,\ell}$, $\ell = 1, \dots, q$, the associated Lagrange polynomials. Then we can define the d -variate **tensor-product polynomial interpolation operator**

$$\begin{aligned} I_D : C^0(D) &\rightarrow \mathcal{TP}_q(\mathbb{R}^d), \\ (I_D f)(x) &= \sum_{k_1=1}^q \cdots \sum_{k_d=1}^q f \left([t_1^{k_1}, \dots, t_d^{k_d}]^\top \right) L_{1,k_1}(x_1) \cdots L_{d,k_d}(x_d), \quad x \in \mathbb{R}^d, \quad f \in C^0(D). \end{aligned} \tag{2.2.1.38}$$

Hence, we have to evaluate f on a **grid** of q^d points, which matches $\dim \mathcal{TP}_{q-1}(\mathbb{R}^d) = q^d$.

Note that

$$\{x \mapsto L_{1,k_1}(x_1) \cdots L_{d,k_d}(x_d) : k_i = 1, \dots, q, i = 1, \dots, d\} \subset \mathcal{TP}_q(\mathbb{R}^d) \tag{2.2.1.39}$$

is an (ordered) cardinal basis of $\mathcal{TP}_q(\mathbb{R}^d)$ with respect to the (ordered) set of interpolation nodes

$$\left\{ [t_1^{k_1}, \dots, t_d^{k_d}]^\top : k_i = 1, \dots, q, i = 1, \dots, d \right\}. \tag{2.2.1.40}$$

We may call this set a “tensor-product grid of interpolation nodes”.

§2.2.1.41 (Low-rank kernel matrix approximation by tensor-product polynomial interpolation) Now let us fit degree- p tensor-product uni-directional polynomial interpolation into the framework laid out in § 2.2.1.30 and § 2.2.1.37. We do this for $d = 2$ and one-dimensional interpolation nodes t_1^0, \dots, t_1^p (first coordinate) and t_2^0, \dots, t_2^p (second coordinate). In this case we have $q = (p + 1)^2$ and the interpolation nodes are

$$\left\{ \underbrace{\begin{bmatrix} t_1^0 \\ t_2^0 \end{bmatrix}}_{=:t^1}, \dots, \underbrace{\begin{bmatrix} t_1^p \\ t_2^0 \end{bmatrix}}_{=:t^{p+1}}, \underbrace{\begin{bmatrix} t_1^0 \\ t_2^1 \end{bmatrix}}_{=:t^{p+2}}, \dots, \underbrace{\begin{bmatrix} t_1^p \\ t_2^1 \end{bmatrix}}_{=:t^{p(p+1)}}, \dots, \underbrace{\begin{bmatrix} t_1^p \\ t_2^p \end{bmatrix}}_{=:t^{(p+1)^2}} \right\}. \tag{2.2.1.42}$$

Their ordering is arbitrary. The corresponding cardinal basis functions are products of 1D Lagrange polynomials ($x \mapsto L_{1,j}(x), x \mapsto L_{2,j}(x)$ for the node sets $\{t_1^j\}, \{t_2^j\} \subset \mathbb{R}$, respectively, $j \in \{1, \dots, q\}$) following the same ordering

$$\begin{aligned} b_1(x) &:= L_{1,0}(x_1)L_{2,0}(x_2), \quad \dots, \quad b_{p+1}(x) := L_{1,p}(x_1)L_{2,0}(x_2), \\ &\vdots \\ b_{p(p+1)}(x) &:= L_{1,0}(x_1)L_{2,p}(x_2), \quad \dots, \quad b_{(p+1)^2}(x) := L_{1,p}(x_1)L_{2,p}(x_2), \end{aligned} \quad x \in \mathbb{R}^2. \tag{2.2.1.43}$$

Then the general formula

$$\tilde{G}(x, y) := \sum_{\ell=1}^q \underbrace{b_\ell(x)}_{=:g_\ell(x)} \underbrace{G(t^\ell, y)}_{=:h_\ell(y)}, \quad (x, y) \in D_x \times D_y, \tag{2.2.1.31}$$

combined with

$$\tilde{\mathbf{M}} := [\tilde{G}(x^i, y^j)]_{\substack{i=1, \dots, n \\ j=1, \dots, m}} = \mathbf{A} \cdot \mathbf{B}^\top, \quad \begin{aligned} \mathbf{A} &= [b_\ell(x^i)]_{\substack{i=1, \dots, n \\ \ell=1, \dots, q}} \in \mathbb{R}^{n,q}, \\ \mathbf{B} &= [G(t^\ell, y^j)]_{\substack{j=1, \dots, m \\ \ell=1, \dots, q}} \in \mathbb{R}^{m,q}. \end{aligned} \tag{2.2.1.32}$$

leads to

$$[G(x^i, y^j)]_{\substack{i=1, \dots, n \\ j=1, \dots, m}} \approx \tilde{\mathbf{M}} := \begin{bmatrix} b_1(x^1) & \dots & b_{(p+1)^2}(x^1) \\ \vdots & & \vdots \\ b_1(x^n) & \dots & b_{(p+1)^2}(x^n) \end{bmatrix} \begin{bmatrix} G(t^1, y^1) & \dots & G(t^1, y^m) \\ \vdots & & \vdots \\ G(t^{(p+1)^2}, y^1) & \dots & G(t^{(p+1)^2}, y^m) \end{bmatrix}$$

Obviously, $\text{rank } \tilde{\mathbf{M}} \leq (p + 1)^2$.

2.2.1.3 Bi-directional interpolation

Many kernel functions $(x, y) \mapsto G(x, y)$ are symmetric in their two arguments. However, separable kernel approximation by means of uni-directional interpolation as introduced in Section 2.2.1.2 treats the x - and y -coordinates rather differently. Another interpolation approach preserves symmetry.

§2.2.1.44 (General “two-dimensional” interpolation) We assume that we are given

- ◆ a continuous kernel function $G : D_x \times D_y \rightarrow \mathbb{R}, G \in C^0(D_x \times D_y)$,

- ◆ a linear interpolation operator $I^x : C^0(D_x) \rightarrow V_x$ according to Def. 2.2.1.25 based on interpolation nodes $t_x^1, \dots, t_x^{q_x} \in D_x$, $q_x \in \mathbb{R}$, and a q_x -dimensional function space $V_x \subset C^0(D_x)$,
- ◆ another linear interpolation operator $I^y : C^0(D_y) \rightarrow V_y$ into a q_y -dimensional space $V_y \subset C^0(D_y)$ with interpolation nodes $t_y^1, \dots, t_y^{q_y} \in D_y$.

We write b_k^x , $k = 1, \dots, q_x$, and b_j^y , $j = 1, \dots, q_y$, for the cardinal functions associated with the respective spaces and sets of interpolation nodes on D_x, D_y .

Then, in the spirit of tensor-product polynomial interpolation from § 2.2.1.37, we can introduce the **tensor-product interpolation operator**

$$\begin{aligned}
 I^x \otimes I^y : C^0(D_x \times D_y) &\rightarrow V_x \otimes V_y \quad , \\
 ((I^x \otimes I^y)f)(x, y) &:= \sum_{k=1}^{q_x} \sum_{j=1}^{q_y} f(t_x^k, t_y^j) b_k^x(x) b_j^y(y) \quad , \quad f \in C^0(D_x \times D_y) \quad .
 \end{aligned}
 \tag{2.2.1.45}$$

┘

Obviously, the tensor-product interpolant is separable. Hence, applying it to G provides a separable “approximation” (its quality depending on k, I^x , and I^y , of course):

$$\tilde{G}(x, y) := ((I^x \otimes I^y)G)(x, y) = \sum_{k=1}^{q_x} \sum_{\ell=1}^{q_y} \underbrace{G(t_x^k, t_y^\ell)}_{=:g_{k,\ell}(x)} \underbrace{b_k^x(x) b_\ell^y(y)}_{=:h_{k,\ell}(y)} \quad .
 \tag{2.2.1.46}$$

Note that in order to obtain \tilde{G} we need only evaluate G at $q^x \cdot q^y$ pairs of interpolation nodes to obtain the values $G(t_x^k, t_y^\ell) \in \mathbb{R}$. Another advantage of (2.2.1.46) is that it inherits the possible simplicity of the cardinal functions.

For given collocation points $x^1, \dots, x^n \in D_x, y^1, \dots, y^m \in D_y$, the approximate kernel collocation matrix $\tilde{M} \in \mathbb{R}^{n,m}$ spawned by \tilde{G} has the special **triple-factor form**

$$\begin{aligned}
 (\tilde{M})_{i,j} &= \sum_{k=1}^{q_x} \sum_{\ell=1}^{q_y} G(t_x^k, t_y^\ell) b_k^x(x^i) b_\ell^y(y^j) \quad , \quad i = 1, \dots, n \quad , \quad j = 1, \dots, m \\
 \blacktriangleright \quad \tilde{M} &= \mathbf{U} \mathbf{C} \mathbf{V}^T \quad , \quad \mathbf{U} := [b_k^x(x^i)]_{\substack{i=1,\dots,n \\ k=1,\dots,q_x}} \in \mathbb{R}^{n,q_x} \quad , \\
 \mathbf{C} &:= [G(t_x^k, t_y^\ell)]_{\substack{k=1,\dots,q_x \\ \ell=1,\dots,q_y}} \in \mathbb{R}^{q_x,q_y} \quad , \\
 \mathbf{V} &:= [b_\ell^y(y^j)]_{\substack{j=1,\dots,m \\ \ell=1,\dots,q_y}} \in \mathbb{R}^{m,q_y} \quad .
 \end{aligned}
 \tag{2.2.1.47}$$

$$\left[\begin{array}{c} \tilde{M} \end{array} \right] = \left[\begin{array}{c} \mathbf{U} \end{array} \right] \left[\begin{array}{c} \mathbf{C} \end{array} \right] \left[\begin{array}{c} \mathbf{V}^T \end{array} \right] \quad .$$

This implies that $\text{rank}(\tilde{M}) \leq \min\{q_x, q_y\}$.

Of course the most widely used interpolation operators I^x and I^y are polynomial interpolations, in particular, Chebychev interpolation [NumCSE Section 6.2.3]. Then the b_k^x/b_j^y will also be polynomials, for which efficient algorithms for evaluation are available, see [NumCSE Section 5.2.3].

EXPERIMENT 2.2.1.48 (Bi-directional interpolation of smooth kernel function) For $d = 1$ we consider the globally smooth kernel function

$$G(x, y) = \frac{1}{1 + (x - y)^2} \text{ on } [0, 1]^2,$$

- and collocation points $\xi_i = \eta_i = \frac{i-1}{n}, i = 1, \dots, n, n \in \mathbb{N}$.

A rank- q^2 separable approximation of G on $[0, 1]^2$ is obtained by bi-directional Chebychev interpolation into tensor-product polynomials $\mathcal{TP}_{q-1}(\mathbb{R}^2)$:

$$\tilde{G} \in \mathcal{TP}_{q-1}(\mathbb{R}^2): \tilde{G}(t^i, t^j) = G(t^i, t^j), \quad i, j \in \{1, \dots, q\}, \tag{2.2.1.49}$$

with Chebychev nodes t^k as defined in (2.2.1.36).

Plot of the *scaled* Frobenius norm of the approximation error of the kernel collocation matrix,

$$\text{err} := \frac{1}{n} \left(\sum_{i,j} \left(G(\xi_i, \eta_j) - \tilde{G}(\xi_i, \eta_j) \right)^2 \right)^{\frac{1}{2}},$$

as a function of the degree $q - 1$ and for $n \in \{100, 500, 1000\}$.

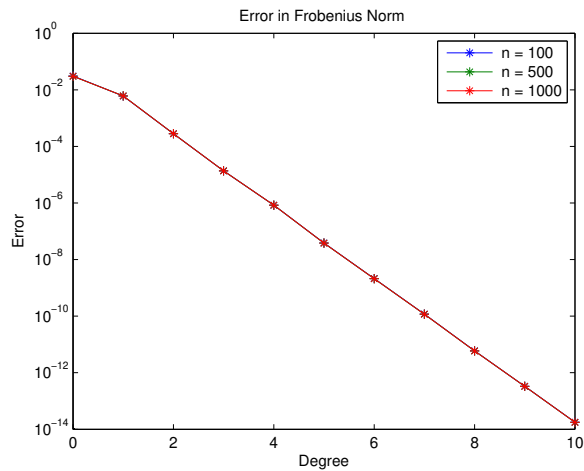


Fig. 72

Observation: Evidence of **exponential convergence** of the approximation error for $q \rightarrow \infty$.

The observation matches theoretical interpolation error estimates for Chebychev interpolation: the kernel $(x, y) \mapsto G(x, y)$ is analytic on $[0, 1]$ (\rightarrow Def. 1.4.3.67) both as a function $x \mapsto G(x, y)$ and $y \mapsto G(x, y)$, uniformly in the other argument. Thus, the results reported in [NumCSE Rem. 6.2.3.26] predict exponential convergence of $\|G - \tilde{G}\|_{L^\infty([0,1]^2)}$, refer to Thm. 1.4.3.70 for quantitative formulas. Details will be given in Section 2.2.2.2. ┘

EXPERIMENT 2.2.1.50 (Global bi-directional interpolation of singular kernel) For $d = 1$ we apply bi-directional Chebychev interpolation into $\mathcal{TP}_{q-1}(\mathbb{R}^2)$ to the singular kernel function

$$G(x, y) = \begin{cases} \frac{1}{|x-y|} & , \text{ if } x \neq y, \\ 0 & , \text{ if } x = y, \end{cases} \quad 0 \leq x, y \leq 1,$$

in order to obtain a separable approximation \tilde{G} .

We use the same collocation points as in Exp. 2.2.1.48.

Plot of $\|\mathbf{M} - \widetilde{\mathbf{M}}\|_F$ as a function of the degree $q - 1$ for $n \in \{100, 500, 1000\}$.

We observe **no convergence** at all.

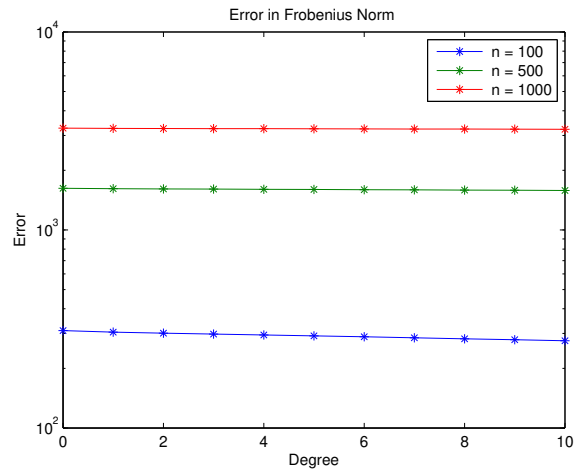


Fig. 73

2.2.2 Error Estimates and Admissibility Condition for Singular Kernels

We embark on an analysis of separable approximation of **singular kernels** like those introduced in Ex. 2.1.3.3, with focus on $d = 1$, the logarithmic kernel

$$G(x, y) = \begin{cases} -\log|x - y| & , \text{ if } x \neq y, \\ 0 & \text{ else,} \end{cases} \quad x, y \in [0, 1].$$

and Chebychev polynomial interpolation. We have seen in Exp. 2.2.1.50 that applying polynomial across the singularity at $x = y$ is pointless. Conversely, Exp. 2.2.1.19 sends the message that singular kernels for $d = 1$ allow exponentially convergent separable approximations on “boxes” $D_x \times D_y \subset \mathbb{R}^2$ away from the “diagonal” $\{(x, y) \in \mathbb{R}^2 : x = y\}$. Now we estimate truncation and interpolation errors to glean quantitative information.

Remark 2.2.2.1 (Asymptotically smooth kernels [Beb08, Sect. 3.2]) The analysis of this section carries over to $d > 1$ and a larger class of singular kernels, which are **asymptotically smooth**.

A kernel function $G : (\mathbb{R}^d \times \mathbb{R}^d) \setminus \{(x, y) \in \mathbb{R}^d \times \mathbb{R}^d : x = y\} \rightarrow \mathbb{R}$ is called **asymptotically smooth**, if

- (i) $G \in C^\infty((\mathbb{R}^d \times \mathbb{R}^d) \setminus \{(x, y) \in \mathbb{R}^d \times \mathbb{R}^d : x = y\})$,
- (ii) and its derivatives satisfy the decay conditions

$$|D_y^\alpha G(x, y)| \leq C |\alpha|! \gamma^{|\alpha|} \frac{|G(x, y)|}{\|x - y\|^{|\alpha|}} \quad \forall \alpha \in \mathbb{N}_0^d, \quad \forall (x, y) \in \mathbb{R} \times \mathbb{R} \setminus \{(x, y) \in \mathbb{R}^d \times \mathbb{R}^d : x = y\}, \quad (2.2.2.2)$$

with constants $C > 0, \gamma > 0$ ($|\alpha| = |\alpha_1| + \dots + |\alpha_d|$).

Straightforward differentiation, cf. (2.2.1.20), confirms that the kernels from Ex. 2.1.3.3 are asymptotically smooth.

2.2.2.1 Truncation Error Estimates for Taylor Expansion

We focus on the asymptotically smooth logarithmic kernel $G(x, y) = -\log|x - y|$ in one dimension, $d = 1$.

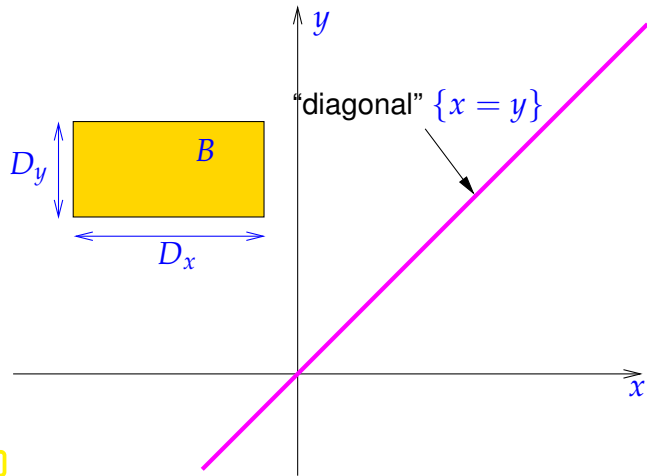


Fig. 74

As in Exp. 2.2.1.19 we consider its rank- q separable approximation by means of truncated Taylor expansion on $D_x \times D_y$, where $D_x, D_y \subset \mathbb{R}$ are *disjoint intervals*: $D_x \cap D_y = \emptyset$.

◁ Approximation on a “box” $B := D_x \times D_y$ away from the diagonal.

(Assume that B is above the diagonal: $y > x$ for all $(x, y) \in B$.)

Using that for $y > x$

$$\frac{\partial^\ell G}{\partial x^\ell}(x, y) = (\ell - 1)!(y - x)^{-\ell} \quad \text{for } (x, y) \in D_x \times D_y, \quad \ell \geq 1, \quad (2.2.1.20)$$

the formula (2.2.1.17) for the rank- q separable approximation by Taylor expansion takes the concrete form

$$\tilde{G}(x, y) = -\log(y - x^*) + \sum_{\ell=1}^{q-1} \underbrace{\frac{1}{\ell}(x - x^*)^\ell}_{=:g_\ell(x)} \underbrace{(y - x^*)^{-\ell}}_{=:h_\ell(y)}, \quad (2.2.2.3)$$

We choose $x^* \in D_x$ as the *midpoint* of D_x : if $D_x = [a, b]$, then $x^* = \frac{1}{2}(a + b)$.

§2.2.2.4 (Heuristics based on maximal analytic extension) Appealing to the arguments of [NumCSE Rem. 6.2.2.67] we find that the domain of analyticity of $z \mapsto -\log(y - z)$, $y \in \mathbb{R}$, is $\mathbb{C} \setminus [y, \infty[$, because the complex logarithm is analytic everywhere except \mathbb{R}_0^- .

Then apply the “rule of thumb” that predicts that

the Taylor series of an *analytic* function $f : D \rightarrow \mathbb{C}$, $D \subset \mathbb{C}$ open (“domain of analyticity”), around $z^* \in D$ converges inside every disk centered at z^* that lies completely inside D .

Thus the Taylor series of $x \mapsto -\log(y - x)$ in $x^* < y$

$$\begin{aligned} G(x, y) = -\log(y - x) &= \sum_{\ell=0}^{\infty} \frac{1}{\ell!} (x - x^*)^\ell \frac{\partial^\ell G}{\partial x^\ell}(x^*, y) \\ &=: \sum_{\ell=0}^{\infty} \gamma_\ell (x - x^*)^\ell \quad \text{for } \gamma_\ell := \frac{1}{\ell!} \frac{\partial^\ell G}{\partial x^\ell}(x^*, y), \end{aligned} \quad (2.2.2.5)$$

has a radius of convergence $\rho = y - x^*$. **Assume** that for $|x - x^*| = \rho$ the terms of the series are still bounded:

$$\gamma_\ell \rho^\ell \leq C \quad \forall \ell \in \mathbb{N}_0. \quad (2.2.2.6)$$

Therefore, if $|x - x^*| < \rho$, we get

$$\begin{aligned} |(G - \tilde{G})(x, y)| &= \left| \sum_{\ell=q}^{\infty} \gamma_\ell (x - x^*)^\ell \right| \leq \sum_{\ell=q}^{\infty} \left| \frac{x - x^*}{\rho} \right|^\ell \gamma_\ell \rho^\ell \\ &\stackrel{(2.2.2.5)}{\leq} C \sum_{\ell=q}^{\infty} \left| \frac{x - x^*}{\rho} \right|^\ell = C \left| \frac{x - x^*}{\rho} \right|^q \frac{\rho}{\rho - |x - x^*|} \end{aligned}$$

Note that by simple geometric arguments

$$|x^* - y| \geq \frac{1}{2} \text{diam}(D_x) + \text{dist}(D_x; D_y) \quad , \quad |x - x^*| \leq \frac{1}{2} \text{diam}(D_x) .$$

$$\blacktriangleright \quad \left| \frac{x - x^*}{\rho} \right| \leq \frac{\eta}{1 + \eta} ,$$

with the **admissibility measure** of the box $B := D_x \times D_y$

$$\eta = \eta(B) := \frac{\max\{\text{diam}(D_x), \text{diam}(D_y)\}}{2 \text{dist}(D_x; D_y)} . \quad (2.2.2.7)$$

Hence, for $x \in D_x$ we expect *exponential convergence* (in terms of $q \rightarrow \infty$) of the q -term Taylor expansion in x with error bounds

$$\|G - \tilde{G}\|_{L^\infty(D_x \times D_y)} \leq C \left(\frac{\eta}{1 + \eta} \right)^q \quad \forall q \in \mathbb{N} . \quad (2.2.2.8)$$

§2.2.2.9 (Remainder estimates for Taylor expansion of logarithmic kernel) Now we make rigorous the heuristic arguments of § 2.2.2.4. Recall the remainder formula for one-dimensional Taylor expansion of $f \in C^{m+1}([a, b])$ around $x^* \in [a, b]$ [Str09, Sect. 5.5],

$$f(x) - \sum_{\ell=0}^{q-1} \frac{1}{\ell!} (x - x^*)^\ell f^{(\ell)}(x^*) = (x - x^*)^q \int_0^1 \frac{1}{(q-1)!} (1-\tau)^{q-1} f^{(q)}(x^* + \tau(x - x^*)) \, d\tau . \quad (2.2.2.10)$$

Apply this formula to $G(x, y)$ in *x-direction only*, regarding y as a parameter. The remainder term for expansion length $q \geq 1$ and x^* chosen as midpoint of D_x reads

$$\begin{aligned} G(x, y) - \tilde{G}(x, y) &= \frac{(x - x^*)^q}{(q-1)!} \int_0^1 (1-\tau)^{q-1} \frac{\partial^q G}{\partial x^q}(x^* + \tau(x - x^*), y) \, d\tau \\ &= \frac{(x - x^*)^q}{(q-1)!} \int_0^1 (1-\tau)^{q-1} (q-1)! |x^* + \tau(x - x^*) - y|^{-q} \, d\tau \\ &= \int_0^1 (1-\tau)^{q-1} \left(\frac{x - x^*}{|x^* - y + \tau(x - x^*)|} \right)^q \, d\tau . \end{aligned}$$

We estimate the remainder in terms of geometric quantities

$$\begin{aligned} |x^* - y| &\geq \frac{1}{2} \text{diam}(D_x) + \text{dist}(D_x; D_y) , \\ |x - x^*| &\leq \frac{1}{2} \text{diam}(D_x) . \end{aligned}$$

where

$$\text{dist}(D_x; D_y) := \max\{|x - y| : x \in D_x, y \in D_y\} > 0 .$$

Hence, for all $(x, y) \in D_x \times D_y$,

$$\begin{aligned}
|G(x, y) - \tilde{G}(x, y)| &\leq \int_0^1 (1 - \tau)^{q-1} \left(\frac{x - x^*}{|x^* - y| - |\tau(x - x^*)|} \right)^q d\tau \\
&\leq \int_0^1 (1 - \tau)^{q-1} \left(\frac{\frac{1}{2} \text{diam}(D_x)}{\text{dist}(D_x; D_y) + \frac{1}{2}(1 - \tau) \text{diam}(D_x)} \right)^q d\tau \\
&\leq \int_0^1 \frac{(1 - \tau)^{q-1}}{(\eta^{-1} + 1 - \tau)^q} d\tau = \int_0^1 \frac{\sigma^{q-1}}{(\eta^{-1} + \sigma)^q} d\sigma \leq \left(\frac{\eta}{1 + \eta} \right)^{q-1} \int_0^1 \frac{\eta}{1 + \eta\sigma} d\sigma \\
&= \left(\frac{\eta}{1 + \eta} \right)^{q-1} \log(1 + \eta) = O(\eta^q) \quad \text{for } q \rightarrow \infty.
\end{aligned} \tag{2.2.2.11}$$

Again the **admissibility measure** η of the box $B := D_x \times D_y$ as defined in (2.2.2.7) crucially enters the bound for the truncation error and determines the “rate” of exponential convergence for $q \rightarrow \infty$.

On boxes away from the diagonal $\{x = y\}$ rank- q separable approximation of asymptotically smooth singular kernels by means of truncated Taylor expansion converges uniformly exponentially for $q \rightarrow \infty$.

The speed of convergence is determined by the **admissibility measure** $\eta = \eta(B)$.

2.2.2.2 Interpolation Error Estimate for Chebychev Interpolation

We conduct a rigorous analysis for separable approximation by uni-directional interpolation as presented in Section 2.2.1.2, see (2.2.1.31). We restrict ourselves to $d = 1$ and **Chebychev interpolation** [NumCSE Section 6.2.3.2].

Specializing (2.2.1.31), the approximate rank- q separable kernel function is given by

$$\tilde{G}(x, y) = \mathcal{I}_{q, D_x} \{x \mapsto G(x, y)\} = \sum_{\ell=1}^q L_\ell(x) G(t^\ell, y), \tag{2.2.2.12}$$

where $\mathcal{I}_{q, D_x} : C^0(D_x) \rightarrow \mathcal{P}_{q-1}$ is the q -node Chebychev interpolation operator, $t^\ell, \ell = 1, \dots, q$, are the **Chebychev nodes** in $D_x := [a, b]$ given by

$$t^j := a + \frac{1}{2}(b - a) \left(\cos \left(\frac{2j-1}{2q} \pi \right) + 1 \right), \quad j = 1, \dots, q. \tag{2.2.1.36}$$

and the functions L_ℓ are the Lagrange polynomials (2.2.1.34) for these nodes, that is, the cardinal basis functions for Chebychev interpolation in $\{t^j\}_{j=1}^q$

§2.2.2.13 (Simple 1D Chebychev interpolation error estimates) Write $\mathcal{I}_\mathcal{T}$ for the well-defined polynomial interpolation operator into \mathcal{P}_{q-1} based on the node set $\mathcal{T} := \{t^1, \dots, t^q\} \subset [-1, 1] \subset \mathbb{R}$. The fundamental error representation from [NumCSE Thm. 6.2.2.15] for $f \in C^q([-1, 1])$

$$(f - \mathcal{I}_\mathcal{T} f)(x) = \frac{f^{(q)}(\tau(x))}{q!} \cdot \prod_{k=1}^q (x - t^k) \quad \text{for some } \tau(x) \in [-1, 1], \tag{2.2.2.14}$$

yields the bound of [NumCSE Eq. (6.2.2.22)]:

$$\|f - \mathcal{I}_q f\|_{L^\infty([-1,1])} \leq \frac{1}{q!} \|f^{(q)}\|_{L^\infty([-1,1])} \max_{t \in [-1,1]} |(t - t^1) \cdots (t - t^q)|. \quad (2.2.2.15)$$

For the special **Chebyshev nodes**

$$t^j := \cos\left(\frac{2j-1}{2q}\pi\right), \quad j = 1, \dots, q, \quad (2.2.1.36)$$

which are zeros of the Chebyshev polynomial T_q [NumCSE Def. 6.2.3.3] we know

$$|(t - t^1) \cdots (t - t^q)| = |2^{1-q} T_q(t)| \leq 2^{1-q} \quad \forall -1 \leq t \leq 1. \quad (2.2.2.16)$$

Plugging this into (2.2.2.15), we get

$$\|f - \widehat{\mathcal{I}}_q f\|_{L^\infty([-1,1])} \leq \frac{2^{1-q}}{q!} \|f^{(q)}\|_{L^\infty([-1,1])}, \quad (2.2.2.17)$$

where we wrote $\widehat{\mathcal{I}}_q$ for the Chebyshev interpolation operator on the reference interval $[-1, 1]$ based on q interpolation nodes.

Affine transformation to a general interval $[a, b]$, $a < b$, [NumCSE § 6.2.1.26] finally leads to an error estimate for the q -node Chebyshev interpolation.

Lemma 2.2.2.18. Chebyshev interpolation error estimate

For any $f \in C^q([a, b])$ the q -node polynomial Chebyshev interpolation operator $\mathcal{I}_{q,[a,b]}$ on the interval $[a, b]$, $a < b$ admits the error estimate

$$\|f - \mathcal{I}_{q,[a,b]} f\|_{L^\infty([a,b])} \leq 2^{1-2q} (b-a)^q \frac{1}{q!} \|f^{(q)}\|_{L^\infty([a,b])}. \quad (2.2.2.19)$$

┘

Now we consider the singular logarithmic kernel $G(x, y) = -\log|x - y|$ on a box $B := D_x \times D_y$, $D_x, D_y \subset \mathbb{R}$, $D_x \cap D_y = \emptyset$, $D_x := [a, b]$, see Fig. 74. There we approximate it by Chebyshev interpolation in x -direction, cf. (2.2.1.31)

$$\widetilde{G}(x, y) := \mathcal{I}_{q,[a,b]} \{x \mapsto -\log|x - y|\}(x, y), \quad (x, y) \in B. \quad (2.2.2.20)$$

Next we apply the estimate of Lemma 2.2.2.18 to $x \mapsto -\log|x - y|$. More precisely, we use Lemma 2.2.2.18 for arbitrary, but fixed $y \in D_y$ and

$$f(x) := G(x, y) \stackrel{(2.2.1.20)}{\blacktriangleright} |f^{(q)}(x)| = \left| \frac{\partial^q G}{\partial x^q}(x, y) \right| = \frac{(q-1)!}{|y-x|^q}, \quad x \neq y.$$

Plugging this into (2.2.2.19) and observing that $|x - y| \geq \text{dist}(D_x; D_y)$ and $\text{diam}(D_x) = b - a$ yields the final estimate

$$\|G - \widetilde{G}\|_{L^\infty(D_x \times D_y)} \leq \frac{2}{q} \left(\frac{\text{diam}(D_x)}{4 \text{dist}(D_x; D_y)} \right)^q \leq \frac{2}{q} \left(\frac{\eta(B)}{2} \right)^q, \quad (2.2.2.21)$$

where, again, we expressed the bound through the **admissibility measure** of the box B

$$\eta(B) := \frac{\max\{\text{diam}(D_x), \text{diam}(D_y)\}}{2 \text{dist}(D_x; D_y)}. \quad (2.2.2.7)$$

Unlike (2.2.2.7), for large η the above estimate (2.2.2.21) does not predict exponential convergence. This can be remedied by stronger estimates that we outline next.

§2.2.2.22 (Interpolation error estimates based on analytic extension) We start with a deep result of approximation theory already given in Thm. 1.4.3.70. Recall the concept of “analyticity” of a function $D \subset \mathbb{C} \rightarrow \mathbb{C}$ from Def. 1.4.3.68:

Definition 1.4.3.68. Analyticity of a function in \mathbb{C}

Let $D \subset \mathbb{C}$ be an *open* set in the complex plane. A function $f : D \rightarrow \mathbb{C}$ is called **analytic/holomorphic** in D , if f has a representation as a convergent power series in a neighborhood of every $z \in D$:

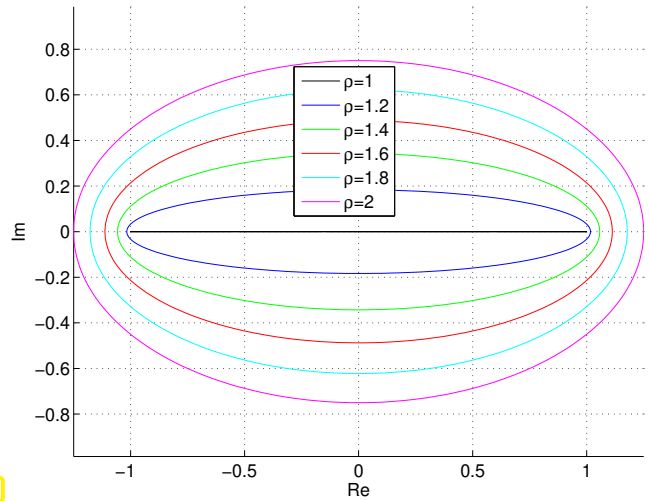
$$\forall z \in D : \exists r_z > 0, (a_k)_{k \in \mathbb{N}_0}, a_k \in \mathbb{C} : f(w) = \sum_{k=0}^{\infty} a_k (w - z)^k \quad \forall w : |z - w| < r_z .$$

Also remember the special closed curves in the complex plane called **Bernstein ellipses**:

$$\begin{aligned} \mathcal{E}_\rho &:= \{z \in \mathbb{C} : |z - 1| + |z + 1| = \rho + \rho^{-1}\} \\ &= \left\{ \begin{array}{l} z = \frac{1}{2}(\rho + \rho^{-1}) \cos \theta + \\ \quad i \frac{1}{2}(\rho - \rho^{-1}) \sin \theta, \\ 0 \leq \theta \leq 2\pi \end{array} \right\}, \end{aligned} \tag{1.4.3.69}$$

with a parameter $\rho > 0$ controlling the size of the ellipse. ▷

Fig. 75



They wrap around the reference interval $[-1, 1]$, have ± 1 as their focal points, and $\rho + \rho^{-1}$ and $\rho - \rho^{-1}$ as lengths of their long and short axes, respectively.

Theorem 2.2.2.23. Chebyshev interpolation of analytic functions [NumCSE Eq. (6.2.3.28)]

If $f : D \subset \mathbb{C} \rightarrow \mathbb{C}$ is *analytic* in the interior of the Bernstein ellipse \mathcal{E}_ρ , $\rho > 1$, and bounded on \mathcal{E}_ρ , then

$$\inf_{p \in \mathcal{P}_m} \|f - \hat{1}_m f\|_{L^\infty([-1,1])} \leq \frac{8}{\pi} \frac{1}{(\rho^m - \rho^{-1})(\rho + \rho^{-1} - 2)} \max_{z \in \mathcal{E}_\rho} |f(z)| \quad \text{for all } m \in \mathbb{N}, \tag{2.2.2.24}$$

where $\hat{1}_m : C^0([-1, 1]) \rightarrow \mathcal{P}_m$ stands for the Chebyshev interpolation operators on $[-1, 1]$.

Again, we point out **exponential convergence** of the maximum norm of the minimal approximation error over \mathcal{P}_m as the degree $m \rightarrow \infty$.

To apply Thm. 2.2.2.23 to the Chebyshev interpolation of $x \mapsto -\log(y - x)$ on $D_x := [a, b]$, $a < b < y$, we first employ an affine pullback to the reference interval [NumCSE § 6.2.1.14] and obtain

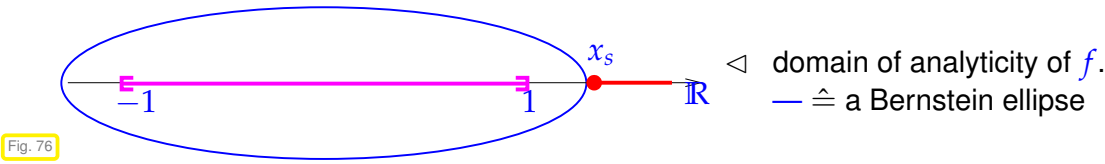
$$f(\hat{x}) := -\log\left(y - \left(\frac{1}{2}(b - a)\hat{x} + \frac{1}{2}(b + a)\right)\right), \quad -1 \leq \hat{x} \leq 1. \tag{2.2.2.25}$$

f is analytic on $\mathbb{C} \setminus [x_s, \infty[$ with

$$\begin{aligned} x_s &:= \frac{2}{b-a} \left(y - \frac{1}{2}(b+a) \right) \geq \frac{2}{\text{diam}(D_x)} \left(\text{dist}(D_x; D_y) + \frac{1}{2} \text{diam}(D_x) \right) \\ &= \frac{2 \text{dist}(D_x; D_y)}{\text{diam}(D_x)} + 1 \geq \frac{1}{\eta} + 1 > 1, \end{aligned}$$

where η is the admissibility measure of the box $D_x \times D_y$,

$$\eta = \eta(B) := \frac{\max\{\text{diam}(D_x), \text{diam}(D_y)\}}{2 \text{dist}(D_x; D_y)}. \tag{2.2.2.7}$$



The range of possible size parameters ρ for the Bernstein ellipses \mathcal{E}_ρ is

$$\rho > 1: \quad \rho + \rho^{-1} < 2x_s, \quad \text{satisfied for} \quad 1 < \rho < \frac{1 + \eta + \sqrt{1 + \eta}}{\eta}. \tag{2.2.2.26}$$

This confirms exponential convergence of Chebychev interpolation of $x \mapsto -\log(y - x)$ on D_x with respect to the polynomial degree, for any $y \notin D_x$, with a rate governed by the admissibility measure η .

For asymptotically smooth singular kernels the admissibility measure η governs the speed of asymptotic exponential convergence of rank- q separable approximations by uni-directional Chebychev interpolation.

2.2.2.3 Estimates for Bi-Directional Interpolation

For separable approximation by bi-directional interpolation as elaborated in Section 2.2.1.3 we have to study the interpolation error

$$G(x, y) - \tilde{G}(x, y) = (\text{Id} - \mathcal{I}^x \otimes \mathcal{I}^y) \{ (x, y) \mapsto G(x, y) \}.$$

§2.2.2.27 (Error estimates for bi-directional interpolation) We revisit the setting of § 2.2.1.44 for bi-directional interpolation based on two linear interpolation operators $\mathcal{I}^x : C^0(D_x) \rightarrow V_x$ and $\mathcal{I}^y : C^0(D_y) \rightarrow V_y$ with interpolation nodes $\mathbf{t}_x^1, \dots, \mathbf{t}_x^{q_x}$, $q_x := \dim V_x$, $\mathbf{t}_y^1, \dots, \mathbf{t}_y^{q_y}$, $q_y := \dim V_y$. Writing b_k^x , $k = 1, \dots, q_x$, and b_j^y , $j = 1, \dots, q_y$ for the associated cardinal functions we can express, compare (2.2.1.28),

$$\begin{aligned} (\mathcal{I}^x g)(\mathbf{x}) &= \sum_{k=1}^{q_x} g(\mathbf{t}_x^k) b_k^x(\mathbf{x}), \quad g \in C^0(D_x), \\ (\mathcal{I}^y h)(\mathbf{y}) &= \sum_{j=1}^{q_y} h(\mathbf{t}_y^j) b_j^y(\mathbf{y}), \quad h \in C^0(D_y). \end{aligned} \tag{2.2.2.28}$$

Interpolation error estimates are closely linked to the stability of the interpolation operators, which is measured by their operator norm(s). The operator norm associated with the maximum norm on function space has a special name.

Definition 2.2.2.29. Lebesgue constant [NumCSE Lemma 5.2.4.10]

The **Lebesgue constant** of a linear interpolation operator $I : C^0(D) \rightarrow V_x$ according to Def. 2.2.1.25 with associated cardinal functions b_ℓ , $\ell = 1, \dots, q$, is the number

$$\lambda(I) := \sum_{\ell=1}^q \|b_\ell\|_{L^\infty(D)} .$$

As an immediate consequence of the definition and \triangle -inequality we mention

$$\|If\|_{L^\infty(D)} \leq \lambda(I) \|f\|_{L^\infty(D)} \quad \forall f \in C^0(D) . \quad (2.2.2.30)$$

Next we rewrite the tensor-product interpolation operator as a composition of unidirectional interpolation operators $I_{2D}^* : C^0(D_x \times D_y) \rightarrow C^0(D_x \times D_y)$, $* = x, y$. We introduce

$$\begin{aligned} (I_{2D}^x f)(x, y) &= \sum_{k=1}^{q_x} f(t_x^k, y) b_k^x(x) , \\ (I_{2D}^y f)(x, y) &= \sum_{k=1}^{q_y} f(x, t_y^k) b_k^y(y) , \end{aligned} \quad f \in C^0(D_x \times D_y) .$$

(2.2.1.45) \blacktriangleright $I^x \otimes I^y = I_{2D}^x \circ I_{2D}^y$ on $C^0(D_x \times D_y)$. (2.2.2.31)

By the very definition of the Lebesgue constant in Def. 2.2.2.29 we conclude

$$\begin{aligned} \|I_{2D}^x f\|_{L^\infty(D)} &\leq \max_{x \in D_x} \max_{y \in D_y} \sum_{k=1}^{q_x} |f(t_x^k, y) b_k^x(x)| \\ &\leq \sum_{k=1}^{q_x} \|f\|_{L^\infty(D_x \times D_y)} \|b_k^x\|_{L^\infty(D_x)} = \lambda(I^x) \|f\|_{L^\infty(D_x \times D_y)} . \end{aligned} \quad (2.2.2.32)$$

This means that $\lambda(I^*)$ provides a bound for the (operator) norm of I_{2D}^* .

Owing to (2.2.2.31) we can separate interpolation directions:

$$I^x \otimes I^y - \text{Id} = I_{2D}^x \circ I_{2D}^y - I_{2D}^x \circ \text{Id} + I_{2D}^x \circ \text{Id} - \text{Id} = I_{2D}^x (I_{2D}^y - \text{Id}) + (I_{2D}^x - \text{Id}) \circ \text{Id} ,$$

and the \triangle -inequality gives the following estimate for $f \in C^0(D_x \times D_y)$:

$$\begin{aligned} \|(I^x \otimes I^y - \text{Id})f\|_{L^\infty(D_x \times D_y)} &\leq \|I_{2D}^x (I_{2D}^y - \text{Id})f\|_{L^\infty(D_x \times D_y)} + \|(I_{2D}^x - \text{Id})f\|_{L^\infty(D_x \times D_y)} \\ &\leq \lambda(I^x) \|(I_{2D}^y - \text{Id})f\|_{L^\infty(D_x \times D_y)} + \|(I_{2D}^x - \text{Id})f\|_{L^\infty(D_x \times D_y)} . \end{aligned}$$

Let us elucidate the contribution of uni-directional interpolation errors (highlighted with color)

$$\begin{aligned} \|(I^x \otimes I^y - \text{Id})f\|_{L^\infty(D_x \times D_y)} &\leq \lambda(I^x) \max_{x \in D_x} \left\{ \max_{y \in D_y} \left| f(x, y) - \sum_{j=1}^{q_y} f(x, t_y^j) b_j^y(y) \right| \right\} + \\ &\quad \max_{y \in D_y} \left\{ \max_{x \in D_x} \left| f(x, y) - \sum_{k=1}^{q_x} f(t_x^k, y) b_k^x(x) \right| \right\} . \end{aligned} \quad (2.2.2.33)$$

Hence, estimates of the interpolation error of I^x and I^y when applied to the functions $x \mapsto f(x, y)$ and $y \mapsto f(x, y)$, respectively, permit us to estimate the interpolation error for $I^x \otimes I^y$, if they are **uniform** in the other argument. ┘

Let us apply the estimate (2.2.2.33) for $d = 1$ to $G(x, y) = \log(y - x)$, $x \in D_x$, $y \in D_y$, $y > x$, in the situation of Fig. 74 with well separated intervals D_x and D_y . We rely on one-dimensional q -node Chebychev interpolation I_q on both D_x and D_y . From [NumCSE Rem. 6.2.3.19] we recall the deep result that in the case of Chebychev interpolation the Lebesgue constant is bounded as

$$\lambda(I_q) \leq \frac{2}{\pi} \log(2 + q) + 1 \quad \forall q \in \mathbb{N}. \tag{2.2.2.34}$$

The function $x \mapsto G(x, y)$ has an analytic extension to the interior of the Bernstein ellipse \mathcal{E}_ρ for all $y \in D_y$, if $\rho > 1$ is chosen according to (2.2.2.26). Thus, invoking Thm. 2.2.2.23, we get exponential convergence of the interpolation error

$$\left\| \{x \mapsto ((\text{Id} - I_q^x)G)(x, y)\} \right\|_{L^\infty(D_x)} \quad \text{for } q \rightarrow \infty,$$

whose speed will be determined by ρ and, indirectly, by the admissibility measure of the box $D_x \times D_y$. Up to a logarithmic factor in q this will also hold for the total approximation error $\|G - \tilde{G}\|_{L^\infty(D_x \times D_y)}$.

EXPERIMENT 2.2.2.35 (Tensor-product Chebychev interpolation of singular kernel) For $d = 1$ we consider the singular asymptotically smooth kernel function

$$G(x, y) := \frac{1}{|x - y|}, \quad x \neq y.$$

We employ tensor-product Chebychev interpolation of degree $q - 1$ with q^2 interpolation nodes on the rectangular boxes

$$B_k := [0.55 + k \cdot 0.05, 0.75 + k \cdot 0.05] \times [0.25 - k \cdot 0.05, 0.45 - k \cdot 0.05], \quad k \in \{0, \dots, 5\}.$$

These boxes are shown in Fig. 77 and their admissibility measures $\eta(B_k)$ according to (2.2.2.7) are given in the following table:

k	0	1	2	3	4	5
$\eta(B_k)$	2.0	1.0	0.66	0.5	0.4	0.33

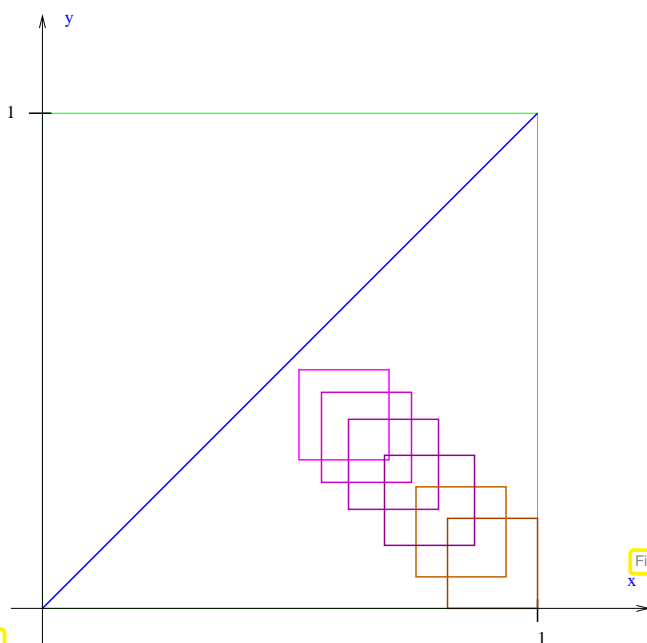


Fig. 77

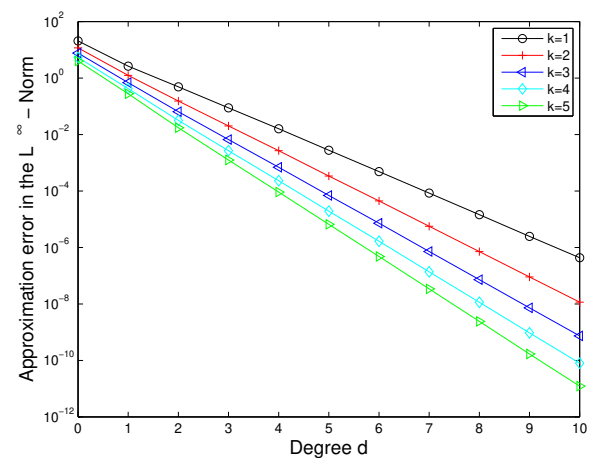


Fig. 78

In Fig. 78 we observe exponential convergence of $\|G - \tilde{G}\|_{L^\infty(B_k)}$ in the degree, the faster the larger k , which also corresponds to smaller admissibility measure of the box.

The previous experiment is repeated with the boxes

$$B_\xi := \left[\frac{1}{2}(\sqrt{2}-1)\xi + \frac{1}{2}, \frac{1}{2}(\sqrt{2}+1)\xi + \frac{1}{2} \right] \times \left[-\frac{1}{2}(\sqrt{2}-1)\xi + \frac{1}{2}, -\frac{1}{2}(\sqrt{2}+1)\xi + \frac{1}{2} \right],$$

$$\xi \in \{0.05, 0.09, 0.13, \dots, 0.41\},$$

whose size increases with increasing distance from the diagonal, keeping their admissibility measures $\eta(B_\xi)$ at the constant value $\sqrt{2}$.

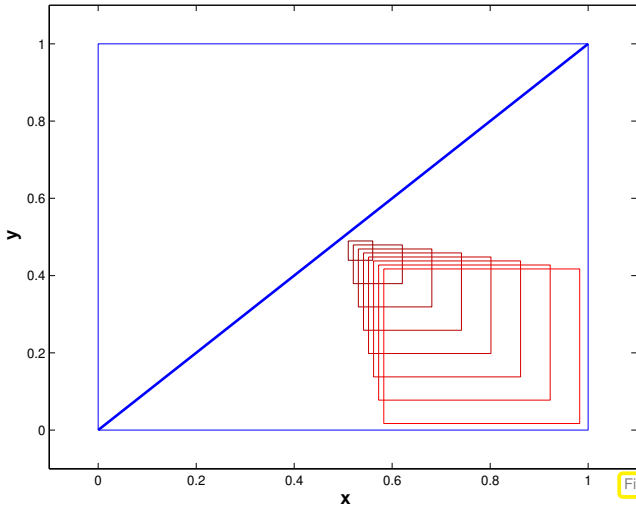


Fig. 79

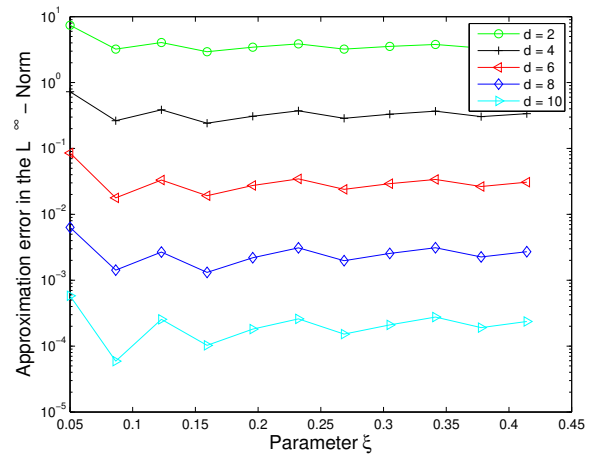


Fig. 80

We observe that the interpolation error $\|G - \tilde{G}\|_{L^\infty(B_\xi)}$ is almost constant for a family of rectangles with about the same admissibility measure. Exponential convergence in the degree is well preserved. \lrcorner

Inspired by the findings of our investigations into the separable approximation of asymptotically smooth singular kernels we will make a general assumption:

Assumption 2.2.2.36. Rank- q separable approximation on admissible boxes

For the kernel function $G : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ under consideration there is an increasing function $\delta : \mathbb{R}^+ \rightarrow [0, 1[$ such that for any disjoint closed sets $D_x, D_y \subset \mathbb{R}^d, D_x \cap D_y = \emptyset$, there is a family $\{\tilde{G}_q\}_{q \in \mathbb{N}}$ of rank- q separable approximations such that

$$\|G - \tilde{G}_q\|_{L^\infty(D_x \times D_y)} \leq \delta(\eta(D_x \times D_y))^q \quad \forall q \in \mathbb{N}, \tag{2.2.2.37}$$

where η is the admissibility measure from (2.2.2.7).

Review question(s) 2.2.2.38 (Approximation of kernel collocation matrices)

(Q2.2.2.38.A) [Sparse matrices] Consider a family of sparse square matrices with a most $K \in \mathbb{N}$ non-zero entries per column. Let $\mathbf{M} \in \mathbb{R}^{n,n}$ belong to that family and let $\tilde{\mathbf{M}}$ a suitable data-sparse approximation satisfying $\|\mathbf{M} - \tilde{\mathbf{M}}\|_F < \epsilon$. What are the exponents $p \in \mathbb{N}_0$ and $q \in \mathbb{N}_0$ in the asymptotic bounds

cost of storage/initialization of $\tilde{\mathbf{M}}$ $= O((m+n) \log^q(m+n) |\log^p \epsilon|)$ for $m, n \rightarrow \infty, \epsilon \rightarrow 0$?
 cost($\tilde{\mathbf{M}}$ \times vector)

(Q2.2.2.38.B) [Power series expansions] We consider the kernel function $G(x, y) = \frac{1}{x-y}, x, y \in \mathbb{R}, x \neq y$. For every $x \in \mathbb{R} (y \in \mathbb{R})$ determine the interval of convergence of the power series expansion of $y \mapsto G(x, y) (x \mapsto G(x, y))$.

(Q2.2.2.38.C) [Basis polynomials] Let $b_\ell, \ell = 1, \dots, p$, a *basis* of the space \mathcal{P}_p of uni-variate polynomials of degree $< p$. Show that the set

$$\{(x, y) \mapsto b_\ell(x)b_k(y)\}_{\ell,k=1}^p \subset C^0(\mathbb{R}^2)$$

is *linearly independent*.

(Q2.2.2.38.D) [Helmholtz kernel] Remember the notion of asymptotic smoothness of a kernel function:

A kernel function $G : (\mathbb{R}^d \times \mathbb{R}^d) \setminus \{(x, y) \in \mathbb{R}^d \times \mathbb{R}^d : x = y\} \rightarrow \mathbb{R}$ is called **asymptotically smooth**, if

- (i) $G \in C^\infty((\mathbb{R}^d \times \mathbb{R}^d) \setminus \{(x, y) \in \mathbb{R}^d \times \mathbb{R}^d : x = y\})$,
- (ii) and its derivatives satisfy the decay conditions

$$|D_y^\alpha G(x, y)| \leq C |\alpha|! \gamma^{|\alpha|} \frac{|G(x, y)|}{\|x - y\|^{|\alpha|}} \quad \forall \alpha \in \mathbb{N}_0^d, \quad \forall (x, y) \in \mathbb{R} \times \mathbb{R} \setminus \{(x, y) \in \mathbb{R}^d \times \mathbb{R}^d : x = y\}, \quad (2.2.2.2)$$

with constants $C > 0, \gamma > 0$ ($|\alpha| = |\alpha_1| + \dots + |\alpha_d|$).

Now consider the *singular* kernel function

$$G(x, y) = \frac{\exp(i\omega|x - y|)}{|x - y|}, \quad x, y \in \mathbb{R}, x \neq y, \quad \omega \in \mathbb{C}.$$

For which ω is it asymptotically smooth, for which will it fail to have this property?

△

2.3 Clustering Techniques

In this section we develop an algorithm that paves the way for data-sparse approximations in the sense of § 2.2.0.1 of kernel collocation matrices associated with **asymptotically smooth** (→ Rem. 2.2.2.1) **singular kernel functions**, see Ex. 2.1.3.3 for examples.

Throughout this section we assume that we are given, *cf.* Def. 2.1.3.1,

- ◆ an asymptotically smooth singular kernel function

$$G = G(x, y) \quad , \quad G \in C^\infty([0, 1]^d \times [0, 1]^d \setminus \{x = y\}),$$

allowing point evaluation for any admissible pair of arguments,

- ◆ collocation points $x^i \in [0, 1]^d, i = 1, \dots, n$, and $y^j \in [0, 1]^d, j = 1, \dots, m, m, n \in \mathbb{N}$.

The kernel function may be available only in procedural form as subroutine providing point evaluations.

2.3.1 Local Separable Approximation

Recall the **admissibility measure** of a box $B := D_x \times D_y, D_x, D_y \subset [0, 1]^d$, defined in (2.2.2.7):

$$\eta = \eta(B) := \frac{\max\{\text{diam}(D_x), \text{diam}(D_y)\}}{2 \text{dist}(D_x; D_y)}. \quad (2.2.2.7)$$

The bottom line of Section 2.2.2.1 and Section 2.2.2.2 for $d = 1$ was that rank- q separable approximation of asymptotically smooth singular kernel functions is possible with fast asymptotic *exponential convergence* for $q \rightarrow \infty$ on boxes $B := D_x \times D_y$ with *small admissibility measure* $\eta(B)$.

Idea: Partition $[0, 1]^d \times [0, 1]^d$ into boxes B_1, \dots, B_K , $K \in \mathbb{N}$, $B_k = D_x^k \times D_y^k$, $D_x, D_y \subset [0, 1]^d$ (also called a *tiling*)



$$[0, 1]^d \times [0, 1]^d = B_1 \cup \dots \cup B_K, \quad B_\ell \cap B_m = \emptyset, \text{ if } \ell \neq m,$$

such that

only $O(m + n)$ pairs (x^i, y^j) of collocation points are contained in boxes with an *admissibility measure* $\eta > \eta_0$, for prescribed $\eta_0 > 0$.

§2.3.1.1 (Near-field and far-field boxes) Formally we split the set $\mathcal{B} := \{B_1, \dots, B_K\}$ of boxes into η_0 -admissible boxes (“**far-field**” boxes) and remainder (“**near-field**” boxes):

$$\mathcal{B} = \mathcal{B}_{\text{far}} \cup \mathcal{B}_{\text{near}}, \quad \mathcal{B}_{\text{far}} := \{B \in \mathcal{B} : \eta(B) \leq \eta_0\}, \\ \mathcal{B}_{\text{near}} := \{B \in \mathcal{B} : \eta(B) > \eta_0\}.$$

Qualitative visualization of near-field \leftrightarrow far-field splitting of $[0, 1]^2$ for $d = 1$



Far-field boxes with admissibility measure $\leq \eta_0$.

Boxes abutting or close to the diagonal $\{x = y\}$ form the set of near-field boxes (not marked).

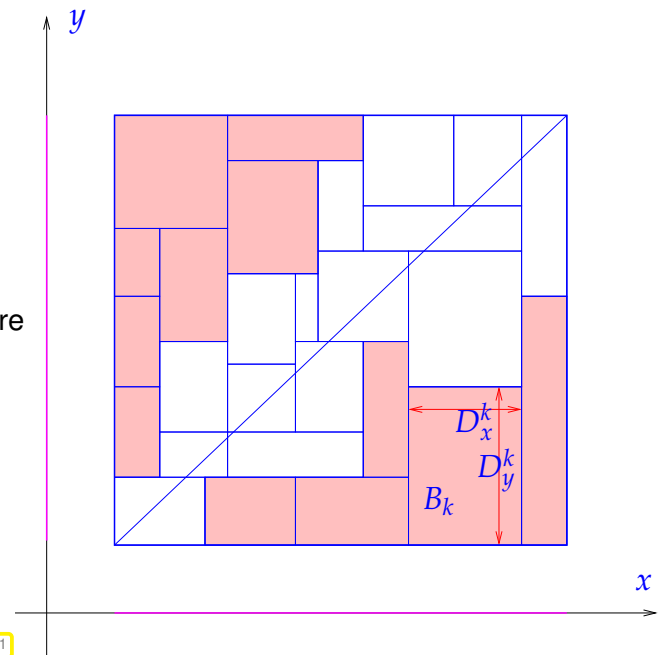
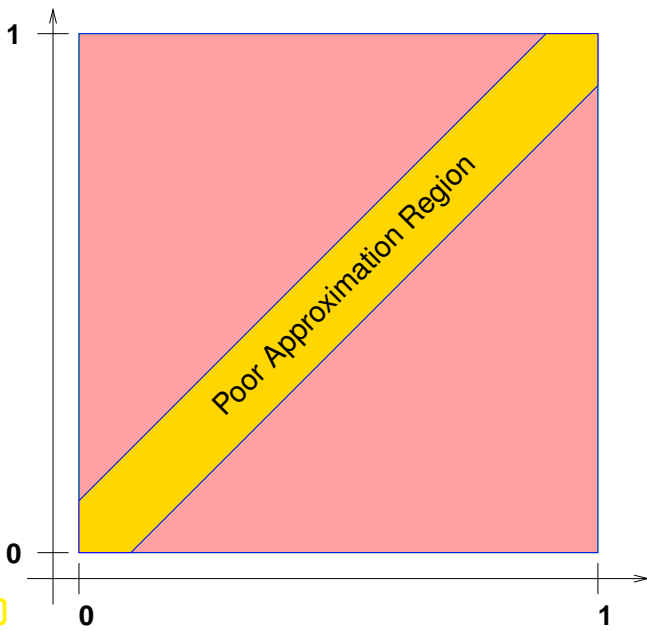


Fig. 81

What we have learned in Section 2.2, refer to Ass. 2.2.2.36:

Separable approximation in the far field

Rank- q separable approximation of G by expansion (\rightarrow Section 2.2.1.1) or interpolation (\rightarrow Section 2.2.1.2/Section 2.2.1.3) is possible on far-field boxes with exponentially decreasing error for $q \rightarrow \infty$.

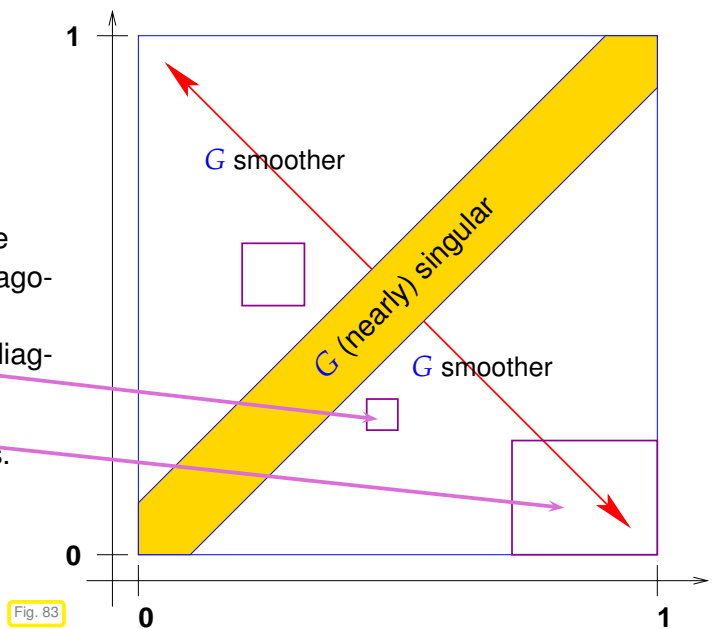


- ◁ A picture helping you to remember the location of near-field and far-field boxes in $d = 1$
- Near field:** No exponentially convergent separable approximation possible
- Far field:** Expansion or interpolation provide exponentially convergent separable approximation.

Meaning of $\eta(B) \leq \eta_0$ for size of far-field boxes:

- Accurate separable approximation possible
 - only on small rectangles near the diagonal
 - also on large rectangles far from the diagonal

Aiming for $\eta(B) \approx \eta_0$ fixes size of far-field boxes.



§2.3.1.3 (Block partitioning of the kernel collocation matrix) Assume a partitioning of $[0, 1]^d \times [0, 1]^d$ into boxes $B_k := D_x^k \times D_y^k \subset [0, 1]^d \times [0, 1]^d, k = 1, \dots, K, D_x^k, D_y^k \subset [0, 1]^d$:

$$[0, 1]^d \times [0, 1]^d = B_1 \cup \dots \cup B_K, \quad B_\ell \cap B_m = \emptyset, \quad \text{if } \ell \neq m.$$

Based on the given collocation points $\mathbf{x}^i \in [0, 1]^d, i = 1, \dots, n, \mathbf{y}^j \in [0, 1]^d, j = 1, \dots, m$, this induces a **block-partitioning** of the kernel collocation matrix $\mathbf{M} = [G(\mathbf{x}^i, \mathbf{y}^j)]_{i,j} \in \mathbb{R}^{n,m}$. Set

$$\begin{aligned} I_k &:= \{i \in \{1, \dots, n\} : \mathbf{x}^i \in D_x^k\}, \\ J_k &:= \{j \in \{1, \dots, m\} : \mathbf{y}^j \in D_y^k\}, \end{aligned} \tag{2.3.1.4}$$

$$\begin{aligned} \blacktriangleright \quad \mathbf{D} &:= \{1, \dots, n\} \times \{1, \dots, m\} = \bigcup_{k=1}^K I_k \times J_k, \\ (I_\ell \times J_\ell) \cap (I_m \times J_m) &= \emptyset \Leftrightarrow \ell \neq m, \end{aligned} \tag{2.3.1.5}$$

and define the matrix blocks by

$$\mathbf{M}_k := \left[G(\mathbf{x}^i, \mathbf{y}^j) \right]_{\substack{i \in I_k \\ j \in J_k}} \in \mathbb{R}^{\#I_k \times \#J_k}, \quad k = 1, \dots, K. \tag{2.3.1.6}$$

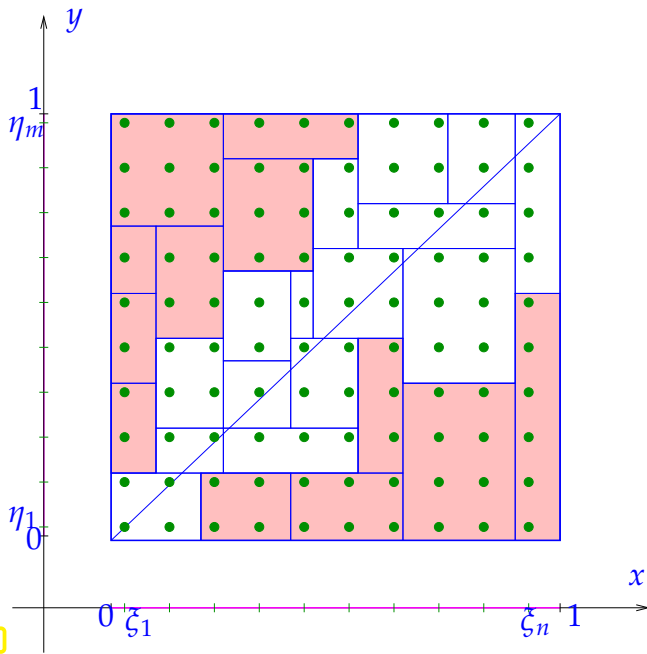


Fig. 84

For $d = 1$ and assuming sorted collocation points

$$0 \leq \xi_1 < \xi_2 < \dots < \xi_n \leq 1, \\ 0 \leq \eta_1 < \eta_2 < \dots < \eta_m \leq 1,$$

the geometric boxes B_k directly correspond to matrix blocks.

◁ • point $(\xi^i, \eta^j) \in \mathbb{R}^d \times \mathbb{R}^d \leftrightarrow$ entry of \mathbf{M}

However, block partitionings of \mathbf{M} (rearrangement of indices allowed) induced by a partitioning (2.3.1.5) of the set $\mathbb{D} := \{1, \dots, n\} \times \{1, \dots, m\}$ of index pairs are more general than geometric partitionings of $[0, 1]^d \times [0, 1]^d$. Therefore, we can now formulate the following objective (in not entirely rigorous terms).

Goal: admissible and efficient block partitioning

Find a partitioning of $\mathbb{D} = \{1, \dots, n\} \times \{1, \dots, m\}$

$$\mathbb{D} = \bigcup_{k=1}^K I_k \times J_k, \quad I_k \subset \{1, \dots, n\}, \quad J_k \subset \{1, \dots, m\}, \quad (I_\ell \times J_\ell) \cap (I_m \times J_m) = \emptyset \Leftrightarrow \ell \neq m, \quad (2.3.1.8)$$

such that, with a **near-field – far-field splitting** of the index set $\{1, \dots, K\}$

$$\mathbb{F}_{\text{near}} \cap \mathbb{F}_{\text{far}} = \emptyset, \quad \mathbb{F}_{\text{near}} \cup \mathbb{F}_{\text{far}} = \{1, \dots, K\}, \quad (2.3.1.9)$$

the cardinalities of the index sets satisfy for some $p \in \mathbb{N}_0$

$$\sum_{k \in \mathbb{F}_{\text{far}}} \#I_k + \#J_k = O((m+n) \log^p(m+n)), \quad (2.3.1.10)$$

$$\sum_{k \in \mathbb{F}_{\text{near}}} \#I_k \cdot \#J_k = O((m+n) \log^p(m+n)), \quad (2.3.1.11)$$

asymptotically “for $n, m \rightarrow \infty$ ” and some fixed $p \in \mathbb{N}$.

Remember our bid for data-sparse approximation (\rightarrow § 2.2.0.1) by low-rank approximation of far-field blocks to appreciate (2.3.1.10) and (2.3.1.11).

To characterize the sets \mathbb{F}_{near} (“near-field” blocks of index pairs) and \mathbb{F}_{far} (“far-field” block of index pairs), we define mutual admissibility of two sets of indices. To prepare its statement, we introduce the concept of bounding boxes.

Definition 2.3.1.12. Bounding box of an index set

The x/y -**bounding boxes** of index sets are

$$I \subset \{1, \dots, n\}: \text{box}_x(I) := \bigotimes_{\ell=1}^d [\min\{x_\ell^i\}_{i \in I}, \max\{x_\ell^i\}_{i \in I}] \subset \mathbb{R}^d,$$

$$J \subset \{1, \dots, m\}: \text{box}_y(J) := \bigotimes_{\ell=1}^d [\min\{y_\ell^j\}_{j \in J}, \max\{y_\ell^j\}_{j \in J}] \subset \mathbb{R}^d.$$

This makes it possible to link an index set with a geometric box.

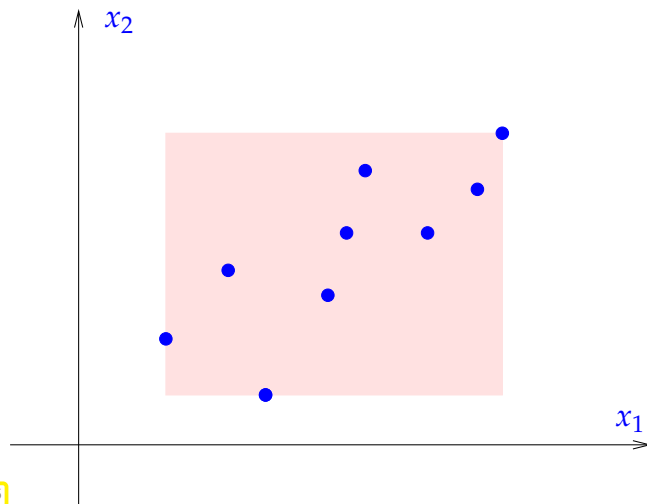


Fig. 85

◁ Axiparallel bounding box of a set $\{x^i\} \subset \mathbb{R}^2$ of points \bullet in the plane ($d = 2$).

Bounding boxes are needed to invoke geometric admissibility measure η , which is an essential ingredient of Ass. 2.2.2.36.

Definition 2.3.1.13. Admissibility of index sets

Given $\eta_0 > 0$ we call the product $I \times J$ of two index sets $I \subset \{1, \dots, n\}$ and $J \subset \{1, \dots, m\}$ η_0 -**admissible**, if

$$\eta(\text{box}_x(I) \times \text{box}_y(J)) := \frac{\max\{\text{diam}(\text{box}_x(I)), \text{diam}(\text{box}_y(J))\}}{2 \text{dist}(\text{box}_x(I); \text{box}_y(J))} \leq \eta_0,$$

where η is the admissibility measure from (2.2.2.7).

Here the distance of two sets $\subset \mathbb{R}^d$ has the natural meaning

$$\text{dist}(X; Y) := \max\{\|x - y\| : x \in X, y \in Y\}, \quad X, Y \subset \mathbb{R}^d. \quad (2.3.1.14)$$

This gives a rigorous criterion to be met by the far field:

Definition 2.3.1.15. Far-field blocks of index pairs

Given $\eta_0 > 0$ and a partitioning of $\mathbb{D} := \{1, \dots, n\} \times \{1, \dots, m\}$

$$\mathbb{D} = \bigcup_{k=1}^K I_k \times J_k, \quad I_k, J_k \subset \mathbb{N}, \quad (I_\ell \times J_\ell) \cap (I_m \times J_m) = \emptyset \quad \Leftrightarrow \quad \ell \neq m, \quad (2.3.1.8)$$

a corresponding η_0 -admissible far-field set of products of index sets has to satisfy

$$\mathbb{F}_{\text{far}} := \{k \in \{1, \dots, K\} : I_k \times J_k \text{ } \eta_0\text{-admissible}\}. \quad (2.3.1.16)$$

┘

§2.3.1.17 (From block partitioning to local low-rank compression) We assume that we are given a partitioning of $\mathbb{D} := \{1, \dots, n\} \times \{1, \dots, m\}$

$$\mathbb{D} = \bigcup_{k=1}^K I_k \times J_k, \quad I_k, J_k \subset \mathbb{N}, \quad (I_\ell \times J_\ell) \cap (I_m \times J_m) = \emptyset \quad \Leftrightarrow \quad \ell \neq m, \quad (2.3.1.8)$$

and a near-field – far-field splitting $\mathbb{F}_{\text{near}} \cap \mathbb{F}_{\text{far}} = \emptyset$, $\mathbb{F}_{\text{near}} \cup \mathbb{F}_{\text{far}} = \{1, \dots, K\}$, with an η_0 -admissible far field \mathbb{F}_{far} according to Def. 2.3.1.15.

Appealing to **Ass. 2.2.2.36**, for every $I_k \times J_k \in \mathbb{F}_{\text{far}}$ and $q \in \mathbb{N}$, we can find a q -separable approximation \tilde{G}_q^k of G (depending on k , of course) such that

$$\left\| G - \tilde{G}_q^k \right\|_{L^\infty(B_k)} \leq \delta(\eta_0)^q, \quad B_k := \text{box}_x(I_k) \times \text{box}_y(J_k), \quad (2.3.1.18)$$

where $\delta : \mathbb{R}^+ \rightarrow [0, 1[$ is the function introduced in Ass. 2.2.2.36.

Based on \tilde{G} we approximate the matrix blocks associated with $I_k \times J_k \in \mathbb{F}_{\text{far}}$ by rank- q matrices represented by their factors according to Lemma 2.2.1.3

$$(\mathbf{M})_{\substack{i \in I_k \\ j \in J_k}} \approx \left(\tilde{\mathbf{M}}_q \right)_{\substack{i \in I_k \\ j \in J_k}} := \left[\tilde{G}_q^k(\mathbf{x}^i, \mathbf{y}^j) \right]_{\substack{i \in I_k \\ j \in J_k}} = \mathbf{U}_k \cdot \mathbf{V}_k^\top, \quad \mathbf{U}_k \in \mathbb{R}^{\#I_k, q}, \quad \mathbf{V}_k \in \mathbb{R}^{\#J_k, q}. \quad (2.3.1.19)$$

If $I_k \times J_k \in \mathbb{F}_{\text{near}}$, then the corresponding block of \mathbf{M} is stored without any approximation:

$$I_k \times J_k \in \mathbb{F}_{\text{near}} \quad \Rightarrow \quad \left(\tilde{\mathbf{M}}_q \right)_{\substack{i \in I_k \\ j \in J_k}} := (\mathbf{M})_{\substack{i \in I_k \\ j \in J_k}} = \left[G(\mathbf{x}^i, \mathbf{y}^j) \right]_{\substack{i \in I_k \\ j \in J_k}}. \quad (2.3.1.20)$$

From (2.2.1.4) we conclude

$$\begin{aligned} I_k \times J_k \in \mathbb{F}_{\text{far}} &\Rightarrow \text{storage}\left(\left(\tilde{\mathbf{M}}\right)_{\substack{i \in I_k \\ j \in J_k}}\right) = q(\#I_k + \#J_k), \\ I_k \times J_k \in \mathbb{F}_{\text{near}} &\Rightarrow \text{storage}\left(\left(\tilde{\mathbf{M}}\right)_{\substack{i \in I_k \\ j \in J_k}}\right) = \#I_k \cdot \#J_k. \end{aligned}$$

This is the rationale behind the goals (2.3.1.10) and (2.3.1.11) stated above.

Storage requirements: Local low-rank compression

The approximate kernel collocation matrix $\widetilde{\mathbf{M}}$ defined by (2.3.1.19) and (2.3.1.20) satisfies

$$\text{storage}(\widetilde{\mathbf{M}}) = \sum_{k \in \mathbb{F}_{\text{far}}} q(\#I_k + \#J_k) + \sum_{k \in \mathbb{F}_{\text{near}}} \#I_k \cdot \#J_k. \quad (2.3.1.22)$$

(Short notation: $k \in \mathbb{F}_* \Leftrightarrow I_k \times J_k \in \mathbb{F}_*$)

The following two code snippets present a possible internal representation of $\widetilde{\mathbf{M}}$ in C++ code based on EIGEN (using namespace `Eigen`; assumed).

C++ code 2.3.1.23: Data structures for blocks of a local low-rank compressed matrix

```

2 // Rank-q matrix block in factorized form
3 template <int q>
4 struct FarFieldBlock {
5     const std::vector<int> i_idx, j_idx;           // contained indices
6     Eigen::Matrix<double, Eigen::Dynamic, q> U, V; // low-rank factors
7 };
8
9 // Submatrix; no special structure assumed
10 struct NearFieldBlock {
11     const std::vector<int> i_idx, j_idx; // contained indices
12     Eigen::MatrixXd Mloc;               // matrix block
13 };

```

C++ code 2.3.1.24: Data structures for low-rank compressed matrix

```

2 template <int q>
3 class PartMatrix {
4     public:
5     PartMatrix(size_t _n, size_t _m);
6     // Matrix×vector operation
7     Eigen::VectorXd operator*(const Eigen::VectorXd &v) const;
8
9     private:
10    size_t m, n; // dimensions of matrix
11    std::vector<FarFieldBlock<q>> farField;
12    std::vector<NearFieldBlock> nearField;
13 };

```

We are not only interested in economical use of memory, but also in fast execution of matrix×vector multiplications. For local low-rank compressed matrices like $\widetilde{\mathbf{M}}$ we get the crucial hint from

$$\text{rank}(\mathbf{M}) = q \implies \text{Cost}(\mathbf{M} \times \text{vector}) = O(q(n+m)) \quad \text{for } n, m \rightarrow \infty, \quad (2.2.1.5)$$

which, again, relies on the factorized form of rank- q matrices. This is available in the data structures of Code 2.3.1.23 and, thus, an efficient implementation of $\widetilde{\mathbf{M}} \times \text{vector}$ is straightforward:

C++ code 2.3.1.25: Matrix×vector multiplication for low-rank compressed matrix

```

2 // Partitioned n×m-matrix split in near-field and
3 // far-field blocks, the latter of rank q
4 template <int q>
5 Eigen::VectorXd PartMatrix<q>::operator*(const Eigen::VectorXd &v) const {
6     if (v.size() != m) throw(std::runtime_error("Size mismatch in *"));

```

```

7  Eigen::VectorXd y(n);
8  y.setZero(); // std::vector for returning result
9  // Traverse far field boxes
10 for (const FarFieldBlock<q> &B : farField) {
11     // Get no. of x and y collocation points in box
12     const size_t nB = B.i_idx.size();
13     const size_t mB = B.j_idx.size();
14     // Obtain values of argument std::vector corresponding to y-points
15     Eigen::VectorXd tmp(mB);
16     for (int j = 0; j < mB; j++) tmp(j) = v(B.j_idx[j]);
17     // Multiply std::vector with low-rank matrix: Effort #Ik + #Jk
18     Eigen::VectorXd res(nB);
19     res = B.U * (B.V.transpose() * tmp);
20     // Accumlate result into components of result std::vector
21     for (int i = 0; i < nB; i++) y(B.i_idx[i]) += res(i);
22 }
23 // Traverse near field boxes
24 for (const NearFieldBlock &B : nearField) {
25     // Get no. of x and y collocation points in box
26     const size_t nB = B.i_idx.size();
27     const size_t mB = B.j_idx.size();
28     // Obtain values of argument std::vector corresponding to y-points
29     Eigen::VectorXd tmp(mB);
30     for (int j = 0; j < mB; j++) tmp(j) = v(B.j_idx[j]);
31     // Multiply std::vector with local collocation matrix
32     Eigen::VectorXd res(nB);
33     res = B.Mloc * tmp;
34     // Accumlate result into components of result std::vector
35     for (int i = 0; i < nB; i++) y(B.i_idx[i]) += res(i);
36 }
37 return (y); // (Move) return result vector
38 }

```

Local low-rank compression: Cost of Matrix × vector

The approximate kernel collocation matrix $\widetilde{\mathbf{M}}$ defined by (2.3.1.19) and (2.3.1.20) can be multiplied with a vector at a cost of

$$\text{cost}(\widetilde{\mathbf{M}} \times \text{vector}) = \sum_{k \in \mathbb{F}_{\text{far}}} q(\#I_k + \#J_k) + \sum_{k \in \mathbb{F}_{\text{near}}} \#I_k \cdot \#J_k. \quad (2.3.1.27)$$

Under **Ass. 2.2.2.36** we can easily estimate the deviation of $\widetilde{\mathbf{M}}$ according to (2.3.1.19) and (2.3.1.20) from the exact kernel collocation matrix \mathbf{M} :

$$(2.2.2.37) \Rightarrow \left\| \mathbf{M} - \widetilde{\mathbf{M}}_q \right\|_F \leq \sqrt{mn} \delta(\eta_0)^q \quad \forall q \in \mathbb{N}, \quad (2.3.1.28)$$

where we also used (2.2.1.14). Hence we can achieve

$$\forall 1 \gg \epsilon > 0: \quad q \geq \left\lceil \frac{\log \epsilon - \frac{1}{2} \log(mn)}{\log \delta(\eta_0)} \right\rceil \quad \blacktriangleright \quad \left\| \mathbf{M} - \widetilde{\mathbf{M}}_q \right\|_F \leq \epsilon. \quad (2.3.1.29)$$

► If for families of larger and larger sets of collocation points we find partitionings of \mathbb{D} according to

(2.3.1.8) satisfying (2.3.1.10) and (2.3.1.11),

$$\sum_{k \in \mathbb{F}_{\text{far}}} \#I_k + \#J_k = O((m+n) \log^p(m+n)),$$

$$\sum_{k \in \mathbb{F}_{\text{near}}} \#I_k \cdot \#J_k = O((m+n) \log^p(m+n)),$$

then local low-rank compression offers a data-sparse approximate representation of the kernel collocation matrices meeting the requirements of § 2.2.0.1. ┘

Now the key issue is to find a partitioning (2.3.1.8) of \mathbb{D} into far-field (\rightarrow Def. 2.3.1.15) and near-field product index sets, such that (2.3.1.10) and (2.3.1.11) are satisfied.

2.3.2 Cluster Trees

As has become clear in the previous section, the challenge is to find a partition of the set $\mathbb{D} := \{1, \dots, n\} \times \{1, \dots, m\}$ of index pairs (corresponding to the set of matrix entries) into products $I_k \times J_k$ of index sets $I_k \subset \{1, \dots, n\}$ and $J_k \subset \{1, \dots, m\}$ such that $\{I_k \times J_k\}_{k=1, \dots, K}$ permits an *economical* decomposition

$$\{I_k \times J_k\}_{k=1, \dots, K} = \mathbb{F}_{\text{far}} \cup \mathbb{F}_{\text{near}} \quad , \quad \mathbb{F}_{\text{far}} \cap \mathbb{F}_{\text{near}} = \emptyset,$$

where, in the context of approximating a kernel collocation matrix (\rightarrow Def. 2.1.3.1) \mathbb{F}_{far} is a valid η_0 -admissible far field according to Def. 2.3.1.15. By “*economical*” we subsume the requirements of § 2.2.0.1,

$$\sum_{k \in \mathbb{F}_{\text{far}}} \#I_k + \#J_k = O((m+n) \log^p(m+n)), \quad (2.3.1.10)$$

$$\sum_{k \in \mathbb{F}_{\text{near}}} \#I_k \cdot \#J_k = O((m+n) \log^p(m+n)), \quad (2.3.1.11)$$

considered in the limit $n, m \rightarrow \infty$ for families of collocation points.



Idea: (inspired by “tree code” algorithm presented in § 2.1.2.8)

Use tree based decomposition of \mathbb{D}

EXAMPLE 2.3.2.1 (Quadtree-based admissible tiling of unit square [Bör21, Sect. 2.4]) We recall from § 4.1.1.9 that a partition of \mathbb{D} can be induced by a tiling (a geometric partition) of the tensor-product domain $D_x \times D_y$. In this example we consider $D_x = D_y = [0, 1]$, $D_x \times D_y$ is the unit square $[0, 1]^2$. We also restrict ourselves to $m = n = 2^{L-1}$, $L \in \mathbb{N}$, and

$$\text{equidistant collocation points: } \zeta_i := \frac{i-1/2}{n} \quad , \quad \eta_j := \frac{j-1/2}{n} \quad , \quad i, j \in \{1, \dots, n\}. \quad (2.3.2.2)$$

Taking the cue from the clustering of stars in § 2.1.2.8, we propose the following *recursive construction* of a box tiling of $[0, 1] \times [0, 1]$.

Pseudocode 2.3.2.3: Geometric tiling

```

split( $B := [a, b] \times [c, d]$ ) {
  if  $|b - a| + |d - c| < \delta$  then return;
  if  $\eta(B) \leq \eta_0$  then
     $\mathbb{F}_{\text{far}} \leftarrow \mathbb{F}_{\text{far}} \cup B$  // Add to far field
  else {
    split( $[a, \frac{1}{2}(a + b)] \times [\frac{1}{2}(c + d), d]$ );
    split( $[\frac{1}{2}(a + b), b] \times [\frac{1}{2}(c + d), d]$ );
    split( $[a, \frac{1}{2}(a + b)] \times [c, \frac{1}{2}(c + d)]$ );
    split( $[\frac{1}{2}(a + b), b] \times [c, \frac{1}{2}(c + d)]$ );
  }
}
    
```

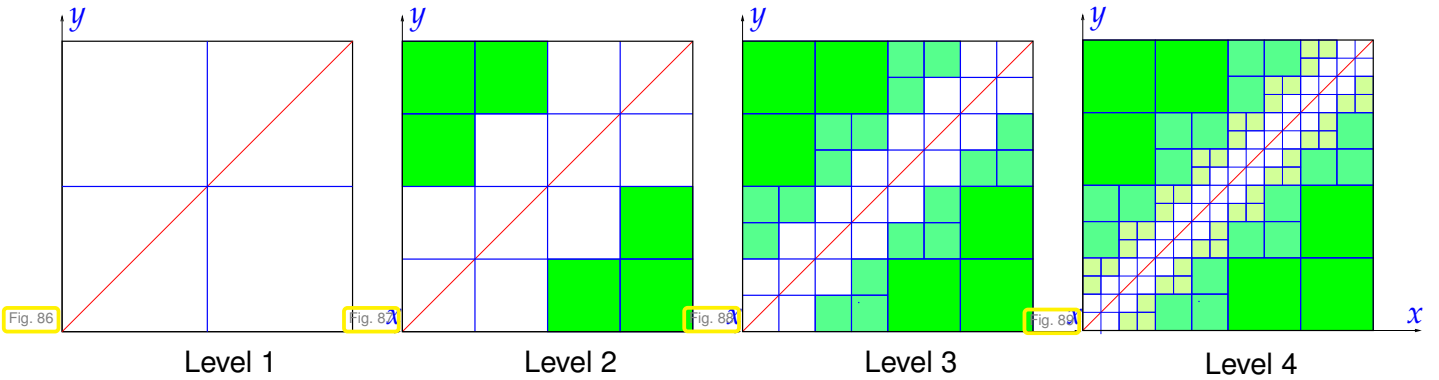
◁ Recursive construction of far-field/near-field tiling of box $\subset \mathbb{R}^2$. Invoke with **split**($[0, 1]^2$).

$\eta(B)$ is the **admissibility measure** of the box $B = D_x \times D_y$

$$\eta(B) := \frac{\max\{\text{diam}(D_x), \text{diam}(D_y)\}}{2 \text{dist}(D_x; D_y)}. \tag{2.2.2.7}$$

$\delta > 0$ controls termination of subdivision.

The following figures display $\frac{1}{2}$ -admissible boxes (shaded) as identified by the recursive algorithm on $[0, 1]$ with $\delta = \frac{1}{16}$.



We discuss the asymptotic cost of the induced block partitioning of the kernel collocation matrix for the admissibility parameter $\eta_0 = \frac{1}{2}$ (see Fig. 86–Fig. 89). For the chosen equidistant collocation points it is immediate that

$$\#\{i : \xi_i \in [a, b]\}, \#\{j : \eta_j \in [a, b]\} \sim |b - a|. \tag{2.3.2.4}$$

Hence, the “cost” of a box $[a, b] \times [c, d] \in \mathbb{F}_{\text{far}}$, that is the amount of memory to store the rank- q approximation of the associated block of the kernel collocation matrix is

$$\text{cost}([a, b] \times [c, d]) = qn \cdot (|b - a| + |d - c|) \text{ for } [a, b] \times [c, d] \in \mathbb{F}_{\text{far}}. \tag{2.3.2.5}$$

We stop the recursion when there is only a single pair of collocation points left in a box and set $\delta = 2^{-(L-1)}$. This implies that we will do $L - 1 = \log_2 n$ levels of recursive calls of **split**.

We also observe that, cf. Fig. 86–Fig. 89, on recursion level ℓ

$$\#\{\text{boxes cut by diagonal}\} = 2^\ell, \tag{2.3.2.6a}$$

$$\#\{\text{boxes touching the diagonal}\} = 2 \cdot (2^\ell - 1), \tag{2.3.2.6b}$$

$$\#\{\text{new boxes} \in \mathbb{F}_{\text{far}}\} = 6(2^{\ell-1} - 1), \tag{2.3.2.6c}$$

because each box cut by the diagonal spawns two of the same kind and two touching the diagonal on the next level, while each box touching the diagonal produces three far-field boxes.

The new far-field boxes on level ℓ contribute a total cost (proportional to their circumference by (2.3.2.5)) of $qn 2^{-\ell} \cdot 6(2^{\ell-1} - 1)$, $q \in \mathbb{N}$ the rank of the approximating matrix blocks, so that, by summing, for $n \rightarrow \infty$,

$$\text{cost}(B \in \mathbb{F}_{\text{far}}) = qn \cdot \sum_{\ell=1}^L 6(1 - 2^{-\ell}) = O(qnL), \quad L = O(\log n). \tag{2.3.2.7}$$

By our stopping criterion the cost of the near field boxes on the last level L is fixed “ $O(1)$ ”, which, by (2.3.2.6a) and (2.3.2.6b), yields the total cost $O(2^L) = O(n)$ for dealing with the near field. Evidently from (2.3.2.7), the storage required for the low-rank matrix blocks corresponding to far-field boxes is the dominant contribution.

By the reasoning of § 2.3.1.17 we conclude that for the chosen n equidistant collocation points

$$\text{storage}(\widetilde{\mathbf{M}}), \text{cost}(\mathbf{M} \times \text{vector}) = O(qn \log n) \quad \text{for } n \rightarrow \infty. \tag{2.3.2.8}$$

Ex. 2.3.2.1 relied on a *geometric* quadtree to define a tiling. This leads to an economical partitioning of \mathbb{D} , provided that the collocation points are equi-distributed. If this assumption is not satisfied, geometric tiling may fail. Therefore we aim for a *direct construction* of block-partitionings

$$\mathbb{D} = \bigcup_{k=1}^K I_k \times J_k, \quad I_k, J_k \subset \mathbb{N}, \quad (I_\ell \times J_\ell) \cap (I_m \times J_m) = \emptyset, \quad \text{if } \ell \neq m, \tag{2.3.1.8}$$

The algorithm will rely on tree data structures defining partitionings of the index sets $\{1, \dots, n\}$ and $\{1, \dots, m\}$.

§2.3.2.9 (Trees) From graph theory we recall the definition of a tree as a *cycle-free* directed graph.

Definition 2.3.2.10. Tree

Let \mathcal{V} be a finite **node/vertex set** and $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ an **edge set**. For some $r \in \mathcal{V}$ we call $\mathcal{T} := (\mathcal{V}, r, \mathcal{E})$ a **tree** with **root** r , if for each $v \in \mathcal{V}$ there is exactly one sequence $v_0, v_1, \dots, v_\ell \in \mathcal{V}$, $\ell \in \mathbb{N}_0$ such that

$$v_0 = r, \quad v_\ell = v, \quad (v_{i-1}, v_i) \in \mathcal{E} \quad \forall i = 1, \dots, \ell.$$

📎 Notation: We write $\text{root}(\mathcal{T})$ for the root of a tree \mathcal{T} . Regularly, we will use the same symbol, e.g. \mathcal{T} , for a tree and its node set.

We also refresh the rich terminology connected with trees:

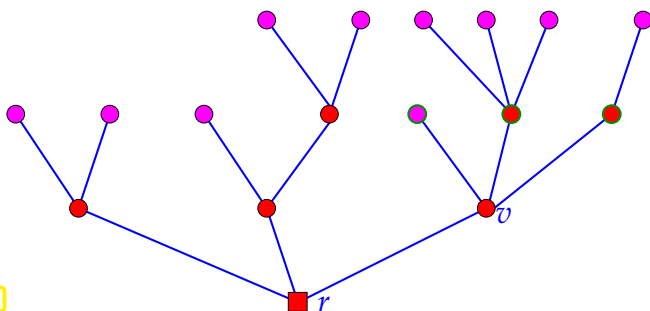
Definition 2.3.2.11. Concepts connected with trees

Let $\mathcal{T} := (\mathcal{V}, r, \mathcal{E})$ be a tree. For each $v \in \mathcal{V}$ we call

$$\text{sons}(v) := \{w \in \mathcal{V} : (v, w) \in \mathcal{E}\}$$

the set of **sons** of the node v . If $\text{sons}(v) = \emptyset$, then v is called a **leaf**.

If, for $v \in \mathcal{V}$, there is a $w \in \mathcal{V}$ such that $(w, v) \in \mathcal{E}$, that w is unique and called the **father** of v .



- ◁ Visualization of a tree
- : root $r = \text{root}(\mathcal{T})$ of \mathcal{T}
- : nodes/vertices of \mathcal{T}
- : leaves of tree
- : sons of v : $\text{sons}(v)$

Obviously, only the root of a tree has no father.

Fig. 90

A special kind of trees are **binary trees**, for which each node is either a leaf or has *two sons*:

$$\#\text{sons}(v) \in \{0, 2\} \quad \forall v \in \mathcal{T}.$$

A tree has a natural multilevel structure based on the distance of vertices from the root.

Definition 2.3.2.12. Level of nodes of tree

For a tree $\mathcal{T} := (\mathcal{V}, r, \mathcal{E})$ we can inductively define the function

$$\text{level} : \mathcal{V} \rightarrow \mathbb{N}_0 ,$$

$$\text{level}(v) := \begin{cases} 0 & , \text{ if } v = r(\text{root}) , \\ \text{level}(w) + 1 & , \text{ if } w \text{ is the father of } v , \end{cases} \quad \forall v \in \mathcal{V} .$$

This is valid definition, since, except for the root, every node has a unique father.

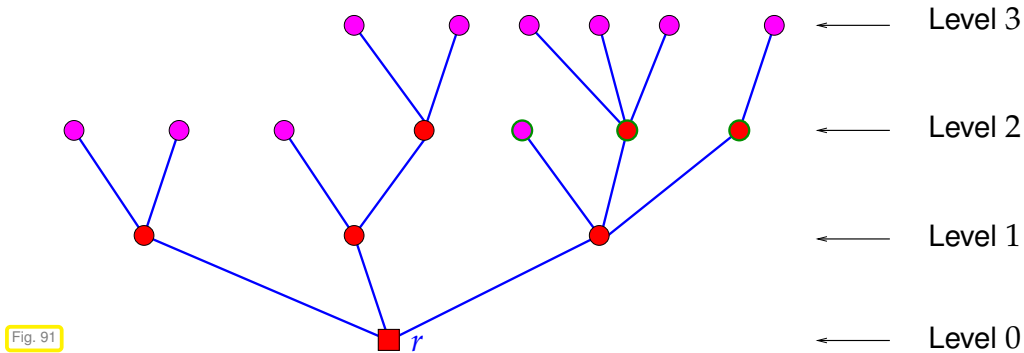


Fig. 91

The **depth** of a tree $\mathcal{T} = (\mathcal{V}, r, \mathcal{E})$ is the maximal level of its nodes

$$\text{depth}(\mathcal{T}) := \max\{\text{level}(v) : v \in \mathcal{V}\} .$$

Hence, Fig. 91 displays a tree of depth 3.

A tree is an intrinsically recursive data structure with each node carrying a sub-tree.

Definition 2.3.2.13. Sub-trees

Let $\mathcal{T} := (\mathcal{V}, r, \mathcal{E})$ be a tree and $w \in \mathcal{V}$. Set set

$$\mathcal{V}_w := \{v \in \mathcal{V} : \exists v_0, v_1, \dots, v_\ell \in \mathcal{V}, \ell \in \mathbb{N}_0 : v_0 = w, v_\ell = v, (v_{i-1}, v_i) \in \mathcal{E} \forall i \in \{1, \dots, \ell\}\}$$

is the set of **descendants** of w and $(\mathcal{V}_w, w, \mathcal{E} \cap (\mathcal{V}_w \times \mathcal{V}_w))$ is a tree, called **sub-tree** of \mathcal{T} with root w .

For $w \in \mathcal{V}$ the **ancestors** of w form the set

$$\{v \in \mathcal{V} : \exists v_0, v_1, \dots, v_\ell \in \mathcal{V}, \ell \in \mathbb{N}_0 : v_0 = v, v_\ell = w, (v_{i-1}, v_i) \in \mathcal{E} \forall i \in \{1, \dots, \ell\}\} .$$

Sub-tree \mathcal{T}_w attached to a node w of a tree.

: nodes/vertices of sub-tree \mathcal{T}_w

w is the root of the subtree.

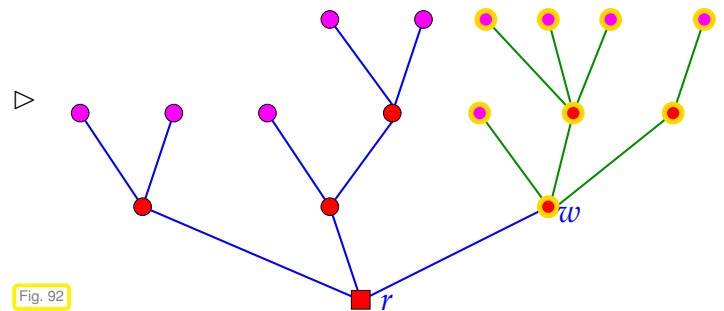


Fig. 92

┘

Now we introduce a key concept that defines a sequence of nested partitions of index sets by means of a tree.

Definition 2.3.2.14. Cluster tree

Let $\mathbb{I} \subset \mathbb{N}$ be an index set, $\mathcal{T} := (\mathcal{V}, r, \mathcal{E})$ a tree, and $\mathcal{I} : \mathcal{V} \rightarrow 2^{\mathbb{I}}$ a mapping that assigns a subset of \mathbb{I} to every node of \mathcal{T} . We call $\mathcal{T}_{\mathbb{I}} := (\mathcal{V}, r, \mathcal{E}, \mathbb{I}, \mathcal{I})$ a **cluster tree** for \mathbb{I} , if

- (i) the subset corresponding to the root is \mathbb{I} : $\mathcal{I}(r) = \mathbb{I}$,
- (ii) the subset associated with each non-leaf node is the union of the subsets of its sons

$$\mathcal{I}(w) = \bigcup \{ \mathcal{I}(v) : v \in \text{sons}(w) \} \quad \forall w \in \mathcal{V}, \text{sons}(w) \neq \emptyset, \tag{2.3.2.15}$$

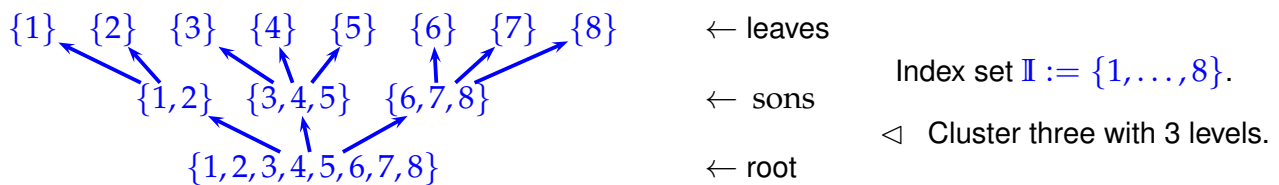
- (iii) the sets belonging to different sons of a node are disjoint

$$\forall w \in \mathcal{V}: v_1, v_2 \in \text{sons}(w) \Rightarrow \mathcal{I}(v_1) \cap \mathcal{I}(v_2) = \emptyset. \tag{2.3.2.16}$$

Terminology: The nodes of a cluster tree are also called **clusters**.

Notation: If $\mathcal{T}_{\mathbb{I}}$ is a cluster tree, we write $\mathcal{L}_{\mathbb{I}}$ for the set of its leaves

EXAMPLE 2.3.2.17 (General cluster tree) We visualize a simple cluster tree by overlaying a subset of the index set to every node of a tree:



Obviously this is not a binary cluster tree. ┘

§2.3.2.18 (Bounding boxes of clusters) In the context of approximating kernel collocation matrices we know the collocation points $\mathbf{x}^i \in \mathbb{R}^d, i \in \mathbb{I} := \{1, \dots, n\}$, and $\mathbf{y}^j \in \mathbb{R}^d, j \in \mathbb{J} := \{1, \dots, m\}$. Thus every subset $I \subset \mathbb{I}$ or $J \subset \mathbb{J}$ of indices also describes a “cloud”/set of points $\{\mathbf{x}^i\}_{i \in I}, \{\mathbf{y}^j\}_{j \in J}$. For instance, if $\mathcal{T}_{\mathbb{I}} := (\mathcal{V}, r, \mathcal{E}, \mathbb{I}, \mathcal{I})$ is a cluster tree associated with the \mathbf{x} -direction, then the

$$\text{node } v \in \mathcal{T}_{\mathbb{I}} \text{ holds the points } \{\mathbf{x}^i\}_{i \in \mathcal{I}(v)}.$$

The smallest axi-parallel box containing all the collocation points held by a cluster is called its **bounding box**, cf. Def. 2.3.1.12.

Definition 2.3.2.19. Bounding boxes of clusters

Let $\mathcal{T}_{\mathbb{L}} := (\mathcal{V}, r, \mathcal{E}, \mathbb{L}, \mathcal{I})$ be a cluster tree (\rightarrow Def. 2.3.2.14) for the index set $\mathbb{L} \subset \mathbb{N}$ and $\{z^k\}_{k \in \mathbb{L}} \subset \mathbb{R}^d$ a set of points. Then for every node $v \in \mathcal{T}_{\mathbb{L}}$ we define its **bounding box** as the bounding box of contained points:

$$\text{box}(v) := \prod_{\ell=1}^d [\min \{z_{\ell}^i\}_{i \in \mathcal{I}(v)}, \max \{z_{\ell}^i\}_{i \in \mathcal{I}(v)}] \subset \mathbb{R}^d.$$

§2.3.2.20 (Construction of cluster trees) Matching the recursive nature of the tree data structure, a natural way to construct a cluster tree is by recursion. We demonstrate this by means of a simple C++ code building a d -dimensional **binary** cluster tree.

C++ code 2.3.2.21: Data structures for a collocation points and bounding boxes →GITLAB

```

2  template <int DIM> // dimension d as template argument
3  struct Point {
4      constexpr static std::size_t dim = DIM;
5      std::size_t idx; // number of collocation point
6      Eigen::Matrix<double, DIM, 1> x; // coordinate vector
7  };

```

```

2  template <int DIM> // dimension d as template argument
3  struct BBox {
4      constexpr static std::size_t dim = DIM;
5      // Bounding box from sequence of points
6      explicit BBox(const std::vector<Point<DIM>> pts);
7      // Size  $\text{diam}(B)$  of a bounding box
8      [[nodiscard]] double diam() const {
9          return (maxc - minc).cwiseAbs().maxCoeff();
10     }
11     // Coordinate vectors of Corner points of bounding box
12     Eigen::Matrix<double, DIM, 1> minc; // Lower-left corner
13     Eigen::Matrix<double, DIM, 1> maxc; // Upper-right corner
14 };

```

The directed edges of the cluster tree are represented by *pointers* to other nodes. Moreover a node holds information about its associated collocation points.

C++ code 2.3.2.22: Data type for a node of the binary cluster tree →GITLAB

```

2  template <int DIM>
3  class CtNode {
4  public:
5      constexpr static std::size_t dim = DIM;
6      // Constructors taking a sequence of points
7      explicit CtNode(const std::vector<Point<DIM>> _pts, int _dir = 0)
8          : pts(std::move(_pts)),
9            sons{nullptr, nullptr},
10            dir(_dir),
11            clust_sect_vec(_pts.size()) {}
12     // Destructor (also attempts to destroy sons!)
13     virtual ~CtNode() {
14         if (sons[0]) {
15             delete sons[0];
16         }
17         if (sons[1]) {
18             delete sons[1];
19         }
20     }
21     // Number of indices owned by the cluster
22     [[nodiscard]] std::size_t noldx() const { return pts.size(); }
23     // Function I: access to owned indices
24     [[nodiscard]] std::vector<std::size_t> I() const;
25     // Access to bounding box (computed on the fly)
26     [[nodiscard]] BBox<DIM> getBBox() const { return BBox<DIM>(pts); }
27     // Is the node a leaf node ?
28     [[nodiscard]] virtual bool isLeaf() const {
29         return !(sons[0]) and !(sons[1]);
30     }
31     // Output operator or recursive output
32     template <int dim>

```



```

33 friend std::ostream &operator<<(std::ostream &o, const CtNode<dim> &node);
34 // Public data member: Pointers to two (binary tree!) sons
35 std::array<CtNode *, 2> sons;
36 // Public data member: Points contained in the cluster
37 std::vector<Point<DIM>> pts;
38 // Temporary storage for cluster-associated vector sections
39 Eigen::VectorXd clust_sect_vec;
40 // Direction for sorting, passed by the constructor
41 int dir;
42 };

```

A cluster tree object essentially holds a pointer to the root of the cluster tree.

C++ code 2.3.2.23: “Envelope” data structure for cluster tree →GITLAB

```

2 template <class NODE>
3 class ClusterTree {
4 public:
5     using node_t = NODE;
6     constexpr static std::size_t dim = NODE::dim; // space dimension d
7     // Idle constructor
8     ClusterTree() : root(nullptr) {}
9     // Effective Constructor taking a sequence of points
10    // (needed, because polymorphism not supported in constructor)
11    void init(const std::vector<Point<dim>> &pts, std::size_t minpts = 1);
12    // Recursive destruction
13    virtual ~ClusterTree() {
14        if (root) delete root;
15    }
16    // Output of tree
17    template <class Nd>
18    friend std::ostream &operator<<(std::ostream &o, const ClusterTree<Nd> &T);
19
20 protected:
21    // Recursive construction
22    virtual void buildRec(NODE *nptr, std::size_t minpts);
23    // Node factory
24    virtual NODE *createNode(const std::vector<Point<dim>> pts, int dir) {
25        return new NODE(pts, dir);
26    }
27
28 public:
29    NODE *root; // pointer to root node
30 };

```

The argument `minpts` to the constructor specifies the minimal number -1 of indices contained in a non-leaf cluster.

In order to cope with non-uniform distributions of collocation points, the recursive construction of the binary cluster tree does not merely split the bounding boxes of clusters in half. Instead, in order to obtain a *balanced cluster tree* the splitting of collocation point sets is done according to the rule:

Balanced nodal index sets

Rule: the cardinalities of the index sets of the sons of a node must not differ by more than 1.

The goal just stated is achieved by **alternating directional splitting**: Let us assume that the collocation

points $\{x^i\}_{i \in \mathcal{I}(w)}$ owned by a node $w \in \mathcal{T}_{\mathbb{I}}$ are sorted according to their ℓ -th component:

$$i, k \in \mathcal{I}(w), i < k \Rightarrow x_{\ell}^i \leq x_{\ell}^k, \text{ for some } \ell \in \{1, \dots, d\}. \quad (2.3.2.25)$$

Then we assign the following index subsets to the sons of w

$$\begin{aligned} \mathcal{I}(\text{1st son of } w) &:= \left\{ 1, \dots, \left\lfloor \frac{\#\mathcal{I}(w)}{2} \right\rfloor \right\}, \\ \mathcal{I}(\text{2nd son of } w) &:= \left\{ \left\lfloor \frac{\#\mathcal{I}(w)}{2} \right\rfloor + 1, \dots, \#\mathcal{I}(w) \right\}. \end{aligned} \quad (2.3.2.26)$$

The direction ℓ cycles through $\{1, \dots, d\}$ as we advanced towards the leaves of the cluster tree.

C++ code 2.3.2.27: Construction of a cluster tree from collocation points →GITLAB

```

2  template <class NODE>
3  void ClusterTree<NODE>::init(const std::vector<Point<dim>> &pts,
4                               std::size_t minpts) {
5      if (!(root = createNode(pts, 0))) {
6          throw(std::runtime_error("Cannot allocate root"));
7      }
8      if (minpts < 1) {
9          throw(std::runtime_error("minpts must be at least 1"));
10     }
11     buildRec(root, minpts);
12 }

```

```

2  template <class NODE>
3  void ClusterTree<NODE>::buildRec(NODE *nptr, std::size_t minpts) {
4      const std::size_t n = nptr->pts.size(); // Number of held indices
5      // Leaf, if minimal number of indices reached
6      if (n > minpts) { //
7          // Points have to be copied and sorted according to direction dir
8          std::vector<Point<dim>> tpts(nptr->pts);
9          // next sorting direction
10         const int dir = (nptr->dir + 1) % dim;
11         // call sort function from standard library
12         std::sort(tpts.begin(), tpts.end(),
13                 [dir](const Point<dim> &p1, const Point<dim> &p2) -> bool {
14             return (bool)(p1.x[dir] < p2.x[dir]);
15         });
16         // Split point sequence and construct sons
17         const std::size_t m = n / 2; // integer arithmetic, m>0 ensured
18         const std::vector<Point<dim>> low_pts(tpts.cbegin(), tpts.cbegin() + m);
19         // First son gets "lower half" of sorted points
20         if (!(nptr->sons[0] = createNode(low_pts, dir))) {
21             throw(std::runtime_error("Cannot allocate first son"));
22         }
23         buildRec(nptr->sons[0], minpts); // recurse into first son
24         // Second son get "upper half" of sorted points
25         const std::vector<Point<dim>> up_pts(tpts.cbegin() + m, tpts.cend());
26         if (!(nptr->sons[1] = createNode(up_pts, dir))) {
27             throw(std::runtime_error("Cannot allocate second son"));
28         }
29         buildRec(nptr->sons[1], minpts); // recurse into 2nd son
30     }
31 }

```

Taking into account sorting the total computational effort for BuildRec in the case of $n := \#\mathbb{I} = 2^L$ is

$$\sum_{k=0}^L (k+1)(L-k)2^{L-k} = O(n \log^2 n).$$

┘

EXAMPLE 2.3.2.28 (Binary cluster tree for $d = 1$)

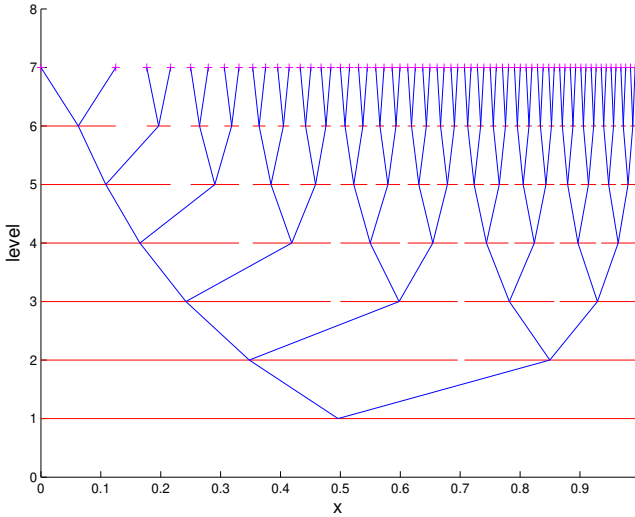


Fig. 93

For $d = 1$ with collocation points

$$\zeta_i := \sqrt{\frac{i-1}{n}}, \quad i = 1, \dots, n, \quad n = 64.$$

These points are *not* evenly distributed in $[0, 1]$.

- ◁ Balanced binary cluster tree (center of bounding box drawn for each cluster)

EXAMPLE 2.3.2.29 (Unbalanced cluster tree) The construction of a binary cluster tree could also be based on a purely **geometry-based distribution** of the indices to the sons. For instance, for $d = 1, w \in \mathcal{T}_{\mathbb{I}}$ a cluster owning the collocation points $\{\zeta_i\}_{i \in \mathcal{I}(w)}$, we could set

$$\begin{aligned} \mathcal{I}(\text{1st son of } w) &= \{i \in \mathcal{I}(w) : \zeta_i \leq \gamma_w\}, \\ \mathcal{I}(\text{2nd son of } w) &= \{i \in \mathcal{I}(w) : \zeta_i > \gamma_w\}, \end{aligned}$$

where $\gamma_w := \frac{1}{2}(\max\{\zeta_i\}_{i \in \mathcal{I}(w)} + \min\{\zeta_i\}_{i \in \mathcal{I}(w)})$ is the “midpoint” of the cluster w .

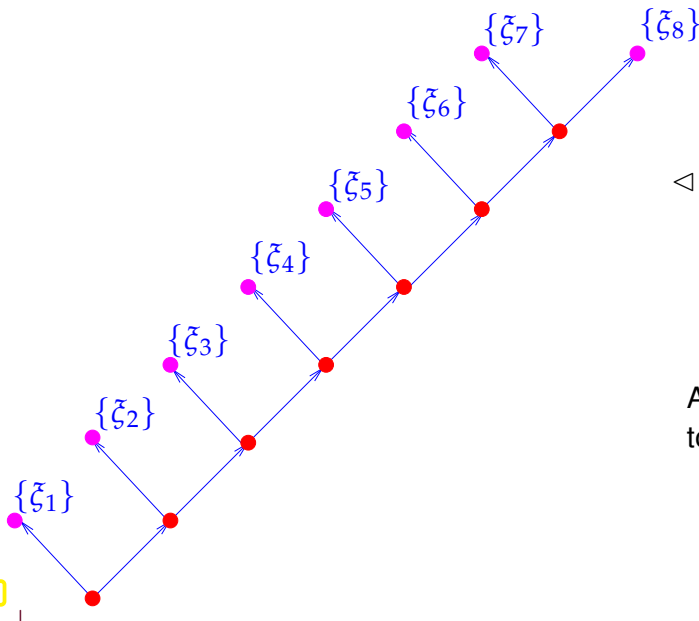


Fig. 94

- ◁ Geometry-based cluster tree for $n = 8$ and the non-uniformly distributed collocation points

$$\zeta_i = 2^{-i+1}, \quad i = 1, \dots, n$$

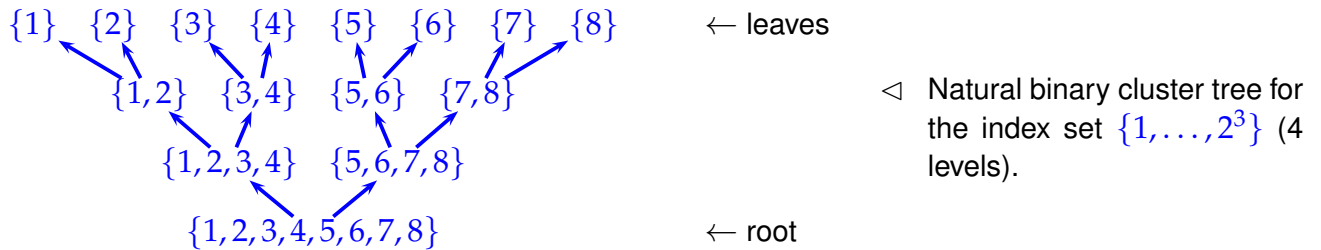
At each level exactly one point will be split off, leading to a highly imbalanced cluster tree.

2.3.3 Building Near- and Far-Field Blocks

Cluster trees of $\{1, \dots, n\}$ or $\{1, \dots, m\}$ provide a hierarchy of partitions of these index sets. However, what we are aiming for is a partition of the product set $\mathbb{D} := \{1, \dots, n\} \times \{1, \dots, m\}$. We construct it

based on cluster trees.

EXAMPLE 2.3.3.1 (Quadtree partition from cluster trees) We reconsider the quadtree-based tiling of $[0, 1]^2$ from Ex. 2.3.2.1 and the induced partition of a kernel collocation matrix based on equidistant collocation points in 1D.



The nodes of the balanced binary cluster tree (\rightarrow Def. 2.3.2.14) for the index set $\{1, \dots, 2^L\}$ on level $\ell = 1, \dots, L$ are associated with the index sets:

$$\mathbb{I}_{\ell,k} := \{(k-1) \cdot 2^{L-\ell} + 1, \dots, k \cdot 2^{L-\ell}\}, \quad \ell = 1, \dots, L.$$

The square matrix blocks arising from quadtree tiling are submatrices belonging to products of such index sets (from the same level): $(\widetilde{\mathbf{M}})_{\mathbb{I}_{\ell,k}, \mathbb{I}_{\ell,m}} \in \mathbb{R}^{2^{L-\ell}, 2^{L-\ell}}$. This is displayed in the following figure for $L = 5$:

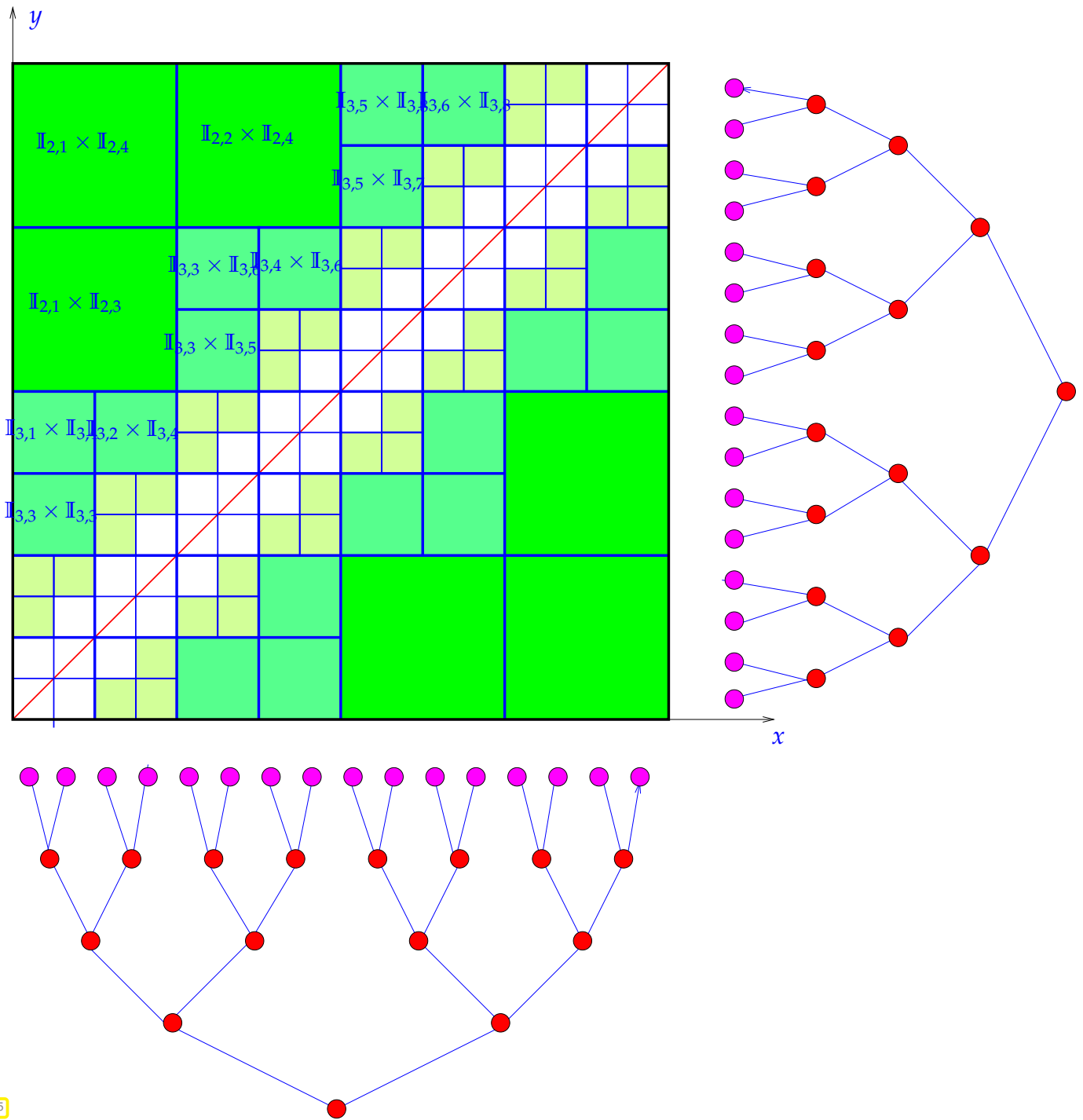


Fig. 95

Now we take the cue from the algorithm in Ex. 2.3.2.1 to devise a recursive algorithm that builds a near-field/far-field matrix partition from binary cluster trees of the index sets $\{1, \dots, n\}$ and $\{1, \dots, m\}$. It will be based on an abstract admissibility condition:

Definition 2.3.3.2. Abstract admissibility condition

Let $\mathcal{T}_{\mathbb{I}}$ and $\mathcal{T}_{\mathbb{J}}$ be cluster trees (\rightarrow Def. 2.3.2.14) for index sets $\mathbb{I} := \{1, \dots, n\}$ and $\mathbb{J} := \{1, \dots, m\}$,
A mapping

$$\text{adm} : \mathcal{T}_{\mathbb{I}} \times \mathcal{T}_{\mathbb{J}} \rightarrow \{\text{true}, \text{false}\}$$

is called an **admissibility condition** for $\mathcal{T}_{\mathbb{I}}$ and $\mathcal{T}_{\mathbb{J}}$, if “admissibility is inherited by the sons”:

$$\begin{aligned} \text{adm}(\tau, \sigma) &\Rightarrow \text{adm}(\tau', \sigma) \quad \forall \tau \in \mathcal{T}_{\mathbb{I}}, \sigma \in \mathcal{T}_{\mathbb{J}}, \tau' \in \text{sons}(\tau), \\ \text{adm}(\tau, \sigma) &\Rightarrow \text{adm}(\tau, \sigma') \quad \forall \tau \in \mathcal{T}_{\mathbb{I}}, \sigma \in \mathcal{T}_{\mathbb{J}}, \sigma' \in \text{sons}(\sigma). \end{aligned}$$

The next code snippet contains the definition a class that can construct and represent the partition of a matrix into two kinds of blocks: near-field blocks and far-field blocks. **Node** must be a type compliant with **CtNode** from Code 2.3.2.22. The template argument type **FFB** and **NFB** stand for data structures describing far-field and near-field blocks of a kernel collocation matrix.

C++ code 2.3.3.3: Class describing a far-field/near-field matrix partition [→GITLAB](#)

```

2  template <class NODE, typename FFB, typename NFB>
3  class BlockPartition {
4  public:
5      using node_t = NODE;
6      using farfieldblock_t = FFB;
7      using nearfieldblock_t = NFB;
8      // Idle constructor
9      BlockPartition (std::shared_ptr<ClusterTree<NODE>> _rowT,
10                    std::shared_ptr<ClusterTree<NODE>> _colT)
11          : rowT(_rowT), colT(_colT) {
12          assertm((rowT != nullptr), "No valid x-tree!");
13          assertm((colT != nullptr), "No valid y-tree!");
14      }
15      // Trigger recursive construction of partition
16      // (Needed, because polymorphic functions not available in
17      // constructor)
18      void init(double eta0 = 0.5);
19      virtual ~BlockPartition() = default;
20      // Size of the matrix
21      [[nodiscard]] size_t cols() const { return ((colT->root)->pts).size(); }
22      [[nodiscard]] size_t rows() const { return ((rowT->root)->pts).size(); }
23      // Admissibility condition adm, see Def. 2.3.3.2
24      [[nodiscard]] virtual bool adm(const NODE *nx, const NODE *ny,
25                                   double eta0) const;
26
27  protected:
28      // Recursive construction from cluster pair
29      virtual void buildRec(NODE *nx, NODE *ny, double eta0);
30      // Construct an instance of far-field block type
31      virtual FFB makeFarFieldBlock(NODE &nx, NODE &ny) { return FFB(nx, ny); }
32      // Construct an instance of near-field block type
33      virtual NFB makeNearFieldBlock(NODE &nx, NODE &ny) { return NFB(nx, ny); }
34
35  public:
36      std::shared_ptr<ClusterTree<NODE>> rowT; // row cluster tree
37      std::shared_ptr<ClusterTree<NODE>> colT; // column cluster tree
38      std::vector<FFB> farField; // index blocks in the far field
39      std::vector<NFB> nearField; // index blocks in the near field
40      static bool dbg; // Debugging flag

```

```
40 };
```

The vectors `farField` and `nearField` contain objects that store two index sets each. A suitable data type for both **FFB** and **FFB** may be defined as follows.

C++ code 2.3.3.4: Data structure for matrix block →GITLAB

```
2  template <class NODE>
3  struct IndexBlock {
4      using node_t = NODE;
5      constexpr static std::size_t dim = NODE::dim;
6      // Constructors extracts indices from clusters
7      IndexBlock(NODE &_nx, NODE &_ny)
8          : nx(_nx), ny(_ny), i_idx(_nx.l()), j_idx(ny.l()) {}
9      virtual ~IndexBlock() = default;
10     NODE &nx, &ny; // contributing clusters
11     const std::vector<size_t> i_idx, j_idx; // contained indices
12 };
```

Other suitable data type are the two classes **FarFieldBlock** and **NearFieldBlock** from Code 2.3.1.23.

The partitioning of $\mathbb{D} := \mathbb{I} \times \mathbb{J}$ is built *recursively* by climbing up both cluster trees in tandem identifying admissible pairs of clusters on the way:

- ◆ If one of the clusters is a *leaf*, then put the pair in the *near field*,
- ◆ else if the pair of clusters is *admissible*, then assign it to the *far field*
- ◆ else continue *recursion* with all pairs of sons.

This is implemented in the following functions:

C++ code 2.3.3.5: Recursive construction of matrix partition →GITLAB

```
2  template <class NODE, typename FFB, typename NFB>
3  void BlockPartition<NODE, FFB, NFB>::init(double eta0) {
4      buildRec(rowT->root, colT->root, eta0);
5  }

2  template <class NODE, typename FFB, typename NFB>
3  void BlockPartition<NODE, FFB, NFB>::buildRec(NODE *nx, NODE *ny, double eta0) {
4      if (nx && ny) {
5          // Add admissible pair to far field
6          if (adm(nx, ny, eta0)) { //
7              farField.push_back(makeFarFieldBlock(*nx, *ny));
8          } else {
9              bool rec = false;
10             for (int isx = 0; isx <= 1; isx++) {
11                 for (int isy = 0; isy <= 1; isy++) {
12                     if (nx->sons[isx] && ny->sons[isy]) {
13                         // Next level of recursion for non-leaves
14                         rec = true;
15                         buildRec(nx->sons[isx], ny->sons[isy], eta0);
16                     }
17                 }
18             }
19             // At least one leaf cluster:
20             // Add cluster pair to near field
21             if (!rec) //
```

```

22     nearField.push_back(makeNearFieldBlock(*nx, *ny));
23   }
24   } else
25     throw(std::runtime_error("Invalid node pointers"));
26 }

```

In compliance with Def. 2.3.1.13 and Def. 2.3.1.15, the implementation of the `adm()` method will rely on a **geometric admissibility condition** invoking the admissibility measure $\eta(B)$ from Eq. (2.2.2.7), where B is the product of the bounding boxes of the two clusters. More precisely, a pair of clusters $(v, w) \in \mathcal{T}_I \times \mathcal{T}_J$ qualifies as η_0 -admissible, $\eta_0 > 0$, if

$$\text{adm}(v, w) = \text{true} \Leftrightarrow \begin{cases} \eta(\text{box}(v), \text{box}(w)) \leq \eta_0, \\ \text{and} \\ v \text{ and } w \text{ is not a leaf} \end{cases} \quad (2.3.3.6)$$

The following implementation of the `adm()`-method realizes (2.3.3.6). The implementations of `dist()` and `diam()` for bounding boxes are given in Code 2.3.2.21.

C++ code 2.3.3.7: Geometric admissibility condition `adm` →GITLAB

```

2  template <class NODE, typename FFB, typename NFB>
3  bool BlockPartition<NODE, FFB, NFB>::adm(const NODE *nx, const NODE *ny,
4      double eta0) const {
5      // In an admissible pair neither node must be a leaf.
6      if (nx->isLeaf() || ny->isLeaf()) return false;
7      // Geometric admissibility condition, see Eq. (2.2.2.7).
8      const BBox<NODE::dim> Bx = nx->getBBox(), By = ny->getBBox();
9      const double bb_dist = dist(Bx, By);
10     if (bb_dist == 0.0) return false;
11     const double eta = std::max(Bx.diam(), By.diam()) / (2 * bb_dist);
12     return (eta < eta0);
13 }

```

The following pictures illustrate what is happening during first few calls of `buildRec`; to be continued by the reader by supplementing Fig. 101.

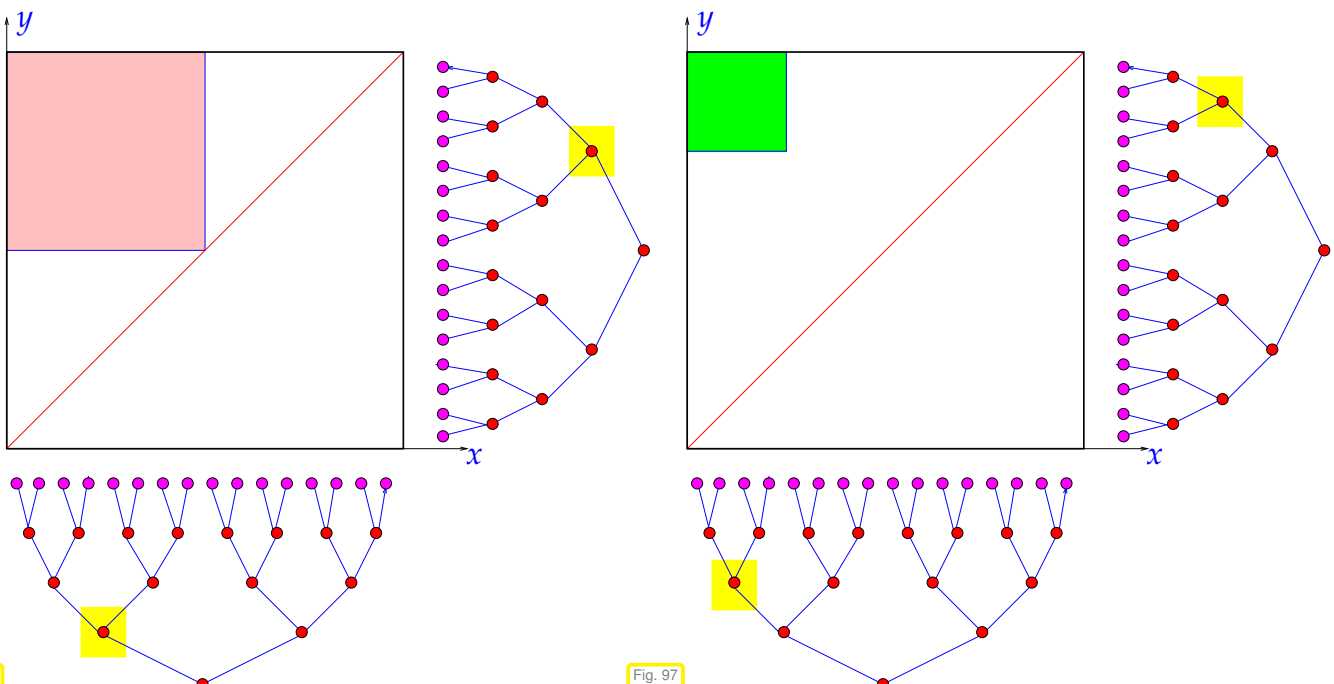


Fig. 96

Fig. 97

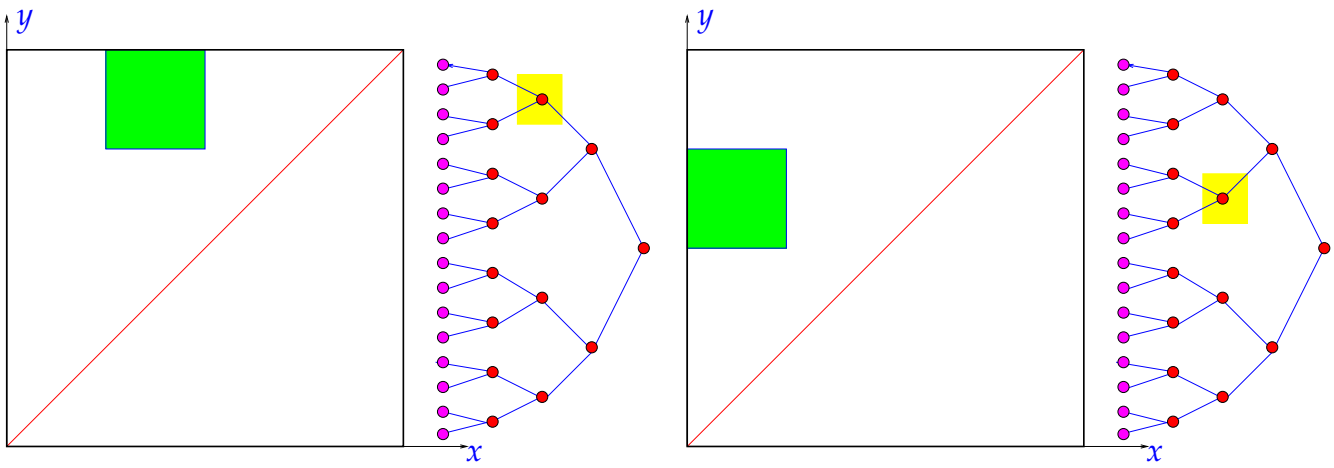


Fig. 98

Fig. 99

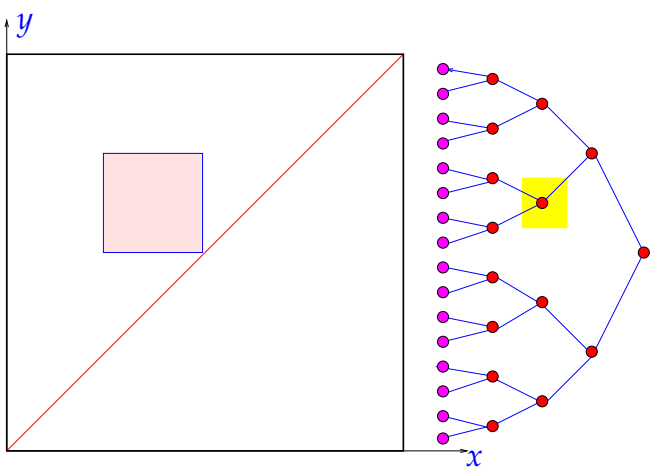
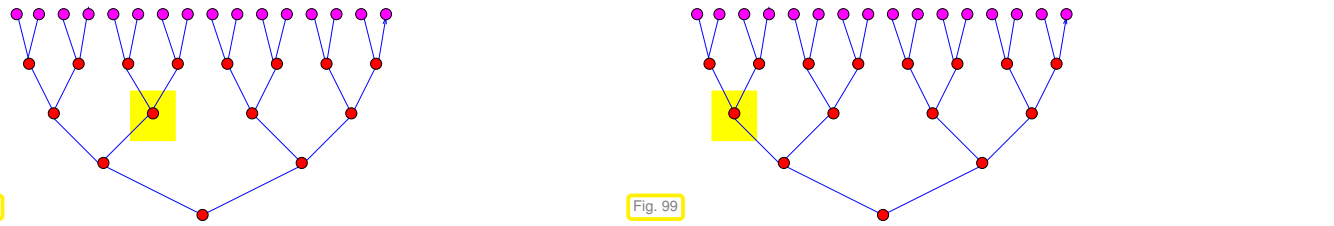


Fig. 100

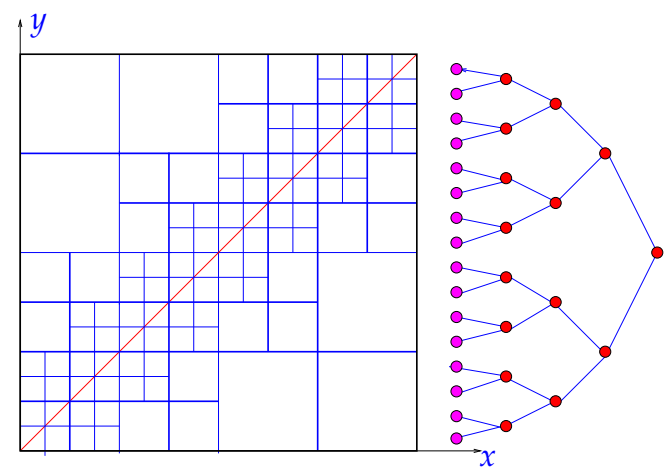


Fig. 101

EXAMPLE 2.3.3.8 (Near- and far-field boxes constructed from cluster trees in 1D) We apply the algorithm of Code 2.3.3.5 with $\eta_0 = \frac{1}{2}$ for equispaced and non-equispaced sets of points and visualize the resulting block partition: $*$ $\hat{=}$ near field point pair, \square $\hat{=}$ product of bounding boxes for far-field cluster pairs.

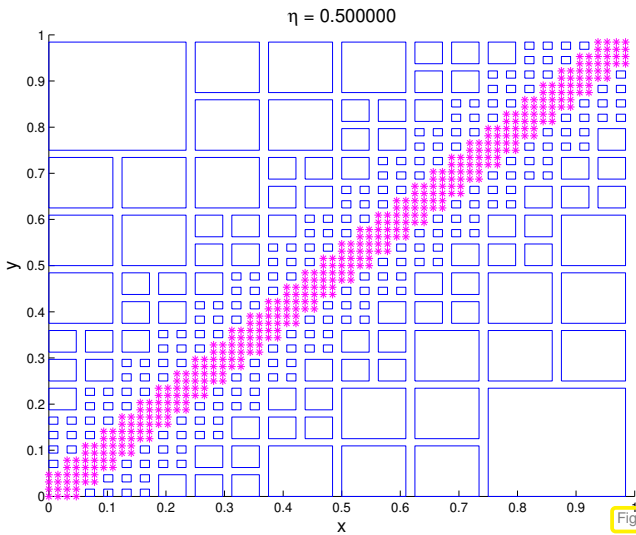


Fig. 102

$$\zeta_i = \eta_i = \frac{i}{64}, i = 0, \dots, 64$$

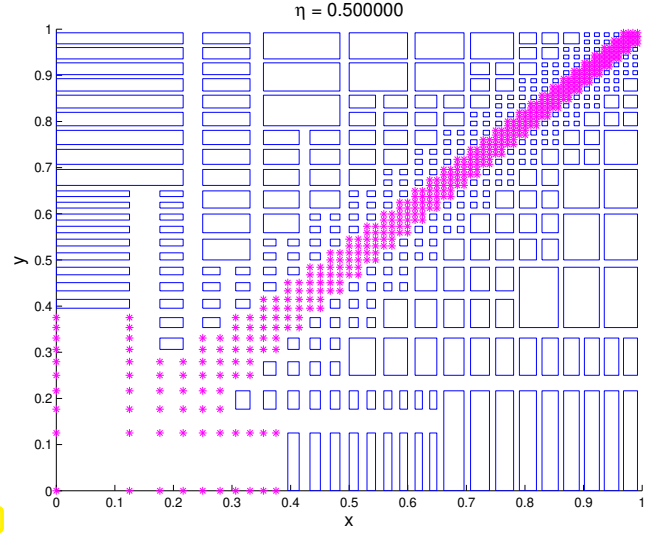


Fig. 103

$$\zeta_i = \eta_i = \sqrt{\frac{i}{64}}, i = 0, \dots, 64$$

2.3.4 Storing Block-Partitioned Kernel Collocation Matrix

We aim for economically storing the local low-rank approximation $\widetilde{\mathbf{M}} \in \mathbb{R}^{n,m}$ of a kernel collocation matrix (\rightarrow Def. 2.1.3.1) $\mathbf{M} = [G(x^i, y^j)]_{\substack{i \in \mathbb{I}, \\ j \in \mathbb{J}}}$, $\mathbb{I} := \{1, \dots, n\}$, $\mathbb{J} := \{1, \dots, m\}$, $x^i, y^j \in [0, 1]^d$. We assume that we are given

- ◆ two binary cluster trees (\rightarrow Def. 2.3.2.14) $\mathcal{T}_{\mathbb{I}}$ (“x-tree”/row tree) and $\mathcal{T}_{\mathbb{J}}$ (“y-tree”/column tree) for the index sets \mathbb{I} and \mathbb{J} . Both are available as instances of **ClusterTree**, see Code 2.3.2.23.
- ✍ Notation: We use the symbol v for clusters $\in \mathcal{T}_{\mathbb{I}}$, and w for clusters $\in \mathcal{T}_{\mathbb{J}}$
- ◆ a far-field/near-field block partition of $\mathbb{D} := \mathbb{I} \times \mathbb{J}$ built from the cluster trees $\mathcal{T}_{\mathbb{I}}$ and $\mathcal{T}_{\mathbb{J}}$ by “admissible” recursive subdivision as done by `buildRec` in Code 2.3.3.5. The block partition is represented as an instance of type **BlockPartition** as defined in Code 2.3.3.3.

As in § 4.1.1.9 we write \mathbb{F}_{far} and \mathbb{F}_{near} for the near field and far field, which are sets of pairs of clusters, corresponding to sets of sets of pairs of indices (!).

Recall that each cluster carries an index set accessible through the function \mathcal{I} , e.g. $\mathcal{I} : \mathcal{T}_{\mathbb{I}} \rightarrow 2^{\mathbb{I}}$, see Def. 2.3.2.14. We adopt the following shorthand notation for blocks of the matrix \mathbf{M} associated with pairs of clusters:

$$\begin{matrix} v \in \mathcal{T}_{\mathbb{I}}, & \mathcal{I}(v) = \{i_1, \dots, i_k\}, \\ w \in \mathcal{T}_{\mathbb{J}}, & \mathcal{I}(w) = \{j_1, \dots, j_\ell\}, \end{matrix} : \mathbf{M}|_{v \times w} := \left[(\mathbf{M})_{i,j} \right]_{\substack{i=i_1, \dots, i_k \\ j=j_1, \dots, j_\ell}} \in \mathbb{R}^{k, \ell}, \quad (2.3.4.1)$$

We remind of the gist of local low-rank approximation of kernel matrices:

If $(v, w) \in \mathbb{F}_{\text{far}}$ the sub-matrix $\mathbf{M}|_{v \times w}$ is approximated by a rank- q matrix arising from a rank- q separable approximation (2.2.1.8) \widetilde{G} of $G|_{\text{box}(v) \times \text{box}(w)}$ with $q \ll \min\{\#\mathcal{I}(v), \#\mathcal{I}(w)\}$.

The bounding box of a cluster is defined in Def. 2.3.2.19.

§2.3.4.2 (\widetilde{G} from uni-directional interpolation \rightarrow Section 2.2.1.2) As explained in § 2.2.1.30, in the

case of uni-directional interpolation we rely on the rank- q separable approximation

$$G(\mathbf{x}, \mathbf{y}) \approx \tilde{G}(\mathbf{x}, \mathbf{y}) := \sum_{\ell=1}^q \underbrace{b_{\ell}^v(\mathbf{x})}_{=:g_{\ell}(\mathbf{x})} \underbrace{G(\mathbf{t}_v^{\ell}, \mathbf{y})}_{=:h_{\ell}(\mathbf{y})}, \quad (\mathbf{x}, \mathbf{y}) \in \text{box}(v) \times \text{box}(w), \quad (2.3.4.3)$$

for any far-field cluster pair $(v, w) \in \mathcal{T}_{\mathbb{I}} \times \mathcal{T}_{\mathbb{J}}$. Here $b_{\ell}^v : \text{box}(v) \rightarrow \mathbb{R}$, $\ell = 1, \dots, q$, are the cardinal functions associated with the interpolation operator in \mathbf{x} -direction, see (2.2.1.29), and \mathbf{t}_v^{ℓ} denote the interpolation nodes.

This leads to the rank- q approximation in factorized form

$$\mathbf{M}|_{v \times w} \approx \tilde{\mathbf{M}}|_{v \times w} := \mathbf{U} \cdot \mathbf{V}^{\top}, \quad \begin{aligned} \mathbf{U} &:= \left[b_{\ell}^v(\mathbf{x}^i) \right]_{\substack{i \in \mathcal{I}(v) \\ \ell=1, \dots, q}} \in \mathbb{R}^{\#\mathcal{I}(v), q}, \\ \mathbf{V} &:= \left[G(\mathbf{t}_v^{\ell}, \mathbf{y}^j) \right]_{\substack{j \in \mathcal{I}(w) \\ \ell=1, \dots, q}} \in \mathbb{R}^{\#\mathcal{I}(w), q}. \end{aligned} \quad (2.3.4.4)$$

To indicate the dependence of the interpolation nodes and of the cardinal functions on the cluster v in (2.3.4.4) we wrote \mathbf{t}_v^{ℓ} and b_{ℓ}^v .

Dependence of local low-rank factors in the case of uni-directional interpolation

For $(v, w) \in \mathbb{F}_{\text{far}}$

- ◆ the low-rank factor \mathbf{U} according to (2.3.4.4) depends on v only,
- ◆ whereas \mathbf{V} depends on both clusters and the kernel function G .



Store all information about the low-rank factor \mathbf{U} in “its” cluster v ($\hat{=}$ node of cluster tree). This enables the reuse for several far-field cluster pairs sharing the same \mathbf{x} -cluster.

► Required total storage for \mathbf{U} -factors = $O(q n \log n)$ for $n \rightarrow \infty$.

§2.3.4.6 (\tilde{G} from bi-directional interpolation \rightarrow Section 2.2.1.3) From (2.2.1.46) we learned how to build a rank- q separable kernel resulting from interpolation in both \mathbf{x} - and \mathbf{y} -direction, here given for the same number q of interpolation points in both directions and applied on the cluster pair $(v, w) \in \mathcal{T}_{\mathbb{I}} \times \mathcal{T}_{\mathbb{J}}$:

$$G(\mathbf{x}, \mathbf{y}) \approx \tilde{G}(\mathbf{x}, \mathbf{y}) := \sum_{k=1}^q \sum_{\ell=1}^q \underbrace{G(\mathbf{t}_v^k, \mathbf{t}_w^{\ell})}_{=:g_{k,\ell}(\mathbf{x})} \underbrace{b_k^v(\mathbf{x})}_{=:h_{k,\ell}(\mathbf{y})} b_{\ell}^w(\mathbf{y}), \quad (\mathbf{x}, \mathbf{y}) \in \text{box}(v) \times \text{box}(w). \quad (2.3.4.7)$$

As above, we write $\{\mathbf{t}_v^k\}_{k=1, \dots, q}$ and $\{\mathbf{t}_w^{\ell}\}_{\ell=1, \dots, q}$ for the sets of interpolation nodes on the tensor-product domains $\text{box}(v)$ and $\text{box}(w)$, respectively. Again, $b_k^v : \text{box}(v) \rightarrow \mathbb{R}$ and $b_{\ell}^w : \text{box}(w) \rightarrow \mathbb{R}$ designate the associated cardinal functions for the underlying interpolation operators.

As in (2.2.1.47) we obtain a rank- q approximation of the block of the kernel collocation matrix in triple-factor form:

$$\mathbf{M}|_{v \times w} \approx \tilde{\mathbf{M}}|_{v \times w} := \mathbf{U}_v \mathbf{C}_{v \times w} \mathbf{V}_w^{\top}, \quad \begin{aligned} \mathbf{U}_v &:= \left[b_k^v(\mathbf{x}^i) \right]_{\substack{i \in \mathcal{I}(v) \\ k=1, \dots, q}} \in \mathbb{R}^{\#\mathcal{I}(v), q}, \\ \mathbf{C}_{v \times w} &:= \left[G(\mathbf{t}_v^k, \mathbf{t}_w^{\ell}) \right]_{k, \ell=1, \dots, q} \in \mathbb{R}^{q, q}, \\ \mathbf{V}_w &:= \left[b_{\ell}^w(\mathbf{y}^j) \right]_{\substack{j \in \mathcal{I}(w) \\ \ell=1, \dots, q}} \in \mathbb{R}^{\#\mathcal{I}(w), q}. \end{aligned} \quad (2.3.4.8)$$

Dependence of local matrix factors or bi-directional interpolation

For $(v, w) \in \mathbb{F}_{\text{far}}$

- ◆ the matrix factor \mathbf{U}_v solely depends on the cluster v ,
- ◆ the matrix factor \mathbf{V}_w solely on the cluster w ,
- ◆ while both clusters and the kernel function \mathbf{G} contribute to $\mathbf{C}_{v \times w}$.



The matrices \mathbf{U}_v and \mathbf{V}_w can be computed and stored in the clusters before even without knowing the kernel function.



Required total storage for \mathbf{U} -/ \mathbf{V} -factors = $O(qn \log n)/O(qm \log m)$ for $n, m \rightarrow \infty$.

┘

§2.3.4.10 (Bi-directional interpolation: Data structures for cluster pairs) We take the cue from (2.3.4.8) and the observation that the matrix factors \mathbf{U}_v and \mathbf{V}_w actually “belong to” a single cluster. This suggests that we extend the data structure for clusters through a derived class type.

C++ code 2.3.4.11: Extended cluster data structure for interpolatory kernel approximation →GITLAB

```

2  template <int DIM>
3  class CtNode {
4  public:
5      constexpr static std::size_t dim = DIM;
6      // Constructors taking a sequence of points
7      explicit CtNode(const std::vector<Point<DIM>> _pts, int _dir = 0)
8          : pts(std::move(_pts)),
9            sons{ nullptr, nullptr },
10           dir(_dir),
11           clust_sect_vec(_pts.size()) {}
12     // Destructor (also attempts to destroy sons!)
13     virtual ~CtNode() {
14         if (sons[0]) {
15             delete sons[0];
16         }
17         if (sons[1]) {
18             delete sons[1];
19         }
20     }
21     // Number of indices owned by the cluster
22     [[nodiscard]] std::size_t noldx() const { return pts.size(); }
23     // Function  $\mathcal{I}$ : access to owned indices
24     [[nodiscard]] std::vector<std::size_t> I() const;
25     // Access to bounding box (computed on the fly)
26     [[nodiscard]] BBox<DIM> getBBox() const { return BBox<DIM>(pts); }
27     // Is the node a leaf node ?
28     [[nodiscard]] virtual bool isLeaf() const {
29         return !(sons[0]) and !(sons[1]);
30     }
31     // Output operator or recursive output
32     template <int dim>
33     friend std::ostream &operator<<(std::ostream &o, const CtNode<dim> &node);
34     // Public data member: Pointers to two (binary tree!) sons
35     std::array<CtNode *, 2> sons;
36     // Public data member: Points contained in the cluster
37     std::vector<Point<DIM>> pts;

```

```

38 // Temporary storage for cluster-associated vector sections
39 Eigen::VectorXd clust_sect_vec;
40 // Direction for sorting, passed by the constructor
41 int dir;
42 };

```

To accommodate the extended argument list of the constructor, also the data structure for **ClusterTree** needs to be extended:

C++ code 2.3.4.12: Extended cluster tree data type built for **InterpNode from Code 2.3.4.11**
[→GITLAB](#)

```

2 template <class NODE>
3 class LLRClusterTree : public ClusterTree<NODE> {
4 public:
5 // Idle constructor just setting rank argument q
6 explicit LLRClusterTree(size_t _q) : q(_q) {}
7 // Actual constructor taking a sequence of points
8 void init(const std::vector<Point<NODE::dim>> pts, std::size_t minpts = 1);
9 virtual ~LLRClusterTree() = default;
10
11 protected:
12 // factory method for relevant type of node taking rank argument
13 virtual NODE *createNode(const std::vector<Point<NODE::dim>> pts, int dir) {
14     return new NODE(pts, q, dir);
15 }
16
17 public:
18     const std::size_t q; // rank of separable approximation on cluster boxes
19 };
20
21 template <class NODE>
22 void LLRClusterTree<NODE>::init(const std::vector<Point<NODE::dim>> pts,
23                               std::size_t minpts) {
24     ClusterTree<NODE>::init(pts, minpts);
25 }

```

In (2.3.4.8) the matrix factor $\mathbf{C} \in \mathbb{R}^{q,q}$ “belongs to” the cluster pair (v, w) . Therefore this matrix should be stored in the object representing the far-field cluster pair.

C++ code 2.3.4.13: Data type for a far-field cluster pair & bidirectional interpolation [→GITLAB](#)

```

2 template <class NODE, typename KERNEL>
3 class BiDirChebInterpBlock : public IndexBlock<NODE> {
4 public:
5     using kernel_t = KERNEL;
6     BiDirChebInterpBlock(const NODE &_nx, const NODE &_ny, KERNEL _Gfun,
7                         std::size_t _q);
8     // Constructor that should not be called, needed to avoid compilation
9     // errors
10    BiDirChebInterpBlock(const NODE &_nx, const NODE &_ny)
11        : IndexBlock<NODE>(_nx, _ny), q(0) {
12        throw std::runtime_error("Invalid constructor");
13    }
14    virtual ~BiDirChebInterpBlock() = default;
15
16    KERNEL G; // kernel function G
17    const int q; // No of interpolation nodes

```

```

17 | Eigen :: MatrixXd C; // C ∈ ℝq,q
18 | };

```

Remark 2.3.4.14 (Bi-directional polynomial interpolation) The low-rank triple-factor approximation of a kernel collocation matrix as introduced in Section 2.2.1.3 involves the two matrix factors

$$\mathbf{U} := [b_\ell^x(\mathbf{x}^i)]_{\substack{i=1,\dots,n \\ \ell=1,\dots,q}} \in \mathbb{R}^{n,q}, \quad \mathbf{V} := [b_\ell^y(\mathbf{y}^j)]_{\substack{j=1,\dots,m \\ \ell=1,\dots,q}} \in \mathbb{R}^{m,q}, \quad (2.3.4.15)$$

see (2.2.1.47). Here, $\mathbf{x}^i \in \mathbb{R}^d$, $i = 1, \dots, n$, and $\mathbf{y}^j \in \mathbb{R}^d$, $j = 1, \dots, m$, are collocation points (\rightarrow Def. 2.1.3.1), and the function b_ℓ^x, b_ℓ^y , $\ell = 1, \dots, q$, are **cardinal basis functions** (\rightarrow (2.2.1.29)) for the underlying interpolation operator and for **interpolation nodes** $t_x^\ell, t_y^\ell \in \mathbb{R}^d$, $\ell = 1, \dots, q$.

We employ tensor-product polynomial interpolation of degree $q - 1$ in each direction, cf. § 2.2.1.33 and § 2.2.1.37. From [NumCSE § 5.2.3.2] we recall the **barycentric interpolation formula** in 1D: Given the set $\{t^1, \dots, t^q\} \subset \mathbb{R}$ of interpolation nodes on the real line the unique polynomial $p \in \mathcal{P}_q$ satisfying the interpolation conditions $p(t^i) = y_i$ for given $y_1, \dots, y_q \in \mathbb{R}$, can be written as

$$p(x) = \sum_{i=1}^q \frac{\lambda_i}{x - t^i} y_i \cdot \left(\sum_{i=1}^q \frac{\lambda_i}{x - t^i} \right)^{-1}, \quad (2.3.4.16)$$

$$\text{with weights} \quad \lambda_i = \frac{1}{(t^i - t^1) \dots (t^i - t^{i-1})(t^i - t^{i+1}) \dots (t^i - t^q)}, \quad i = 1, \dots, q. \quad (2.3.4.17)$$

The cardinal basis functions for polynomial interpolation are the Lagrange polynomials

$$L_\ell(x) := \prod_{\substack{j=1 \\ j \neq \ell}}^q \frac{x - t^j}{t^\ell - t^j}, \quad x \in \mathbb{R}, \quad \ell = 1, \dots, q \Rightarrow L_\ell(t^k) = \delta_{\ell,k}, \quad \ell, k = 1, \dots, q. \quad (2.2.1.34)$$

For them we get the barycentric formula

$$L_\ell(x) = \begin{cases} \frac{\lambda_\ell}{x - t^\ell} \cdot \left(\sum_{i=1}^q \frac{\lambda_i}{x - t^i} \right)^{-1}, & x \neq t^k, \quad k = 1, \dots, q, \\ 1, & x = t^\ell, \\ 0, & x = t^k, \quad k \neq \ell, \end{cases} \quad (2.3.4.18)$$

to be supplemented with $L_\ell(t^\ell) = 1$.

Now we discuss the case $d = 2$ and the computation of $\mathbf{U} = [b_\ell^x(\mathbf{x}^j)]$. We assume a tensor-product grid of interpolation points with lexicographic ordering

$$t_x^j = \begin{bmatrix} t_1^k \\ t_2^m \end{bmatrix}, \quad k, m \in \{1, \dots, q\}, \quad j = (k - 1)q + m,$$

based on sets of one-dimensional interpolation points $\{t_1^1, \dots, t_1^q\}$ and $\{t_2^1, \dots, t_2^q\}$. As explained in § 2.2.1.37, cf. (2.2.1.38), in this case the cardinal functions are given by products of 1D Lagrange polynomials

$$b_j^x(\mathbf{x}) = L_k^1(x_1) \cdot L_m^2(x_2), \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad k, m \in \{1, \dots, q\}, \quad j = (k - 1)q + m. \quad (2.3.4.19)$$

This suggests the following algorithm for the computations of \mathbf{U} :

- ❶ For $* = 1, 2$ precompute the weights

$$\lambda_i = \frac{1}{(t_*^i - t_*^1) \dots (t_*^i - t_*^{i-1})(t_*^i - t_*^{i+1}) \dots (t_*^i - t_*^q)}, \quad i = 1, \dots, q.$$

- ❷ For all collocation points $\mathbf{x}^j = \begin{bmatrix} x_1^j \\ x_2^j \end{bmatrix}$ do:

- (i) Compute $L_\ell^*(\mathbf{x}_*^j)$, $* = 1, 2$, using the formula (2.3.4.18).
- (ii) Form the tensor product matrix $\mathbf{L}_j := \left[L_k^1(x_1^j) \cdot L_m^2(x_2^j) \right]_{k,m=1}^q \in \mathbb{R}^{q,q}$.
- (iii) Reshape \mathbf{K}_j as a row vector of length q^2 and insert it into \mathbf{U} as j -th row.

┘

§2.3.4.20 (Storage requirements) A general expression for the amount of storage required by an instance of **BlockPartition** (\rightarrow Code 2.3.3.3) has already been given in (2.3.1.22):

$$\text{storage}(\widetilde{\mathbf{M}}) = \sum_{k \in \mathbb{F}_{\text{far}}} q(\#I_k + \#J_k) + \sum_{k \in \mathbb{F}_{\text{near}}} \#I_k \cdot \#J_k. \quad (2.3.1.22)$$

Now we are going to refine this expression for the partition generated by the clustering algorithm of Code 2.3.3.5 based on the *binary* cluster tree recursively built as in Code 2.3.2.27 and with the admissibility condition (2.3.3.6). To obtain the rank- q far-field blocks we employ bi-directional interpolation, which results in a matrix factorization as in (2.3.4.8)

$$\widetilde{\mathbf{M}} \Big|_{v \times w} = \mathbf{U}_v \cdot \mathbf{C}_{v \times w} \cdot \mathbf{V}_w^\top, \quad (2.3.4.21)$$

$$\mathbf{U}_v \in \mathbb{R}^{\#\mathcal{I}(v), q}, \quad \mathbf{C}_{v \times w} \in \mathbb{R}^{q, q}, \quad \mathbf{V}_w^\top \in \mathbb{R}^{\#\mathcal{I}(w), q}. \quad (2.3.4.22)$$

The factors \mathbf{U}_v and \mathbf{V}_w are stored in the respective nodes.

The if-statement in Line 6 of `ClusterTree<Node>::buildRec()` ensures that there is a small number $r_L \in \mathbb{N}$ that bounds the number of indices held by the leaves of the cluster trees:

$$\forall u \in \mathcal{T}_*: \quad u \text{ is leaf} \Rightarrow \#\mathcal{I}(u) \leq r_L, \quad * = \mathbb{I}, \mathbb{J}. \quad (2.3.4.23)$$

On the one hand, in `BlockPartition<>::buildRec()` from Code 2.3.3.5 the admissibility check of Line 6 rules out that a cluster pair containing one leaf cluster is added to the far field set. On the other hand, the if-statement of Line 21 has such a cluster pair invariably added to the near field set:

$$(v, w) \in \mathbb{F}_{\text{near}} \Rightarrow v \text{ is leaf of } \mathcal{T}_{\mathbb{I}} \text{ or } w \text{ is leaf of } \mathcal{T}_{\mathbb{J}}. \quad (2.3.4.24)$$

These insights combined lead to the estimate

$$(v, w) \in \mathbb{F}_{\text{near}} \Rightarrow \#\mathcal{I}(v) \cdot \#\mathcal{I}(w) \leq r_L(\#\mathcal{I}(v) + \#\mathcal{I}(w)). \quad (2.3.4.25)$$

Thus, the amount of memory required by an instance of **BlockPartition** is bounded by

$$\text{storage}(\widetilde{\mathbf{M}}) \leq \sum_{(v,w) \in \mathbb{F}_{\text{far}}} q^2 + \sum_{v \in \mathcal{T}_{\mathbb{I}}} q \cdot \#\mathcal{I}(v) + \sum_{w \in \mathcal{T}_{\mathbb{J}}} q \cdot \#\mathcal{I}(w) + \sum_{(v,w) \in \mathbb{F}_{\text{near}}} r_L(\#\mathcal{I}(v) + \#\mathcal{I}(w)).$$

For the last three terms in this sum the cluster tree structure immediately gives the estimates

$$\sum_{v \in \mathcal{T}_{\mathbb{I}}} \#\mathcal{I}(v) \leq \text{depth}(\mathcal{T}_{\mathbb{I}}) \cdot n, \quad \sum_{w \in \mathcal{T}_{\mathbb{J}}} \#\mathcal{I}(w) \leq \text{depth}(\mathcal{T}_{\mathbb{J}}) \cdot m. \quad (2.3.4.26)$$

To tackle the sum over the far-field pairs we have to make an assumption on the sparsity of the block partition [GH03]:

Definition 2.3.4.27. Sparsity measure of block partition

Let $\mathbb{F} := \{I_k \times J_k\}_k$ be a block partition of $\mathbb{D} := \mathbb{I} \times \mathbb{J}$ based on the cluster trees $\mathcal{T}_{\mathbb{I}}$ and $\mathcal{T}_{\mathbb{J}}$. Then the **sparsity measure** of \mathbb{F} bounds the number of occurrences of a cluster in cluster pairs

$$\text{spm}(\mathbb{F}) := \max \left\{ \max_{v \in \mathcal{T}_{\mathbb{I}}} \#\{w \in \mathcal{T}_{\mathbb{J}} : (v, w) \in \mathbb{F}\}, \max_{w \in \mathcal{T}_{\mathbb{J}}} \#\{v \in \mathcal{T}_{\mathbb{I}} : (v, w) \in \mathbb{F}\} \right\}. \quad (2.3.4.28)$$

Thus, the sparsity measure $\text{spm}(\mathbb{F}_{\text{far}})$ counts the maximal number of far-field blocks to which a single cluster can contribute.

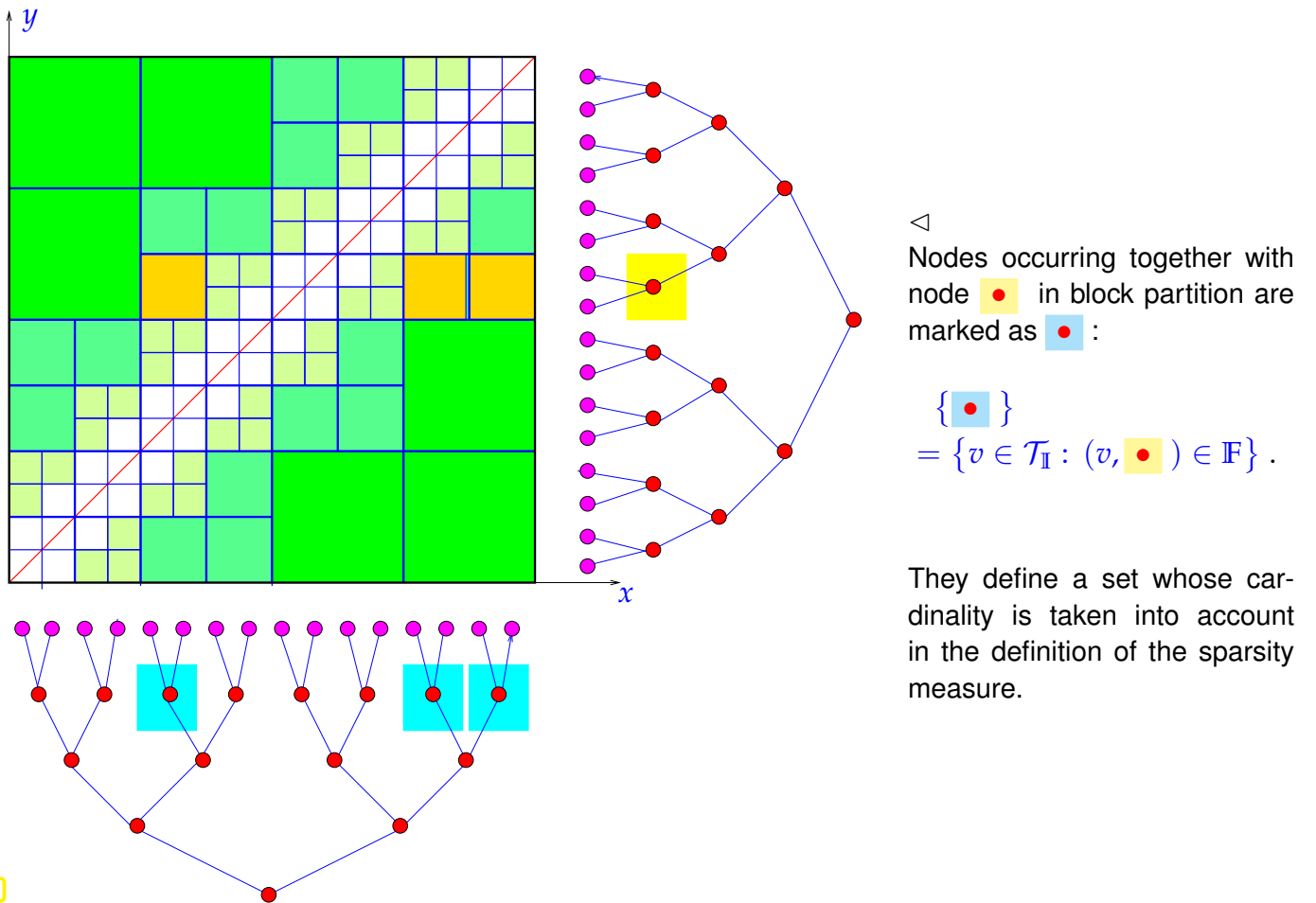


Fig. 104

The next estimates are immediate from the definition of the sparsity measure:

$$\begin{aligned} \sum_{(v,w) \in \mathbb{F}_{\text{far}}} 1 &= \sum_{v \in \mathcal{T}_{\mathbb{I}}} \#\{w \in \mathcal{T}_{\mathbb{J}} : (v, w) \in \mathbb{F}_{\text{far}}\} = \sum_{w \in \mathcal{T}_{\mathbb{J}}} \#\{v \in \mathcal{T}_{\mathbb{I}} : (v, w) \in \mathbb{F}_{\text{far}}\} \\ &\leq \text{spm}(\mathbb{F}) \cdot \min\{\#\mathcal{T}_{\mathbb{I}}, \#\mathcal{T}_{\mathbb{J}}\}, \\ \sum_{(v,w) \in \mathbb{F}_{\text{near}}} \#\mathcal{I}(v) + \#\mathcal{I}(w) &\leq \text{spm}(\mathbb{F}) \cdot \left(\sum_{v \in \mathcal{T}_{\mathbb{I}}} \#\mathcal{I}(v) + \sum_{w \in \mathcal{T}_{\mathbb{J}}} \#\mathcal{I}(w) \right) \\ &\leq \text{spm}(\mathbb{F}) \cdot (n \text{ depth}(\mathcal{T}_{\mathbb{I}}) + m \text{ depth}(\mathcal{T}_{\mathbb{J}})). \end{aligned}$$

For the cluster trees $\mathcal{T}_{\mathbb{I}}$ and $\mathcal{T}_{\mathbb{J}}$ the total number of clusters is smaller than $\text{depth}(\mathcal{T}_{\mathbb{I}}) \cdot \#\mathbb{I}$ or $\text{depth}(\mathcal{T}_{\mathbb{I}}) \cdot \#\mathbb{J}$,

respectively. Thus we conclude

$$\sum_{(v,w) \in \mathbb{F}_{\text{far}}} q^2 \leq q^2 \text{spm}(\mathbb{F}) \cdot \min\{\text{depth}(\mathcal{T}_{\mathbb{I}}) \cdot n, \text{depth}(\mathcal{T}_{\mathbb{J}}) \cdot m\}, \quad (2.3.4.29)$$

which highlights the key role of the sparsity measure when gauging the efficiency of clustering algorithms. For balanced binary cluster trees as built by `buildRec()` we obtain

$$\text{storage}(\widetilde{\mathbf{M}}) \leq ((r_L + q^2)\text{spm}(\mathbb{F}) + q) \cdot (n \lceil \log_2 n \rceil + m \lceil \log_2 m \rceil). \quad (2.3.4.30)$$

Remark 2.3.4.31 (Bounding the sparsity measure) Our policy of using balanced trees as basis for block partitions as implemented in the `buildRec()` functions of Code 2.3.2.27 and Code 2.3.3.5 does not permit us to bound the sparsity measure of the resulting $\mathbb{F} := \mathbb{F}_{\text{far}} \cup \mathbb{F}_{\text{near}}$, unless some uniformity of the distribution of collocation points is assumed.

An alternative geometric clustering policy similar to the quadtree-based approach of Ex. 2.3.2.1 [GH03] makes possible rigorous bounds on $\text{spm}(\mathbb{F})$, but for general locations of collocation points the depth of the cluster trees may grow linearly with $\#\mathbb{I}/\#\mathbb{J}$.

EXPERIMENT 2.3.4.32 (Sparsity measure for clustering in 1D) We revisit the setting of Ex. 2.3.3.8; based balanced binary cluster tree and the algorithm of Code 2.3.3.5 with admissibility parameter $\eta_0 = \frac{1}{2}$ (\rightarrow (2.3.3.6)) we carry out the far-field/near-field partitioning of $\{1, \dots, n\} \times \{1, \dots, n\}$, $n \geq 2$, for the following two distributions of collocation points $\in [0, 1]$

$$(I) \quad \xi_i = \eta_i = \frac{i}{n-1} \quad \text{and} \quad (II) \quad \xi_i = \eta_i = \sqrt{\frac{i}{n-1}}, \quad i = 0, \dots, n-1.$$

For the resulting block partitionings we compute the **sparsity measure** (\rightarrow Def. 2.3.4.27) of \mathbb{F}_{far} and \mathbb{F}_{near} as a function of the number n of collocation points in one direction.

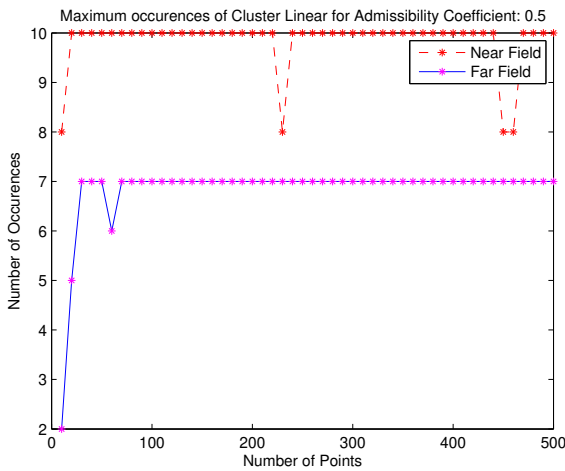


Fig. 105

$$\xi_i = \eta_i = \frac{i}{n-1}, \quad i = 0, \dots, n$$

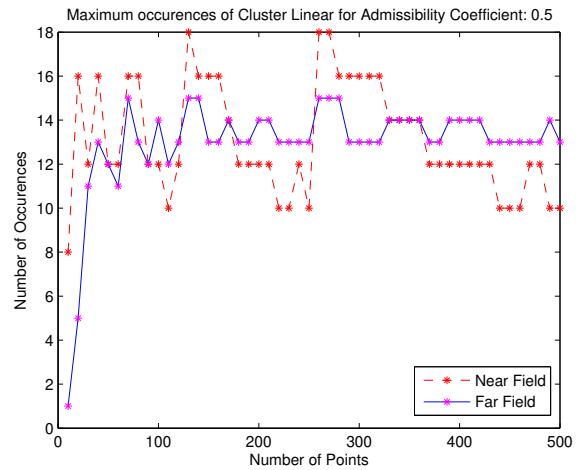


Fig. 106

$$\xi_i = \eta_i = \sqrt{\frac{i}{n-1}}, \quad i = 0, \dots, n$$

2.3.5 Matrix \times Vector: Efficient Implementation

We discuss the implementation of $\widetilde{\mathbf{M}} \cdot \vec{\mu}$, $\vec{\mu} \in \mathbb{R}^m$ in the setting of the previous section and for separable kernel approximation by bi-directional interpolation, see § 2.3.4.6, in particular (2.3.4.8): For a cluster pair $(v, w) \in \mathbb{F}_{\text{far}}$, $v \in \mathcal{T}_{\mathbb{I}}$ (“ x -cluster tree”), $w \in \mathcal{T}_{\mathbb{J}}$ (“ y -cluster tree”) we have

$$\widetilde{\mathbf{M}} \Big|_{v \times w} = \mathbf{U}_v \mathbf{C}_{v \times w} \mathbf{V}_w^T = \underbrace{\left[b_k^v(x^i) \right]_{\substack{i \in \mathcal{I}(v) \\ k=1, \dots, q}}}_{\in \mathbb{R}^{\#\mathcal{I}(v), q}} \underbrace{\left[G(t_v^k, t_w^\ell) \right]_{k, \ell=1, \dots, q}}_{\in \mathbb{R}^{q, q}} \underbrace{\left[b_\ell^w(y^j) \right]_{\substack{j \in \mathcal{I}(w) \\ \ell=1, \dots, q}}}_{\in \mathbb{R}^{q, \#\mathcal{I}(w)}}.$$

We adapt the general algorithm given in Code 2.3.1.25 to this situation. To elucidate the ideas we introduce two essential operations and their matrix representations:

- ❶ **Restrict-to-cluster:** For $w \in \mathcal{T}_{\mathbb{J}}$ we define the extraction of a sub-vector defined by the cluster's index set

$$\mathbf{R}_w : \mathbb{R}^m \rightarrow \mathbb{R}^{\#\mathcal{I}(w)} , \quad \mathbf{R}_w(\vec{\mu}) := \begin{bmatrix} \mu_{j_1} \\ \vdots \\ \mu_{j_\ell} \end{bmatrix} , \quad \text{with } \mathcal{I}(w) = \{j_1, \dots, j_\ell\}, \ell := \#\mathcal{I}(w) . \quad (2.3.5.1)$$

This is a linear mapping and can be described by a “fat” matrix $\mathbf{R}_w \in \{0, 1\}^{\#\mathcal{I}(w), m}$.

- ❷ **Expand-from-cluster:** For $v \in \mathcal{T}_{\mathbb{I}}$ we introduce the embedding of cluster-associated sub-vector into a full vector,

$$\mathbf{E}_v : \mathbb{R}^{\#\mathcal{I}(v)} \rightarrow \mathbb{R}^n , \quad (\mathbf{E}_v \vec{v})_i := \begin{cases} v_\ell & , \text{ if } i_\ell = i , \\ 0 & , \text{ if } k \notin \mathcal{I}(v) , \end{cases} \quad \text{with } \mathcal{I}(v) = \{i_1, \dots, i_k\}, k := \#\mathcal{I}(v) . \quad (2.3.5.2)$$

The matrix associated with \mathbf{E}_v will be denoted by $\mathbf{E}_v \in \{0, 1\}^{n, \#\mathcal{I}(v)}$.

Remark 2.3.5.3 (Expand and restrict as adjoint operations) If $\mathbb{I} = \mathbb{J}$ and $\mathcal{T}_{\mathbb{I}} = \mathcal{T}_{\mathbb{J}}$ ($m = n$ and same cluster tree for both directions), then we have $\mathbf{E}_v = \mathbf{R}_v^\top$. \lrcorner

§2.3.5.4 (Matrix \times vector: three-pass algorithm) The reduction/expansion operations make it possible to write the multiplication of $\widetilde{\mathbf{M}}$ with a vector in a concise way:

$$\begin{aligned} \widetilde{\mathbf{M}}\vec{\mu} &= \sum_{(v,w) \in \mathbb{F}_{\text{far}} \cup \mathbb{F}_{\text{near}}} \mathbf{E}_v \cdot \widetilde{\mathbf{M}} \Big|_{v \times w} \cdot \mathbf{R}_w \vec{\mu} \\ &= \sum_{(v,w) \in \mathbb{F}_{\text{near}}} \mathbf{E}_v \cdot \mathbf{M} \Big|_{v \times w} \cdot \mathbf{R}_w \vec{\mu} + \sum_{(v,w) \in \mathbb{F}_{\text{far}}} (\mathbf{E}_v \mathbf{U}_v) \mathbf{C}_{v \times w} (\mathbf{V}_w^\top \mathbf{R}_w) \vec{\mu} . \end{aligned} \quad (2.3.5.5)$$

This suggests a **3-pass** approach:

- (I) For each $w \in \mathcal{T}_{\mathbb{J}}$ compute $\vec{\omega}_w := \mathbf{V}_w^\top \mathbf{R}_w \vec{\mu} \in \mathbb{R}^q$.

► Total effort = $\sum_{w \in \mathcal{T}_{\mathbb{J}}} q \#\mathcal{I}(w) = O(q m \log m)$ for $m \rightarrow \infty$.

- (II) In parallel carry out the following operations:

- For each cluster pair $(v, w) \in \mathbb{F}_{\text{far}}$ update $\vec{\zeta}_v \leftarrow \vec{\zeta}_v + \mathbf{C}_{v \times w} \vec{\omega}_w$, $\vec{\zeta}_v \in \mathbb{R}^{\#\mathcal{I}(v)}$.
- For each cluster pair $(v, w) \in \mathbb{F}_{\text{near}}$ update $\vec{\phi}_v \leftarrow \vec{\phi}_v + \mathbf{M} \Big|_{v \times w} \mathbf{R}_w \vec{\mu}$, $\vec{\phi}_v \in \mathbb{R}^{\#\mathcal{I}(v)}$.

Total effort = ?

- (III) For each $v \in \mathcal{T}_{\mathbb{I}}$ do $\vec{\rho} \leftarrow \vec{\rho} + \mathbf{E}_v (\mathbf{U}_v \vec{\zeta}_v + \vec{\phi}_v)$, $\vec{\rho} \in \mathbb{R}^n$.

► Total effort = $\sum_{v \in \mathcal{T}_{\mathbb{I}}} \#\mathcal{I}(v) = O(n \log n)$

Of course, all vectors into which we accumulate results have to be initialized with zero. \lrcorner

§2.3.5.6 (Complexity Estimates) We adopt the setting and notations of § 2.3.4.20. In Section 2.3.1 we have derived a general estimate for the effort of matrix \times vector multiplication with $\widetilde{\mathbf{M}}$:

$$\text{cost}(\widetilde{\mathbf{M}} \times \text{vector}) = \sum_{k \in \mathbb{F}_{\text{far}}} q(\#\mathcal{I}_k + \#\mathcal{J}_k) + \sum_{k \in \mathbb{F}_{\text{near}}} \#\mathcal{I}_k \cdot \#\mathcal{J}_k . \quad (2.3.1.27)$$

Since this bound is the same that for the storage requirements in (2.3.1.22), we can appeal to the derivation of (2.3.4.30) and get

$$\text{cost}(\widetilde{\mathbf{M}} \times \text{vector}) \leq (r_L + q^2) \text{spm}(\mathbf{F}) \cdot (n \lceil \log_2 n \rceil + m \lceil \log_2 m \rceil) . \quad (2.3.5.7)$$

┘

2.3.6 Panel Clustering

We discuss the application of clustering techniques for the local low-rank compression of **boundary element Galerkin matrices** as they have been introduced in Section 1.4 and Section 1.5. We recall the general setting

- ◆ The domain $\Omega \subset \mathbb{R}^d$, $d = 2, 3$, is a bounded curved Lipschitz polygon/polyhedron with boundary $\Gamma := \partial\Omega$.
- ◆ The boundary Γ is equipped with a mesh $\mathcal{G} = \{\pi_k\}_{k=1}^K$ according to Def. 1.4.2.5 ($d = 2$) or Def. 1.5.1.4 ($d = 3$).
- ◆ Based on \mathcal{G} we build a boundary element space V_N , either $\mathcal{S}_{p-1}^{-1}(\mathcal{G})$ or $\mathcal{S}_p^0(\mathcal{G})$, $p \in \mathbb{N}$, see (1.4.2.10)/(1.5.2.5) and (1.4.2.11)/(1.5.2.6), piecewise polynomial under edge/face-wise pullback to the parameter domain.
- ◆ The boundary element space is spanned by **locally supported nodal basis functions**:

$$V_N = \text{Span}\{b_N^1, \dots, b_N^N\}, \quad N := \dim V_N .$$

Refer to Ex. 1.4.2.17, Ex. 1.4.2.19, Ex. 1.5.2.18, and Ex. 1.5.2.19 for concrete examples.

Then the entries of the Galerkin matrix associated with the single layer boundary integral operator \mathbf{V} for $-\Delta$ read

$$\mathbf{V} := \left[\int_{\Gamma} \int_{\Gamma} G^\Delta(\mathbf{x}, \mathbf{y}) b_N^j(\mathbf{y}) b_N^i(\mathbf{x}) dS(\mathbf{y}) dS(\mathbf{x}) \right]_{i,j=1}^N \in \mathbb{R}^{N,N}, \quad (2.3.6.1)$$

with the fundamental solution

$$G^\Delta(\mathbf{x}, \mathbf{y}) = \begin{cases} -\frac{1}{2\pi} \log \|\mathbf{x} - \mathbf{y}\| & , \text{ if } d = 2, \\ \frac{1}{4\pi} \frac{1}{\|\mathbf{x} - \mathbf{y}\|} & , \text{ if } d = 3. \end{cases} \quad (1.2.2.33)$$

Note that G^Δ provides an **asymptotically smooth singular** kernel function, see Rem. 2.2.2.1. As such it allows rank- q separable approximation on “admissible” boxes $\subset \mathbb{R}^d \times \mathbb{R}^d$, exponentially accurate in q , in the spirit of Ass. 2.2.2.36.

In order to transfer the clustering techniques from kernel collocation matrices to \mathbf{V} we have to answer two questions:

Q1 What will play the role of the collocation points \mathbf{x}^i and \mathbf{y}^j ?

Q2 How to obtain low-rank approximations of “admissible” blocks of \mathbf{V} ?

§2.3.6.2 (Answer to Q1) The index sets will be $\mathbb{I} = \mathbb{J} = \{1, \dots, N\}$, that is $n, m = N$, and instead of collocation points we consider the **basis functions** b_N^i , $i = 1, \dots, N$. Recall that each basis function has a small support $\text{supp}(b_N^i) \subset \Gamma$. These will be used to define **bounding boxes** for sets of basis functions, cf. Def. 2.3.2.19. For $I \subset \mathbb{I}$ we define

$$\text{box}\{b_N^i\}_{i \in I} = \prod_{\ell=1}^d \left[\min\{x_\ell : \mathbf{x} \in \bigcup_{i \in I} \text{supp}(b_N^i)\}, \max\{x_\ell : \mathbf{x} \in \bigcup_{i \in I} \text{supp}(b_N^i)\}, \right]. \quad (2.3.6.3)$$

This also defines the bounding box $\text{box}(v)$ of each node v of a cluster tree (\rightarrow Def. 2.3.2.14) for \mathbb{I} , because v can be identified with a unique subset of indices/basis functions. Given bounding boxes we can compute the diameter of a cluster and the distance of two clusters in the usual way, see Code 2.3.2.21.

The following could be a replacement of the **Point** class from Code 2.3.2.21.

C++ code 2.3.6.4: Data type boundary element basis function \rightarrow [GITLAB](#)

```

2  template <int d> // dimension d as template argument
3  struct BasisFn {
4      std::size_t idx; // Index of basis function
5      Eigen::Matrix<double, d, 1> xmin, xmax; // Corners of bounding box
6  };

```

§2.3.6.5 (Answer to Q2) Assume that we have run the clustering algorithm and constructed far-field/near-field block partition. Consider a cluster $(v, w) \in \mathbb{F}_{\text{far}}$. Hence $(i, j) \in \mathcal{I}(v) \times \mathcal{I}(w)$ means that

$$\text{supp}(b_N^i) \times \text{supp}(b_N^j) \subset B := \text{box}(v) \times \text{box}(w), \quad \eta(B) \leq \eta_0, \quad (2.3.6.6)$$

$\eta(B)$ the admissibility measure from (2.2.2.7) and $\eta_0 > 0$ the admissibility threshold.

Thanks to Ass. 2.2.2.36, on B we can get a rank- q separable approximation of G^Δ by means of **bi-directional interpolation**, see (2.2.1.46),

$$G^\Delta \Big|_B(x, y) \approx \sum_{k=1}^q \sum_{\ell=1}^q G^\Delta(\mathbf{t}_x^k, \mathbf{t}_y^\ell) c_k^x(x) c_\ell^y(y), \quad (x, y) \in B. \quad (2.3.6.7)$$

with interpolation nodes \mathbf{t}_x^k for $\text{box}(v)$, \mathbf{t}_y^ℓ for $\text{box}(w)$, and associated cardinal basis functions c_k^x and c_ℓ^y . We plug this approximation into the double integrals defining the entries of the Galerkin matrix \mathbf{V} from (2.3.6.1):

$$\begin{aligned}
 \mathbf{V}|_{v \times w} &\approx \left[\sum_{k=1}^q \sum_{\ell=1}^q G^\Delta(\mathbf{t}_x^k, \mathbf{t}_y^\ell) \int_\Gamma c_\ell^y(\mathbf{y}) b_N^j(\mathbf{y}) dS(\mathbf{y}) \cdot \int_\Gamma c_k^x(\mathbf{x}) b_N^i(\mathbf{x}) dS(\mathbf{x}) \right]_{\substack{i \in \mathcal{I}(v) \\ j \in \mathcal{I}(w)}} \\
 &= \left[\int_\Gamma c_k^x(\mathbf{x}) b_N^i(\mathbf{x}) dS(\mathbf{x}) \right]_{\substack{i \in \mathcal{I}(v) \\ k=1, \dots, q}} \cdot \left[G^\Delta(\mathbf{t}_x^k, \mathbf{t}_y^\ell) \right]_{k, \ell=1, \dots, q} \left[\int_\Gamma c_\ell^y(\mathbf{y}) b_N^j(\mathbf{y}) dS(\mathbf{y}) \right]_{\substack{j \in \mathcal{I}(w) \\ \ell=1, \dots, q}}^\top \\
 &= \mathbf{U}_v \cdot \mathbf{C} \cdot \mathbf{V}_w^\top \in \mathbb{R}^{\#\mathcal{I}(v), \#\mathcal{I}(w)},
 \end{aligned} \quad (2.3.6.8)$$

which gives us a rank- q matrix already in triple-factor form, cf. (2.2.1.47).

If we rely on tensor-product polynomial interpolation, the cardinal functions c_k^x and c_ℓ^y will be product of Lagrange polynomials. As a consequence, the integrands of the integrals defining the matrices \mathbf{U} and \mathbf{V} will be analytic after local analytic pullback to the parameter domain/reference element. For instance, the contribution of a single panel π with associated local parameterization $\gamma_\pi : \hat{K} \rightarrow \pi$ (\rightarrow § 1.4.2.24, § 1.5.2.15) is

$$\int_\pi c_k^x(\mathbf{x}) b_N^i(\mathbf{x}) dS(\mathbf{x}) = \int_{\hat{K}} c_k^x(\gamma_\pi(\hat{\mathbf{x}})) \hat{b}^i(\hat{\mathbf{x}}) \sqrt{\det(D\gamma_\pi(\hat{\mathbf{x}})^\top D\gamma_\pi(\hat{\mathbf{x}}))} d\hat{\mathbf{x}}, \quad (2.3.6.9)$$

where $\hat{b}^j : \hat{K} \rightarrow \mathbb{R}$ is the **polynomial** (!) reference shape function spawning b_N^i : $\hat{b}^j = \gamma_\pi^* b_N^i|_\pi$, see (1.4.2.27). The integrand in (2.3.6.9) will inherit analyticity from γ and can be evaluated accurately by (exponentially converging) families of high-order numerical quadrature rules on \hat{K} . \square

Remark 2.3.6.10 (Compressing discrete BIEs with double layer kernels) The entries of boundary element Galerkin matrices for the double layer boundary integral operator \mathbf{K} with the integral representation formula

$$\mathbf{K}(\mathbf{v})(\mathbf{x}) = \int_{\Gamma} \frac{\mathbf{x} - \mathbf{y}}{\omega_d \|\mathbf{x} - \mathbf{y}\|^d} \cdot \mathbf{n}(\mathbf{y}) \mathbf{v}(\mathbf{y}) \, dS(\mathbf{y}), \quad \mathbf{x} \in \text{smooth part of } \Gamma, \quad (1.3.4.14)$$

are given by the singular integrals

$$(\mathbf{K})_{i,j} = \int_{\Gamma} \int_{\Gamma} \frac{\mathbf{x} - \mathbf{y}}{\omega_d \|\mathbf{x} - \mathbf{y}\|^d} \cdot \mathbf{n}(\mathbf{y}) b_N^j(\mathbf{y}) \beta_N^i(\mathbf{x}) \, dS(\mathbf{y}) \, dS(\mathbf{x}), \quad (2.3.6.11)$$

where $\{b_N^j\}_{j=1}^N$ is a nodal basis of $\mathcal{S}_p^0(\mathcal{G})$, $p \in \mathbb{N}$, and $\{\beta_N^i\}_{i=1}^K$ a nodal basis of $\mathcal{S}_{p-1}^{-1}(\mathcal{G})$, see Section 1.4.2.3 and Section 1.5.2.2.

Following the policy of § 2.3.6.5 and interpolating the singular, asymptotically smooth kernel $k_{\mathbf{K}}(\mathbf{x}, \mathbf{y}) := \frac{\mathbf{x} - \mathbf{y}}{\omega_d \|\mathbf{x} - \mathbf{y}\|^d} \cdot \mathbf{n}(\mathbf{y})$ on far-field boxes encounters difficulties, because it requires its evaluation also off the boundary Γ , where the normal vector field \mathbf{n} is not defined!

We remember that

$$k_{\mathbf{K}}(\mathbf{x}, \mathbf{y}) := \frac{\mathbf{x} - \mathbf{y}}{\omega_d \|\mathbf{x} - \mathbf{y}\|^d} \cdot \mathbf{n}(\mathbf{y}) = \mathbf{grad}_{\mathbf{y}} G^{\Delta}(\mathbf{x}, \mathbf{y}) \cdot \mathbf{n}(\mathbf{y}), \quad \mathbf{x}, \mathbf{y} \in \Gamma, \quad \mathbf{x} \neq \mathbf{y}. \quad (2.3.6.12)$$



Idea: Obtain a separable approximation of the double layer kernel $k_{\mathbf{K}}$ by applying the differential operator $\mathbf{n}(\mathbf{y}) \cdot \mathbf{grad}_{\mathbf{y}}$ to a birectional interpolant of G^{Δ} !

Recalling (2.3.6.7), this leads to the rank- q separable approximation

$$k_{\mathbf{K}}(\mathbf{x}, \mathbf{y}) \approx \sum_{k=1}^q \sum_{\ell=1}^q G^{\Delta}(\mathbf{t}_x^k, \mathbf{t}_y^{\ell}) c_k^x(\mathbf{x}) (\mathbf{grad}_{\mathbf{y}} c_{\ell}^y)(\mathbf{y}) \cdot \mathbf{n}(\mathbf{y}), \quad (\mathbf{x}, \mathbf{y}) \in B \cap \Gamma, \quad (2.3.6.13)$$

where $B \subset \mathbb{R}^d \times \mathbb{R}^d$ is a far-field box as in (2.3.6.6), associated to the cluster pair (\mathbf{v}, \mathbf{w}) . We end up with the rank- q matrix block

$$\begin{aligned} \mathbf{K}|_{\mathbf{v} \times \mathbf{w}} &\approx \left[\sum_{k=1}^q \sum_{\ell=1}^q G^{\Delta}(\mathbf{t}_x^k, \mathbf{t}_y^{\ell}) \int_{\Gamma} (\mathbf{grad}_{\mathbf{y}} c_{\ell}^y(\mathbf{y}) \cdot \mathbf{n}(\mathbf{y})) b_N^j(\mathbf{y}) \, dS(\mathbf{y}) \cdot \int_{\Gamma} c_k^x(\mathbf{x}) b_N^i(\mathbf{x}) \, dS(\mathbf{x}) \right]_{\substack{i \in \mathcal{I}(\mathbf{v}) \\ j \in \mathcal{I}(\mathbf{w})}} \\ &= \left[\int_{\Gamma} c_k^x(\mathbf{x}) b_N^i(\mathbf{x}) \, dS(\mathbf{x}) \right]_{\substack{i \in \mathcal{I}(\mathbf{v}) \\ k=1, \dots, q}} \cdot \left[G^{\Delta}(\mathbf{t}_x^k, \mathbf{t}_y^{\ell}) \right]_{k, \ell=1, \dots, q} \left[\int_{\Gamma} (\mathbf{grad}_{\mathbf{y}} c_{\ell}^y(\mathbf{y}) \cdot \mathbf{n}(\mathbf{y})) b_N^j(\mathbf{y}) \, dS(\mathbf{y}) \right]_{\substack{j \in \mathcal{I}(\mathbf{w}) \\ \ell=1, \dots, q}}^{\top} \\ &= \mathbf{U}_{\mathbf{v}} \cdot \mathbf{C} \cdot \mathbf{V}_{\mathbf{w}}^{\top} \in \mathbb{R}^{\#\mathcal{I}(\mathbf{v}), \#\mathcal{I}(\mathbf{w})}. \end{aligned} \quad (2.3.6.14)$$

┘

Remark 2.3.6.15 (Iterative solution methods for linear systems of equations → [NumCSE Chapter 10]) After local low-rank compression the boundary element Galerkin matrices are available only in a special data format like **PartMatrix** from Code 2.3.1.24. However, direct solution algorithms for dense linear systems of equations like **Gaussian elimination** [NumCSE Section 2.3] usually operate on matrices stored in contiguous memory.

Direct elimination-based solution methods for linear systems of equations cannot be applied to system matrices compressed with clustering techniques.

Fortunately, the matrix data formats arising from local low-rank compression support fast matrix×vector operations, see Code 2.3.1.25 and Section 2.3.5. Thus, they well mesh with **iterative solution methods** for linear systems of equations that can compute approximate solutions with a prescribed tolerance based on **system matrix×vector operations alone**.

The typical generic interface to these methods reads:

```
template <typename MatrixType, typename Rhs, typename Dest, typename
  Preconditioner>
void iterative_solver(const MatrixType& mat, const Rhs& rhs, Dest& x,
const Preconditioner& preconditioner, size_t maxit,
typename Dest::RealScalar& tol_error);
```

- **Rhs**, **Dest** have to be vector types, for instance, **Eigen::VectorXd**. The argument `rhs` holds the right-hand side vector and `x` contains the **initial guess** and is also used to return the approximate solution after the iteration has terminated.
- **MatrixType** has to provide a method `Rhs operator * (const Dest &) const` that implements the matrix×vector product. The argument `mat` of this type passes the system matrix, more precisely, the linear operator described by the system matrix.
- The argument `maxit` specifies the maximal number of iterations and `tol_error` a relative tolerance for termination.
- **Preconditioner** is a type for a linear operator providing a method `Dest solve(const Rhs &) const` that is supposed to emulate an approximate inverse of the system matrix. It is meant to accelerate convergence, see [NumCSE Section 10.3]. Default is the identity mapping.

The following iterative solution methods are widely used. They all belong to the class of **Krylov subspace methods**.

- **Conjugate Gradient Method (CG)** [NumCSE Section 10.2]:

Applicable to linear systems of equations with **symmetric positive definite (s.p.d.)** system matrices, like those arising from the Galerkin boundary element discretization of first-kind direct or indirect BIEs for boundary value problems for $-\Delta$, see Section 1.3.5.1, § 1.3.6.3 and § 1.3.6.7.

A single step of the iteration involves one evaluation $\mathbf{A} \times \text{vector}$, one evaluation $\mathbf{P} \times \text{vector}$, three dot products and 3 elementary vector (SAXPY) operations.

Speed of convergence (measured in the energy norm induced by the system matrix) is governed by the spectral condition number $\kappa(\mathbf{PA})$, where \mathbf{A} is the system matrix and \mathbf{P} the matrix representation of the preconditioner, see [NumCSE Thm. 10.2.3.5].

Note that for boundary element Galerkin matrices \mathbf{A} on families of uniformly shape-regular curve/surface meshes we observe $\kappa(\mathbf{A}) = o(h_{\min}^{-1})$, $h_{\min} \hat{=}$ minimal size of panels of the mesh. Therefore, without preconditioner, the CG will converge more slowly on finer meshes.

- **Bi-Conjugate Gradient Stabilized Method (BiCGStab)** [NumCSE Section 10.4.2]:

This iterative method can be applied to general linear systems of equations. Unfortunately, no rigorous convergence theory is available. One step, beside a few dot products and SAXPY operations, one step executes two $\mathbf{A} \times \text{vector}$ and $\mathbf{P} \times \text{vector}$ evaluations.

- **Generalized Minimal Residual Method (GMRES)** [NumCSE Section 10.4.1]:

This is another iterative solution method for general linear systems of equations. It enjoys robust convergence, but in the ℓ -th step ℓ dot products and SAXPY operations have to be carried out, beside a single $\mathbf{A} \times \text{vector}$ and $\mathbf{P} \times \text{vector}$ product.

┘

2.4 Hierarchical Matrices

2.4.1 Definition

§2.4.1.1 (Recap of local low-rank approximation based on cluster trees) The clustering algorithm as presented in Section 2.3 yielded a *data-sparse approximate representation* $\widetilde{\mathbf{M}} \in \mathbb{R}^{n,m}$ of kernel collocation matrices $\mathbf{M} \in \mathbb{R}^{n,m}$ for asymptotically smooth singular kernels like

$$(x, y) \mapsto -\log\|x - y\|, \frac{1}{\|x - y\|}, \frac{(x - y) \cdot \mathbf{f}(y)}{\|x - y\|^d}, \quad x, y \in \mathbb{R}^d, x \neq y,$$

see Rem. 2.2.2.1 for the definition. Key elements of the data structure are

- ◆ **cluster trees** (\rightarrow Def. 2.3.2.14) $\mathcal{T}_{\mathbb{I}}$ and $\mathcal{T}_{\mathbb{J}}$ defining subsets (clusters) and partitions of the index sets $\mathbb{I} := \{1, \dots, n\}$, $\mathbb{J} := \{1, \dots, m\}$,
- ◆ a far-field/near-field **block partition** $\mathbb{F} = \mathbb{F}_{\text{far}} \cup \mathbb{F}_{\text{near}}$ of the product index set $\mathbb{D} := \mathbb{I} \times \mathbb{J}$ recursively built by the algorithm implemented in the method `buildRec()` of Code 2.3.3.5 based on an **admissibility condition** according to Def. 2.3.3.2.
- ◆ a *low-rank factorized representation* of the sub-matrices of $\widetilde{\mathbf{M}}$ corresponding to the far-field blocks.

Recall that each cluster $v \in \mathcal{T}_{\mathbb{I}}$ and $w \in \mathcal{T}_{\mathbb{J}}$ can be identified with a subset of indices $\mathcal{I}(v) \subset \mathbb{I}$, $\mathcal{I}(w) \subset \mathbb{J}$. This endows \mathbb{F}_{far} , \mathbb{F}_{near} , and $\mathbb{F} := \mathbb{F}_{\text{far}} \cup \mathbb{F}_{\text{near}}$ with two meanings

1. as sets of subsets of the product index set $\mathbb{I} \times \mathbb{J}$,
2. as set of cluster pairs (v, w) , $v \in \mathcal{T}_{\mathbb{I}}$, $w \in \mathcal{T}_{\mathbb{J}}$.

We also remind of the notation $\mathbf{X}|_{v \times w} := (\widetilde{\mathbf{X}})_{\mathcal{I}(v), \mathcal{I}(w)}$ for sub-matrices of a matrix $\mathbf{X} \in \mathbb{R}^{n,m}$. ┘

A special name has been introduced for the data structure built by the clustering algorithm:

Definition 2.4.1.2. Hierarchical matrix

Given $n, m \in \mathbb{N}$, $q \in \mathbb{N}$, a matrix $\mathbf{H} \in \mathbb{R}^{n,m}$ is called a **hierarchical matrix** or **\mathcal{H} -matrix** of local rank q , if there exist

- cluster trees $\mathcal{T}_{\mathbb{I}}$ (**row tree**) and $\mathcal{T}_{\mathbb{J}}$ (**column tree**) for $\mathbb{I} := \{1, \dots, n\}$ and $\mathbb{J} := \{1, \dots, m\}$,
- and an abstract admissibility condition $\text{adm}_{\mathbf{H}} : \mathcal{T}_{\mathbb{I}} \times \mathcal{T}_{\mathbb{J}} \rightarrow \{\text{true}, \text{false}\}$

such that

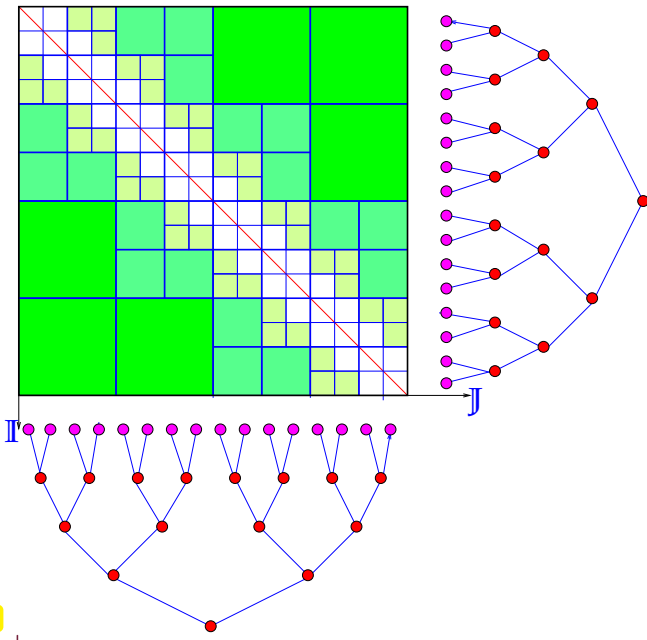
$$\text{rank}(\mathbf{H}|_{v \times w}) \leq q \quad \forall (v, w) \in \mathbb{F} := \mathbb{F}_{\text{far}} \cup \mathbb{F}_{\text{near}} \subset \mathcal{T}_{\mathbb{I}} \times \mathcal{T}_{\mathbb{J}},$$

where

$$\{\mathcal{I}(v) \times \mathcal{I}(w)\}_{(v,w) \in \mathbb{F}} = \{\mathcal{I}(v) \times \mathcal{I}(w)\}_{(v,w) \in \mathbb{F}_{\text{far}}} \cup \{\mathcal{I}(v) \times \mathcal{I}(w)\}_{(v,w) \in \mathbb{F}_{\text{near}}}$$

is a *partition* of $\mathbb{I} \times \mathbb{J}$ generated by the algorithm implemented in the method `buildRec()` of Code 2.3.3.5 based on `adm()`.

EXAMPLE 2.4.1.3 (The prototypical simple \mathcal{H} -matrix)



The essence of the hierarchical matrix data structure for binary trees \mathcal{T}_I and \mathcal{T}_J is captured in the figure beside:

- ◁ ■ $\hat{=}$ matrix blocks in the far field $\in \mathbb{F}_{\text{far}}$
- $\hat{=}$ matrix blocks in the near field $\in \mathbb{F}_{\text{near}}$



Note that Fig. 107 illustrates the rather special case of $n = m$, $\mathcal{T}_I = \mathcal{T}_J$ and that \mathbb{F}_{near} contains only products of leaves of \mathcal{T}_I .

Fig. 107

📎 Notation: Bold greek letters σ, τ, ρ will be used for elements of $\mathcal{T}_I \times \mathcal{T}_J$, so-called **blocks**.

Given $\mathbf{X} \in \mathbb{R}^{n,m}$, a block $\sigma = (v, w)$ singles out the sub-matrix $\mathbf{X}|_\sigma := \mathbf{X}|_{v \times w} := (\mathbf{X})_{\substack{i \in \mathcal{I}(v) \\ j \in \mathcal{I}(w)}}$.

The algorithm of `buildRec()` for the construction of a hierarchical matrix ensures

$$(v, w) \in \mathbb{F}_{\text{far}} \quad \Rightarrow \quad \text{adm}(v, w) = \text{true}, \tag{2.4.1.4}$$

$$(v, w) \in \mathbb{F}_{\text{near}} \quad \Rightarrow \quad v \text{ is a leaf of } \mathcal{T}_I \text{ or } w \text{ is a leaf of } \mathcal{T}_J. \tag{2.4.1.5}$$

Lemma 2.2.1.3 guarantees that for a hierarchical matrix $\mathbf{H} \in \mathbb{R}^{n,m}$ with local rank q (as in Def. 2.4.1.2) holds

$$\forall \sigma = (v, w) \in \mathbb{F}_{\text{far}}: \exists \mathbf{A}_\sigma \in \mathbb{R}^{\#\mathcal{I}(v), q}, \mathbf{B}_\sigma \in \mathbb{R}^{\#\mathcal{I}(w), q}: \mathbf{H}|_\sigma = \mathbf{A}_\sigma \cdot \mathbf{B}_\sigma^\top. \tag{2.4.1.6}$$

Assumption 2.4.1.7. Availability of low-rank factor matrices

Whenever we regard a hierarchical matrix \mathbf{H} (\rightarrow Def. 2.4.1.2) as given, we assume that for each block $\sigma = (v, w) \in \mathbb{F}_{\text{far}}$ all entries of the dense matrices \mathbf{A}_σ and \mathbf{B}_σ as in (2.4.1.6) can be accessed with small constant effort.

Our implementation of `buildRec()` in Code 2.3.2.27 always creates binary cluster trees. This matches the following assumption, which is made for the sake of simplicity and by no means essential for hierarchical matrices and their handling.

Assumption 2.4.1.8. Binary cluster trees

Below we assume that all cluster trees underlying hierarchical matrices are **binary trees** (but not necessarily balanced).

§2.4.1.9 (Block tree)

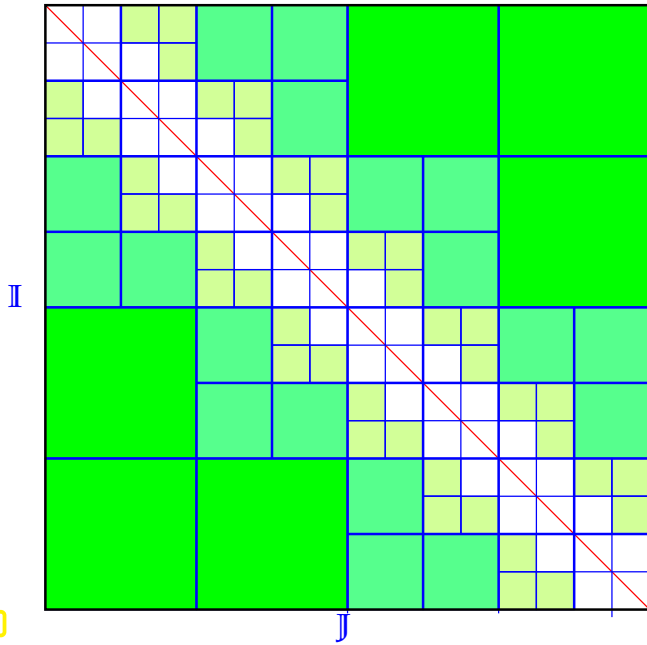


Fig. 108

We return to the \mathcal{H} -matrix of Ex. 2.4.1.3.

We assume $\mathbb{I} := \{1, \dots, n\}, \mathbb{J} := \{1, \dots, m\}$.

The “tiling” of the $n \times m$ -matrix depicted in Fig. 108 is obviously one that can be described by a two-dimensional tree of *quadtrees* type, for which a node can have up to four sons.

More generally, the block partition of every hierarchical matrix \mathbf{H} induced by $\mathbb{F} := \mathbb{F}_{\text{far}} \cup \mathbb{F}_{\text{near}}$ is related to a “quadtrees-type” tree, whose leaves are in one-to-one correspondence to product index sets (\leftrightarrow submatrices of \mathbf{H}) in \mathbb{F} .

The following figures illustrate levels 1–4 of the two-dimensional tree underlying the matrix partition show beside.

Below, each node of the tree has four sons, unless it is a leaf: geometrically, each square is split into four smaller squares.

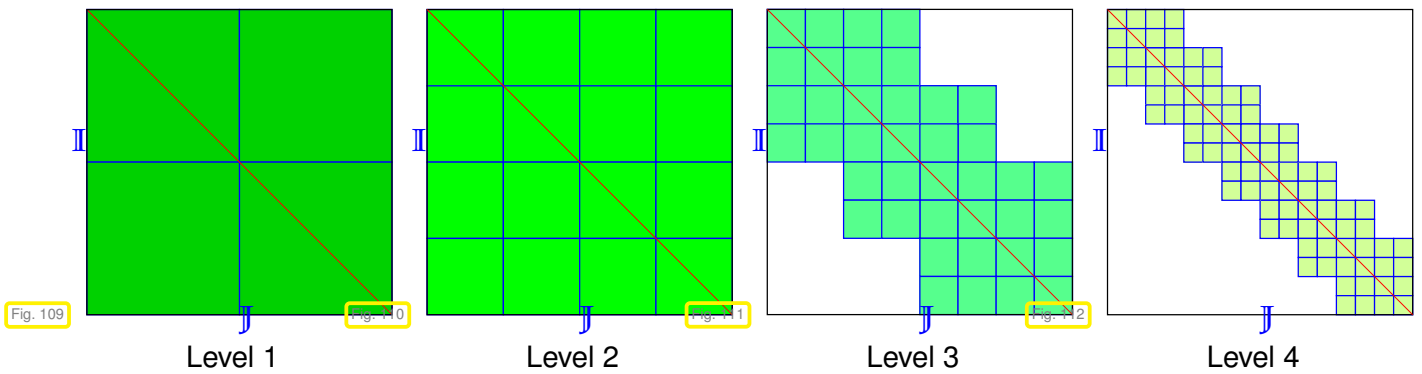


Fig. 109

Fig. 110

Fig. 111

Fig. 112

Now we formalize what we have just observed. Recall that $\mathbb{F} := \mathbb{F}_{\text{far}} \cup \mathbb{F}_{\text{near}}$ is the set of all matrix blocks occurring in the hierarchical matrix, cf. Def. 2.4.1.2.

Definition 2.4.1.10. Block tree underlying a hierarchical matrix

The **block tree** $\mathcal{B}_{\mathbb{I} \times \mathbb{J}}$ for a hierarchical matrix based on the row tree $\mathcal{T}_{\mathbb{I}}$ and column tree $\mathcal{T}_{\mathbb{J}}$ is a tree $(\mathcal{V}, r, \mathcal{E})$ (\rightarrow Def. 2.3.2.10)

- ◆ with pairs of clusters as vertices

$$\mathcal{V} \subset \{(v, w) \in \mathcal{T}_{\mathbb{I}} \times \mathcal{T}_{\mathbb{J}} : \mathcal{I}(v) \times \mathcal{I}(w) \text{ is the union of product index sets in } \mathbb{F}\},$$

- ◆ with root $r := r_{\mathbb{I}} \times r_{\mathbb{J}}$, where r_* is the root of \mathcal{T}_* , $* = \mathbb{I}, \mathbb{J}$,
- ◆ and with the son-father relation defined as

$$\text{sons}(\sigma) = \begin{cases} (\text{sons}(v) \times \text{sons}(w)) \cap \mathcal{V}, & \text{if } \text{sons}(v) \neq \emptyset \text{ and } \text{sons}(w) \neq \emptyset, \\ \emptyset & \text{otherwise,} \end{cases} \tag{2.4.1.11}$$

for all $\sigma = (v, w) \in \mathcal{V}$.

The algorithm implemented in `buildRec()` ensures that Def. 2.4.1.10 defines a tree in the sense of Def. 2.3.2.10.

- The set of leaf nodes of a block tree $\mathcal{B}_{I \times J}$ for a hierarchical matrix can be identified with the set \mathbb{F} of matrix blocks:

$$\mathbb{F} := \{(v, w) \in \mathcal{T}_I \times \mathcal{T}_J : (v, w) \text{ is a leaf of } \mathcal{B}_{I \times J}\}. \quad (2.4.1.12)$$

§2.4.1.13 (Recursive algorithm for building a block tree) The description of the algorithm relies on two abstract data types.

- (I) A data type for the node of a binary cluster tree Def. 2.3.2.14:

Pseudocode 2.4.1.14: (Incomplete) data type for the node of a binary cluster tree

```
1 struct Cluster { Cluster *sons[2] = { nil, nil }; };
```

- (II) A type providing the bare-bones data for the node of a block tree:

Pseudocode 2.4.1.15: Data type for the node of a block tree

```
1 struct BlockNode {
2   BlockNode(Cluster *cr, Cluster *cc);
3   Cluster *cluster_row;
4   Cluster *cluster_col;
5   BlockCluster *sons[4] = { nil, nil, nil, nil };
6   enum { FARFIELD, NEARFIELD, NOLEAF } flag;
7 };
```

The constructor only initializes the **Cluster** * data members. Obviously, this data type lacks matrix-related data members, but will be sufficient to explain the gist of the construction.

The recursive construction is inspired by `builrRec()` from Code 2.3.3.5. It ensures that any node of the block tree containing a leaf node of one of the underlying cluster trees is itself a leaf node of the block tree.

Pseudocode 2.4.1.16: Recursive construction of a block tree

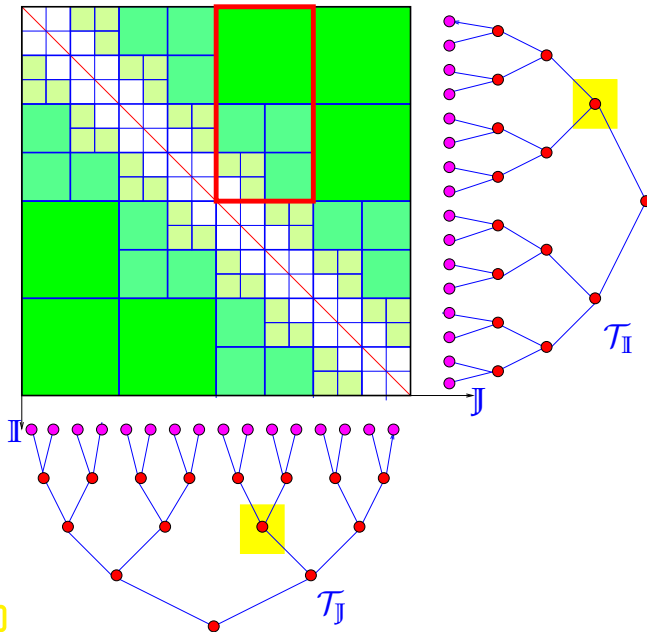
```
1 void buildBlockTree(BlockNode *σ) {
2   Cluster *s[2] = σ->cluster_row.sons;
3   Cluster *t[2] = σ->cluster_col.sons;
4   if ((s[0] = nil) or (s[1] = nil) or (t[0] = nil) or (t[1] = nil)) {
5     // near-field leaf block
6     flag = NEARFIELD;
7   }
8   else {
9     if (adm(σ->cluster_row, σ->cluster_col)) {
10      // far-field leaf block
11      flag = FARFIELD;
12    }
13    else {
14      // recursion
15      flag = NOLEAF;
```

```

16   sons = { BlockNode(s[0], t[0]), BlockNode(s[0], t[1]),
17           BlockNode(s[1], t[0]), BlockNode(s[1], t[1]) };
18   foreach (τ ∈ sons) buildBlockTree(τ);
19   }
    
```

The function assumes that the **Cluster** * data members of its argument have already been initialized properly. Note that every non-leaf node of a block tree built by `buildBlockTree()` will have four children.

§2.4.1.17 (Hierarchical matrices – a recursive data structure)



Let $\mathbf{H} \in \mathbb{R}^{n,m}$ be a hierarchical matrix with local rank q based on the cluster trees \mathcal{T}_I and \mathcal{T}_J . For $v \in \mathcal{T}_I$ and $w \in \mathcal{T}_J$ such that (v,w) belongs to the block tree $\mathcal{B}_{I \times J}$ (\rightarrow Def. 2.4.1.10), $(v,w) \in \mathcal{B}_{I \times J}$, denote by \mathcal{T}_v and \mathcal{T}_w the **sub-trees** (\rightarrow Def. 2.3.2.13) of \mathcal{T}_I and \mathcal{T}_J with roots v and w , respectively. Then $\mathbf{H}|_{v \times w} \in \mathbb{R}^{\#I(v), \#I(w)}$ is another hierarchical matrix of local rank q based on \mathcal{T}_v and \mathcal{T}_w . The admissibility condition remains the same.

◁ hierarchical sub-matrix belonging to a pair of clusters (■).

Fig. 113

In other words, every sub-tree of the block tree $\mathcal{B}_{I \times J}$ defines, through its root, a sub-matrix of \mathbf{H} , which is a valid hierarchical matrix of the same local rank and with the same admissibility condition.

§2.4.1.18 (Recursive algorithm for \mathcal{H} -matrix \times vector) Hierarchical matrices may not be stored in a linear fashion in a data structure similar to that given in Code 2.3.1.24, but in a recursive fashion through a **block tree data structure**. Of course, also in this case the multiplication of a hierarchical matrix with a vector can be done by the algorithm implemented in Code 2.3.1.25, but loops have to be replaced with tree traversal.

Pseudocode 2.4.1.19: Recursive $\vec{\zeta} = \vec{\zeta} + \mathbf{H}\vec{\mu}$

```

1 void hmv( $\mathbf{H} \in \mathbb{R}^{n,m}$ , ref  $\vec{\zeta} \in \mathbb{R}^n$ ,  $\vec{\mu} \in \mathbb{R}^m$ ) {
2    $\sigma :=$  root of block tree for  $\mathbf{H}$ ;
3   if ( $\text{sons}(\sigma) = \emptyset$ ) { // a leaf  $\in \mathbb{F}$ 
4     if ( $\sigma \in \mathbb{F}_{\text{far}}$ ) {
5        $\vec{\zeta} := \vec{\zeta} + \mathbf{A}_\sigma \cdot (\mathbf{B}_\sigma \vec{\mu})$ ; //  $\rightarrow$  (2.2.1.6)
6     }
7     else {  $\vec{\zeta} := \vec{\zeta} + \mathbf{H} \cdot \vec{\mu}$ ; }
8     else foreach ( $\tau = (v,w) \in \text{sons}(\sigma)$ ) {
9       hmv( $\mathbf{H}|_\tau, \vec{\zeta}|_v, \mu|_w$ );
10    }
11  }
```

◁ The argument \mathbf{H} should be a **hierarchical matrix** in recursive block-tree-based format. Then this argument need only pass a node of the block tree, cf. Line 2.

Line 8: see (2.4.1.11).

The cost of `hmv()` remains the same as the estimate (2.3.5.7) found in § 2.3.5.6:

$$\text{cost}(\text{hmv}) = O((n + m) \log(n + m))$$

for $n, m \rightarrow \infty$, where the constants will depend on the **sparsity measure** $\text{spm}(\mathbb{F})$, see Def. 2.3.4.27.

In Code 2.4.1.19 the argument $\vec{\zeta}$ is passed through a reference, which means that this argument can be modified during the execution of the function. In C++ this can be done by passing function parameters as reference: `f(X &x)`. ┘

Our goal in this section is to find an algorithm that can be used to **approximately** solve linear systems of equations whose coefficient matrix is provided in hierarchical matrix (\mathcal{H} -matrix) format, refer to Def. 2.4.1.2 and Ass. 2.4.1.7. This will turn out to be a highly complex algorithm with many components. Those are provided in the following sections.

Remark 2.4.1.20 (\mathcal{H} -matrix \times dense matrix) The multiplication of a \mathcal{H} -matrix $\mathbf{H} \in \mathbb{R}^{n,m}$ with a dense matrix $\mathbf{D} \in \mathbb{R}^{m,k}$ can be carried out by feeding all the columns of \mathbf{D} to `hmv(\mathbf{H}, \dots)`. ┘

EXAMPLE 2.4.1.21 (Preview: multiplication of hierarchical matrices)

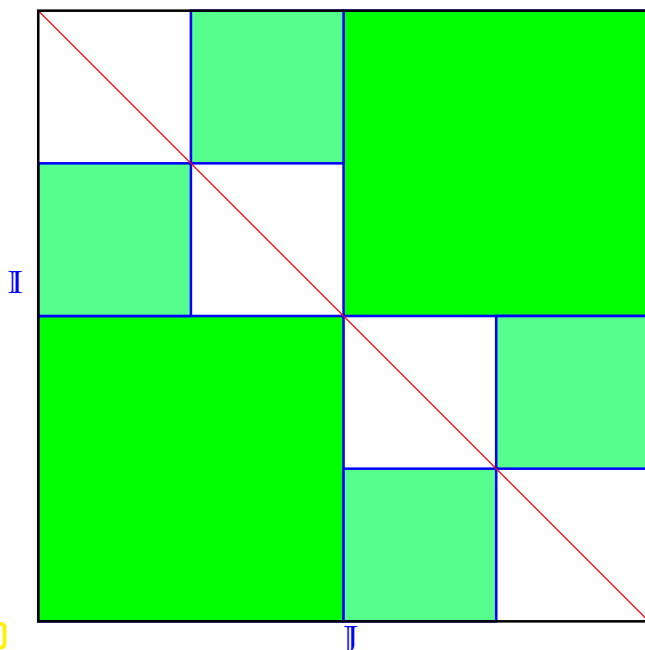


Fig. 114

The setting is rather special for the sake of lucidity:

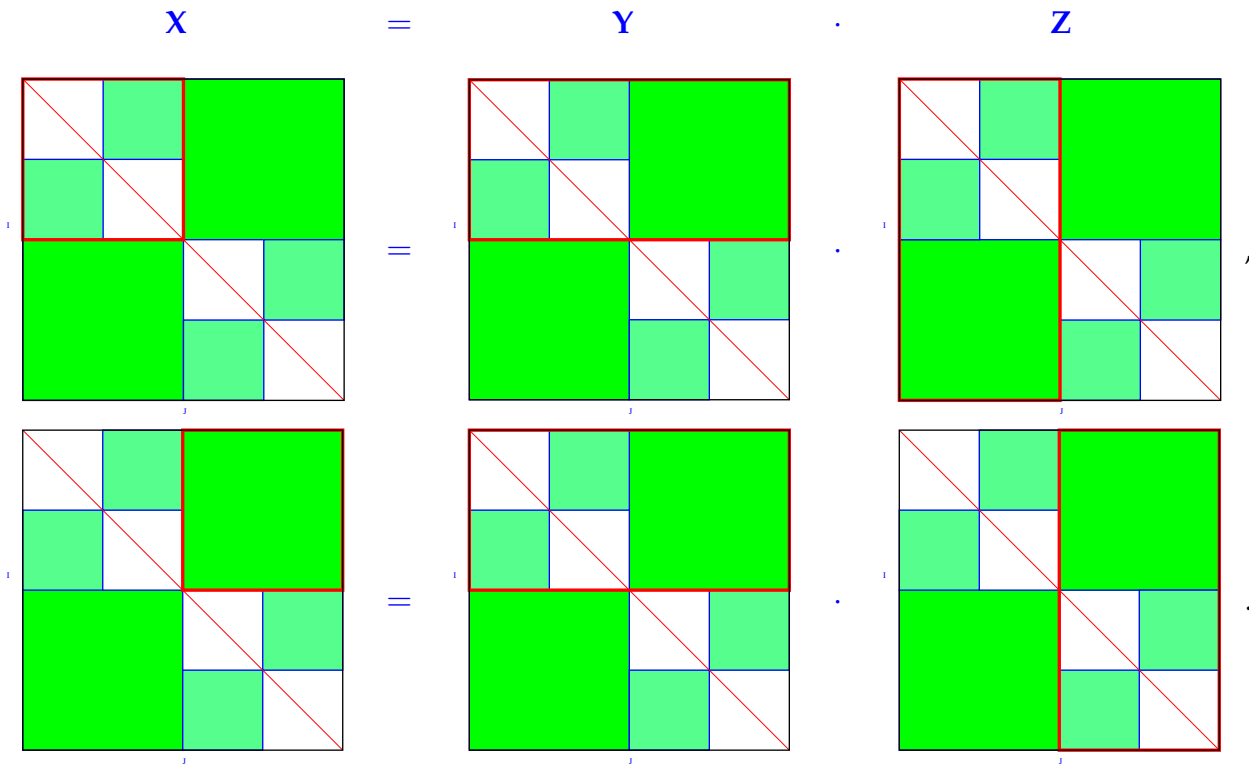
We consider two square hierarchical matrices $\mathbf{Y}, \mathbf{Z} \in \mathbb{R}^{n,n}$ with local rank q based on the same binary balanced row and column cluster tree \mathcal{T}_I , $I := \{1, \dots, n\}$. We used

$$\text{adm}(v, w) = \text{true} \Leftrightarrow I(v) \cap I(w) = \emptyset.$$

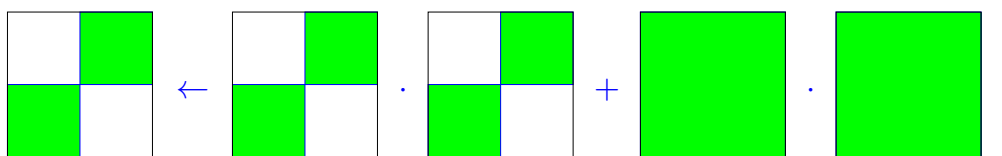
◁ block structure of simple hierarchical matrices in this example (The rank- q far-field blocks are colored green.)

Goal: Approximate the product $\mathbf{Y} \cdot \mathbf{Z}$ by an $n \times n$ -hierarchical matrix based on row/column tree \mathcal{T}_I and the same admissibility condition, that is, the same block structure, the same $\mathbb{F}_{\text{near}}, \mathbb{F}_{\text{far}}$.

The following situations are encountered when forming the matrix product:



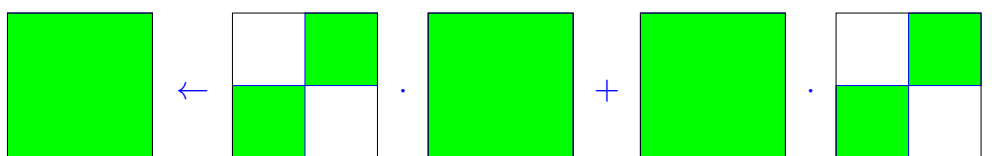
Top row of (2.4.1.21): To compute the upper left block of the matrix product we face



To accomplish this we have to

- compute the product of two smaller hierarchical matrices ➤ recursion,
- add a rank- q matrix, namely the product of two rank- q blocks, to a hierarchical matrix.

Bottom row of (2.4.1.21): The evaluation of the upper right block boils down to



To accomplish this we have to

- compute the product of a hierarchical matrix with a rank- q matrix in any order,
- incorporate the sum of two rank- q matrices into a rank- q block.

2.4.2 Low-Rank Matrices: Algorithms

Now we repeat fundamental concepts and algorithms from numerical linear algebra.

§2.4.2.1 ((Economical/thin) Singular value decomposition → [NumCSE Section 3.4])

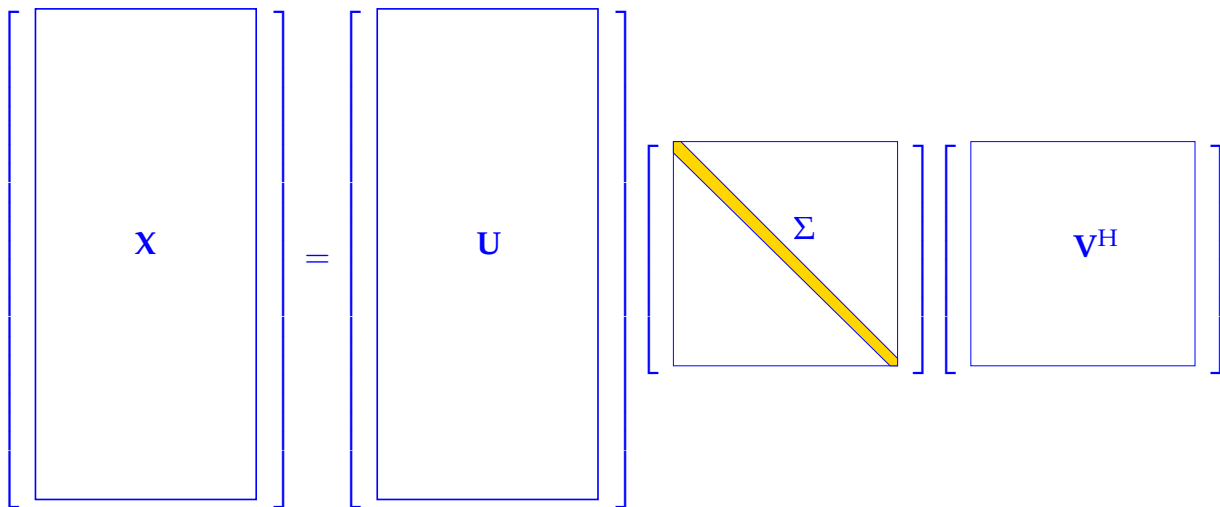
Theorem 2.4.2.2. Singular Value Decomposition (SVD)

For any $\mathbf{X} \in \mathbb{R}^{n,m}$, $n, m \in \mathbb{N}$, $r := \min\{n, m\}$ there are matrices $\mathbf{U} \in \mathbb{R}^{n,r}$ and $\mathbf{V} \in \mathbb{R}^{m,r}$ with *orthonormal columns* and a *diagonal* matrix $\mathbf{\Sigma} \in \mathbb{R}^{r,r}$ with non-negative entries such

$$\mathbf{X} = \mathbf{U} \cdot \mathbf{\Sigma} \cdot \mathbf{V}^\top. \quad (2.4.2.3)$$

Recall that a matrix $\mathbf{Y} \in \mathbb{R}^{k,m}$, $k, m \in \mathbb{N}$, has orthonormal columns, if $\mathbf{Y}^\top \mathbf{Y} = \mathbf{I}_m$, $\mathbf{I}_m \triangleq m \times m$ -identity matrix.

The matrix factorization (2.4.2.3) is called the **economical/thin singular value decomposition (SVD)** of \mathbf{X} . For $k > l$ it can be visualized as follows.



The diagonal entries of $\mathbf{\Sigma} \in \mathbb{R}^{r,r}$ are called the (non-zero) **singular values** of \mathbf{X} , denoted by $\sigma_1, \sigma_2, \dots, \sigma_r$ and assumed to be ordered

$$0 \leq \sigma_r \leq \sigma_{r-1} \leq \dots \leq \sigma_1.$$

The computation of the singular value decomposition of a matrix relies on a sophisticated algorithm [GV13, Sect. 8.6]. This algorithm is perfectly stable and returns the results with relative error of the same size as the machine precision `eps`. The effort for computing the SVD of a densely populated matrix is substantial:

$$\text{cost}(\text{economical/thin SVD of } \mathbf{X} \in \mathbb{R}^{n,m}) = O(\min\{n, m\}nm) \text{ for } n, m \rightarrow \infty. \quad (2.4.2.4)$$

The following C++ function computes the factors of the singular value decomposition of a matrix in EIGEN, see also [NumCSE Code 3.4.2.1]. Note that EIGEN has to be instructed to compute the economical/thin version instead of the full SVD with square orthogonal factors. Of course, one usually does not build the matrix $\mathbf{\Sigma}$ as a dense matrix.

C++ code 2.4.2.5: Computing the economical/thin SVD in EIGEN

```
1 std::tuple<MatrixXd, MatrixXd, MatrixXd> svd_eco(const MatrixXd& X) {
2   Eigen::JacobiSVD<MatrixXd> svd(X, Eigen::ComputeThinU | Eigen::ComputeThinV);
3   MatrixXd U = svd.matrixU(); // get unitary (square) matrix U
4   MatrixXd V = svd.matrixV(); // get unitary (square) matrix V
5   VectorXd sv = svd.singularValues(); // get singular values as vector
6   MatrixXd Sigma = sv.asDiagonal(); // build diagonal matrix Σ
7   return std::tuple<MatrixXd, MatrixXd, MatrixXd>(U, Sigma, V);
```

8 }

The SVD owes its key role in numerical algorithms to the fact that it paves the way for computing the *rank- q best approximation* of a given matrix.

Theorem 2.4.2.6. Best low rank approximation → [NumCSE Thm. 3.4.4.19]

Let $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ be the SVD of $\mathbf{X} \in \mathbb{R}^{n,m}$ (→ Thm. 2.4.2.2). For $1 \leq q \leq \text{rank}(\mathbf{X})$ set $\mathbf{U}_q := (\mathbf{U})_{:,q} \in \mathbb{R}^{n,q}$, $\mathbf{V}_q := (\mathbf{V})_{:,q} \in \mathbb{R}^{m,q}$, $\mathbf{\Sigma}_q := \text{diag}(\sigma_1, \dots, \sigma_q) \in \mathbb{R}^{q,q}$. Then, for $\|\cdot\| = \|\cdot\|_F$ and $\|\cdot\| = \|\cdot\|_2$, holds true

$$\|\mathbf{X} - \mathbf{U}_q \mathbf{\Sigma}_q \mathbf{V}_q^\top\| \leq \|\mathbf{X} - \mathbf{F}\| \quad \forall \mathbf{F} \in \mathbb{R}^{n,m}, \text{rank}(\mathbf{F}) = q,$$

that is, the *truncated SVD* realizes the *rank- q best approximation* of \mathbf{X} with respect to both the Frobenius norm (2.2.0.8) and the Euclidean matrix norm.

Norms of approximation error can be computed easily: Writing $\mathbf{X}_q := \mathbf{U}_q \mathbf{\Sigma}_q \mathbf{V}_q^\top$ we have $\text{rank}(\mathbf{X}_q) \leq q$ and

$$\|\mathbf{X} - \mathbf{X}_q\| = \|\mathbf{\Sigma} - \mathbf{\Sigma}_q\| = \begin{cases} \sigma_{q+1} & \text{for } \|\cdot\| = \|\cdot\|_2, \\ \sqrt{\sigma_{q+1}^2 + \dots + \sigma_r^2} & \text{for } \|\cdot\| = \|\cdot\|_F. \end{cases} \quad (2.4.2.7)$$

This is a straightforward consequence of the fact that both norms satisfy

$$\|\mathbf{U}\mathbf{X}\|_F = \|\mathbf{X}\|_F, \quad \|\mathbf{U}\mathbf{X}\|_2 = \|\mathbf{X}\|_2 \quad \forall \mathbf{X} \in \mathbb{R}^{k,l}, \mathbf{U} \in \mathbb{R}^{k,k}, \mathbf{U}^\top \mathbf{U} = \mathbf{I}_k. \quad (2.4.2.8)$$

§2.4.2.9 (QR-decomposition → [NumCSE Section 3.3.3]) Appealing to the Gram-Schmidt orthonormalization algorithm we derived the following theorem about a special matrix factorization:

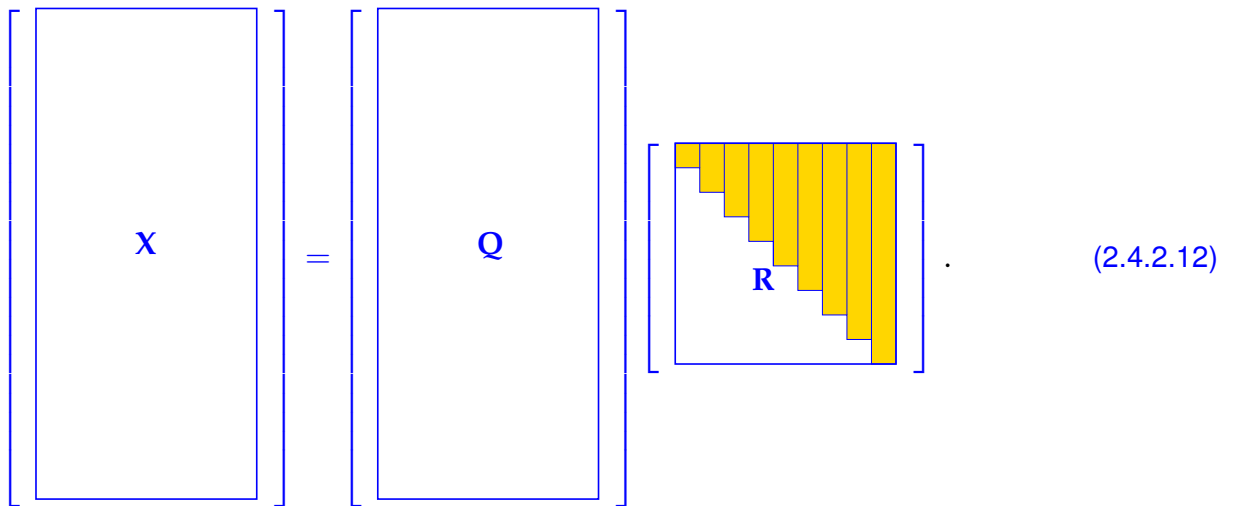
Theorem 2.4.2.10. Economical QR-decomposition

For any matrix $\mathbf{X} \in \mathbb{R}^{k,l}$, $k, l \in \mathbb{N}$, $k \geq l$, with $\text{rank}(\mathbf{X}) = l$ there exists a unique matrix $\mathbf{Q} \in \mathbb{R}^{k,l}$ with orthonormal columns $\mathbf{Q}^\top \mathbf{Q} = \mathbf{I}_l$ and a unique *upper triangular* matrix $\mathbf{R} \in \mathbb{R}^{l,l}$ with $(\mathbf{R})_{i,i} > 0$, $1 \leq i \leq l$, such that

$$\mathbf{X} = \mathbf{Q} \cdot \mathbf{R}. \quad (2.4.2.11)$$

The factorization (2.4.2.11) of \mathbf{X} is called **QR-decomposition**. It can be visualized in the following way for $k \geq l$:

$$\mathbf{X} = \mathbf{Q} \cdot \mathbf{R}, \quad \mathbf{Q} \in \mathbb{K}^{k,l}, \quad \mathbf{R} \in \mathbb{K}^{l,l} \text{ upper triangular,}$$



$$\begin{bmatrix} \\ \\ \\ \end{bmatrix} = \begin{bmatrix} \\ \\ \\ \end{bmatrix} \begin{bmatrix} & & & \\ & & & \\ & & & \\ & & & \end{bmatrix} . \quad (2.4.2.12)$$

A stable algorithm for computing the QR-decomposition of a dense matrix relies on successive Householder transformations, see [NumCSE § 3.3.3.11]. The asymptotic effort required for finding a QR-decomposition are the same as for computing the SVD:

$$\text{cost}(\text{economical QR-decomposition of } \mathbf{X} \in \mathbb{R}^{k,l}) = O(\min\{k,l\}kl) \quad \text{for } k, l \rightarrow \infty . \quad (2.4.2.13)$$

┘

§2.4.2.14 (Low-rank approximation of low-rank matrices) Assume that the matrix $\mathbf{X} \in \mathbb{R}^{k,l}$ with $\text{rank}(\mathbf{X}) = p \leq \min\{k, l\}$ is given in factorized form

$$\mathbf{X} = \mathbf{A} \cdot \mathbf{B}^T, \quad \mathbf{A} \in \mathbb{R}^{k,p}, \quad \mathbf{B} \in \mathbb{R}^{l,p},$$

according to Lemma 2.2.1.3. In order to obtain further compression we want to determine the rank- q best approximation \mathbf{Y} of \mathbf{X} for some $q < p$

$$\mathbf{Y} \in \mathbb{R}^{k,l}, \text{rank}(\mathbf{Y}) = q: \quad \|\mathbf{X} - \mathbf{Y}\|_F \leq \|\mathbf{X} - \mathbf{F}\|_F \quad \forall \mathbf{F} \in \mathbb{R}^{k,l}, \text{rank}(\mathbf{F}) = q .$$

Of course, we want to find the low-rank factors $\tilde{\mathbf{A}} \in \mathbb{R}^{k,q}, \tilde{\mathbf{B}} \in \mathbb{R}^{l,q}$ of \mathbf{Y} such that $\mathbf{Y} = \tilde{\mathbf{A}} \cdot \tilde{\mathbf{B}}^T$.

We start with an (economical/thin) **QR-decomposition** of \mathbf{A} according to Thm. 2.4.2.10:

$$\mathbf{A} = \mathbf{QR} \quad , \quad \mathbf{Q} \in \mathbb{R}^{k,p}, \quad \mathbf{Q}^T \mathbf{Q} = \mathbf{I}_p \quad , \quad \mathbf{R} \in \mathbb{R}^{p,p} \quad \text{upper triangular} . \quad (2.4.2.15)$$

Then we compute the (economical/thin) **SVD** of $\mathbf{RB}^T \in \mathbb{R}^{p,l}$:

$$\mathbf{RB}^T = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T, \quad \mathbf{U} \in \mathbb{R}^{p,p}, \quad \mathbf{\Sigma} \in \mathbb{R}^{p,p}, \quad \mathbf{V} \in \mathbb{R}^{p,l},$$

where \mathbf{U} and \mathbf{V} have orthonormal columns and $\mathbf{\Sigma}$ is a diagonal matrix with non-negative entries. Combining the two factorizations yields

$$\mathbf{X} = \mathbf{AB}^T = \mathbf{QRB}^T = \mathbf{QU}\mathbf{\Sigma}\mathbf{V}^T = \underbrace{\tilde{\mathbf{U}}\mathbf{\Sigma}\mathbf{V}^T}_{\text{SVD of } \mathbf{X}!}, \quad \tilde{\mathbf{U}} := \mathbf{QU}, \quad \tilde{\mathbf{U}}^T \tilde{\mathbf{U}} = \mathbf{U}^T \mathbf{Q}^T \mathbf{QU} = \mathbf{I}_p . \quad (2.4.2.16)$$

Thus, invoking Thm. 2.4.2.6 and adopting its notations, we have found

$$\mathbf{Y} = \underbrace{\mathbf{U}_q}_{=: \tilde{\mathbf{A}}} \underbrace{\mathbf{\Sigma}_q \mathbf{V}_q^T}_{=: \tilde{\mathbf{B}}^T}, \quad \mathbf{U}_q := (\tilde{\mathbf{U}})_{:,q}, \quad \mathbf{\Sigma}_q := (\mathbf{\Sigma})_{1:q,1:q}, \quad \mathbf{V}_q := (\mathbf{V})_{:,q} . \quad (2.4.2.17)$$

Pseudocode 2.4.2.18: Low-rank “recompression”

```

1 [Matrix, Matrix] ← low_rank_recompress (
2 Matrix A, Matrix B, int q) {
3   k := A.rows(); l := B.rows();
4   if (q > min(k, l)) { return (A, B); }
5   [Q, R] = qr(A);
6   [U, Σ, V] = svd(R · BT);
7   Ũ := Q · U; // see (2.4.2.16)
8   Ã := (U):,q; // first q columns of U
9   B̃ := VqΣq; // see (2.4.2.17)
10  return (Ã, B̃);
11 }

```

The asymptotic computational effort of `low_rank_recompress` is determined by the calls to `qr()` and `svd()`.

If $\mathbf{A} \in \mathbb{R}^{k,p}$, $\mathbf{B} \in \mathbb{R}^{l,p}$, then from (2.4.2.4) and (2.4.2.13) we conclude

$$\text{cost}(\text{low_rank_recompress}) = O(p^2(k+l)) \text{ for } k, l \rightarrow \infty. \quad (2.4.2.19)$$

Remark 2.4.2.20 (Adaptive low-rank recompression) According to (2.4.2.7) the discarded singular values provide information about the error committed during low-rank compression of a matrix. Thus, writing $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_p$ for the singular values of \mathbf{X} available as diagonal entries of Σ in (2.4.2.16), we may set ($\sigma_{p+1} := 0$)

$$q \in \{1, \dots, p\}: \sigma_{q+1} \leq \text{rtol} \cdot \sigma_1, \quad (2.4.2.21)$$

for some prescribed relative tolerance $\text{rtol} > 0$. This gives control of the recompression error. \lrcorner

§2.4.2.22 (Recompression of sums of low-rank matrices) We are given two rank- q matrices

$$\mathbf{X}_i = \mathbf{A}_i \mathbf{B}_i^T, \quad \mathbf{A}_i \in \mathbb{R}^{k,q}, \quad \mathbf{B}_i \in \mathbb{R}^{l,q}, \quad i = 1, 2,$$

and want to compute the rank- q best approximation of $\mathbf{X}_1 + \mathbf{X}_2$. This can be done with a single call to `low_rank_recompress` from Code 2.4.2.18, because, thanks to

$$\mathbf{X}_1 + \mathbf{X}_2 = [\mathbf{A}_1 \quad \mathbf{A}_2] \cdot \begin{bmatrix} \mathbf{B}_1^T \\ \mathbf{B}_2^T \end{bmatrix}, \quad (2.4.2.23)$$

we immediately have a rank- $2q$ factorization of $\mathbf{X}_1 + \mathbf{X}_2$ at our disposal.

Pseudocode 2.4.2.24: Approximation of sum of low-rank matrices

```

1 [Matrix, Matrix] ← low_rank_sum(Matrix A1, Matrix B1,
2 Matrix A2, Matrix B2) {
3   q := A.cols(); // target rank for compression
4   A* := [A1 A2]; B* := [B1 B2];
5   return low_rank_recompress(A*, B*, q);
6 }

```

The asymptotic cost is $O(q^2(l+k))$ for $k, l \rightarrow \infty$. \lrcorner

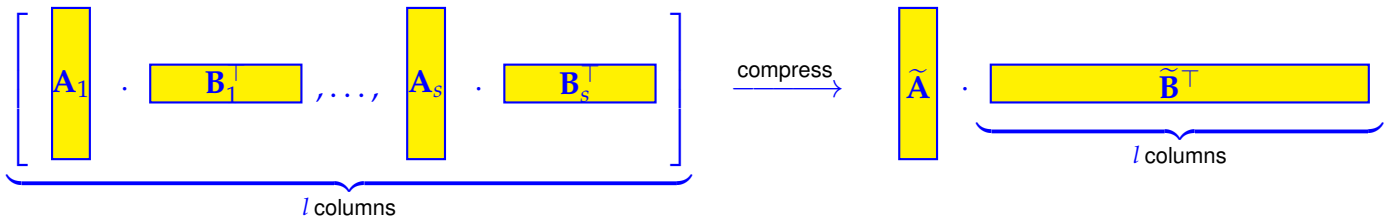
§2.4.2.25 (Compressing stacked low-rank matrices) We arrange $s \in \mathbb{N}$ rank- q matrices

$$\mathbf{X}_i = \mathbf{A}_i \mathbf{B}_i^T \in \mathbb{R}^{k,l_i}, \quad \mathbf{A}_i \in \mathbb{R}^{k,q}, \quad \mathbf{B}_i \in \mathbb{R}^{l_i,q}, \quad l_i \in \mathbb{N}, \quad i = 1, \dots, s,$$

next to each other, an operation also called **horizontal concatenation**,

$$\mathbf{Z} := [\mathbf{X}_1 \quad \mathbf{X}_2 \quad \dots \quad \mathbf{X}_s] \in \mathbb{R}^{k,l}, \quad l := l_1 + \dots + l_s.$$

and aim to determine a rank- q best approximation $\mathbf{Y} = \tilde{\mathbf{A}} \cdot \tilde{\mathbf{B}}^\top$, $\tilde{\mathbf{A}} \in \mathbb{R}^{k,q}$, $\tilde{\mathbf{B}} \in \mathbb{R}^{l,q}$, of \mathbf{Z} in factorized form:



Of course, as in § 2.4.2.14 we use Thm. 2.4.2.6 but, again, we cannot afford to compute the SVD of \mathbf{Z} directly. As in § 2.4.2.14 it can be obtained efficiently via **QR-decompositions** (\rightarrow Thm. 2.4.2.10) of low-rank factors:

$$\mathbf{B}_i = \mathbf{Q}_i \mathbf{R}_i, \quad \mathbf{Q}_i^\top \mathbf{Q}_i = \mathbf{I}_q, \quad \mathbf{R}_i \text{ upper triangular.}$$

This yields as factorization of \mathbf{Z}

$$\mathbf{Z} = \underbrace{[\mathbf{A}_1 \mathbf{R}_1^\top \quad \dots \quad \mathbf{A}_s \mathbf{R}_s^\top]}_{=: \hat{\mathbf{Z}} \in \mathbb{R}^{k,sq}} \cdot \begin{bmatrix} \mathbf{Q}_1^\top & & & \\ & \mathbf{Q}_2^\top & & \\ & & \ddots & \\ & & & \mathbf{Q}_s^\top \end{bmatrix}.$$

Obviously, the transpose of the second factor features orthogonal columns. Then compute the (economical/thin) **SVD** of the first factor according to Thm. 2.4.2.2

$$\hat{\mathbf{Z}} = \mathbf{U} \cdot \boldsymbol{\Sigma} \cdot \mathbf{V}^\top, \quad \mathbf{U} \in \mathbb{R}^{k,sq}, \quad \boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_{qs} \end{bmatrix} \in \mathbb{R}^{qs,qs}, \quad \mathbf{V} \in \mathbb{R}^{l,qs},$$

$\mathbf{U}^\top \mathbf{U} = \mathbf{V}^\top \mathbf{V} = \mathbf{I}_{qs}$, gives the SVD of \mathbf{Z} :

$$\mathbf{Z} = \mathbf{U} \cdot \boldsymbol{\Sigma} \cdot \underbrace{\left(\mathbf{V}^\top \cdot \begin{bmatrix} \mathbf{Q}_1^\top & & & \\ & \mathbf{Q}_2^\top & & \\ & & \ddots & \\ & & & \mathbf{Q}_s^\top \end{bmatrix} \right)}_{=: \tilde{\mathbf{V}}^\top \in \mathbb{R}^{qs,ls}}. \tag{2.4.2.26}$$

Thus, the low-rank factors of the rank= q best approximation are

$$\tilde{\mathbf{A}} := \mathbf{U}_q \cdot \boldsymbol{\Sigma}_q, \quad \mathbf{U}_q := (\mathbf{U})_{:,1,\dots,q}, \quad \boldsymbol{\Sigma}_q := \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_q \end{bmatrix} \in \mathbb{R}^{q,q}, \tag{2.4.2.27a}$$

$$\tilde{\mathbf{B}} := (\tilde{\mathbf{V}})_{:,1,\dots,q} \in \mathbb{R}^{l,q}. \tag{2.4.2.27b}$$

The total asymptotic computational effort is dominated by the cost of computing SVD and QR-decomposition. For $sq \leq k$ it amounts to

$$\text{cost} = O((sq)^2(k+l)) \quad \text{for } k, l \rightarrow \infty. \tag{2.4.2.28}$$

A similar algorithm can be applied for the low-rank compression of a matrix arising from stacking low-rank matrices on top of each other (vertical concatenation). \lrcorner

2.4.3 \mathcal{H} -Addition of Hierarchical Matrices

Armed with the algorithm of Code 2.4.2.24 we can efficiently *add and recompress* two hierarchical matrices $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{n,m}$ provided that

- ◆ they are based on the *same* row and column *cluster trees*,
- ◆ their far-field/near-field block partitions coincide (which will follow, if the *same admissibility condition* is used for their construction).

This means that both \mathcal{H} -matrices are based on the same block trees; their partitionings into near-field and far-field blocks coincide.

Due to recompression \mathcal{H} -addition differs from the exact addition of the matrices. Therefore we designate it with a special symbol.

📎 Notation: We write \oplus for the addition with recompression of hierarchical matrices.

The following pseudocode performs the operation $\mathbf{H} \leftarrow \mathbf{H} \oplus \mathbf{H}'$ for two hierarchical matrices of the same local rank q , whose far-field blocks are provided in factorized form $\mathbf{H}|_{\sigma} = \mathbf{A}_{\sigma} \mathbf{B}_{\sigma}^{\top}$ and $\mathbf{H}'|_{\sigma} = \mathbf{A}'_{\sigma} (\mathbf{B}'_{\sigma})^{\top}$, $\sigma \in \mathbb{F}_{\text{far}}$, according to (2.4.1.6). Note that the block trees of \mathbf{H} and \mathbf{H}' agree.

Pseudocode 2.4.3.1: In-situ summation and recompression of hierarchical matrices

```

1 void hmat_add(ref  $\mathcal{H}$ -matrix  $\mathbf{H} \in \mathbb{R}^{n,m}$ , const  $\mathcal{H}$ -matrix  $\mathbf{H}' \in \mathbb{R}^{n,m}$ ) {
2    $\sigma :=$  root of block tree for  $\mathbf{H}$ ;
3   if (sons( $\sigma$ ) =  $\emptyset$ ) { // leaf of block tree
4     if ( $\sigma \in \mathbb{F}_{\text{far}}$ ) { // sum and truncate
5       [ $\mathbf{A}_{\sigma}, \mathbf{B}_{\sigma}$ ] := low_rank_sum( $\mathbf{A}_{\sigma}, \mathbf{B}_{\sigma}, \mathbf{A}'_{\sigma}, \mathbf{B}'_{\sigma}$ );
6     }
7     else { // dense near-field block
8        $\mathbf{H} := \mathbf{H} + \mathbf{H}'$ ;
9     }
10  }
11  else { // recursion
12    foreach ( $\tau = (v, w) \in$  sons( $\sigma$ )) {
13      hmat_add( $\mathbf{H}|_{\tau}, \mathbf{H}'|_{\tau}$ );
14    }
15  }
16 }
```

§2.4.3.2 (Low-Rank Modification of a Hierarchical Matrix) Let a row tree $\mathcal{T}_{\mathbf{I}}$ for $\mathbf{I} := \{1, \dots, n\}$ and a column tree $\mathcal{T}_{\mathbf{J}} := \{1, \dots, m\}$, $n, m \in \mathbb{N}$, be given (and some admissibility condition according to Def. 2.3.3.2). Since the rank of a sub-matrix is at least as big as the rank of the matrix itself, it is clear that any rank- q matrix $\mathbf{Y} = \mathbf{U}\mathbf{V}^{\top} \in \mathbb{R}^{n,m}$, $\mathbf{U} \in \mathbb{R}^{n,q}$, $\mathbf{V} \in \mathbb{R}^{m,q}$, can be treated as a hierarchical matrix based on $\mathcal{T}_{\mathbf{I}}$ and $\mathcal{T}_{\mathbf{J}}$.

Pseudocode 2.4.3.3: Recursive low-rank update of \mathcal{H} -matrix: $\mathbf{H} \leftarrow \mathbf{H} \oplus \mathbf{UV}^\top$

```

1 void low_rank_update(ref  $\mathcal{H}$ -matrix  $\mathbf{H} \in \mathbb{R}^{n,m}$ ,
2 Matrix  $\mathbf{U} \in \mathbb{R}^{n,q}$ , Matrix  $\mathbf{V} \in \mathbb{R}^{m,q}$ ) {
3    $\sigma$  := root of block tree for  $\mathbf{H}$ ;
4   if (sons( $\sigma$ ) =  $\emptyset$ ) { // leaf of block tree
5     if ( $\sigma \in \mathbb{F}_{\text{far}}$ ) {
6       [ $\mathbf{A}_\sigma, \mathbf{B}_\sigma$ ] := low_rank_sum( $\mathbf{A}_\sigma, \mathbf{B}_\sigma, \mathbf{U}, \mathbf{V}$ );
7     }
8     else { // near-field block
9        $\mathbf{H} := \mathbf{H} + \mathbf{U} \cdot \mathbf{V}$ ;
10    }
11   else {
12     foreach ( $\tau = (v, w) \in \text{sons}(\sigma)$ ) {
13       low_rank_update( $\mathbf{H}|_\tau, (\mathbf{U})_{\mathcal{I}(v),:}, (\mathbf{V})_{\mathcal{I}(w),:}$ );
14     }
15   }
16 }

```

Given a hierarchical matrix $\mathbf{H} \in \mathbb{R}^{n,m}$ with local rank q stored in a block tree compatible format Code 2.4.3.3 computes recursively $\mathbf{H} \oplus \mathbf{UV}^\top$ for $\mathbf{U} \in \mathbb{R}^{n,q}$ and $\mathbf{V} \in \mathbb{R}^{m,q}$, and stores the result in \mathbf{H} again. Refer to Code 2.4.1.19 for a related algorithm. \lrcorner

2.4.4 \mathcal{H} -Multiplication of Hierarchical Matrices [Bör21, Sect. 5.6]

We are given two hierarchical matrices $\mathbf{Y} \in \mathbb{R}^{n,k}$, $\mathbf{Z} \in \mathbb{R}^{k,m}$, $n, k, m \in \mathbb{N}$ with local ranks q_Y and q_Z , respectively, according to Def. 2.4.1.2. Our goal is to compute an approximation $\mathbf{X} \approx \mathbf{Y} \cdot \mathbf{Z}$, which is itself a hierarchical matrix.

We assume that

- ◆ $\mathbf{Y} \in \mathbb{R}^{n,k}$ is based on the binary cluster trees $\mathcal{T}_{\mathbb{I}}$ of $\mathbb{I} := \{1, \dots, n\}$ and $\mathcal{T}_{\mathbb{K}}$ of $\mathbb{K} := \{1, \dots, k\}$,
- ◆ $\mathbf{Z} \in \mathbb{R}^{k,m}$ is based on the binary cluster trees $\mathcal{T}_{\mathbb{K}}$ of $\mathbb{K} := \{1, \dots, k\}$, and $\mathcal{T}_{\mathbb{J}}$ of $\mathbb{J} := \{1, \dots, m\}$.
- ◆ $\mathbf{X} \in \mathbb{R}^{n,m}$ is based on the binary cluster trees $\mathcal{T}_{\mathbb{I}}$ of $\mathbb{I} := \{1, \dots, n\}$ and $\mathcal{T}_{\mathbb{J}}$ of $\mathbb{J} := \{1, \dots, m\}$.



Note that the column tree of \mathbf{Y} and the row tree of \mathbf{Z} have to agree. Otherwise $\mathcal{T}_{\mathbb{I}}$, $\mathcal{T}_{\mathbb{J}}$, and $\mathcal{T}_{\mathbb{K}}$ may be unrelated.

notation: We write $\mathcal{B}_X := \mathcal{B}_{\mathbb{I} \times \mathbb{J}} \subset \mathcal{T}_{\mathbb{I}} \times \mathcal{T}_{\mathbb{J}}$, $\mathcal{B}_Y := \mathcal{B}_{\mathbb{I} \times \mathbb{K}} \subset \mathcal{T}_{\mathbb{I}} \times \mathcal{T}_{\mathbb{K}}$, $\mathcal{B}_Z := \mathcal{B}_{\mathbb{K} \times \mathbb{J}} \subset \mathcal{T}_{\mathbb{K}} \times \mathcal{T}_{\mathbb{J}}$, for the block trees of \mathbf{X} , \mathbf{Y} , \mathbf{Z} .

The sets of matrix blocks of \mathbf{X} , \mathbf{Y} and \mathbf{Z} , which correspond to the leaves of \mathcal{B}_X , \mathcal{B}_Y , and \mathcal{B}_Z , will be denoted by \mathbb{F}^* , $*$ = X, Y, Z . Superscripts X, Y, Z will also tag the corresponding far-field and near field blocks: $\mathbb{F}_{\text{far}}^*$, $\mathbb{F}_{\text{near}}^*$, $*$ = X, Y, Z . We take for granted that far-field matrix block are available in factorized form (2.4.1.6):

$$\begin{aligned}
\forall \sigma = (v, w) \in \mathbb{F}_{\text{far}}^X: & \exists \mathbf{A}_\sigma^X \in \mathbb{R}^{\#\mathcal{I}(v), q}, \quad \mathbf{B}_\sigma^X \in \mathbb{R}^{\#\mathcal{I}(w), q}: \quad \mathbf{X}|_\sigma = \mathbf{A}_\sigma^X \cdot (\mathbf{B}_\sigma^X)^\top, \\
\forall \tau = (v, u) \in \mathbb{F}_{\text{far}}^Y: & \exists \mathbf{A}_\tau^Y \in \mathbb{R}^{\#\mathcal{I}(v), q}, \quad \mathbf{B}_\tau^Y \in \mathbb{R}^{\#\mathcal{I}(u), q}: \quad \mathbf{Y}|_\tau = \mathbf{A}_\tau^Y \cdot (\mathbf{B}_\tau^Y)^\top, \\
\forall \kappa = (u, w) \in \mathbb{F}_{\text{far}}^Z: & \exists \mathbf{A}_\kappa^Z \in \mathbb{R}^{\#\mathcal{I}(u), q}, \quad \mathbf{B}_\kappa^Z \in \mathbb{R}^{\#\mathcal{I}(w), q}: \quad \mathbf{Z}|_\kappa = \mathbf{A}_\kappa^Z \cdot (\mathbf{B}_\kappa^Z)^\top.
\end{aligned} \tag{2.4.4.1}$$

We also point out that the matrix blocks corresponding to near-field cluster pairs are stored as dense matrices, e.g.

$$\kappa := (u, w) \in \mathbb{F}_{\text{near}}^Z \Rightarrow \text{dense matrix } \mathbf{N}_\tau^Z := \mathbf{Z}|_{u \times w} \in \mathbb{R}^{\#\mathcal{I}(u), \#\mathcal{I}(w)} \text{ is directly accessible.}$$

Note that the admissibility condition $\text{adm}_X : \mathcal{T}_\mathbb{I} \times \mathcal{T}_\mathbb{J} \rightarrow \{\text{true}, \text{false}\}$ used for \mathbf{X} need not have anything to do with the admissibility conditions underlying \mathbf{Y} and \mathbf{Z} . For ease of presentation we assume $q := q_X = q_Y = q_Z$ and that every leaf cluster contains $\leq q$ indices:

$$\begin{aligned} v \in \mathcal{T}_\mathbb{I}, \quad \text{sons}(v) = \emptyset &\Rightarrow \#\mathcal{I}(v) \leq q, \\ w \in \mathcal{T}_\mathbb{J}, \quad \text{sons}(w) = \emptyset &\Rightarrow \#\mathcal{I}(w) \leq q, \\ u \in \mathcal{T}_\mathbb{K}, \quad \text{sons}(u) = \emptyset &\Rightarrow \#\mathcal{I}(u) \leq q. \end{aligned} \tag{2.4.4.2}$$

Thus, matrix blocks defined by leaves are small, need not be stored in low-rank factorized form, and will invariably be assigned to the near field:

$$(x, y) \in \mathbb{F}^*, \quad \text{sons}(x) = \emptyset \text{ or } \text{sons}(y) = \emptyset \Rightarrow (x, y) \in \mathbb{F}_{\text{near}}^*. \tag{2.4.4.3}$$

We remind of the constraint that near-field cluster pairs contain at least one leaf

$$\begin{aligned} (v, w) \in \mathbb{F}_{\text{near}}^X &\Rightarrow \text{sons}(v) = \emptyset \text{ or } \text{sons}(w) = \emptyset, \\ (v, u) \in \mathbb{F}_{\text{near}}^Y &\Rightarrow \text{sons}(v) = \emptyset \text{ or } \text{sons}(u) = \emptyset, \\ (u, w) \in \mathbb{F}_{\text{near}}^Z &\Rightarrow \text{sons}(u) = \emptyset \text{ or } \text{sons}(w) = \emptyset. \end{aligned} \tag{2.4.4.4}$$

As a consequence of (2.4.4.2) and (2.4.4.4), near-field matrix blocks will be small in one dimension and, in particular, have rank $\leq q$.

All these constraints are satisfied by partitions generated by the algorithm implemented in `buildBlockTree()` in Code 2.4.1.16 provided that the admissibility condition from (2.3.3.6) is used.

Assumption 2.4.4.5. Structure of result matrix

The matrix product $\mathbf{Y} \cdot \mathbf{Z} \in \mathbb{R}^{n, m}$ allows an *approximate* representation by a hierarchical matrix \mathbf{X} with local rank $q_X \in \mathbb{N}$, based on the cluster trees $\mathcal{T}_\mathbb{I}$ and $\mathcal{T}_\mathbb{J}$, and a given block tree \mathcal{B}_X .

Thus, similar to the case of adding hierarchical matrices we will resort to low-rank truncation while carrying out the operations of matrix multiplication.

§2.4.4.6 (Matrix multiplication: recursive perspective) Inspired by Ex. 2.4.1.21 we aim for a recursive algorithm, which exploits the recursive structure of block-oriented matrix multiplication, remember [NumCSE § 1.3.1.13]. Let us look at the \mathcal{H} -matrices \mathbf{X} , \mathbf{Y} , and \mathbf{Z} from the roots $(v, w) := \sigma := \text{root}(\mathcal{B}_X)$, $(v, u) := \tau := \text{root}(\mathcal{B}_Y)$, $(u, w) := \kappa := \text{root}(\mathcal{B}_Z)$ of their respective block trees.

If neither v nor w nor u is a leaf, their two son clusters induce a 2×2 block partitioning of \mathbf{X} , \mathbf{Y} , and \mathbf{Z} . The root nodes of the block trees have four sons each.

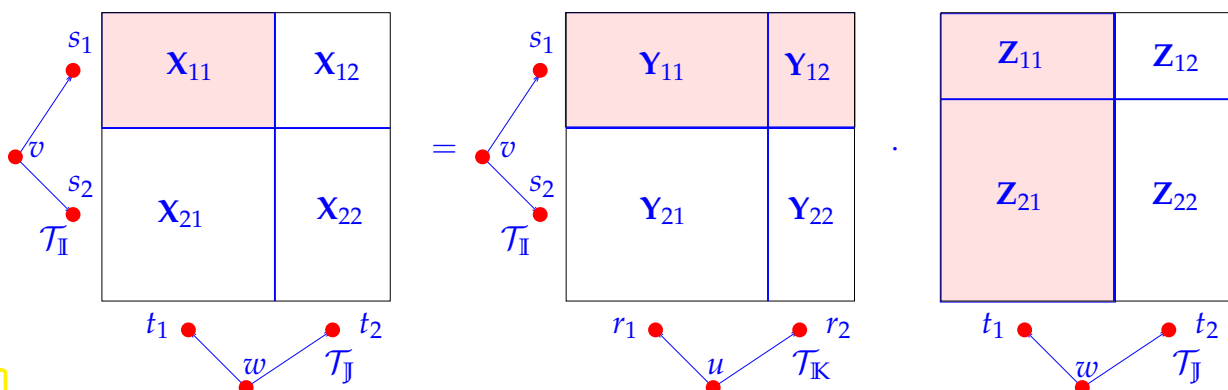
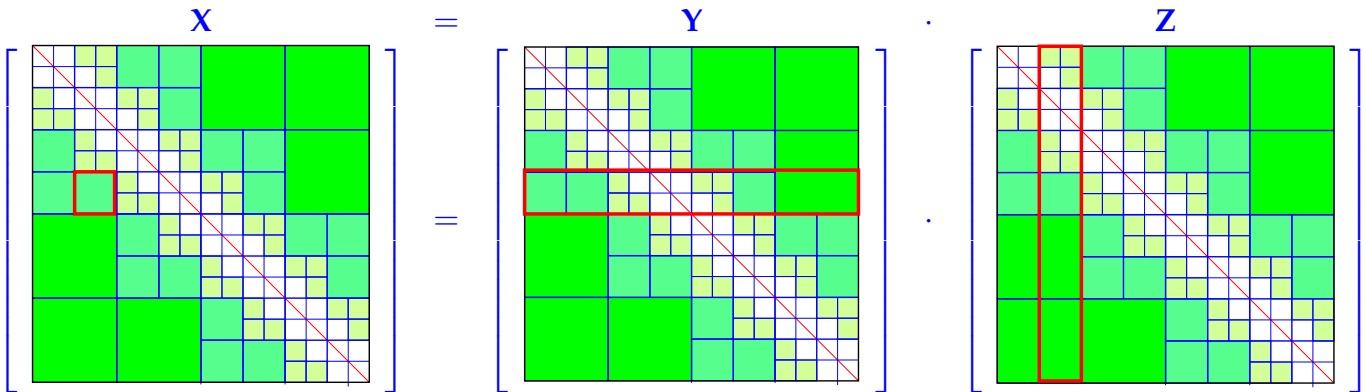


Fig. 115

$$\begin{aligned}
 \text{sons}(v) &= \{s_1, s_2\}, \\
 \text{sons}(w) &= \{t_1, t_2\}, \\
 \text{sons}(u) &= \{r_1, r_2\}
 \end{aligned}
 \quad \blacktriangleright \quad
 \begin{aligned}
 \mathbf{X}|_{s_1 \times t_1} &= \mathbf{Y}|_{s_1 \times r_1} \cdot \mathbf{Z}|_{r_1 \times t_1} + \mathbf{Y}|_{s_1 \times r_2} \cdot \mathbf{Z}|_{r_2 \times t_1}, \\
 \mathbf{X}|_{s_1 \times t_2} &= \mathbf{Y}|_{s_1 \times r_1} \cdot \mathbf{Z}|_{r_1 \times t_2} + \mathbf{Y}|_{s_1 \times r_2} \cdot \mathbf{Z}|_{r_2 \times t_2}, \\
 \mathbf{X}|_{s_2 \times t_1} &= \mathbf{Y}|_{s_2 \times r_1} \cdot \mathbf{Z}|_{r_1 \times t_1} + \mathbf{Y}|_{s_2 \times r_2} \cdot \mathbf{Z}|_{r_2 \times t_1}, \\
 \mathbf{X}|_{s_2 \times t_2} &= \mathbf{Y}|_{s_2 \times r_1} \cdot \mathbf{Z}|_{r_1 \times t_2} + \mathbf{Y}|_{s_2 \times r_2} \cdot \mathbf{Z}|_{r_2 \times t_2}.
 \end{aligned}
 \tag{2.4.4.7}$$

Note that all the sub-matrices occurring in (2.4.4.7) are themselves hierarchical matrices based on subtrees of the cluster trees, see § 2.4.1.17.

§2.4.4.8 (Visualization: Block view of \mathcal{H} -matrix multiplication) In Ex. 2.4.1.21 we could already catch a glimpse of the substantial additional complications compared to addition. The following drawing illustrates that blocks of various kinds and levels contribute to a single block of the matrix product. The setting, which is borrowed from Ex. 2.4.1.3, is rather simple in that all involved \mathcal{H} -matrices are square and are based on the same cluster tree and block tree.



The marked block of \mathbf{X} belongs to $\mathbb{F}_{\text{far}}^{\mathbf{X}}$ and it is formed by summing contributions from all kinds of blocks of \mathbf{Y} and \mathbf{Z} ,

- contributions from blocks that are genuine \mathcal{H} -matrices,
- contributions from blocks $\in \mathbb{F}_{\text{far}}^{\mathbf{Y}}, \mathbb{F}_{\text{far}}^{\mathbf{Z}}$, and
- contributions from *parts of* far-field blocks of \mathbf{Y} and \mathbf{Z} .

§2.4.4.9 (\mathcal{H} -matrix multiplication: Basic block update operation) We consider an arbitrary node $\sigma := (v, w) \in \mathcal{B}_{\mathbf{X}}$, $v \in \mathcal{T}_{\mathbb{I}}, w \in \mathcal{T}_{\mathbb{J}}$, of the block tree $\mathcal{B}_{\mathbf{X}}$ of \mathbf{X} , which need not be a leaf. In particular, σ can be the root of $\mathcal{B}_{\mathbf{X}}$. That node defines a matrix block $\mathbf{X}|_{\sigma} = \mathbf{X}|_{v \times w} = (\mathbf{X})_{\substack{i \in \mathcal{I}(v) \\ j \in \mathcal{I}(w)}}$.

Our goal is to devise a recursive algorithm for *approximating* the following basic update operation:

$$\mathbf{X}|_{\sigma} = \mathbf{X}|_{v \times w} \leftarrow \mathbf{X}|_{v \times w} + \mathbf{Y}|_{v \times u} \cdot \mathbf{Z}|_{u \times w}, \quad (v, u) \in \mathcal{B}_{\mathbf{Y}}, (u, w) \in \mathcal{B}_{\mathbf{Z}}. \tag{2.4.4.10}$$

It is important to keep in mind that $u \in \mathcal{T}_{\mathbb{K}}$ cannot be any cluster, but has to be chosen such that (v, u) and (u, w) are nodes of the block trees $\mathcal{B}_{\mathbf{X}}$ and $\mathcal{B}_{\mathbf{Y}}$, respectively. We implement the algorithm in the function

```
void hmat_mult_add(ref  $\mathcal{H}$ -matrix  $\mathbf{X}$ , const  $\mathcal{H}$ -matrix  $\mathbf{Y}$ , const  $\mathcal{H}$ -matrix  $\mathbf{Z}$ )
```

so that

$$\text{hmat_mult_add}(\mathbf{X}|_{v \times w}, \mathbf{Y}|_{v \times u}, \mathbf{Z}|_{u \times w})$$

approximately realizes the update operation (2.4.4.10). ┘

Remark 2.4.4.11. If $v = \text{root}(\mathcal{T}_I)$, $w = \text{root}(\mathcal{T}_J)$, $u = \text{root}(\mathcal{T}_K)$, and $\mathbf{X} = \mathbf{O}$, then (2.4.4.10) yields the matrix product $\mathbf{Y} \cdot \mathbf{Z}$. ┘

§2.4.4.12 (Update operation (2.4.4.10) for low-rank target matrix) A rather special case is $\sigma \in \mathbb{F}_{\text{far}}^X$, that is, $\mathbf{X}|_{v \times w}$ has to be a rank- q block stored in factorized form (2.4.4.1): $\mathbf{X}|_{v \times w} = \mathbf{A}_\sigma^X (\mathbf{B}_\sigma^X)^\top$. Then the approximate execution of the update operation (2.4.4.10) must boil down to an update of the two low-rank factors $\mathbf{A}_\sigma^X \in \mathbb{R}^{\#\mathcal{I}(v), q}$ and $\mathbf{B}_\sigma^X \in \mathbb{R}^{\#\mathcal{I}(w), q}$. This special case will repeatedly be addressed in the following discussion and is marked as “→ LRT” (low-rank target block).

For this case we provide a special variant of `hmat_mult_add()`:

```

1 void lrt_mult_add(
2 ref matrix A, ref matrix B, const H-matrix Y, const H-matrix Z)
    
```

When invoked as $(\mathbf{A} \in \mathbb{R}^{\#\mathcal{I}(v), q}, \mathbf{B} \in \mathbb{R}^{\#\mathcal{I}(w), q})$

$$\text{lrt_mult_add}(\mathbf{A}, \mathbf{B}, \mathbf{Y}|_{v \times u}, \mathbf{Z}|_{u \times w}),$$

in its first two arguments it is supposed to return the rank- q factors of an approximation of $\mathbf{A}\mathbf{B}^\top + \mathbf{Y}|_{v \times u} \cdot \mathbf{Z}|_{u \times w}$. ┘

§2.4.4.13 (Recursive approximate \mathcal{H} -matrix block multiplication: cases) Depending on the nature of the clusters $v \in \mathcal{T}_I$, $w \in \mathcal{T}_J$, and $u \in \mathcal{T}_K$, and of the blocks $\sigma := (v, w) \in \mathcal{B}_X$, $\tau := (v, u) \in \mathcal{B}_Y$, and $\kappa := (u, w) \in \mathcal{B}_Z$, the evaluation of

$$\mathbf{X}|_\sigma = \mathbf{X}|_{v \times w} \leftarrow \mathbf{X}|_{v \times w} + \mathbf{Y}|_{v \times u} \cdot \mathbf{Z}|_{u \times w}, \quad (v, u) \in \mathcal{B}_Y, (u, w) \in \mathcal{B}_Z. \tag{2.4.4.10}$$

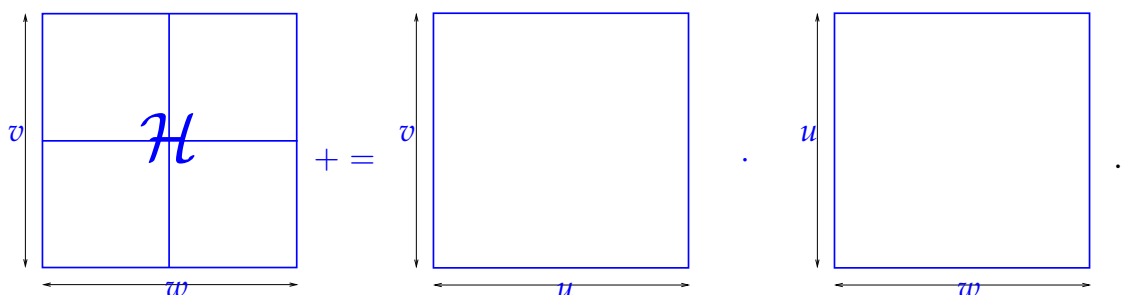
has to distinguish between several cases.

The order of the cases matters; only the first matching case is executed. Recursive calls will be made, if a particular case involves other update operations of the kind (2.4.4.10).

- ❶ Case: $u \in \mathcal{T}_K$ is a leaf of the cluster tree \mathcal{T}_K (containing $\leq q$ indices).

In this case, owing to (2.4.4.3) both nodes $\tau := (v, u) \in \mathcal{B}_Y$ and $\kappa := (u, w) \in \mathcal{B}_Z$ of the block trees are *leaves*, belong to $\mathbb{F}_{\text{near}}^*$, $*$ = Y, Z . and we have immediate access to the small dense near-field blocks

$$\mathbf{N}_\tau^Y = \mathbf{Y}|_{v \times u} \in \mathbb{R}^{\#\mathcal{I}(v), q} \quad \text{and} \quad \mathbf{N}_\kappa^Z = \mathbf{Z}|_{u \times w} \in \mathbb{R}^{q, \#\mathcal{I}(w)}. \tag{2.4.4.14}$$



Hence, the update operation (2.4.4.10) boils down to the *rank- q update* of the generic \mathcal{H} -matrix $\mathbf{X}|_{v \times w}$:

$$\mathbf{X}|_{v \times w} \leftarrow \mathbf{X}|_{v \times w} + \mathbf{N}_\tau^Y \mathbf{N}_\kappa^Z. \quad (2.4.4.15)$$

Here, $\mathbf{X}|_{v \times w}$ can be treated as a *generic \mathcal{H} -matrix* and (2.4.4.15) can be implemented as

$$\mathbf{low_rank_update}(\mathbf{X}|_{v \times w}, \mathbf{N}_\tau^Y, (\mathbf{N}_\kappa^Z)^\top),$$

using the function from Code 2.4.3.3.

→ **LRT**: To update the rank- q factorized format of $\mathbf{X}|_\sigma$ we can invoke

$$[\mathbf{A}_\sigma^X, \mathbf{B}_\sigma^X] = \mathbf{low_rank_sum}(\mathbf{A}_\sigma^X, \mathbf{B}_\sigma^X, \mathbf{N}_\tau^Y, (\mathbf{N}_\kappa^Z)^\top).$$

② Case: $v \in \mathcal{T}_I$ is a leaf of the cluster tree \mathcal{T}_I (holding q indices).

In this case (2.4.4.3) ensures that

$$\sigma := v \times w \in \mathbb{F}_{\text{near}}^X, \quad \mathbf{X}|_{v \times w} = \mathbf{N}_\sigma^X \in \mathbb{R}^{q, \#\mathcal{I}(w)}, \quad (2.4.4.16a)$$

$$\tau := v \times u \in \mathbb{F}_{\text{near}}^Y, \quad \mathbf{Y}|_{v \times u} = \mathbf{N}_\tau^Y \in \mathbb{R}^{q, \#\mathcal{I}(u)}. \quad (2.4.4.16b)$$

As a consequence, the update formula (2.4.4.10) becomes

$$\mathbf{N}_\sigma^X \leftarrow \mathbf{N}_\sigma^X + \widehat{\mathbf{D}}^\top, \quad \mathbf{D} := \mathbf{Z}|_{u \times w}^\top \cdot (\mathbf{N}_\tau^Y)^\top. \quad (2.4.4.17)$$

The dense matrix $\widehat{\mathbf{D}} \in \mathbb{R}^{\#\mathcal{I}(w), q}$ can be computed by multiplying the q columns of $(\mathbf{N}_\tau^Y)^\top$ with the *generic \mathcal{H} -matrix* $\mathbf{Z}|_{u \times w}^\top$ using an algorithm similar to the one implemented in **hmv()** from Code 2.4.1.19.

③ Case: $w \in \mathcal{T}_J$ is a leaf of the cluster tree \mathcal{T}_J (holding $\leq q$ indices).

From (2.4.4.3) we conclude that

$$\sigma := v \times w \in \mathbb{F}_{\text{near}}^X, \quad \mathbf{X}|_{v \times w} = \mathbf{N}_\sigma^X \in \mathbb{R}^{\#\mathcal{I}(v), q}, \quad (2.4.4.18a)$$

$$\kappa := u \times w \in \mathbb{F}_{\text{near}}^Z, \quad \mathbf{Z}|_{u \times w} = \mathbf{N}_\kappa^Z \in \mathbb{R}^{\#\mathcal{I}(u), q}. \quad (2.4.4.18b)$$

Similar to the previous case the update formula (2.4.4.10) becomes

$$\mathbf{N}_\sigma^X \leftarrow \mathbf{N}_\sigma^X + \widehat{\mathbf{D}}, \quad \widehat{\mathbf{D}} := \mathbf{Y}|_{v \times u} \cdot \mathbf{N}_\kappa^Z. \quad (2.4.4.19)$$

As proposed in Rem. 2.4.1.20, the matrix product $\widehat{\mathbf{D}} \in \mathbb{R}^{\#\mathcal{I}(v), q}$ involving the *generic \mathcal{H} -matrix* $\mathbf{Y}|_{v \times u}$ can be computed based on **hmv()** from Code 2.4.1.19.

At this point, none of the clusters $v \in \mathcal{T}_I$, $w \in \mathcal{T}_J$, and $u \in \mathcal{T}_K$ is a leaf of the respective cluster tree and, by virtue of

$$\begin{aligned} (v, w) \in \mathbb{F}_{\text{near}}^X &\Rightarrow \text{sons}(v) = \emptyset \quad \mathbf{or} \quad \text{sons}(w) = \emptyset, \\ (v, u) \in \mathbb{F}_{\text{near}}^Y &\Rightarrow \text{sons}(v) = \emptyset \quad \mathbf{or} \quad \text{sons}(u) = \emptyset, \\ (u, w) \in \mathbb{F}_{\text{near}}^Z &\Rightarrow \text{sons}(u) = \emptyset \quad \mathbf{or} \quad \text{sons}(w) = \emptyset. \end{aligned} \quad (2.4.4.4)$$

none of the blocks $\sigma := (v, w) \in \mathcal{B}_X$, $\tau := (v, u) \in \mathcal{B}_Y$, and $\kappa := (u, w) \in \mathcal{B}_Z$ can belong to $\mathbb{F}_{\text{near}}^*$, $* = X, Y, Z$.

④ Case: $\tau := (v, u) \in \mathbb{F}_{\text{far}}^Y$.

As expressed in (2.4.4.1), $\mathbf{Y}|_{v \times u}$ is given in rank- q factorized form

$$\mathbf{Y}|_{\tau} = \mathbf{Y}|_{v \times u} = \mathbf{A}_{\tau}^Y (\mathbf{B}_{\tau}^Y)^{\top}, \quad \mathbf{A}_{\tau}^Y \in \mathbb{R}^{\#\mathcal{I}(v), q}, \quad \mathbf{B}_{\tau}^Y \in \mathbb{R}^{\#\mathcal{I}(u), q}, \quad (2.4.4.20)$$

which makes it possible to convert the update operation

$$\mathbf{X}|_{\sigma} = \mathbf{X}|_{v \times w} \leftarrow \mathbf{X}|_{v \times w} + \mathbf{Y}|_{v \times u} \cdot \mathbf{Z}|_{u \times w} \quad (2.4.4.10)$$

to

$$\mathbf{X}|_{\sigma} = \mathbf{X}|_{v \times w} \leftarrow \mathbf{X}|_{v \times w} + \mathbf{A}_{\tau}^Y \widehat{\mathbf{V}}^{\top}, \quad \widehat{\mathbf{V}} := \mathbf{Z}|_{u \times w}^{\top} \cdot \mathbf{B}_{\tau}^Y \in \mathbb{R}^{\#\mathcal{I}(w), q}. \quad (2.4.4.21)$$

(I) We first compute the dense $\#\mathcal{I}(w) \times q$ -matrix $\widehat{\mathbf{V}}$ using a variant of **hmv()** from Code 2.4.1.19.

(II) Then we perform a rank- q update for the *generic \mathcal{H} -matrix* $\mathbf{X}|_{v \times w}$ calling

$$\mathbf{low_rank_update}(\mathbf{X}|_{v \times w}, \mathbf{A}_{\tau}^Y, \widehat{\mathbf{V}}).$$

→ **LRT**: When $\mathbf{X}|_{v \times w} = \mathbf{A}_{\sigma}^X (\mathbf{B}_{\sigma}^X)^{\top}$ we simply update these low-rank factors by

$$[\mathbf{A}_{\sigma}^X, \mathbf{B}_{\sigma}^X] = \mathbf{low_rank_sum}(\mathbf{A}_{\sigma}^X, \mathbf{B}_{\sigma}^X, \mathbf{A}_{\tau}^Y, \widehat{\mathbf{V}}).$$

⑤ Case: $\kappa := u \times w \in \mathbb{F}_{\text{far}}^Z$.

Hardly surprising, the considerations are fairly similar to the previous case. Owing to (2.4.4.1), a rank- q factorization of $\mathbf{Z}|_{u \times w}$ is available:

$$\mathbf{Z}|_{\kappa} = \mathbf{Z}|_{u \times w} = \mathbf{A}_{\kappa}^Z (\mathbf{B}_{\kappa}^Z)^{\top}, \quad \mathbf{A}_{\kappa}^Z \in \mathbb{R}^{\#\mathcal{I}(u), q}, \quad \mathbf{B}_{\kappa}^Z \in \mathbb{R}^{\#\mathcal{I}(w), q}. \quad (2.4.4.22)$$

Thus, the concrete update operation becomes

$$\mathbf{X}|_{\sigma} = \mathbf{X}|_{v \times w} \leftarrow \mathbf{X}|_{v \times w} + \widehat{\mathbf{U}} \cdot (\mathbf{B}_{\kappa}^Z)^{\top}, \quad \widehat{\mathbf{U}} := \mathbf{Y}|_{v \times u} \cdot \mathbf{A}_{\kappa}^Z \in \mathbb{R}^{\#\mathcal{I}(v), q}. \quad (2.4.4.23)$$

It can be broken down into the following two evaluations:

(I) Compute the dense $\#\mathcal{I}(v) \times q$ -matrix $\widehat{\mathbf{U}}$ based on **hmv()** from Code 2.4.1.19.

(II) Carry out a rank- q update of the *generic \mathcal{H} -matrix* $\mathbf{X}|_{v \times w}$ by means of

$$\mathbf{low_rank_update}(\mathbf{X}|_{v \times w}, \widehat{\mathbf{U}}, \mathbf{B}_{\kappa}^Z).$$

→ **LRT**: Again, we invoke the function **low_rank_sum()** from Code 2.4.2.24 and pass the low-rank factors of $\mathbf{X}|_{v \times w}$, $\widehat{\mathbf{U}}$, and \mathbf{B}_{κ}^Z as arguments.

At this point we know that

$$\tau := (v, u) \in \mathcal{B}_Y \setminus \mathbb{F}^Y \quad \text{and} \quad \kappa := (u, w) \in \mathcal{B}_Z \setminus \mathbb{F}^Z:$$

neither of these nodes of the block trees is a leaf and both will have four sons, as is guaranteed by the construction algorithm implemented in **build_block_tree()** from Code 2.4.1.16.

Concretely, if

$$\mathbf{sons}(u) = \{r_1, r_2\}, \quad \mathbf{sons}(v) = \{s_1, s_2\}, \quad \mathbf{sons}(w) = \{t_1, t_2\},$$

then

$$\mathbf{sons}(v \times u) = \{s_1 \times r_1, s_1 \times r_2, s_2 \times r_1, s_2 \times r_2\} \subset \mathcal{B}_Y, \quad (2.4.4.24a)$$

$$\mathbf{sons}(u \times w) = \{r_1 \times t_1, r_1 \times t_2, r_2 \times t_1, r_2 \times t_2\} \subset \mathcal{B}_Z. \quad (2.4.4.24b)$$

⑥ Case: $\sigma := (v, w) \in \mathcal{B}_X \setminus \mathbb{F}^X$ (no leaf of block tree).

As a non-leaf node of \mathcal{B}_X , σ has four sons:

$$\text{sons}(v \times w) = \{s_1 \times t_1, s_1 \times t_2, s_2 \times t_1, s_2 \times t_2\} \subset \mathcal{B}_X .$$

At this point we rely on the *recursive* application of the algorithm approximately implementing the update operation

$$\mathbf{X}|_{\sigma} = \mathbf{X}|_{v \times w} \leftarrow \mathbf{X}|_{v \times w} + \mathbf{Y}|_{v \times u} \cdot \mathbf{Z}|_{u \times w}, \quad (v, u) \in \mathcal{B}_Y, (u, w) \in \mathcal{B}_Z . \quad (2.4.4.10)$$

Since all involved blocks have four sons, see (2.4.4.24), the rules of matrix multiplication imply that, ignoring approximation errors, (2.4.4.10) is equivalent to

$$\begin{aligned} \mathbf{X}|_{s_1 \times t_1} &\leftarrow \mathbf{X}|_{s_1 \times t_1} + \mathbf{Y}|_{s_1 \times r_1} \cdot \mathbf{Z}|_{r_1 \times t_1} & \& \quad \mathbf{X}|_{s_1 \times t_1} &\leftarrow \mathbf{X}|_{s_1 \times t_1} + \mathbf{Y}|_{s_1 \times r_2} \cdot \mathbf{Z}|_{r_2 \times t_1} , \\ \mathbf{X}|_{s_1 \times t_2} &\leftarrow \mathbf{X}|_{s_1 \times t_2} + \mathbf{Y}|_{s_1 \times r_1} \cdot \mathbf{Z}|_{r_1 \times t_2} & \& \quad \mathbf{X}|_{s_1 \times t_2} &\leftarrow \mathbf{X}|_{s_1 \times t_2} + \mathbf{Y}|_{s_1 \times r_2} \cdot \mathbf{Z}|_{r_2 \times t_2} , \\ \mathbf{X}|_{s_2 \times t_1} &\leftarrow \mathbf{X}|_{s_2 \times t_1} + \mathbf{Y}|_{s_2 \times r_1} \cdot \mathbf{Z}|_{r_1 \times t_1} & \& \quad \mathbf{X}|_{s_2 \times t_1} &\leftarrow \mathbf{X}|_{s_2 \times t_1} + \mathbf{Y}|_{s_2 \times r_2} \cdot \mathbf{Z}|_{r_2 \times t_1} , \\ \mathbf{X}|_{s_2 \times t_2} &\leftarrow \mathbf{X}|_{s_2 \times t_2} + \mathbf{Y}|_{s_2 \times r_1} \cdot \mathbf{Z}|_{r_1 \times t_2} & \& \quad \mathbf{X}|_{s_2 \times t_2} &\leftarrow \mathbf{X}|_{s_2 \times t_2} + \mathbf{Y}|_{s_2 \times r_2} \cdot \mathbf{Z}|_{r_2 \times t_2} . \end{aligned} \quad (2.4.4.25)$$

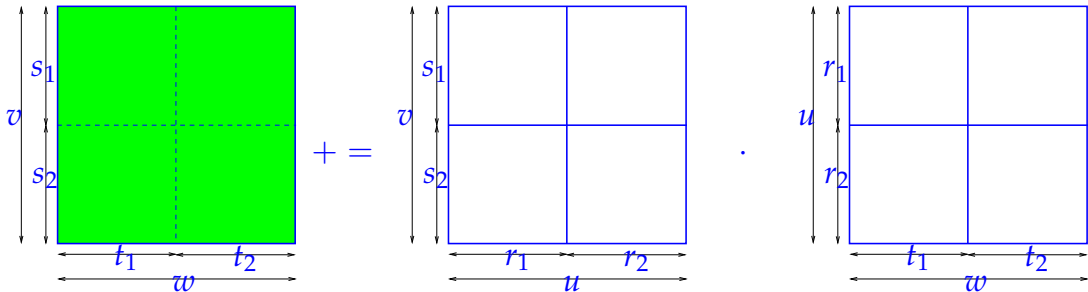
A visualization of these formulas is given in § 2.4.4.6, Fig. 115. All occurring matrices should be viewed as *generic \mathcal{H} -matrices*!

⑦ Case \rightarrow LRT: $\sigma := (v, w) \in \mathbb{F}_{\text{far}}^X$ (far-field leaf of \mathcal{B}_X)

From (2.4.4.1) we learn that $\mathbf{X}|_{v \times w}$ is a rank- q matrix to be stored in factorized form

$$\mathbf{X}|_{v \times w} = \mathbf{A}_{\sigma}^X (\mathbf{B}_{\sigma}^X)^{\top}, \quad \mathbf{A}_{\sigma}^X \in \mathbb{R}^{\#\mathcal{I}(v), q}, \quad \mathbf{B}_{\sigma}^X \in \mathbb{R}^{\#\mathcal{I}(w), q},$$

and this format must be preserved after the update. Conversely, both $\mathbf{Y}|_{v \times u}$ and $\mathbf{Z}|_{u \times w}$ should be regarded as *generic \mathcal{H} -matrices*.



Using the notation for the sons of the clusters $v \in \mathcal{T}_I$ and $w \in \mathcal{T}_J$ from above, we introduce four *temporary* rank- q matrices to be stored in factorized form

$$\mathbf{T}_{j,\ell} \in \mathbb{R}^{\#\mathcal{I}(s_j), \#\mathcal{I}(t_\ell)}, \quad \mathbf{T}_{j,\ell} = \mathbf{A}_{j,\ell} (\mathbf{B}_{j,\ell})^{\top}, \quad \mathbf{A}_{j,\ell} \in \mathbb{R}^{\#\mathcal{I}(s_j), q}, \quad \mathbf{B}_{j,\ell} \in \mathbb{R}^{\#\mathcal{I}(t_\ell), q}, \quad j, \ell \in \{1, 2\} . \quad (2.4.4.26)$$

The temporary matrices are supposed to represent the sub-blocks $\mathbf{X}|_{s_j \times t_\ell}$ of $\mathbf{X}|_{v \times w}$. Therefore, they are initialized through selecting a subset of the rows of \mathbf{A}_{σ}^X and \mathbf{B}_{σ}^X , respectively,

$$\mathbf{A}_{j,\ell} := \left(\mathbf{A}_{\sigma}^X \right)_{\mathcal{I}(s_j),:}, \quad \mathbf{B}_{j,\ell} := \left(\mathbf{B}_{\sigma}^X \right)_{\mathcal{I}(t_\ell),:}, \quad j, \ell \in \{1, 2\} . \quad (2.4.4.27)$$

After initialization, in analogy to (2.4.4.25), we update the temporary matrices according to

$$\begin{aligned} \mathbf{T}_{1,1} &\stackrel{r}{\leftarrow} \mathbf{T}_{1,1} + \mathbf{Y}|_{s_1 \times r_1} \cdot \mathbf{Z}|_{r_1 \times t_1} & \& \quad \mathbf{T}_{1,1} &\stackrel{r}{\leftarrow} \mathbf{T}_{1,1} + \mathbf{Y}|_{s_1 \times r_2} \cdot \mathbf{Z}|_{r_2 \times t_1} , \\ \mathbf{T}_{1,2} &\stackrel{r}{\leftarrow} \mathbf{T}_{1,2} + \mathbf{Y}|_{s_1 \times r_1} \cdot \mathbf{Z}|_{r_1 \times t_2} & \& \quad \mathbf{T}_{1,2} &\stackrel{r}{\leftarrow} \mathbf{T}_{1,2} + \mathbf{Y}|_{s_1 \times r_2} \cdot \mathbf{Z}|_{r_2 \times t_2} , \\ \mathbf{T}_{2,1} &\stackrel{r}{\leftarrow} \mathbf{T}_{2,1} + \mathbf{Y}|_{s_2 \times r_1} \cdot \mathbf{Z}|_{r_1 \times t_1} & \& \quad \mathbf{T}_{2,1} &\stackrel{r}{\leftarrow} \mathbf{T}_{2,1} + \mathbf{Y}|_{s_2 \times r_2} \cdot \mathbf{Z}|_{r_2 \times t_1} , \\ \mathbf{T}_{2,2} &\stackrel{r}{\leftarrow} \mathbf{T}_{2,2} + \mathbf{Y}|_{s_2 \times r_1} \cdot \mathbf{Z}|_{r_1 \times t_2} & \& \quad \mathbf{T}_{2,2} &\stackrel{r}{\leftarrow} \mathbf{T}_{2,2} + \mathbf{Y}|_{s_2 \times r_2} \cdot \mathbf{Z}|_{r_2 \times t_2} . \end{aligned} \quad (2.4.4.28)$$

We wrote $\stackrel{r}{\leftarrow}$ for “assignment after compression to rank- q format”. In terms of recursive calls to `lrt_mult_add()` the operations of (2.4.4.28) can be realized through

- (i) Initialize factors of temporary rank- q matrices as in (2.4.4.27).
- (ii) `lrt_mult_add(Aj,ℓ, Bj,ℓ, Y|sj×r1, Z|r1×tℓ); $j, \ell \in \{1, 2\}$ recursion.`
- (iii) `lrt_mult_add(Aj,ℓ, Bj,ℓ, Y|sj×r2, Z|r2×tℓ); $j, \ell \in \{1, 2\}$ recursion.`

Then we have to resort to the techniques presented in § 2.4.2.25 to merge the temporary matrices into a rank- q matrix:

$$\mathbf{A}_\sigma^X (\mathbf{B}_\sigma^X)^\top \stackrel{r}{\leftarrow} \begin{bmatrix} \mathbf{A}_{1,1} (\mathbf{B}_{1,1})^\top & \mathbf{A}_{1,2} (\mathbf{B}_{1,2})^\top \\ \mathbf{A}_{2,1} (\mathbf{B}_{2,1})^\top & \mathbf{A}_{2,2} (\mathbf{B}_{2,2})^\top \end{bmatrix}. \quad (2.4.4.29)$$

┘

§2.4.4.30 (\mathcal{H} -matrix multiplication: Summary of algorithm) We recast the scheme outlined above into a pseudocode. Beside functions introduced in Section 2.4.2, we need two more functions

The first is a function that computes the matrix×matrix product of a \mathcal{H} -matrix (argument X) and a regular (dense) matrix (argument M) returning another dense matrix:

```
matrix hmat_mult_dense(const  $\mathcal{H}$ -matrix X, const matrix M);
```

The second function returns the $X^\top \cdot M$ as a dense matrix, where X is a \mathcal{H} -matrix (argument X) and M is a regular (dense) matrix:

```
matrix hmat_transpose_mult_dense(const  $\mathcal{H}$ -matrix X, const matrix M);
```

Compatible sizes of the matrix arguments are taken for granted. Both functions can easily be implemented using the function `hmv()` from Code 2.4.1.19.

Pseudocode 2.4.4.31: Recursive \mathcal{H} -multiplication

```

1 void hmat_mult_add(ref  $\mathcal{H}$ -matrix X  $\in \mathbb{R}^{n,m}$ ,
2  $\mathcal{H}$ -matrix Y  $\in \mathbb{R}^{n,k}$ ,  $\mathcal{H}$ -matrix Z  $\in \mathbb{R}^{k,m}$ ) {
3    $\sigma := (v, w) \in \mathcal{B}_X := \mathcal{T}_I \times \mathcal{T}_J :=$  root of block tree associated with X
4    $\tau := (v, u) \in \mathcal{B}_Y := \mathcal{T}_I \times \mathcal{T}_K :=$  root of block tree associated with Y
5    $\kappa := (u, w) \in \mathcal{B}_Z := \mathcal{T}_K \times \mathcal{T}_J :=$  root of block tree associated with Z
6   if ( $\sigma \in \mathbb{F}_{\text{far}}^X$ ) // Case LRT
7     lrt_mult_add(AσX, BσX, Y, Z);
8   else {
9     switch {
10      case (sons(u) = ∅) { // Case ❶
11        low_rank_update(X, NτY, NκZ); // (2.4.4.15), Code 2.4.3.3
12        break;
13      }
14      case (sons(v) = ∅) { // Case ❷
15        matrix D = hmat_transpose_mult_dense(Z, (NτY)⊤) // (2.4.4.17)
16        NσX := NσX + D.tranpose(); // (2.4.4.17)
17        break;
18      }
19      case (sons(w) = ∅) { // Case ❸
20        matrix D = hmat_mult_dense(Y, NκZ); // (2.4.4.19)

```

```

21      $N_\sigma^X := N_\sigma^X + D$ ; // (2.4.4.19)
22     break;
23 }
24 case ((v,u) ∈  $\mathbb{F}_{\text{far}}^Y$ ) { // Case ④
25     matrix V = hmat_transpose_mult_dense(Z,  $B_\tau^Y$ ); // (2.4.4.21)
26     low_rank_update(X,  $A_\tau^Y$ , V); // (2.4.4.21), Code 2.4.3.3
27     break;
28 }
29 case ((u,w) ∈  $\mathbb{F}_{\text{far}}^Z$ ) { // Case ⑤
30     matrix U = hmat_mult_dense(Y,  $A_\kappa^Z$ ); // (2.4.4.23)
31     low_rank_update(X, U,  $B_\kappa^Z$ ); // (2.4.4.23), Code 2.4.3.3
32     break;
33 }
34 default: { // Case ⑥: recursion
35     foreach s ∈ sons(v) {
36         foreach t ∈ sons(w) {
37             foreach r ∈ sons(u) {
38                 hmat_mult_add( $X|_{s \times t}$ ,  $Y|_{s \times t}$ ,  $Z|_{r \times t}$ ); // (2.4.4.25)
39             } } } } }
40 }

```

Pseudocode 2.4.4.32: Recursive \mathcal{H} -multiplication

```

1 void lrt_mult_add(ref matrix A ∈  $\mathbb{R}^{n,q}$ , ref matrix B ∈  $\mathbb{R}^{m,q}$ ,
2 const  $\mathcal{H}$ -matrix Y ∈  $\mathbb{R}^{n,k}$ , const  $\mathcal{H}$ -matrix Z ∈  $\mathbb{R}^{k,m}$ ) {
3      $\tau := (v,u) \in \mathcal{B}_Y := \mathcal{T}_I \times \mathcal{T}_K :=$  root of block tree associated with Y
4      $\kappa := (u,w) \in \mathcal{B}_Z := \mathcal{T}_K \times \mathcal{T}_J :=$  root of block tree associated with Z
5     switch {
6         case (sons(u) =  $\emptyset$ ) { // Case ①
7             [A, B] = low_rank_sum(A, B,  $N_\tau^Y$ ,  $N_\kappa^Z$ ); // low-rank version of
8                 (2.4.4.15), Code 2.4.2.24
9             break;
10        }
11        case ( $\tau \in \mathbb{F}_{\text{far}}^Y$ ) { // Case ④
12            matrix V = hmat_transpose_mult_dense(Z,  $B_\tau^Y$ ); // (2.4.4.21)
13            [A, B] = low_rank_sum(A, B,  $A_\tau^Y$ , V);
14            break;
15        }
16        case ( $\kappa \in \mathbb{F}_{\text{far}}^Z$ ) { // Case ⑤
17            matrix U = hmat_mult_dense(Y,  $A_\kappa^Z$ ); // (2.4.4.23)
18            low_rank_sum(A, B, U,  $B_\kappa^Z$ ); // Code 2.4.2.24
19            break;
20        }
21        default: { // Case ⑦
22            for i=0,1 {
23                Cluster s = v.sons[i]; // Type defined in Code 2.4.1.14
24                for j=0,1 {
25                    Cluster t = w.sons[j]
26                    matrix At[i][j] = A( $\mathcal{I}(s)$ , :); // Initialization (2.4.4.27)

```

```

26   matrix Bt[i][j] = B(I(t),:); // Initialization (2.4.4.27)
27   foreach r ∈ sons(u) {
28       lrt_mult_add(At[i][j], Bt[i][j], Y|s×r, Z|r×t); // (2.4.4.28)
29   }
30 }
31 }
32 [A, B] = vert_horiz_merge(At, Bt);
33 } } }
    
```

The function `vert_horiz_merge()` takes one 2×2 -array $\begin{pmatrix} \mathbf{A}_{1,1} & \mathbf{A}_{1,2} \\ \mathbf{A}_{2,1} & \mathbf{A}_{2,2} \end{pmatrix}$ of matrices of size $n_1 \times q$ (for $\mathbf{A}_{1,1}$ and $\mathbf{A}_{1,2}$) and $n_2 \times q$ (for $\mathbf{A}_{2,1}$ and $\mathbf{A}_{2,2}$), and another 2×2 -array $\begin{pmatrix} \mathbf{B}_{1,1} & \mathbf{B}_{1,2} \\ \mathbf{B}_{2,1} & \mathbf{B}_{2,2} \end{pmatrix}$ of matrices with $\mathbf{B}_{1,1}, \mathbf{B}_{2,1} \in \mathbb{R}^{m_1 \times q}$, $\mathbf{B}_{1,2}, \mathbf{B}_{2,2} \in \mathbb{R}^{m_2 \times q}$ and returns the rank- q factors of a rank- q best approximation of the block matrix

$$\begin{bmatrix} \mathbf{A}_{1,1} \mathbf{B}_{1,1}^\top & \mathbf{A}_{2,1} \mathbf{B}_{2,1}^\top \\ \mathbf{A}_{1,2} \mathbf{B}_{1,2}^\top & \mathbf{A}_{2,2} \mathbf{B}_{2,2}^\top \end{bmatrix} \in \mathbb{R}^{n,m}, \quad n := n_1 + n_2, \quad m := m_1 + m_2.$$

Remark 2.4.4.33 (Partial blocks contributing to target block) In § 2.4.4.8 we saw that when multiplying \mathcal{H} -matrices parts of far-field blocks of a factor matrix may contribute to a block of the result matrix. Where is this case handled in Code 2.4.4.31?

We study a simple situation, where the matrix \mathbf{Z} consists of a single far-field block, whereas \mathbf{X} and \mathbf{Y} are split into four blocks of any type.

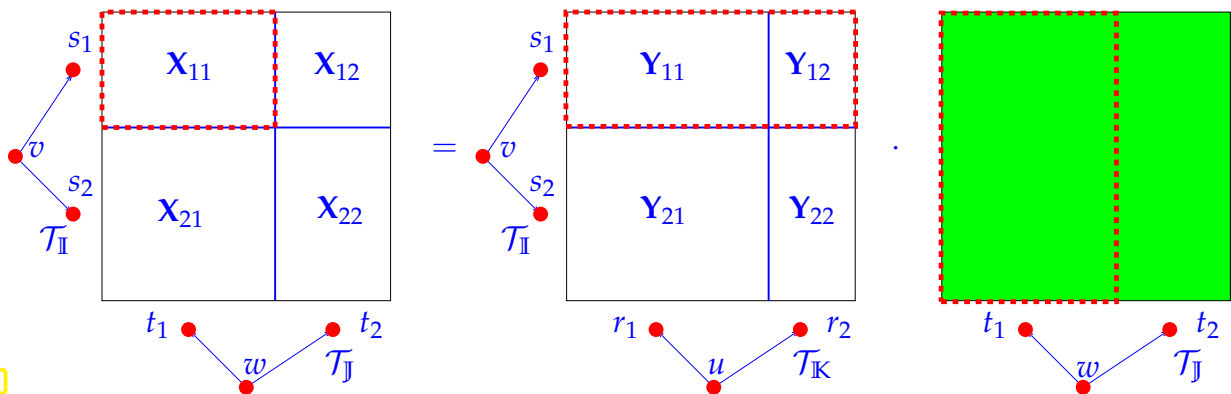


Fig. 116

Evidently, parts of the far-field block of \mathbf{Z} will have to be accessed when computing \mathbf{X}_{11} -block.

Running the algorithm of Code 2.4.4.31 we will hit Case ⑤, where the low-rank update of the generic \mathcal{H} -matrix \mathbf{X} will distribute the information contained in the far-field block of \mathbf{z} to all blocks of \mathbf{X} ! Thus, a special treatment of “partial blocks” is not necessary. ┘

EXAMPLE 2.4.4.34 (Multiplication of “simple” \mathcal{H} -matrices)

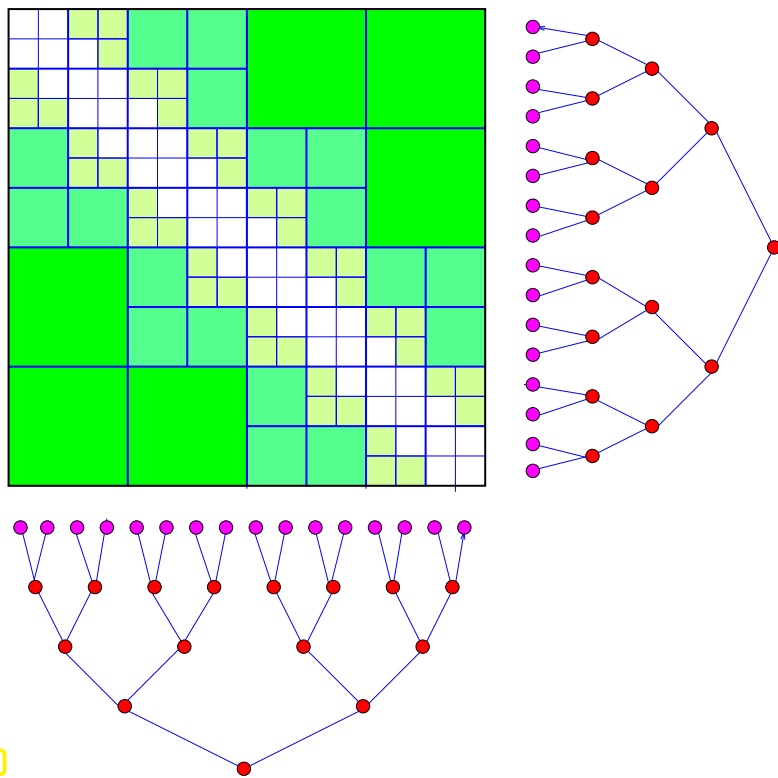


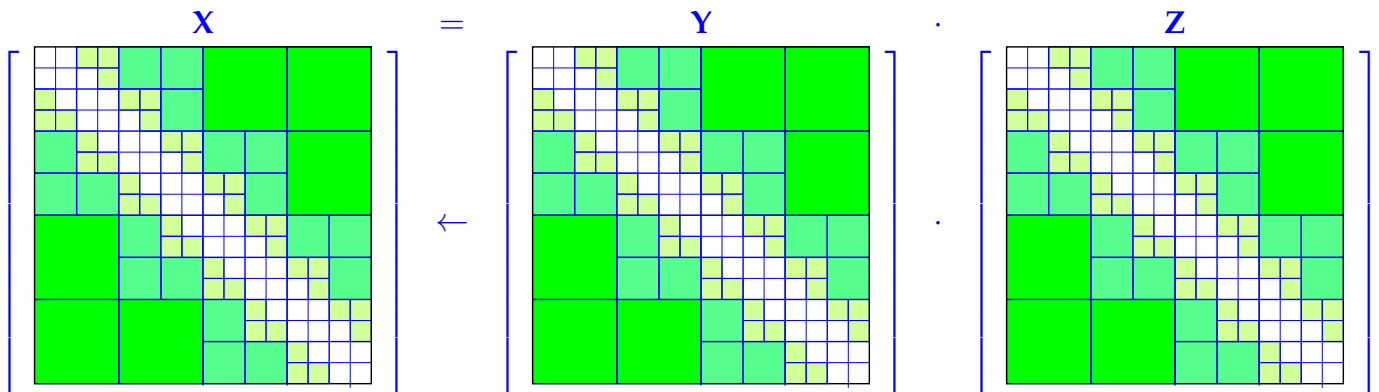
Fig. 117

We return to the “simple” square $2^L \times 2^L$ \mathcal{H} -matrix that we have already seen in Ex. 2.4.1.3 and § 2.4.1.9.

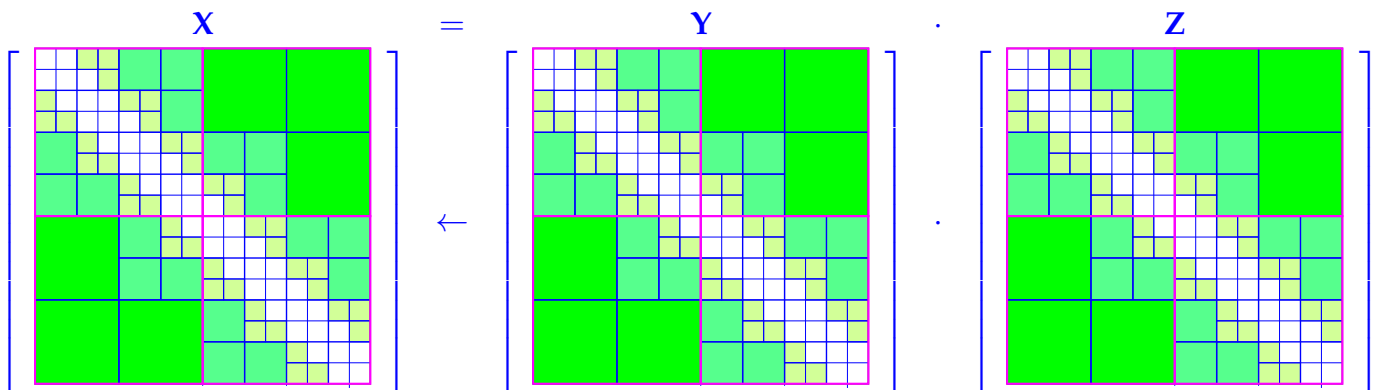
It is based on two identical balanced binary cluster trees. We restrict ourselves to the case $L = 4$ depicted in ??.

We examine the algorithm for the multiplication of \mathcal{H} -matrices when all involved matrices have this particular structure.

The reader is encouraged to play out the recursion with “pen and paper” using the provided graphical rendering of the \mathcal{H} -matrices.

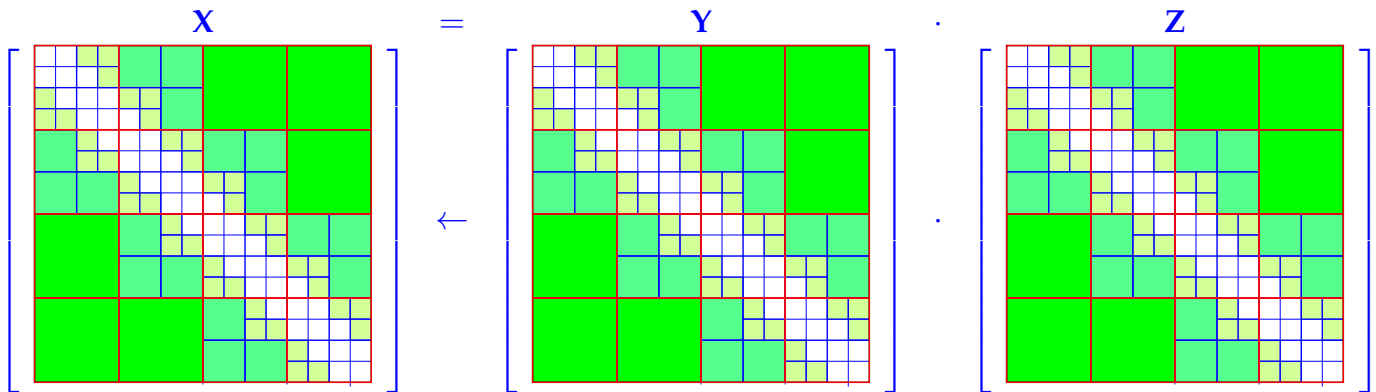


On the first level of the recursion only Case ⑦ is relevant, because no leaves of the block trees are visited:



On the next level of the recursion we encounter the first leaves of the block tree, all belonging to \mathbb{F}_{far}^* . This will trigger the Cases ④ and ⑤, whenever far-field blocks of Y and Z are involved. Calls to `lrt_mult_add()`

will be caused by far-field leaves of X



Remark 2.4.4.35 (Asymptotic complexity of \mathcal{H} -multiplication) Estimating the complexity of the algorithm `hmat_mult_add` from Code 2.4.4.31 is a formidable task and can only be done under some restrictive assumptions, see [Hac15, Sect. 7.8.3], [Bör21, Sect. 5.7]. For **balanced binary cluster trees**, we obtain

$$\text{cost}(\text{hmat_mult_add}) = O(q^2(n + m + k)) \quad \text{for } n, m, k \rightarrow \infty. \tag{2.4.4.36}$$

Remark 2.4.4.37 (Error incurred in multiplication of \mathcal{H} -matrices) During the execution of `hmat_mult_add()` errors are inevitably introduced in the various *recompression steps* where matrices of higher rank are projected onto rank- q matrices. This happens in Cases ❶, ❷ and ❸ when invoking `low_rank_update()`. It also happens in `lrt_mult_add()` during calls to `low_rank_sum()`.

However, the most severe approximation errors are usually introduced in `lrt_mult_add()` in the recompression step hidden in `vert_horiz_merge()`, which rely on vertical and horizontal recompression in turns, see § 2.4.2.25. To make matters worse, several invocations of `vert_horiz_merge()` may be nested during the recursion.

All this means that so far little progress has been made in quantifying the error incurred when multiplying two \mathcal{H} -matrices by means of `hmat_mult_add()`. This lack of error estimates is reflected by Ass. 2.4.4.5, which makes using `hmat_mult_add()` a matter of faith or optimism.

Fortunately, for \mathcal{H} -matrices arising from the compression of kernel matrices with asymptotically smooth kernels (\rightarrow Rem. 2.2.2.1) this crucial assumption seems to be satisfied.

2.4.5 Hierarchical LU-Decomposition

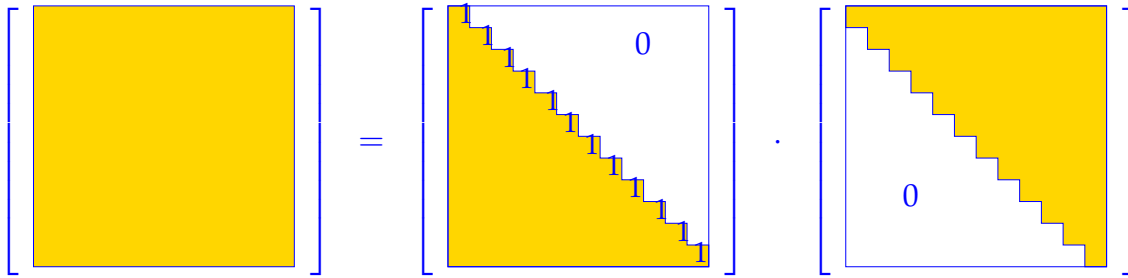
§2.4.5.1 (LU-decomposition: definition and existence) In [NumCSE Section 2.3.2] the LU-decomposition of square matrices was introduced as a matrix factorization leading to an algorithm for implementing **Gaussian elimination** in a two-stage way.

Definition 2.4.5.2. LU-dcomposition [NumCSE Def. 2.3.2.3]

Given a square matrix $A \in \mathbb{R}^{n,n}$, an *upper triangular matrix* $U \in \mathbb{R}^{n,n}$ and a normalized *lower triangular* matrix $L \in \mathbb{R}^{n,n}$ provide an **LU-decomposition** of A , if $A = L \cdot U$.

Refer to [NumCSE Def. 1.1.2.3] to learn the definition of a (normalized) triangular matrix, that is, a triangular

matrix with all diagonal entries = 1.



Without reordering an LU-decomposition of a square matrix may not exist, see [NumCSE Lemma 2.3.3.14]. [NumCSE Lemma 2.8.0.9] and [NumCSE Thm. 2.8.0.11] give us matrix properties ensuring the existence of an LU-decomposition, for instance the following:

Theorem 2.4.5.3. LU-decomposition of s.p.d. matrices

If $\mathbf{A} \in \mathbb{R}^{n,n}$ is symmetric positive definite (s.p.d., [NumCSE Def. 1.1.2.6]), then it has a *unique* LU-decomposition $\mathbf{A} = \mathbf{L} \cdot \mathbf{U}$ according to Def. 2.4.5.2.

Remark 2.4.5.4 (S.p.d. boundary element Galerkin matrices) S.p.d. Galerkin matrices usually arise from the boundary element discretization of the single layer and hypersingular boundary integral operators associated with $\mathbf{L}u := -\text{div}(\mathbf{A} \text{grad } u)$, \mathbf{A} s.p.d., see Thm. 1.3.5.17, Thm. 1.3.5.21, and Thm. 1.3.5.26 for details.

From now we consider a *symmetric positive definite* (s.p.d.) square hierarchical matrix $\mathbf{H} \in \mathbb{R}^{n,n}$ with fixed local rank q based on the *same binary cluster tree* $\mathcal{T}_{\mathbb{I}}$ of $\mathbb{I} := \{1, \dots, n\}$ for both rows and columns. Ex. 2.4.1.3 presents such a class of \mathcal{H} -matrices.

§2.4.5.5 (Compatible ordering) Obviously, the property of a matrix to be triangular will be destroyed by reordering its rows and columns. To make sense of a “triangular hierarchical matrix” the ordering of the index set has to match the structure of the cluster trees. To that end we assume a so-called **compatible ordering** of the indices, namely

1. that the sons of non-leaf clusters are ordered; we write $\text{sons}(v) = (s_1, s_2) \quad \forall v \in \mathcal{T}_{\mathbb{I}}$.
2. that

$$\text{sons}(v) = (s_1, s_2) \Rightarrow \{i \in \mathcal{I}(s_1) \ , \ j \in \mathcal{I}(s_2) \Rightarrow i < j\} . \tag{2.4.5.6}$$

A compatible ordering can easily be achieved by an index re-mapping based on [depth-first pre-order tree traversal](#).

Assumption 2.4.5.7.

For any square hierarchical matrix based on a binary row/column cluster tree and designated as triangular a compatible ordering of the index set is assumed.

Goal: Find (lower/upper) triangular square hierarchical matrices $\mathbf{L}_{\mathcal{H}}$ and $\mathbf{U}_{\mathcal{H}}$ of local rank q based on the same row and column cluster trees as \mathbf{H} and with the same block partition as \mathbf{H} such that $\mathbf{H} \approx \mathbf{L}_{\mathcal{H}} \cdot \mathbf{U}_{\mathcal{H}}$ with a small error in a relevant matrix norm.

§2.4.5.8 (Triangular linear systems of equations) The rationale for trying to find \mathcal{H} -LU factors $\mathbf{L}_{\mathcal{H}}$ and $\mathbf{U}_{\mathcal{H}}$ is the same as for the computation of an exact LU-decomposition, see [NumCSE § 2.3.2.15]. With LU-factors $\mathbf{L}_{\mathcal{H}}$ and $\mathbf{U}_{\mathcal{H}}$ of \mathbf{H} at our disposal we can (approximately) solve the linear system of equations by successive **forward** and **backward substitution**

$$\text{Solve } \mathbf{H}\vec{\mu} = \vec{\varphi} \Leftrightarrow \begin{cases} \bullet \text{ Solve } \mathbf{L}_{\mathcal{H}}\vec{\zeta} = \vec{\varphi}, \\ \bullet \text{ Solve } \mathbf{U}_{\mathcal{H}}\vec{\mu} = \vec{\zeta}. \end{cases} \quad (2.4.5.9)$$

Thus we need efficient algorithms for solving linear systems of equations with triangular hierarchical coefficient matrices.

Let $\mathbf{L}_{\mathcal{H}} \in \mathbb{R}^{n,n}$ be a square invertible lower triangular hierarchical matrix based on the row and column cluster tree $\mathcal{T}_{\mathbb{I}}$. We seek

$$\vec{\zeta} \in \mathbb{R}^{n,n}: \mathbf{L}_{\mathcal{H}}\vec{\zeta} = \vec{\varphi}, \quad \vec{\varphi} \in \mathbb{R}^n. \quad (2.4.5.10)$$

For $\mathbf{L}_{\mathcal{H}}$ to be invertible its diagonal blocks must be regular and cannot have low rank (compared to their size). They should all be near-field blocks:

Assumption 2.4.5.11. Near-field diagonal blocks

$$\text{adm}(v, v) = \text{false} \quad \forall v \in \mathcal{T}_{\mathbb{I}}$$

In particular, for all leaves $v \in \mathcal{T}_{\mathbb{I}}$ the matrix blocks $\mathbf{L}_{\mathcal{H}}|_{v \times v}$ are densely populated, invertible, lower triangular matrices directly available in the data structure.



Idea: Employ a **recursion** for solving (2.4.5.10): If

$$\text{sons}(\text{root}(\mathcal{T}_{\mathbb{I}})) = (s_1, s_2),$$

$$\blacktriangleright \mathbf{L}_{\mathcal{H}}\vec{\zeta} = \vec{\varphi} \Leftrightarrow \begin{bmatrix} \mathbf{L}_{\mathcal{H}}|_{s_1 \times s_1} & \mathbf{0} \\ \mathbf{L}_{\mathcal{H}}|_{s_2 \times s_1} & \mathbf{L}_{\mathcal{H}}|_{s_2 \times s_2} \end{bmatrix} \begin{bmatrix} \vec{\zeta}|_{s_1} \\ \vec{\zeta}|_{s_2} \end{bmatrix} = \begin{bmatrix} \vec{\varphi}|_{s_1} \\ \vec{\varphi}|_{s_2} \end{bmatrix}, \quad (2.4.5.12)$$

$$\blacktriangleright \vec{\zeta}|_{s_1} = (\mathbf{L}_{\mathcal{H}}|_{s_1 \times s_1})^{-1} \vec{\varphi}|_{s_1}, \quad \vec{\zeta}|_{s_2} = (\mathbf{L}_{\mathcal{H}}|_{s_2 \times s_2})^{-1} \left(\vec{\varphi}|_{s_2} - \mathbf{L}_{\mathcal{H}}|_{s_2 \times s_1} \vec{\zeta}|_{s_1} \right),$$

- with
- regular square lower triangular \mathcal{H} -matrices $\mathbf{L}_{\mathcal{H}}|_{s_1 \times s_1}, \mathbf{L}_{\mathcal{H}}|_{s_2 \times s_2}$,
 - general rectangular \mathcal{H} -matrix $\mathbf{L}_{\mathcal{H}}|_{s_2 \times s_1}$.

Apart from recursive calls, all we need is a code for the multiplication of the \mathcal{H} -matrix $\mathbf{L}_{\mathcal{H}}|_{s_2 \times s_1}$ with a vector. This is supplied by the function `hmv()` from Code 2.4.1.19.

Pseudocode 2.4.5.13: Solving a triangular linear system with \mathcal{H} -coefficient matrix

```

1 vector hmat_forw_elim(const  $\mathcal{H}$ -matrix  $\mathbf{L}_{\mathcal{H}}$ , const vector  $\vec{\varphi}$ ) {
2    $v :=$  root of cluster tree  $\mathcal{T}_{\mathbb{I}}$ , on which  $\mathbf{L}_{\mathcal{H}}$  is based
3   if (sons( $v$ ) =  $\emptyset$ ) return  $\mathbf{L}^{-1}\vec{\varphi}$ ; // standard forward elimination
4   else { // recursion according to (2.4.5.12)
5     ( $s_1, s_2$ ) := sons( $v$ );
6      $\vec{\mu}_1 :=$  hmat_forw_elim( $\mathbf{L}_{\mathcal{H}}|_{s_1 \times s_1}, \vec{\varphi}|_{s_1}$ );
7     vector  $\vec{\zeta} := \mathbf{0}$ ; hmv( $\mathbf{L}_{\mathcal{H}}|_{s_2 \times s_1}, \vec{\zeta}, \vec{\mu}_1$ );
8      $\vec{\tau} := \vec{\varphi}|_{s_2} - \vec{\zeta}$ ;
9     return hmat_forw_elim( $\mathbf{L}_{\mathcal{H}}|_{s_2 \times s_2}, \vec{\tau}$ );

```

```

10 | }
11 | }

```

We emphasize that `hmat_forw_elim()` returns the *exact* solution of the triangular linear system $\mathbf{L}_{\mathcal{H}}\vec{\zeta} = \vec{\phi}$. No approximation is done at any stage. \lrcorner

§2.4.5.14 (Recursive tiling algorithm for LU-decomposition → [NumCSE Rem. 2.3.2.12])

The recursive computation of the LU-decomposition of an s.p.d. matrix $\mathbf{H} \in \mathbb{R}^{n,n}$ is immediate from the following block matrix product:

$$\mathbf{L} \cdot \mathbf{U} = \mathbf{H} \Leftrightarrow \left[\begin{array}{c|c} \mathbf{L}_{11} & \mathbf{O} \\ \mathbf{L}_{21} & \mathbf{L}_{22} \end{array} \right] \cdot \left[\begin{array}{c|c} \mathbf{U}_{11} & \mathbf{U}_{12} \\ \mathbf{O} & \mathbf{U}_{22} \end{array} \right] = \left[\begin{array}{c|c} \mathbf{H}_{11} & \mathbf{H}_{12} \\ \mathbf{H}_{21} & \mathbf{H}_{22} \end{array} \right]. \quad (2.4.5.15)$$

Equating matching matrix blocks leads to the following steps for finding the unknown blocks of the normalized lower triangular matrix \mathbf{L} and the upper triangular matrix \mathbf{U} ($l + k = n$)

- ① Find $\mathbf{L}_{11}, \mathbf{U}_{11} \in \mathbb{R}^{k,k}$: $\mathbf{L}_{11} \cdot \mathbf{U}_{11} = \mathbf{H}_{11}$ $\hat{=}$ LU-decomposition \triangleright recursion,
- ② Find $\mathbf{U}_{12} \in \mathbb{R}^{k,l}$: $\mathbf{L}_{11}\mathbf{U}_{12} = \mathbf{H}_{12}$ $\hat{=}$ forward elimination,
Find $\mathbf{L}_{21} \in \mathbb{R}^{l,k}$: $\mathbf{L}_{21}\mathbf{U}_{11} = \mathbf{H}_{21}$ $\hat{=}$ forward elimination,
- ③ Find $\mathbf{L}_{22}, \mathbf{U}_{22} \in \mathbb{R}^{l,l}$: $\mathbf{L}_{22} \cdot \mathbf{U}_{22} = \mathbf{H}_{22} - \mathbf{L}_{21}\mathbf{U}_{12}$ $\hat{=}$ LU-decomposition \triangleright recursion.

The same scheme can be applied to an s.p.d. hierarchical matrix $\mathbf{H} \in \mathbb{R}^{n,n}$ seeking triangular hierarchical matrices $\mathbf{L}_{\mathcal{H}}, \mathbf{U}_{\mathcal{H}} \in \mathbb{R}^{n,n}$ based on the same row/column cluster tree $\mathcal{T}_{\mathbb{I}}$, with one new twist however:

In the set of hierarchical matrices we cannot solve the matrix equations exactly, but only approximately.

- ① Find triangular \mathcal{H} -matrices $\mathbf{L}_{11}, \mathbf{U}_{11} \in \mathbb{R}^{k,k}$: $\mathbf{L}_{11} \cdot \mathbf{U}_{11} \approx \mathbf{H}_{11}$ $\hat{=}$ \mathcal{H} -LU-decomposition, \triangleright recursion.
- ② Find \mathcal{H} -matrix $\mathbf{U}_{12} \in \mathbb{R}^{k,l}$: $\mathbf{L}_{11}\mathbf{U}_{12} \approx \mathbf{H}_{12}$ $\hat{=}$ forward elimination,
Find \mathcal{H} -matrix $\mathbf{L}_{21} \in \mathbb{R}^{l,k}$: $\mathbf{L}_{21}\mathbf{U}_{11} \approx \mathbf{H}_{21}$ $\hat{=}$ forward elimination,
- ③ Find triangular \mathcal{H} -matrices $\mathbf{L}_{22}, \mathbf{U}_{22} \in \mathbb{R}^{l,l}$: $\mathbf{L}_{22} \cdot \mathbf{U}_{22} \approx \mathbf{H}_{22} \ominus \mathbf{L}_{21} \odot \mathbf{U}_{12}$ $\hat{=}$ LU-decomposition, \triangleright recursion.

Note that the matrix operation $\mathbf{H}_{22} \ominus \mathbf{L}_{21} \odot \mathbf{U}_{12}$ has to be conducted in \mathcal{H} -arithmetic, because storage of any intermediate dense matrix will exceed the memory constraints of the data-sparse approach. Fortunately, this operation can be delivered by the \mathcal{H} -arithmetic routine `hmat_mult_add()` from Code 2.4.4.31. \lrcorner

§2.4.5.16 (Staggered matrix equations in \mathcal{H} -arithmetic) A key component of the recursive computation of an \mathcal{H} -LU-decomposition is the approximate solution of the linear system of equations

$$\mathbf{L}_{\mathcal{H}}\mathbf{X}_{\mathcal{H}} = \mathbf{Y}_{\mathcal{H}},$$

where

- ◆ $\mathbf{L}_{\mathcal{H}} \in \mathbb{R}^{n,n}$ is a *lower triangular* hierarchical matrix based on row/column cluster tree $\mathcal{T}_{\mathbb{I}}$, with an admissibility condition satisfying Ass. 2.4.5.11 and a compatible ordering of the index set, cf. Ass. 2.4.5.7.

- ◆ $\mathbf{Y} \in \mathbb{R}^{n,m}$ is a general hierarchical matrix based on the row cluster tree $\mathcal{T}_{\mathbb{I}}$ and column cluster tree $\mathcal{T}_{\mathbb{J}}$,
- ◆ $\mathbf{X}_{\mathcal{H}} \in \mathbb{R}^{n,m}$ is the *unknown* general \mathcal{H} -matrix based on the row cluster tree $\mathcal{T}_{\mathbb{I}}$ and column cluster tree $\mathcal{T}_{\mathbb{J}}$.

In order to motivate the recursive algorithm we single out two clusters: $v \in \mathcal{T}_{\mathbb{I}}$, $w \in \mathcal{T}_{\mathbb{J}}$.

If $\mathcal{I}(v) = \{1, \dots, n_1\}$, $n_1 \leq n$, and the indices in $\mathcal{I}(w)$ are assumed to be contiguous, then the matrix equations can be block-partitioned as follows

$$\begin{bmatrix} \mathbf{L}_{\mathcal{H}}|_{v \times v} & \mathbf{O} \\ * & * \end{bmatrix} \cdot \begin{bmatrix} * & \mathbf{X}_{\mathcal{H}}|_{v \times w} & * \\ * & * & * \end{bmatrix} = \begin{bmatrix} * & \mathbf{Y}_{\mathcal{H}}|_{v \times w} & * \\ * & * & * \end{bmatrix}$$

$$\blacktriangleright \mathbf{L}_{\mathcal{H}}|_{v \times v} \cdot \mathbf{X}_{\mathcal{H}}|_{v \times w} = \mathbf{Y}_{\mathcal{H}}|_{v \times w} . \tag{2.4.5.17}$$

Depending on the cluster pair $(v, w) \in \mathcal{T}_{\mathbb{I}} \times \mathcal{T}_{\mathbb{J}}$, which corresponds to a block of both $\mathbf{X}_{\mathcal{H}}$ and $\mathbf{Y}_{\mathcal{H}}$, we distinguish several cases:

- ❶: $(v, w) \in \mathbb{F}_{\text{near}}^{\mathbf{X}} = \mathbb{F}_{\text{near}}^{\mathbf{Y}}$ ($\mathbf{X}_{\mathcal{H}}|_{v \times w}$, $\mathbf{X}_{\mathcal{H}}|_{v \times w} \hat{=}$ near-field blocks)

In this case, while $\mathbf{L}_{\mathcal{H}}|_{v \times v}$ has to be regarded as a general lower triangular invertible hierarchical matrix, both $\mathbf{X}_{\mathcal{H}}|_{v \times w}$ and $\mathbf{X}_{\mathcal{H}}|_{v \times w}$ are stored as densely populated small matrices and we can apply `hmat_forw_elim()` from Code 2.4.5.13 to find the columns of $\mathbf{X}_{\mathcal{H}}|_{v \times w}$, because

$$\mathbf{L}_{\mathcal{H}}|_{v \times v} (\mathbf{X}_{\mathcal{H}}|_{v \times w})_{:,k} = (\mathbf{Y}_{\mathcal{H}}|_{v \times w})_{:,k} , \quad k \in \mathcal{I}(w) .$$

- ❷: $\sigma := (v, w) \in \mathbb{F}_{\text{far}}^{\mathbf{X}} = \mathbb{F}_{\text{far}}^{\mathbf{Y}}$ ($\mathbf{X}_{\mathcal{H}}|_{v \times w}$, $\mathbf{X}_{\mathcal{H}}|_{v \times w} \hat{=}$ far-field blocks)

Both $\mathbf{X}_{\mathcal{H}}|_{v \times w}$ and $\mathbf{X}_{\mathcal{H}}|_{v \times w}$ are rank- q matrices stored in factorized form, e.g.

$$\mathbf{Y}_{\mathcal{H}}|_{v \times w} = \mathbf{A}_{\sigma}^{\mathbf{Y}} \cdot (\mathbf{B}_{\sigma}^{\mathbf{Y}})^{\top} , \quad \mathbf{A}_{\sigma}^{\mathbf{Y}} \in \mathbb{R}^{\#\mathcal{I}(v),q} , \mathbf{B}_{\sigma}^{\mathbf{Y}} \in \mathbb{R}^{\#\mathcal{I}(w),q} . \tag{2.4.5.18}$$

We have to find the corresponding low-rank factors $\mathbf{A}_{\sigma}^{\mathbf{X}}$ and $\mathbf{B}_{\sigma}^{\mathbf{X}}$ for $\mathbf{X}_{\mathcal{H}}$. This can be done exactly, because, very generally, for a regular matrix $\mathbf{M} \in \mathbb{R}^{n,n}$, $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n,q}$, the matrix equation has a rank- q solution

$$\mathbf{M}\mathbf{X} = \mathbf{A}\mathbf{B}^{\top} \quad \blacktriangleright \quad \mathbf{X} = (\mathbf{M}^{-1}\mathbf{A})\mathbf{B}^{\top} . \tag{2.4.5.19}$$

The matrix $\mathbf{M}^{-1}\mathbf{A}$ is the solution $\hat{\mathbf{A}} \in \mathbb{R}^{n,1}$ of $\mathbf{M}\hat{\mathbf{A}} = \mathbf{A}$.

Thus, we resort to `hmat_forw_elim()` from Code 2.4.5.13 to determine the columns of $\mathbf{A}_{\sigma}^{\mathbf{X}}$ and just copy $\mathbf{B}_{\sigma}^{\mathbf{Y}}$

$$\mathbf{L}_{\mathcal{H}}|_{v \times v} (\mathbf{A}_{\sigma}^{\mathbf{X}})_{:,k} = (\mathbf{A}_{\sigma}^{\mathbf{Y}})_{:,k} , \quad k = 1, \dots, q , \quad \mathbf{B}_{\sigma}^{\mathbf{X}} = \mathbf{B}_{\sigma}^{\mathbf{Y}} , \tag{2.4.5.20}$$

which ensures $\mathbf{L}_{\mathcal{H}}|_{v \times v} \cdot \mathbf{X}_{\mathcal{H}}|_{v \times w} = \mathbf{Y}_{\mathcal{H}}|_{v \times w}$.

③: $(v, w) \notin \mathbb{F}^X = \mathbb{F}^Y$: \triangleright *recursion*

In this case neither v is a leaf of \mathcal{T}_I nor w is a leaf of \mathcal{T}_J and both will have two sons:

$$\text{sons}(v) =: (s_1, s_2) \quad [\text{ordered}] \quad , \quad \text{sons}(w) =: \{t_1, t_2\} .$$

The block-wise matrix product yields recursive formulas analogous to those derived from (2.4.5.12) and implemented in Code 2.4.5.13. deduced from

$$\begin{bmatrix} \mathbf{L}_{\mathcal{H}}|_{s_1 \times s_1} & \mathbf{O} \\ \mathbf{L}_{\mathcal{H}}|_{s_2 \times s_1} & \mathbf{L}_{\mathcal{H}}|_{s_2 \times s_2} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{X}_{\mathcal{H}}|_{s_1 \times t_1} & \mathbf{X}_{\mathcal{H}}|_{s_1 \times t_2} \\ \mathbf{X}_{\mathcal{H}}|_{s_2 \times t_1} & \mathbf{X}_{\mathcal{H}}|_{s_2 \times t_2} \end{bmatrix} = \begin{bmatrix} \mathbf{Y}_{\mathcal{H}}|_{s_1 \times t_1} & \mathbf{Y}_{\mathcal{H}}|_{s_1 \times t_2} \\ \mathbf{Y}_{\mathcal{H}}|_{s_2 \times t_1} & \mathbf{Y}_{\mathcal{H}}|_{s_2 \times t_2} \end{bmatrix} .$$

$$\blacktriangleright \begin{cases} \mathbf{L}_{\mathcal{H}}|_{s_1 \times s_2} \mathbf{X}_{\mathcal{H}}|_{s_1 \times t_1} = \mathbf{Y}_{\mathcal{H}}|_{s_1 \times t_1} , \\ \mathbf{L}_{\mathcal{H}}|_{s_1 \times s_1} \mathbf{X}_{\mathcal{H}}|_{s_1 \times t_2} = \mathbf{Y}_{\mathcal{H}}|_{s_1 \times t_2} , \\ \mathbf{L}_{\mathcal{H}}|_{s_2 \times s_2} \mathbf{X}_{\mathcal{H}}|_{s_2 \times t_1} = \mathbf{Y}_{\mathcal{H}}|_{s_2 \times t_1} \ominus \mathbf{L}_{\mathcal{H}}|_{s_2 \times s_1} \odot \mathbf{X}_{\mathcal{H}}|_{s_1 \times t_1} , \\ \mathbf{L}_{\mathcal{H}}|_{s_2 \times s_2} \mathbf{X}_{\mathcal{H}}|_{s_2 \times t_2} = \mathbf{Y}_{\mathcal{H}}|_{s_2 \times t_2} \ominus \mathbf{L}_{\mathcal{H}}|_{s_2 \times s_1} \odot \mathbf{X}_{\mathcal{H}}|_{s_1 \times t_2} , \end{cases}$$

where the operations \odot and \ominus indicate that some right-hand side matrices have to be computed using \mathcal{H} -arithmetic, more precisely the function `hmat_mult_add()` from (2.4.4.31).

Pseudocode 2.4.5.21: Approximately solving a triangular matrix equation in \mathcal{H} -arithmetic

```

1   $\mathcal{H}$ -matrix  $\leftarrow$  hmat_triag_solve( $\mathcal{H}$ -matrix  $\mathbf{L}_{\mathcal{H}}$ ,  $\mathcal{H}$ -matrix  $\mathbf{Y}_{\mathcal{H}}$ ) {
2   $\mathbf{X}_{\mathcal{H}}$  :=  $\mathcal{H}$ -matrix with block structure of  $\mathbf{Y}_{\mathcal{H}}$ 
3   $(v, w)$  := root of block tree of  $\mathbf{X}_{\mathcal{H}}/\mathbf{Y}_{\mathcal{H}}$ 
4   $(v, v)$  := root of block tree of  $\mathbf{L}_{\mathcal{H}}$ 
5  switch {
6  case  $((v, w) \in \mathbb{F}_{\text{near}}^Y)$ : { // Case ①: dense near-field block
7  foreach  $k \in \mathcal{I}(w)$  {
8   $(\mathbf{X})_{:,k}$  := hmat_forw_elim( $\mathbf{L}_{\mathcal{H}}$ ,  $(\mathbf{Y}_{\mathcal{H}}|_{v \times w})_{:,k}$ );
9  }
10 break;
11 }
12 case  $((v, w) \in \mathbb{F}_{\text{far}}^Y)$ : { // Case ②: far-field cluster pair
13 // In this case low-rank factorized representation:  $\mathbf{X}_{\mathcal{H}} = \mathbf{A}^X(\mathbf{B}^X)^{\top}$ 
14 for  $k := 1$  to  $q$  { // column-by-column triangular solve
15  $(\mathbf{A}^X)_{:,k}$  := hmat_forw_elim( $\mathbf{L}_{\mathcal{H}}$ ,  $(\mathbf{A}^Y)_{:,k}$ );
16 }
17  $\mathbf{B}^X := \mathbf{B}^Y$ ; break;
18 }
19 default: { // Case ③: recursion
20  $(s_1, s_2) := \text{sons}(v)$ ;  $(t_1, t_2) := \text{sons}(w)$ ;
21  $\mathbf{X}_{\mathcal{H}}|_{s_1 \times t_1} :=$  hmat_triag_solve( $\mathbf{L}_{\mathcal{H}}|_{s_1 \times s_1}$ ,  $\mathbf{Y}_{\mathcal{H}}|_{s_1 \times t_1}$ );
22  $\mathbf{X}_{\mathcal{H}}|_{s_1 \times t_2} :=$  hmat_triag_solve( $\mathbf{L}_{\mathcal{H}}|_{s_1 \times s_1}$ ,  $\mathbf{Y}_{\mathcal{H}}|_{s_1 \times t_2}$ );
23 hmat_mult_add( $\mathbf{Y}_{\mathcal{H}}|_{s_2 \times t_1}$ ,  $\mathbf{L}_{\mathcal{H}}|_{s_2 \times s_1}$ ,  $\mathbf{X}_{\mathcal{H}}|_{s_1 \times t_1}$ );
24 hmat_mult_add( $\mathbf{Y}_{\mathcal{H}}|_{s_2 \times t_2}$ ,  $\mathbf{L}_{\mathcal{H}}|_{s_2 \times s_1}$ ,  $\mathbf{X}_{\mathcal{H}}|_{s_1 \times t_2}$ );
25  $\mathbf{X}_{\mathcal{H}}|_{s_2 \times t_1} :=$  hmat_triag_solve( $\mathbf{L}_{\mathcal{H}}|_{s_2 \times s_2}$ ,  $\mathbf{Y}_{\mathcal{H}}|_{s_2 \times t_1}$ );
26  $\mathbf{X}_{\mathcal{H}}|_{s_2 \times t_2} :=$  hmat_triag_solve( $\mathbf{L}_{\mathcal{H}}|_{s_2 \times s_2}$ ,  $\mathbf{Y}_{\mathcal{H}}|_{s_2 \times t_2}$ );

```

```

27     }
28   }
29   return  $X_{\mathcal{H}}$ ;
30 }

```

We point out that it is only the use of the function `hmat_mult_add()` that renders the triangular solve by `hmat_triag_solve()` inexact. \lrcorner

§2.4.5.22 (Recursive LU-decomposition in \mathcal{H} -arithmetic \rightarrow § 2.4.5.14) Armed with the function `hmat_triag_solve()` from Code 2.4.5.21 we can implement the recursive algorithm outlined in § 2.4.5.14.

Pseudocode 2.4.5.23: Recursive \mathcal{H} -LU decomposition

```

1  [ $\mathcal{H}$ -matrix  $\mathcal{H}$ -matrix]  $\leftarrow$  hmat_lu_dec( $\mathcal{H}$ -matrix  $\mathbf{H}_{\mathcal{H}}$ ) {
2   $r :=$  root of cluster tree  $\mathcal{T}_{\mathbb{I}}$ , on which  $\mathbf{H}_{\mathcal{H}}$  is based;
3  if (sons( $r$ ) ==  $\emptyset$ ) { // leaf block
4    return lu_dec( $\mathbf{H}_{\mathcal{H}}$ ); // standard LU-decomposition
5  }
6  else {
7    ( $s_1, s_2$ ) := sons( $v$ );
8     $\mathbf{L}_{\mathcal{H}}, \mathbf{U}_{\mathcal{H}} :=$   $\mathcal{H}$ -matrices with the same block structure as  $\mathbf{H}_{\mathcal{H}}$ 
9    [ $\mathbf{L}_{\mathcal{H}}|_{s_1 \times s_1}, \mathbf{U}_{\mathcal{H}}|_{s_1 \times s_1}$ ] := hmat_lu_dec( $\mathbf{H}_{\mathcal{H}}|_{s_1 \times s_1}$ );
10    $\mathbf{U}|_{s_1 \times s_2} :=$  hmat_triag_solve( $\mathbf{L}|_{s_1 \times s_1}, \mathbf{H}_{\mathcal{H}}|_{s_1 \times s_2}$ );
11    $\mathbf{L}_{\mathcal{H}}|_{s_2 \times s_1} :=$  hmat_triag_solve( $\mathbf{U}|_{s_1 \times s_1}^{\top}, \mathbf{H}_{\mathcal{H}}|_{s_2 \times s_1}^{\top}$ );
12   hmat_mult_add( $\mathbf{H}|_{s_2 \times s_2}, -\mathbf{L}_{\mathcal{H}}|_{s_2 \times s_1}, \mathbf{U}_{\mathcal{H}}|_{s_1 \times s_2}$ );
13   [ $\mathbf{L}_{\mathcal{H}}|_{s_2 \times s_2}, \mathbf{U}|_{\mathcal{H}}|_{s_2 \times s_2}$ ] := hmat_lu_dec( $\mathbf{H}_{\mathcal{H}}|_{s_2 \times s_2}$ );
14   return [ $\mathbf{L}_{\mathcal{H}}, \mathbf{U}_{\mathcal{H}}$ ];
15 }

```

Remark 2.4.5.24 (\mathcal{H} -LU decomposition as preconditioner) On the one hand, thanks to powerful error estimates for the local separable approximation of singular asymptotically smooth kernels (\rightarrow Section 2.2.2), we have a rather good control of error committed when approximating a kernel collocation matrix or a boundary element Galerkin matrix by means of clustering techniques with geometric admissibility conditions.



On the other hand, the errors introduced by \mathcal{H} -arithmetic, which offers only an approximation of linear algebra operations, are very difficult to estimate. Therefore, the use of \mathcal{H} -LU decompositions $\mathbf{H}_{\mathcal{H}} = \mathbf{L}_{\mathcal{H}} \cdot \mathbf{U}_{\mathcal{H}}$ together with `hmat_forw_elim()` as an approximate solver for the linear system of equations $\mathbf{H}_{\mathcal{H}} \vec{\mu} = \vec{\varphi}$ is *not recommended*.

Fortunately, **preconditioners** for iterative Krylov subspace solvers (\rightarrow Rem. 2.3.6.15) need supply only approximate solvers. If the approximation is bad, convergence of the iterative solver will usually suffer, but it will not break down. Poor approximation afflicting \mathcal{H} -arithmetic can thus be offset.

Preconditioners based on \mathcal{H} -arithmetic

Inverses and LU-decompositions computed by \mathcal{H} -arithmetic should be used for preconditioning iterative solvers.

2.4.6 \mathcal{H}^2 -Matrices

In § 2.3.4.6 we observed that in the case of local low-rank compression of a kernel matrix based on bi-directional interpolation information about the underlying cardinal basis functions can be stored in the nodes of the cluster trees instead in the nodes of the block tree. This unlocks possibilities for more efficient implementation. This idea also forms the foundation for an enhanced \mathcal{H} -matrix format.

§2.4.6.1 (Triple-factor low-rank factorization) Let us return to the local rank- q separable approximation by **bi-directional interpolation** as introduced and analyzed in Section 2.2.1.3. Recall that on a box $B \subset D_x \times D_y \subset \mathbb{R}^d \times \mathbb{R}^d$ the kernel function $G : D_x \times D_y \rightarrow \mathbb{R}$ is replaced with

$$\tilde{G}(x, y) := \sum_{k=1}^q \sum_{\ell=1}^q \underbrace{G(\mathbf{t}_x^k, \mathbf{t}_y^\ell) b_k^x(x)}_{=:g_{k,\ell}(x)} \underbrace{b_\ell^y(y)}_{=:h_{k,\ell}(y)}, \quad (2.4.6.2)$$

- where
- ◆ $\mathbf{t}_x^k \in D_x, k = 1, \dots, q$, and $\mathbf{t}_y^\ell \in D_y, k = 1, \dots, q$ are **interpolation nodes**, and
 - ◆ $b_k^x : D_x \rightarrow \mathbb{R}$ and $b_\ell^y : D_y \rightarrow \mathbb{R}$ are the **cardinal functions** of the underlying interpolation operator, see § 2.2.1.22.

Thus, given collocation points $\mathbf{x}^1, \dots, \mathbf{x}^n \in D_x, \mathbf{y}^1, \dots, \mathbf{y}^m \in D_y$, the approximate kernel collocation matrix $\tilde{\mathbf{M}} \in \mathbb{R}^{n,m}$ is based on \tilde{G} , has rank q , and can be represented in a special **triple-factor form**

$$\begin{aligned} (\tilde{\mathbf{M}})_{i,j} &= \sum_{k=1}^q \sum_{\ell=1}^q G(\mathbf{t}_x^k, \mathbf{t}_y^\ell) b_k^x(\mathbf{x}^i) b_\ell^y(\mathbf{y}^j), \quad i = 1, \dots, n, \quad j = 1, \dots, m \\ \blacktriangleright \quad \tilde{\mathbf{M}} &= \mathbf{U} \mathbf{C} \mathbf{V}^\top, \quad \mathbf{C} := \left[G(\mathbf{t}_x^k, \mathbf{t}_y^\ell) \right]_{\substack{k=1, \dots, qx \\ \ell=1, \dots, q}} \in \mathbb{R}^{q,q}, \quad (2.2.1.47) \\ \mathbf{U}' &:= \left[b_k^x(\mathbf{x}^i) \right]_{\substack{i=1, \dots, n \\ k=1, \dots, q}} \in \mathbb{R}^{n,q}, \\ \mathbf{V} &:= \left[b_\ell^y(\mathbf{y}^j) \right]_{\substack{j=1, \dots, m \\ \ell=1, \dots, q}} \in \mathbb{R}^{m,q}. \end{aligned}$$

$$\left[\begin{array}{c} \tilde{\mathbf{M}} \end{array} \right] = \left[\begin{array}{c} \mathbf{U} \end{array} \right] \left[\begin{array}{c} \mathbf{C} \end{array} \right] \left[\begin{array}{c} \mathbf{V}^\top \end{array} \right].$$

A very similar triple-factor low-rank representation arises from bi-directional interpolation combined with clustering local low-rank compression applied to boundary element Galerkin matrices, see (2.3.6.8) in Section 2.3.6. ┘

Assume we use clustering with local rank- q separable approximation of a singular asymptotically smooth kernel obtained by bi-directional interpolation to build a hierarchical matrix representation $\mathbf{M}_{\mathcal{H}} \in \mathbb{R}^{n,m}$ (\rightarrow Def. 2.4.1.2) of a kernel collocation matrix $\mathbf{M} \in \mathbb{R}^{n,m}$ based on cluster trees $\mathcal{T}_{\mathbb{I}}$ (row cluster tree) and $\mathcal{T}_{\mathbb{J}}$ (column cluster tree). Then, using the notations of Def. 2.4.1.2 and, for a far-field cluster pair $\sigma = (v, w) \in \mathbb{F}_{\text{far}}$, writing $\mathbf{A}_\sigma \in \mathbb{R}^{\#\mathcal{I}(v),q}$ and $\mathbf{B}_\sigma \in \mathbb{R}^{\#\mathcal{I}(w),q}$ for the low-rank factors of $\mathbf{H}|_{v \times w}$ according to (2.4.1.6), we can *choose*

$$\mathbf{A} = \mathbf{U} \mathbf{C}, \quad \mathbf{B} = \mathbf{V} \quad \text{or} \quad \mathbf{A} = \mathbf{U}, \quad \mathbf{B} = \mathbf{V} \mathbf{C}^\top.$$

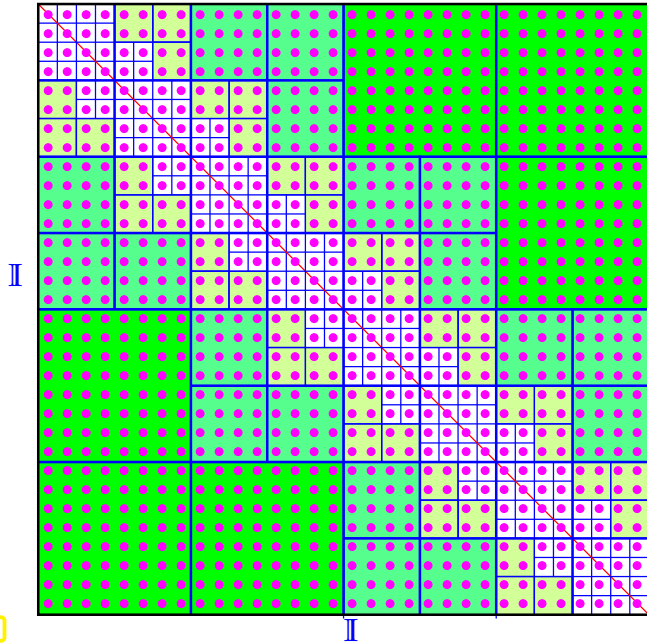
Do we really have to break the beautiful symmetry inherent in bi-directional interpolation in this way? Of course not, because we can simply retain the three matrix factors as we have already seen in § 2.3.4.10.

EXAMPLE 2.4.6.3 (Storage requirements of double-factor and triple-factor representations) In this example we revisit Ex. 2.3.2.1, which discussed clustering for $d = 1$ applied to a kernel collocation matrix

$\mathbf{M} = [G(\xi_i, \eta_j)]_{i,j=1}^n \in \mathbb{R}^{n,n}$, $n = 2^{L-1}$, and equidistant collocation points

$$\xi_i := \frac{i - 1/2}{n}, \quad \eta_j := \frac{j - 1/2}{n}, \quad i, j \in \{1, \dots, n\}. \tag{2.3.2.2}$$

We use the geometric admissibility condition $\eta(B) \leq \frac{1}{2}$ based on the admissibility measure η as defined in (2.2.2.7). Here we adopt the convention that a single collocation point has a centered square bounding box of width $\frac{1}{2n}$. We use the same row and column balanced binary cluster tree, whose leaves contain a single collocation point, see Fig. 107. Far-field cluster pairs must not comprise leaves.



◁ Visualization of hierarchical matrix structure for $L = 6$.
 Each \bullet corresponds to a matrix entry.
 $\square \hat{=}$ far-field blocks

Fig. 118

Counting as in Ex. 2.3.2.1, see (2.3.2.6), we find

$$\begin{aligned} \#\{\text{near-field blocks}\} &= 3 \cdot 2^L - 8, \\ \#\{\text{far-field blocks on level } \ell\} &= 6(2^\ell - 1), \quad \ell = 1, \dots, L - 3. \end{aligned}$$

Each near field block contains a single matrix entry, each far-field block on level $\ell \in \{1, \dots, L - 3\}$ holds $2^{L-\ell-2}$ indices. Hence the total floating-point storage requirements for the standard hierarchical matrix data structure with local rank q are

$$\text{storage}(\mathcal{H}\text{-matrix}) = 3 \cdot 2^L - 8 + \sum_{\ell=1}^{L-3} 6(2^\ell - 1) \cdot 2q \cdot 2^{L-\ell-2} = O(Ln) \quad \text{for } L \rightarrow \infty. \tag{2.4.6.4}$$

near-field blocks
no. of far-field blocks
low-rank factors

In § 2.3.4.6 we learned that once a triple-factor representation of far-field blocks $\sigma = (v, w) \in \mathcal{T}_I \times \mathcal{T}_I$ is available, the matrices \mathbf{U}_σ and \mathbf{V}_σ depend only on the clusters v and w , respectively, see (2.3.4.8). Thus they can be stored in the nodes of the cluster trees (except for the leaf level in this example). Only the coupling matrices \mathbf{C}_σ remain to be stored in the far-field blocks. This leads to total floating point storage requirements

$$\text{storage}(\text{"§ 2.3.4.6"}) = 3 \cdot 2^L - 8 + \sum_{\ell=1}^{L-3} q^2 6(2^\ell - 1) + 2 \cdot \sum_{\ell=0}^{L-1} 2^\ell \cdot 2^{L-\ell} = O(Ln) \quad \text{for } L \rightarrow \infty. \tag{2.4.6.5}$$

near-field blocks = $O(n)$
storage for $\mathbf{C}_\sigma = O(n)$
storage for $\mathbf{U}_v, \mathbf{V}_w$

We observe that the asymptotic storage requirements are determined by the last term! ┘

§2.4.6.6 (Transfer matrices) Let us assume that in Eq. (2.2.1.47) we use a bi-directional interpolation scheme based on tensor-product polynomial interpolation of degree $p \in \mathbb{N}$, as explained in § 2.2.1.37. This means, $q = (p + 1)^d$. The space, in which we approximate the kernel $(x, y) \mapsto G(x, y)$ on every far-field cluster box $\text{box}(v) \times \text{box}(w)$, $(v, w) \in \mathbb{F}_{\text{far}}$, will be *the same* for all far-field clusters, namely the tensor-product polynomial space $\mathcal{TP}_p(\mathbb{R}^{2d})$ (\rightarrow Def. 1.4.3.80). Moreover, for all clusters $v \in \mathcal{T}_{\mathbb{I}}$, $w \in \mathcal{T}_{\mathbb{J}}$, the space spanned by the cardinal functions (aka tensor-product Lagrange polynomials) $x \mapsto b_k^x(x)$ and $y \mapsto b_\ell^y(y)$, respectively, will coincide with $\mathcal{TP}_p(\mathbb{R}^d)$:

$$\forall v \in \mathcal{T}_{\mathbb{I}}: \text{Span}\{b_k^x\}_{k=1}^q = \mathcal{TP}_p(\mathbb{R}^d) \quad , \quad \forall w \in \mathcal{T}_{\mathbb{J}}: \text{Span}\{b_\ell^y\}_{\ell=1}^q = \mathcal{TP}_p(\mathbb{R}^d) . \quad (2.4.6.7)$$

Though not expressed by the notation, the cardinal functions depend on the clusters, of course.

Let us restrict ourselves to the row tree $\mathcal{T}_{\mathbb{I}}$ and focus on non-leaf clusters $v \in \mathcal{T}_{\mathbb{I}}$. For the associated cardinal functions $\in \mathcal{TP}_p(\mathbb{R}^d)$ we write b_k^v , $k = 1, \dots, q$. Owing to (2.4.6.7) they can be represented by linear combinations of the cardinal functions of each son cluster:

$$\forall s \in \text{sons}(v): \quad b_k^v = \sum_{\nu=1}^q t_{k,\nu}^{v,s} b_\nu^s, \quad t_{k,\nu}^{v,s} = b_k^v(t_\nu^s), \quad k, \nu \in \{1, \dots, q\}, \quad (2.4.6.8)$$

with $\{t_1^s, \dots, t_q^s\} \subset \text{box}(s)$ standing for the set of interpolation nodes on the son cluster $s \in \mathcal{T}_{\mathbb{I}}$. The formula for the expansion coefficients $t_{k,\nu}^{v,s}$ is immediate from (2.2.1.28). This permits us to rewrite the low-rank factor matrix \mathbf{U}_v for the cluster v in terms of the corresponding matrices for its sons:

$$(\mathbf{U}_v)_{i,k} = b_k^v(x^i) = \sum_{\nu=1}^q t_{k,\nu}^{v,s} b_\nu^s(x^i), \quad i \in \mathcal{I}(v), \quad k = 1, \dots, q, \quad (2.4.6.9a)$$


$$\blacktriangleright \quad s \in \text{sons}(v) \quad \Rightarrow \quad (\mathbf{U}_v)_{i,k} = \sum_{\nu=1}^q t_{k,\nu}^{v,s} (\mathbf{U}_s)_{i,\nu}, \quad i \in \mathcal{I}(s) . \quad (2.4.6.9b)$$

For a cluster $v \in \mathcal{T}_{\mathbb{I}}$ and one of its sons $s \in \text{sons}(v)$ we collect the coefficients $t_{k,\nu}^{v,s}$ from (2.4.6.9) in the **transfer matrix** $\mathbf{T}^{v,s} \in \mathbb{R}^{q,q}$:

$$(\mathbf{T}^{v,s})_{k,\nu} = t_{k,\nu}^{v,s} = b_k^v(t_\nu^s), \quad k, \nu = 1, \dots, q . \quad (2.4.6.10)$$

In the particular case of a binary cluster tree $\mathcal{T}_{\mathbb{I}}$ with $\text{sons}(v) = (s_1, s_2)$, the rules of matrix multiplication imply

$$(2.4.6.9b) \quad \blacktriangleright \quad \mathbf{U}_v = \begin{bmatrix} \mathbf{U}_{s_1} (\mathbf{T}^{v,s_1})^\top \\ \mathbf{U}_{s_2} (\mathbf{T}^{v,s_2})^\top \end{bmatrix} . \quad (2.4.6.11)$$

 Notation: Since every cluster, except for the root cluster, has exactly one father, we may associate the transfer matrix $\mathbf{T}^{v,s}$ with the son cluster s and, when doing so, denote it by \mathbf{T}_s .

┘

§2.4.6.12 (Storing hierarchical matrix based on transfer matrices) Let us assume the setting of the previous paragraph § 2.4.6.6 with triple-factor low-rank representation $\mathbf{M}_{\mathcal{H}}|_\sigma = \mathbf{U}_v \mathbf{C}_\sigma \mathbf{V}_w^\top$ of the far-field matrix blocks as in (2.2.1.47). The relationship (2.4.6.11) suggests a more efficient way to store the hierarchical matrix $\mathbf{M}_{\mathcal{H}}$.



- Idea:
- ◆ Store $\mathbf{U}_v/\mathbf{V}_w$ in the leaf nodes only.
 - ◆ Store the **transfer matrices** $\mathbf{T}_v/\mathbf{T}_w$ in all (son) clusters $v \in \mathcal{T}_{\mathbb{I}}/w \in \mathcal{T}_{\mathbb{J}}$.

$$\blacktriangleright \quad \text{storage}(\text{transfer matrices}) = q^2 \cdot (\#\mathcal{T}_{\mathbb{I}} + \#\mathcal{T}_{\mathbb{J}}) . \quad (2.4.6.13)$$

In the case of **balanced binary trees**, we know $\#\mathcal{T}_{\mathbb{I}} \leq 2\#\mathbb{I}$ and $\#\mathcal{T}_{\mathbb{J}} \leq 2\#\mathbb{J}$ and in this case

$$\blacktriangleright \text{storage}(\text{transfer matrices}) \leq 2q^2(m+n).$$

Recall that in Ex. 2.4.6.3 the asymptotically largest amount of storage was used for the cluster-specific factors of the triple-factor low-rank factorization, cf. (2.4.6.5). So a data structure relying on transfer matrices can achieve an asymptotic memory complexity of $O(n)$ in this example! \lrcorner

§2.4.6.14 (\mathcal{H}^2 -matrices) Storing hierarchical matrices with triple-factor low-rank representations of far-field blocks and the possibility of a “leaf-down” successive computation of the cluster-specific transfer matrices according to (2.4.6.11) can be abstracted into a new variant of hierarchical matrices.

Definition 2.4.6.15. \mathcal{H}^2 -matrices

Given $n, m \in \mathbb{N}$, a matrix $\mathbf{H} \in \mathbb{R}^{n,m}$ is a **\mathcal{H}^2 -matrix** with local rank q , if there exist

- cluster trees $\mathcal{T}_{\mathbb{I}}$ (**row tree**) and $\mathcal{T}_{\mathbb{J}}$ (**column tree**) for $\mathbb{I} := \{1, \dots, n\}$ and $\mathbb{J} := \{1, \dots, m\}$,
- an abstract admissibility condition $\text{adm} : \mathcal{T}_{\mathbb{I}} \times \mathcal{T}_{\mathbb{J}} \rightarrow \{\text{true}, \text{false}\}$,
- **transfer matrices** $\mathbf{T}_v/\mathbf{T}_w$ for all $v \in \mathcal{T}_{\mathbb{I}} \setminus \{\text{root}(\mathcal{T}_{\mathbb{I}})\}/w \in \mathcal{T}_{\mathbb{J}} \setminus \{\text{root}(\mathcal{T}_{\mathbb{J}})\}$,

such that

$$(i) \quad \mathbf{H}|_{v \times w} = \mathbf{U}_v \mathbf{C}_{(v,w)} \mathbf{V}_w^T, \quad \begin{cases} \mathbf{U}_v \in \mathbb{R}^{\#\mathcal{I}(v),q}, \\ \mathbf{C}_{(v,w)} \in \mathbb{R}^{q,q}, \\ \mathbf{V}_w \in \mathbb{R}^{\#\mathcal{I}(w)} \end{cases}, \quad \forall (v,w) \in \mathbb{F}_{\text{far}}, \quad (2.4.6.16a)$$

$$(ii) \quad \mathbf{U}_v = [\mathbf{U}_s(\mathbf{T}_s)^T]_{s \in \text{sons}(v)}, \quad v \in \mathcal{T}_{\mathbb{I}}, \quad \mathbf{V}_w = [\mathbf{V}_t(\mathbf{T}_t)^T]_{t \in \text{sons}(w)}, \quad w \in \mathcal{T}_{\mathbb{J}}, \quad (2.4.6.16b)$$

$$(iii) \quad \#\mathcal{I}(v), \#\mathcal{I}(w) \leq q \quad \forall \text{leaves } v \in \mathcal{T}_{\mathbb{I}}, w \in \mathcal{T}_{\mathbb{J}}, \quad (2.4.6.16c)$$

where the far field $\mathbb{F}_{\text{far}} \subset \mathcal{T}_{\mathbb{I}} \times \mathcal{T}_{\mathbb{J}}$ is defined as in Def. 2.4.1.2.

The matrices \mathbf{U}_v , $v \in \mathcal{T}_{\mathbb{I}}$, are called the **row cluster bases**, \mathbf{V}_w , $w \in \mathcal{T}_{\mathbb{J}}$, the **column cluster basis**, and $\mathbf{C}_{(v,w)}$ the coupling matrices.

The estimate

$$\blacktriangleright \text{storage}(\text{transfer matrices}) = q^2 \cdot (\#\mathcal{T}_{\mathbb{I}} + \#\mathcal{T}_{\mathbb{J}}). \quad (2.4.6.13)$$

for the amount of memory needed to store the transfer matrices still holds for \mathcal{H}^2 -matrices. For leaf clusters v , w of $\mathcal{T}_{\mathbb{I}}$ or $\mathcal{T}_{\mathbb{J}}$, respectively, we have to keep \mathbf{U}_v or \mathbf{V}_w , which will consume another $\leq q(\#\mathcal{T}_{\mathbb{I}} + \#\mathcal{T}_{\mathbb{J}})$ floating point numbers. For estimates addressing the amount of storage needed for the coupling matrices and the near-field blocks refer to § 2.3.4.20 and (2.3.4.30); the role of the sparsity measure $\text{spm}(\mathbb{F})$ from Def. 2.3.4.27 remains unchanged. Summing up, we can bound

$$\text{storage}(\mathcal{H}^2\text{-matrix}) \leq \sum_{(v,w) \in \mathbb{F}_{\text{near}}} \#\mathcal{I}(v) + \#\mathcal{I}(w) + (\text{spm}(\mathbb{F}_{\text{far}}) + 1)q^2 \cdot (\#\mathcal{T}_{\mathbb{I}} + \#\mathcal{T}_{\mathbb{J}}). \quad (2.4.6.17)$$

In the special case of a **balanced binary tree** as constructed by `buildRec` from Code 2.3.3.5, the number of near-field blocks and the number of clusters is bounded by $n + m$, which implies

$$\text{storage}(\mathcal{H}^2\text{-matrix from } \text{buildRec}()) = O(q^2(m+n)) \quad \text{for } n, m \rightarrow \infty, \quad (2.4.6.18)$$

where we assumed that the sparsity measure as introduced in Def. 2.3.4.27 is uniformly bounded with respect to n, m . Compared to Ex. 2.4.6.3, (2.4.6.5), we could remove the number of levels from the asymptotic estimate of the required storage. \lrcorner

Remark 2.4.6.19 (Data structure for \mathcal{H}^2 -matrices) Any object of a type compatible with the concept of an \mathcal{H}^2 -matrix with local rank q according to Def. 2.4.6.15 must provide

- access to suitable objects for both row and column cluster tree $\mathcal{T}_{\mathbb{I}}$ and $\mathcal{T}_{\mathbb{J}}$,
- instant access to
 - the cluster bases $\mathbf{U}_v \in \mathbb{R}^{\#\mathcal{I}(v),q}$ and $\mathbf{V}_w \in \mathbb{R}^{\#\mathcal{I}(w),q}$ for leaf nodes,
 - the transfer matrices $\mathbf{T}_v \in \mathbb{R}^{q,q}$, $\mathbf{T}_w \in \mathbb{R}^{q,q}$ for all $v \in \mathcal{T}_{\mathbb{I}}$ and $w \in \mathcal{T}_{\mathbb{J}}$.
 - the coupling matrices $\mathbf{C}_{(v,w)} \in \mathbb{R}^{q,q}$ for all far-field cluster pairs $(v,w) \in \mathbb{F}_{\text{far}}$,
 - the dense near-field blocks $\mathbf{H}|_{v \times w} \in \mathbb{R}^{\#\mathcal{I}(v),\#\mathcal{I}(w)}$ for all $(v,w) \in \mathbb{F}_{\text{near}}$.

┘

§2.4.6.20 (\mathcal{H}^2 -matrix \times vector multiplication) We extend the considerations of Section 2.3.5 about how to organize the matrix \times vector product $\vec{\rho} := \mathbf{H}\vec{\mu}$ efficiently in the case of local triple-factor low-rank representation

$$\widetilde{\mathbf{M}}|_{v \times w} = \mathbf{U}_v \cdot \mathbf{C}_{(v,w)} \cdot \mathbf{V}_w^\top, \quad (v,w) \in \mathbb{F}_{\text{far}},$$

of the matrix $\widetilde{\mathbf{M}}$ to \mathcal{H}^2 -matrices, which feature the additional component of transfer matrices, see Def. 2.4.6.15, (2.4.6.16b).

Recall the **restrict-to-cluster** restriction and index remapping operation for $w \in \mathcal{T}_{\mathbb{J}}$

$$\mathbf{R}_w : \mathbb{R}^m \rightarrow \mathbb{R}^{\#\mathcal{I}(w)}, \quad \mathbf{R}_w(\vec{\mu}) := \begin{bmatrix} \mu_{j_1} \\ \vdots \\ \mu_{j_\ell} \end{bmatrix}, \quad \text{with } \mathcal{I}(w) = \{j_1, \dots, j_\ell\}, \quad \ell := \#\mathcal{I}(w). \quad (2.3.5.1)$$

and the **expand-from-cluster** assembly operation for a row cluster $v \in \mathcal{T}_{\mathbb{I}}$:

$$\mathbf{E}_v : \mathbb{R}^{\#\mathcal{I}(v)} \rightarrow \mathbb{R}^n, \quad (\mathbf{E}_v \vec{v})_i := \begin{cases} v_\ell & , \text{ if } i_\ell = i, \\ 0 & , \text{ if } k \notin \mathcal{I}(v), \end{cases} \quad \text{with } \mathcal{I}(v) = \{i_1, \dots, i_k\}, \quad k := \#\mathcal{I}(v), \quad (2.3.5.2)$$

with associated matrices \mathbf{R}_w and \mathbf{E}_v . Consider a far-field cluster pair $(v,w) \in \mathbb{F}_{\text{far}}$ consisting of non-leaf clusters $v \in \mathcal{T}_{\mathbb{I}}$, $w \in \mathcal{T}_{\mathbb{J}}$ with

$$\text{sons}(v) = \{s_1, s_2\}, \quad \text{sons}(w) = \{t_1, t_2\}.$$

Then, (2.4.6.16b) implies

$$\mathbf{U}_v = \begin{bmatrix} \mathbf{U}_{s_1}(\mathbf{T}_{s_1})^\top \\ \mathbf{U}_{s_2}(\mathbf{T}_{s_2})^\top \end{bmatrix}, \quad \mathbf{V}_w = \begin{bmatrix} \mathbf{V}_{t_1}(\mathbf{T}_{t_1})^\top \\ \mathbf{V}_{t_2}(\mathbf{T}_{t_2})^\top \end{bmatrix}. \quad (2.4.6.21)$$

We right-multiply $\mathbf{H} \in \mathbb{R}^{n,m}$ with a column vector $\vec{\mu} \in \mathbb{R}^m$, storing the result in $\vec{\rho} \in \mathbb{R}^n$. We focus on the contribution of the far-field block $\mathbf{H}|_{v \times w}$, $(v,w) \in \mathbb{F}_{\text{far}}$, \mathbf{H} in \mathcal{H}^2 -format:

$$(2.4.6.16a) \quad \blacktriangleright \quad \vec{\rho} \leftarrow \vec{\rho} + \mathbf{E}_v \mathbf{U}_v \mathbf{C}_{(v,w)} \mathbf{V}_w^\top \mathbf{R}_w \vec{\mu}.$$

Next, based on (2.4.6.21) the second summand can be expressed as

$$\mathbf{E}_v \mathbf{U}_v \mathbf{C}_{(v,w)} \mathbf{V}_w^\top \mathbf{R}_w \vec{\mu} = \left(\mathbf{E}_{s_1} \mathbf{U}_{s_1} \underbrace{\mathbf{T}_{s_1}^\top}_{\text{FtS}} + \mathbf{E}_{s_2} \mathbf{U}_{s_2} \underbrace{\mathbf{T}_{s_2}^\top}_{\text{FtS}} \right) \cdot \mathbf{C}_{(v,w)} \cdot \left(\underbrace{\mathbf{T}_{t_1}}_{\text{StF}} \mathbf{V}_{t_1}^\top \mathbf{R}_{t_1} + \underbrace{\mathbf{T}_{t_2}}_{\text{StF}} \mathbf{V}_{t_2}^\top \mathbf{R}_{t_2} \right) \vec{\mu}.$$

We observe that restrict-to-cluster and multiplication with the column cluster basis as well as multiplication with the row cluster basis and expand-from-cluster can be done on the level of the sons. This has to be supplemented by son \rightarrow father (StF) and father \rightarrow son (FtS) transformations through the transfer matrices. Thus, by **recursion** all reduction and expansion operations can be pushed to the leaf level of the cluster trees.

The following algorithm does this for restrict-to-cluster and multiplication with the column cluster bases and implements the so-called **forward transformation**.

Pseudocode 2.4.6.22: Recursive transformation into column cluster bases

```

1 void forward_trf( $\mathcal{H}^2$ -matrix  $\mathbf{M}$ , cluster  $w \in \mathcal{T}_{\mathbb{J}}$ ,
2 vector  $\vec{\mu} \in \mathbb{R}^m$ , ref vectors  $(\vec{\omega}_w)_{w \in \mathcal{T}_{\mathbb{J}}}$ ) {
3   if (sons( $w$ ) ==  $\emptyset$ ) { // leaf cluster
4      $\vec{\omega}_w := \mathbf{V}_w^T \vec{\mu}|_{\mathcal{I}(w)}$ ;
5   }
6   else { // recurse into sons for father clusters
7     foreach  $t \in \text{sons}(w)$  {
8       forward_trf( $\mathbf{M}$ ,  $t$ ,  $\vec{\mu}$ ,  $(\vec{\omega}_w)_{w \in \mathcal{T}_{\mathbb{J}}}$ );
9        $\vec{\omega}_w += \mathbf{T}_t \vec{\omega}|_{\mathcal{I}(t)}$ ;
10    }
11  }
12 }
```

The **backward transformation** realizes the multiplication with the row cluster bases \mathbf{U}_v for each $v \in \mathcal{T}_{\mathbb{I}}$ and the subsequent expand-from-cluster operation:

Pseudocode 2.4.6.23: Recursive transformation into column cluster bases

```

1 void backward_trf( $\mathcal{H}^2$ -matrix  $\mathbf{M}$ , cluster  $v \in \mathcal{T}_{\mathbb{I}}$ ,
2 vectors  $(\vec{\zeta}_v)_{v \in \mathcal{T}_{\mathbb{I}}}$ , ref vector  $\vec{\rho} \in \mathbb{R}^n$ ) {
3   if (sons( $v$ ) ==  $\emptyset$ ) { // leaf cluster
4      $\vec{\rho}|_{\mathcal{I}(v)} += \mathbf{U}_v \vec{\zeta}_v$ ;
5   }
6   else { // recurse into sons for father clusters
7     foreach  $s \in \text{sons}(v)$  {
8        $\vec{\zeta}_s += \mathbf{T}_s^T \vec{\zeta}_v$ ;
9       backward_trf( $\mathbf{M}$ ,  $s$ ,  $(\vec{\zeta}_v)_{v \in \mathcal{T}_{\mathbb{I}}}$ ,  $\vec{\rho}$ );
10    }
11  }
12 }
```

The argument vector $\vec{\rho}$ used for returning the result has to be initialized with zero.

These two recursive functions are building blocks for a **3-pass** computation of $\mathbf{M}\vec{\mu}$ analogous to the algorithm from Section 2.3.5:

Pseudocode 2.4.6.24: Recursive transformation into column cluster bases

```

1 vector ← h2mv( $\mathcal{H}^2$ -matrix  $\mathbf{M}$ , vector  $\vec{\mu} \in \mathbb{R}^n$ ) {
2   vectors  $(\vec{\omega}_w)_{w \in \mathcal{T}_{\mathbb{J}}} := \mathbf{0}$ ; forward_trf( $\mathbf{M}$ ,  $\text{root}(\mathcal{T}_{\mathbb{J}})$ ,  $\vec{\mu}$ ,  $(\vec{\omega}_w)_w$ );
3   vectors  $(\vec{\zeta}_v)_{v \in \mathcal{T}_{\mathbb{I}}} = \mathbf{0}$ ;
4   // far-field blocks: multiplication with coupling matrices
5   foreach  $((v,w) \in \mathbb{F}_{\text{far}})$  {  $\vec{\zeta}_v += \mathbf{C}_{(v,w)} \vec{\omega}_w$ ; }
6   // near-field blocks: direct multiplication
7   foreach  $((v,w) \in \mathbb{F}_{\text{near}})$  {  $\vec{\zeta}_v += \mathbf{H}|_{v \times w} \vec{\omega}_w$ ; }
8   vector  $\vec{\rho} \in \mathbb{R}^n := \mathbf{0}$ ; backward_trf( $\mathbf{M}$ ,  $\text{root}(\mathcal{T}_{\mathbb{I}})$ ,  $(\vec{\zeta}_v)_{v \in \mathcal{T}_{\mathbb{I}}}$ ,  $\vec{\rho}$ );
9   return  $\vec{\rho}$ ;
10 }
```

Review question(s) 2.4.6.25.

1. We can define a class \mathcal{H}^* of $n \times m$ hierarchical matrices, $n, m \in \mathbb{N}$ by fixing the row and column cluster trees and the block partition \mathbb{F} . Outline an algorithm that, given $\mathbf{M} \in \mathbb{R}^{n,m}$ constructs

$$\mathbf{H}^* := \underset{\mathbf{H} \in \mathcal{H}^*}{\operatorname{argmin}} \|\mathbf{M} - \mathbf{H}\|_F.$$

△

┘

Bibliography

- [BK16] Jonas Ballani and Daniel Kressner. “Matrices with hierarchical low-rank structures”. In: *Exploiting hidden structure in matrix computations: algorithms and applications*. Vol. 2173. Lecture Notes in Math. Springer, Cham, 2016, pp. 161–209 (cit. on p. 157).
- [Beb08] M. Bebendorf. *Hierarchical Matrices: A Means to Efficiently Solve Elliptic Boundary Value Problems*. Vol. 63. Lecture Notes in Computational Science and Engineering (LNCSE). Springer-Verlag, 2008 (cit. on p. 179).
- [Bör21] S. Börm. *Numerical Methods for Non-Local Operators*. Lecture Notes. Universität Kiel, 2021 (cit. on pp. 198, 241, 252).
- [GV13] Gene H. Golub and Charles F. Van Loan. *Matrix computations*. Fourth. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, MD, 2013, pp. xiv+756 (cit. on p. 235).
- [GH03] L. Grasedyck and W. Hackbusch. “Construction and arithmetics of \mathcal{H} -matrices”. In: *Computing* 70 (2003), pp. 295–334 (cit. on pp. 219, 221).
- [Hac15] Wolfgang Hackbusch. *Hierarchical Matrices: Algorithms and Analysis*. Vol. 49. Springer Series in Computational Mathematics. Springer, Heidelberg, 2015, pp. xxv+511 (cit. on p. 252).
- [HM21] Helmut Harbrecht and Michael Multerer. *Samplets: A new paradigm for data compression*. 2021 (cit. on p. 198).
- [NS02] K. Nipp and D. Stoffer. *Lineare Algebra*. 5th ed. Zürich: vdf Hochschulverlag, 2002 (cit. on p. 169).
- [Str09] M. Struwe. *Analysis für Informatiker*. Lecture notes, ETH Zürich. 2009 (cit. on p. 181).

Chapter 3

Convolution Quadrature

This chapter studies a class of modern numerical methods for particular evolution problems, which are models with a particular direction of propagation, usually called time. In these models we can distinguish past and future and the latter must not have any influence on the former, a feature called **causality**. The mathematical description of many evolution models relies on **initial value problems** (IVP) for **ordinary differential equations** (ODEs), see [NumCSE Section 11.1]. They seek an for an unknown function $\mathbf{y} : I \subset \mathbb{R} \rightarrow V$ satisfying (the symbol $\dot{\cdot}$ stands for the derivative with respect to time t)

$$\dot{\mathbf{y}} = \mathbf{f}(t, \mathbf{y}) \quad , \quad \mathbf{y}(t_0) = \mathbf{y}_0 \quad , \quad (3.0.0.1)$$

with $\mathbf{y}_0 \in V$ and $\mathbf{f} : I \times V \rightarrow V$. Here, V is the **state space**, either $V = \mathbb{R}^d$, $d \in \mathbb{N}$, or a more general Banach space. The latter case also covers evolution problems for partial differential equations like parabolic initial boundary value problems [NumPDE Section 9.2] and wave equations [NumPDE Section 9.3]. In this case V will be a Sobolev space like $H^1(\Omega)$.

One may call (3.0.0.1) a “time-local” evolution, because the direction of evolution depends only on the current state. This is in contrast to **causal evolution problems with memory**, which will be our focus now. In these problems (the change of) the current state will be influenced by the entire past from some initial time. This will entail fundamentally new approaches to the construction of stable and efficient numerical integrators (timestepping schemes).

Contents

3.1	Basic Concepts and Tools	269
3.1.1	Convolution of Causal Functions	269
3.1.2	Discrete Convolutions	273
3.1.3	The Laplace Transform	276
3.1.4	Diagonalizing Convolutions	281
3.1.5	Toeplitz Matrix Numerical Linear Algebra	285
3.2	Convolution Equations: Examples	291
3.2.1	Tomography: Abel Integral Equation	291
3.2.2	Impedance Boundary Conditions	293
3.2.3	Time-Domain Boundary Integral Equations	296
3.3	Implicit-Euler Convolution Quadrature	299
3.3.1	Setting and Goal	299
3.3.2	Derivation of Implicit Euler CQ	301
3.3.3	Properties of implicit-Euler Convolution Quadrature	307
3.3.4	Convergence	310
3.4	Multistep Convolution Quadrature (MSCQ)	313
3.4.1	Linear Multi-Step Numerical Integrators	314
3.4.2	Multi-Step Convolution Quadrature: Weights	321

3.4.3	Multi-Step Convolution Quadrature: Algorithms	329
3.5	Runge-Kutta Convolution Quadrature (RKCQ)	334
3.5.1	Implicit Runge-Kutta Single-Step Methods	335
3.5.2	Runge-Kutta CQ weights	336
3.6	Fast and Oblivious Convolution Quadrature	339



Supplementary literature.

- [HS16]: A survey of convolution quadrature and its application to retarded potential time-domain integral equations with a focus on algorithms
- [Say16]: A comprehensive treatment of retarded potential time-domain integral equations for wave scattering problems. Convolution quadrature is discussed in several chapters.
- [BS21]: A complete treatment of convolution quadrature, theory and algorithms, for the solution of convolution-type evolution problems.
- [LS02; SLL65]: Fast & oblivious convolution quadrature, two variants.

3.1 Basic Concepts and Tools

3.1.1 Convolution of Causal Functions

Everybody knows what is meant by the pointwise multiplication of two (continuous) functions $\mathbb{R} \mapsto \mathbb{C}$, which yields another continuous function $\mathbb{R} \mapsto \mathbb{C}$. Now we will learn about another operation on pairs of functions $\mathbb{R} \mapsto \mathbb{C}$, which is as important, but rather obscure.

We introduce that operation as a fundamental binary operation on absolutely integrable functions $\mathbb{R} \rightarrow \mathbb{R}$, that are functions belonging to

$$L^1(\mathbb{R}) := \{f : \mathbb{R} \rightarrow \mathbb{R} \text{ integrable: } \|f\|_{L^1(\mathbb{R})} := \int_{\mathbb{R}} |f(x)| dx < \infty\}. \tag{3.1.1.1}$$

Definition 3.1.1.2. Convolution on the real line

Given two functions $f, g \in L^1(\mathbb{R})$, their **convolution** $f * g \in L^1(\mathbb{R})$ is defined as

$$(f * g)(t) := \int_{\mathbb{R}} f(t - \xi) g(\xi) d\xi = \int_{\mathbb{R}} f(\xi) g(t - \xi) d\xi, \quad t \in \mathbb{R}.$$

From the very definition we conclude that

$$* : L^1(\mathbb{R}) \times L^1(\mathbb{R}) \rightarrow L^1(\mathbb{R}) \text{ is continuous, bilinear, and symmetric.}$$

Here, “symmetric” means that $f * g = g * f$, which is a straightforward consequence of making the substitution $x_i' := t - \xi$. Continuity in $L^1(\mathbb{R})$ can be concluded by applying **Fubini’s theorem** [STRLN89]:

$$\begin{aligned} \|f * g\|_{L^1(\mathbb{R})} &= \int_{\mathbb{R}} \left| \int_{\mathbb{R}} f(t - \xi) g(\xi) d\xi \right| dt \\ &\leq \int_{\mathbb{R}} \int_{\mathbb{R}} |f(t - \xi)| |g(\xi)| d\xi dt \\ &\leq \int_{\mathbb{R}} |g(\xi)| \int_{\mathbb{R}} |f(t - \xi)| dt d\xi \leq \|f\|_{L^1(\mathbb{R})} \|g\|_{L^1(\mathbb{R})} \quad \forall f, g \in L^1(\mathbb{R}). \end{aligned}$$

Interchanging orders of integration (again possible owing to Fubini’s theorem) reveals another important property of convolution:

Corollary 3.1.1.3. Associativity of convolution

$$(f * g) * h = f * (g * h) \quad \forall f, g, h \in L^1(\mathbb{R})$$

EXAMPLE 3.1.1.4 (Some special convolutions)

- Convolution with the Heaviside function boils down to integration:

$$f(t) = \begin{cases} 1 & , \text{ if } t \geq 0, \\ 0 & , \text{ if } t < 0, \end{cases} \quad \blacktriangleright \quad (f * g)(t) = \int_{-\infty}^t g(\xi) \, d\xi, \quad t \in \mathbb{R}. \quad (3.1.1.5)$$

- Convolution reproduces (complex) exponentials:

$$(f * \{t \mapsto \exp(i\omega t)\}) = \exp(i\omega t) \cdot \int_{-\infty}^{\infty} f(\xi) \exp(-i\omega \xi) \, d\xi, \quad t \in \mathbb{R}. \quad (3.1.1.6)$$

Sloppily speaking, when considering the convolution with a fixed function f as a linear mapping $g \mapsto f * g$, then the exponentials $\{t \mapsto \exp(i\omega t)\}$ can be regarded as **eigenfunctions**. However, note that they do not belong to $L^1(\mathbb{R})$!

┘

Remark 3.1.1.7 (Convolution in $L^p(\mathbb{R})$ -spaces) As a generalization of $L^1(\mathbb{R})$, for $1 \leq p < \infty$ and an interval $I \subset \mathbb{R}$ we may consider the space of functions

$$L^p(I) := \{f : I \rightarrow \mathbb{R} \text{ integrable: } \|f\|_{L^p(I)}^p := \int_I |f(x)|^p \, dx < \infty\}. \quad (3.1.1.8)$$

This family is completed by $L^\infty(\mathbb{R})$ the space of essentially bounded functions equipped with a generalized supremum norm. All these spaces are Banach spaces. We can define the convolution on certain pairs of them.

Theorem 3.1.1.9. Young's inequality for convolutions [McL00, Thm. 3.1]

If $p, q, r \in [1, \infty]$ satisfy $p^{-1} + q^{-1} = 1 + r^{-1}$, then the convolution can be extended to a **continuous** mapping $* : L^p(\mathbb{R}) \times L^q(\mathbb{R}) \rightarrow L^r(\mathbb{R})$, in particular

$$\|f * g\|_{L^r(\mathbb{R})} \leq \|f\|_{L^p(\mathbb{R})} \cdot \|g\|_{L^q(\mathbb{R})} \quad \forall f \in L^p(\mathbb{R}), \quad g \in L^q(\mathbb{R}). \quad (3.1.1.10)$$

The case $p = r = \infty, q = 1$, furnishes pointwise estimates

$$(f * g)(t) \leq \|f\|_{L^1(\mathbb{R})} \cdot \|g\|_{L^\infty(\mathbb{R})} \quad \forall f \in L^1(\mathbb{R}), \quad g \in L^\infty(\mathbb{R}). \quad (3.1.1.11)$$

┘

Remark 3.1.1.12 (Convolution of distributions [Rud73, pp. 170]) Sloppily speaking, a distribution on \mathbb{R} is a linear functional on the space $C_0^\infty(\mathbb{R})$ of smooth compactly supported functions. The evaluation of a distribution ϕ for $g \in C_0^\infty(\mathbb{R})$ is usually written as a formal integral:

$$\phi(g) =: \langle \phi, g \rangle =: \int_{\mathbb{R}} \phi(\xi) g(\xi) \, d\xi, \quad \forall g \in C_0^\infty(\mathbb{R}).$$

In this sense, we can read the convolution of a distribution with a smooth compactly supported function $g \in C_0^\infty(\mathbb{R})$

$$(\phi * g)(t) := \int_{\mathbb{R}} \phi(t - \xi)g(\xi) d\xi = \int_{\mathbb{R}} \phi(\xi)g(t - \xi) d\xi := \langle \phi, \{\xi \mapsto g(t - \xi)\} \rangle, \quad t \in \mathbb{R}. \tag{3.1.1.13}$$

In some cases the resulting expression remains meaningful even for functions g of limited smoothness. One such case is the δ -distribution

$$\delta_x : C_0^\infty(\mathbb{R}) \rightarrow \mathbb{R}, \quad \delta_x(g) := g(x) \quad x \in \mathbb{R}, \tag{3.1.1.14}$$

for which convolution becomes a shift operation that makes sense for for very general functions

$$(\delta_x * g)(t) = g(t - x), \quad g \in L^\infty(\mathbb{R}). \tag{3.1.1.15}$$

┘

§3.1.1.16 (Convolutions of operators) In Def. 3.1.1.2, we considered the convolution of two real-valued functions. By componentwise consideration, we can instantly extend this to integrable matrix-valued and vector-valued functions

$$\begin{aligned} & \mathbf{F} : \mathbb{R} \rightarrow \mathbb{R}^{n,m}, \quad \mathbf{g} : \mathbb{R} \rightarrow \mathbb{R}^m. \\ \blacktriangleright \quad & (\mathbf{F} * \mathbf{g})(t) := \int_{\mathbb{R}} \mathbf{F}(t - \xi) \cdot \mathbf{g}(\xi) d\xi = \int_{\mathbb{R}} \mathbf{F}(\xi) \cdot \mathbf{g}(t - \xi) d\xi \in \mathbb{R}^n, \quad t \in \mathbb{R}. \end{aligned}$$

Here, \cdot designates the matrix \times vector product. Generalizations of the associativity property, Cor. 3.1.1.3, and 3.1.1.9 to this case are straightforward. Of course, this kind of convolution can no longer be commutative.

We can even go one step further and for Banach spaces X, Y consider the convolution with a one-parameter family of bounded linear operators represented by an integrable “linear-operator-valued” function $f : \mathbb{R} \rightarrow L(X, Y)$, $L(X, Y)$ the vector space of bounded linear mappings $X \rightarrow Y$:

$$f : \mathbb{R} \rightarrow L(X, Y), \quad g : \mathbb{R} \rightarrow X: \quad (f * g)(t) := \int_{\mathbb{R}} \underbrace{f(t - \xi)(g(\xi))}_{\in Y} d\xi \in Y, \quad t \in \mathbb{R}. \tag{3.1.1.17}$$

Of course, once operators are involved, convolutions can no longer be symmetric.

Read $f * g$ as “convolution operator f acting on g ”.

┘

§3.1.1.18 (Causal functions) Causal evolutions model processes that start at some point $t = 0$ in time. They can be described by functions on \mathbb{R} that vanish for $t < 0$. This gives time a direction.

Definition 3.1.1.19. Causal functions

For a vector space X an integrable function $f : \mathbb{R} \rightarrow X$ is called **causal**, if $f(t) = 0$ for almost all $t < 0$.

Note that causal functions are defined on the whole real line \mathbb{R} . Thus, a causal function g that is continuous will automatically satisfy $g(0) = 0$. If g is k -times continuously differentiable, then $g^{(\ell)}(0) = 0$ for all $0 \leq \ell \leq k$.

The convolution of two \mathbb{C} -valued causal functions takes a special form and yields another causal function

$$f, g \text{ causal} \Rightarrow (f * g)(t) = \int_0^t f(t - \xi)g(\xi) d\xi = \int_0^t f(\xi)g(t - \xi) d\xi, \quad t \geq 0, \quad (3.1.1.20)$$

$$\Rightarrow (f * g)(t) \text{ depends on } f|_{[0,t]}, g|_{[0,t]} \text{ only.} \quad (3.1.1.21)$$

Thus, in the causal case, Thm. 3.1.1.9 leads to the estimates

$$\|f * g\|_{L^r([0,T])} \leq \|f\|_{L^p([0,T])} \cdot \|g\|_{L^q([0,T])}, \quad (3.1.1.22)$$

if $p, q, r \in [1, \infty]$ satisfy the assumptions of Thm. 3.1.1.9: $p^{-1} + q^{-1} = 1 + r^{-1}$. ┘

Remark 3.1.1.23 (Signal-processing background) A function $f : \mathbb{R} \rightarrow \mathbb{R}$ can be regarded as a time-continuous, analog signal. Such a signal can be sent over a **causal, linear, time-invariant** channel, which, mathematically speaking, is a linear operator $T : L^\infty(\mathbb{R}) \rightarrow L^\infty(\mathbb{R})$ that

- ◆ maps causal functions to causal functions, that is, $g(t) = 0$ for $t < 0$ then $(Tg)(t) = 0$ for $t < 0$, too.
- ◆ satisfies

$$T(\{t \mapsto g(t - \xi)\})(t) = (Tg)(t - \xi), \quad \forall g \in L^\infty(\mathbb{R}), \quad \forall \xi \in \mathbb{R}. \quad (3.1.1.24)$$

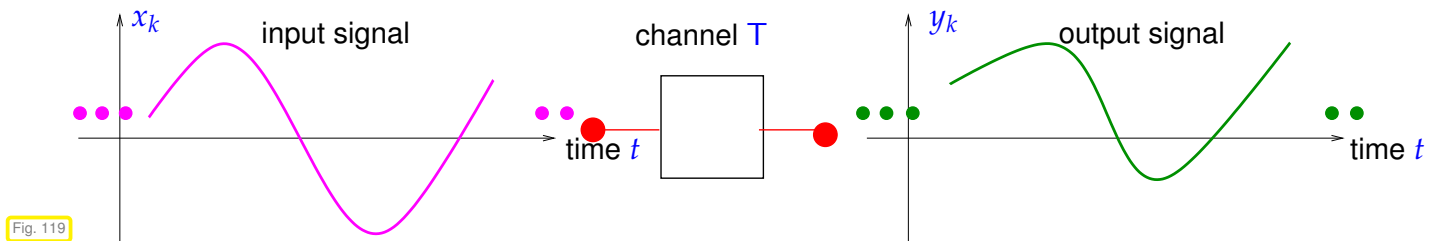


Fig. 119

Then there is a $f \in L^1(\mathbb{R})$ such that

$$Tg = f * g \quad \forall g \in L^\infty(\mathbb{R}).$$

The function f is called the **impulse response (function)** of T , because “ $f = T(\delta_0)$ ” hints that it can be obtained as output, when feeding the “impulse” δ_0 into the channel.

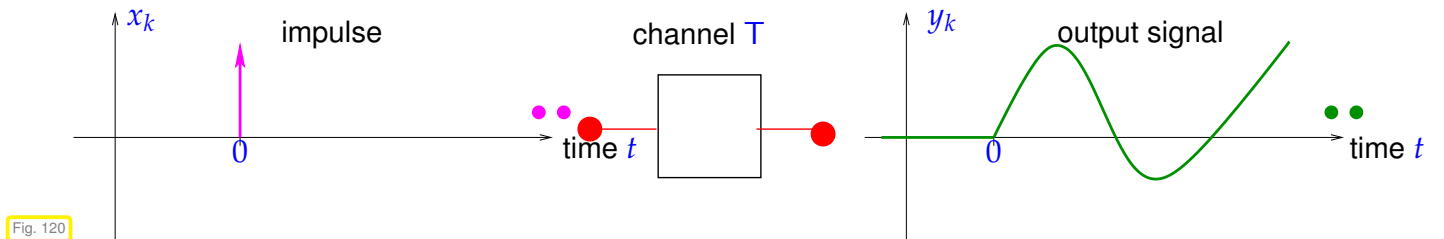


Fig. 120

§3.1.1.25 (Convolution equations (CEQs)) As in § 3.1.1.16, let X, Y be Banach spaces. Given a causal continuous function $y : \mathbb{R} \rightarrow Y$ and a causal operator-valued function $f : \mathbb{R} \rightarrow L(X, Y)$, we can state the **convolution equation**

$$(f * u)(t) = \int_0^t f(t - \xi)(u(\xi)) d\xi = y(t), \quad t \in \mathbb{R}, \quad (3.1.1.26)$$

for the unknown causal function $u : \mathbb{R} \rightarrow X$. At first glance this looks like a simpler form of the integral equations tackled in Chapter 1, but it is *fundamentally different* because it encodes a direction of propagation, since $u|_{[0,T]}$ should depend on $y|_{[0,T]}$ only (**causality!**). This is also reflected by the fact that the domain of integration depends on t unlike in the case of integral equations of the form

$$u : \Gamma \rightarrow \mathbb{R}: \int_{\Gamma} k(x, y) u(y) dS(y) = y(x), \quad y \in \Gamma.$$

This chapter will be dedicated to

1. approximating the convolution (3.1.1.20) of causal functions, given in a particular form, namely through their Laplace transform.
2. approximately solving convolution equations like (3.1.1.26).

Since both types of tasks address evolution problems, the methods will have the flavor of timestepping schemes. ┘

3.1.2 Discrete Convolutions

§3.1.2.1 (Sequences) We consider the **sampling** of a continuous function $f : \mathbb{R} \rightarrow X$, X a vector space, on an **equidistant lattice** with step size $\tau > 0$,

$$\mathcal{G}_{\tau} := \{t_{\ell} := \tau \cdot \ell\}_{\ell \in \mathbb{Z}}. \tag{3.1.2.2}$$

This yields a **sequence** $(f_{\ell}) : \mathbb{Z} \rightarrow X, f_{\ell} := f(t_{\ell}), \ell \in \mathbb{Z}$ “ $(f_{\ell}) = f|_{\mathcal{G}_{\tau}}$ ”.

Notation: We write (x_{ℓ}) for a sequence $\mathbb{Z} \rightarrow X$ with terms $x_{\ell} \in X$. Sometimes the index range will be restricted to a subset of \mathbb{Z} .

Replacing the improper integral in Def. 3.1.1.2 with a bi-infinite sum yields the convolution of real-valued absolutely summable sequences:

Definition 3.1.2.3. Convolution of sequences

If the sequences $(f_{\ell}), (g_{\ell}) : \mathbb{Z} \rightarrow \mathbb{C}$ satisfy $\sum_{\ell \in \mathbb{Z}} |f_{\ell}| < \infty$ and $\sum_{\ell \in \mathbb{Z}} |g_{\ell}| < \infty$, then

$$((f_{\ell}) * (g_{\ell}))_n := \sum_{\ell \in \mathbb{Z}} f_{n-\ell} \cdot g_{\ell} = \sum_{\ell \in \mathbb{Z}} f_{\ell} \cdot g_{n-\ell}, \quad n \in \mathbb{Z}. \tag{3.1.2.4}$$

defines another summable sequence $\mathbb{Z} \rightarrow \mathbb{R}$ the **discrete convolution** of (f_{ℓ}) and (g_{ℓ}) .

The discrete convolution operation enjoys similar properties as the convolution on \mathbb{R} :

Theorem 3.1.2.5. Properties of discrete convolution of sequences

The discrete convolution according to Def. 3.1.2.3 is a **symmetric, bilinear, and associative** mapping of the space $\ell^1(\mathbb{Z})$ of summable sequences into itself.

Young’s inequality of Thm. 3.1.1.9 also carries over:

$$\left(\sum_{n=-\infty}^{\infty} |((f_{\ell}) * (g_{\ell}))_n|^r \right)^{\frac{1}{r}} \leq \left(\sum_{n=-\infty}^{\infty} |f_{\ell}|^p \right)^{\frac{1}{p}} \cdot \left(\sum_{n=-\infty}^{\infty} |g_{\ell}|^q \right)^{\frac{1}{q}} \tag{3.1.2.6}$$

for $p, q, r \in [1, \infty]$ with $p^{-1} + q^{-1} = 1 + r^{-1}$ and for all sequences for which the right hand side of (3.1.2.6) is finite. If $p, r = \infty, q = 1$, the maximum modulus term of the sequence has to be picked. \lrcorner

Remark 3.1.2.7 (Sequences as distributions) Given a sequence $(f_\ell) \subset \mathbb{R}$, for $\tau > 0$ we can define the distribution

$$\varphi := \sum_{\ell=-\infty}^{\infty} f_\ell \delta_{\tau\ell}, \quad \delta_{\tau\ell} \hat{=} \delta\text{-distribution located at } \tau\ell, \text{ cf. (3.1.1.14).} \tag{3.1.2.8}$$

Based on

$$(\varphi * g)(t) := \int_{\mathbb{R}} \varphi(t - \xi)g(\xi) d\xi = \int_{\mathbb{R}} \varphi(\xi)g(t - \xi) d\xi := \langle \varphi, \{\xi \mapsto g(t - \xi)\} \rangle, \quad t \in \mathbb{R}, \tag{3.1.1.13}$$

we find that for $g \in C_0^\infty(\mathbb{R})$

$$\{t \mapsto (\varphi * g)(t)\} = \{t \mapsto \sum_{\ell=-\infty}^{\infty} f_\ell g(t - \tau\ell)\} \in C^\infty(\mathbb{R}). \tag{3.1.2.9}$$

A closer inspection shows that with φ given in (3.1.2.8)

$$(\varphi * g)|_{\mathcal{G}_\tau} = (f_\ell) * (g|_{\mathcal{G}_\tau}). \tag{3.1.2.10}$$

Beware: the $*$ on the left designated the convolution of (generalized) functions according to Def. 3.1.1.2, whereas the $*$ on the right means the convolution of sequences from Def. 3.1.2.3. \lrcorner

§3.1.2.11 (Causal sequences) If $f : \mathbb{R} \rightarrow X$ is causal, the sequence $(f_\ell) := f|_{\mathcal{G}_\tau}$ is **causal** in the sense that $f_\ell = 0$ for $\ell < 0$. In analogy to (3.1.1.20) the discrete convolution of two causal sequences yields another causal sequence according to

$$(f_\ell), (g_\ell) \text{ causal} \Rightarrow ((f_\ell) * (g_\ell))_n = \sum_{\ell=0}^n f_{n-\ell} \cdot g_\ell = \sum_{\ell=0}^n f_\ell \cdot g_{n-\ell}, \quad n \in \mathbb{N}_0. \tag{3.1.2.12}$$

Causal sequences are a powerful abstraction: In a signal-processing context a causal sequence represents a **time-discrete** analog signal, recall [NumCSE § 4.0.0.1]. Regarding the causal sequence (g_ℓ) as input the convolution $(f_\ell) * (g_\ell)$ represents the output of a time-invariant, linear, causal **filter** with **impulse response** (f_ℓ) : the impulse g_ℓ at $t = t_\ell$ triggers the response $(f_{k-\ell})_{k \geq \ell}$ and the output signal results from the linear superposition of all these responses. More details are given in [NumCSE Section 4.1]. \lrcorner

§3.1.2.13 (Operator-valued sequences) The generalization pursued in § 3.1.1.16 can also be pursued for causal sequences. For normed vector spaces X, Y let $(f_\ell) \subset L(X, Y)$ stand for a causal sequence of bounded linear operators $X \rightarrow Y$, and $(g_\ell) \subset X$ for a causal sequence in X . The natural way to extend the discrete convolution to these sequences is

$$((f_\ell) * (g_\ell))_n := \sum_{\ell \in \mathbb{Z}} f_{n-\ell}(g_\ell) = \sum_{\ell \in \mathbb{Z}} f_\ell(g_{n-\ell}) \in Y, \quad n \in \mathbb{Z}. \tag{3.1.2.14}$$

This defines a causal sequence in Y .

A **discrete convolution equation** for causal sequences has the simple structure of an infinite **triangular** linear system of operator equations. If $(f_\ell) \subset L(X, Y), (y_\ell) \subset Y$ are causal sequences, then

$$(f_\ell) * (u_\ell) = (y_\ell) \Leftrightarrow \begin{bmatrix} f_0 & 0 & \dots & \dots & \dots & \dots \\ f_1 & f_0 & 0 & \dots & & \\ f_2 & f_1 & f_0 & 0 & \dots & \\ \vdots & \ddots & \ddots & \ddots & \ddots & \\ \vdots & & \ddots & \ddots & \ddots & \ddots \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ \vdots \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ \vdots \end{bmatrix}. \tag{3.1.2.15}$$

If and only if f_0 is invertible, this operator equation can be solved recursively similar to the forward elimination step in Gaussian elimination:

$$f_0 u_n = y_n - \sum_{\ell=0}^{n-1} f_{n-1-\ell} u_\ell, \quad n \in \mathbb{N}_0. \tag{3.1.2.16}$$

This simple scheme is also known as **marching on in time (MOT)** algorithm in the area of timestepping methods for evolution problems. ┘

Our goal will be the discretization of the convolution of causal functions through replacement by a discrete convolution: for causal $f : \mathbb{R} \rightarrow L(X, Y)$, $g : \mathbb{R} \rightarrow X$ we seek τ -dependent sequences $(w_\ell^{f, \tau})$ of **convolution weights** such that

$$(f * g)|_{\mathcal{G}_\tau} \approx (w_\ell^{f, \tau}) * g|_{\mathcal{G}_\tau} \tag{3.1.2.17}$$

$$\begin{aligned} &\Downarrow \\ \int_0^{n\tau} f(t_n - \xi) (g(\xi)) \, d\xi &\approx \sum_{\ell=0}^n w_{n-\ell}^{f, \tau} g(\ell\tau), \end{aligned} \tag{3.1.2.18}$$

where “ \approx ” should be read as “convergence in a suitable norm for $\tau \rightarrow 0$ ”. The origin of the name **convolution quadrature** for this approximation is clear, because (3.1.2.18) can be regarded as the approximation of an integral value by a weighted sum, similar to a quadrature formula as defined in Def. 1.4.3.41.

We consider the approximation problem for $f * g$ in a particular setting: the function f may have awkward properties or not be available at all. Instead, its Laplace transform may be simple and known and we should rely on it to determine the convolution weights $w_\ell^{f, \tau}$.

3.1.3 The Laplace Transform

The exponentials $e_s : t \mapsto \exp(st)$ have the unique property that they are “eigenfunctions” of both the differentiation operator $\frac{d}{dt} : C^\infty(\mathbb{R}) \rightarrow C^\infty(\mathbb{R})$ and the translations $g \mapsto g(\cdot - \tau)$, $\tau \in \mathbb{R}$:

$$\frac{d}{dt} \{t \mapsto \exp(st)\} = s \{t \mapsto \exp(st)\}, \quad \exp(s(t - \tau)) = \exp(-s\tau) \exp(st).$$

So exponentials are the right building blocks for function spaces to use, when dealing with (linear) equations involving differentiation and translations. The latter play a prominent role in convolutions.

§3.1.3.1 (Fourier transform on \mathbb{R}) Considering the exponentials on the entire real line \mathbb{R} and demanding that they or their L^p -norms are bounded, leaves $s := i\omega$, $\omega \in \mathbb{R}$ as the only option, which leads to the famous **Fourier transform**

$$\mathcal{F} : L^1(\mathbb{R}) \rightarrow L^\infty(\mathbb{R}), \quad \mathcal{F}f(\omega) := \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} f(t) \exp(-i\omega t) \, dt. \tag{3.1.3.2}$$

By the **Plancherel theorem** \mathcal{F} gives rise to an isometric isomorphism of $L^2(\mathbb{R})$:

$$\|f\|_{L^2(\mathbb{R})} = \|\mathcal{F}f\|_{L^2(\mathbb{R})} \quad \forall f \in L^2(\mathbb{R}), \tag{3.1.3.3}$$

which means

$$\int_{\mathbb{R}} |f(t)|^2 \, dt = \int_{\mathbb{R}} |\mathcal{F}f(\omega)|^2 \, d\omega.$$

Thus the Fourier transform is invertible on $L^2(\mathbb{R})$ and for $\mathcal{F}(f) \in L^2(\mathbb{R}) \cap L^1(\mathbb{R})$ we have the **inversion formula**

$$f(t) = \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} \mathcal{F}f(\omega) \exp(i\omega t) d\omega . \tag{3.1.3.4}$$

Morally speaking, by means of the Fourier transform, a function $f : \mathbb{R} \rightarrow \mathbb{C}$ can be broken down into a superposition of exponentials $\{t \mapsto \exp(i\omega t)\}$. ┘

On the half real line \mathbb{R}_0^+ a much larger family of exponentials does not blow up: $\{t \mapsto \exp(st)\}$ for $\text{Re}(s) \leq 0$. This gives much more freedom for writing functions as a superposition of exponentials.

Definition 3.1.3.5. Causal polynomially bounded functions

For a vector space X denote by $\mathcal{CF}(X)$ the space of causal (\rightarrow Def. 3.1.1.19) integrable functions $\mathbb{R} \rightarrow X$ satisfying a polynomial growth bound:

$$\forall f \in \mathcal{CF}(X): \exists M > 0, m \in \mathbb{N}: \|f(t)\|_X \leq M(1 + |t|)^m \quad \forall t \in \mathbb{R} .$$

In the case $X = \mathbb{R}$ the space $\mathcal{CF}(X)$ is closed under convolution: $f, g \in \mathcal{CF}(X) \Rightarrow f * g \in \mathcal{CF}(X)$, because the convolution of two polynomial causal functions is another polynomial causal function.

Definition 3.1.3.6. Laplace transform

For $f \in \mathcal{CF}(X)$, its **Laplace transform** $\mathcal{L}f$ is an X -valued function on the right half plane $\mathbb{C}^+ := \{z \in \mathbb{C} : \text{Re}(z) > 0\}$ defined as

$$\mathcal{L}f(s) := \int_0^\infty f(t) e^{-st} dt, \quad s \in \mathbb{C}^+ .$$

The improper integral is well defined because

$$\|f(t)e^{-st}\|_X \leq \|f(t)\|_X \exp(-\text{Re}(s)t) \stackrel{\text{Def. 3.1.3.5}}{\leq} M(1 + t)^m \exp(-\text{Re}(s)t) .$$

The bound on the right-hand side is an integrable function of t for any $\text{Re}(s) > 0$.

The next theorem is proved by differentiation under the integral and a limit argument (Weierstrass theorem) to deal with the improper integral.

Theorem 3.1.3.7. Analyticity of Laplace transforms

For every $f \in \mathcal{CF}(X)$ its Laplace transform $\mathcal{L}f$ is an analytic/holomorphic function (\rightarrow Def. 1.4.3.68) on \mathbb{C}^+ .

Analytic functions initially defined on open subsets of \mathbb{C} possess an intrinsic extension to a maximal domain of definition. This also applies to Laplace transforms.

EXAMPLE 3.1.3.8 (Laplace transform of causal power function) We consider the causal power function

$$f(t) := t_+^q := \begin{cases} t^q & \text{for } t \geq 0, \\ 0 & \text{for } t < 0 \end{cases}, \quad q > -1 .$$

(The constraint $q > -1$ is meant to ensure integrability, because for $q < 0$ the function has a singularity at $t = 0$.) This function belongs to $\mathcal{CF}(\mathbb{R})$.

We directly compute the Laplace transform, first for $s > 0$,

$$\begin{aligned} \mathcal{L}f(s) &= \int_0^\infty t^q e^{-st} dt = \int_0^\infty \left(\frac{\eta}{s}\right)^q e^{-\eta} s^{-1} d\eta \quad [\text{Subst. } \eta := st] \\ &= \frac{1}{s^{q+1}} \int_0^\infty \eta^q e^{-\eta} d\eta = \frac{\Gamma(q+1)}{s^{q+1}}, \end{aligned}$$

where Γ stands for the **Gamma function**, which interpolates the factorials: $\Gamma(n) = (n-1)!$ for all $n \in \mathbb{N}$.

Next, we consider analytic continuation beyond \mathbb{R}^+ :

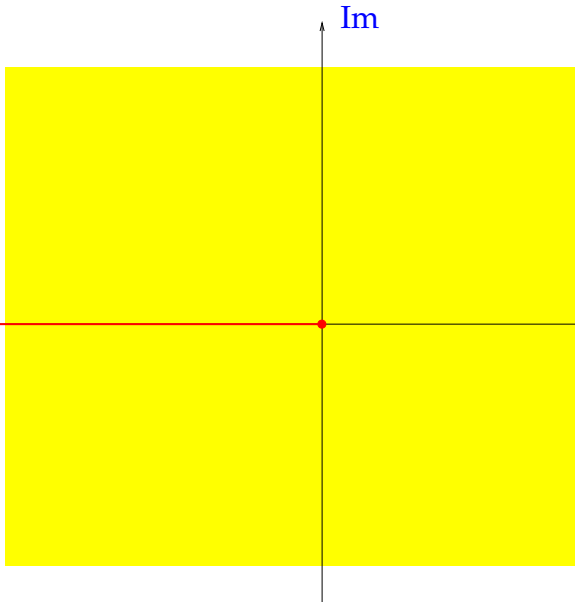


Fig. 121

We can rewrite $s^{-(q+1)} = \exp(-(q+1) \log(s))$ to find the maximal domain of analyticity of $\mathcal{L}f$: The (main branch of the) complex logarithm is analytic in $\mathbb{C} \setminus [-\infty, 0]$, which also yields the domain of analyticity of $\mathcal{L}f$:

$$\mathcal{L}\{t \mapsto t_+^q\} \text{ is analytic in } \mathbb{C} \setminus [-\infty, 0]$$

Obviously this domain of analyticity extends far beyond \mathbb{C}^+ . This is a common phenomenon with Laplace transforms: Though defined only on \mathbb{C}^+ by means of the integral formula, they often permit an **analytic extension** into large subdomains of $\mathbb{C}^- := \{z \in \mathbb{C} : \text{Re}(z) < 0\}$. The extension is then understood as $\mathcal{L}f$.

Remark 3.1.3.9 (Complex contour integrals) In complex analysis you have seen complex **contour integrals**, the integral of a function $f : D \subset \mathbb{C} \rightarrow \mathbb{C}$ along a C_{pw}^1 -curve $\Gamma \subset D$, given by a parameterization $\gamma : I \subset \mathbb{R} \rightarrow \mathbb{C}, \Gamma := \gamma(I), I$ an interval:

$$\int_\Gamma f(z) dz := \int_I f(\gamma(\xi)) \cdot \dot{\gamma}(\xi) d\xi, \tag{3.1.3.10}$$

where \cdot is multiplication in \mathbb{C} , and $\dot{\gamma}$ is the derivative with respect to the parameter.

For example, the unit circle $S^1 \subset \mathbb{C}$ around 0 viewed as an oriented closed curve has the parameterization $\xi \mapsto \exp(2\pi i \xi), \xi \in I := [0, 1]$. Hence, the contour integral of a \mathbb{C} -valued function f defined in a neighborhood of S^1 can be computed via

$$\int_{S^1} f(z) dz = 2\pi i \int_0^1 f(\exp(2\pi i \xi)) \exp(2\pi i \xi) d\xi.$$

§3.1.3.11 (Laplace inversion formula) Restricting the Laplace transform to a line parallel to the imaginary axis reveals a close connection with the Fourier transform on \mathbb{R} addressed in § 3.1.3.1. For an integrable causal function $f \in \mathcal{CF}(X)$ we formally compute

$$s = \sigma + i\omega \Rightarrow \mathcal{L}f(s) = \int_{-\infty}^\infty f(t) e^{-(\sigma+i\omega)t} dt = \sqrt{2\pi} \mathcal{F}(\{t \mapsto e^{-\sigma t} f(t)\})(\omega), \tag{3.1.3.12}$$

where we have split $s \in \mathbb{C}^+$ into real and imaginary part: $s = \sigma + i\omega$, $\sigma \in \mathbb{R}^+$, $\omega \in \mathbb{R}$. Apply the Fourier inversion formula

$$(\mathcal{F}^{-1}G)(t) := \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} G(\omega) \exp(i\omega t) d\omega, \quad t \in \mathbb{R}, \tag{3.1.3.4}$$

to (3.1.3.12) to obtain

$$\sqrt{2\pi}e^{-\sigma t}f(t) = \mathcal{F}^{-1}\{\omega \mapsto (\mathcal{L}f)(\sigma + i\omega)\}(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \mathcal{L}f(\sigma + i\omega) \exp(i\omega t) d\omega.$$

Multiply with $\exp(\sigma t)$ and recall the tool of complex contour integration from Rem. 3.1.3.9, which permits us to rewrite

$$\int_{-\infty}^{\infty} \mathcal{L}f(\sigma + i\omega) \exp((\sigma + i\omega)t) d\omega = \frac{1}{i} \int_{\sigma+i\mathbb{R}} \mathcal{L}f(s) \exp(st) ds,$$

where $\sigma + i\mathbb{R}$ is a “curve” in \mathbb{C} , a line parallel to the imaginary axis, for which we have used the natural parameterization $\omega \rightarrow \sigma + i\omega$.

Theorem 3.1.3.13. Inverse Laplace transform

If $F : \mathbb{C}^+ \rightarrow X$ is analytic in \mathbb{C}^+ and satisfies the decay condition

$$\|F(s)\|_X \leq |s|^\mu \quad \text{for } \mu < -1, \tag{3.1.3.14}$$

then, for any $\sigma > 0$, F is the Laplace transform of the causal function given by the improper contour integral (**Bromwich integral**)

$$(\mathcal{L}^{-1}F)(t) := \frac{1}{2\pi i} \int_{\sigma+i\mathbb{R}} F(s) \exp(st) ds, \quad t \in \mathbb{R} \tag{3.1.3.15}$$

The decay of $s \mapsto F(s)$ stipulated by (3.3.1.2) guarantees the existence of the improper integral. By the **Cauchy integral theorem** that we recall next

- the value of the contour integral does not depend on $\sigma > 0$, and
- the function from (3.1.3.15) is causal.

Theorem 3.1.3.16. Cauchy integral theorem

Let $D \subset \mathbb{C}$ be open and simply connected. If $f : D \rightarrow \mathbb{C}$ is analytic on D and $\Gamma \subset D$ is a **closed** C^1_{pw} -curve then the contour integral (\rightarrow Rem. 3.1.3.9) of f over Γ vanishes

$$\int_{\Gamma} f(z) dz = 0.$$

┘

§3.1.3.17 (Vanishing Bromwich integrals) Now, for $\sigma > 0$ and $R > 0$ consider the contour

$$\Gamma_R := \{\sigma + i[-R, R]\} \cup \{|z - \sigma| = R, \operatorname{Re}(z) > \sigma\}, \tag{3.1.3.18}$$

marked in color in Fig. 122. We also assume that $f \in \mathcal{CF}(X)$ is such that its Laplace transform $\mathcal{L}f$ is polynomially bounded,

$$\exists C > 0, m \in \mathbb{N}: |\mathcal{L}f(s)| \leq C|s|^m \quad \forall s \in \mathbb{C}^+. \tag{3.1.3.19}$$

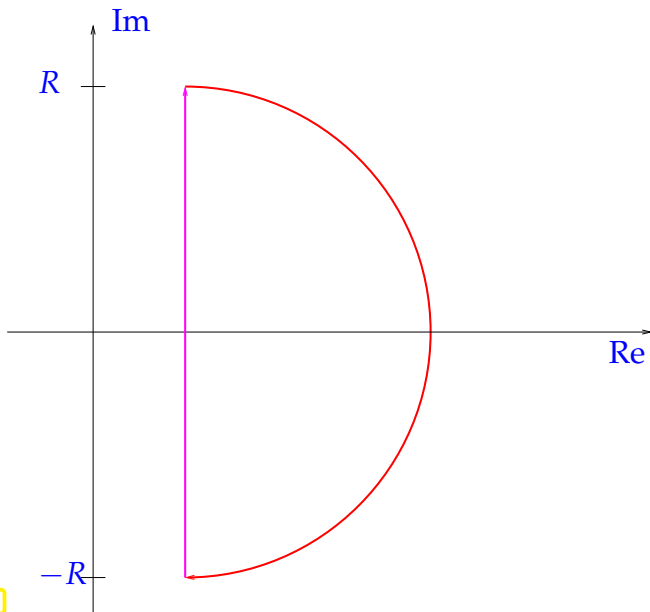


Fig. 122

To begin with, Thm. 3.1.3.7 combined with by the Cauchy integral theorem imply

$$\int_{\Gamma_R} F(s) e^{st} ds = 0.$$

If $t < 0$, then

$$|e^{st}| = e^{t \operatorname{Re} s} \leq 1 \quad \forall s \in \mathbb{C}^+. \tag{3.1.3.20}$$

This identity together with (3.1.3.19) also gives us the bound

$$\left| \int_{\substack{|z-\sigma|=R \\ \operatorname{Re}(z) > \sigma}} \mathcal{L}f(s) e^{st} ds \right| \leq C \int_{\substack{|z-\sigma|=R \\ \operatorname{Re}(z) > \sigma}} |s|^m e^{t \operatorname{Re} s} ds$$

We parameterize the half circle as $\varphi \in [-\frac{\pi}{2}, \frac{\pi}{2}] \mapsto \begin{bmatrix} \sigma + R \cos \varphi \\ R \sin \varphi \end{bmatrix}$, which permits us to express the previous bound as

$$\int_{\substack{|z-\sigma|=R \\ \operatorname{Re}(z) > \sigma}} \mathcal{L}f(s) e^{st} ds \leq C(r + \sigma)^m \int_{-\pi/2}^{\pi/2} e^{t(\sigma + R \cos \varphi)} d\varphi \rightarrow 0 \quad \text{for } R \rightarrow \infty$$

by **Lebesgue's dominated convergence theorem**. Hence also the integral of $\mathcal{L}f$ over $\sigma + i\mathbb{R}$ has to vanish. \lrcorner

§3.1.3.21 (Differentiation in Laplace domain) Now we will reap a first fruit of the fact that exponentials are “eigenfunctions” of the differentiation operator.

Theorem 3.1.3.22. Differentiation formula for Laplace transform

For a causal continuously differentiable function $f \in \mathcal{CF}(X) \cap C^1(\mathbb{R})$ (“of time”)

$$\mathcal{L}\dot{f}(s) = s \cdot \mathcal{L}f(s), \quad s \in \mathbb{C}^+,$$

where \dot{f} is the (temporal) derivative of f .

The proof is straightforward integration by parts. We mention two consequences of this theorem.

- 1 The Laplace transform converts linear ordinary differential equations (ODEs) with constant coefficients into algebraic equations in the “Laplace domain”

For the initial value problem for a second-order ODE,

$$\ddot{y}(t) - a^2 y(t) = c(t), \quad a \in \mathbb{R}, \quad y(0) = \dot{y}(0) = 0,$$

set $Y(s) := \mathcal{L}y(s)$ and obtain

$$s^2 Y(s) = a^2 Y(s) + \mathcal{L}c(s) \quad \blacktriangleright \quad Y(s) = \frac{\mathcal{L}c(s)}{s^2 - a^2}.$$

- Thm. 3.1.3.22 makes it possible to extend the Laplace inversion formula to functions violating the decay condition (3.3.1.2), see also [Say16, Prop. 3.1.2]

Let $F : \mathbb{C}^+ \rightarrow X$ be analytic and comply with the power-law growth bound

$$\exists \mu \in \mathbb{R}, M > 0: \|F(s)\|_X \leq M|s|^\mu . \tag{3.1.3.23}$$

Then, by Thm. 3.1.3.13, for $m \in \mathbb{N}, m > \mu + 1$, the function

$$f_m(t) := \frac{1}{2\pi i} \int_{\sigma+i\mathbb{R}} s^{-m} F(s) e^{st} ds, \quad t \in \mathbb{R},$$

is causal and its Laplace transform satisfies

$$s^m \cdot \mathcal{L}f_m(s) = F(s) \quad \forall s \in \mathbb{C}^+ .$$

Then, we can invoke Thm. 3.1.3.22 and find

$$\mathcal{L}(f) = F \quad \text{for } f := \frac{d^m f_m}{dt^m} \text{ defined in the sense of distributions,}$$

□

3.1.4 Diagonalizing Convolutions

We started Section 3.1.3 by pointing out that exponentials $e_s : t \mapsto \exp(st)$ are “eigenfunctions” of every translation operator in $L^\infty(\mathbb{R})$. Note that convolution

$$(f * g)(t) := \int_{\mathbb{R}} f(\xi) g(t - \xi) d\xi, \quad t \in \mathbb{R},$$

seen as an operator $g \mapsto f * g$ is essentially a superposition of translations of g . Hence, it comes as no surprise that exponentials will also be “eigenfunctions” of this convolution operator: For $f \in L^1(\mathbb{R})$

$$f * e_{i\omega} = e_{i\omega} \cdot \int_{\mathbb{R}} f(\xi) \exp(-i\omega\xi) d\xi = e_{i\omega} \cdot (\mathcal{F}f)(\omega), \quad \omega \in \mathbb{R}, \tag{3.1.4.1}$$

where \mathcal{F} is the Fourier transform on \mathbb{R} . This immediately leads to the famous convolution theorem for the Fourier transform.

Theorem 3.1.4.2. Convolution theorem for Fourier transform

For all $f, g \in L^1(\mathbb{R})$: $\mathcal{F}(f * g) = \mathcal{F}f \cdot \mathcal{F}g$ pointwise on \mathbb{R} .

Proof. (formal) Appeal to the inverse Fourier transform and boldly exchange convolution and integration:

$$\begin{aligned} (f * g)(t) &= (f * \int_{\mathbb{R}} (\mathcal{F}g)(\omega) e_{i\omega}(\cdot) d\omega)(t) = \int_{\mathbb{R}} (\mathcal{F}g)(\omega) (f * e_{i\omega})(t) d\omega \\ &= \int_{\mathbb{R}} (\mathcal{F}g)(\omega) (\mathcal{F}f)(\omega) e_{i\omega}(t) d\omega \quad \text{by (3.1.4.1)}. \end{aligned}$$

□

Demanding that f is causal (\rightarrow Def. 3.1.1.19) we can admit $s \in \mathbb{C}^+$ in the above reasoning, which gives us a similar result for the Laplace transform. Again, relying on formal computations for causal $f, g \in \mathcal{CF}(\mathbb{C})$

and (3.1.1.20)

$$\begin{aligned} \mathcal{L}(f * g)(s) &= \int_0^\infty \int_0^t f(\zeta)g(t - \zeta) \, d\zeta \exp(-st) \, dt = \int_0^\infty \left(\int_\zeta^\infty \exp(-st)f(\zeta)g(t - \zeta) \, dt \right) d\zeta \\ &= \int_0^\infty \exp(-s\zeta)f(\zeta) \cdot \int_\zeta^\infty \exp(-s(t - \zeta))g(t - \zeta) \, dt \, d\zeta = (\mathcal{L}f)(s) \cdot (\mathcal{L}g)(s), \quad s \in \mathbb{C}^+. \end{aligned}$$

The next theorem restates this result in the more general context of vector-valued/operator-valued causal functions, cf. § 3.1.1.16.

Theorem 3.1.4.3. Convolution theorem for Laplace transform

For Banach spaces X, Y and $g \in \mathcal{CF}(X), f \in \mathcal{CF}(L(X, Y))$ holds

$$\mathcal{L}(f * g)(s) = ((\mathcal{L}f)(s))((\mathcal{L}g)(s)), \quad s \in \mathbb{C}^+.$$

§3.1.4.4 (Operational calculus) Convolution with $f \in \mathcal{CF}(L(X, Y))$ is now regarded as a family of linear mappings $\mathcal{CF}(X) \rightarrow \mathcal{CF}(Y), g \mapsto f * g$ parameterized by f . If the Laplace transform $F(s) := \mathcal{L}f(s)$ is more easily accessible than f itself, we can also use F as “parameter”. This leads to the “operational calculus” view of convolution, introduced by Ch. Lubich [Lub88].

Definition 3.1.4.5. Operational calculus

For $F : \mathbb{C}^+ \rightarrow \mathbb{C}$ analytic we define the linear operator

$$F(\partial_t) : \begin{cases} \mathcal{CF}(X) \cap C_0^\infty(\mathbb{R}, X) & \rightarrow \mathcal{CF}(Y) \cap C^\infty(\mathbb{R}, Y) \\ g & \mapsto F(\partial_t)g := \mathcal{L}^{-1}(\{s \mapsto F(s) \cdot \mathcal{L}g(s)\}) \end{cases},$$

induced by the **transfer function** F .

Equivalently, we can write

$$\text{Def. 3.1.4.5} \quad \Rightarrow \quad \begin{cases} (\mathcal{L}F(\partial_t)g)(s) = F(s) \cdot \mathcal{L}g(s), \quad s \in \mathbb{C}^+, & (3.1.4.6) \\ F(\partial_t)g = \mathcal{L}^{-1}F * g, & (3.1.4.7) \end{cases}$$

where the last identity is a consequence of Thm. 3.1.4.3. Hence, operational calculus is another way to encode causal convolution with emphasis on the Laplace transform of one factor.

Operational calculus can also be viewed as a generalization of differentiation, because Thm. 3.1.3.22 implies for $m \in \mathbb{N}_0$

$$F(s) = s^m \quad \Rightarrow \quad F(\partial_t)g(t) = \frac{d^m g}{dt^m}(t) \quad t \in \mathbb{R}. \quad (3.1.4.8)$$

Already these formulas hint that the restriction to $g \in \mathcal{CF}(X) \cap C^\infty(\mathbb{R}, X)$ in Def. 3.1.4.5 is not necessary. If the growth of F admits a polynomial bound, argument functions of finite smoothness can be accommodated.

Lemma 3.1.4.9. Pointwise estimate for convolution

Assume that $F : \mathbb{C}^+ \rightarrow \mathbb{C}$ is analytic and satisfies the power law growth bound

$$\exists \mu \in \mathbb{R}, M > 0: |F(s)| \leq M|s|^\mu \quad \forall s \in \mathbb{C}^+, \quad (3.1.4.10)$$

Then, for every $m \in \mathbb{N}, m > \mu + 1$, there holds the pointwise estimate

$$\|(F(\partial_t)g)(t)\|_X \leq \int_0^t e^{-\sigma\zeta} \|g^{(m)}(\zeta)\|_X d\zeta \cdot \frac{e^{\sigma t}}{2\pi} \int_{\sigma+i\mathbb{R}} \frac{|F(s)|}{|s|^m} ds, \quad t \in \mathbb{R}, \quad (3.1.4.11)$$

for all causal $g \in \mathcal{CF}(X)$ for which the right-hand side is finite.

Proof. For the sake of simplicity, consider $X = \mathbb{C}$. Thm. 3.1.3.22 gives the identity ($\sigma > 0$)

$$F(\partial_t)g(t) = \frac{1}{2\pi i} \int_{\sigma+i\mathbb{R}} \left(\frac{F(s)}{s^m}\right) (s^m \mathcal{L}g(s)) e^{st} ds = \frac{1}{2\pi i} \int_{\sigma+i\mathbb{R}} \left(\frac{F(s)}{s^m}\right) \mathcal{L}g^{(m)}(s) e^{st} ds.$$

The assertion of the lemma follows from the estimate

$$|\mathcal{L}g^{(m)}(s)| = \left| \int_0^\infty g^{(m)}(t) e^{-st} dt \right| \leq \int_0^\infty e^{-\sigma t} |g^{(m)}(t)| dt, \quad s \in \sigma + i\mathbb{R},$$

and the fact that $F(\partial_t)g(t)$ depends on $g|_{[0,t]}$ only. □

Knowing the growth of $F(s)$, the right-hand side of (3.1.4.11) can be estimated further, which yields the following refined bound after elementary but tedious calculus.

Theorem 3.1.4.12. Pointwise estimate for convolution II [Say16, Prop. 3.2.2]

Assume that

- ◆ the operator-valued function $H : \mathbb{C}^+ \rightarrow L(X, Y)$, X, Y Banach spaces, is analytic, and
- ◆ satisfies the power law growth bound

$$\exists \mu \geq 0, m \in \mathbb{N}, M > 0: \|H(s)\| \leq M \max\{1, (\operatorname{Re} s)^{-m}\} |s|^\mu \quad \forall s \in \mathbb{C}^+, \quad (3.1.4.13)$$

- ◆ and that the causal X -valued function $g \in \mathcal{CF}(X)$ belongs to $\mathcal{C}^n(\mathbb{R}, X)$ for some $n \in \mathbb{N}, n > \mu + 1$, and
- ◆ that its n -th derivative $g^{(n)}$ is integrable on \mathbb{R} .

Then we can estimate

$$\|H(\partial_t)g(t)\|_Y \leq M2^\mu \frac{1 + \delta t^\delta \max\{1, t^m\}}{\pi\delta} \int_0^t \left\| \sum_{\ell=0}^n \binom{k}{\ell} g^{(\ell)}(\tau) \right\|_X d\tau,$$

with $\delta := n - (\mu + 1)$. ┘

§3.1.4.14 (Diagonalizing discrete convolutions) We return to the convolution of (scalar) sequences according to Def. 3.1.2.3,

$$((f_\ell) * (g_\ell))_n := \sum_{\ell \in \mathbb{Z}} f_{n-\ell} \cdot g_\ell = \sum_{\ell \in \mathbb{Z}} f_\ell \cdot g_{n-\ell}, \quad n \in \mathbb{Z}. \quad (3.1.2.4)$$

for sequences $(f_\ell), (g_\ell) : \mathbb{Z} \rightarrow \mathbb{C}$. Regard (3.1.2.4) as “sequence (f_ℓ) acting on sequence (g_ℓ) ”. What are “eigensequences” of this linear mapping?

Inspired by the central role of the exponential function in the diagonalization of the convolution of functions we try

$$g_\ell := g_\ell^z := \exp(-s\ell) = z^{-\ell}, \quad \ell \in \mathbb{Z} \quad \text{with} \quad z := e^s, \quad s \in \mathbb{C} : \tag{3.1.4.15}$$

$$((f_\ell) * (g_\ell^z))_n := \sum_{\ell \in \mathbb{Z}} f_\ell \cdot g_{n-\ell}^z = \sum_{\ell \in \mathbb{Z}} f_\ell \cdot z^{-n+\ell} = \underbrace{z^{-n}}_{=g_n^z} \cdot \underbrace{\sum_{\ell \in \mathbb{Z}} f_\ell z^\ell}_{\text{“eigenvalue”}} \quad \forall n \in \mathbb{Z}. \tag{3.1.4.16}$$

This formal computation reveals that $\ell \mapsto z^{-\ell}$ is an “eigensequence” with associated “eigenvalue” $\sum_{\ell \in \mathbb{Z}} f_\ell z^\ell$. Of course, that sequence bounded, if and only if $|z| = 1 \iff s = i\omega$ for some $\omega \in \mathbb{R}$. Notice the similarity to the Fourier transform.

If (f_ℓ) is causal, $f_\ell = 0$ for $\ell < 0$, then the eigenvalue is given by the evaluation of a **power series**. Let us associate a (formal) power series to every causal sequence

$$(f_\ell) \text{ causal} \iff (\mathcal{Z}(f_\ell))(z) := \sum_{\ell=0}^{\infty} f_\ell z^\ell, \quad z \in \mathbb{C}. \tag{3.1.4.17}$$

The function possibly defined by this power series is called the **z-transform** of (f_ℓ) . If (f_ℓ) is summable, then the series will converge inside the unit disc $\{z \in \mathbb{C} : |z| < 1\} \subset \mathbb{C}$ and define an analytic function there.

For (formal) power series the discrete convolution formula

$$((f_\ell) * (g_\ell))_n = \sum_{\ell=0}^n f_{n-\ell} \cdot g_\ell = \sum_{\ell=0}^n f_\ell \cdot g_{n-\ell}, \quad n \in \mathbb{N}_0. \tag{3.1.2.12}$$

agrees with the **Cauchy product** of the two sequences. An important consequence is that the product of z-transforms of two summable causal sequences is equivalent to the power series expansion of the discrete convolution of the given sequences.

Theorem 3.1.4.18. z-Transform and discrete convolution

If (g_ℓ) and (f_ℓ) are causal summable sequences, then

$$\mathcal{Z}((f_\ell) * (g_\ell))(z) = \mathcal{Z}((f_\ell))(z) \cdot \mathcal{Z}((g_\ell))(z), \quad \forall z \in \{z \in \mathbb{C} : |z| < 1\}. \tag{3.1.4.19}$$

Note that in (3.1.4.19) \cdot is the multiplication in \mathbb{C} . Again, we have mapped a convolution in “sequence domain” to a simple pointwise multiplication in “z-domain”. This is the same structure captured in Thm. 3.1.4.2 and Thm. 3.1.4.3. ┘

§3.1.4.20 (Truncated convolution of causal sequences: matrix view) We continue our study of discrete convolutions (3.1.2.12) of causal sequences $(f_\ell) \subset \mathbb{C}$ and $(g_\ell) \subset \mathbb{C}$:

$$((f_\ell) * (g_\ell))_n = \sum_{\ell=0}^n f_{n-\ell} \cdot g_\ell = \sum_{\ell=0}^n f_\ell \cdot g_{n-\ell}, \quad n \in \mathbb{N}_0. \tag{3.1.2.12}$$

In computations, we are interested in only a finite number $N + 1, N \in \mathbb{N}$, of terms,

$$y_n := ((f_\ell) * (g_\ell))_n = \sum_{\ell=0}^n f_{n-\ell} \cdot g_\ell = \sum_{\ell=0}^n f_\ell \cdot g_{n-\ell}, \quad n = 0, \dots, N. \tag{3.1.4.21}$$

which can be expressed as, see also (3.1.2.15),

$$\begin{bmatrix} y_0 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} f_0 & 0 & \dots & \dots & \dots & 0 \\ f_1 & f_0 & 0 & \dots & \dots & \vdots \\ f_2 & f_1 & f_0 & 0 & \dots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & 0 \\ f_N & f_{N-1} & \dots & f_2 & f_1 & f_0 \end{bmatrix} \begin{bmatrix} g_0 \\ \vdots \\ g_N \end{bmatrix} \Leftrightarrow \mathbf{y} = \mathbf{K}\mathbf{g}, \quad \mathbf{K} \in \mathbb{C}^{N+1,N+1}, \quad (3.1.4.22)$$

that is, a matrix×vector multiplication with a lower-triangular matrix \mathbf{K} of a very special structure: It is a lower-triangular **Toeplitz matrix**. ┘

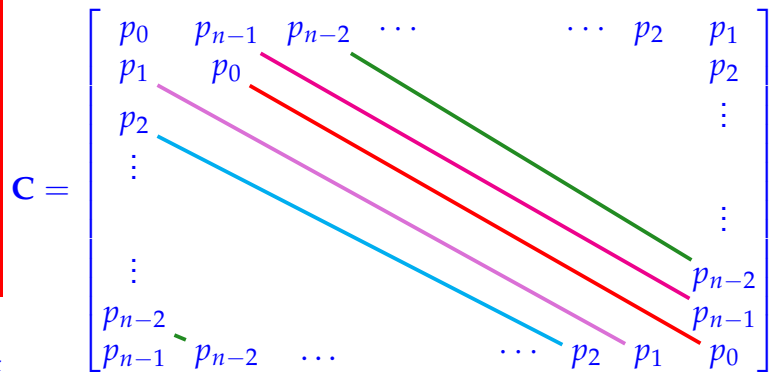
3.1.5 Toeplitz Matrix Numerical Linear Algebra

We now revisit [NumCSE Chapter 4], which in great detail discusses the relationships of and algorithms for periodic convolutions (\rightarrow [NumCSE Def. 4.1.4.7]) and causal discrete convolution, see [NumCSE Section 4.1] and, in particular, [NumCSE Rem. 4.1.4.15]. As explained in [NumCSE Section 4.2.2], diagonalization of periodic convolutions will lead to the **discrete Fourier transform (DFT)** as the fundamental linear transformation underlying all algorithms connected with discrete convolutions. The **Fast Fourier Transform (FFT)** offers an optimal-complexity implementation of DFT, see [NumCSE Section 4.3]. We give a summary of the considerations leading to an optimal algorithm for causal discrete convolution.

§3.1.5.1 (Tool: Circulant matrices \rightarrow [NumCSE § 4.1.4.11])

Definition 3.1.5.2. circulant matrix \rightarrow [NumCSE Def. 4.1.4.12]

A matrix $\mathbf{C} \in \mathbb{C}^{n,n}$, $n \in \mathbb{N}$, is **circulant**, if there exists an n -periodic sequence $(p_k)_{k \in \mathbb{Z}}$ such that

$$(\mathbf{C})_{\ell,j} = p_{\ell-j}, \quad 1 \leq \ell, j \leq n.$$


A sequence $(p_k)_{k \in \mathbb{Z}}$ is n -periodic, if $p_{k+n} = p_k$ for all $k \in \mathbb{Z}$.

► The columns and rows of a circulant $n \times n$ -matrix can be generated by successive cycling shifting of the entries of an n -vector.

The multiplication of a circulant matrix $\mathbf{C} \in \mathbb{C}^{n,n}$ generated by the n -periodic sequence (p_k) with a vector $\mathbf{x} = [x_1, \dots, x_n]^T \in \mathbb{C}^n$ amounts to **periodic discrete convolution** [NumCSE Def. 4.1.4.7]:

$$(\mathbf{C}\mathbf{x})_n = \sum_{\ell=1}^n p_{n-\ell} x_\ell, \quad n = 1, \dots, N. \quad (3.1.5.3)$$

An elementary and fundamental observation is that all circulant matrices $\in \mathbb{C}^{n,n}$ commute and, therefore, share the same basis of eigenvectors.

Theorem 3.1.5.4. Diagonalization of circulant matrices

For any circulant matrix $\mathbf{C} \in \mathbb{C}^{n,n}$, $n \in \mathbb{N}$, $(\mathbf{C})_{\ell,j} = p_{\ell-j}$, (p_k) an n -periodic sequence of complex numbers, holds

$$\mathbf{C}\bar{\mathbf{F}}_n = \bar{\mathbf{F}}_n \text{diag}(\lambda_1, \dots, \lambda_n) \quad , \quad \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_n \end{bmatrix} := \mathbf{F}_n \begin{bmatrix} p_0 \\ \vdots \\ p_{n-1} \end{bmatrix} \quad , \quad (3.1.5.5)$$

where $\mathbf{F}_n \in \mathbb{C}^{n,n}$ is the *Fourier matrix*

$$(\mathbf{F}_n)_{\ell,j} = \omega_n^{(\ell-1)(j-1)} \quad , \quad \ell, j \in \{1, \dots, n\} \quad , \quad \omega_n := \exp(-\frac{2\pi i}{n}) \quad . \quad (3.1.5.6)$$

Proof. (See also [NumCSE § 4.2.1.6]) Throughout this proof we adopt C++ indexing for matrix entries and vector components. In the following elementary manipulations we exploit the n -periodicity $\omega_n^{nj} = 1$ for all $j \in \mathbb{Z}$ and $\bar{\omega}_n = \omega_n^{-1}$.

$$\begin{aligned} (\mathbf{C}(\bar{\mathbf{F}}_n)_{:j})_k &= \sum_{\ell=0}^k p_\ell (\bar{\mathbf{F}}_n)_{k-\ell,j} + \sum_{\ell=k+1}^{n-1} p_\ell (\bar{\mathbf{F}}_n)_{n-\ell+k,j} \\ &= \sum_{\ell=0}^k p_\ell \bar{\omega}_n^{j(k-\ell)} + \sum_{\ell=k+1}^{n-1} p_\ell \bar{\omega}_n^{j(n-\ell+k)} \\ &= \bar{\omega}_n^{jk} \cdot \sum_{\ell=0}^k p_\ell \bar{\omega}_n^{-j\ell} + \bar{\omega}_n^{jk} \cdot \sum_{\ell=k+1}^{n-1} p_\ell \bar{\omega}_n^{-j\ell} \\ &= \bar{\omega}_n^{jk} \cdot \sum_{\ell=0}^{n-1} p_\ell \omega_n^{j\ell} = (\bar{\mathbf{F}}_n)_{k,j} \cdot (\mathbf{F}_n[p_\ell]_{\ell=0}^{n-1})_j \quad , \quad j = 0, \dots, n-1 \quad . \end{aligned}$$

This means that the j -th column of the Fourier matrix \mathbf{F}_n is an eigenvector of \mathbf{C} with eigenvalue $(\mathbf{F}_n \mathbf{p})_j$, $\mathbf{p} := [p_0, \dots, p_{n-1}]^\top \in \mathbb{C}^n$. □

Since ω_n is a root of unity, the Fourier matrix as defined in (3.1.5.6) is, up to scaling with $\frac{1}{\sqrt{n}}$, *unitary* \rightarrow [NumCSE Lemma 4.2.1.14],

$$\mathbf{F}_n^{-1} = \frac{1}{n} \mathbf{F}_n^H = \frac{1}{n} \bar{\mathbf{F}}_n \quad , \quad (3.1.5.7)$$

which implies the diagonalization formula [NumCSE Eq. (4.2.1.17)]

$$\mathbf{C} = \mathbf{F}_n^{-1} \text{diag}(\lambda_1, \dots, \lambda_n) \mathbf{F}_n \quad , \quad (3.1.5.8)$$

that is, the columns of the Fourier matrix provide an eigenbasis for every circulant matrix.

The multiplication of a Fourier matrix \mathbf{F}_n with a vector is known as *discrete Fourier transform (DFT)*:

$$\begin{aligned} \mathbf{c} = \mathbf{F}_n \mathbf{y} &\Leftrightarrow \mathbf{y} = \frac{1}{n} \bar{\mathbf{F}}_n \mathbf{c} \quad , \quad \mathbf{c} = [c_k]_{k=1}^n \quad , \quad \mathbf{y} = [y_j]_{j=1}^n \in \mathbb{C}^n \\ \blacktriangleright \quad c_k &= \sum_{j=1}^n y_j \omega_n^{(k-1)(j-1)} \Leftrightarrow y_j = \frac{1}{n} \sum_{k=1}^n c_k \omega_n^{-(k-1)(j-1)} \quad , \quad k, j = 1, \dots, n \quad . \end{aligned} \quad (3.1.5.9)$$

Notation: We write $\text{FFT}_n(\mathbf{y}) := \mathbf{F}_n \mathbf{y}$ and $\text{IFFT}_n(\mathbf{c}) := \mathbf{F}_n^{-1} \mathbf{c}$

$$\text{FFT}_n \mathbf{y} = \left[\sum_{j=1}^n y_j \omega_n^{(k-1)(j-1)} \right]_{k=1}^n \Leftrightarrow \text{IFFT}_n \mathbf{c} = \left[\frac{1}{n} \sum_{k=1}^n c_k \omega_n^{-(k-1)(j-1)} \right]_{j=1}^n, \quad k, j = 1, \dots, n. \tag{3.1.5.10}$$

Thanks to (3.1.5.8), the multiplication of a vector with a circulant matrix $\mathbf{C} \in \mathbb{C}^{n,n}$ generated by the n -periodic sequence (p_k) can be expressed as ($\cdot *$ stands for entrywise multiplication, MATLAB syntax.)

$$\mathbf{C}\mathbf{x} = \text{IFFT}_n(\text{FFT}_n([p_0, \dots, p_{n-1}]^\top) \cdot * \text{FFT}_n(\mathbf{x})) \quad \forall \mathbf{x} \in \mathbb{C}^n. \tag{3.1.5.11}$$

A C++ implementation based on a DFT library function of EIGEN is given in [NumCSE Code 4.2.2.4]. \lrcorner

§3.1.5.12 (Fast Fourier Transform (FFT)) The Fast Fourier Transform is a divide-and-conquer algorithm for the efficient computation of the discrete Fourier transform of complex vectors, see [NumCSE Section 4.3].

Asymptotic computational effort for DFT

cost(DFT of a vector $\in \mathbb{C}^n$) = $O(n \log n)$ for $n \rightarrow \infty$

► Owing to (3.1.5.11) the asymptotic computational effort for multiplying a circulant matrix $\in \mathbb{C}^{n,n}$ with a vector is

$$\text{cost}(\text{circulant } n \times n\text{-matrix} \times \text{vector}) = O(n \log n) \quad \text{for } n \rightarrow \infty$$

\lrcorner

§3.1.5.14 (Techniques for Toeplitz matrices) We observe that the matrix $\mathbf{K} \in \mathbb{C}^{N+1, N+1}$ from (3.1.4.22) has “constant (off-)diagonals” and, therefore, belongs to a special class of matrices \rightarrow [NumCSE Def. 4.5.1.7].

Definition 3.1.5.15. Toeplitz matrix

$\mathbf{T} \in \mathbb{C}^{m,n}$, $m, n \in \mathbb{N}$, is a **Toeplitz matrix** generated by the sequence $(u_{-m+1}, \dots, u_{n-1})$ of $n + m - 1$ complex numbers, if

$$(\mathbf{T})_{ij} = u_{j-i}, \quad 1 \leq i \leq m, \quad 1 \leq j \leq n.$$

$$\mathbf{T} = \begin{bmatrix} u_0 & u_1 & \cdots & \cdots & u_{n-1} \\ u_{-1} & u_0 & u_1 & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ \vdots & & & \ddots & \ddots & u_1 \\ u_{1-m} & \cdots & \cdots & u_{-1} & u_0 \end{bmatrix}$$

Obviously a Toeplitz matrix $\mathbf{T} \in \mathbb{C}^{m,n}$ has an information content of merely $m + n + 1$ numbers. This sets a strict lower bound for the asymptotic complexity of operations involving Toeplitz matrices.



Idea behind fast algorithms for Toeplitz matrices:

Circulant augmentation: embed Toeplitz matrix into larger circulant matrix

Lemma 3.1.5.16. Circulant augmentation of Toeplitz matrix

Given a sequence $(u_{-m+1}, \dots, u_{n-1})$ of $m + n - 1$ numbers, let $\mathbf{C} \in \mathbb{C}^{n+m, n+m}$ be the *circulant matrix* (\rightarrow Def. 3.1.5.2) generated by the $m + n$ -periodic sequence

$$(u_0, u_{-1}, u_{-2}, \dots, u_{-m+1}, 0, u_{n-1}, u_{n-2}, \dots, u_1).$$

Then the upper-left $m \times n$ -block $(\mathbf{C})_{1:m, 1:n}$ of \mathbf{C} is the $m \times n$ *Toeplitz matrix* (\rightarrow Def. 3.1.5.15) generated by the sequence

$$(u_{-m+1}, u_{-m+2}, \dots, u_0, \dots, u_{n-2}, u_{n-1}).$$

Appropriately the matrix \mathbf{C} is called the *circulant augmentation* of \mathbf{T} .

The following formula demonstrates the structure of \mathbf{C} in the case $m = n$ with the Toeplitz block highlighted in color.

$$\mathbf{C} = \begin{bmatrix} u_0 & u_1 & \cdots & \cdots & u_{n-1} & 0 & u_{1-n} & \cdots & \cdots & u_{-1} \\ u_{-1} & u_0 & u_1 & & \vdots & u_{n-1} & 0 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \vdots & \vdots & & \ddots & \ddots & \vdots \\ \vdots & & & \ddots & u_1 & \vdots & & \ddots & \ddots & u_{1-n} \\ u_{1-n} & \cdots & \cdots & u_{-1} & u_0 & u_1 & & & u_{n-1} & 0 \\ \hline 0 & u_{1-n} & \cdots & \cdots & u_{-1} & u_0 & u_1 & \cdots & \cdots & u_{n-1} \\ u_{n-1} & 0 & \ddots & & \vdots & u_{-1} & u_0 & u_1 & & \vdots \\ \vdots & \ddots & \ddots & & \vdots & \vdots & \ddots & \ddots & & \vdots \\ \vdots & & & \ddots & \vdots & \vdots & & \ddots & \ddots & \vdots \\ \vdots & & & & u_{1-n} & \vdots & & \ddots & \ddots & u_1 \\ u_1 & & & & u_{n-1} & 0 & u_{1-n} & \cdots & \cdots & u_{-1} & u_0 \end{bmatrix}$$

The case of a rectangular Toeplitz block with $m = 6, n = 4$ is shown next:

$$\mathbf{C} = \begin{bmatrix} u_0 & u_1 & u_2 & u_3 & 0 & u_{-5} & u_{-4} & u_{-3} & u_2 & u_{-1} \\ u_{-1} & u_0 & u_1 & u_2 & u_3 & 0 & u_{-5} & u_{-4} & u_{-3} & u_{-2} \\ u_{-2} & u_{-1} & u_0 & u_1 & u_2 & u_3 & 0 & u_{-5} & u_{-4} & u_{-3} \\ u_{-3} & u_{-2} & u_{-1} & u_0 & u_1 & u_2 & u_3 & 0 & u_{-5} & u_{-4} \\ u_{-4} & u_{-3} & u_{-2} & u_{-1} & u_0 & u_1 & u_2 & u_3 & 0 & u_{-5} \\ u_{-5} & u_{-4} & u_{-3} & u_{-2} & u_{-1} & u_0 & u_1 & u_2 & u_3 & 0 \\ \hline 0 & u_{-5} & u_{-4} & u_{-3} & u_{-2} & u_{-1} & u_0 & u_1 & u_2 & u_3 \\ u_3 & 0 & u_{-5} & u_{-4} & u_{-3} & u_{-2} & u_{-1} & u_0 & u_1 & u_2 \\ u_2 & u_3 & 0 & u_{-5} & u_{-4} & u_{-3} & u_{-2} & u_{-1} & u_0 & u_1 \\ u_1 & u_2 & u_3 & 0 & u_{-5} & u_{-4} & u_{-3} & u_{-2} & u_{-1} & u_0 \end{bmatrix}$$

The message of Lemma 3.1.5.16 is that for a given Toeplitz matrix $\mathbf{T} \in \mathbb{C}^{m,n}$, we can find a circulant matrix $\mathbf{C} \in \mathbb{C}^{m+n, m+n}$ such that

$$\mathbf{C} = \begin{bmatrix} \mathbf{T} & * \\ * & * \end{bmatrix}, \quad * \hat{=} \text{matrix blocks of suitable size.} \tag{3.1.5.17}$$

As a consequence the product of a Toeplitz matrix $\mathbf{T} \in \mathbb{C}^{m,n}$ with a vector $\mathbf{u} \in \mathbb{C}^n$ can be computed by the multiplication of its circulant augmentation with a “zero-padded” argument vector:

$$\mathbf{C} \begin{bmatrix} \mathbf{u} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{T} & * \\ * & * \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{T}\mathbf{u} \\ * \end{bmatrix}, \tag{3.1.5.18}$$

where $\mathbf{C} \in \mathbb{C}^{m+n,m+n}$ is the circulant matrix from Lemma 3.1.5.16 with $(\mathbf{C})_{1:m,1:n} = \mathbf{T}$. This shows how to harness the power of FFT for multiplying a Toeplitz matrix with a vector.

Toeplitz matrix \times vector

The multiplication of a Toeplitz matrix with a vector can be converted to the multiplication of a circulant matrix with a vector:

$$\text{cost}(m \times n \text{ Toeplitz matrix} \times \text{vector}) = O((m+n) \log(m+n)) \text{ for } m, n \rightarrow \infty$$

§3.1.5.20 (Diagonalization-based algorithms for discrete convolutions) It is clear from (3.1.4.22) that the FFT-based multiplication of a general Toeplitz matrix with a vector can immediately be applied for the computation of the initial $N + 1$ terms of a discrete convolution (3.1.5.22) of causal sequences, because

$$\mathbf{K} := \begin{bmatrix} f_0 & 0 & \dots & \dots & \dots & 0 \\ f_1 & f_0 & 0 & \dots & & \vdots \\ f_2 & f_1 & f_0 & 0 & \dots & \\ \vdots & \ddots & \ddots & \ddots & \ddots & \\ \vdots & & \ddots & \ddots & \ddots & 0 \\ f_N & f_{N-1} & \dots & f_2 & f_1 & f_0 \end{bmatrix} \in \mathbb{C}^{N+1,N+1}$$

obviously is a **Toeplitz matrix** generated by the sequence

$$(f_N, f_{N-1}, \dots, f_0, 0, \dots, 0) \in \mathbb{C}^{2N+1}.$$

The $N + 1$ first terms of the discrete convolution of causal sequences can be computed with an asymptotic effort of $O(N \log N)$ for $N \rightarrow \infty$.

§3.1.5.21 (Efficient solution of convolution equations) We consider the (truncated) convolution equation (3.1.2.15) in the simple setting $X = Y = \mathbb{C}$. Given $\mathbf{y} \in \mathbb{C}^n$ we seek a vector $\mathbf{u} \in \mathbb{C}^n$ such that

$$\begin{bmatrix} f_0 & 0 & \dots & \dots & \dots & 0 \\ f_1 & f_0 & 0 & \dots & & \vdots \\ f_2 & f_1 & f_0 & 0 & \dots & \\ \vdots & \ddots & \ddots & \ddots & \ddots & \\ \vdots & & \ddots & \ddots & \ddots & 0 \\ f_{n-1} & f_{n-2} & \dots & f_2 & f_1 & f_0 \end{bmatrix} \begin{bmatrix} u_1 \\ \vdots \\ \vdots \\ u_n \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ \vdots \\ y_n \end{bmatrix} \Leftrightarrow \mathbf{K}\mathbf{u} = \mathbf{y}. \tag{3.1.5.22}$$

We assume $f_0 \neq 0$, which ensures that the lower-triangular coefficient matrix of (3.1.5.22) is invertible. The simple forward elimination according to (3.1.2.16),

$$u_\ell = f_0^{-1} \left(y_\ell - \sum_{k=1}^{\ell-1} f_{\ell-k} u_k \right), \quad \ell = 1, \dots, n,$$

gives the result vector with an asymptotic effort of $O(n^2)$ for $n \rightarrow \infty$. A faster method uses the efficient algorithms for Toeplitz matrices from § 3.1.5.14.



Idea: Divide-and-conquer algorithm:

Apply **recursion** to 2×2 -block split linear system

For $1 \leq k < n$, preferably $k \approx n/2$, consider

$$\mathbf{K}\mathbf{u} = \mathbf{y} \Leftrightarrow \begin{bmatrix} (\mathbf{K})_{1:k,1:k} & \mathbf{O} \\ (\mathbf{K})_{k+1:n,1:k} & (\mathbf{K})_{k+1:n,k+1:n} \end{bmatrix} \begin{bmatrix} (\mathbf{u})_{1:k} \\ (\mathbf{u})_{k+1:n} \end{bmatrix} = \begin{bmatrix} (\mathbf{y})_{1:k} \\ (\mathbf{y})_{k+1:n} \end{bmatrix}, \quad (3.1.5.23)$$

and note that

- ◆ both $(\mathbf{K})_{1:k,1:k}$ and $(\mathbf{K})_{k+1:n,k+1:n}$ are lower-triangular Toeplitz matrices again, and
- ◆ $(\mathbf{K})_{k+1:n,1:k}$ is a Toeplitz matrix.

This suggests the following algorithm:

- ❶ solve $(\mathbf{K})_{1:k,1:k}(\mathbf{u})_{1:k} = (\mathbf{y})_{1:k} \quad \triangleright \text{recursion}$
- ❷ Compute $\mathbf{t} := (\mathbf{y})_{k+1:n} - (\mathbf{K})_{k+1:n,1:k}(\mathbf{u})_{1:k}$ (Toeplitz matrix \times vector)
- ❸ Solve $(\mathbf{K})_{k+1:n,k+1:n}(\mathbf{u})_{k+1:n} = \mathbf{t} \quad \triangleright \text{recursion}$

The asymptotic complexity can easily be determined for the case $n = 2^p$, where at each level of the recursion the task is split into two problems of half the size. Denoting by $W(p)$ the computational effort for $n = 2^p$ and taking into account that the multiplication of a vector with an $2^p \times 2^p$ Toeplitz matrix involves asymptotic computational cost of $O(p2^p)$, by trivial induction we arrive at the estimate [BHS80]

$$W(p) \leq 2W(p-1) + C2^p p \quad \triangleright \quad W(p) \leq C2^p p^2.$$

Hence, in this case, the discrete convolution equation can be solved with an asymptotic effort of $O(n \log^2 n)$. This holds for all system sizes.

The asymptotic cost for computing n components of the solution of the discrete convolution equation (3.1.2.15) is $O(n \log^2 n)$.

┘

3.2 Convolution Equations: Examples

Convolution equations occur in a wide range of mathematical models of phenomena with non-local interactions and, in particular, “memory in time”. We highlight a few simple examples.

3.2.1 Tomography: Abel Integral Equation

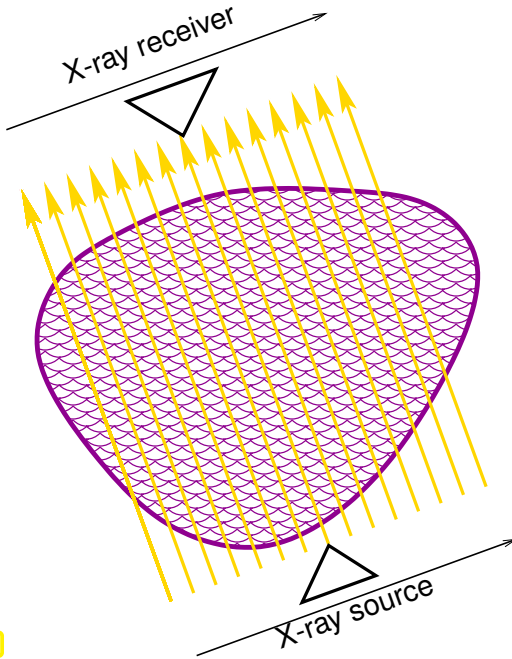


Fig. 123

◁ 2D cross-section of a tomography set-up.

In **X-ray tomography** parallel X-rays are shot through an object and their attenuation is measured. From the attenuation regarded as a function of the ray line the spatial density distribution of the object can be computed by means of the **Radon transform** [Rie03, Sect. 1.1].

This method is the mathematical foundation of **CT-scans**, which is a widely used technology in medical imaging.

We study only a substantially simplified setting.

We assume that the object is a long straight circular cylinder with radius 1 and that its density ρ is a function of the radius only: $\rho = \rho(x_1, x_2) = \rho(r)$, $r := \sqrt{x_1^2 + x_2^2}$.

Hence, only a single ray direction is required, let it be the x_2 -direction. The ray position can be characterized by its x_1 -coordinate.

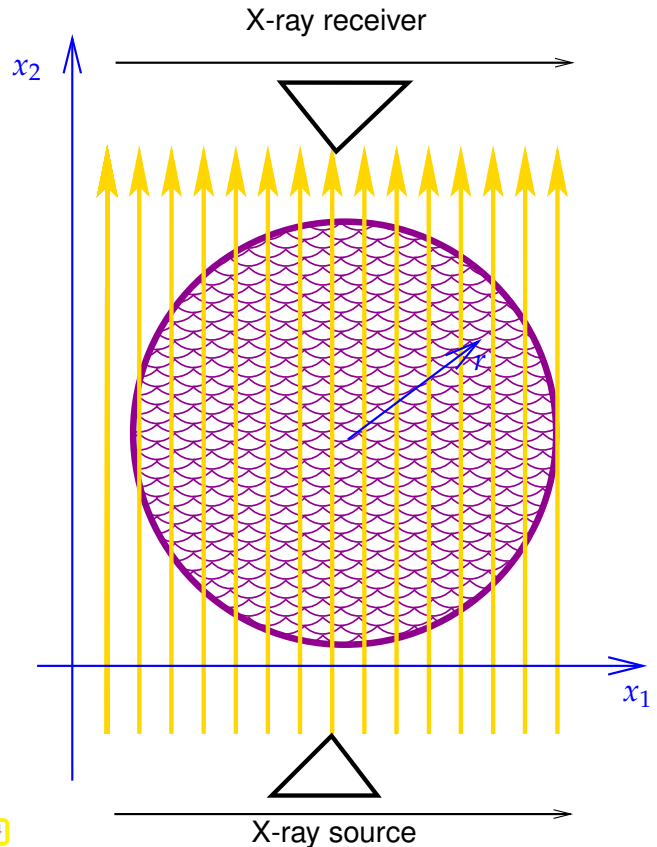
Let $I = I(x_1, x_2)$ denote the intensity of the X-rays. It is governed by the **attenuation equation**

$$\frac{\partial I}{\partial x_2}(x_1, x_2) = -\rho(x_1, x_2)I(x_1, x_2), \quad (3.2.1.1)$$

a simple linear ordinary differential equation with x_1 acting as a parameter. Dividing by $I(x_1, x_2)$ we get from the chain rule

$$\frac{\partial}{\partial x_2} \log(I(x_1, x_2)) = -\rho(x_1, x_2). \quad (3.2.1.2)$$

Fig. 124



Write $I_S = I_S(x_1)$ for the intensity at the source, and $I_R = I_R(x_1)$ for the intensity measured by the receiver. Integrating (3.2.1.2) in x_2 -direction over $[-1, 1]$ yields ($\rho(x_1, x_2) = 0$ for $x_1^2 + x_2^2 > 1$)

$$g(x_1) := -\log \frac{I_R(x_1)}{I_S(x_1)} = 2 \int_0^{\sqrt{1-x_1^2}} \rho(x_1, x_2) dx_2, \quad -1 \leq x_1 \leq 1, \quad (3.2.1.3)$$

where we also used the symmetry of ρ : $\rho(x_1, x_2) = \rho(x_1, -x_2)$.

The task is to tease out $\rho = \rho(\sqrt{x_1^2 + x_2^2})$ from the data $g = g(x_1)$, which have to satisfy $g(-1) = g(1) = 0$ and $g(-x_1) = g(x_1)$. This amounts to seeking a non-negative function $\rho = \rho(r)$

defined on $[0, 1]$ and solving the **integral equation**

$$2 \int_0^{\sqrt{1-x_1^2}} \rho(\sqrt{x_1^2 + x_2^2}) dx_2 = g(x_1), \quad 0 \leq x_1 \leq 1. \quad (3.2.1.4)$$

We perform the substitutions

$$\begin{aligned} t &:= 1 - x_1^2 &\Rightarrow x_1 &= \sqrt{1-t}, \\ \xi &:= 1 - x_1^2 - x_2^2 &\Rightarrow x_2 &= \sqrt{t-\xi}, \quad d\xi = -2x_2 dx_2, \end{aligned}$$

which converts the integral equation (3.2.1.4) into

$$\int_t^0 \frac{\rho(\sqrt{1-\xi})}{\sqrt{t-\xi}} d\xi = g(\sqrt{1-t}), \quad 0 \leq t \leq 1. \quad (3.2.1.5)$$

We continue with substitutions and set

$$u(\xi) := \rho(\sqrt{1-\xi}), \quad 0 \leq \xi \leq 1, \quad y(t) := -g(\sqrt{1-t}), \quad 0 \leq t \leq 1,$$

and, finally, end up with the **Abel integral equation** for $u : [0, 1] \rightarrow \mathbb{R}$

$$\int_0^t \frac{u(\xi)}{\sqrt{t-\xi}} d\xi = y(t), \quad 0 \leq t \leq 1. \quad (3.2.1.6)$$

Notice the structure of a **convolution equation** (with singular kernel $k(t, \xi) = \frac{1}{\sqrt{t-\xi}}$) for causal functions as presented abstractly in § 3.1.1.25. An equivalent way to write (3.2.1.6) is

$$(Au)(t) := (\{t \mapsto \frac{1}{\sqrt{t}}\} * u)(t) = y(t), \quad 0 \leq t \leq 1, \quad (3.2.1.7)$$

where A is known as **Abel integral operator**. The restriction to the finite interval $[0, 1]$ is irrelevant thanks to causality.

In Ex. 3.1.3.8 we established

$$\mathcal{L}\{t \mapsto t_+^{-1/2}\}(s) = \Gamma(1/2)s^{-1/2} = \frac{\sqrt{\pi}}{\sqrt{s}}, \quad s \in \mathbb{C}^+. \quad (3.2.1.8)$$

Thus, the Abel integral operator can be fit into operational calculus

$$A = F(\partial_t) \quad \text{with} \quad F(s) = \frac{\sqrt{\pi}}{\sqrt{s}}. \quad (3.2.1.9)$$

Remark 3.2.1.10 (The square of the Abel integral operator) From (3.2.1.9) we conclude

$$\mathcal{L}A^2u(s) = \frac{\sqrt{\pi}}{\sqrt{s}} \mathcal{L}Au(s) = \frac{\pi}{s} (\mathcal{L}u)(s).$$

Thm. 3.1.3.22 tells us that division by s in the Laplace domain corresponds to integration in time domain: for a continuous causal function f satisfying a polynomial growth condition we have

$$\mathcal{L}\{t \mapsto \int_0^t f(\tau) d\tau\}(s) = \frac{1}{s} (\mathcal{L}f)(s), \quad s \in \mathbb{C}^+. \quad (3.2.1.11)$$

Inverting the Laplace transform this implies for a continuous causal function $u \in C^0(\mathbb{R})$

$$\begin{aligned} (A^2u)(t) &= \pi \int_0^t u(\tau) d\tau, \quad t \geq 0, \\ \Rightarrow \frac{d}{dt}(A^2u)(t) &= \pi u(t), \quad t \geq 0. \end{aligned}$$

In a sense, the Abel integral operator A is the square root of the antiderivative. ┘

3.2.2 Impedance Boundary Conditions

A typical task in computational electromagnetics: A current-carrying “infinite” straight co-axial cable is aligned x_3 -direction. The goal is to compute the electric and magnetic fields in the vicinity of the cable.

In this setting the magneto-quasistatic (eddy current) approximation of Maxwell’s equations can be used, which boils down to the evolution equations

$$\begin{aligned} \operatorname{curl} \mathbf{E}(\mathbf{x}, t) &= -\frac{\partial}{\partial t}(\mu(\mathbf{x})\mathbf{H}(\mathbf{x}, t)), & \text{in } \mathbb{R}^3 \times]0, T[. \\ \operatorname{curl} \mathbf{H}(\mathbf{x}, t) &= \sigma(\mathbf{x})\mathbf{E}(\mathbf{x}, t) + \mathbf{j}_s(\mathbf{x}, t) \end{aligned} \quad (3.2.2.1)$$

Here,

- ◆ $\mathbf{E} = \mathbf{E}(\mathbf{x}, t)$ is the **electric field** ($[\mathbf{E}] = \text{Vm}^{-1}$),
- $\mathbf{H} = \mathbf{H}(\mathbf{x}, t)$ is the **magnetic field** ($[\mathbf{H}] = \text{Am}^{-1}$),
- ◆ $\mathbf{j}_s = \mathbf{j}_s(\mathbf{x}, t)$ is a given source **current density** ($[\mathbf{j}_s] = \text{Am}^{-2}$),
- ◆ $\sigma = \sigma(\mathbf{x}) \geq 0$ is the conductivity ($[\sigma] = \frac{\text{A}}{\text{Vm}}$),
- ◆ $\mu = \mu(\mathbf{x})$ is the uniformly positive magnetic permeability ($[\mu] = \frac{\text{Vs}}{\text{Am}}$).

We assume that

- The geometry is perfectly translation-invariant in x_3 -direction,
- the conductivity σ does not depend on x_3 and μ is constant,
- the source current flows in x_3 -direction, $\mathbf{j}_s(\mathbf{x}) = j(x_1, x_2) \cdot [0, 0, 1]^\top$.

Then, in (3.2.2.1) all partial derivatives $\frac{\partial}{\partial x_3}$ vanish, and so do the field components E_1 , E_2 , and H_3 . We eliminate \mathbf{H} and retain the x_3 -component of \mathbf{E} as unknown $u(x_1, x_2, t) := E_3(x_1, x_2, t)$, $[x_1, x_2]^\top \in \mathbb{R}^2$. Assuming vanishing fields at initial time $t = 0$ and now writing $\mathbf{x} := [x_1, x_2]^\top$ we arrive at the degenerate parabolic initial-value problem in two space dimensions

$$\begin{aligned} \frac{\partial}{\partial t}(\sigma(\mathbf{x})\mu u) - \Delta u &= f(\mathbf{x}, t) := \mu \frac{\partial j}{\partial t}(\mathbf{x}, t) & \text{in } \mathbb{R}^2 \times [0, T], \\ u(\mathbf{x}, 0) &= 0 & \text{in } \mathbb{R}^2. \end{aligned} \quad (3.2.2.2)$$

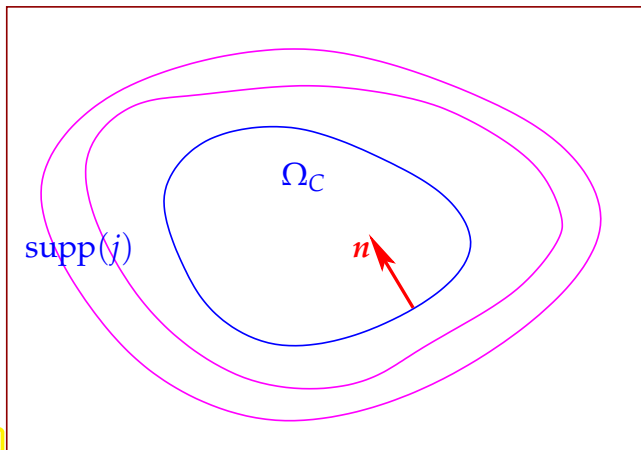


Fig. 125

The co-axial cable has a copper core with constant and large conductivity $\sigma > 0$, whereas the space outside Ω_C is non-conducting:

$$\sigma(\mathbf{x}) = \begin{cases} \sigma & \text{in } \Omega_C, \\ 0 & \text{in } \mathbb{R}^2 \setminus \Omega_C. \end{cases} \quad (3.2.2.3)$$

The core’s cross-section occupies the bounded domain $\Omega_C \subset \mathbb{R}^2$.

The core is surrounded by a cladding carrying a time-dependent source current $\mathbf{j}_s(\mathbf{x}, t) = j_s(x_1, x_2, t)[001]^\top$ in x_3 -direction, which provides the exciting source in the model. The spatial support of j , $\Omega_J := \operatorname{supp}(j) \subset \mathbb{R}^2$, is bounded and outside Ω_C . The source current is switched on at time $t = 0$.

By linearity, the evolution problem (3.2.2.2) can be transformed from time domain to the Laplace domain using Thm. 3.1.3.22. We apply the Laplace transform on both sides of the PDE in (3.2.2.2) and arrive at

$$s\sigma(x)\mu(\mathcal{L}u)(s) - \Delta(\mathcal{L}u)(s) = \widehat{f}(s) := \mathcal{L}\{t \mapsto f(\cdot, t)\}(s). \tag{3.2.2.4}$$

In the sequel we write $\widehat{u}(s) := \mathcal{L}u(s)$. Note that $s \in \mathbb{C}^+$ can be regarded as a *parameter* in (3.2.2.4).

Two *approximations* are commonly applied to the model (3.2.2.4):

- (I) Instead on the whole space \mathbb{R}^3 the spatial computational domain is truncated to a bounded domain $\Omega \subset \mathbb{R}^2$ containing both Ω_J and Ω_C and indicated by the outer box in Fig. 125,
- (II) The interaction of the conducting core and the electromagnetic fields is taken into account by imposing **impedance boundary conditions** on the surface of Ω_C :

$$\mathbf{grad} \widehat{u}(x, s) \cdot \mathbf{n}(x) = -\sqrt{s} \eta \widehat{u}(x, s) \quad \text{for all } x \in \partial\Omega_C, \tag{3.2.2.5}$$

where \mathbf{n} is the unit normal vectorfield on $\partial\Omega_C$ pointing into the interior of Ω_C , and $\eta := \sqrt{\mu\sigma}$. A derivation is given in Rem. 3.2.2.14 below.

Imposing homogeneous Neumann boundary conditions for \widehat{u} at the artificial truncation boundary, the final boundary value problem in Laplace domain seeks $\widehat{u} = \widehat{u}(x, s)$ satisfying

$$\begin{aligned} -\Delta_x \widehat{u}(x, s) &= \widehat{f}(x, s) && \text{in } \Omega_e := \Omega \setminus \overline{\Omega}_C, \\ \mathbf{grad}_x \widehat{u}(\cdot, s) \cdot \mathbf{n} &= -\sqrt{s} \eta \widehat{u}(\cdot, s) && \text{on } \partial\Omega_C, \\ \mathbf{grad}_x \widehat{u}(\cdot, s) \cdot \mathbf{n} &= 0 && \text{on } \partial\Omega, \end{aligned} \quad s \in \mathbb{C}^+. \tag{3.2.2.6}$$

This is a second-order elliptic boundary value problem with linear impedance-type boundary conditions. As explained in [NumPDE Ex. 1.8.0.6] its weak (variational) formulation reads: Given any $s \in \mathbb{C}^+$ seek $\widehat{u}(s) \in H^1(\Omega_e)$ such that

$$\int_{\Omega_e} \mathbf{grad} \widehat{u}(s) \cdot \mathbf{grad} v \, dx + \int_{\partial\Omega_C} \sqrt{s} \eta \widehat{u}(s) v \, dS = \int_{\Omega_e} \widehat{f}(s) v \, dx \quad \forall v \in H^1(\Omega_e). \tag{3.2.2.7}$$

Appealing to Thm. 3.1.3.22 again, we can transform (3.2.2.7) back into time domain. We obtain an evolution problem for $u = u(x, t)$ with the following spatial variational formulation: seek $u(t) \in H^1(\Omega_e)$

$$\underbrace{\int_{\Omega_e} \mathbf{grad} u(t) \cdot \mathbf{grad} v \, dx}_{=:a(u,v)} + \underbrace{\int_0^t k(t-\tau) \int_{\partial\Omega_C} \eta u(\tau) v \, dS}_{\text{convolution term}} = \underbrace{\int_{\Omega_e} f(t) v \, dx}_{=:l(v)} \quad \forall v \in H^1(\Omega_e), \tag{3.2.2.8}$$

with $k(t) := \frac{1}{\sqrt{\pi t}}$, as derived in Ex. 3.1.3.8.

Note that the multiplication with \sqrt{s} in Laplace domain has become a *convolution* in time domain. What we also know about the convolution kernel k in (3.2.2.8) is its Laplace transform: $(\mathcal{L}k)(s) = s^{1/2}$.

Let us rewrite (3.2.2.8) resorting to operational calculus from Def. 3.1.4.5. We define the operator-valued Laplace transform

$$F := \begin{cases} \mathbb{C} \setminus]-\infty, 0] & \rightarrow L(H^1(\Omega_e), L(H^1(\Omega_e), \mathbb{C})) \\ s & \mapsto \begin{cases} H^1(\Omega_e) & \rightarrow L(H^1(\Omega_e), \mathbb{C}) \\ u & \mapsto \begin{cases} H^1(\Omega_e) & \rightarrow \mathbb{C} \\ v & \mapsto \sqrt{s} \int_{\partial\Omega_C} \eta u(x) v(x) \, dS(x) \end{cases} \end{cases} \end{cases} \tag{3.2.2.9}$$

The space $L(H^1(\Omega_e), L(H^1(\Omega_e), \mathbb{C}))$ can be identified with $L(H^1(\Omega_e) \times H^1(\Omega_e), \mathbb{C})$, the vector space of continuous bilinear forms on $H^1(\Omega_e)$, which means that $F(s)$ is a bilinear form on $H^1(\Omega_e)$. This and the abbreviations from (3.2.2.8) permit to rewrite (3.2.2.8) as

$$u \in H^1(\Omega_e): \quad a(u, v) + F(\partial_t)(u, v) = \ell(v) \quad \forall v \in H^1(\Omega_e). \quad (3.2.2.10)$$

Remark 3.2.2.11 (Kernel with known Laplace transform) This example illustrates a mathematical model with a convolution term in time, whose kernel has a simple Laplace transform. ┘

Remark 3.2.2.12 (Finite element discretization) In the spirit of the method of lines introduced in [NumPDE Section 9.2.4] we can achieve the spatial semi-discretization of (3.2.2.8) through a Galerkin approach using $H^1(\Omega_e)$ -conforming finite elements on a triangulation of Ω_e . The simplest choice would be triangular linear Lagrangian finite elements, see [NumPDE Section 2.4].

Writing $N \in \mathbb{N}$ for the dimension of the finite element space and $\vec{\mu}(t)$ for coefficient vector of the basis expansion of the finite element approximation of $u(t)$, this will result in the convolution equation

$$\mathbf{A}\vec{\mu}(t) + (\mathbf{K} * \vec{\mu})(t) = \vec{\varphi}(t), \quad \mathbf{K}(\tau) = k(\tau)\mathbf{B}, \quad (3.2.2.13)$$

where $\triangleright \mathbf{A} \in \mathbb{R}^{N,N}$ is the finite element Galerkin matrix (“stiffness matrix”) for $-\Delta$,
 $\triangleright \mathbf{B} \in \mathbb{R}^{N,N}$ arises from the boundary bilinear form in (3.2.2.8). ┘

Remark 3.2.2.14 (Derivation of impedance conditions) If the conductivity σ of the conducting core is large, the electromagnetic field do not penetrate deep, which is known as **skin effect**.

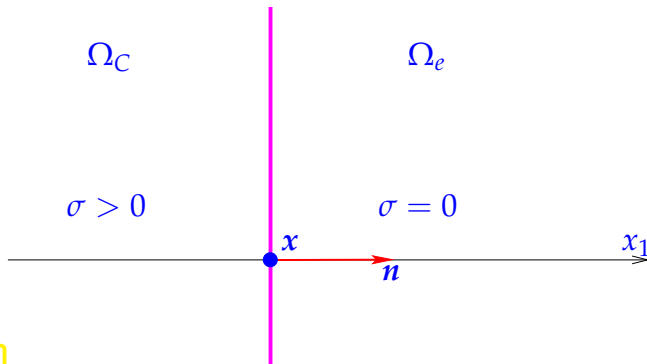


Fig. 126

If the (smooth) boundary $\partial\Omega$ at the length scale of the penetration depth can be well approximated by a plane, then we can describe the local behavior of the fields by a **1D half-space model**. Given $x \in \partial\Omega_C$ we choose the direction of the exterior normal vector $n(x)$ as x_1 -direction. All variations of the fields in other directions are suppressed, which turns (3.2.2.4) into the ordinary differential equation for $x_1 \mapsto \hat{u}(x_1, s)$,

$$-\frac{d^2\hat{u}}{dx_1^2} + s\mu\sigma(x_1)\hat{u} = 0 \quad \text{for } x_1 < 0. \quad (3.2.2.15)$$

Only solutions that decay $\rightarrow 0$ as $x_1 \rightarrow -\infty$ are of interest and that solution is

$$\hat{u}(x_1, s) = \exp(\sqrt{s\mu\sigma} x_1)\hat{u}(0, s), \quad x_1 < 0, \quad (3.2.2.16)$$

$$\blacktriangleright \frac{\partial\hat{u}}{\partial x_1}(0, s) = -\sqrt{s\eta}\hat{u}(0, s), \quad \eta := \sqrt{\mu\sigma}. \quad (3.2.2.17)$$

Remember that the exterior normal unit vector $n(x)$ at $\partial\Omega_C$ points in x_1 -direction.

$$\blacktriangleright \mathbf{grad} \hat{u}(x, s) \cdot n(x) \approx \frac{d\hat{u}}{dx_1}(0, s) = -\sqrt{s\eta}\hat{u}(0, s) = -\sqrt{s\eta}\hat{u}(x, s). \quad (3.2.2.18)$$

This is the impedance boundary condition stated in (3.2.2.5). ┘

3.2.3 Time-Domain Boundary Integral Equations

§3.2.3.1 (Acoustic Scattering) Freely propagating acoustic waves are described by a time-dependent pressure distribution $u = u(\mathbf{x}, t)$ in the air region $\Omega \subset \mathbb{R}^3$, governed by the linear wave equation, cf. [NumPDE § 9.3.1.9]

$$\frac{\partial^2 p}{\partial t^2} - c^2 \Delta_{\mathbf{x}} p = 0 \quad \text{in } \Omega \times]0, T[, \tag{3.2.3.2}$$

for fixed final observation time $T > 0$. Here, $c > 0$ is the constant wave speed, $[c] = \frac{\text{m}}{\text{s}}$, which agrees with the maximal speed of propagation in the model. For in-depth explanations refer to [NumPDE Section 9.3.2].

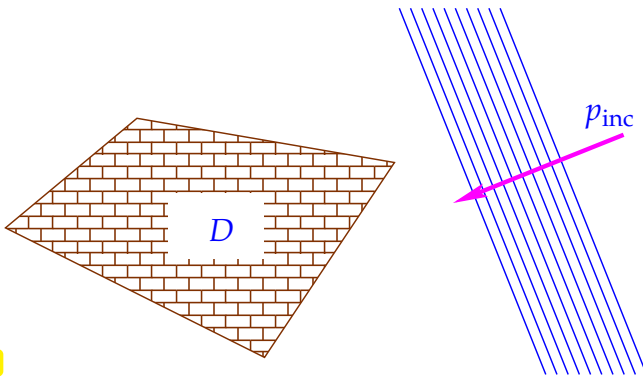


Fig. 127

We are interested in simulating the scattering of an incident plane acoustic wave propagating in direction $\mathbf{d} \in \mathbb{R}^3, \|\mathbf{d}\| = 1$,

$$p_{\text{inc}}(\mathbf{x}, t) := \Psi(\mathbf{d} \cdot \mathbf{x} + ct), \quad \begin{matrix} \mathbf{x} \in \mathbb{R}^3 \\ t \in \mathbb{R} \end{matrix}, \tag{3.2.3.3}$$

$$\text{with smooth } \Psi : \mathbb{R} \rightarrow \mathbb{R}, \tag{3.2.3.4}$$

impinging on a **sound-soft** (*) object occupying $D \subset \mathbb{R}^3$.

(*) “Sound-soft” means that $p(t) = 0$ on $\Gamma := \partial D$ for all times t : the total pressure field p satisfies homogeneous Dirichlet boundary conditions on ∂D .

We assume that p_{inc} is causal: $p_{\text{inc}}(\mathbf{x}, t) = 0$ for $t \leq 0$ and \mathbf{x} in a neighborhood of D . To simplify the presentation, we also rescale units of space and time to achieve $c = 1$.

This scattering problem is modeled by an **exterior Dirichlet problem** for the unknown **scattered field** $u := p - p_{\text{inc}}$ on the unbounded spatial domain $\Omega := \mathbb{R}^3 \setminus \overline{D}$:

$$\frac{\partial^2 u}{\partial t^2} - \Delta_{\mathbf{x}} u = 0 \quad \text{in } \Omega \times]0, T[, \tag{3.2.3.5a}$$

$$u(\mathbf{x}, t) = -p_{\text{inc}}(\mathbf{x}, t) \quad \text{for } \mathbf{x} \in \partial D, \quad t \in]0, T[, \tag{3.2.3.5b}$$

$$u(\mathbf{x}, 0) = \frac{\partial u}{\partial t}(\mathbf{x}, 0) = 0 \quad \text{for } \mathbf{x} \in \Omega. \tag{3.2.3.5c}$$

§3.2.3.6 (Scattering boundary integral equations in Laplace domain) Since u is causal as a function of time and all the equations in (3.2.3.5) are linear, we can apply the Laplace transform in time and get the following parameterized family of boundary value problems for the transformed unknown $\hat{u}(\mathbf{x}, s) := (\mathcal{L}\{t \mapsto u(\mathbf{x}, t)\})(s), s \in \mathbb{C}^+$,

$$s^2 \hat{u}(\mathbf{x}, s) - \Delta_{\mathbf{x}} \hat{u}(\mathbf{x}, s) = 0 \quad \text{in } \Omega, \tag{3.2.3.7a}$$

$$\hat{u}(\mathbf{x}, s) = -\mathcal{T}_D \hat{p}_{\text{inc}}(\mathbf{x}, s), \quad \hat{p}_{\text{inc}} := \mathcal{L} p_{\text{inc}}, \quad \text{for } \mathbf{x} \in \partial D. \tag{3.2.3.7b}$$

For no $s \in \mathbb{C}^+$ the solution $\hat{u}(s)$ may suffer blow-up as $\|\mathbf{x}\| \rightarrow \infty$. Therefore we supplement (3.2.3.7) with decay conditions at ∞ , analogous to what we did in Section 1.1.7.

$$\hat{u}(\mathbf{x}, s) \rightarrow 0 \quad \text{for } \|\mathbf{x}\| \rightarrow \infty. \tag{3.2.3.8}$$

Note that (3.2.3.7) is an exterior Dirichlet boundary value problem (BVP) for the parameterized partial differential equation $-\Delta \hat{u}(s) + s^2 \hat{u}(s) = 0$. If the term $s^2 \hat{u}(s)$ was not present, we would already know a

way to solve it: As elaborated in § 1.3.6.3 in this case we can convert the BVP into an equivalent **indirect first-kind boundary integral equation** (1.3.6.4) for the unknown Neumann data. The only obstacle to doing this for the more general PDE (3.2.3.7a) is the missing fundamental solution. The next lemma will provide it.

Lemma 3.2.3.9. Fundamental solution for $L := -\Delta + s^2$

The fundamental solution for the second-order linear differential operator $Lu := -\Delta u + s^2 u$, $s \in \mathbb{C}^+$, in three dimensions is

$$G_s(\mathbf{x}, \mathbf{y}) := \frac{\exp(-s\|\mathbf{x} - \mathbf{y}\|)}{4\pi\|\mathbf{x} - \mathbf{y}\|}, \quad \mathbf{x} \neq \mathbf{y}. \quad (3.2.3.10)$$

Of course, for $s = 0$ we recover the fundamental solution (1.2.2.33) for the Laplacian $-\Delta$. Also notice that $\mathbf{x} \mapsto G_s(\mathbf{x}, \mathbf{y})$ decays exponentially for $\|\mathbf{x}\| \rightarrow \infty$.

Lemma 3.2.3.9 can be proved by a slight generalization of the computations presented in Ex. 1.2.2.24, see [STE09b]. We remark that all essential results of Section 1.2, in particular the representation formula from Thm. 1.2.4.5, and of Section 1.3, in particular the jump relations from Thm. 1.3.3.15, carry over to the more general differential operator L .

Thus, following the policy of Section 1.3.6 we represent $\widehat{u}(s)$ in Ω by means of the single layer potential acting on an **s -dependent** unknown density $\widehat{\phi}(s) \in H^{-\frac{1}{2}}(\partial\Omega)$, $\Gamma := \partial\Omega$:

$$\widehat{u}(x, s) = \Psi_{\text{SL}}^s(\widehat{\phi}(s))(x) \quad \text{in } \Omega, \quad \Psi_{\text{SL}}^s(\phi)(x) = \int_{\Gamma} \frac{\exp(-s\|\mathbf{x} - \mathbf{y}\|)}{4\pi\|\mathbf{x} - \mathbf{y}\|} \phi(\mathbf{y}) \, dS(\mathbf{y}), \quad \mathbf{x} \notin \Gamma. \quad (3.2.3.11)$$

We apply the Dirichlet trace operator T_D on $\Gamma := \partial D$ and take into account the prescribed Dirichlet data (3.2.3.5b), which yields the boundary integral equation (also given in variational form)

$$V(s)\widehat{\phi}(s) = -T_D \widehat{p}_{\text{inc}}(s) \quad \text{in } H^{\frac{1}{2}}(\partial\Omega) \quad (3.2.3.12)$$

$$\Updownarrow$$

$$\widehat{\phi}(s) \in H^{-\frac{1}{2}}(\partial\Omega): \quad a(s; \widehat{\phi}(s), \psi) := \int_{\Gamma} (V(s)\widehat{\phi}(s))(x) \psi(x) \, dS(x) = - \int_{\Gamma} \widehat{p}_{\text{inc}}(x, s) \psi(x) \, dS(x) \quad \forall \psi \in H^{-\frac{1}{2}}(\partial\Omega),$$

with the s -dependent single-layer boundary integral operator

$$V(s) : \begin{cases} H^{-\frac{1}{2}}(\partial\Omega) & \rightarrow H^{\frac{1}{2}}(\partial\Omega) \\ \phi & \mapsto (V(s)\phi)(x) := \int_{\Gamma} \frac{\exp(-s\|\mathbf{x} - \mathbf{y}\|)}{4\pi\|\mathbf{x} - \mathbf{y}\|} \phi(\mathbf{y}) \, dS(\mathbf{y}), \quad \mathbf{x} \in \Gamma. \end{cases} \quad (3.2.3.13)$$

§3.2.3.14 (Boundary element discretization → Section 1.5) As explained in Section 1.5 the s -dependent variational problem (3.2.3.12) set in $H^{-\frac{1}{2}}(\partial\Omega)$ is amenable to Galerkin boundary element discretization using piecewise constant trial and test functions on a surface mesh (\rightarrow Def. 1.5.1.4) \mathcal{G} of Γ : use $\mathcal{S}_1^{-1}(\mathcal{M})$ as trial and test space.

The main challenge faced when computing the entries of the Galerkin matrix arises from the singularity of the integral kernel. Up to a modulation with the continuous functions $(\mathbf{x}, \mathbf{y}) \mapsto \exp(-s\|\mathbf{x} - \mathbf{y}\|)$, this singularity is the same as the one for the single-layer boundary integral operator for the Laplacian $-\Delta$. Therefore, the techniques from Section 1.5.3 can be applied unchanged.

This gives us a family of linear systems of equations, parameterized with s :

$$\mathbf{V}(s)\vec{\phi}(s) = \vec{\rho}(s), \quad s \in \mathbb{C}^+, \quad (3.2.3.15)$$

with a dense boundary element Galerkin matrix $\mathbf{V}(s) \in \mathbb{C}^{N,N}$, $N := \dim \mathcal{S}_1^{-1}(\mathcal{M})$, and $\vec{\phi}$ standing for the basis expansion coefficient vector of the approximate solution.

§3.2.3.16 (Retarded potential integral equations) The left-hand side of (3.2.3.12) is a bilinear expression, a product, involving the s -dependent boundary integral operator $\mathbf{V}(s) \in L(H^{-\frac{1}{2}}(\partial\Omega), H^{\frac{1}{2}}(\partial\Omega))$ and the s -dependent density $\widehat{\phi}(s) \in H^{-\frac{1}{2}}(\partial\Omega)$. According to the rule “multiplication in Laplace domain corresponds to convolution in time domain” expressed in Thm. 3.1.4.3, the boundary integral equation (3.2.3.12) can be transformed back to time domain and we obtain a **convolution equation** for the **time-dependent** density $\phi : [0, T] \rightarrow H^{-\frac{1}{2}}(\partial\Omega)$

$$(k_V * \phi)(t) = -T_D u_{\text{inc}}(t), \quad t \in [0, T], \quad (3.2.3.17)$$

with kernel $k_V : [0, T] \rightarrow L(H^{-\frac{1}{2}}(\partial\Omega), H^{\frac{1}{2}}(\partial\Omega))$, whose Laplace transform is explicitly available from (3.2.3.13).

In fact, by the inverse Laplace transform, we can obtain an explicit formula for K_V and the convolution in (3.2.3.17). Recall the formal inverse Laplace transform of an exponential:

$$\mathcal{L}^{-1}(\{s \mapsto \exp(-s\tau)\})(t) = \delta(t - \tau), \quad \tau > 0,$$

where δ is the δ -distribution. This formula can be used to deal with the numerator of the fundamental solution $G_s(\mathbf{x}, \mathbf{y})$:

$$\mathcal{L}^{-1}(\{s \mapsto \mathbf{V}(s)\widehat{\phi}(s)\})(t) = \int_0^t \int_{\Gamma} \frac{\delta(t - \|\mathbf{x} - \mathbf{y}\|)}{4\pi\|\mathbf{x} - \mathbf{y}\|} \phi(\mathbf{y}, t) \, dS(\mathbf{y}) \, dt = \int_{\Gamma} \frac{\phi(\mathbf{y}, t - \|\mathbf{x} - \mathbf{y}\|)}{4\pi\|\mathbf{x} - \mathbf{y}\|} \, dS(\mathbf{y}),$$

where $\phi = \phi(\mathbf{y}, t) := \mathcal{L}^{-1}\{s \mapsto \widehat{\phi}(\mathbf{y}, s)\}$, $\mathbf{y} \in \Gamma$, $0 \leq t \leq T$, is a time-dependent surface density. Hence, the time-domain version of the integral equations (3.2.3.12) reads:

$$\int_{\Gamma} \frac{\phi(\mathbf{y}, t - \|\mathbf{x} - \mathbf{y}\|)}{4\pi\|\mathbf{x} - \mathbf{y}\|} \, dS(\mathbf{y}) = -p_{\text{inc}}(\mathbf{x}, t), \quad \mathbf{x} \in \Gamma, \quad 0 \leq t \leq T, \quad (3.2.3.18)$$

for obvious reasons called a **retarded-potential boundary integral equation**. From its solution ϕ the scattered pressure field can be reconstructed through

$$u(\mathbf{x}, t) = \int_{\Gamma} \frac{\phi(\mathbf{y}, t - \|\mathbf{x} - \mathbf{y}\|)}{4\pi\|\mathbf{x} - \mathbf{y}\|} \, dS(\mathbf{y}), \quad \mathbf{x} \in \Omega. \quad (3.2.3.19)$$

This is called a Kirchhoff representation formula; the scattered field is given by the superposition of fields radiated by time-dependent point sources on the boundary Γ of the scatterer.

3.3 Implicit-Euler Convolution Quadrature

3.3.1 Setting and Goal

Throughout we are given a **transfer function** $F(s) := \mathcal{L}f(s)$, the Laplace transform (\rightarrow Def. 3.1.3.6) of a causal function $f : \mathbb{R} \rightarrow \mathbb{C}$.

Assumption 3.3.1.1. Properties of transfer function

$F : \mathbb{C}^+ \rightarrow \mathbb{C}$ is analytic on the right half plane and satisfies the **decay condition**

$$\exists M > 0: |F(s)| \leq M|s|^\mu \quad \forall s \in \mathbb{C}^+ \quad \text{and some } \mu < -1. \quad (3.3.1.2)$$

notation: We write \mathcal{A} for the set of transfer functions satisfying Ass. 3.3.1.1

$$\mathcal{A} := \{F : \mathbb{C}^+ \rightarrow \mathbb{C} \text{ analytic} : \exists M > 0, \mu < -1 : |F(s)| \leq M|s|^\mu\}.$$

Recall that $\mathcal{G}_\tau = \tau\mathbb{Z}$ for some **timestep** $\tau > 0$ denotes an equidistant temporal grid. Also remember operational calculus introduced in Def. 3.1.4.5, here, for the sake of simplicity, used with $X = Y = \mathbb{C}$:

$$(F(\partial_t)g)(t) := (f * g)(t) = \int_0^t f(\xi)g(t - \xi) d\xi = \int_0^t f(t - \xi)g(\xi) d\xi.$$

The **goal** of convolution quadrature is to find a linear mapping

$$\text{CQ}_\tau : \mathcal{A} \rightarrow \{\text{causal sequences } \mathbb{Z} \rightarrow \mathbb{C}\},$$

depending on the timestep $\tau > 0$, such that

$$F(\partial_t)g|_{\mathcal{G}_\tau} \approx \text{CQ}_\tau(F) * g|_{\mathcal{G}_\tau} \quad \text{for } g \in \mathcal{CF}(\mathbb{C}) \cap C_0^\infty(\mathbb{R}, \mathbb{C}), \quad (3.3.1.3)$$

where \approx means the **convergence** requirement

$$\lim_{\tau \rightarrow 0} \left\| F(\partial_t)g|_{\mathcal{G}_\tau \cap [0, T]} - \text{CQ}_\tau(F) * g|_{\mathcal{G}_\tau \cap [0, T]} \right\| = 0 \quad \forall g \text{ "sufficiently smooth"}, \quad (3.3.1.4)$$

for some finite time $T > 0$ and a suitable (semi-)norm $\|\cdot\|$ on the space of causal sequences $\mathbb{Z} \rightarrow \mathbb{C}$.

We can rewrite (3.3.1.3) more explicitly as

$$\int_0^{n\tau} f(t - \xi)g(\xi) d\xi \approx \sum_{\ell=0}^n (\text{CQ}_\tau(F))_{n-\ell} g(\tau\ell), \quad n \in \mathbb{N}_0. \quad (3.3.1.5)$$

This conveys why the terms of the sequence $\text{CQ}_\tau(F) : \mathbb{Z} \rightarrow \mathbb{C}$ are called **convolution quadrature weights**. They will usually depend on both F and τ and, therefore we write $w_\ell^{F, \tau} \in \mathbb{C}, \ell \in \mathbb{Z}$:

$$\text{CQ}_\tau(F) =: \left(w_\ell^{F, \tau} \right)_{\ell \in \mathbb{Z}} \succ \text{CQ}(F) * g|_{\mathcal{G}_\tau} = \left(\sum_{\ell=0}^n w_{n-\ell}^{F, \tau} g_\ell \right)_{n \in \mathbb{Z}}, \quad g_\ell := g(\ell\tau).$$

We mention two desirable algebraic structural properties of CQ_τ :

- 1 **CQ** should preserve the neutral element of convolution (Ass. 3.3.1.1 violated!):

$$\text{CQ}_\tau(\{s \mapsto 1\}) = (\delta_{0, \ell})_{\ell \in \mathbb{Z}}, \quad (3.3.1.6)$$

② and CQ should be compatible with the convolution theorem for the Laplace transform

$$\text{CQ}_\tau(F_1 \cdot F_2) = \text{CQ}_\tau(F_1) * \text{CQ}_\tau(F_2) , \tag{3.3.1.7}$$

for transfer functions $F_1, F_2 : \mathbb{C}^+ \rightarrow \mathbb{C}$ complying with Ass. 3.3.1.1. Note that in (3.3.1.7) $*$ is the discrete convolution of causal sequences, see (3.1.2.12). Thus this formula is the discrete counterpart of the convolution theorem for the Laplace transform, Thm. 3.1.4.3,

$$(F_1 \cdot F_2)(\partial_t)g = (f_1 * f_2) * g = f_1 * (f_2 * g) = F_1(\partial_t)(F_2(\partial_t)g) .$$

Relationship (3.3.1.7) can be expressed by a **commuting diagram**.

$$\begin{array}{ccc} \mathcal{A} \times \mathcal{A} & \xrightarrow{\cdot} & \mathcal{A} \\ \text{(CQ}_\tau, \text{CQ}_\tau) \downarrow & & \downarrow \text{CQ}_\tau \\ \mathcal{CS} \times \mathcal{CS} & \xrightarrow{*} & \mathcal{CS} \end{array} , \quad \begin{array}{l} \cdot \triangleq \text{pointwise product,} \\ * \triangleq \text{discrete convolution Def. 3.1.2.3,} \end{array}$$

where, for the sake of brevity, we wrote \mathcal{CS} for the vector space of causal sequences $\mathbb{Z} \rightarrow \mathbb{C}$.

Remark 3.3.1.8 (Approximately solving convolution equations by convolution quadrature) As in § 3.1.1.25 let us consider a **convolution equation**

$$u \in \mathcal{CF}(\mathbb{C}) : [F(\partial_t)u(t) = (f * u)(t) =] \int_0^t f(t - \xi)u(\xi) = y(t) , \quad t \in \mathbb{R} , \tag{3.3.1.9}$$

for given causal $y \in \mathcal{CF}(\mathbb{C})$. By the convolution theorem for the Laplace transform

Theorem 3.1.4.3. Convolution theorem for Laplace transform

For Banach spaces X, Y and $g \in \mathcal{CF}(X), f \in \mathcal{CF}(L(X, Y))$ holds

$$\mathcal{L}(f * g)(s) = (\mathcal{L}f)(s)((\mathcal{L}g)(s)) , \quad s \in \mathbb{C}^+ .$$

we can lift (3.3.1.9) to the Laplace domain

$$\int_0^t f(t - \xi)u(\xi) = y(t) \quad \Leftrightarrow \quad F(s) \cdot (\mathcal{L}u)(s) = (\mathcal{L}y)(s) , \quad s \in \mathbb{C}^+ , \tag{3.3.1.10}$$

where the transfer function F is the Laplace transform of $f \in \mathcal{CF}(\mathbb{C})$.

Applying convolution quadrature to the convolution equation (3.3.1.9) converts it to a **discrete convolution equation**, cf. (3.1.2.15),

$$\text{seek } (u_n) : \mathbb{Z} \rightarrow \mathbb{C} \text{ causal: } \text{CQ}_\tau(F) * (u_n) = (y_n) := y|_{\mathcal{G}_\tau} , \tag{3.3.1.11}$$

set in the space containing causal sequences.

Assume that $F(s) \neq 0$ for all $s \in \mathbb{C}^+$. Then $s \mapsto F(s)^{-1}$ will also be analytic in \mathbb{C}^+ and the solution of the convolution equation can be obtained as

$$\mathcal{L}u(s) = F^{-1}(s) \cdot \mathcal{L}y(s) \quad \Leftrightarrow \quad u(t) = \mathcal{L}^{-1}(F^{-1} \cdot \mathcal{L}y)(t) . \tag{3.3.1.12}$$

The key observation is that the properties (3.3.1.6) and (3.3.1.7) enable an analogous formula on the discrete level

$$\begin{aligned} \text{CQ}_\tau(F) * (u_n) = (y_n) & \Leftrightarrow \text{CQ}_\tau(F^{-1}) * \text{CQ}_\tau(F) * (u_n) = \text{CQ}_\tau(F^{-1}) * (y_n) \\ \stackrel{(3.3.1.7)}{\Leftrightarrow} & \text{CQ}_\tau(F^{-1} \cdot F) * (u_n) = \text{CQ}_\tau(F^{-1}) * (y_n) \\ \stackrel{(3.3.1.6)}{\Leftrightarrow} & (u_n) = \text{CQ}_\tau(F^{-1}) * (y_n) . \end{aligned} \tag{3.3.1.13}$$

Thus, if convolution quadrature satisfies the structural properties (3.3.1.6) and (3.3.1.7), then a convolution equation can be solved approximately by a simple discrete convolution. ┘

3.3.2 Derivation of Implicit Euler CQ

Let $F : \mathbb{C}^+ \rightarrow \mathbb{C}$ be a transfer function satisfying Ass. 3.3.1.1 related to a causal function/distribution f through Laplace transform (\rightarrow Def. 3.1.3.6) and its inverse (\rightarrow Thm. 3.1.3.13)

$$F(s) = \int_0^\infty f(t)e^{-st} dt \Leftrightarrow f(t) = \frac{1}{2\pi i} \int_{\sigma+i\mathbb{R}} F(s)e^{st} ds, \quad \sigma > 0.$$

§3.3.2.1 (Reduction to ordinary differential equations) Using the Laplace inversion formula and boldly changing the order of integration permits us to rewrite convolution

$$\begin{aligned} F(\partial_t)g(t) &= (f * g)(t) = \int_0^t f(t-\zeta)g(\zeta) d\zeta \\ &= \int_0^t \frac{1}{2\pi i} \int_{\sigma+i\mathbb{R}} F(s)e^{s(t-\zeta)} ds \cdot g(\zeta) d\zeta = \frac{1}{2\pi i} \int_{\sigma+i\mathbb{R}} F(s) \cdot \underbrace{\int_0^t e^{s(t-\zeta)} g(\zeta) d\zeta}_{=:y(s;t)} ds \end{aligned}$$

Surprisingly, the highlighted integral expression, in the sequel abbreviated by $y(s;t)$ is related to a family of simple initial value problems for ordinary differential equations.

Lemma 3.3.2.2. Variation of constants formula

For a continuous causal function $g : \mathbb{R} \rightarrow \mathbb{C}$ and any $s \in \mathbb{C}$ the solution $t \mapsto y(s;t)$ of the initial value problem (IVP)

$$\dot{y}(t) = sy(t) + g(t), \quad t \in \mathbb{R}, \quad y(0) = 0, \tag{3.3.2.3}$$

has the integral representation

$$y(s;t) = \int_0^t e^{s(t-\zeta)} g(\zeta) d\zeta. \tag{3.3.2.4}$$

Proof. The initial value problem (3.3.2.3) for a simple scalar linear ordinary differential equation has a solution $y : \mathbb{R} \rightarrow \mathbb{C}$. We make the transformation

$$z(t) = e^{-st}y(t) \Leftrightarrow y(t) = e^{st}z(t), \quad t \in \mathbb{R}.$$

By the product rule we find

$$\dot{z}(t) = -se^{-st}y(t) + e^{-st}\dot{y}(t) = -se^{-st}y(t) + e^{-st}(sy(t) + g(t)) = e^{-st}g(t).$$

$$\blacktriangleright \quad z(t) = \int_0^t e^{-s\zeta} g(\zeta) d\zeta \Leftrightarrow y(t) = \int_0^t e^{s(t-\zeta)} g(\zeta) d\zeta, \quad t \in \mathbb{R}.$$

□

As a consequence the convolution $F(\partial_t)g$ can be written as a contour integral involving solutions of a family of linear initial value problems

$$F(\partial_t)g(t) = \frac{1}{2\pi i} \int_{\sigma+i\mathbb{R}} F(s) y(s;t) ds, \quad t \in \mathbb{R}, \quad \sigma > 0. \tag{3.3.2.5}$$

┘

Idea: Use numerical integration of the IVPs for $\dot{y} = sy + g(t)$ on the temporal grid \mathcal{G}_τ , producing a sequence



$$(y_n(s))_{n \in \mathbb{Z}}: y_n(s) \approx y(s; n\tau), \tag{3.3.2.6}$$

and then, inspired by (3.3.2.5), approximate

$$(F(\partial_t)g)(n\tau) \approx \frac{1}{2\pi i} \int_{\sigma+i\mathbb{R}} F(s) y_n(s) ds \tag{3.3.2.7}$$

§3.3.2.8 (Implicit Euler (IE)/backward Euler timestepping → [NumCSE Section 11.2.2]) The implicit Euler method converts the ordinary differential equation (ODE) $\dot{y} = g(y, t)$ into a difference equation by using a **backward difference quotient** to approximate the temporal derivative

$$\dot{y} = g(y, t) \quad \triangleright \quad \frac{y(t) - y(t - \tau)}{\tau} \approx g(y(t), t) \quad \text{with timestep } \tau > 0,$$

and restricting the difference quotient to the temporal grid \mathcal{G}_τ :

$$y_n - y_{n-1} = \tau g(y_n, t_n), \quad t_n := \tau n, \quad k \in \mathbb{Z}. \tag{3.3.2.9}$$

Thinking of timestepping $y_{n-1} \rightarrow y_n$, given y_{n-1} this is an equation for y_n . Consult [NumCSE Rem. 11.2.2.3] for an explanation why (3.3.2.9) has a unique solution y_n provided that g is differentiable w.r.t y and the timestep τ is sufficiently small. ┘

§3.3.2.10 (Implicit Euler for scalar linear ODEs) We elaborate the above idea in the concrete case of numerical integration by means of the implicit Euler method. We apply implicit Euler timestepping (3.3.2.9) with uniform timestep size $\tau > 0$ to (3.3.2.3) (ODE $\dot{y} = sy + g(t)$), that is, with the right-hand side function $f(t, y) := sy + g(t)$ and $y_k(s) := y(k\tau) = 0$ for all $k < 0$. As in (3.3.2.6) we write $(y_n(s))$ for the resulting causal sequence, which, if $\tau s \neq 1$, it is defined by $(n \in \mathbb{N})$

$$y_n(s) = y_{n-1}(s) + \tau s y_n(s) + \tau g(t_n), \quad n \in \mathbb{N}_0, \quad y_{-1}(s) = 0$$

$$\Updownarrow$$

$$y_n(s) = (1 - \tau s)^{-1} (y_{n-1}(s) + \tau g(\tau n)).$$

►
$$y_n(s) = \tau \sum_{\ell=0}^n (1 - \tau s)^{-(\ell+1)} g_{n-\ell}, \quad n \in \mathbb{N}, \quad g_\ell := g(\tau \ell), \tag{3.3.2.11}$$

because we have $g_0 = g(0) = 0$ for the causal continuous function g . ┘

If $\sigma < \frac{1}{\tau}$, then we can plug (3.3.2.11) into (3.3.2.7):

$$F(\partial_t)g(n\tau) \approx \frac{1}{2\pi i} \int_{\sigma+i\mathbb{R}} F(s) y_n(s) ds = \frac{1}{2\pi i} \int_{\sigma+i\mathbb{R}} F(s) \tau \sum_{\ell=0}^n (1 - \tau s)^{\ell+1} g_{n-\ell} ds \tag{3.3.2.12}$$

$$= \sum_{\ell=0}^n \frac{\tau}{2\pi i} \int_{\sigma+i\mathbb{R}} F(s) (1 - \tau s)^{-(\ell+1)} ds \cdot g_{n-\ell}.$$

Strikingly, this amounts to a discrete convolution:

$$(F(\partial_t)g)(n\tau) \approx \sum_{\ell=0}^n w_\ell^{F,\tau} \cdot g_{n-\ell} \quad \text{with} \quad w_\ell^{F,\tau} := \frac{\tau}{2\pi i} \int_{\sigma+i\mathbb{R}} F(s) (1 - \tau s)^{-(\ell+1)} ds. \tag{3.3.2.13}$$

We have found our first convolution quadrature scheme!

Definition 3.3.2.14. Implicit Euler convolution quadrature (IE-CQ)

Given the transfer function $F : \mathbb{C}^+ \rightarrow \mathbb{C}$, **convolution quadrature** based on **implicit Euler** timestepping with timestep $\tau > 0$ is defined as ($0 < \sigma < \tau^{-1}$)

$$\text{CQ}_\tau^{\text{IE}}(F) := \left(w_\ell^{F,\tau} := \frac{\tau}{2\pi i} \int_{\sigma+i\mathbb{R}} F(s)(1-\tau s)^{-(\ell+1)} ds \right)_{\ell \in \mathbb{N}_0} .$$

Remark 3.3.2.15 (Well-defined IE-CQ) An elementary estimate yields $|1 - \tau s| > \tau |s| - 1$, which implies $|(1 - \tau s)^{-(\ell+1)}| < (\tau |s| - 1)^{-(\ell+1)}$. Thus under the decay condition from Ass. 3.3.1.1, the improper contour integrals in the definition of $\text{CQ}_\tau^{\text{IE}}(F)$ are always well-defined for $\ell \in \mathbb{N}_0$, if $\sigma < \tau^{-1}$. \lrcorner

Remark 3.3.2.16 (Convolution quadrature based on explicit Euler timestepping?) Another simple timestepping scheme is the explicit Euler method which replaces the temporal derivative with a forward difference quotient, see [NumCSE Section 11.2.1]:

$$\dot{y} = g(y, t) \quad \triangleright \quad \frac{y(t + \tau) - y(t)}{\tau} \approx g(y(t), t) \quad \text{with timestep } \tau > 0 .$$

For the initial value problem (3.3.2.3) and uniform timestep $\tau > 0$ this yields the recurrence

$$y_{n+1}(s) = y_n(s) + \tau s y_n(s) + \tau g_n, \quad y_0 = 0 \quad \blacktriangleright \quad y_n(s) = \tau \sum_{\ell=1}^n (1 + \tau s)^\ell g_{n-\ell} \quad (3.3.2.17)$$

This would lead to convolution weights defined by

$$w_\ell := \frac{\tau}{2\pi i} \int_{\sigma+i\mathbb{R}} F(s)(1 + \tau s)^\ell ds, \quad \ell \in \mathbb{N}, \quad w_0 := 0 .$$

Yet, $|1 + \tau s| \geq \tau |s| - 1$ such that the improper integrals will in general diverge for almost all $\ell \in \mathbb{N}$, unless F decays exponentially for $|s| \rightarrow \infty$, which cannot be expected. Hence, explicit Euler timestepping is not suitable for defining a convolution quadrature scheme. \lrcorner

§3.3.2.18 (CQ weights by integration over compact contour) To manipulate the formula for the convolution quadrature weights for IE-CQ from Def. 3.3.2.14

$$w_\ell^{F,\tau} := \frac{\tau}{2\pi i} \int_{\sigma+i\mathbb{R}} F(s)(1 - \tau s)^{-(\ell+1)} ds, \quad \ell \in \mathbb{N}_0, \quad 0 < \sigma < \frac{1}{\tau}, \quad (3.3.2.19)$$

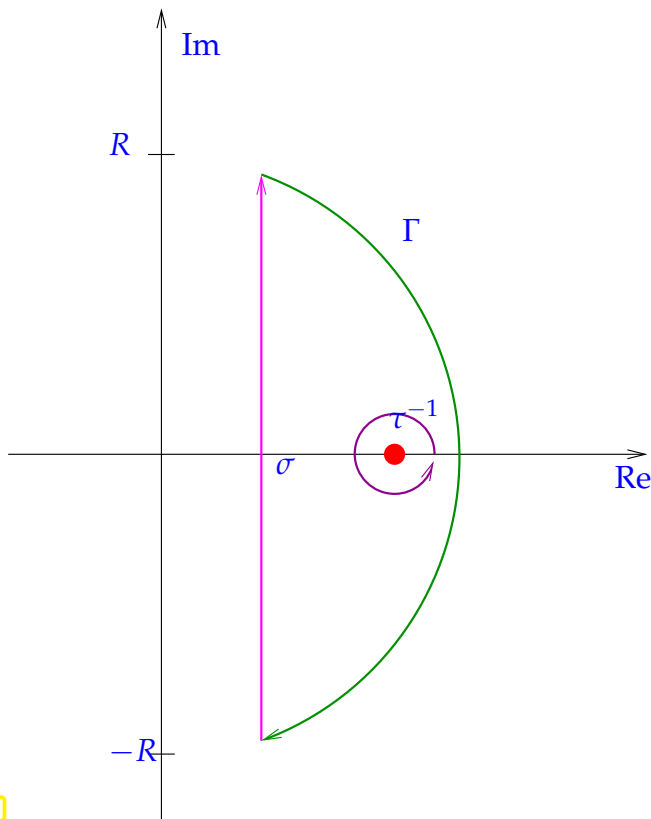


Fig. 128

Assume $\sigma < 1/\tau$. Note that the integrand

$$s \mapsto F(s)(1 - \tau s)^{-(\ell+1)}$$

is analytic in $\mathbb{C}^+ \setminus \{\tau^{-1}\}$. Thus by the **Cauchy integral theorem** Thm. 3.1.3.16 its path integral over the contour

$$\begin{aligned} \Gamma &:= \Gamma_\sigma \cup \Gamma_R \cup \Gamma_r, \\ \Gamma_\sigma &:= \sigma + i[-R, R], \\ \Gamma_R &:= \{s : |s| = R, \operatorname{Re} z \geq \sigma\}, \\ \Gamma_r &:= \{s : |s - \tau^{-1}| = r\}, \end{aligned}$$

with $r, R > 0$, $R > \tau^{-1} + r$ and suitable orientations of the pieces, vanishes. Thanks to the decay properties of F from Ass. 3.3.1.1, we have

$$\int_{\Gamma_R} F(s)(1 - \tau s)^{-(\ell+1)} ds \rightarrow 0 \quad \text{for } R \rightarrow \infty.$$

Hence, the convolution quadrature weight can also be computed by integrating over a (small) circle centered at τ^{-1} and oriented counter-clockwise:

$$\begin{aligned} w_\ell^{F,\tau} &:= \frac{\tau}{2\pi i} \int_{\sigma+i\mathbb{R}} F(s)(1 - \tau s)^{-(\ell+1)} ds = -\frac{\tau}{2\pi i} \int_{|s-\frac{1}{\tau}|=r} F(s)(1 - \tau s)^{-(\ell+1)} ds \quad (3.3.2.20) \\ &= -\frac{1}{2\pi i \tau^\ell} \int_{|s-\frac{1}{\tau}|=r} \frac{F(s)}{(\tau^{-1} - s)^{\ell+1}} ds. \end{aligned}$$

Note that the formula (3.3.2.20) is well-defined for any F analytic in \mathbb{C}^+ ; we can dispense with the decay condition stated in Ass. 3.3.1.1!

┘

§3.3.2.21 (CQ weights through Taylor expansion) A fundamental result of complex analysis reveals another benefit of switching to an integration contour surrounding τ^{-1} .

Theorem 3.3.2.22. Cauchy integral formula [Rem84, §7.2]

If $g : D \subset \mathbb{C} \rightarrow \mathbb{C}$ is analytic in D , $c \in D$, and $B := \{s : |s - z| \leq r\} \subset D$ for some $r > 0$, then

$$g(z) = \frac{1}{2\pi i} \int_{\partial B} \frac{g(s)}{s - z} ds \quad \forall z \in B,$$

where the integral is a complex contour integral and the circle ∂B is oriented counterclockwise.

Proof. Fix $z \in D$. Then

$$s \in D \setminus \{z\} \mapsto \frac{g(s) - g(z)}{s - z}$$

is analytic and has the limit for $s \rightarrow z$ exists, which means that this function is analytic *everywhere* in D . By the Cauchy integral theorem Thm. 3.1.3.16 and for $B \subset D$

$$0 = \int_{\partial B} \frac{g(s) - g(z)}{s - z} ds = \int_{\partial B} \frac{g(s)}{s - z} ds - 2\pi i g(z),$$

from which the assertion follows immediately. □

By formal differentiation under the integral we obtain a similar representation of all derivatives of g :

Corollary 3.3.2.23. Cauchy differentiation formula [Rem84, §7.3.4]

If $g : D \subset \mathbb{C} \rightarrow \mathbb{C}$ is analytic in D , $c \in D$, and $B := \{z : |z - c| \leq r\} \subset D$ for some $r > 0$, then the ℓ -th derivative of g can be computed as the contour integral

$$g^{(\ell)}(z) = \frac{\ell!}{2\pi i} \int_{\partial B} \frac{g(s)}{(s - z)^{\ell+1}} ds \quad \forall z \in B, \ell \in \mathbb{N}_0.$$

Remember that $s \in \mathbb{C}^+ \mapsto F(s)$ is analytic, which permits us to this formula with $g = F$, $z = 1/\tau$, $B := \{z \in \mathbb{C} : |z - \tau^{-1}| = r\}$, $r < \tau^{-1} - \sigma$:

$$w_\ell^{F,\tau} = -\frac{1}{2\pi i \tau^\ell} \int_{|s - \frac{1}{\tau}| = r} \frac{F(s)}{(\tau^{-1} - s)^{\ell+1}} ds = \frac{(-1)^\ell}{\ell! \tau^\ell} F^{(\ell)}(1/\tau) = \frac{1}{\ell!} \frac{d^\ell}{dz^\ell} \left\{ z \mapsto F\left(\frac{1-z}{\tau}\right) \right\} \Bigg|_{z=0}.$$

Recall the local Taylor expansion for a function g that is analytic in an neighborhood of $c \in \mathbb{C}$:

$$g(z) = \sum_{\ell=0}^{\infty} \frac{1}{\ell!} g^{(\ell)}(c) (z - c)^\ell \quad \text{for all } z : |z - c| \text{ sufficiently small.}$$

Obviously, the convolution weights are the Taylor coefficients of $\{z \mapsto F(\frac{1-z}{\tau})\}$ when expanded around $z = 0$.

Lemma 3.3.2.24. Convolution quadrature weights are Taylor expansion coefficients

If $F : \mathbb{C}^+ \rightarrow \mathbb{C}$ is analytic and complies with Ass. 3.3.1.1, then $z \mapsto F(\frac{1-z}{\tau})$ is a *generating function* for the convolution quadrature weights from Def. 3.3.2.14, that is,

$$F\left(\frac{1-z}{\tau}\right) = \sum_{\ell=0}^{\infty} w_\ell^{F,\tau} z^\ell \quad \text{for } |z| < 1. \tag{3.3.2.25}$$

Note that the power series in (3.3.2.25) converges for $|z| < 1$, because $z \mapsto F(\frac{1-z}{\tau})$ is analytic for $\text{Re } z < 1$. Also note that this formula makes sense for any F that is analytic in a neighborhood of 1 and, thus, extends Def. 3.3.2.14, which requires decay properties of F . ┘

Remark 3.3.2.26 (Real-valued convolution quadrature weights) If $F(s) \in \mathbb{R}$ for $s \in \mathbb{R}$, then

$$G(z) := F\left(\frac{1-z}{\tau}\right) \in \mathbb{R}, \quad \text{if } z \in \mathbb{R}.$$

Hence, all derivatives $G^{(m)}(0)$ will be real and so will be the convolution quadrature weights $w_\ell^{F,\tau}$ for IE-CQ. ┘

EXAMPLE 3.3.2.27 (Direct computation of convolution quadrature weights) For simple transfer functions F Lemma 3.3.2.24 paves the way for computing the convolution quadrature weights $w_\ell^{F,\tau}$, $\ell \in \mathbb{N}_0$, by Taylor expansion/repeated differentiation. We elaborate this for two examples

(I) If $F(s) = s^\mu$, $\mu \in \mathbb{R} \setminus \mathbb{N}_0$, then

$$\begin{aligned} G(z) &:= F\left(\frac{1-z}{\tau}\right) = \tau^{-\mu}(1-z)^\mu, \\ \Rightarrow G^{(\ell)}(z) &= \tau^{-\mu}(-1)^\ell \mu(\mu-1)\cdots(\mu-\ell+1)(1-z)^{\mu-\ell}, \\ \blacktriangleright G^{(\ell)}(0) &= \tau^{-\mu}(-1)^\ell \mu(\mu-1)\cdots(\mu-\ell+1), \\ \blacktriangleright G(z) &= \sum_{\ell=0}^{\infty} \tau^{-\mu}(-1)^\ell \prod_{k=0}^{\ell-1} \frac{\mu-k}{k+1} z^\ell. \end{aligned}$$

Thus we find the IE-CQ weights

$$w_0^{F,\tau} = \tau^{-\mu}, \quad w_\ell^{F,\tau} = \tau^{-\mu}(-1)^\ell \prod_{k=0}^{\ell-1} \frac{\mu-k}{k+1}, \quad \ell \in \mathbb{N}. \tag{3.3.2.28}$$

(II) For $F(s) = (s^2 + \omega^2)^{-1}$, $\omega > 0$, we rely on a **factorization approach**:

$$\begin{aligned} F\left(\frac{1-z}{\tau}\right) &= \frac{1}{\left(\frac{1-z}{\tau}\right)^2 + \omega^2} = \frac{1}{\frac{1-z}{\tau} - i\omega} \cdot \frac{1}{\frac{1-z}{\tau} + i\omega} \\ &= \frac{\tau^2}{1 + \omega^2\tau^2} \cdot \left(\sum_{n=0}^{\infty} (1 - i\omega\tau)^{-n} z^n\right) \cdot \left(\sum_{n=0}^{\infty} (1 + i\omega\tau)^{-n} z^n\right), \end{aligned}$$

where the last step employed the geometric series. By the Cauchy product formula for power series, cf. Thm. 3.1.4.18, we obtain the convolution quadrature weights by discrete convolution:

$$w_\ell^{F,\tau} = \frac{\tau^2}{1 + \omega^2\tau^2} \cdot \sum_{n=0}^{\ell} (1 - i\omega\tau)^{-n+\ell} (1 + i\omega\tau)^{-\ell}, \quad \ell \in \mathbb{N}_0. \tag{3.3.2.29}$$

┘

3.3.3 Properties of implicit-Euler Convolution Quadrature

Does the convolution quadrature scheme as introduced in the previous section (\rightarrow Def. 3.3.2.14, Lemma 3.3.2.24) satisfy the crucial properties (3.3.1.6) and (3.3.1.7)?

- ❶ We consider the constant transfer function $F(s) = 1$ and use Lemma 3.3.2.24 that obviously gives $w_\ell^{\{s \mapsto 1\},\tau} = \delta_{\ell,0}$, which is (3.3.1.6). This is the neutral element of discrete convolution.
- ❷ Given two analytic transfer functions $F_1, F_2 : \mathbb{C}^+ \rightarrow \mathbb{C}$, we appeal to Lemma 3.3.2.24

$$F_i\left(\frac{1-z}{\tau}\right) = \sum_{\ell=0}^{\infty} w_\ell^{F_i,\tau} z^\ell, \quad i = 1, 2, \quad (F_1 \cdot F_2)\left(\frac{1-z}{\tau}\right) = \sum_{\ell=0}^{\infty} w_\ell^{F_2 \cdot F_1,\tau} z^\ell.$$

The **Cauchy product** formula for power series immediately gives

$$\begin{aligned} (F_1 \cdot F_2)\left(\frac{1-z}{\tau}\right) &= F_1\left(\frac{1-z}{\tau}\right) \cdot F_2\left(\frac{1-z}{\tau}\right) = \left(\sum_{\ell=0}^{\infty} w_\ell^{F_1,\tau} z^\ell\right) \cdot \left(\sum_{\ell=0}^{\infty} w_\ell^{F_2,\tau} z^\ell\right) \\ &= \sum_{\ell=0}^{\infty} \left(\sum_{k=0}^{\ell} w_{\ell-k}^{F_1,\tau} w_k^{F_2,\tau}\right) z^\ell. \end{aligned}$$

Comparing Taylor coefficients we conclude

$$w_\ell^{F_2 \cdot F_2, \tau} = \sum_{k=0}^{\ell} w_{\ell-k}^{F_1, \tau} w_k^{F_2, \tau} \Leftrightarrow \text{CQ}_\tau^{\text{IE}}(F_1 \cdot F_2) = \text{CQ}_\tau^{\text{IE}}(F_1) * \text{CQ}_\tau^{\text{IE}}(F_2) \hat{=} (3.3.1.7). \quad (3.3.3.1)$$

§3.3.3.2 (Continuous-in-time (c.i.t.) convolution quadrature [Say16, Sect. 4.4]) A new perspective is opened by considering an alternative motivation for implicit Euler convolution quadrature. Remember the **shift operator**

$$T_\tau : \mathcal{CF}(\mathbb{C}) \rightarrow \mathcal{CF}(\mathbb{C}) \quad , \quad (T_\tau g)(t) := g(t - \tau) \quad , \quad \tau > 0. \quad (3.3.3.3)$$

Also recall the following correspondences for the Laplace transform:

	time domain	Laplace domain
Derivative:	$\frac{d}{dt}$	$s \cdot$
Backward difference quotient:	$\frac{1 - T_\tau}{\tau} \approx \frac{d}{dt}$	$\frac{1 - \exp(-s\tau)}{\tau} \approx s \cdot$

We point out that the backward difference quotient is the approximation of the derivative underlying the implicit Euler timestepping scheme, cf. § 3.3.2.8.

We define a modified transfer function

$$F_\tau(s) := F\left(\frac{1 - \exp(-s\tau)}{\tau}\right) \quad , \quad F_\tau : \mathbb{C}^+ \rightarrow \mathbb{C} \quad \text{analytic.} \quad (3.3.3.4)$$

The formula for the convolution quadrature weights $w_\ell^{F, \tau}$ from Lemma 3.3.2.24,

$$F\left(\frac{1 - z}{\tau}\right) = \sum_{\ell=0}^{\infty} w_\ell^{F, \tau} z^\ell \quad \text{for } |z| < 1, \quad (3.3.2.25)$$

with $z := e^{-s\tau}$ gives us

$$F_\tau(s) = F\left(\frac{1 - \exp(-s\tau)}{\tau}\right) = \sum_{\ell=0}^{\infty} w_\ell^{F, \tau} e^{-s\tau\ell}.$$

Recall the Laplace transform of a shifted δ -distribution

$$\mathcal{L}\{t \mapsto \delta(t - \tau)\}(s) = \int_{\mathbb{R}} \delta(t - \tau) \exp(-st) dt = \exp(-s\tau). \quad (3.3.3.5)$$

This gives us the time-domain counterpart of F_τ as a causal distribution:

$$f_\tau(t) := (\mathcal{L}^{-1} F_\tau)(t) = \sum_{\ell=0}^{\infty} w_\ell^{F, \tau} \delta(t - \ell\tau). \quad (3.3.3.6)$$

Convolution with this **comb function** is straightforward:

$$F_\tau(\partial_t)g = (f_\tau * g)(t) = \sum_{\ell=0}^{\infty} w_\ell^{F, \tau} g(t - \ell\tau) \quad (3.3.3.7)$$

Using the definition of convolution quadrature, this formula reveals that

$$\text{CQ}_\tau^{\text{IE}}(F) * g|_{\mathcal{G}_\tau} = F_\tau(\partial_t)g|_{\mathcal{G}_\tau}. \quad (3.3.3.8)$$

Continuous-in-time (c.i.t.) convolution quadrature

Implicit-Euler convolution quadrature realizes (continuous) operational calculus with

$$F(s) \text{ replaced with } F_\tau(s) := F\left(\frac{1 - \exp(-s\tau)}{\tau}\right), s \in \mathbb{C}^+.$$

For $t \mapsto (F_\tau(\partial_t)g)(t)$ we sometimes write $t \mapsto (\text{CQ}_\tau^{\text{IE}}(F) * g)(t)$ and call this the continuous-in-time (implicit Euler) convolution quadrature. We regard $\text{CQ}_\tau^{\text{IE}}(F) *$ as mapping a causal function on \mathbb{R} to another causal function on \mathbb{R} .

Moreover, the formula (3.3.3.8) again confirms (3.3.1.7) for IE-CQ as a simple consequence of the obvious fact $(F_1 \cdot F_2)_\tau = F_{1,\tau} \cdot F_{2,\tau}$ and of the convolution theorem for the Laplace transform Thm. 3.1.4.3. \square

Remark 3.3.3.10 (“Differentiation theorem” for convolution quadrature) Consider the transfer function $F(s) = s$, for which we have by the differentiation formula for the Laplace transform (\rightarrow Thm. 3.1.3.22):

$$F(s) := s \quad \Rightarrow \quad F(\partial_t)g(t) = \frac{dg}{dt}(t), \quad t \in \mathbb{R},$$

see also (3.1.4.8). The corresponding convolution quadrature is straightforward by Lemma 3.3.2.24:

$$F\left(\frac{1-z}{\tau}\right) = \frac{1-z}{\tau} \Leftrightarrow \left(\text{CQ}_\tau^{\text{IE}}(s \mapsto s)\right)_\ell = \begin{cases} 1/\tau & \text{for } \ell = 0, \\ -1/\tau & \text{for } \ell = 1, \\ 0 & \text{else.} \end{cases}$$

This means that convolution quadrature is reduced to applying the **backward difference quotient**. Adopting the continuous-in-time point of view, for any causal $g : \mathbb{R} \rightarrow \mathbb{C}$

$$(\text{CQ}_\tau^{\text{IE}}(\{s \mapsto s\}) * g)(t) = \frac{g(t) - g(t - \tau)}{\tau} = \left(\frac{\text{Id} - T_\tau}{\tau} g\right)(t), \quad t \in \mathbb{R}, \tag{3.3.3.11}$$

where we have used the shift operator

$$T_\tau : \mathcal{CF}(\mathbb{C}) \rightarrow \mathcal{CF}(\mathbb{C}) \quad , \quad (T_\tau g)(t) := g(t - \tau), \quad \tau > 0.$$

The right-hand side in (3.3.3.11) can be regarded as an approximation of $\frac{dg}{dt}$ in the points of the temporal mesh \mathcal{G}_τ .

Generalizing these considerations we immediately get the convolution quadratures induced by powers as transfer functions

$$(\text{CQ}_\tau^{\text{IE}}(\{s \mapsto s^m\}) * g)(t) = \left(\left(\frac{\text{Id} - T_\tau}{\tau}\right)^m g\right)(t). \tag{3.3.3.12}$$

Again, we recognize a backward difference quotient approximations of $g^{(m)}$. For instance, in the case $m = 2$, we find

$$(\text{CQ}_\tau^{\text{IE}}(\{s \mapsto s^2\}) * g)(t) = \left(\left(\frac{\text{Id} - T_\tau}{\tau}\right)^2 g\right)(t) = \frac{g(t) - 2g(t - \tau) + g(t - 2\tau)}{\tau^2} = \frac{d^2g}{dt^2}(t) + O(\tau) \quad \text{for } \tau \rightarrow 0$$

which is a second backward difference quotient. Finally, we can combine these formulas with (3.3.1.7) and get

$$\text{CQ}_\tau^{\text{IE}}(\{s \mapsto s^m F(s)\}) * g = \text{CQ}_\tau^{\text{IE}}(F) * \underbrace{\left(\frac{\text{Id} - T_\tau}{\tau}\right)^m g}_{\text{approximation of } g^{(m)}}, \tag{3.3.3.13}$$

a CQ-counterpart of Thm. 3.1.3.22. ┘

Remark 3.3.3.14 (Polynomially growing transfer functions) What if F fails to satisfy the decay conditions of Ass. 3.3.1.1, but still has polynomially bounded growth:

$$\exists M > 0: |F(s)| \leq M|s|^\mu \quad \forall s \in \mathbb{C}^+ \quad \text{and some } \mu \in \mathbb{R}. \quad (3.3.3.15)$$

If $\mu \geq -1$ we can pick $m \in \mathbb{N}, m > \mu + 1$, and apply (3.3.3.13).

$$\text{CQ}_\tau^{\text{IE}}(F) * g = \text{CQ}_\tau^{\text{IE}}(\{s \mapsto \frac{F(s)}{s^m}\} \cdot \{s \mapsto s^m\}) * g = \text{CQ}_\tau^{\text{IE}}(\{s \mapsto \frac{F(s)}{s^m}\}) * \left(\frac{\text{Id} - T_\tau}{\tau}\right)^m g. \quad (3.3.3.16)$$

Note that that function $\{s \mapsto \frac{F(s)}{s^m}\}$ satisfies Ass. 3.3.1.1 so that all consideration of this section apply. Compare ② in § 3.1.3.21. ┘

3.3.4 Convergence

This section present quantitative results about the asymptotic convergence of convolution quadrature as the timestep $\tau \rightarrow 0$. In particular we are interested in the maximum error at points of the temporal grid in a finite time interval $[0, T], T > 0$:

$$\text{err}(\tau) := \max_{n=0, \dots, N} \left| (F(\partial_t)g)(\tau n) - (\text{CQ}_\tau^{\text{IE}}(F) * g|_{\mathcal{G}_\tau})_n \right|, \quad \tau := T/N, \quad N \in \mathbb{N}, \quad (3.3.4.1)$$

for a given causal function $g : \mathbb{R} \rightarrow \mathbb{C}$. We first report some empirical results in order to see what kind of convergence can be expected.

EXPERIMENT 3.3.4.2 (Convergence of implicit Euler convolution quadrature) Throughout this experiment we consider $F(s) = \frac{1}{s}$, which corresponds to **Abel integral operator**, cf. (3.2.1.6),

$$F(\partial_t)g(t) = \frac{1}{\sqrt{\pi}} \int_0^t \frac{g(\xi)}{\sqrt{t-\xi}} d\xi.$$

We choose $T = 1$.

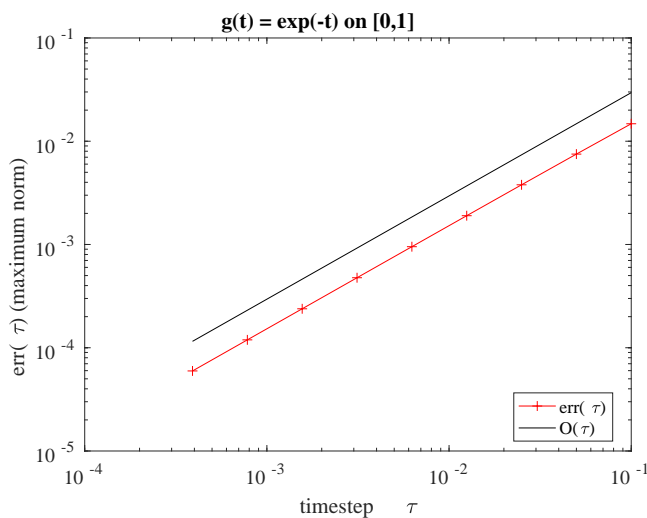


Fig. 12

- ① We consider $g(t) = 1 - e^{-t}, t \geq 0$, and find $F(\partial_t)g(t) = \frac{2}{\sqrt{\pi}}(\sqrt{t} - F_D(\sqrt{t}))$, where F_D is the Dawson function

$$F_D(t) = e^{-t^2} \int_0^t e^{\zeta^2} d\zeta.$$

- ◁ We observe **algebraic convergence** of order 1:

$$\text{err}(\tau) = O(\tau) \quad \text{for } \tau \rightarrow 0.$$

(Error points located close to a line with slope 1 in doubly-logarithmic plot)

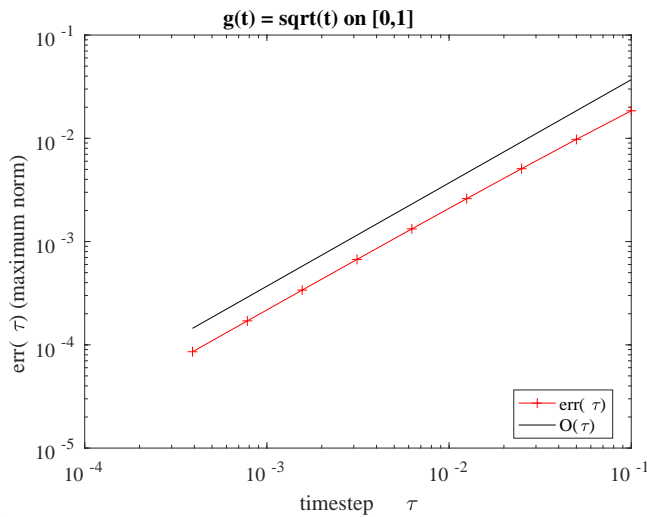


Fig. 13

- ② Now we choose the non-smooth $g(t) = \sqrt{t}$, which implies $F(\partial_t)g(t) = \sqrt{\pi t}/2$.
- ◁ Though g is not continuously differentiable on $[0, 1]$, we still observe algebraic convergence of order 1:

$$\text{err}(\tau) = O(\tau) \quad \text{for } \tau \rightarrow 0.$$

In both cases we observe perfect algebraic convergence with rate 1 as $\tau \rightarrow 0$. ┘

We provide a rigorous justification of the convergence observed in Exp. 3.3.4.2 for the case $X = \mathbb{C}$ and assuming at most polynomial growth of F .

Assumption 3.3.4.3. Polynomial growth of F

We assume that $F : \mathbb{C}^+ \rightarrow \mathbb{C}$ is analytic and satisfies the growth condition

$$\exists M > 0: |F(s)| \leq M|s|^\mu \quad \forall s \in \mathbb{C}^+ \quad \text{and some } \mu \geq 0. \quad (3.3.4.4)$$

The starting point is the fundamental relationship (3.3.3.8) from § 3.3.3.2

$$\begin{aligned} & \text{CQ}_\tau^{\text{IE}}(F) * g|_{\mathcal{G}_\tau} = F_\tau(\partial_t)g|_{\mathcal{G}_\tau} \\ \blacktriangleright \quad & \max_{n=0, \dots, N} |F(\partial_t)g(\tau n) - (\text{CQ}_\tau^{\text{IE}}(F) * g|_{\mathcal{G}_\tau})_n| \leq \sup_{0 \leq t \leq T} |(F - F_\tau)(\partial_t)g(t)|, \end{aligned} \quad (3.3.4.5)$$

with $[0, T]$ the time interval of interest and $\tau > 0$ the timestep. Pointwise estimates for convolutions are available through

Theorem 3.1.4.12. Pointwise estimate for convolution II

Assume that

- ◆ the operator-valued function $H : \mathbb{C}^+ \rightarrow L(X, Y)$, X, Y Banach spaces, is analytic, and
- ◆ satisfies the power law growth bound

$$\exists \mu \geq 0, m \in \mathbb{N}, M > 0: \|H(s)\| \leq M \max\{1, (\text{Re } s)^{-m}\} |s|^\mu \quad \forall s \in \mathbb{C}^+, \quad (3.1.4.13)$$

- ◆ and that the causal X -valued function $g \in \mathcal{CF}(X)$ belongs to $\mathcal{C}^n(\mathbb{R}, X)$ for some $n \in \mathbb{N}$, $n > \mu + 1$, and
- ◆ that its n -th derivative $g^{(n)}$ is integrable on \mathbb{R} .

Then we can estimate

$$\|H(\partial_t)g(t)\|_Y \leq M2^\mu \frac{1 + \delta t^\delta \max\{1, t^m\}}{\pi \delta (1+t)^\delta} \int_0^t \left\| \sum_{\ell=0}^n \binom{k}{\ell} g^{(\ell)}(\tau) \right\|_X d\tau,$$

with $\delta := n - (\mu + 1)$.

and we intend to apply that theorem with $F \leftarrow F - F_\tau$.



Idea: Verify the assumption (3.1.4.13) of Thm. 3.1.4.12 for $F - F_\tau$ with

$$M \leq C\tau \quad , \quad C > 0 \text{ independent of } \tau .$$

To begin with, we use the **mean value theorem** for complex-valued functions

$$F(s) - F_\tau(s) = F(s) - F\left(\frac{1 - \exp(-s\tau)}{\tau}\right) \leq \left|s - \frac{1 - \exp(-s\tau)}{\tau}\right| \max_{z \in \Xi(s)} |F'(z)| , \quad (3.3.4.6)$$

with the line segment $\Xi(s) \subset \mathbb{C}$ connecting s and $\frac{1 - \exp(-s\tau)}{\tau}$:

$$\Xi(s) := \left\{ \zeta s + (1 - \zeta) \frac{1 - \exp(-s\tau)}{\tau}, \quad 0 \leq \zeta \leq 1 \right\} .$$

By Taylor expansion for small $|s|$ and elementary estimates for large $|s|$ one can bound the length of $\Xi(s)$ by

$$\left|s - \frac{1 - \exp(-s\tau)}{\tau}\right| \leq C \frac{1}{\tau} |\tau s|^2 = C\tau |s|^2 , \quad (3.3.4.7)$$

with some universal constant $C > 0$.

Next, we tackle $|F'(z)|$ by means of the **Cauchy differentiation formula** from Cor. 3.3.2.23,

$$F^{(\ell)}(z) = \frac{\ell!}{2\pi i} \int_{\partial B} \frac{F(w)}{(w - z)^{\ell+1}} dw \quad \forall z \in B, \quad z \in \text{disk } B \subset \mathbb{C}^+, \quad \ell \in \mathbb{N}_0 ,$$

taking into account that F is analytic in the right half-plane \mathbb{C}^+ .

$$|F'(z)| \leq \frac{1}{\pi} \left| \int_{|z-w|=\frac{1}{2}\text{Re}(z)} \frac{F(w)}{(w-z)^2} dw \right| .$$

On the circle $\{w : |z - w| = \frac{1}{2} \text{Re}(z)\}$ we have $\text{Re } w \geq \frac{1}{2} \text{Re } z$ and $|w| \leq \frac{3}{2}|z|$, which yields the estimate ($M > 0$ from Ass. 3.3.4.3)

$$|F'(z)| \leq M \left(\frac{3}{2}\right)^\mu \frac{2}{\text{Re } z} |z|^\mu \quad \forall z \in \mathbb{C}^+ . \quad (3.3.4.8)$$

Again by Taylor expansion and elementary estimates we see

$$\begin{aligned} \text{for } z \in \Xi(s): \quad \text{Re } z &\geq \min\left\{\text{Re } s, \text{Re} \frac{1 - \exp(-s\tau)}{\tau}\right\} \geq \frac{1}{2} \min\{1, \text{Re } s\} , \\ |z| &\leq \max\left\{|s|, \left|\frac{1 - \exp(-s\tau)}{\tau}\right|\right\} \leq C|s| , \end{aligned}$$

with another universal constant $C > 0$.

$$\blacktriangleright \quad |F'(z)| \leq \frac{CM}{\min\{1, \text{Re } s\}} |s|^\mu \quad \forall z \in \Xi(s), \quad s \in \mathbb{C}^+ , \quad (3.3.4.9)$$

with $C > 0$ independent of s and τ . Combine this with the estimate (3.3.4.7) for the length of the segment $\Xi(s)$:

$$|F(s) - F_\tau(s)| \leq \tau \frac{CM}{\min\{1, \text{Re } s\}} |s|^{\mu+2} \quad \forall s \in \mathbb{C}^+ . \quad (3.3.4.10)$$

Plugging this into the estimate provided by Thm. 3.1.4.12 (for $m = 1$) gives us

$$|((F - F_\tau)(\partial_t)g)(t)| \leq CM\tau \int_0^T \left| \sum_{\ell=0}^n g^{(\ell)}(\tau) \right|, \quad 0 \leq t \leq T, \quad (3.3.4.11)$$

with $n \in \mathbb{N}$, $n \geq \mu + 3$, and $C > 0$ independent of τ , but, of course, depending on $T > 0$. Finally, we invoke (3.3.4.5).

Theorem 3.3.4.12. Convergence of IE-CQ

Under Ass. 3.3.4.3 on F and assuming g to be causal and $g \in C^n(\mathbb{R})$, $n > \mu + 3$, we have

$$\max_{n=0, \dots, N} \left| F(\partial_t)g(\tau n) - (\text{CQ}_\tau^{\text{IE}}(F) * g|_{\mathcal{G}_\tau})_n \right| \leq CM\tau \int_0^T \left| \sum_{\ell=0}^n g^{(\ell)}(\tau) \right|, \quad \tau := \frac{T}{N},$$

with $C > 0$ independent of g and N .

3.4 Multistep Convolution Quadrature (MSCQ)

In § 3.3.2.1 the derivation of convolution quadrature schemes centered around the formula

$$(F(\partial_t)g)(n\tau) = (f * g)(n\tau) = \int_0^{n\tau} f(n\tau - \xi)g(\xi) d\xi \approx \frac{1}{2\pi i} \int_{\sigma+i\mathbb{R}} F(s) y_n(s) ds. \quad (3.3.2.7)$$

where $y_n(s)$ is a *numerical approximation* at $t = n\tau$ of the solution $t \mapsto y(s; t)$ of the initial-value problem

$$\dot{y}(s; t) := sy(s; t) + g(t), \quad t \in \mathbb{R}, \quad y(s; 0) = 0 \quad \text{with } s \in \mathbb{C}^+. \quad (3.3.2.3)$$

In Section 3.3 we focused on the implicit Euler single-step method (3.3.2.9) as numerical integrator for (3.3.2.3). Now we consider a whole class of timestepping schemes, which will spawn a class of CQ schemes.

3.4.1 Linear Multi-Step Numerical Integrators

§3.4.1.1 (Recalled: Single-step numerical integrators) [NumCSE Chapter 11], [NumCSE Chapter 12], [NumPDE § 10.2.2.7], and [NumPDE Chapter 7] exclusively treat single step methods for the approximate solution of initial-value problems (IVP) for an ordinary differential equation (ODE) $\dot{\mathbf{y}} = \mathbf{f}(t, \mathbf{y})$. They produce sequences (\mathbf{y}_k) of states, approximating $\mathbf{y}_k \approx \mathbf{y}(t_k)$ on temporal mesh $\{t_0 < t_1 < t_2 < \dots < t_M\}$ according to the formula

$$\mathbf{y}_{k+1} := \Psi(t_{k+1}, t_k, \mathbf{y}_k), \quad k = 0, \dots, M-1,$$

where Ψ is a suitable **discrete evolution operator**. Obviously, these methods have *no memory*; at the k -th step they forget about all previous approximations except \mathbf{y}_{k-1} , which could mean needlessly discarding information. \lrcorner

Now we consider linear multi-step methods (MSMs) for the solution of the IVP

$$\dot{\mathbf{y}} = \mathbf{f}(t, \mathbf{y}), \quad \mathbf{y}(t_0) = \mathbf{y}_0, \quad (3.4.1.2)$$

with right-hand side function $\mathbf{f} : I \times D \rightarrow \mathbb{R}^N$, $I \subset \mathbb{R}$ an interval, $D \subset \mathbb{R}^N$ the state space. These methods try to gain efficiency, by resorting to several past terms in the sequence $(\mathbf{y}_k)_{k=0}^M$ of approximate states they produce: \mathbf{y}_{k+1} is computed from $\mathbf{y}_{k-m+1}, \dots, \mathbf{y}_k$ in the case of a ***m*-step method**.

In the sequel, we confine ourselves to $t_0 = 0$ and ***equidistant temporal meshes*** $\mathcal{M}_\tau := \{t_j := j\tau\}_{j=0}^M$, with the timestep size $\tau > 0$.

EXAMPLE 3.4.1.3 (Explicit midpoint method) The solution of $t \mapsto \mathbf{y}(t)$ of (3.4.1.2) satisfies

$$\mathbf{y}((k+1)\tau) = \mathbf{y}((k-1)\tau) + \int_{t_{k-1}}^{t_{k+1}} \mathbf{f}(\xi, \mathbf{y}(\xi)) \, d\xi .$$

We can approximate the integral by the midpoint quadrature rule,

$$\mathbf{y}((k+1)\tau) \approx \mathbf{y}((k-1)\tau) + 2\tau \mathbf{f}(k\tau, \mathbf{y}(k\tau)) ,$$

which leads to the ***explicit midpoint method***, a ***2-step method***,

$$\mathbf{y}_{k+1} := \mathbf{y}_{k-1} + 2\tau \mathbf{f}(k\tau, \mathbf{y}_k) , \quad k = 1, \dots, M-1 . \tag{3.4.1.4}$$

This method is ***explicit***, because \mathbf{y}_{k+1} can be computed solely based on \mathbf{f} -evaluations. ┘

EXAMPLE 3.4.1.5 (Backward difference formulas (BDF), [DB02, Sect. 7.3.2]) We consider the IVP (3.4.1.2) with $t_0 = 0$ and equidistant temporal meshes $\mathcal{M}_\tau := \{t_j := j\tau\}_{j=0}^M$.

Given states $\mathbf{y}_j \approx \mathbf{y}(t_j)$, $j = k-m+1, \dots, k$, compute $\mathbf{y}_{k+1} := \mathbf{q}(t_{k+1})$, where $\mathbf{q} \in (\mathcal{P}_{m+1})^N$ is an uni-variate \mathbb{R}^N -valued polynomial of degree m , uniquely defined by the



$$\text{interpolation conditions: } \mathbf{q}(t_j) = \mathbf{y}_j , \quad j = k-m+1, \dots, k , \text{ and the} \tag{3.4.1.6}$$

$$\text{collocation condition: } \dot{\mathbf{q}}(t_{k+1}) = \mathbf{f}(t_{k+1}, \mathbf{y}_{k+1}) . \tag{3.4.1.7}$$

This defines an ***implicit m*-step method**. Let us extract more explicit formulas from (3.4.1.6) and (3.4.1.7). We write L_i , $i = 0, \dots, m$, for the $m+1$ Lagrange polynomials [NumCSE § 5.2.2.3] belonging to the nodes $(0, 1, 2, \dots, m)$: $L_i \in \mathcal{P}_m$, $L_i(j) = \delta_{i,j}$, $i, j \in \{0, \dots, m\}$. Then, by the cardinal basis property of the Lagrange polynomials and the interpolation conditions satisfied by q ,

$$\mathbf{q}(t) = \sum_{i=0}^m \mathbf{y}_{k-m+1+i} L_i \left(\frac{t - t_{k-m+1}}{\tau} \right) , \quad t \in \mathbb{R} .$$

This already respects (3.4.1.6). It remains to take into account (3.4.1.7) and this is done through demanding

$$\left[\dot{\mathbf{q}}(t_{k+1}) = \right] \frac{1}{\tau} \sum_{i=0}^m \mathbf{y}_{k-m+1+i} \dot{L}_i \left(\frac{t_{k+1} - t_{k-m+1}}{\tau} \right) = \mathbf{f}(t_{k+1}, \mathbf{y}_{k+1}) ,$$

which becomes

$$\sum_{i=0}^m \mathbf{y}_{k-m+1+i} \dot{L}_i(m) = \tau \mathbf{f}(t_{k+1}, \mathbf{y}_{k+1}) . \tag{3.4.1.8}$$

This is a (non-linear) equation defining \mathbf{y}_{k+1} (for sufficiently small $\tau > 0$), which accounts for the attribute “implicit” used above. The simplest BDF schemes contained in (3.4.1.8) are,

$$m = 1: \quad \mathbf{y}_{k+1} - \mathbf{y}_k = \tau \mathbf{f}(t_{k+1}, \mathbf{y}_{k+1}) , \tag{3.4.1.9}$$

$$m = 2: \quad \frac{3}{2}\mathbf{y}_{k+1} - 2\mathbf{y}_k + \frac{1}{2}\mathbf{y}_{k-1} = \tau\mathbf{f}(t_{k+1}, \mathbf{y}_{k+1}), \quad (3.4.1.10)$$

$$m = 3: \quad \frac{11}{6}\mathbf{y}_{k+1} - 3\mathbf{y}_k + \frac{3}{2}\mathbf{y}_{k-1} - \frac{1}{3}\mathbf{y}_{k-2} = \tau\mathbf{f}(t_{k+1}, \mathbf{y}_{k+1}). \quad (3.4.1.11)$$

The method (3.4.1.9) is the familiar implicit Euler 1-step method, the 2-step method (3.4.1.10) is known as **BDF-2**. .

In the above examples we can identify a general pattern.

Definition 3.4.1.12. Linear multi-step method

A general **linear m -step method**, $m \in \mathbb{N}$, for the discretization of the ODE $\dot{\mathbf{y}} = \mathbf{f}(t, \mathbf{y})$ on an equidistant temporal mesh with step size $\tau > 0$ is characterized by the recurrence relation

$$\sum_{\ell=0}^m \alpha_\ell \mathbf{y}_{j+\ell} = \tau \sum_{\ell=0}^m \beta_\ell \mathbf{f}(t_{j+\ell}, \mathbf{y}_{j+\ell}), \quad j = 0, \dots, M - m, \quad (3.4.1.13)$$

where $\alpha_\ell, \beta_\ell \in \mathbb{R}$, $\ell = 0, \dots, m$, are given coefficients satisfying $|\alpha_0| + |\beta_0| > 0$ and $\alpha_m \neq 0$.

Obviously, if $\beta_m \neq 0$, then we deal with an **implicit m -step method**. For all BDF schemes from Ex. 3.4.1.5 we have $\beta_m = 1$ and $\beta_j = 0$ for all $j \in \{0, \dots, m - 1\}$.

Remark 3.4.1.14 (Special initial steps for multi-point methods) While single-step numerical integrators can directly start from the initial state \mathbf{y}_0 , their multi-step counterparts need **special initial steps**: for an m -step method, $m > 1$, we need to determine $\mathbf{y}_1, \dots, \mathbf{y}_{m-1}$ before we can apply it directly to generate the sequence of approximate states. So Def. 3.4.1.12 does not immediately define a numerical integration scheme.

However, since we are mainly interested in causal solutions of (3.3.2.3), for $m > 1$ we can formally set $\mathbf{y}_{-m+1} = \dots = \mathbf{y}_{-1} = \mathbf{0}$ to launch the multi-step method for that special initial-value problem. Hence, we gloss over the issue of how to start multi-step integrators.

§3.4.1.15 (Characteristic polynomials) We introduce

- the sequences $(\mathbf{y}_k)_k$ (approximate states) and $(\mathbf{f}_k := \mathbf{f}(t_k, \mathbf{y}_k))_k$, and
- the **sequence left-shift operator** $S : \{\mathbb{Z} \rightarrow \mathbb{R}^N\} \rightarrow \{\mathbb{Z} \rightarrow \mathbb{R}^N\}$, $(S(\mathbf{z}_k))_\ell = \mathbf{z}_{\ell+1}$, $\ell \in \mathbb{Z}$.

Then (3.4.1.13) can be rephrased as

$$\sum_{\ell=0}^m \alpha_\ell S^\ell(\mathbf{y}_k) = \tau \sum_{\ell=0}^m \beta_\ell S^\ell(\mathbf{f}_k). \quad (3.4.1.16)$$

Another way to write is by means of the two **characteristic polynomials**

$$\rho(z) := \sum_{\ell=0}^m \alpha_\ell z^\ell, \quad \sigma(z) := \sum_{\ell=0}^m \beta_\ell z^\ell : \quad (3.4.1.17)$$

$$\blacktriangleright \quad (3.4.1.16) \Leftrightarrow \rho(S)(\mathbf{y}_k) = \sigma(S)(\mathbf{f}_k). \quad (3.4.1.18)$$

§3.4.1.19 (Consistency of multi-step methods) A difference equation of a numerical integration scheme like (3.4.1.13) is called consistent of order $p \in \mathbb{N}$ with the ODE $\dot{\mathbf{y}} = \mathbf{f}(t, \mathbf{y})$, if any solution of the ODE,

when restricted to the temporal mesh \mathcal{M}_τ , satisfies the difference equation up to a residual of size $O(\tau^{p+1})$ as $\tau \rightarrow 0$.

Hence an m -step method according to Def. 3.4.1.12 is certainly **consistent of order** $p \in \mathbb{N}$, if

$$\sum_{\ell=0}^m \alpha_\ell \mathbf{y}((j+\ell)\tau) - \tau \sum_{\ell=0}^m \beta_\ell \dot{\mathbf{y}}((j+\ell)\tau) = O(\tau^{p+1}) \quad \text{for } \tau \rightarrow 0 \tag{3.4.1.20}$$

and for *all* smooth $t \rightarrow \mathbf{y}(t)$. Here, we use the ODE to replace $\mathbf{f}((j+\ell)\tau, \mathbf{y}((j+\ell)\tau))$ with $\dot{\mathbf{y}}((j+\ell)\tau)$. By Taylor expansion we see the equivalence

$$(3.4.1.20) \Leftrightarrow \sum_{\ell=0}^m \alpha_\ell q((j+\ell)\tau) - \tau \sum_{\ell=0}^m \beta_\ell \dot{q}((j+\ell)\tau) = 0 \quad \forall q \in \mathcal{P}_{p+1}. \tag{3.4.1.21}$$

In turns, this is equivalent to

$$(3.4.1.22) \Leftrightarrow \sum_{\ell=0}^m \alpha_\ell \ell^k - \sum_{\ell=0}^m \beta_\ell k \ell^{k-1} = 0, \quad k \in \{0, \dots, p\}. \tag{3.4.1.22}$$

Hence, a multi-step method from Def. 3.4.1.12 with characteristic polynomials ρ and σ is at least *first-order consistent*, if

$$(I) \quad \sum_{\ell=0}^m \alpha_\ell = 0 \Leftrightarrow \rho(1) = 0, \quad \text{and} \tag{3.4.1.23a}$$

$$(II) \quad \sum_{\ell=0}^m \ell \alpha_\ell = \sum_{\ell=0}^m \beta_\ell \Leftrightarrow \rho'(1) := \frac{d\rho}{dz}(1) = \sigma(1). \tag{3.4.1.23b}$$

┘

Remark 3.4.1.24 (Order of consistency of some multi-step methods)

- The explicit midpoint method

$$\mathbf{y}_{k+1} := \mathbf{y}_{k-1} + 2\tau \mathbf{f}(k\tau, \mathbf{y}_k), \quad k = 1, \dots, M-1. \tag{3.4.1.4}$$

is characterized by the coefficients

$$m = 2, \quad (\alpha_0, \alpha_1, \alpha_2) = (-1, 0, 1), \quad (\beta_0, \beta_1, \beta_2) = (0, 2, 0). \tag{3.4.1.25}$$

We check (3.4.1.22) for $k = 0, 1, 2$:

$$\begin{aligned} \alpha_0 + \alpha_1 + \alpha_2 &= 0 && \checkmark, \\ 1 \cdot \alpha_1 + 2 \cdot \alpha_2 &= \beta_0 + \beta_1 + \beta_2 && \checkmark, \\ 1^2 \cdot \alpha_1 + 2^2 \cdot \alpha_2 &= 2(1 \cdot \beta_1 + 2 \cdot \beta_2) && \checkmark. \end{aligned}$$

Hence, the explicit midpoint 2-step method is of second order.

- By construction the m -step BDF multi-point methods as introduced in Ex. 3.4.1.5 are consistent of order m .

┘

Now we investigate the issue of stability of multi-step methods, when they are applied to linear ODEs. When do we have a guarantee that they produce bounded sequences of approximate states, if all IVPs for the ODE have bounded solutions? We start with a disturbing observation and continue with fundamental considerations.

EXAMPLE 3.4.1.26 (Instability of a 2-step method) The explicit midpoint method from Ex. 3.4.1.3 is a 2-step method of *order 2*.

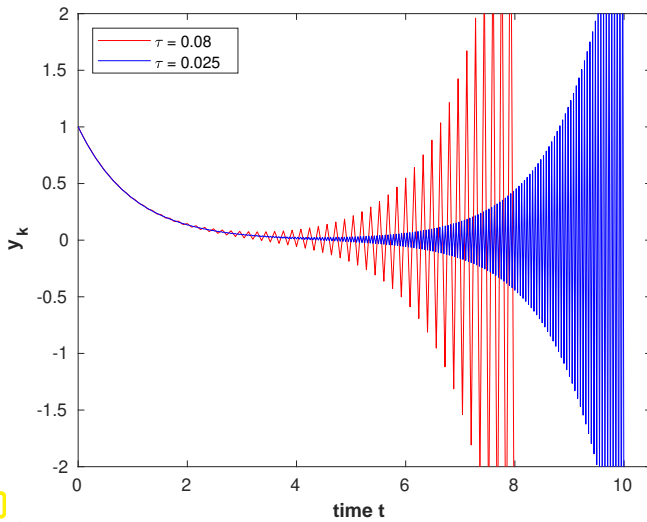


Fig. 131

◁ Explicit midpoint method applied to decay equation $\dot{y} = -y$ with implicit Euler starting step $y_0 = 1, y_1 = 1/1+\tau$, two different timestep sizes τ .

We observe exponential blow-up of the solutions!

This exponential blow-up will occur for any timestep size $\tau > 0$; the method is unconditionally unstable.

§3.4.1.27 (Stability of linear recurrence relations) A sequence $(x_k)_{k \in \mathbb{N}_0}$ of complex numbers is said to solve an $m + 1$ -term linear recurrence relation, $m \in \mathbb{N}$, if

$$\exists \gamma_\ell \in \mathbb{C}, \gamma_0, \gamma_m \neq 0: \sum_{\ell=0}^m \gamma_\ell x_{j+\ell} = 0 \quad \forall j \in \mathbb{N}_0. \tag{3.4.1.28}$$

We try $x_k = \zeta^k$ for some $\zeta \in \mathbb{C} \setminus \{0\}$. This sequence will solve the linear recurrence relation, if

$$\sum_{\ell=0}^m \gamma_\ell \zeta^\ell = 0 \iff \pi(\zeta) = 0 \quad \text{for } \pi(z) := \sum_{\ell=0}^m \gamma_\ell z^\ell \in \mathcal{P}_m. \tag{3.4.1.29}$$

The polynomial π is called the **characteristic polynomial** of the recurrence relation. From (3.4.1.29) we learn that solutions of the linear recurrence relations and roots of its characteristic polynomial are closely related. We also learn, that the location of those roots allows to predict the growth of solutions.

Theorem 3.4.1.30. Growth of solutions of linear recurrence relations, [DB02, Thm. 3.40]

Consider an $m + 1$ -term linear recurrence relation with characteristic polynomial $\pi \in \mathcal{P}_m, m \in \mathbb{N}$, whose set of complex root we denote with $\mathcal{R}, \mathcal{R} := \{\zeta \in \mathbb{C} : \pi(\zeta) = 0\}$.

- (i) If all root of π have modulus $< 1, \mathcal{R} \subset \{z \in \mathbb{C} : |z| < 1\}$, then all solutions *decay exponentially*.
- (ii) If there is a root with modulus $> 1, \exists \zeta \in \mathcal{R} : |\zeta| > 1$, then there is an *exponentially increasing solution*.

More refined results are available: If π has a double root on the unit circle,

$$\exists \zeta \in \mathbb{C} : |\zeta| = 1, \pi(\zeta) = \pi'(\zeta) = 0,$$

then the recurrence relation will have a *polynomially increasing* solution:

$$\pi'(\zeta) = 0 \iff \sum_{\ell=0}^m \gamma_\ell \ell \zeta^{\ell-1} = 0 \iff \sum_{\ell=0}^m \gamma_\ell x_{j+\ell} = 0 \quad \text{for } x_k := k\zeta^{k-1}, \quad k \in \mathbb{N}_0.$$

EXAMPLE 3.4.1.31 (Explicit midpoint method for decay equation, Ex. 3.4.1.26 cnt'd) The explicit midpoint method (timestep $\tau > 0$) applied to the decay equation $\dot{y} = -y$ gives rise to the linear 3-term recurrence relation

$$y_{k+1} = y_{k-1} - 2\tau y_k \quad \blacktriangleright \quad \pi(z) = -1 + 2\tau z + z^2.$$

The characteristic polynomial π has roots $\zeta_{\pm} = -\tau \pm \sqrt{\tau^2 + 1}$ with $|\zeta_-| > 1$, which explains the exponential blow-up observed in Ex. 3.4.1.26. ┘

§3.4.1.32 (Zero-stability of multi-step methods) What we definitely do not want is a blow-up of the solution sequence produced by a multi-step method applied to the trivial ODE $\dot{y} = 0$. The resulting linear recurrence relation is

$$\sum_{\ell=0}^m \alpha_{\ell} y_{j+\ell} = 0 \quad \text{with characteristic polynomial} \quad \pi(z) = \rho(z) = \sum_{\ell=0}^m \alpha_{\ell} z^{\ell} .$$

From § 3.4.1.27 we conclude that all solutions of this recurrence relation will be bounded, if all roots of the polynomial ρ will have modulus ≤ 1 and all roots on the unit circle are simple,

$$\zeta \in \mathbb{C}, \quad \rho(\zeta) = 0 \quad \implies \quad |\zeta| < 1 \quad \text{or} \quad \left(|\zeta| = 1 \quad \implies \quad \rho'(\zeta) \neq 0 \right) . \quad (3.4.1.33)$$

Multi-step methods satisfying this are often called **zero-stable**. ┘

Keep in mind that for the construction of CQ methods we want to apply multi-step numerical integrators to the *linear* ODE $\dot{y} = sy + g(t)$. We badly need to avoid blow-up of solutions $(y_n)_{n \in \mathbb{N}_0}$, at least in the stable case $\text{Re } s \leq 0$, cf. Rem. 3.3.2.16.

§3.4.1.34 (A-stability of multi-step methods) We adapt the key technique of linear model problem analysis from [NumCSE Section 12.1]/[NumPDE Section 7.1] to linear multistep methods. We investigate, if and when the multi-step method applied to the autonomous scalar linear ODE $\dot{y} = \lambda y, \lambda \in \mathbb{C}$, can have (exponentially) increasing solutions.

A linear m -step method according to Def. 3.4.1.12 applied to $\dot{y} = \lambda y, \lambda \in \mathbb{C}$, will give rise to the linear $m + 1$ -term recurrence relation

$$\sum_{\ell=0}^m \alpha_{\ell} y_{j+\ell} = \tau \lambda \sum_{\ell=0}^m \beta_{\ell} y_{j+\ell} \quad \Leftrightarrow \quad \sum_{\ell=0}^m (\alpha_{\ell} - z \beta_{\ell}) y_{j+\ell} = 0, \quad z := \tau \lambda \in \mathbb{C} . \quad (3.4.1.35)$$

In light of the results of § 3.4.1.27 this makes it possible to adapt the concept of region of stability [NumCSE Def. 12.1.0.51]/[NumPDE Def. 7.1.0.51] to linear multi-step methods.

Definition 3.4.1.36. Region of stability for linear multi-step method

The region of (absolute) stability of a linear m -step method defined by the two characteristic polynomials $\rho, \sigma \in \mathcal{P}_{m+1}, m \in \mathbb{N}$, cf. (3.4.1.17), is

$$\mathcal{S} := \left\{ w \in \mathbb{C} : \zeta \in \mathbb{C}, \pi_w(\zeta) = 0 \implies |\zeta| < 1 \text{ or } (|\zeta| = 1 \implies \pi'_w(\zeta) \neq 0) \right\}, \quad (3.4.1.37)$$

where $\pi_w := \rho - w \cdot \sigma \in \mathcal{P}_{m+1}, w \in \mathbb{C}$.

Another concept from single-step methods, see [NumCSE Def. 12.3.4.9]/[NumPDE Def. 7.3.4.9], can naturally be extended to multi-step methods.

Definition 3.4.1.38. A-Stability of linear multi-step methods [D802, Sect. 7.2.2]

A multi-step method is **A-stable**, if $\{z \in \mathbb{C} : \text{Re } z \leq 0\} \subset \mathcal{S}$.

Lemma 3.4.1.39. A sign condition for A-stability of multi-step methods

The polynomials $\rho, \sigma \in \mathcal{P}_{m+1}$ characterizing an *A-stable* m -step method, cf. (3.4.1.17), satisfy

$$z \in \mathbb{C}, \quad |z| > 1 \quad \implies \quad \operatorname{Re} \frac{\rho(z)}{\sigma(z)} > 0.$$

Proof. Let $w \in \mathbb{C}, \operatorname{Re} w \leq 0, \zeta \in \mathbb{C}$ satisfy

$$\pi_w(\zeta) = \rho(\zeta) - w \cdot \sigma(\zeta) = 0. \tag{3.4.1.40}$$

As the method is A-stable, $\{z \in \mathbb{C} : \operatorname{Re} z \leq 0\} \subset \mathcal{S}$, we know that $w \in \mathcal{S}$, which implies $|\zeta| \leq 1$.

By modus tollens we infer that, if (3.4.1.40) has a solution $\zeta \in \mathbb{C}$ with $|\zeta| > 1$, then $\operatorname{Re} w > 0$ must hold. In addition, $\sigma(\zeta) = 0$ can be ruled out in this case, because it would clash with the requirement (3.4.1.33) of zero-stability. As a consequence

$$0 < \operatorname{Re} w = \operatorname{Re} \left\{ \frac{\rho(\zeta)}{\sigma(\zeta)} \right\}. \tag{3.4.1.41}$$

□

Remark 3.4.1.42 (A-stability and order of linear multi-step methods) The message of Rem. 3.3.2.16 is that A-stability of the underlying numerical integration scheme is essential for the construction of viable CQ schemes. Therefore the following result is somewhat daunting.

Theorem 3.4.1.43. Second Dahlquist barrier, [DB02, Thm. 7.36]

An A-stable linear multi-step method (\rightarrow Def. 3.4.1.12) is *at best 2nd-order consistent*.

┘

EXAMPLE 3.4.1.44 (Region of stability for BDF methods) The 2-step BDF method (BDF-2) introduced in Ex. 3.4.1.5 is characterized by the polynomials

$$\rho(z) = \frac{1}{2} - 2z + \frac{3}{2}z^2, \quad \sigma(z) = z^2 \quad \blacktriangleright \quad \pi_w(z) = \frac{1}{2} - 2z + \left(\frac{3}{2} - w\right)z^2. \tag{3.4.1.45}$$

The zeros of π_w are

$$\zeta_{\pm} = \frac{1}{2\left(\frac{3}{2} - w\right)} \left(2 \pm \sqrt{4 - 2\left(\frac{3}{2} - w\right)} \right), \quad w \neq \frac{3}{2}.$$

We see that $\eta_{\pm} \rightarrow 0$ for $|w| \rightarrow \infty$, which means that the complement $\mathbb{C} \setminus \mathcal{S}$ is bounded. A closer inspection reveals that the 2-step BDF method is *A-stable*, which is still possible in light of Thm. 3.4.1.43. No higher-order BDF multi-step can be A-stable.

The pink areas in the following figures mark the stability regions of the BDF-2 and BDF-3 multi-step methods¹. BDF-3 barely misses A-stability.

¹Fig. 132, Fig. 133 by Jitse Niesen, CC0, [Wikipedia](#)

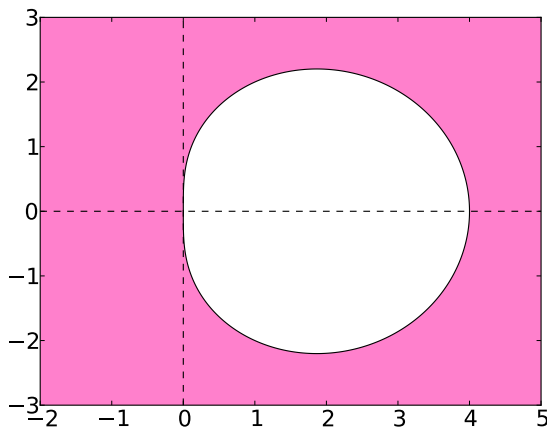


Fig. 132

BDF-2

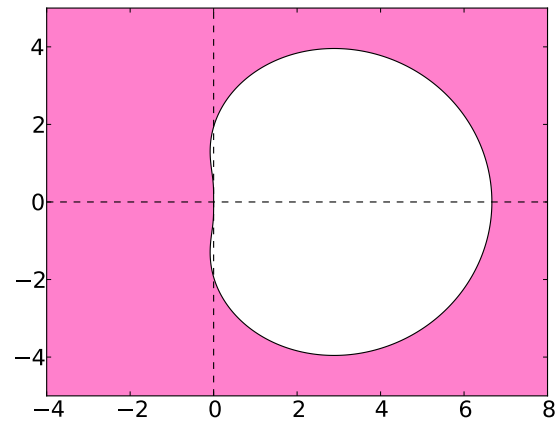


Fig. 133

BDF-3

§3.4.1.46 (Convergence of linear multi-step methods [DB02, Sect. 7.1.3]) On finite time intervals order- p single-step numerical integrators for initial-value problems for ODEs generate sequences of approximate states that *converge algebraically* to the exact solution (sampled on the temporal grid) in maximum norm, see [NumCSE Section 11.3.2] or [NumPDE Section 6.3.2].

Mere order- p consistency is not enough for linear multistep method. In addition zero-stability is required.

Theorem 3.4.1.47. A stable and consistent linear MSM is convergent

Let $\mathbf{y} \in C^{p+1}([0, T], \mathbb{R}^N)$ be the solution of the IVP $\dot{\mathbf{y}} = \mathbf{f}(t, \mathbf{y}), \mathbf{y}(0) = \mathbf{y}_0$. Then for a *zero-stable* linear m -step method with *consistency order* p there exist constants $C > 0, \epsilon_* > 0, \tau_* > 0$, such that for uniform timestep size $\tau := \frac{T}{M} < \tau_*$ and starting error

$$\epsilon_0 := \max\{\|\mathbf{y}_\ell - \mathbf{y}(\tau\ell)\| : \ell \in \{0, \dots, m-1\}\} \leq \epsilon_*$$

the method generates a sequence $\mathbf{y}_\ell, \ell = 0, \dots, M$, of approximate states, which satisfies

$$\|\mathbf{y}_\ell - \mathbf{y}(\tau\ell)\| \leq C(\epsilon_0 + \tau^p) \quad \forall \ell = 0, \dots, M.$$

For a Runge-Kutta single-step method the number of stages limits the order of consistency: The maximal order of an s -stage implicit RK-SSM is $2s$. A similar result also applies to linear multi-step methods.

Theorem 3.4.1.48. First Dahlquist barrier [DB02, Thm. 7.16]

The order p of consistency of a *zero-stable* linear m -step method is subject to the following restrictions:

- (i) $p \leq m + 2$ for even m ,
- (ii) $p \leq m + 1$ for odd m ,
- (iii) $p \leq q$, if the method is explicit ($\beta_m = 0$).

3.4.2 Multi-Step Convolution Quadrature: Weights

Let the transfer function $F : \mathbb{C}^+ \rightarrow \mathbb{C}$ be the Laplace transform (\rightarrow Def. 3.1.3.6) of a causal function/distribution f and write $g : \mathbb{R} \rightarrow \mathbb{C}$ for a continuous causal function. From § 3.3.2.1 we recall the formula

$$F(\partial_t)g(t) = (f * g)(t) = \int_0^t f(t - \xi)g(\xi) d\xi = \frac{1}{2\pi i} \int_{\sigma + i\mathbb{R}} F(s) y(s; t) ds, \quad \begin{matrix} t \geq 0, \\ \sigma > 0, \end{matrix} \quad (3.3.2.5)$$

where $t \mapsto y(s; t)$, $s \in \mathbb{C}^+$, is the unique solution of the scalar linear initial-value problem

$$\dot{y}(s; t) := \frac{\partial y}{\partial t}(s; t) = sy(s; t) + g(t), \quad t \geq 0, \quad y(s; t) = 0 \quad \forall t \leq 0. \quad (3.3.2.3)$$

§3.4.2.1 (Multi-step numerical integration of (3.3.2.3)) Multi-step convolution quadrature with step size $\tau > 0$ boils down to the approximation

$$(F(\partial_t)g)(\tau n) \approx \frac{1}{2\pi i} \int_{\sigma + i\mathbb{R}} F(s) y_n(s) ds, \quad n \in \mathbb{N}_0, \quad (3.4.2.2)$$

with $(y_n(s))_{n \in \mathbb{N}_0}$ the (causal) sequence of approximations $y_n(s) \approx y(s; \tau n)$ generated by some multi-step method applied to (3.3.2.3) with uniform step size $\tau > 0$ and zero starting states. Concretely, using a linear m -step method according to

Definition 3.4.1.12. Linear multi-step method

A general **linear m -step method** for the discretization of the ODE $\dot{\mathbf{y}} = \mathbf{f}(t, \mathbf{y})$ on an equidistant temporal mesh with step size $\tau > 0$ is given by the recurrence relation

$$\sum_{\ell=0}^m \alpha_\ell \mathbf{y}_{j+\ell} = \tau \sum_{\ell=0}^m \beta_\ell \mathbf{f}(t_{j+\ell}, \mathbf{y}_{j+\ell}), \quad j = 0, \dots, M - m, \quad (3.4.1.13)$$

where $\alpha_\ell, \beta_\ell \in \mathbb{R}$, $\ell = 0, \dots, m$, are given coefficients satisfying $|\alpha_0| + |\beta_0| > 0$ and $\alpha_m \neq 0$.

with $m \in \mathbb{N}$, the sequence $(y_n(s))$ is defined by the recurrence relation

$$\sum_{\ell=0}^m \alpha_\ell y_{n+\ell}(s) = \tau s \sum_{\ell=0}^m \beta_\ell y_{n+\ell}(s) + \tau \sum_{\ell=0}^m \beta_\ell g_{n+\ell}, \quad n \in \mathbb{N}_0, \quad (3.4.2.3a)$$

$$y_j(s) = 0 \quad \forall j \in \mathbb{Z}, j \leq 0. \quad (3.4.2.3b)$$

Here we wrote $g_j := g(\tau j)$, $j \in \mathbb{N}_0$, and we extended (y_n) by zero to obtain a causal sequence. This will define a valid timestepping scheme only if $\alpha_m - \tau s \beta_m \neq 0$, which we assume in the sequel.

Assumption 3.4.2.4. Properties of linear multi-step method underlying CQ

The linear multi-step method defining (y_n) by means of (3.4.2.3) and $\sigma > 0$ are expected to satisfy

$$\beta_m \alpha_m > 0 \quad (\implies \text{implicit MSM}) \quad \text{and} \quad \alpha_m - \tau s \beta_m \neq 0 \quad \forall s \in \mathbb{C}, \operatorname{Re} s = \sigma. \quad (3.4.2.5)$$

┘

§3.4.2.6 (Multi-step recurrence in z-domain) Straightforward computations show that under the z-transform as defined in (3.1.4.17) shift operators in sequence space become multiplication with powers of z in the z-domain. For a bounded causal sequence (x_j)

$$\mathcal{Z}(S((x_j)))(z) = \sum_{\ell=0}^{\infty} (S(x_j))_\ell z^\ell = \sum_{\ell=0}^{\infty} x_{\ell+1} z^\ell = \frac{1}{z} (\mathcal{Z}((x_j))(z) - x_0), \quad z \in \mathbb{C}, |z| < 1. \quad (3.4.2.7)$$

We can exploit this, because both sides of (3.4.2.3a) act on causal sequences through a linear combination of powers of shift operators.



Idea: Apply the **z-transform** \mathcal{Z} from (3.1.4.17) to both sides of (3.4.2.3a)

We will not use (3.4.2.7), but, for the sake of clarity, we directly work with the definition (3.1.4.17) and manipulate power series. We start with a change of index $n \rightarrow n - m$ in (3.4.2.3a):

$$\sum_{\ell=0}^m \alpha_{\ell} y_{n-m+\ell}(s) = \tau s \sum_{\ell=0}^m \beta_{\ell} y_{n-m+\ell}(s) + \tau \sum_{\ell=0}^m \beta_{\ell} g_{n-m+\ell}, \quad n \in \mathbb{N}_0. \quad (3.4.2.8)$$

Next, we multiply both sides of (3.4.2.8) with z^n and sum over n , assuming that the series converge for $|z|$ sufficiently small:

$$\sum_{n=0}^{\infty} \left(\sum_{\ell=0}^m \alpha_{\ell} y_{n-m+\ell}(s) \right) \cdot z^n = \sum_{n=0}^{\infty} \left(\tau s \sum_{\ell=0}^m \beta_{\ell} y_{n-m+\ell}(s) \right) \cdot z^n + \sum_{n=0}^{\infty} \left(\tau \sum_{\ell=0}^m \beta_{\ell} g_{n-m+\ell} \right) \cdot z^n, \quad z \in \mathbb{C}.$$

What we are aiming for is to obtain an equation featuring the z-transforms of the involved causal sequences,

$$Y(s; z) := \mathcal{Z}((y_n(s))_n) = \sum_{n=-\infty}^{\infty} y_n(s) z^n, \quad G(z) := \mathcal{Z}((g_j)_j) = \sum_{j=-\infty}^{\infty} g_j z^j, \quad z \in \mathbb{C}, |z| < 1. \quad (3.4.2.9)$$

To that end we first distribute powers of z :

$$\begin{aligned} \sum_{n=0}^{\infty} \sum_{\ell=0}^m \alpha_{\ell} y_{n-m+\ell}(s) z^{n-m+\ell} \cdot z^{m-\ell} &= \tau s \sum_{n=0}^{\infty} \sum_{\ell=0}^m \beta_{\ell} y_{n-m+\ell}(s) z^{n-m+\ell} \cdot z^{m-\ell} + \\ &\tau \sum_{n=0}^{\infty} \sum_{\ell=0}^m \beta_{\ell} g_{n-m+\ell} z^{n-m+\ell} \cdot z^{m-\ell}. \end{aligned} \quad (3.4.2.10)$$

We change the order of summation and then extract n -independent factors from the inner sums:

$$\begin{aligned} \sum_{\ell=0}^m \alpha_{\ell} z^{m-\ell} \sum_{n=0}^{\infty} y_{n-m+\ell}(s) z^{n-m+\ell} &= \\ \tau s \sum_{\ell=0}^m \beta_{\ell} z^{m-\ell} \sum_{n=0}^{\infty} y_{n-m+\ell}(s) z^{n-m+\ell} &+ \tau \sum_{\ell=0}^m \beta_{\ell} z^{m-\ell} \sum_{n=0}^{\infty} g_{n-m+\ell} z^{n-m+\ell}, \end{aligned} \quad (3.4.2.11)$$

which means, since (y_n) and (g_n) are causal sequences,

$$\begin{aligned} \sum_{\ell=0}^m \alpha_{\ell} z^{m-\ell} Y(s; z) &= \tau s \sum_{\ell=0}^m \beta_{\ell} z^{m-\ell} Y(s; z) + \tau \sum_{\ell=0}^m \beta_{\ell} z^{m-\ell} G(z), \quad z \in \mathbb{C}, |z| \ll 1 \quad (3.4.2.12) \\ \iff z^m \rho(z^{-1}) Y(s; z) &= \tau s z^m \sigma(z^{-1}) Y(s; z) + \tau z^m \sigma(z^{-1}) G(z), \quad z \in \mathbb{C}, |z| \ll 1. \end{aligned}$$

Now we can solve for $Y(s; z)$:

$$Y(s; z) = \left(\frac{1}{\tau^{-1} \delta(z) - s} \right) G(z), \quad \delta(z) := \frac{z^m \rho(z^{-1})}{z^m \sigma(z^{-1})} = \frac{\sum_{\ell=0}^m \alpha_{\ell} z^{m-\ell}}{\sum_{\ell=0}^m \beta_{\ell} z^{m-\ell}}. \quad (3.4.2.13)$$

Summing up, with (3.4.2.13) we have found an expression for the solution of (3.4.2.3) in the “z-domain”. ┘

§3.4.2.14 (Properties of δ)

- ◆ $z \mapsto \delta(z)$ as defined in (3.4.2.13) is a *rational function* of degree (m, m) .
- ◆ By virtue of Ass. 3.4.2.4 $z \mapsto \delta(z)$ is analytic in a neighborhood of $z = 0$ and

$$\delta(0) = \alpha_m / \beta_m > 0. \tag{3.4.2.15}$$

- ◆ If the multi-step method is *A-stable* (\rightarrow Def. 3.4.1.38), then Lemma 3.4.1.39 implies

$$|z| < 1 \implies \operatorname{Re} \delta(z) > 0, \tag{3.4.2.16}$$

because $\delta(z) = \rho(z^{-1}) / \sigma(z^{-1})$.

┘

§3.4.2.17 (Formula for multi-step CQ weights) As explained in Section 3.3.1, for given transfer function $F; \mathbb{C}^+ \rightarrow \mathbb{C}$ and timestep $\tau > 0$ we want to find a causal sequence $\text{CQ}_\tau(F) : \mathbb{N}_0 \rightarrow \mathbb{C}$ such that

$$F(\partial_t)g|_{\mathcal{G}_\tau} \approx \text{CQ}_\tau(F) * g|_{\mathcal{G}_\tau} = \left(\frac{1}{2\pi i} \int_{\sigma+i\mathbb{R}} F(s) y_n(s) ds \right)_{n \in \mathbb{N}_0}, \tag{3.4.2.18}$$

with $(y_n(s))$ the sequence generated by the linear recurrence relation (3.4.2.3). The formula (3.4.2.13) implicitly defines that sequence $(y_n(s))$ through its z-transform $z \mapsto Y(s; z)$.



Idea: Apply the z-transform on both sides of (3.4.2.18).

We abbreviate $Q(z) := (\mathcal{Z}(\text{CQ}_\tau(F)))(z)$, $z \in \mathbb{C}$, $|z|$ small, and appeal to the convolution theorem for the z-transform:

Theorem 3.1.4.18. z-Transform and discrete convolution

If (g_ℓ) and (f_ℓ) are causal summable sequences, then

$$\mathcal{Z}((f_\ell) * (g_\ell))(z) = \mathcal{Z}((f_\ell))(z) \cdot \mathcal{Z}((g_\ell))(z), \quad \forall z \in \{z \in \mathbb{C} : |z| < 1\}. \tag{3.1.4.19}$$

Thus, applying the z-transform to both sides of (3.4.2.18), we arrive at

$$\begin{aligned} Q(z)G(z) &= \sum_{n=0}^{\infty} \left(\frac{1}{2\pi i} \int_{\sigma+i\mathbb{R}} F(s) y_n(s) ds \right) \cdot z^n \\ &= \frac{1}{2\pi i} \int_{\sigma+i\mathbb{R}} F(s) \sum_{n=0}^{\infty} y_n(s) \cdot z^n ds = \frac{1}{2\pi i} \int_{\sigma+i\mathbb{R}} F(s) Y(s; z) ds. \end{aligned} \tag{3.4.2.19}$$

We can immediately plug in the formula (3.4.2.13) for $Y(s; z)$ and get for all $z \in \mathbb{C}$ sufficiently close to zero

$$Q(z)G(z) = \frac{1}{2\pi i} \int_{\sigma+i\mathbb{R}} F(s) \left(\frac{1}{\tau^{-1}\delta(z) - s} \right) ds \cdot G(z), \tag{3.4.2.20}$$

which means

$$Q(z) = \sum_{n=0}^{\infty} (CQ_{\tau}(F))_n z^n = \frac{1}{2\pi i} \int_{\sigma+i\mathbb{R}} F(s) \left(\frac{1}{\tau^{-1}\delta(z) - s} \right) ds. \tag{3.4.2.21}$$

From this point the transfer function is supposed to satisfy Ass. 3.3.1.1.

Assumption 3.3.1.1. Properties of transfer function

$F : \mathbb{C}^+ \rightarrow \mathbb{C}$ is analytic on the right half plane \mathbb{C}^+ and satisfies the **decay condition**

$$\exists M > 0: |F(s)| \leq M|s|^{\mu} \quad \forall s \in \mathbb{C}^+ \quad \text{and some } \mu < -1. \tag{3.3.1.2}$$

This makes it possible to pursue the same considerations as in § 3.3.2.21. To begin with, Ass. 3.4.2.4 ensures that

$$\operatorname{Re} \frac{\delta(z)}{\tau} > \sigma \quad \text{for } |z| < 1 \quad \text{and } \sigma \text{ sufficiently small.} \tag{3.4.2.23}$$

Then note that the integrand has a singularity in $s = \tau^{-1}\delta(z)$ and is holomorphic in $\mathbb{C}^+ \setminus \{\tau^{-1}\delta(z)\}$, This makes it possible to “deform the contour of integration”:

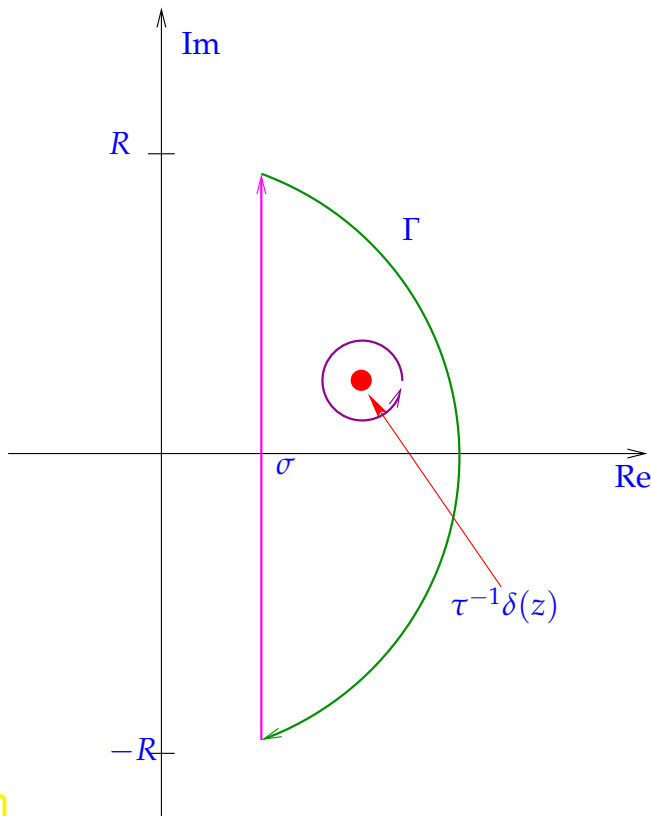


Fig. 134

Since the integrand

$$s \mapsto F(s) \left(\frac{1}{\tau^{-1}\delta(z) - s} \right)$$

is analytic in $\mathbb{C}^+ \setminus \{\tau^{-1}\delta(z)\}$, by the **Cauchy integral theorem** Thm. 3.1.3.16 its path integral over the contour

$$\begin{aligned} \Gamma &:= \Gamma_{\sigma} \cup \Gamma_R \cup \Gamma_r, \\ \Gamma_{\sigma} &:= \sigma + i[-R, R], \\ \Gamma_R &:= \{s : |s| = R, \operatorname{Re} z \geq \sigma\}, \\ \Gamma_r &:= \{s : |s - \tau^{-1}\delta(z)| = r\}, \end{aligned}$$

with $r, R > 0$, $R > \tau^{-1}\delta(z) + r$ and suitable orientations of the pieces, vanishes. The decay properties of F from Ass. 3.3.1.1 imply

$$\int_{\Gamma_R} F(s) \left(\frac{1}{\tau^{-1}\delta(z) - s} \right) ds \rightarrow 0 \quad \text{for } R \rightarrow \infty.$$

Hence, instead of integrating along a line parallel to the imaginary axis we can evaluate a path integral over a small circle centered at $z_0 = \tau^{-1}\delta(z)$:

$$\frac{1}{2\pi i} \int_{\sigma+i\mathbb{R}} F(s) \left(\frac{1}{\tau^{-1}\delta(z) - s} \right) ds = -\frac{1}{2\pi i} \int_{|s-\delta(z)/\tau|} \frac{F(s)}{\tau^{-1}\delta(z) - s} ds = F\left(\frac{\delta(z)}{\tau}\right), \tag{3.4.2.24}$$

where we applied the Cauchy integral formula to the holomorphic function $F : \mathbb{C}^+ \rightarrow \mathbb{C}$:

Theorem 3.3.2.22. Cauchy integral formula

If $g : D \subset \mathbb{C} \rightarrow \mathbb{C}$ is *analytic* in D , $c \in D$, and $B := \{s : |s - z| \leq r\} \subset D$ for some $r > 0$, then

$$g(z) = \frac{1}{2\pi i} \int_{\partial B} \frac{g(s)}{s - z} ds \quad \forall z \in B,$$

where the integral is a complex contour integral and the circle ∂B is oriented counterclockwise.

Obviously, combining (3.4.2.21) and (3.4.2.24) yields

$$Q(z) = \sum_{n=0}^{\infty} (CQ_{\tau}(F))_n z^n = F\left(\frac{\delta(z)}{\tau}\right), \quad |z| < 1.$$

We have arrived at a generalization of Lemma 3.3.2.24:

Lemma 3.4.2.25. Convolution quadrature weights for multi-step CQ

We consider CQ with uniform timestep $\tau > 0$ based on a linear multi-step method according to Def. 3.4.1.12, whose coefficients satisfy Ass. 3.4.2.4. If $F : \mathbb{C}^+ \rightarrow \mathbb{C}$ is analytic and complies with Ass. 3.3.1.1, then

$$z \mapsto F\left(\frac{\delta(z)}{\tau}\right), \quad \delta(z) := \frac{z^m \rho(z^{-1})}{z^m \sigma(z^{-1})} = \frac{\sum_{\ell=0}^m \alpha_{\ell} z^{m-\ell}}{\sum_{\ell=0}^m \beta_{\ell} z^{m-\ell}}, \quad z \in \mathbb{C}, |z| \ll 1, \quad (3.4.2.26)$$

is a *generating function* for the convolution quadrature weights from Def. 3.3.2.14, that is,

$$F\left(\frac{\delta(z)}{\tau}\right) = \sum_{\ell=0}^{\infty} (CQ_{\tau}(F))_{\ell} z^{\ell} \quad \text{for } |z| \text{ sufficiently small.} \quad (3.4.2.27)$$

Note that for an *A-stable* (\rightarrow Def. 3.4.1.38) multi-step method (3.4.2.16) guarantees that $\delta(z)/\tau \in \mathbb{C}^+$, so that the argument of F will always lie in its domain of definition. This is another reason, why CQ schemes are mostly based on A-stable timestepping methods. ┘

EXAMPLE 3.4.2.28 (δ for simple BDF multi-step schemes)

- The implicit Euler method, the first-order BDF scheme, fits Def. 3.4.1.12 with $m = 1$ and

$$\begin{aligned} & \alpha_0 = -1, \quad \alpha_1 = 1, \quad \beta_1 = 1, \quad \beta_0 = 0, \\ \blacktriangleright \quad & \delta(z) = 1 - z, \end{aligned}$$

which shows that Lemma 3.3.2.24 is just a special case of Lemma 3.4.2.25.

- For the 2-step BDF method of order 2 from (3.4.1.10) we have

$$\begin{aligned} & (\alpha_0, \alpha_1, \alpha_2) = \left(\frac{1}{2}, -2, \frac{3}{2}\right), \quad (\beta_0, \beta_1, \beta_2) = (0, 0, 1), \\ \blacktriangleright \quad & \delta(z) = \frac{1}{2}z^2 - 2z + \frac{3}{2}. \end{aligned}$$

- The general formular for the m -step BDF schemes introduced in Ex. 3.4.1.5 is

$$\delta(z) = \sum_{j=1}^m \frac{1}{j} (1 - z)^j, \quad z \in \mathbb{C} \implies \delta \in \mathcal{P}_{m+1}. \quad (3.4.2.29)$$

┘

§3.4.2.30 (Continuous-in-time multi-step convolution quadrature) In § 3.3.3.2 we saw that the CQ scheme based on implicit Euler timestepping (with uniform timestep $\tau > 0$) can be viewed as approximating the operator $F(\partial_t)$ by $F_\tau(\partial_t)$ where $F \approx F_\tau$, see Section 3.3.4. This was succinctly expressed by

$$\text{CQ}_\tau^{\text{IE}}(F) * g|_{\mathcal{G}_\tau} = F_\tau(\partial_t)g|_{\mathcal{G}_\tau}. \tag{3.3.3.8}$$

Hardly surprising, this can be generalized to multi-step CQ schemes. To simplify notation we write $\text{CQ}_\tau(F) = (w_\ell^{F,\tau})_{\ell \in \mathbb{N}_0}$ for the sequence of CQ weights. defined by (3.3.2.25):

$$F\left(\frac{\delta(z)}{\tau}\right) = \sum_{\ell=0}^{\infty} w_\ell^{F,\tau} z^\ell \quad \text{for } |z| \text{ sufficiently small.} \tag{3.4.2.31}$$

Formally, for “continuous time argument” $t \in \mathbb{R}$ we can rewrite convolution quadrature by means of convolution with shifted δ -distributions:

$$(F_\tau(\partial_t)g)(t) = \sum_{\ell=0}^{\infty} w_\ell^{F,\tau} g(t - \ell\tau) = \left(\sum_{\ell=0}^{\infty} w_\ell^{F,\tau} (\delta_{\ell\tau} * g) \right)(t). \tag{3.4.2.32}$$

With the Laplace transform of a shifted δ -distribution

$$\mathcal{L}\{t \mapsto \delta(t - \tau)\}(s) = \int_{\mathbb{R}} \delta(t - \tau) \exp(-st) dt = \exp(-s\tau), \tag{3.3.3.5}$$

Laplace-transforming both sides of (3.4.2.32) we get with $G(s) := \mathcal{L}g(s)$

$$F_\tau(s)G(s) = \mathcal{L}(F_\tau(\partial_t)g) = \left(\sum_{\ell=0}^{\infty} w_\ell^{F,\tau} e^{-s\ell\tau} \right) \cdot G(s). \tag{3.4.2.33}$$

Taking into account (3.4.2.31) we end up with

$$F_\tau(s) = F\left(\frac{\delta(e^{-s\tau})}{\tau}\right), \quad s \in \mathbb{C}^+. \tag{3.4.2.34}$$

┘

Remark 3.4.2.35 (Approximations underlying (3.4.2.34)) The function $y(t) = e^{-st}$ solves the ODE $\dot{y} = -sy$, $s \in \mathbb{C}^+$ fixed. Consider a m -step method according to Def. 3.4.1.12 of order p . In § 3.4.1.19 we argued that it must satisfy

$$\begin{aligned} \sum_{\ell=0}^m \alpha_\ell e^{-s(j+\ell)\tau} - \tau s \sum_{\ell=0}^m \beta_\ell e^{-s(j+\ell)\tau} &= O(\tau^{p+1}) \quad \text{for } \tau \rightarrow 0 \\ &\Updownarrow [z := s\tau] \\ \sum_{\ell=0}^m \alpha_\ell e^{-z\ell} + z \sum_{\ell=0}^m \beta_\ell e^{-z\ell} &= O(z^{p+1}) \quad \text{for } z \rightarrow 0 \\ &\Updownarrow \\ \sum_{\ell=0}^m \alpha_\ell e^{z(m-\ell)} + z \sum_{\ell=0}^m \beta_\ell e^{z(m-\ell)} &= O(z^{p+1}) \quad \text{for } z \rightarrow 0. \end{aligned}$$

Since $\beta_m \neq 0$ we can divide by $\sum_{\ell=0}^m \beta_\ell e^{z(m-\ell)}$ for sufficiently small z and get

$$\delta(e^z) + z = O(z^{p+1}) \quad \text{for } z \rightarrow 0. \tag{3.4.2.36}$$

In words, $w \rightarrow \delta(w)$ is a rational approximation of $w \mapsto -\log w$ in a neighborhood of 1. Combining (3.4.2.36) with (3.4.2.34) we conclude

$$\frac{\delta(e^{-s\tau})}{\tau} \rightarrow s \quad \text{for } \tau \rightarrow 0 \implies F_\tau(s) \rightarrow F(s) \quad \text{for } \tau \rightarrow 0. \tag{3.4.2.37}$$

┘

Remark 3.4.2.38 (Convergence of multi-step CQ) If a multi-step convolution quadrature scheme is based on an A-stable multi-step method of order $p \in \mathbb{N}$, then, under several technical assumptions on F and for sufficiently smooth causal $g : \mathbb{R} \rightarrow \mathbb{C}$, we can expect [BS21, Sect. 2.5]

$$\max_{n \in \{0, \dots, T/\tau\}} |(((F - F_\tau)(\partial_t))(g))(\tau n)| = O(\tau^p) \quad \text{for } \tau \rightarrow 0. \tag{3.4.2.39}$$

┘

EXPERIMENT 3.4.2.40 (Convergence of CQ based on BDF-2) We revisit the setting of Exp. 3.3.4.2 with $F(s) = s^{-1/2}$ (Abel integral operator), choose $g(t) = t$ and $g(t) = t^2$ for $t > 0$ and monitor² the error (3.4.2.39) for $T = 1$.

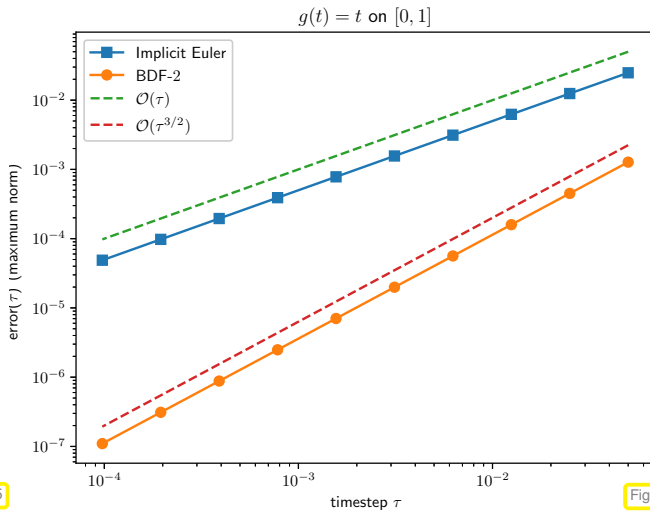
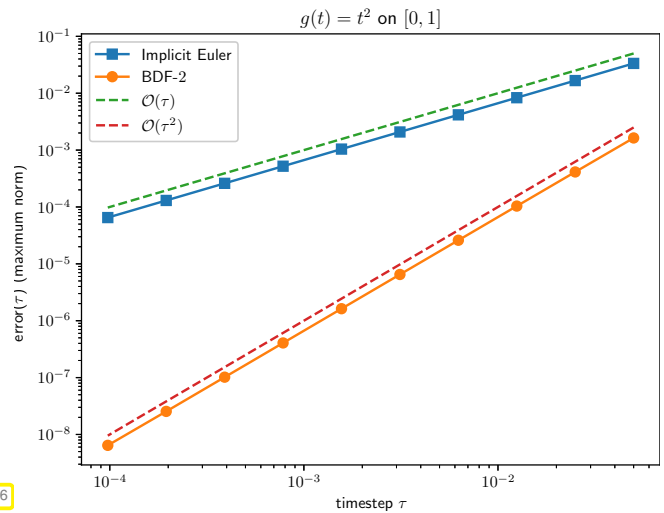


Fig. 135

Fig. 136

$$g(t) = t$$



$$g(t) = t^2$$

We clearly observe empirical algebraic convergence of the error (norm) in τ , because the error point a neatly located on straight lines in a doubly logarithmic plot.

The function $g(t) = t^2$ enjoys global C^1 -smoothness, no kink at $t = 0$, whereas $g(t) = t$ has a discontinuous derivative. So in the latter case the optimal rate 2 of asymptotic algebraic convergence for $\tau \rightarrow 0$ of BDF-2 CQ cannot be realized.

┘

3.4.3 Multi-Step Convolution Quadrature: Algorithms

We consider a CQ scheme founded on an A-stable linear multistep method (\rightarrow Def. 3.4.1.12) as introduced in Section 3.4.2. The transfer function F is to comply with Ass. 3.3.4.3. Moreover, we have to deal with the following constraint.

²Python codes by L. Banjai, <https://github.com/lehelb/TDBIE-CQ-book>

Constraint in transfer function F

The transfer function $F : \mathbb{C}^+ \rightarrow \mathbb{C}$ is given only in **procedural form**, that is, as
`complex<double> F (complex<double> z) ; ;`
 only point evaluations are possible, cf. [NumCSE Rem. 5.1.0.9].

Again, we abbreviate the CQ weights as $w_\ell^{F,\tau} := (\text{CQ}_\tau(F))_\ell$ and assume that we are given a causal function $g : \mathbb{R} \rightarrow \mathbb{C}$, also in procedural form.

Task: Our goal is to approximately (*) compute the first $M \in \mathbb{N}$ terms of the CQ-discretized convolution $f * g$, namely

$$y_n \approx \sum_{\ell=0}^n w_{n-\ell}^{F,\tau} g_\ell, \quad n \in \{0, \dots, M\}, \quad g_\ell := g(\tau\ell), \quad \tau > 0. \tag{3.4.3.2}$$

(*): approximation is inevitable, owing to the above constraint on F !

Often one wants to evaluate a convolution up to some final time $T > 0$. In this case, choose $M \approx T/\tau$.

§3.4.3.3 (CQ weights through complex path integrals) The formula

$$F\left(\frac{\delta(z)}{\tau}\right) = \sum_{\ell=0}^{\infty} w_\ell^{F,\tau} z^\ell \quad \text{for } |z| < 1, \tag{3.4.2.27}$$

from Lemma 3.4.2.25 shows that the CQ weights are the coefficients of a Taylor series expansion of function holomorphic in a neighborhood of $z = 0$. As such they can be computed from derivatives in $z = 0$, which, in turns, can be expressed via

Corollary 3.3.2.23. Cauchy differentiation formula

If $g : D \subset \mathbb{C} \rightarrow \mathbb{C}$ is analytic in D , $z \in D$, and $B := \{s \in \mathbb{C} : |s - z| \leq r\} \subset D$ for some $r > 0$, then the ℓ -th derivative of g can be computed as the contour integral

$$g^{(\ell)}(z) = \frac{\ell!}{2\pi i} \int_{\partial B} \frac{g(s)}{(s - z)^{\ell+1}} ds \quad \forall z \in B, \quad \ell \in \mathbb{N}_0.$$

$$\blacktriangleright w_\ell^{F,\tau} = \frac{1}{\ell!} \frac{d^\ell}{dz^\ell} \left\{ z \mapsto F\left(\frac{\delta(z)}{\tau}\right) \right\}_{z=0} = \frac{1}{2\pi i} \int_{|z|=r} \frac{1}{z^{\ell+1}} F\left(\frac{\delta(z)}{\tau}\right) dz, \quad \ell \in \mathbb{Z}, \tag{3.4.3.4}$$

for *any* $r \in]0, 1[$. Only in rare cases the weights can be computed explicitly, refer to Ex. 3.3.2.27. Usually we will have to rely on numerical approximations, which start from the parameterization $\varphi \in [0, 1] \mapsto r \exp(2\pi i \varphi)$ of the circular path of integration in (3.4.3.4):

$$\begin{aligned} w_\ell^{F,\tau} &= \frac{1}{2\pi i} \int_0^1 \frac{1}{(re^{2\pi i \varphi})^{\ell+1}} F\left(\frac{\delta(re^{2\pi i \varphi})}{\tau}\right) 2\pi i r e^{2\pi i \varphi} d\varphi \\ &= r^{-\ell} \int_0^1 e^{-2\pi i \ell \varphi} F\left(\frac{\delta(re^{2\pi i \varphi})}{\tau}\right) d\varphi, \quad \ell \in \mathbb{Z}. \end{aligned} \tag{3.4.3.5}$$

Note that the integrand is *1-periodic* and *analytic* in a strip along the real axis. We also point out that for $\ell < 0$ the integrand in (3.4.3.4) is holomorphic on $z \in \mathbb{C} : |z| < 1$. Then, by the Cauchy integral formula of Thm. 3.3.2.22 we infer $w_\ell^{F,\tau} = 0$ for $\ell < 0$: (3.4.3.4) defines a causal sequence. \lrcorner

EXPERIMENT 3.4.3.6 (The “magic” of the equidistant composite trapezoidal quadrature rule, [NumCSE Exp. 7.5.0.16]) For the approximation of an integral one may use the $N + 1$ -point, $N \geq 1$, equidistant composite trapezoidal rule, see [NumCSE Eq. (7.5.0.4)], [NumCSE Code 7.5.0.6]

$$\int_a^b f(t) dt \approx T_N(f) := h \left(\frac{1}{2}f(a) + \sum_{k=1}^{N-1} f(kh) + \frac{1}{2}f(b) \right), \quad h := \frac{b-a}{N}. \tag{3.4.3.7}$$

Its order [NumCSE Def. 7.4.1.1] is merely 2 [NumCSE Ex. 7.4.1.10] and for smooth integrands f one would expect an asymptotic behavior of the quadrature error like $O(N^{-2})$ for $N \rightarrow \infty$.

We apply this quadrature rule to the *1-periodic* smooth (*analytic*) integrand

$$f(t) = \frac{1}{\sqrt{1 - a \sin(2\pi t - 1)}}, \quad 0 < a < 1,$$

on different intervals. As “exact value of integral” we use T_{500} in the computation of the quadrature errors.

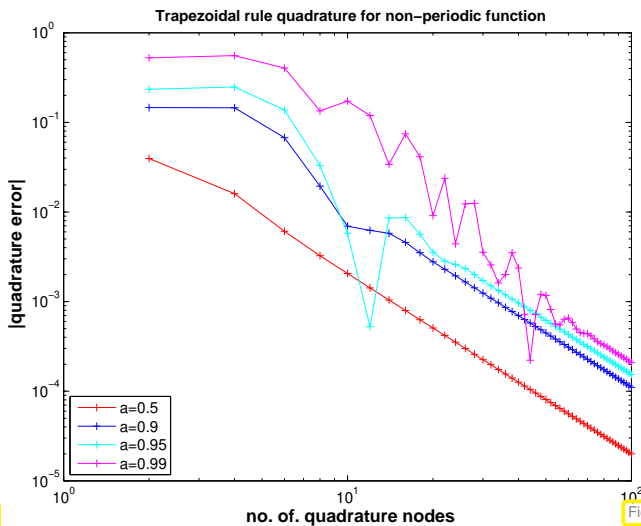


Fig. 137

quadrature error for $T_n(f)$ on $[0, \frac{1}{2}]$

Merely **algebraic convergence**

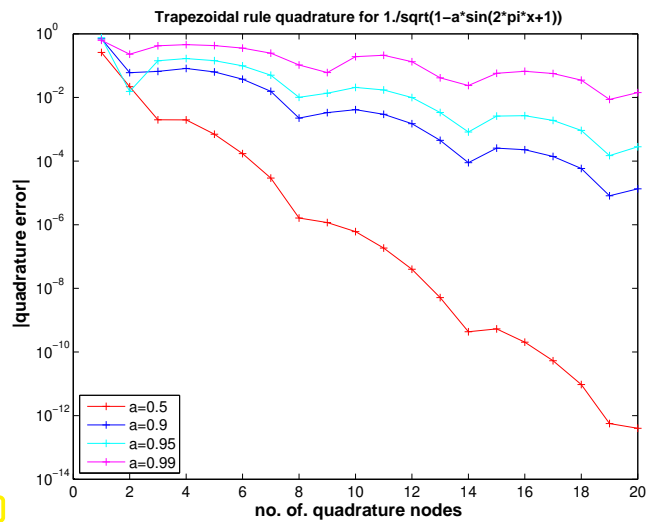


Fig. 138

quadrature error for $T_n(f)$ on $[0, 1]$

Impressive **exponential convergence**

Integrating over the full period boosts the decay of the quadrature error from algebraic to exponential³

§3.4.3.8 (Convergence of the equidistant trapezoidal composite quadrature rule for analytic periodic integrands) We start with a simple observation about the equidistant $N + 1$ -point trapezoidal composite quadrature rule on $[0, 1]$,

$$\int_0^1 f(t) dt \approx T_N(f) := h \left(\frac{1}{2}f(0) + \sum_{k=1}^{N-1} f(kh) + \frac{1}{2}f(1) \right), \quad h := \frac{1}{N} : \tag{3.4.3.9}$$

³For the concepts of algebraic and exponential convergence consult [NumCSE Def. 6.2.2.7].

$$f(t) = e^{2\pi i k t}, \quad k \in \mathbb{Z} \quad \blacktriangleright \quad \begin{cases} \int_0^1 f(t) dt = \begin{cases} 0 & , \text{if } k \neq 0, \\ 1 & , \text{if } k = 0. \end{cases} \\ T_N(f) = \frac{1}{n} \sum_{l=0}^{N-1} e^{\frac{2\pi i}{n} l k} = \begin{cases} 0 & , \text{if } k \notin N\mathbb{Z}, \\ 1 & , \text{if } k \in N\mathbb{Z}. \end{cases} \end{cases} \quad (3.4.3.10)$$

The second identity is a consequence of the geometric sum formula [NumCSE Eq. (4.2.1.9)]. An immediate consequence of (3.4.3.9) is the following result, cf. [NumCSE Lemma 7.5.0.22].

Lemma 3.4.3.11. Exact quadrature by equidistant trapezoidal rule

The $N + 1$ -point equidistant trapezoidal quadrature rule on $[0, 1]$

$$\int_0^1 f(t) dt \approx T_N(f) := h \left(\frac{1}{2}f(0) + \sum_{k=1}^{N-1} f(kh) + \frac{1}{2}f(1) \right), \quad h := \frac{1}{N}. \quad (3.4.3.9)$$

is exact for **trigonometric polynomials** [NumCSE Section 6.5.1] of degree $\leq 2N - 2$, that is, for

$$f \in \mathcal{P}_{2N-1}^T := \text{Span}\{t \mapsto \exp(2\pi i j t) : j = -N + 1, \dots, N - 1\}.$$

Hence the quadrature error can immediately be bounded by the best-approximation error with respect to the space \mathcal{P}_{2N-2}^T of trigonometric polynomials:

$$\left| \int_0^1 f(t) dt - T_N(f) \right| \leq 2 \inf_{q \in \mathcal{P}_{2N-2}^T} \|f - q\|_{L^\infty([0,1])}. \quad (3.4.3.12)$$

That best approximation error can be predicted by means of the following result for trigonometric interpolation [NumCSE Section 6.5.1].

Theorem 3.4.3.13. Exponential convergence of trigonometric interpolation for analytic interpolands, [NumCSE Thm. 6.5.3.14]

If $f : \mathbb{R} \rightarrow \mathbb{C}$ is **1-periodic** and possesses an **analytic extension** to the strip

$$\bar{S}_\eta := \{z \in \mathbb{C} : -\eta \leq \text{Im } z \leq \eta\}, \quad \text{for some } \eta > 0,$$

then there is $C_\eta > 0$ depending only on η such that

$$\|f - I_n^T f\|_* \leq C_\eta e^{-\pi\eta N} \|f\|_{L^\infty(\bar{S})}, \quad N \in \mathbb{N}, \quad (* = L^2(]0,1[), L^\infty(]0,1[)), \quad (3.4.3.14)$$

where $I_n^T : C^0([0,1]) \rightarrow \mathcal{P}_{2N-1}^T$, $N \in \mathbb{N}$, is the **trigonometric interpolation operator** uniquely defined by the interpolation conditions

$$(I_n^T f) \left(\frac{j}{2N-1} \right) = f \left(\frac{j}{2N-1} \right) \quad \forall j = 0, \dots, 2N-2. \quad (3.4.3.15)$$

Summing up, for a **1-periodic** integrand, **analytic** on \bar{S}_η , the quadrature error of equidistant trapezoidal composite quadrature rules on $[0, 1]$ can be bounded as

$$\exists C > 0: \quad \left| \int_0^1 f(t) dt - T_N(f) \right| \leq C e^{-\pi\eta N} \quad \text{for all } N \in \mathbb{N} \dots \quad (3.4.3.16)$$

┘

§3.4.3.17 (CQ weights by quadrature and DFT) In light of the preceding §s it is clear that the ideal quadrature formula for the approximate computation of the CQ weights $w_\ell^{F,\tau}$ in

$$w_\ell^{F,\tau} = r^{-\ell} \int_0^1 e^{-2\pi i \ell \varphi} F\left(\frac{\delta(r e^{2\pi i \varphi})}{\tau}\right) d\varphi, \quad \ell \in \mathbb{Z}, \quad 0 < r < 1. \tag{3.4.3.5}$$

is the equidistant composite trapezoidal rule. Using $N + 1$ quadrature points we obtain

$$w_\ell^{F,\tau} \approx \tilde{w}_\ell^{F,\tau} := \frac{r^{-\ell}}{N+1} \sum_{k=0}^N \exp(-2\pi i \frac{\ell k}{N+1}) f_k, \quad \ell \in \mathbb{Z}, \tag{3.4.3.18}$$

$$f_k := F\left(\frac{1}{\tau} \delta(r \exp(2\pi i \frac{k}{N+1}))\right), \quad k = 0, \dots, N. \tag{3.4.3.19}$$

- We have seen $w_\ell^{F,\tau} = 0$ for $\ell < 0$, but $\tilde{w}_\ell^{F,\tau}$ need not vanish for all $\ell < 0$, $(\tilde{w}_\ell^{F,\tau})_\ell$ is not exactly causal. Fortunately, for r bounded away from 1 and large N those weights will be extremely small, $\tilde{w}_\ell^{F,\tau} \approx 0$.
- We have to link to the number of quadrature nodes to the number M of CQ weights we want to compute in (3.4.3.2). We can choose $N = M$, we can also employ more quadrature nodes, but not less, because then we would face aliasing.

The summations in (3.4.3.18) boils down to a **discrete Fourier transform (DFT)** that we already saw in § 3.1.5.1.

Definition [NumCSE Def. 4.2.1.18]. Discrete Fourier transform (DFT)

The linear map $\text{FFT}_n : \mathbb{C}^n \mapsto \mathbb{C}^n$, $\text{FFT}_n(\mathbf{f}) := \mathbf{F}_n \mathbf{f}$, $\mathbf{F}_n = \left[\exp(-2\pi i u \frac{kj}{n}) \right]_{k,j=0}^{n-1}$ the Fourier matrix (3.1.5.6), $\mathbf{f} \in \mathbb{C}^n$, $n \in \mathbb{N}$, is called **discrete Fourier transform (DFT)**, i.e. for $[y_0, \dots, y_{n-1}] := \text{FFT}_n(\mathbf{f})$

$$y_k = \sum_{j=0}^{n-1} f_j \omega_n^{kj} = \sum_{j=0}^{n-1} f_j \exp(-2\pi i \frac{kj}{n}), \quad k = 0, \dots, n-1. \quad [\text{NumCSE Eq. (4.2.1.19)}]$$

$$\blacktriangleright \quad \text{FFT}_n \mathbf{f} = \left[\sum_{j=0}^{n-1} \exp(-2\pi i \frac{jk}{n}) \right]_{k=0}^{n-1}, \quad \mathbf{f} = [f_0, \dots, f_{n-1}]^\top \in \mathbb{C}^n,$$

$$\text{IFFT}_n \mathbf{y} := \text{FFT}_n^{-1} \mathbf{y} = \left[\frac{1}{n} \sum_{k=0}^{n-1} y_k \exp(2\pi i \frac{kj}{n}) \right]_{j=0}^{n-1}, \quad \mathbf{y} = [y_0, \dots, y_{n-1}] \in \mathbb{C}^n.$$

Pseudocode 3.4.3.20: Efficient implementation of (3.4.3.18), C++ indexing

```

1 Vector ← cqweightsByDFT(
2   FUNCTOR F, FUNCTOR δ, real τ, integer N) {
3   r := 2(N+1)√EPS; // Refer to Rem. 3.4.3.21
4   Vector f ∈ CN+1;

```

```

5 |   for k:=0 to N { f[k] :=  $F(\delta(r \exp(2\pi i k / (N+1))) / \tau)$ ; }
6 |   Vector w  $\in \mathbb{C}^{N+1}$ ;
7 |   w := FFT(f) / (N+1);
8 |   for k:=0 to N { w[k] := w[k] /  $r^k$ ; }
9 |   return w;
10| }

```

The asymptotic computational effort is $O(N \log N)$ for $N \rightarrow \infty$, the cost for the discrete Fourier transform, cf. § 3.1.5.12. \lrcorner

Remark 3.4.3.21 (Choice of integration radius r) In exact arithmetic it would not make a difference how $r \in]0, 1[$ is chosen in (3.4.3.18). Yet, the choice of r very much affects the result when the algorithm is executed in floating-point arithmetic. A deeper round-off error analysis shows that $r := \sqrt[2(N+1)]{\text{EPS}}$, EPS the machine precision [NumCSE Ass. 1.5.3.11], is optimal in terms of numerical stability [BS21, Sect 3.1 & 3.4]. \lrcorner

§3.4.3.22 (All-steps-at-once convolution quadrature) Above we merely computed the CQ weights and the discrete convolution has to be carried out in a second step using the algorithm from § 3.1.5.13 for the multiplication of a Toeplitz matrix with a vector.

It is possible to merge both steps and evaluate (3.4.3.2) with approximate CQ weights $\tilde{w}_\ell^{F,\tau}$ from

$$\begin{aligned}
 w_\ell^{F,\tau} &\approx \tilde{w}_\ell^{F,\tau} := \frac{r^{-\ell}}{N+1} \sum_{k=0}^N \exp(-2\pi i \frac{\ell k}{N+1}) f_k, \quad \ell \in \mathbb{Z}, \\
 f_k &:= F\left(\frac{1}{\tau} \delta(r \exp(2\pi i \frac{k}{N+1}))\right), \quad k = 0, \dots, N,
 \end{aligned} \tag{3.4.3.18}$$

in one fell swoop:

$$\begin{aligned}
 y_n &= \sum_{\ell=0}^n \tilde{w}_{n-\ell}^{F,\tau} g_\ell \stackrel{(*)}{=} \sum_{\ell=0}^N \tilde{w}_{n-\ell}^{F,\tau} g_\ell, \quad g_\ell := g(\ell\tau) \quad [(*) \text{ as } \tilde{w}_j^{F,\tau} \approx 0 \text{ for } j < 0], \\
 &= \sum_{\ell=0}^N \frac{r^{-(n-\ell)}}{N+1} \sum_{k=0}^N \exp\left(-2\pi i \frac{(n-\ell)k}{N+1}\right) f_k g_\ell, \quad f_k := F\left(\frac{\delta(r \exp(2\pi i \frac{k}{N+1}))}{\tau}\right), \\
 &= \frac{r^{-n}}{N+1} \sum_{k=0}^N f_k \left(\sum_{\ell=0}^N r^\ell g_\ell \exp(2\pi i \frac{\ell k}{N+1}) \right) \exp(-2\pi i \frac{kn}{N+1}), \\
 &= r^{-n} \sum_{k=0}^N f_k \left(\text{IFFT}_{N+1} \left[r^\ell g_\ell \right]_{\ell=0}^N \right)_k \exp(-2\pi i \frac{kn}{N+1}), \quad n = 0, \dots, M, \\
 &= r^{-n} \text{FFT}_{N+1} \left(\mathbf{f} \cdot * \text{IFFT}_{N+1} \left[r^\ell g_\ell \right]_{\ell=0}^N \right), \quad n = 0, \dots, M.
 \end{aligned}$$

Be aware of the requirement $N \geq M$. Obviously, this formula involves two *nested DFTs* of length $N+1$.

Pseudocode 3.4.3.23: All-steps-at-once CQ based on a linear multi-step method, scalar setting

```

1 | Vector  $\leftarrow$  asaoCQ(
2 |   FUNCTION  $F$ , FUNCTION  $\delta$ , real  $\tau$ , integer  $N$ , const Vector  $g$ ) {
3 |    $r := \sqrt[2(N+1)]{\text{EPS}}$ ; // Refer to Rem. 3.4.3.21

```

```

4 |   Vector  $\mathbf{h} \in \mathbb{C}^{N+1}$ ,  $\mathbf{y} \in \mathbb{C}^{N+1}$ ;
5 |   for  $k := 0$  to  $N$  do {  $\mathbf{h}[k] := r^k * \mathbf{g}[k]$ ; }
6 |    $\mathbf{t} := \text{IFFT}(\mathbf{h})$ ;
7 |   for  $k := 0$  to  $N$  do {  $\mathbf{t}[k] *= F(\delta(r \exp(2\pi i k / (N + 1))) / \tau)$ ; }
8 |    $\mathbf{y} := \text{FFT}(\mathbf{t})$ ;
9 |   for  $n := 0$  to  $N$  do {  $\mathbf{y}[n] *= r^{-n}$ ; }
10 |  return  $\mathbf{y}$ ;
11 | }

```

The asymptotic computational effort for $N \rightarrow \infty$ is dominated by the FFTs: $O(N \log N)$. ┘

3.5 Runge-Kutta Convolution Quadrature (RKCQ)

The most popular numerical integrators for initial-value problems for ODEs are Runge-Kutta single-step methods (RK-SSM). It is natural to wonder whether they can also be used to approximate the solution $t \mapsto \mathbf{y}(s; t)$ of the initial-value problem

$$\dot{\mathbf{y}}(t) = s\mathbf{y}(t) + \mathbf{g}(t), \quad t \in \mathbb{R}, \quad \mathbf{y}(0) = \mathbf{0}, \quad (3.3.2.3)$$

in the formula

$$F(\partial_t)\mathbf{g}(t) = (f * \mathbf{g})(t) = \int_0^t f(t - \xi)\mathbf{g}(\xi) d\xi = \frac{1}{2\pi i} \int_{\sigma + i\mathbb{R}} F(s)\mathbf{y}(s; t) ds, \quad \begin{matrix} t \geq 0, \\ \sigma > 0. \end{matrix} \quad (3.3.2.5)$$

3.5.1 Implicit Runge-Kutta Single-Step Methods

From [NumCSE Def. 12.3.3.1] we recall the definition of a general implicit Runge-Kutta single-step method for the initial-value problem (IVP)

$$\dot{\mathbf{y}} = \mathbf{f}(t, \mathbf{y}), \quad \mathbf{y}(0) = \mathbf{y}_0 \in \mathbb{R}^N, \quad (3.4.1.2)$$

with right-hand side function $\mathbf{f} : I \times D \rightarrow \mathbb{R}^N$, $D \subset \mathbb{R}^N$ open, $I \subset \mathbb{R}$ and interval.

Definition 3.5.1.1. Implicit Runge-Kutta single-step method, [NumCSE Def. 12.3.3.1]

An m -stage **implicit Runge-Kutta single-step method**, $m \in \mathbb{N}$, applied to (3.4.1.2) with uniform timestep size $\tau > 0$ produces the sequence $(\mathbf{y}_n)_n \in \mathbb{R}^N$ of approximate states according to

$$\mathbf{k}_{i,n} := \mathbf{f}(t_n + c_i\tau, \mathbf{y}_n + \tau \sum_{j=1}^m a_{i,j}\mathbf{k}_{j,n}), \quad i = 1, \dots, m, \quad \mathbf{y}_{n+1} := \mathbf{y}_n + \tau \sum_{i=1}^m b_i\mathbf{k}_{i,n}, \quad n \in \mathbb{N}_0.$$

Here $b_i, a_{i,j} \in \mathbb{R}$, $i, j \in \{1, \dots, m\}$, are given coefficients and $c_i := \sum_{j=1}^m a_{i,j}$, $i = 1, \dots, m$.

From [NumCSE Eq. (12.3.3.3)] recall the **Butcher scheme notation**,

$$\begin{array}{c|c} \mathbf{c} & \mathfrak{A} \\ \hline & \mathbf{b}^T \end{array}, \quad \begin{matrix} \mathfrak{A} := [a_{i,j}]_{i,j=1}^m \in \mathbb{R}^{m,m}, \\ \mathbf{b} := [b_j]_{j=1}^m \in \mathbb{R}^m, \quad \mathbf{c} := [c_i]_{i=1}^m \in \mathbb{R}^m. \end{matrix} \quad (3.5.1.2)$$

We can rewrite the RK-SSM in stage form [NumCSE Rem. 12.3.3.6] setting $\mathbf{v}_{i,n} := \mathbf{y}_n + \tau \sum_{j=1}^m a_{i,j} \mathbf{k}_{j,n}$, $i = 1, \dots, m$:

$$\begin{cases} \mathbf{v}_{i,n} = \mathbf{y}_n + \tau \sum_{j=1}^m a_{i,j} \mathbf{f}(t_n + c_i \tau, \mathbf{v}_{j,n}), & i = 1, \dots, m, \\ \mathbf{y}_{n+1} = \mathbf{y}_n + \tau \sum_{j=1}^m b_j \mathbf{f}(t_n + c_j \tau, \mathbf{v}_{j,n}), \end{cases} \quad n \in \mathbb{N}_0. \quad (3.5.1.3)$$

Applying the RK-SSM in stage form to (3.3.2.3) (where $N = 1$, $\mathbf{f}(t, \mathbf{y}) := s\mathbf{y} + \mathbf{g}(t)$) results in

$$\begin{cases} v_{i,n}(s) = \mathbf{y}_n(s) + \tau \sum_{j=1}^m a_{i,j} (s v_{j,n}(s) + \mathbf{g}(t_n + c_j \tau)), & i = 1, \dots, m, \\ y_{n+1}(s) = \mathbf{y}_n(s) + \tau \sum_{j=1}^m b_j (s v_{j,n}(s) + \mathbf{g}(t_n + c_j \tau)), \end{cases} \quad n \in \mathbb{N}_0. \quad (3.5.1.4)$$

Remark 3.5.1.5. The stages $\mathbf{v}_{i,n}$ provide approximations to $\mathbf{y}(t_n + c_i \tau)$, where $t \mapsto \mathbf{y}(t)$ solves (3.4.1.2). ┘

For the sake of feasibility of the derivation and also simplicity we restrict ourselves to a particular class of implicit m -stage RK-SSMs.

Assumption 3.5.1.6. Requirements on implicit RK-SSM for CQ

- (i) The Butcher matrix $\mathfrak{A} \in \mathbb{R}^{m,m}$ is invertible.
- (ii) The RK-SSM is A-stable [NumCSE Def. 12.3.4.9].
- (iii) We have $b_j = a_{m,j}$, $j = 1, \dots, m$.

Remark 3.5.1.7 (Stiffly accurate RK-SSMs) Runge-Kutta single-step methods satisfying Item (iii) are called **stiffly accurate**. According to the considerations of [NumCSE Rem. 12.3.4.18], Item (ii) and Item (iii) imply that the RK-SSM is **L-stable** [NumCSE Def. 12.3.4.15]. ┘

EXAMPLE 3.5.1.8 (Important stiffly accurate RK-SSMs) The following two implicit Runge-Kutta single-step methods comply with Ass. 3.5.1.6:

$$\begin{array}{c|cc} \frac{1}{3} & \frac{5}{12} & -\frac{1}{12} \\ \frac{1}{2} & \frac{1}{6} & -\frac{1}{12} \\ 1 & \frac{1}{6} & \frac{1}{6} \\ \hline & \frac{3}{4} & \frac{1}{4} \end{array} \quad (3.5.1.9)$$

$$\begin{array}{c|ccc} 0 & \frac{1}{6} & -\frac{1}{3} & \frac{1}{6} \\ \frac{1}{2} & \frac{1}{6} & \frac{5}{12} & -\frac{1}{12} \\ 1 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{array} \quad (3.5.1.10)$$

2-stage Radau IIA, order = 3

3-stage, Lobatto IIIC, order = 4

These are only two simple members of families of Radau *A and Lobatto *C methods, which can be of arbitrarily high order. ┘

3.5.2 Runge-Kutta CQ weights

Obviously, for stiffly-accurate RK-SSMs the last stage gives the next state, $\mathbf{v}_{m,n} = \mathbf{y}_{n+1}$, which renders the second equation in (3.5.1.3) and (3.5.1.4) redundant. Thus, under assumption Ass. 3.5.1.6, an equivalent

way of writing (3.5.1.4) is

$$v_{i,n}(s) = v_{m,n-1}(s) + s\tau \sum_{j=1}^m a_{i,j}v_{j,n}(s) + \tau \sum_{j=1}^m a_{i,j}g(t_n + c_j\tau), \quad i = 1, \dots, m, \quad n \in \mathbb{N}_0. \quad (3.5.2.1)$$

§3.5.2.2 (Runge-Kutta recurrence in z-domain) We want to adapt the z-transform approach of Section 3.4.2 to (3.5.2.1). The difficulty is that, evidently, the causal function g is sampled on the *non-uniform* temporal grid $\{t_n + c_i\tau\}_{i,n}$. This seems to rule out the use of the z-transform.



Rewrite (3.5.2.1) as a recurrence relation for the *vectors*

$$\vec{v}_n(s) := [v_{i,n}(s)]_{i=1}^m \in \mathbb{C}^m, \quad \vec{g}_n := [g(t_n + c_i\tau)]_{i=1}^m \in \mathbb{C}^m, \quad n \in \mathbb{N}_0. \quad (3.5.2.3)$$

$$(3.5.2.1) \quad \blacktriangleright \quad \vec{v}_n(s) = \mathbf{1e}_m^\top \vec{v}_{n-1}(s) + \tau s \mathfrak{A} \vec{v}_n(s) + \tau \mathfrak{A} \vec{g}_n \quad (3.5.2.4)$$

$$\Leftrightarrow (\mathbf{I}_m - \tau s \mathfrak{A}) \vec{v}_n(s) = \mathbf{1e}_m^\top \vec{v}_{n-1}(s) + \tau \mathfrak{A} \vec{g}_n. \quad (3.5.2.5)$$

where $\mathbf{1} := [1, \dots, 1]^\top \in \mathbb{R}^m$, $\mathbf{e}_m = [0, \dots, 0, 1] \in \mathbb{R}^m$, and $\mathfrak{A} \in \mathbb{R}^{m,m}$ is the Butcher matrix from (3.5.1.2). We z-transform (3.5.2.5):

$$\begin{aligned} \sum_{n=0}^{\infty} (\mathbf{I}_m - \tau s \mathfrak{A}) \vec{v}_n(s) z^n &= \sum_{n=0}^{\infty} \mathbf{1e}_m^\top \vec{v}_{n-1}(s) z^n + \sum_{n=0}^{\infty} \tau \mathfrak{A} \vec{g}_n z^n \\ \Leftrightarrow (\mathbf{I}_m - \tau s \mathfrak{A}) \vec{V}(z) &= z \mathbf{1e}_m^\top \vec{V}(s; z) + \tau \mathfrak{A} \vec{G}(z). \end{aligned}$$

where we abbreviated the z-transforms

$$\vec{V}(s; z) := \sum_{n=0}^{\infty} \vec{v}_n(s) z^n \in \mathbb{C}^m, \quad \vec{G}(z) := \sum_{n=0}^{\infty} \vec{g}_n z^n \in \mathbb{C}^m$$

We can solve for $\vec{V}(s; z)$:

$$\vec{V}(s; z) = (\Delta(z) / \tau - s \mathbf{I}_m)^{-1} \vec{G}(z) \in \mathbb{C}^m, \quad \Delta(z) := \mathfrak{A}^{-1} (\mathbf{I}_m - z \mathbf{1e}_m^\top) \in \mathbb{C}^{m,m}, \quad (3.5.2.6)$$

unless τs is an eigenvalue of $\Delta(z)$. This is the Runge-Kutta counterpart of (3.4.2.13). ┘

§3.5.2.7 (Vector-expanded Runge-Kutta convolution quadrature) Carrying on with the “vectorization” idea of § 3.5.2.2, we conclude that convolution quadrature, so far written as

$$(F(\partial_t)g)(\tau n) \approx \sum_{\ell=0}^n w_{n-\ell}^{F,\tau} g(\tau \ell), \quad w_j^{F,\tau} := (\text{CQ}_\tau(F))_j \in \mathbb{C}, \quad n \in \mathbb{N}_0, \quad (3.3.1.5)$$

should become

$$(F(\partial_t)\vec{g})|_{\mathcal{G}_\tau} \approx \left(\sum_{\ell=0}^n \mathbf{W}_{n-\ell}^{F,\tau} \vec{g}_\ell \right)_{n \in \mathbb{N}_0} \quad \text{with } \textit{matrix-valued} \text{ weights } \mathbf{W}_{n-\ell}^{F,\tau} \in \mathbb{C}^{m,m}. \quad (3.5.2.8)$$

Stretching notation we wrote $\vec{g}(t) := [g(t + c_1\tau), \dots, g(t + c_m\tau)]^\top \in \mathbb{C}^m$ and $\vec{g}_\ell := \vec{g}(\ell\tau) \in \mathbb{C}^m$. Using those notations we obtain via (3.3.2.5)

$$(F(\partial_t)\vec{g})(t) = \frac{1}{2\pi i} \int_{\sigma+i\mathbb{R}} F(s) \vec{y}(s; t) ds, \quad t \in \mathbb{R}, \sigma > 0, \quad (3.5.2.9)$$

where $t \mapsto \vec{y}(s; t) \in \mathbb{C}^m$ solves the IVP

$$\frac{\partial \vec{y}(s; t)}{\partial t} = s \vec{y}(s; t) + \vec{g}(t), \quad s \in \mathbb{C}, \quad \vec{y}(s; 0) = \mathbf{0}, \tag{3.5.2.10}$$

Remember that the stages computed during the execution of an implicit RK-SSM (with uniform timestep $\tau > 0$) provide approximations of the solution trajectory at times $t_j + c\tau, t_j \in \mathcal{G}_\tau$. Thus

$$\vec{v}_n(s) \approx \vec{y}(s; t_n) \quad \forall n \in \mathbb{N}_0, \quad \vec{v}_n(s) \text{ as in (3.5.2.3)}. \tag{3.5.2.11}$$

In the spirit of § 3.3.2.1 we approximate

$$(F(\partial_t)\vec{g})(\tau n) \approx \frac{1}{2\pi i} \int_{\sigma+i\mathbb{R}} F(s) \vec{v}_n(s) ds \stackrel{!}{=} \sum_{\ell=0}^n \mathbf{W}_{n-\ell}^{F, \tau} \vec{g}_\ell. \tag{3.5.2.12}$$

As in § 3.4.2.17 we write $\mathbf{Q}(z) \in \mathbb{C}^{m,m}$ for the (matrix-valued!) z -transform of the matrix sequence $(\mathbf{W}_j^{F, \tau})_{j=0}^\infty$. In analogy to (3.4.2.20), z -transforming both sides of $\stackrel{!}{=}$ in (3.5.2.12) we obtain

$$\mathbf{Q}(z) \vec{G}(z) = \frac{1}{2\pi i} \int_{\sigma+i\mathbb{R}} F(s) \vec{V}(s; z) ds, \quad \vec{V}(s; z) \triangleq z\text{-transform of } (\vec{v}_n(s))_n. \tag{3.5.2.13}$$

The next steps run parallel to those in § 3.4.2.17. They also require Ass. 3.3.4.3 concerning the transfer function $F : \mathbb{C}^+ \rightarrow \mathbb{C}$. We insert the explicit formula (3.5.2.6) for $\vec{V}(s; z)$ into (3.5.2.13), deform the path of integration to a closed contour enclosing the spectrum of $\Delta(z)/\tau$, invoke the Cauchy integral formula Thm. 3.3.2.22, and “cancel” $\vec{G}(z)$, which yields

$$\mathbf{Q}(z) = \sum_{n=0}^\infty \mathbf{W}_n^{F, \tau} z^n = F\left(\frac{\Delta(z)}{\tau}\right). \tag{3.5.2.14}$$

The Runge-Kutta convolution quadrature weights turn out to be the Taylor series coefficients of the analytic *matrix-valued* function $z \mapsto F\left(\frac{\Delta(z)}{\tau}\right)$ at $z = 0$.

Note that thanks to Ass. 3.5.1.6, the use of a stiffly accurate RK-SMM for which the m -th stage also supplies the next state, we can directly extract an approximate discrete convolution from (3.5.2.8),

$$(F(\partial_t)g)(\tau n) \approx \left(\sum_{\ell=0}^{n-1} \mathbf{W}_{n-\ell}^{F, \tau} \vec{g}_\ell \right)_m, \quad n \in \mathbb{N}. \tag{3.5.2.15}$$

┘

Remark 3.5.2.16 (Matrix functions) In (3.5.2.14) we plug a matrix into the function F . What does this mean? If F is a polynomial, the meaning of $F(\mathbf{M})$, $\mathbf{M} \in \mathbb{C}^{m,m}$, is clear,

$$F(z) = \sum_{n=0}^p \alpha_n z^n \quad \blacktriangleright \quad F(\mathbf{M}) := \sum_{n=0}^p \alpha_n \mathbf{M}^n \in \mathbb{C}^{m,m}.$$

If F can be represented by a power series with *radius of convergence* $> \|\mathbf{M}\|$ for some matrix norm, then

$$F(z) = \sum_{n=0}^\infty \alpha_n z^n \quad \blacktriangleright \quad F(\mathbf{M}) := \sum_{n=0}^\infty \alpha_n \mathbf{M}^n \in \mathbb{C}^{n,n}, \tag{3.5.2.17}$$

because the series will converge in $\mathbb{C}^{m,m}$. What if F is analytic, but the radii of convergence of *local* power series expansions may not be large enough to permit us to plug in the matrix \mathbf{M} ? Assume that \mathbf{M} can be diagonalized,

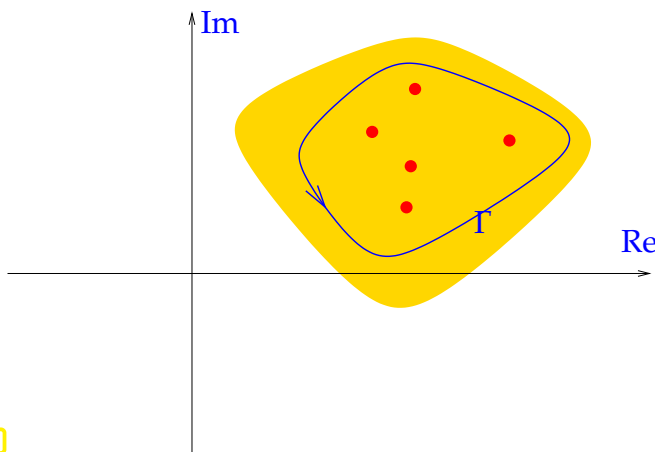
$$\exists \mathbf{S} \in \mathbb{C}^{m,m}, \mathbf{S} \text{ regular}, \lambda_i \in \mathbb{C}: \mathbf{S}^{-1}\mathbf{M}\mathbf{S} = \mathbf{D} := \text{diag}(\lambda_1, \dots, \lambda_m).$$

Then we can rewrite (3.5.2.17)

$$\begin{aligned} F(\mathbf{M}) &= \sum_{n=0}^{\infty} \alpha_n (\mathbf{S}\mathbf{D}\mathbf{S}^{-1})^n = \mathbf{S} \text{diag} \left(\sum_{n=0}^{\infty} \alpha_n \lambda_i^n \right)_{i=1}^m \mathbf{S}^{-1} \\ &= \mathbf{S} \text{diag}(F(\lambda_1), \dots, F(\lambda_m)) \mathbf{S}^{-1}, \end{aligned} \tag{3.5.2.18}$$

which can be used to make sense of \mathbf{M} for any matrix $\mathbf{M} \in \mathbb{R}^{m,m}$ that can be diagonalized provided that $F(\lambda)$ is well-defined for every eigenvalue λ of \mathbf{M} .

Can we avoid diagonalization of the matrix argument $\mathbf{M} \in \mathbb{C}^{m,m}$? This is possible, if F is *analytic/holomorphic* in a simply connected open set $D \subset \mathbb{C}$ that contains all eigenvalues of \mathbf{M} .



- ◁ ■ $\hat{=}$ domain D of holomorphy of F
- $\hat{=}$ eigenvalue λ_i of \mathbf{M}
- $\hat{=}$ contour of integration

If $\Gamma \subset D$ is a positively oriented contour of integration enclosing all eigenvalues $\lambda_i, i = 1, \dots, m$, of \mathbf{M} , then by the Cauchy integral formula

$$F(\lambda_i) = \frac{1}{2\pi i} \int_{\Gamma} \frac{F(s)}{s - \lambda_i} ds, \quad i = 1, \dots, m.$$

Fig. 139

We can insert this formula into (3.5.2.18), provided that Γ once winds around (inside D , of course) the whole set of eigenvalues, the spectrum, of \mathbf{M} :

$$\begin{aligned} F(\mathbf{M}) &= \mathbf{S} \text{diag} \left(\frac{1}{2\pi i} \int_{\Gamma} F(s)(s - \lambda_i)^{-1} ds \right)_{i=1}^m \mathbf{S}^{-1} \\ &= \mathbf{S} \frac{1}{2\pi i} \int_{\Gamma} F(s) (s\mathbf{I}_m - \text{diag}(\lambda_1, \dots, \lambda_m))^{-1} ds \mathbf{S}^{-1} \\ &= \frac{1}{2\pi i} \int_{\Gamma} F(s) (s\mathbf{I}_m - \mathbf{M})^{-1} ds. \end{aligned} \tag{3.5.2.19}$$

We could remove all traces of diagonalization! This formula can be used in case \mathbf{M} cannot be diagonalized. ┘

§3.5.2.20 (Computation of Runge-Kutta convolution quadrature weights) As explained in § 3.4.3.3, we can use the Cauchy differentiation formula Cor. 3.3.2.23 to obtain from (3.5.2.14)

$$\begin{aligned} \mathbf{W}_{\ell}^{F,\tau} &= \frac{1}{\ell!} \frac{d^{\ell}}{dz^{\ell}} \left\{ z \mapsto F\left(\frac{\Delta(z)}{\tau}\right) \right\}_{z=0} = \frac{1}{2\pi i} \int_{|z|=r} \frac{1}{z^{\ell+1}} F\left(\frac{\Delta(z)}{\tau}\right) dz, \\ &= r^{-\ell} \int_0^1 e^{-2\pi i \ell \varphi} F\left(\frac{\Delta(re^{2\pi i \varphi})}{\tau}\right) d\varphi, \quad \ell \in \mathbb{N}_0, \end{aligned} \tag{3.5.2.21}$$

for all sufficiently small $r > 0$. We approximate the integral, which features an analytic and 1-periodic integrand, by means of the equidistant $N + 1$ -point trapezoidal composite quadrature rule.

$$\mathbf{W}_\ell^{F,\tau} \approx \widetilde{\mathbf{W}}_\ell^{F,\tau} := \frac{r^{-\ell}}{M+1} \sum_{k=0}^{M+1} \exp(-2\pi i \frac{\ell k}{M+1}) F\left(\frac{1}{\tau} \Delta(re^{2\pi i \frac{k}{N+1}})\right). \quad (3.5.2.22)$$

Refer to Code 3.4.3.20 for the efficient **FFT-based** evaluation of the sum (3.5.2.22). In practical implementations the evaluation of the matrix function $F(\dots)$ will usually rely on diagonalization as in (3.5.2.18). This is no undue effort, because matrix sizes are small, $m \in \{2, 3, 4\}$. \lrcorner

Remark 3.5.2.23 (Algorithms for Runge-Kutta-based convolution quadrature) It goes without saying that all the algorithms elaborated for multi-step CQ in § 3.4.3.22 carry over to Runge-Kutta CQ. \lrcorner

3.6 Fast and Oblivious Convolution Quadrature

EXAMPLE 3.6.0.1 (Convolution evolution partial differential equations) Let $\Omega \subset \mathbb{R}^d$, $d = 2, 3$, be a spatial domain, $f : [0, T] \rightarrow L^2(\Omega)$ a time-dependent source function. Then the **convolution evolution PDE** induced by the transfer function $F : \mathbb{C}^+ \rightarrow \mathbb{C}$

$$F(\partial_t)u(x, t) - \Delta_x u(x, t) = f(x, t) \quad \text{in } \Omega \times]0, T[\quad , \quad u(\cdot, t) = 0 \quad \text{on } \partial\Omega \times]0, T[\quad , \quad (3.6.0.2)$$

can be regarded as a generalization of the heat equation $\dot{u} - \Delta_x u = f$, to which it boils down when $F(s) := s$. Here we relied on the operational calculus notation of Def. 3.1.4.5. Seeking causal solutions of (3.6.0.2) we supplement it with the “initial condition” $u(\cdot, t) = 0$ for all $t < 0$.

In the spirit of the **method of lines** [NumPDE Section 9.2.4] we perform a spatial Galerkin (finite-element) discretization based on an N -dimensional discrete trial and test space $V_h \subset H_0^1(\Omega)$, $N \in \mathbb{N}$, $N \gg 1$, equipped with an ordered basis $\{b_h^1, \dots, b_h^N\}$. This yields the generalization of the method-of-lines ODE [NumPDE Eq. (9.2.4.4)] for the time-dependent basis expansion coefficient vector $t \mapsto \vec{\mu}(t) \in \mathbb{C}^N$ of the spatially semidiscrete solution $t \mapsto u_h(t) \in V_h$:

$$(F(\partial_t)\mathbf{M}\vec{\mu})(t) + \mathbf{A}\vec{\mu}(t) = \vec{\varphi}(t) \quad , \quad t \in [0, T] \quad , \quad \vec{\mu}(t) = 0 \quad \forall t < 0 \quad , \quad (3.6.0.3)$$

$$\text{with } \mathbf{M} := \left[\int_{\Omega} b_h^i(x) b_h^j(x) dx \right]_{i,j=1}^N \in \mathbb{R}^{N,N} \quad ,$$

$$\mathbf{A} := \left[\int_{\Omega} \mathbf{grad} b_h^i(x) \cdot \mathbf{grad} b_h^j(x) dx \right]_{i,j=1}^N \in \mathbb{R}^{N,N} \quad , \quad (3.6.0.4)$$

$$\vec{\varphi}(t) := \left[\int_{\Omega} f(x) b_h^i(x) dx \right]_{i=1}^N \in \mathbb{R}^N \quad .$$

The equation (3.6.0.3) can be rewritten as a true causal **convolution equation**

$$(\mathbf{G}(\partial_t)\vec{\mu})(t) = \vec{\varphi}(t) \quad , \quad t \in \mathbb{R} \quad , \quad \text{with } \mathbf{G}(s) := F(s)\mathbf{M} + \mathbf{A} \quad , \quad (3.6.0.5)$$

which fits the framework of convolution in an operator setting § 3.1.1.16. Its solution is given by

$$\vec{\mu}(t) = (\mathbf{G}^{-1}(\partial_t)\vec{\varphi})(t) \quad \text{with } \mathbf{G}^{-1}(s) := (F(s)\mathbf{M} + \mathbf{A})^{-1} \quad , \quad s \in \mathbb{C}^+ \quad . \quad (3.6.0.6)$$

Since both \mathbf{M} and \mathbf{A} are symmetric positive definite (s.p.d.) $\mathbf{G}(s)$ is invertible for all $s \in \mathbb{C}^+$, provided that $F(\mathbb{C}^+) \subset \mathbb{C}^+$.

We employ (multi-step) convolution quadrature to discretize (3.6.0.6) in time. Writing $\mathbf{W}_\ell^{\mathbf{G}^{-1}, \tau} \in \mathbb{C}^{N, N}$ for the matrix-valued CQ weights for uniform timestep $\tau := T/M$, $M \in \mathbb{N}$, we get the a discrete convolution formula for $\vec{\mu}_n \in \mathbb{C}^N$, $n = 0, \dots, M$:

$$\vec{\mu}(\tau n) \approx \mu_n := \sum_{\ell=0}^n \mathbf{W}_{n-\ell}^{\mathbf{G}^{-1}, \tau} \vec{\phi}_\ell, \quad n = 0, \dots, M, \quad \vec{\phi}_\ell := \vec{\phi}(\tau \ell). \quad (3.6.0.7)$$

The coefficient vectors $\vec{\mu}_n \in \mathbb{C}^N$ can now be computed with the algorithm of § 3.4.3.22, which will require (assuming that \mathbf{M} and \mathbf{A} are finite-element Galerkin matrices with only a few non-zero entries per row/column)

- $O(N \cdot M \log M)$ computational effort for $N, M \rightarrow \infty$ (excluding \mathbf{G}^{-1} -evaluations),
- $O(N \cdot M)$ memory for $N, M \rightarrow \infty$ (\mathbf{G}^{-1} -evaluations not taken into account),
- $O(M)$ evaluations of $s \mapsto \mathbf{G}^{-1}(s)$, each of which amounts to solving a different $N \times N$ linear system of equations.

Compare this with the fully discrete heat equation in the same setting using implicit Euler timestepping. This will involve an asymptotic (for $M, N \rightarrow \infty$) computational effort of $O(N \cdot M)$ (ignoring linear solves), require only $O(N)$ memory, and M solves of the same $N \times N$ sparse linear system of equations. \lrcorner

§3.6.0.8 (Convolution with sums of exponentials) Consider the causal function

$$f(t) = \begin{cases} \sum_{j=1}^m c_j e^{\lambda_j t} & \text{for } t \geq 0, \\ 0 & \text{for } t < 0, \end{cases} \quad c_j, \lambda_j \in \mathbb{C}, \quad (3.6.0.9)$$

whose Laplace transform is a rational function,

$$F(s) := (\mathcal{L}f)(s) = \sum_{j=1}^m \frac{c_j}{s - \lambda_j}, \quad s \in \mathbb{C} \setminus \{\lambda_1, \dots, \lambda_j\}. \quad (3.6.0.10)$$

Thanks to the **variation-of-constants** formula of Lemma 3.3.2.2, a convolution with f can be reduced to solving m initial value problems for linear ODEs:

$$(f * g)(t) = \int_0^t f(t - \xi) g(\xi) d\xi = \sum_{j=1}^m c_j \int_0^t e^{\lambda_j(t-\xi)} g(\xi) d\xi = \sum_{j=1}^m c_j y(t; \lambda_j), \quad (3.6.0.11)$$

where $t \mapsto y(t; s)$ stands for the (unique) solution of the initial value problem ($g : \mathbb{R} \rightarrow \mathbb{C}$ causal)

$$\frac{dy(t; s)}{dt} = sy(t; s) + g(t) \quad , \quad y(0; s) = 0. \quad (3.6.0.12)$$

The formula (3.6.0.11) remains valid in a matrix setting. If

$$\mathbf{F}(t) = \begin{cases} \sum_{j=1}^m \mathbf{C}_j \exp(\lambda_j t) & \text{for } t \geq 0, \\ \mathbf{O} & \text{for } t < 0, \end{cases} \quad \mathbf{C}_j \in \mathbb{C}^{N, N}, \quad \lambda_j \in \mathbb{C}, \quad N \in \mathbb{N}, \quad (3.6.0.13)$$

and $\mathbf{g} : \mathbb{R} \rightarrow \mathbb{C}^N$ is causal, then

$$(\mathbf{F} * \mathbf{g})(t) = \int_0^t \mathbf{F}(t - \xi) \cdot \mathbf{g}(\xi) d\xi = \sum_{j=1}^m \mathbf{C}_j \int_0^t \exp(\lambda_j(t - \xi)) \mathbf{g}(\xi) d\xi = \sum_{j=1}^m \mathbf{C}_j \mathbf{y}(t; \lambda_j), \quad (3.6.0.14)$$

with $t \mapsto \mathbf{y}(t; \lambda) \in \mathbb{C}^N$ solving

$$\frac{d\mathbf{y}(\lambda; \cdot)}{dt}(t) = \lambda \mathbf{y}(t; \lambda) + \mathbf{g}(t) \quad , \quad \mathbf{y}(0; \lambda) = 0 . \tag{3.6.0.15}$$

Hence, (3.6.0.14) can be discretized by applying (implicit) timestepping to m initial value problems of the type (3.6.0.15) with $\lambda \leftarrow \lambda_j$. If we use the implicit Euler single-step method with timestep $\tau > 0$, we obtain

$$\mathbf{y}_{n,j} := (1 - \tau \lambda_j)^{-1} (\mathbf{y}_{n-1,j} + \tau \mathbf{g}(n\tau)) \quad , \quad n = 1, \dots, m \quad , \quad (\mathbf{F} * \mathbf{g})(\tau n) \approx \sum_{j=1}^m \mathbf{C}_j \mathbf{y}_{n,j} .$$

Obviously, M uniform timesteps will require $O(mN)$ memory and computing mM matrix \times vector products. ┘

For the sake of lucidity, we temporarily restrict ourselves to the scalar case, $f : \mathbb{R} \rightarrow \mathbb{C} \leftrightarrow F : \mathbb{C}^+ \rightarrow \mathbb{C}$, $g : \mathbb{R} \rightarrow \mathbb{C}$.

Remark 3.6.0.16 (Exponential sum approximation by quadrature) If $F : \mathbb{C}^+ \rightarrow \mathbb{C}$, $F(s) := (\mathcal{L}f)(s)$ is analytic/holomorphic in \mathbb{C}^+ and $|F(s)| \leq C|s|^{-\mu}$ on \mathbb{C}^+ for $\mu > 1$ and $C > 0$, then f can be computed by the Bromwich contour integral, see Thm. 3.1.3.13: for any $\sigma > 0$

$$f(t) = (\mathcal{L}^{-1}F)(t) = \frac{1}{2\pi i} \int_{\sigma + i\mathbb{R}} F(s) e^{st} ds = \frac{e^\sigma}{2\pi} \int_{-\infty}^{\infty} F(\sigma + i\theta) e^{i\theta t} d\theta . \tag{3.1.3.15}$$

Approximating this integral by means of an m -point quadrature formula on $]-\infty, \infty[$ with weights ω_j and nodes $\lambda_j \in \mathbb{R}$ we obtain

$$f(t) \approx \frac{e^\sigma}{2\pi} \sum_{j=1}^m \omega_j F(\sigma + i\lambda_j) e^{i\lambda_j t} .$$

This is a weighted *sum of exponentials*!

Unfortunately this is hardly ever feasible, unless F decays rapidly, because

- the integrand $\theta \mapsto F(\sigma + i\theta) e^{i\theta t}$ will be *rapidly oscillating*, in particular for larger values of $|t|$, and
 - the decay of that integrand for $|\theta| \rightarrow \infty$ will usually be rather slow: $O(|\theta|^{-\mu})$ as $\theta \rightarrow \infty$.
- ┘

§3.6.0.17 (Sectorial transfer functions) The incredible power of the Cauchy integral theorem stated in Thm. 3.1.3.16 breathes new life into the idea floated in the previous §, at least for a rather large class of transfer functions F .

Definition 3.6.0.18. Sectorial transfer function/-parabolic symbol

A function $F : \mathbb{C}^+ \rightarrow \mathbb{C}$ is called **α -sectorial**, $0 \leq \alpha < \frac{\pi}{2}$, if F possesses an *analytic extension* to

$$\mathbb{C}^-(\alpha) := \{z = re^{i\varphi} \in \mathbb{C} : -\pi + \alpha \leq \varphi \leq \pi - \alpha\} .$$

■ $\hat{=}$ domain of analyticity of α -sectorial F \triangleright

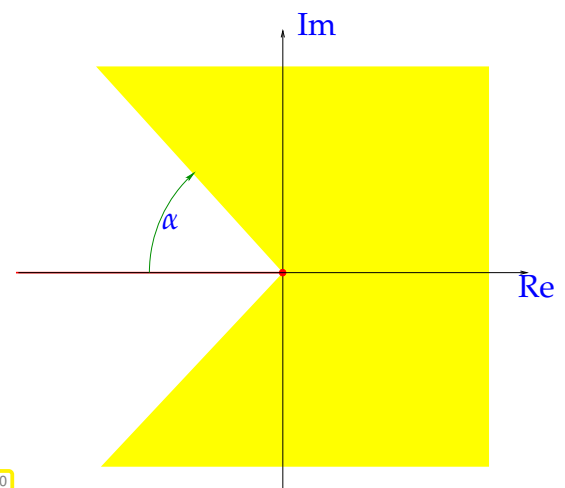


Fig. 140

A prime example of sectorial transfer functions are the power functions

$$F(s) = s^q = \exp(q \log(s)), \quad q \in \mathbb{R} \setminus \mathbb{N}_0, \quad \text{analytic in } \mathbb{C} \setminus \mathbb{R}_0^- \quad \blacktriangleright \quad \text{0-sectorial}. \quad (3.6.0.19)$$

Let us make the rather mild assumption

$$|F(s)| \rightarrow 0 \quad \text{uniformly as } |s| \rightarrow \infty.$$

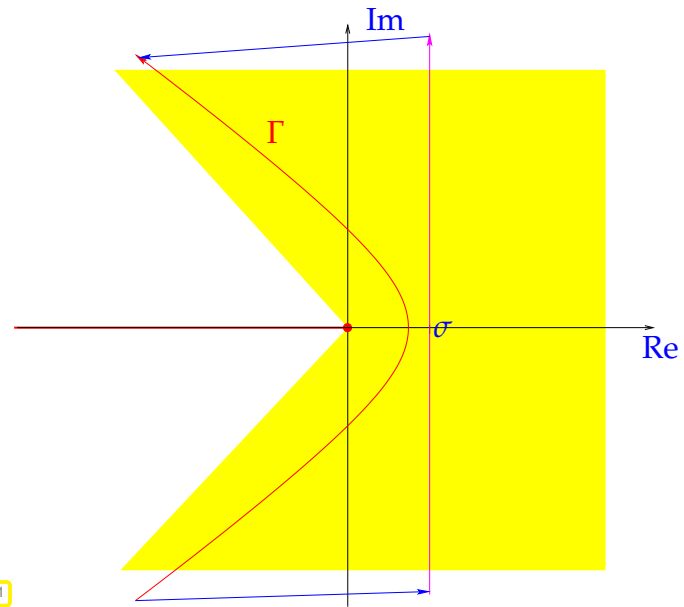
Then by the Cauchy integral theorem

$$\int_{\Gamma} F(s)e^{st} ds = \int_{\sigma+i\mathbb{R}} F(s)e^{st} ds \quad (3.6.0.20)$$

where $\Gamma \subset \mathbb{C}^-(\alpha)$ is the left-bending contour \rightarrow in Fig. 141. \triangleright

Note that the integrals over the \rightarrow -paths in Fig. 141 tends to zero as those “move towards ∞ ”.

Fig. 141



The crucial observation is that $s \mapsto F(s)e^{st}$ decays to zero exponentially as $s \in \Gamma \rightarrow \infty$. This is consequence of $t > 0$, $|F(s)e^{st}| = |F(s)|e^{\text{Re}\{s\}t}$, and $\text{Re}\{s\} < 0$ on the “far parts” of Γ . This may pave the way for sufficiently accurate numerical quadrature on Γ . \lrcorner

§3.6.0.21 (Left-bending paths of integration for α -sectorial functions) We define the contour of integration through a parameterization $\gamma : I \rightarrow \mathbb{C}, I \subset \mathbb{R}$.

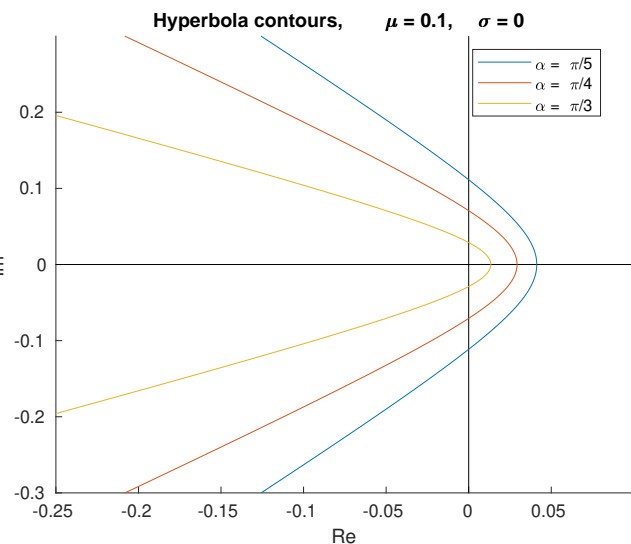
For α -sectorial F :

Hyperbola contours

$$\gamma_H(\theta) := \mu \sin(\varphi + i\theta), \quad \theta \in \mathbb{R}, \quad (3.6.0.22) \Xi$$

with $\varphi < \pi/2 - \alpha$. The parameter φ controls the “opening angle” and $\mu > 0$ is a stretching parameter.

Fig. 142



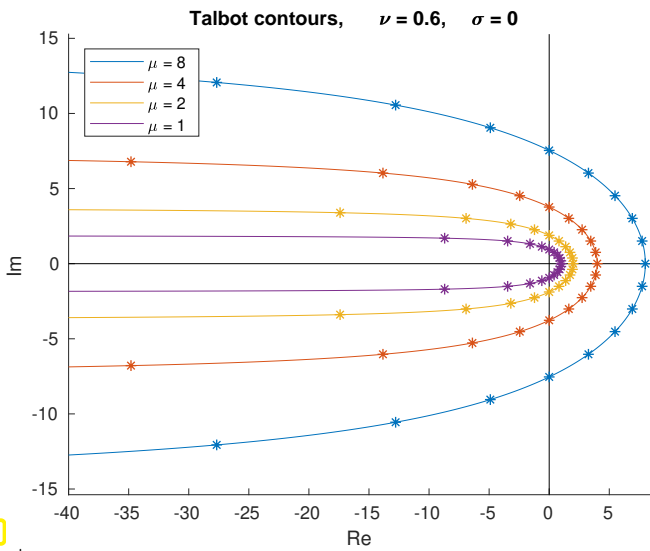


Fig. 143

For 0-sectorial F :

argument

Talbot contour

$$\gamma_T(\mu; \theta) = \mu(-\theta \cot(\theta) + \nu\theta), \quad (3.6.0.23)$$

$$-\pi < \theta < \pi,$$

with width parameter $\nu > 0$ and stretching parameter $\mu > 0$.

◁ These integration contours lie in $\mathbb{C}^-(0)$ and have asymptotes parallel to the real axis.

EXAMPLE 3.6.0.24 (Quadrature over Talbot contour) On the Talbot contour Γ (3.6.0.23) we use the equidistant **trapezoidal rule** on the parameter domain.

$$\frac{1}{2\pi i} \int_{\Gamma} F(s) e^{st} ds = \frac{1}{2\pi i} \int_{-\pi}^{\pi} F(\gamma_T(\mu; \theta)) \exp(\gamma_T(\theta)t) \frac{d\gamma_T(\mu; \cdot)}{d\theta}(\theta) d\theta$$

$$\approx \tilde{f}_{m,\mu}(t) := \frac{1}{2\pi i} \cdot \frac{1}{2m} \sum_{j=-m+1}^{m-1} F(\gamma_T(\mu; \theta_j)) \exp(\gamma_T(\mu; \theta_j)t) \frac{d\gamma_T(\mu; \cdot)}{d\theta}(\theta_j), \quad \theta_j := \frac{\pi j}{m}. \quad (3.6.0.25)$$

The mapped quadrature nodes $\gamma_T(\theta_j)$ are marked with $*$ in Fig. 143. Note that the integrand decays exponentially to zero as $\theta \rightarrow \pm\pi$, which permits us to drop the endpoints. In light of § 3.4.3.8 this also guarantees exponential convergence $\rightarrow 0$ of the quadrature error for $m \rightarrow \infty$.

For $F(s) = \frac{1}{\sqrt{s}}$ ($\leftrightarrow f(t) = \frac{1}{\sqrt{\pi t}}$) we report the absolute quadrature error as a function of $t \in [0.001, 2]$.

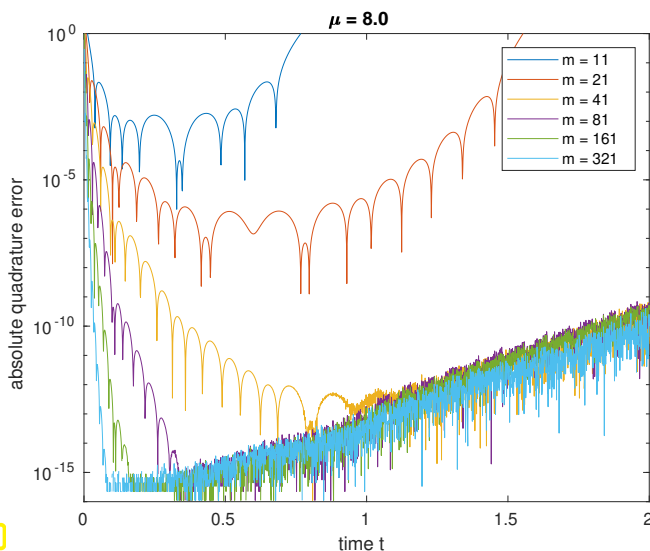


Fig. 144

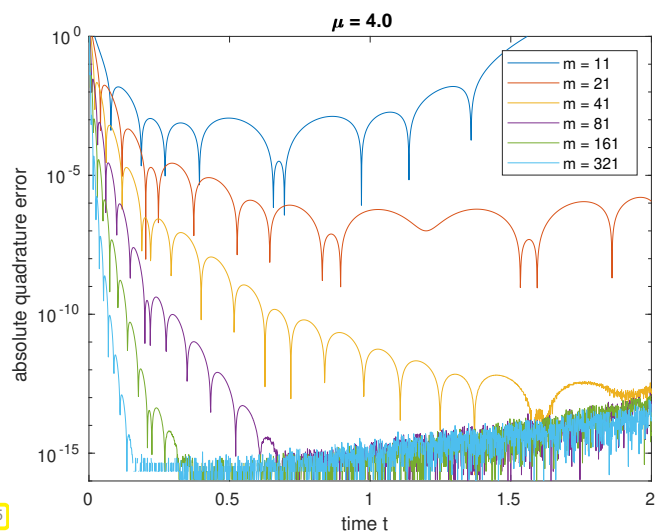


Fig. 145

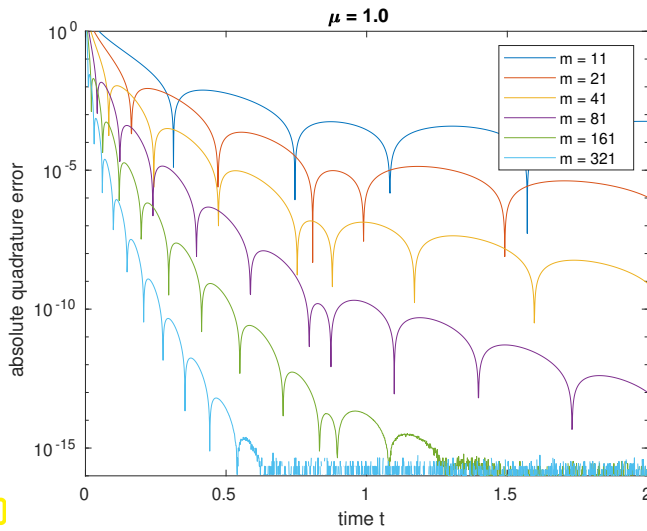


Fig. 146

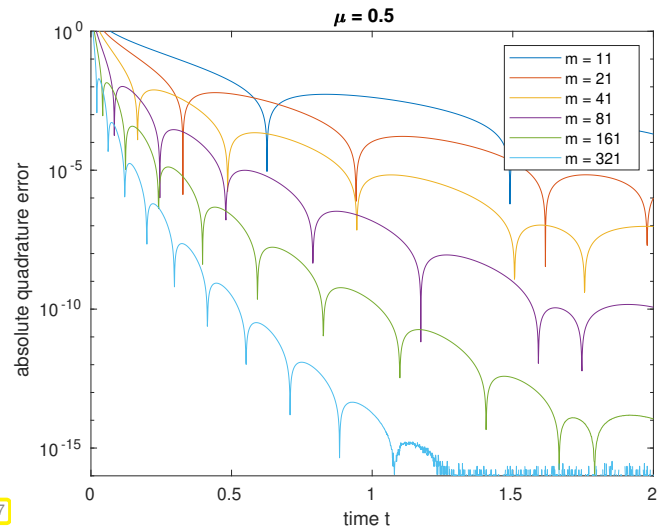


Fig. 147

- We observe that the quadrature error strongly depends on t .
- For very small times $0 < t \ll 1$ quadrature approximation struggles to achieve any accuracy.
- For different values of the Talbot contour parameter μ the quadrature error is small(est) for different ranges of t .
- Convergence $\tilde{f}_{m,\mu}(t) \rightarrow f(t)$ is asymptotically *exponential* for $t > 0$ and $m \rightarrow \infty$ (Error points have almost constant gaps on the logarithmic error scale).

┘

§3.6.0.26 (Scaling quadrature on Talbot contours) We try to understand the observations made in Ex. 3.6.0.24 and let these insights inspire algorithm development. As in Ex. 3.6.0.24 we consider $F(s) = s^{-1/2}$, which means $f(t) = (\pi t)^{-1/2}$.

Write γ^* for a Talbot contour, for which we assume t -uniform exponential convergence $\tilde{f}_{m,\mu^*}(t) \rightarrow f(t)$ for $m \rightarrow \infty$ on $[1, B]$, $B > 1$. More precisely, we assume an exponentially decaying relative quadrature error

$$\exists C > 0, q \in [0, 1[: \quad \left| \tilde{f}_{m,\mu^*}(t) - f(t) \right| \leq Cq^m |f(t)| \quad \forall t \in [1, B]. \quad (3.6.0.27)$$

Now pick $t \in [\xi, B\xi]$ for some $\xi > 0$. We apply the quadrature approximation (3.6.0.25) on a Talbot contour with stretching parameter $\mu := \mu^*/\xi$. Noting that $\gamma_T(\mu; \theta) = \xi^{-1}\gamma_T(\mu^*; \theta)$ we get

$$\begin{aligned} \tilde{f}_{m,\mu}(t) &= \frac{1}{4\pi i m} \sum_{j=-m+1}^{m-1} \frac{1}{\sqrt{\gamma_T(\mu; \theta_j)}} \exp(\gamma_T(\mu; \theta_j)t) \frac{d\gamma_T}{d\theta}(\mu; \theta_j) \\ &= \frac{1}{4\pi i m} \sum_{j=-m+1}^{m-1} \frac{\sqrt{\xi}}{\sqrt{\gamma_T(\mu^*; \theta_j)}} \exp(\gamma_T(\mu^*; \theta_j)t/\xi) \frac{1}{\xi} \frac{d\gamma_T}{d\theta}(\mu^*; \theta_j) \\ &= f(t) \sqrt{\pi t/\xi} \tilde{f}_{m,\mu^*}(\hat{t}), \quad \hat{t} := \frac{t}{\xi} \in [1, B]. \end{aligned} \quad (3.6.0.28)$$

Next, use (3.6.0.27) in the form

$$\tilde{f}_{m,\mu^*}(\hat{t}) = f(\hat{t})(1 + \delta_m) \quad \text{with} \quad |\delta_m| \leq Cq^m,$$

which yields $f(t) = 1/\sqrt{\pi t}$

$$\tilde{f}_{m,\mu}(t) = f(t) \sqrt{\pi t/\xi} f(\hat{t})(1 + \delta_m) = f(t)(1 + \delta_m) \quad \forall t \in [\xi, B\xi]. \quad (3.6.0.29)$$

Summing up, we get the convergence expressed by (3.6.0.27) for *any dilated t-interval* $[\xi, B\xi]$, $\xi > 0$, provided that we use a Talbot contour with the right stretching parameter $\mu := \mu^* / \xi!$

§3.6.0.30 (Local trapezoidal rule quadrature) The above consideration have been pursued for the transfer function $F(s) := s^{-1/2}$, but they apply to virtually all 0-sectorial functions and suggest a policy for doing t -uniformly accurate quadrature approximation of (3.6.0.20).



Idea: Achieve t -uniform convergence of the quadrature approximation (3.6.0.25) on $[\tau, M\tau]$, $\tau > 0$, $M \in \mathbb{N}$, $M \gg 1$, by applying it *locally* with suitable stretching parameters μ on a partition of $[\tau, T]$ into intervals $[\xi, B\xi[$ for some $B > 1$.

Of course, the partition should involve as few intervals as possible, which suggests letting their lengths grow geometrically:

$$[\tau, M\tau] \subset \bigcup_{k=1}^K I_k, \quad I_k := [B^{k-1}\tau, B^k\tau[, \quad K \in \mathbb{N} \text{ minimal: } B^K \geq M. \quad (3.6.0.31)$$

Write μ_k for the Talbot contour stretching parameter suitable for the t -interval I_k and Γ_k for the corresponding Talbot contour: $\Gamma_k = \gamma_T(\mu_k;]-\pi, \pi[)$. We also abbreviate the quadrature points/quadrature weights for the $2m + 1$ -point trapezoidal rule on Γ_k

$$\begin{aligned} s_j^{(k)} &:= \gamma_T(\mu_k, \theta_j), \\ \omega_j^{(k)} &:= \frac{1}{4\pi i m} \frac{d\gamma_T}{d\theta}(\mu_k; \theta_j), \end{aligned} \quad \theta_j := \frac{\pi j}{m}, \quad j = -m + 1, \dots, m - 1. \quad (3.6.0.32)$$

This leads to t -uniform exponential sum approximation of $f(t)$ on $[\tau, M\tau]$:

$$f(t) \approx \tilde{f}_{m, \mu_k}(t) := \sum_{j=-m+1}^{m-1} F(s_j^{(k)}) \exp(s_j^{(k)} t) \omega_j^{(k)} \quad \text{for } t \in I_k. \quad (3.6.0.33)$$

EXPERIMENT 3.6.0.34 (Quadrature error of t -local trapezoidal rule quadrature) We study the quadrature error of the t -local trapezoidal rule quadrature approximation (3.6.0.33) for $F(s) = s^{-1/2}$ on $[\tau, M\tau]$ with $\tau = 10^{-3}$, $M = 2000$.

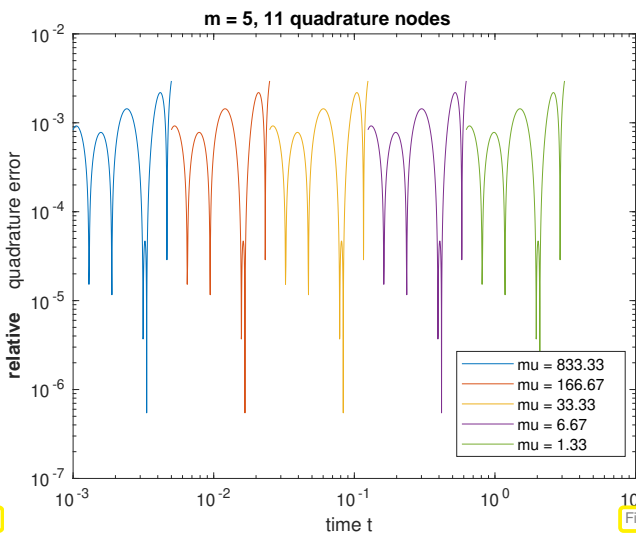


Fig. 148

$m = 5, B = 5$

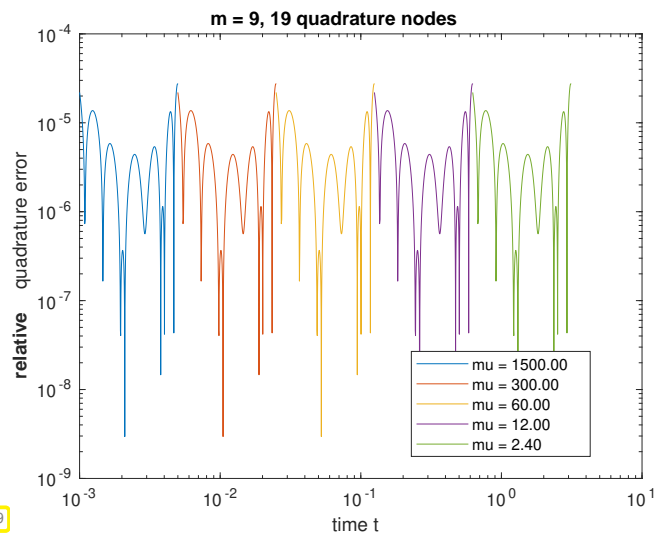


Fig. 149

$m = 9, B = 5$

As expected from the preceding analysis we see an identical t -dependence of the relative quadrature error in the intervals $[B^{k-1}\tau, B^k\tau], k \in \mathbb{N}$.

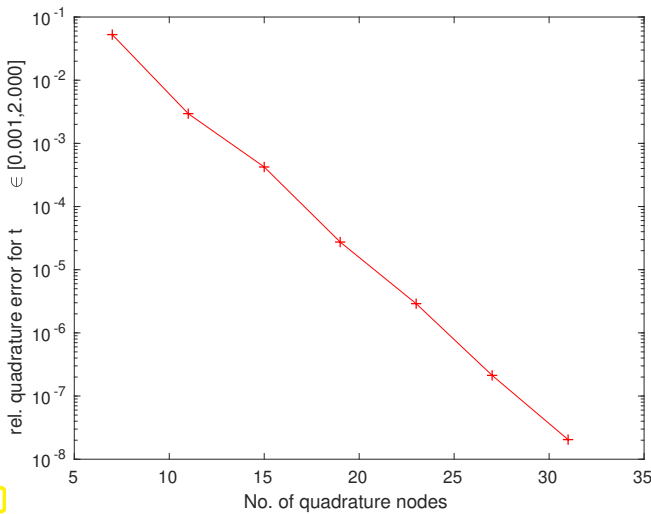


Fig. 150

◁ Quadrature error as a function of the number of quadrature nodes, maximum over $t \in \{0.001, 2\}$
 We observe empiric t -uniform exponential convergence for increasing number of quadrature nodes.

§3.6.0.35 (Time-local exponential sum approximation for convolution) Given a causal $g : \mathbb{R} \rightarrow \mathbb{C}$, a 0-sectorial transfer function $F : \mathbb{C} \setminus \mathbb{R}_0^- \rightarrow \mathbb{C}$, and a timestep $\tau > 0, M \in \mathbb{N}, M \gg 1$, we want to compute approximately

$$x_n := (F(\partial_t)g)(\tau n) = \int_0^{n\tau} f(n\tau - \xi)g(\xi) d\xi, \quad n = 1, \dots, M,$$



Idea: In the convolution integral use t -dependent quadrature formulas to obtain t -local exponential sum approximations of $f(t)$.

Given a “interval growth factor” $B > 0$ we write

$$a_k := B^k\tau, \quad k \in \mathbb{N}_0, \quad I_k := [a_{k-1}, a_k], \quad k \in \mathbb{N}.$$

Harking back to § 3.6.0.30 note that this defines a non-overlapping covering of $[0, M\tau]$,

$$[0, M\tau] = [0, \tau[\cup I_1 \cup I_2 \cup \dots \cup I_L, \quad L := \left\lceil \frac{\log M - \log \tau}{\log B} \right\rceil. \quad (3.6.0.36)$$

We write the convolution integral as a sum of integrals, for each of which $t - \xi$ belongs to a single interval of the partition (3.6.0.36):

$$\int_0^t f(t - \xi)g(\xi) d\xi = \int_{t-\tau}^t f(t - \xi)g(\xi) d\xi + \underbrace{\sum_{k=1}^{K(t)} \int_{t-a_k}^{t-a_{k-1}} f(t - \xi)g(\xi) d\xi}_{t-\xi \in I_k!}, \quad K(t) = \left\lceil \frac{\log t - \log \tau}{\log B} \right\rceil$$

When $t - \xi \in I_k, k \in \mathbb{N}$, we use the $2m + 1$ -point quadrature approximation (3.6.0.33):

$$\begin{aligned} \int_{t-a_k}^{t-a_{k-1}} f(t - \xi)g(\xi) d\xi &\approx \int_{t-a_k}^{t-a_{k-1}} \sum_{j=-m+1}^{m-1} \omega_j^{(k)} F(s_j^{(k)}) \exp(s_j^{(k)}(t - \xi)) g(\xi) d\xi \\ &= \sum_{j=-m+1}^{m-1} \omega_j^{(k)} F(s_j^{(k)}) \exp(a_{k-1}s_j^{(k)}) \int_{t-a_k}^{t-a_{k-1}} \exp((t - a_{k-1}) - \xi)g(\xi) d\xi. \end{aligned}$$

Appealing to the **variation-of-constants formula** from Lemma 3.3.2.2, as in § 3.6.0.8 we can replace the integral with a solution of an IVP for a linear ODE. For $a \in \mathbb{R}$ and $s \in \mathbb{C}$ write

$$t \mapsto y(t, a; s) \text{ for the solution of } \begin{cases} \dot{y} = sy + g(t), \\ y(\max\{0, a\}) = 0. \end{cases} \quad (3.6.0.37)$$

This permits us to express

$$\int_{t-a_k}^{t-a_{k-1}} f(t-\xi)g(\xi) d\xi \approx \sum_{j=-m+1}^{m-1} \omega_j^{(k)} F(s_j^{(k)}) \exp(a_{k-1}s_j^{(k)}) y(t-a_{k-1}, t-a_k; s_j^{(k)}). \quad (3.6.0.38)$$

Combining these approximations, we arrive at a formula that relies on solutions of $\lceil \log_B \frac{t}{\tau} \rceil$ linear ordinary differential equations.

$$\int_0^t f(t-\xi)g(\xi) d\xi \approx \int_{t-\tau}^t f(t-\xi)g(\xi) d\xi + \sum_{k=1}^{K(t)} \sum_{j=-m+1}^{m-1} \omega_j^{(k)} F(s_j^{(k)}) \exp(a_{k-1}s_j^{(k)}) y(t-a_{k-1}, t-a_k; s_j^{(k)}). \quad (3.6.0.39)$$

┘

§3.6.0.40 (Approximation of “near past” convolution [HS03, Eq. (29)]) To deal with the first summand on the right-hand side of (3.6.0.39) for $\tau \ll 1$ we replace $t \mapsto g(t)$ on $[t-\tau, t]$ with its linear interpolant $\xi \mapsto (g(t-\tau) - g(t))\frac{t-\xi}{\tau} + g(t)$.

$$\int_{t-\tau}^t f(t-\xi)g(\xi) d\xi \approx \int_0^\tau f(\xi)(c_1(\tau-\xi) + c_0) d\xi, \quad c_1 := \frac{g(t) - g(t-\tau)}{\tau}, \quad c_0 := g(t-\tau).$$

By the fundamental theorem of calculus and integration by parts,

$$\int_0^\tau f(\xi) d\xi = \pi_1(\tau) \text{ for causal } \pi_1, \quad \pi_1' = f,$$

$$\int_0^\tau f(\xi)(\tau-\xi) d\xi = \int_0^\tau \pi_1(\xi) d\xi = \pi_2(\tau) \text{ for causal } \pi_2, \quad \pi_2'' = f.$$

The (anti-)differentiation formula for the Laplace transform from Thm. 3.1.3.22 gives

$$(\mathcal{L}\pi_1)(s) = \frac{F(s)}{s}, \quad (\mathcal{L}\pi_2)(s) = \frac{F(s)}{s^2}. \quad (3.6.0.41)$$

Thus, $\pi_1(\tau)$ and $\pi_2(\tau)$, which have 0-sectorial Laplace transforms, can be computed by inverting Laplace transforms and we can approximate them by quadrature on the Talbot contour Γ_1 (meant for the interval $[\tau, B\tau]$):

$$\pi_\nu(\tau) = \frac{1}{2\pi i} \int_{\Gamma_1} \frac{F(s)}{s^\nu} e^{s\tau} ds \approx \sum_{j=-m+1}^{m-1} \omega_j^{(1)} \frac{F(s_j^{(1)})}{(s_j^{(1)})^\nu} \exp(s_j^{(1)}\tau), \quad \nu = 1, 2. \quad (3.6.0.42)$$

These numbers need only be computed once in the beginning in order to be able to evaluate

$$\int_{t-\tau}^t f(t-\zeta)g(\zeta) d\zeta \approx \Phi_1 g(t-\tau) + \Phi_2 \frac{g(t) - g(t-\tau)}{\tau}, \quad \Phi_1 := \pi_1(\tau), \Phi_2 := \pi_2(\tau), \quad (3.6.0.43)$$

for any $t \geq \tau$ ┘

§3.6.0.44 (Sampling (3.6.0.39) [LS02, Sect. 2.4]) Remember that we want to approximate $x_n := (F(\partial_t)g)(\tau n)$ for $n = 1, \dots, M, M \gg 1$, and “timestep” $\tau > 0$. A key concern will be the *reuse of information* contained in the solutions $t \mapsto y(t, \cdot, s_j^{(k)})$ of IVPs (3.6.0.37) occurring in (3.6.0.39).

For the sake of simplicity we will only⁴ consider the case $B = 2$, which means that different Talbot contours are used for t in the intervals

$$I_k = [2^{k-1}\tau, 2^k\tau], \quad k \in \mathbb{N}: \quad I_1 = [\tau, 2\tau], \quad I_2 = [2\tau, 4\tau], \quad I_3 = [4\tau, 8\tau] \quad \dots$$

Below we use the following shorthand notation to indicate the splitting of intervals over which we integrate. For instance,

$$\int_0^{15\tau} f(15\tau - \zeta)g(\zeta) d\zeta = \int_{14\tau}^{15\tau} \dots d\zeta + \underbrace{\int_{12\tau}^{14\tau} \dots d\zeta}_{t-\zeta \in I_1} + \underbrace{\int_{8\tau}^{12\tau} \dots d\zeta}_{t-\zeta \in I_2} + \underbrace{\int_0^{8\tau} \dots d\zeta}_{t-\zeta \in I_3}$$

is written as

$$[0, 15\tau] = [14\tau, 15\tau] \cup [12\tau, 14\tau] \cup [8\tau, 12\tau] \cup [0, 8\tau].$$

The colors correspond to the colors used in the following visualization of the different decompositions employed for the cases $t = n\tau, n = 1, \dots, 15$, cf. [LS02, Fig. 2.5].

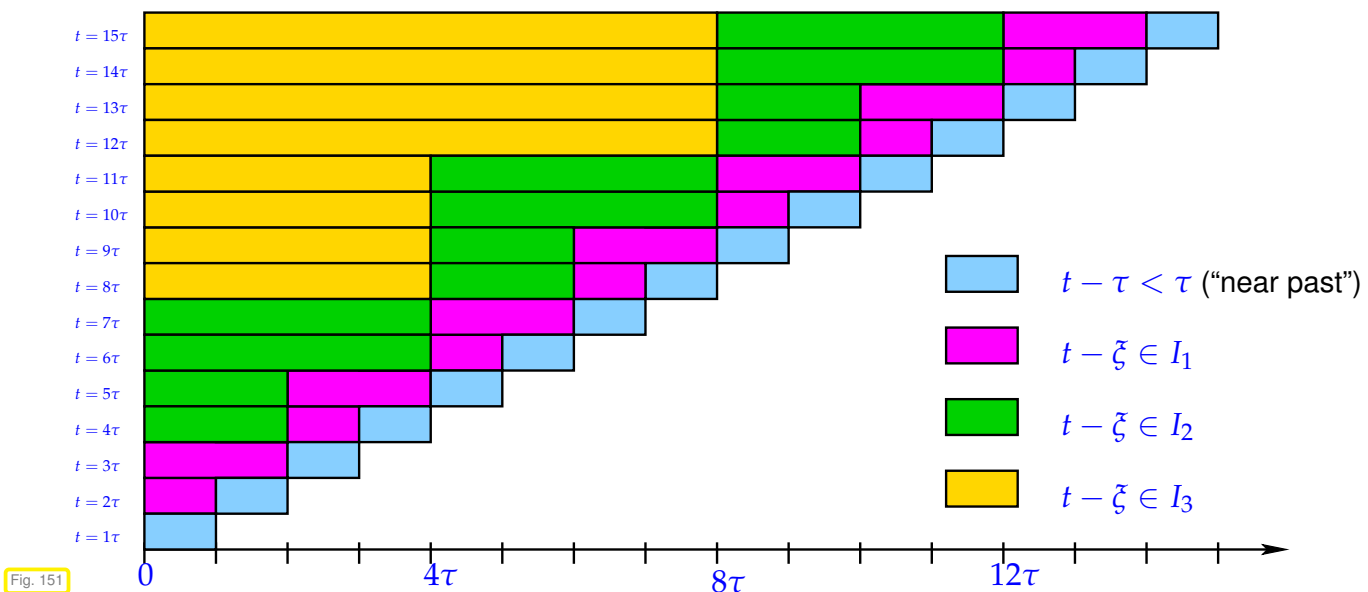


Fig. 151

► $n = 1$: Use the formula (3.6.0.43), $g_j := g(j\tau)$,

$$x_1 \approx \Phi_1 g_0 + \Phi_2 \frac{g_1 - g_0}{\tau}. \quad (3.6.0.45)$$

⁴For general B the algorithm is presented in [LS02, Sect. 2.5] and [HS03, Sect. 4]

► $n = 2$: Split $[0, 2\tau] = [\tau, 2\tau] \cup [0, \tau]$.

$$x_2 \approx \Phi_1 g_1 + \Phi_2 \frac{g_2 - g_1}{\tau} + \sum_{j=-m+1}^{m-1} \omega_j^{(1)} F(s_j^{(1)}) \exp(\tau s_j^{(1)}) y(\tau, 0; s_j^{(1)}). \quad (3.6.0.46)$$

► $n = 3$: Split $[0, 3\tau] = [2\tau, 3\tau] \cup [0, 2\tau]$.

$$x_3 \approx \Phi_1 g_2 + \Phi_2 \frac{g_3 - g_2}{\tau} + \sum_{j=-m+1}^{m-1} \omega_j^{(1)} F(s_j^{(1)}) \exp(\tau s_j^{(1)}) y(2\tau, 0; s_j^{(1)}). \quad (3.6.0.47)$$

► $n = 4$: Split $[0, 4\tau] = [3\tau, 4\tau] \cup [2\tau, 3\tau] \cup [0, 2\tau]$.

$$x_4 \approx \Phi_1 g_2 + \Phi_2 \frac{g_3 - g_2}{\tau} + \sum_{j=-m+1}^{m-1} \omega_j^{(1)} F(s_j^{(1)}) \exp(\tau s_j^{(1)}) y(3\tau, 2\tau; s_j^{(1)}) + \sum_{j=-m+1}^{m-1} \omega_j^{(2)} F(s_j^{(2)}) \exp(2\tau s_j^{(2)}) y(2\tau, 0; s_j^{(2)}). \quad (3.6.0.48)$$

► $n = 5$: Split $[0, 5\tau] = [4\tau, 5\tau] \cup [2\tau, 4\tau] \cup [0, 2\tau]$.

$$x_5 \approx \Phi_1 g_4 + \Phi_2 \frac{g_5 - g_4}{\tau} + \sum_{j=-m+1}^{m-1} \omega_j^{(1)} F(s_j^{(1)}) \exp(\tau s_j^{(1)}) y(4\tau, 2\tau; s_j^{(1)}) + \sum_{j=-m+1}^{m-1} \omega_j^{(2)} F(s_j^{(2)}) \exp(3\tau s_j^{(2)}) y(2\tau, 0; s_j^{(2)}). \quad (3.6.0.49)$$

► $n = 6$: Split $[0, 6\tau] = [5\tau, 6\tau] \cup [4\tau, 5\tau] \cup [0, 4\tau]$.

$$x_6 \approx \Phi_1 g_5 + \Phi_2 \frac{g_6 - g_5}{\tau} + \sum_{j=-m+1}^{m-1} \omega_j^{(1)} F(s_j^{(1)}) \exp(\tau s_j^{(1)}) y(5\tau, 4\tau; s_j^{(1)}) + \sum_{j=-m+1}^{m-1} \omega_j^{(2)} F(s_j^{(2)}) \exp(2\tau s_j^{(2)}) y(4\tau, 0; s_j^{(2)}). \quad (3.6.0.50)$$

► $n = 7$: Split $[0, 7\tau] = [6\tau, 7\tau] \cup [4\tau, 6\tau] \cup [0, 4\tau]$.

$$x_7 \approx \Phi_1 g_6 + \Phi_2 \frac{g_7 - g_6}{\tau} + \sum_{j=-m+1}^{m-1} \omega_j^{(1)} F(s_j^{(1)}) \exp(\tau s_j^{(1)}) y(6\tau, 4\tau; s_j^{(1)}) + \sum_{j=-m+1}^{m-1} \omega_j^{(2)} F(s_j^{(2)}) \exp(3\tau s_j^{(2)}) y(4\tau, 0; s_j^{(2)}). \quad (3.6.0.51)$$

► $n = 8$: Split $[0, 8\tau] = [7\tau, 8\tau] \cup [6\tau, 7\tau] \cup [4\tau, 6\tau] \cup [0, 4\tau]$.

$$x_8 \approx \Phi_1 g_7 + \Phi_2 \frac{g_8 - g_7}{\tau} + \sum_{j=-m+1}^{m-1} \omega_j^{(1)} F(s_j^{(1)}) \exp(\tau s_j^{(1)}) y(7\tau, 6\tau; s_j^{(1)}) + \sum_{j=-m+1}^{m-1} \omega_j^{(2)} F(s_j^{(2)}) \exp(2\tau s_j^{(2)}) y(6\tau, 4\tau; s_j^{(2)}) + \sum_{j=-m+1}^{m-1} \omega_j^{(3)} F(s_j^{(3)}) \exp(4\tau s_j^{(3)}) y(4\tau, 0; s_j^{(3)}). \quad (3.6.0.52)$$

► $n = 9$: Split $[0, 9\tau] = [8\tau, 9\tau] \cup [6\tau, 8\tau] \cup [4\tau, 6\tau] \cup [0, 4\tau]$.

$$x_9 \approx \Phi_1 g_8 + \Phi_2 \frac{g_9 - g_8}{\tau} + \sum_{j=-m+1}^{m-1} \omega_j^{(1)} F(s_j^{(1)}) \exp(\tau s_j^{(1)}) y(8\tau, 6\tau; s_j^{(1)}) + \quad (3.6.0.53)$$

$$\sum_{j=-m+1}^{m-1} \omega_j^{(2)} F(s_j^{(2)}) \exp(3\tau s_j^{(2)}) y(6\tau, 4\tau; s_j^{(2)}) +$$

$$\sum_{j=-m+1}^{m-1} \omega_j^{(3)} F(s_j^{(3)}) \exp(5\tau s_j^{(3)}) y(4\tau, 0; s_j^{(3)}).$$

► $n = 10$: Split $[0, 10\tau] = [9\tau, 10\tau] \cup [8\tau, 9\tau] \cup [4\tau, 8\tau] \cup [0, 4\tau]$.

$$x_{10} \approx \Phi_1 g_9 + \Phi_2 \frac{g_{10} - g_9}{\tau} + \sum_{j=-m+1}^{m-1} \omega_j^{(1)} F(s_j^{(1)}) \exp(\tau s_j^{(1)}) y(9\tau, 8\tau; s_j^{(1)}) + \quad (3.6.0.54)$$

$$\sum_{j=-m+1}^{m-1} \omega_j^{(2)} F(s_j^{(2)}) \exp(2\tau s_j^{(2)}) y(8\tau, 4\tau; s_j^{(2)}) +$$

$$\sum_{j=-m+1}^{m-1} \omega_j^{(3)} F(s_j^{(3)}) \exp(6\tau s_j^{(3)}) y(4\tau, 0; s_j^{(3)}).$$

► $n = 11$: Split $[0, 11\tau] = [10\tau, 11\tau] \cup [8\tau, 10\tau] \cup [4\tau, 8\tau] \cup [0, 4\tau]$.

$$x_{11} \approx \Phi_1 g_{10} + \Phi_2 \frac{g_{11} - g_{10}}{\tau} + \sum_{j=-m+1}^{m-1} \omega_j^{(1)} F(s_j^{(1)}) \exp(\tau s_j^{(1)}) y(10\tau, 8\tau; s_j^{(1)}) + \quad (3.6.0.55)$$

$$\sum_{j=-m+1}^{m-1} \omega_j^{(2)} F(s_j^{(2)}) \exp(3\tau s_j^{(2)}) y(8\tau, 4\tau; s_j^{(2)}) +$$

$$\sum_{j=-m+1}^{m-1} \omega_j^{(3)} F(s_j^{(3)}) \exp(7\tau s_j^{(3)}) y(4\tau, 0; s_j^{(3)}).$$

► $n = 12$: Split $[0, 12\tau] = [11\tau, 12\tau] \cup [10\tau, 11\tau] \cup [8\tau, 10\tau] \cup [0, 8\tau]$.

$$x_{12} \approx \Phi_1 g_{11} + \Phi_2 \frac{g_{12} - g_{11}}{\tau} + \sum_{j=-m+1}^{m-1} \omega_j^{(1)} F(s_j^{(1)}) \exp(\tau s_j^{(1)}) y(11\tau, 10\tau; s_j^{(1)}) + \quad (3.6.0.56)$$

$$\sum_{j=-m+1}^{m-1} \omega_j^{(2)} F(s_j^{(2)}) \exp(2\tau s_j^{(2)}) y(10\tau, 8\tau; s_j^{(2)}) +$$

$$\sum_{j=-m+1}^{m-1} \omega_j^{(3)} F(s_j^{(3)}) \exp(6\tau s_j^{(3)}) y(8\tau, 0; s_j^{(3)}).$$

► $n = 13$: Split $[0, 13\tau] = [12\tau, 13\tau] \cup [10\tau, 12\tau] \cup [8\tau, 10\tau] \cup [0, 8\tau]$.

$$x_{13} \approx \Phi_1 g_{12} + \Phi_2 \frac{g_{13} - g_{12}}{\tau} + \sum_{j=-m+1}^{m-1} \omega_j^{(1)} F(s_j^{(1)}) \exp(\tau s_j^{(1)}) y(12\tau, 10\tau; s_j^{(1)}) + \quad (3.6.0.57)$$

$$\sum_{j=-m+1}^{m-1} \omega_j^{(2)} F(s_j^{(2)}) \exp(3\tau s_j^{(2)}) y(10\tau, 8\tau; s_j^{(2)}) +$$

$$\sum_{j=-m+1}^{m-1} \omega_j^{(3)} F(s_j^{(3)}) \exp(5\tau s_j^{(3)}) y(8\tau, 0; s_j^{(3)}) .$$

► $n = 14$: Split $[0, 14\tau] = [13\tau, 14\tau] \cup [12\tau, 13\tau] \cup [8\tau, 12\tau] \cup [0, 8\tau]$.

$$x_{14} \approx \Phi_1 g_{13} + \Phi_2 \frac{g_{14} - g_{13}}{\tau} + \sum_{j=-m+1}^{m-1} \omega_j^{(1)} F(s_j^{(1)}) \exp(\tau s_j^{(1)}) y(13\tau, 12\tau; s_j^{(1)}) + \quad (3.6.0.58)$$

$$\sum_{j=-m+1}^{m-1} \omega_j^{(2)} F(s_j^{(2)}) \exp(2\tau s_j^{(2)}) y(12\tau, 8\tau; s_j^{(2)}) +$$

$$\sum_{j=-m+1}^{m-1} \omega_j^{(3)} F(s_j^{(3)}) \exp(6\tau s_j^{(3)}) y(8\tau, 0; s_j^{(3)}) .$$

► $n = 15$: Split $[0, 15\tau] = [14\tau, 15\tau] \cup [12\tau, 14\tau] \cup [8\tau, 12\tau] \cup [0, 8\tau]$.

$$x_{13} \approx \Phi_1 g_{14} + \Phi_2 \frac{g_{15} - g_{14}}{\tau} + \sum_{j=-m+1}^{m-1} \omega_j^{(1)} F(s_j^{(1)}) \exp(\tau s_j^{(1)}) y(14\tau, 12\tau; s_j^{(1)}) + \quad (3.6.0.59)$$

$$\sum_{j=-m+1}^{m-1} \omega_j^{(2)} F(s_j^{(2)}) \exp(3\tau s_j^{(2)}) y(12\tau, 8\tau; s_j^{(2)}) +$$

$$\sum_{j=-m+1}^{m-1} \omega_j^{(3)} F(s_j^{(3)}) \exp(7\tau s_j^{(3)}) y(8\tau, 0; s_j^{(3)}) .$$

Also for $n = 1, \dots, 15$ let us tabulate the solutions of different initial-value problems of the type (3.6.0.37) used in the n -th step

$n = 1 :$			
$n = 2 :$	$y(\tau, 0; s_j^{(1)})$		
$n = 3 :$	<u>$y(2\tau, \tau; s_j^{(1)})$</u>		
$n = 4 :$	<u>$y(3\tau, 2\tau; s_j^{(1)})$</u>	$y(2\tau, 0; s_j^{(2)})$	
$n = 5 :$	<u>$y(4\tau, 2\tau; s_j^{(1)})$</u>	$y(2\tau, 0; s_j^{(2)})$	
$n = 6 :$	<u>$y(5\tau, 4\tau; s_j^{(1)})$</u>	$y(4\tau, 0; s_j^{(2)})$	
$n = 7 :$	<u>$y(6\tau, 4\tau; s_j^{(1)})$</u>	<u>$y(4\tau, 0; s_j^{(2)})$</u>	
$n = 8 :$	<u>$y(7\tau, 6\tau; s_j^{(1)})$</u>	<u>$y(6\tau, 4\tau; s_j^{(2)})$</u>	$y(4\tau, 0; s_j^{(3)})$
$n = 9 :$	<u>$y(8\tau, 6\tau; s_j^{(1)})$</u>	$y(6\tau, 4\tau; s_j^{(2)})$	$y(4\tau, 0; s_j^{(3)})$
$n = 10 :$	<u>$y(9\tau, 8\tau; s_j^{(1)})$</u>	$y(8\tau, 4\tau; s_j^{(2)})$	$y(4\tau, 0; s_j^{(3)})$
$n = 11 :$	<u>$y(10\tau, 8\tau; s_j^{(1)})$</u>	<u>$y(8\tau, 4\tau; s_j^{(2)})$</u>	$y(4\tau, 0; s_j^{(3)})$
$n = 12 :$	<u>$y(11\tau, 10\tau; s_j^{(1)})$</u>	$y(10\tau, 8\tau; s_j^{(2)})$	$y(8\tau, 0; s_j^{(3)})$
$n = 13 :$	<u>$y(12\tau, 10\tau; s_j^{(1)})$</u>	$y(10\tau, 8\tau; s_j^{(2)})$	$y(8\tau, 0; s_j^{(3)})$
$n = 14 :$	<u>$y(13\tau, 12\tau; s_j^{(1)})$</u>	$y(12\tau, 8\tau; s_j^{(2)})$	$y(8\tau, 0; s_j^{(3)})$
$n = 15 :$	<u>$y(14\tau, 12\tau; s_j^{(1)})$</u>	<u>$y(12\tau, 8\tau; s_j^{(2)})$</u>	<u>$y(8\tau, 0; s_j^{(3)})$</u>

An underlined table entry means that in the following step the initial condition will be set to a more advanced time (“reset”). ┘

§3.6.0.60 (Exponential numerical integration of (3.6.0.37) [HS03, Eq. (27)]) In the formulas above we need (approximate) solutions of initial-value problems ($s \in \mathbb{C}$)

$$\dot{y}(t) = sy(t) + g(t) \quad , \quad y(\ell\tau) = 0 \quad \text{for some } \ell \in \mathbb{N} \tag{3.6.0.61}$$

and for times $t = i\tau, i \in \mathbb{N}, i > \ell$.

Approximate $t \mapsto g(t)$ by its linear interpolant $g_\tau : \mathbb{R} \rightarrow \mathbb{R}$ on $\mathcal{G}_\tau := \{\tau i\}_{i \in \mathbb{Z}}$:



$$g_\tau(t) := g_i + (t - \tau i) \frac{g_{i+1} - g_i}{\tau} \quad \text{for } i\tau \leq t < (i+1)\tau, \quad i \in \mathbb{Z}. \tag{3.6.0.62}$$

where we set $g_i := g(i\tau), i \in \mathbb{Z}$.

The advantage of replacing $g \rightarrow g_\tau$ is that the initial value problem

$$\dot{y}(t) = sy(t) + g_\tau(t) \quad , \quad y(\ell\tau) = y_\ell \tag{3.6.0.63}$$

can be solved *analytically* using the extended variation of constants formula

$$y(t) = y_0 e^{st} + \int_{\ell\tau}^t e^{s(t-\xi)} g_\tau(\xi) d\xi, \quad t \in \mathbb{R}, \tag{3.6.0.64}$$

which we apply with $\ell = 0, g_0 := g(0), g_1 := g(\tau)$:

$$\begin{aligned} y_1 := y(\tau) &= e^{s\tau} y_0 + \int_0^\tau e^{s(\tau-\xi)} (g_0 + \frac{g_1 - g_0}{\tau} \xi) d\xi \\ &= e^{s\tau} y_0 + \frac{g_0}{s} (e^{s\tau} - 1) + \frac{g_1 - g_0}{\tau s^2} (e^{s\tau} - 1 - s\tau) \\ &= y_0 + \frac{\exp(s\tau) - 1}{s\tau} \left(s\tau y_0 + \tau g_0 + \frac{g_1 - g_0}{s} \right) - \frac{g_1 - g_0}{s}. \end{aligned} \tag{3.6.0.65}$$

Hence, the solution of (3.6.0.63) sampled on the temporal grid \mathcal{G}_τ can be computed according to

$$y_n = y_{n-1} + \frac{\exp(s\tau) - 1}{s\tau} \left(s\tau y_0 + \tau g_{n-1} + \frac{g_n - g_{n-1}}{s} \right) - \frac{g_n - g_{n-1}}{s}, \quad n \in \mathbb{Z}, \quad (3.6.0.66)$$

where we wrote $g_i := g(i\tau)$, $i \in \mathbb{Z}$. ┘

§3.6.0.67 (Fast & oblivious CQ: Algorithm) Now we abandon the scalar setting and return to the situation of Ex. 3.6.0.1, where $s \mapsto F(s)$ is *matrix-valued*, in particular, where $F(s) \in \mathbb{C}^{N,N}$, $s \in \mathbb{C} \setminus]-\infty, 0]$, is, formally, the inverse of a large sparse matrix, which we can never afford to compute explicitly. All we can do is to apply $F(s)$ to a vector $\in \mathbb{C}^N$.

Thus, we regard $F(s)$ as a **functor object**, which supplies an `eval()` member function realizing the product of $F(s) \in \mathbb{C}^{N,N}$ with a vector. In a sense, now F should be viewed as a *functor-valued* function of a complex argument!

The argument function g has to be \mathbb{C}^N -valued. Given a timestep size $\tau > 0$, the algorithm will only use the values $g_i := g(\tau i)$, $i \in \mathbb{N}_0$. If we know in advance the final time $T := M\tau$, $M \in \mathbb{N}$, then we may simply pass the sequence $(g_i)_{i=0}^M$.

Pseudocode 3.6.0.68: Fast & oblivious convolution quadrature, $B = 2$

```

1  Matrix focq(FUNCTOR  $F$ , integer  $m$ , real  $\tau$ , integer  $M$ , Sequence  $(g_i)_{i=0}^M$ ) {
2    Matrix  $X \in \mathbb{R}^{N,M}$ ;
3     $L := \lfloor \log_2 M \rfloor$ ;
4    Compute  $s_j^{(k)}$ ,  $\omega_j^{(k)}$ ,  $j = -m+1, \dots, m-1$ ,  $k = 1, \dots, L$  // See (3.6.0.32)
5    Evaluate  $F(j, k) := F(s_j^{(k)})$ ,  $j = -m+1, \dots, m-1$ ,  $k = 1, \dots, L$ ;
6    // Vectors keeping solutions states  $y(*, *, s_j^{(k)})$  for (3.6.0.37)
7    Vector  $y_j^{(k)} \in \mathbb{R}^N$ ,  $j = -m+1, \dots, m-1$ ,  $k = 1, \dots, L$ ;
8    Compute  $\Phi_1$ ,  $\Phi_2$ ; // (3.6.0.42), (3.6.0.43)
9    // Initial times for  $k$ -th family of solutions of (3.6.0.37)
10   integer  $b[k] = 0$ ,  $k = 1, \dots, L$ ;
11   // Main timestepping loop
12   for  $n = 1$  to  $M$  do {
13     // Near-past convolution
14      $(X)_{:,n} := \Phi_1 g_{n-1} + \Phi_2 \frac{g_n - g_{n-1}}{\tau}$ ;
15     // Far-past updates using different Talbot contours
16     for  $k = 1$  to  $L$  do {
17       if  $(n \geq 2^k)$  {
18         if  $(n \bmod 2^k = 0)$  {
19            $y_j^{(k)} := \mathbf{0}$ ,  $j = -m+1, \dots, m-1$ ; // "reset"
20         }
21         if  $(n \bmod 2^{k-1} = 0)$  {
22           for  $l = 1$  to  $2^{k-1}$  do {
23             for  $j = -m+1$  to  $m-1$  do {
24               // Timestepping according to (3.6.0.66)
25                $\gamma := \frac{g_{b[k]+1} - g_{b[k]}}{s_j^{(k)}}$ 

```



```

26 |          $\mathbf{y}_j^{(k)} += \frac{\exp(s_j^{(k)}\tau) - 1}{s_j^{(k)}\tau} (s_j^{(k)}\tau\mathbf{y}_j^{(k)} + \tau\mathbf{g}_{b[k]} + \gamma) - \gamma; //$ 
27 |     }
28 |     b[k] := b[k] + 1;
29 | }
30 | }
31 | // Add Talbot contour contribution as in (3.6.0.39)
32 | (X):,n +=  $\sum_{j=-m+1}^{m-1} \omega_j^{(k)} \exp((n - b[k])\tau s_j^{(k)}) \mathbf{F}(j, k).eval(\mathbf{y}_j^{(k)}); //$ 
33 | }
34 | }
35 | }
36 | return X;
37 | }

```

§3.6.0.69 (Fast & oblivious CQ: Cost) We determine the asymptotic computational effort and storage requirements of the function `focq()` from Code 3.6.0.68 for $M \rightarrow \infty$, where M stands for the number of timesteps. Some estimates will also involve the dimension N of the state space (“number of degrees of freedom”) and the number m of quadrature nodes.

- The amount of memory `focq()` needs to compute approximations for *all* $x_n := (F(\partial_t)\mathbf{g})(\tau n)$, $n = 1, \dots, M$, is

$$\text{storage}(\text{focq}()) = O(mN \log M) \quad \text{for } M \rightarrow \infty. \quad (3.6.0.70)$$

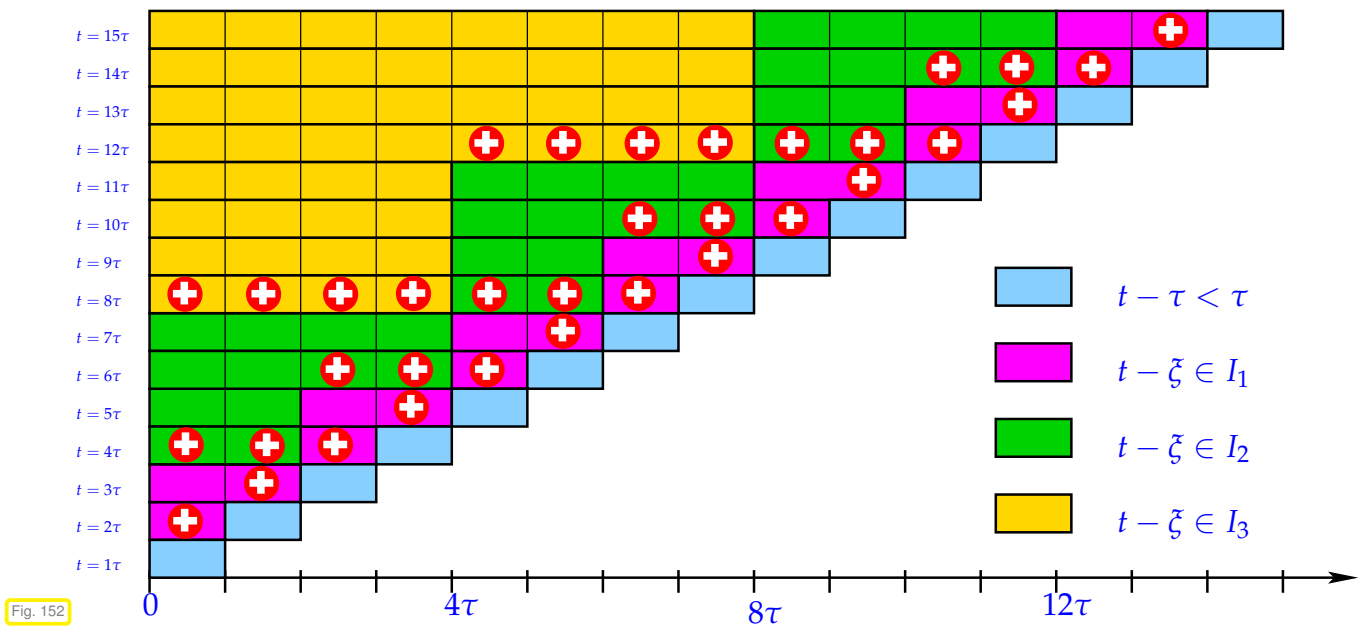
This is the combined size of all the auxiliary vectors $\mathbf{y}_j^{(k)} \in \mathbb{C}^N$, $j = -m + 1, \dots, m - 1$, $k = 1, \dots, L$, $L := \lfloor \log_2 M \rfloor$. In (3.6.0.70) we ignored the memory required for the result matrix \mathbf{X} , which just records the output states.

- Simple inspection of Code 3.6.0.68 and counting yields

$$\#\{F(\cdot)\text{-evaluations}\} = O(m \log M) \quad \text{for } M \rightarrow \infty, \quad (3.6.0.71)$$

$$\#\{F.eval(\cdot)\text{-invocations}\} = O(m M \log M) \quad \text{for } M \rightarrow \infty. \quad (3.6.0.72)$$

- The bulk of elementary matrix-vector operations in Code 3.6.0.68 is due to the update in Line 26. For $n = 15$ these updates are marked with \oplus in the following figure.



For every Talbot contour (index k) at most M update steps have to be carried out when executing $\text{focq}()$. This amounts to

$$\text{cost}(\text{vector operations in } \text{focq}()) = O(mNM \log M) \quad \text{for } M \rightarrow \infty. \quad (3.6.0.73)$$

This also includes the update operations in Line 32.

┘

Remark 3.6.0.74 (Fast & oblivious CQ: Properties & variants)

- Unlike the convolution quadrature schemes introduced in Section 3.4 and Section 3.5 the function $\text{focq}()$ from Code 3.6.0.68 provides an approximation of $F(\partial_t)$ by a discrete convolution, which fails to possess the crucial associativity property (3.3.3.1).
- The empiric *asymptotic convergence* of the approximation provided by $\text{focq}()$ in the maximum norm over all timesteps is $O(\tau^2 + q^m)$ for $\tau \rightarrow 0$, $m \rightarrow \infty$ and some $g \in [0, 1[$. The method converges algebraically with rate 2 in the timestep size τ and exponentially in the number m of quadrature points. Rigorous theoretical estimates are not available.
- The work [SLL65] devised an algorithm related to that of ??, which uses time-local quadrature on families on integration contours to obtain “fast & oblivious” approximations of all the CQ schemes presented in Section 3.4 and Section 3.5. The cost estimates of § 3.6.0.69 apply. Theses schemes enjoy associativity and allow a rigorous theoretical analysis [BS21, Sect. 8.2].

┘

Bibliography

- [BS21] L. Banjai and F.-J. Sayas. *Integral equation methods for evolutionary PDE*. Springer, 2021 (cit. on pp. 269, 328, 333, 356).
- [BHS80] Jon Louis Bentley, Dorothea Haken, and James B. Saxe. “A General Method for Solving Divide-and-conquer Recurrences”. In: *SIGACT News* 12.3 (Sept. 1980), pp. 36–44. DOI: [10.1145/1008861.1008865](https://doi.org/10.1145/1008861.1008865) (cit. on p. 290).
- [DB02] P. Deuffhard and F. Bornemann. *Scientific Computing with Ordinary Differential Equations*. 2nd ed. Vol. 42. Texts in Applied Mathematics. New York: Springer, 2002 (cit. on pp. 314, 318–321).
- [HS16] Matthew Hassell and Francisco-Javier Sayas. “Convolution quadrature for wave simulations”. In: *Numerical simulation in physics and engineering*. Vol. 9. SEMA SIMAI Springer Ser. Cham: Springer, 2016, pp. 71–159 (cit. on p. 269).
- [HS03] R. Hiptmair and A. Schädle. “Non-reflecting boundary conditions for Maxwell’s equations”. In: *Computing* 71.3 (2003), pp. 165–292 (cit. on pp. 348, 349, 353).
- [Lub88] C. Lubich. “Convolution quadrature and discretized operational calculus. II”. In: *Numer. Math.* 52.4 (1988), pp. 413–425 (cit. on p. 282).
- [LS02] Ch. Lubich and A. Schädle. “Fast convolution for non-reflecting boundary conditions”. In: *SIAM J. Sci. Comp.* 24 (2002), pp. 161–182 (cit. on pp. 269, 348, 349).
- [McL00] W. McLean. *Strongly Elliptic Systems and Boundary Integral Equations*. Cambridge, UK: Cambridge University Press, 2000 (cit. on p. 270).
- [Rem84] R. Remmert. *Funktionentheorie I*. Grundlehren Mathematik 5. Berlin: Springer, 1984 (cit. on pp. 305, 306).
- [Rie03] Andreas Rieder. *Keine Probleme mit inversen Problemen*. Braunschweig: Friedr. Vieweg & Sohn, 2003, pp. xiv+300. DOI: [10.1007/978-3-322-80234-7](https://doi.org/10.1007/978-3-322-80234-7) (cit. on p. 291).
- [Rud73] W. Rudin. *Functional Analysis*. 1st. McGraw–Hill, 1973 (cit. on p. 271).
- [Say16] Francisco-Javier Sayas. *Retarded potentials and time domain boundary integral equations*. Vol. 50. Springer Series in Computational Mathematics. Springer, [Cham], 2016, pp. xv+242 (cit. on pp. 269, 281, 283, 308).
- [SLL65] A. Schädle, M. Lopez-Fernandez, and C. Lubich. “Fast and oblivious convolution quadrature”. In: *SIAM J. Sci. Comp.* 28.2 (20065), p. 421 (cit. on pp. 269, 356).

Chapter 4

(Algebraic) Multigrid Methods



Supplementary literature. [TOS00] is a comprehensive textbook about and introduction into the foundations and algorithmic aspects of various kinds of multigrid methods:

- An outline of geometric multigrid is given in Chapter 2c “Basic Multigrid I”,
- Appendix A titled “Introduction to Algebraic Multigrid” is the text underlying parts of the presentation in Section 4.3.

4.1 Solvers for Finite Element Linear Systems

[NumPDE Chapter 2] introduced low-order finite element methods with small fixed polynomial degree of the local trial spaces for the approximate solution of second-order elliptic boundary value problems. However, the discussion completely glossed over a key issue: How can we solve the arising **large sparse linear systems** of equations fast?

Here, “large” hints at huge matrix dimensions that can go up to several billions as of 2023.

4.1.1 Elliptic Model Boundary Value Problems

The focus in this chapter is on scalar elliptic boundary value problems (BVPs) with homogeneous Dirichlet boundary conditions on bounded connected polyhedral domains [NumPDE Section 1.5] $\Omega \subset \mathbb{R}^d$, $d = 2, 3$:

$$-\operatorname{div}(\mathbf{A}(x) \operatorname{grad} u) + c(x)u = f \quad \text{in } \Omega, \quad u = 0 \quad \text{on } \partial\Omega. \quad (4.1.1.1)$$

The unknown is a function $u : \Omega \rightarrow \mathbb{R}$ and the source function f must be square integrable: $f \in L^2(\Omega)$.

Further, $\mathbf{A} : \Omega \rightarrow \mathbb{R}^{d,d}$ is a matrix-valued function, often called **diffusion coefficient**. We demand that

- $\mathbf{A} \in (C_{\text{pw}}^0(\Omega))^{d,d}$, that is, \mathbf{A} is piecewise continuous with respect to a subdomain partition of Ω ,
- $\mathbf{A}(x)$ is *symmetric* for all $x \in \Omega$, and
- \mathbf{A} is bounded and **uniformly positive definite** [NumPDE Def. 1.2.2.9]: there are constants $0 < \alpha^- \leq \alpha^+$ such that

$$\alpha^- \|z\|^2 \leq z^\top \mathbf{A}(x) z \leq \alpha^+ \|z\|^2 \quad \forall z \in \mathbb{R}^d, \quad \forall x \in \Omega. \quad (4.1.1.2)$$

The function $c : \Omega \rightarrow \mathbb{R}$ is called **reaction coefficient**, has to belong to $C_{pw}^0(\Omega)$ and to satisfy $c(x) \geq 0$ for all $x \in \Omega$.

EXAMPLE 4.1.1.3 (Poisson equation) In the special case $\mathbf{A}(x) = \mathbf{I}$ (identity matrix) and $c \equiv 0$, we face a homogeneous Dirichlet boundary value problem for the **Poisson equation**

$$-\Delta u = f \quad \text{in } \Omega, \quad u = 0 \quad \text{on } \partial\Omega. \tag{4.1.1.4}$$

┘

§4.1.1.5 (Two-point boundary value problems) A special case is $d = 1$, where $\Omega =]a, b[$, $a, b \in \mathbb{R}$, $a < b$. Then (4.1.1.1) reads:

$$\frac{d}{dx} \left(a(x) \frac{du}{dx}(x) \right) = f(x) \quad \text{for } x \in]a, b[\quad , \quad u(a) = u(b) = 0. \tag{4.1.1.6}$$

The Dirichlet boundary conditions reduce to prescribing the value of the solution u at two points, which is why (4.1.1.6) has been dubbed **two-point boundary-value problem** [NumPDE ??].

┘

§4.1.1.7 (Variational formulation) The finite element method relies on the variational formulation of (4.1.1.1), also known as the **weak form** [NumPDE Section 1.4]: seek $u \in H_0^1(\Omega)$

$$\underbrace{\int_{\Omega} \mathbf{A}(x) \mathbf{grad} u(x) \cdot \mathbf{grad} v(x) + c(x)u(x)v(x) \, dx}_{=:a(u,v)} = \underbrace{\int_{\Omega} f(x)v(x) \, dx}_{=:l(v)} \quad \forall v \in H_0^1(\Omega). \tag{4.1.1.8}$$

For the **Sobolev space** $H_0^1(\Omega)$ refer to [NumPDE Section 1.3.4]. Under the above assumptions on \mathbf{A} and c existence and uniqueness of solutions of (4.1.1.8) can be taken for granted.

┘

§4.1.1.9 (Equivalent minimization problem) As explained in [NumPDE Section 1.4] the linear variational problem (4.1.1.8) is equivalent to the **quadratic minimization problem**

$$u = \underset{v \in H_0^1(\Omega)}{\operatorname{argmin}} J(v) \quad , \quad J(v) := \frac{1}{2}a(v, v) - l(v), \tag{4.1.1.10}$$

with a and l the bilinear form and linear form as defined in (4.1.1.8).

┘

§4.1.1.11 (Finite element Galerkin discretization)

We equip Ω with a simplicial mesh/triangulation \mathcal{M} in the sense of [NumPDE Def. 2.5.1.1]. For $d = 1$ it will be a partitioning of the interval Ω into smaller intervals (cells), for $d = 2$ a special tiling of Ω with triangles.

A triangular mesh in 2D, edges drawn in **blue**, those on the boundary $\partial\Omega$ in **red**



We take for granted that the interior angles of all triangles are above a fixed threshold, which ensures a uniformly bounded shape-regularity measure [NumPDE Def. 3.3.2.20].

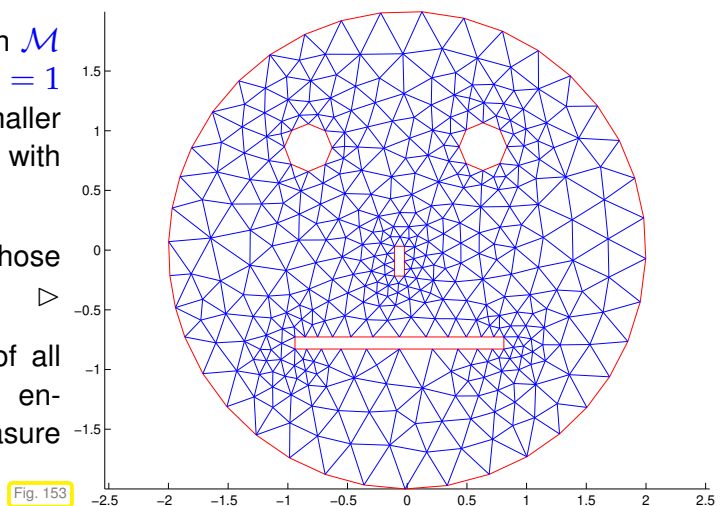


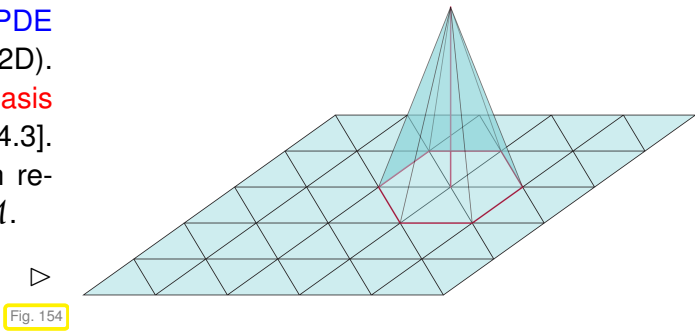
Fig. 153

The finite element method converts (4.1.1.8) into a discrete variational formulation by replacing $H_0^1(\Omega)$ by a finite-dimensional subspace V_h , a procedure called **Ritz-Galerkin discretization** [NumPDE Section 2.2]: seek $u_h \in V_h$

$$\underbrace{\int_{\Omega} \mathbf{A}(x) \mathbf{grad} u_h(x) \cdot \mathbf{grad} v_h(x) + c(x)u_h(x)v_h(x) dx}_{=:a(u_h, v_h)} = \underbrace{\int_{\Omega} f(x)v_h(x) dx}_{=:l(v_h)} \quad \forall v_h \in V_h. \quad (4.1.1.12)$$

We restrict ourselves to linear Lagrangian finite elements and use $V_h = \mathcal{S}_{1,0}^0(\mathcal{M})$, see [NumPDE § 2.3.1.4] (1D) and [NumPDE Section 2.4.2] (2D). We use “tent function” locally supported **nodal basis functions** as explained in [NumPDE Section 2.4.3]. They provide a cardinal basis of $\mathcal{S}_{1,0}^0(\mathcal{M})$ with respect to point evaluation at interior vertices of \mathcal{M} .

A single “tent function” on a triangular mesh (Graph is a pyramid with height 1.)



Inserting the nodal basis expansion of $u_h \in \mathcal{S}_{1,0}^0(\mathcal{M})$, the discrete variational problem can be converted into an equivalent linear system of equations $\mathbf{A}\vec{\mu} = \vec{\varphi}$, where $\mathbf{A} \in \mathbb{R}^{N,N}$ is the **Galerkin matrix**, $\vec{\mu} \in \mathbb{R}^N$ the vector of the basis expansion coefficients of $u_h \in V_h$, and $\vec{\varphi} \in \mathbb{R}^N$ the **load vector**. Throughout $N \in \mathbb{N}$ will stand for the dimension of the finite element space $N := \dim V_h$. It agrees with the number of interior nodes of \mathcal{M} .

The structure of the variational problem (4.1.1.8) implies particular properties of Galerkin matrices:

Lemma 4.1.1.13. Symmetric positive definite Galerkin matrices

Every matrix $\mathbf{A} \in \mathbb{R}^{N,N}$ arising from a Galerkin discretization of (4.1.1.8) based on the trial and test space $V_h \subset H_0^1(\Omega)$ will be **symmetric** and **positive definite**, that is

$$\mathbf{A} = \mathbf{A}^\top \quad \text{and} \quad \vec{v}^\top \mathbf{A} \vec{v} > 0 \quad \forall \vec{v} \in \mathbb{R}^N \setminus \{0\}. \quad (4.1.1.14)$$

┘

§4.1.1.15 (Finite element computations based on local quadrature rules) The occurrence in (4.1.1.8) of “general functions” $\mathbf{A} = \mathbf{A}(x)$, $c = c(x)$, and $f = f(x)$ that may be accessible through point evaluation only entails using numerical quadrature on the cells of the mesh in order to evaluate $a(u_h, v_h)$ and $l(v_h)$ approximately.

For $V_h = \mathcal{S}_{1,0}^0(\mathcal{M})$ it is sufficient to rely on the **composite trapezoidal rule**, *locally* defined by

$$\int_K \varphi(x) dx \approx \frac{1}{3}|K|(\varphi(\mathbf{a}^1) + \varphi(\mathbf{a}^2) + \varphi(\mathbf{a}^3)), \quad K \in \mathcal{M} \text{ triangle with vertices } \mathbf{a}^1, \mathbf{a}^2, \mathbf{a}^3, \quad (4.1.1.16)$$

for the approximation of all integrals:

$$\int_{\Omega} \varphi(x) dx \approx \sum_{K \in \mathcal{M}} \frac{1}{3}|K|(\varphi|_K(\mathbf{a}^1) + \varphi|_K(\mathbf{a}^2) + \varphi|_K(\mathbf{a}^3)) \quad (4.1.1.17)$$

§4.1.1.18 (Sparsity of finite element Galerkin matrices) The nodal basis functions b_h^1, \dots, b_h^N of $v_h = \mathcal{S}_{1,0}^0(\mathcal{M})$ are “tent functions” associated with the interior nodes/vertices x^1, \dots, x^N of the mesh \mathcal{M} . Since

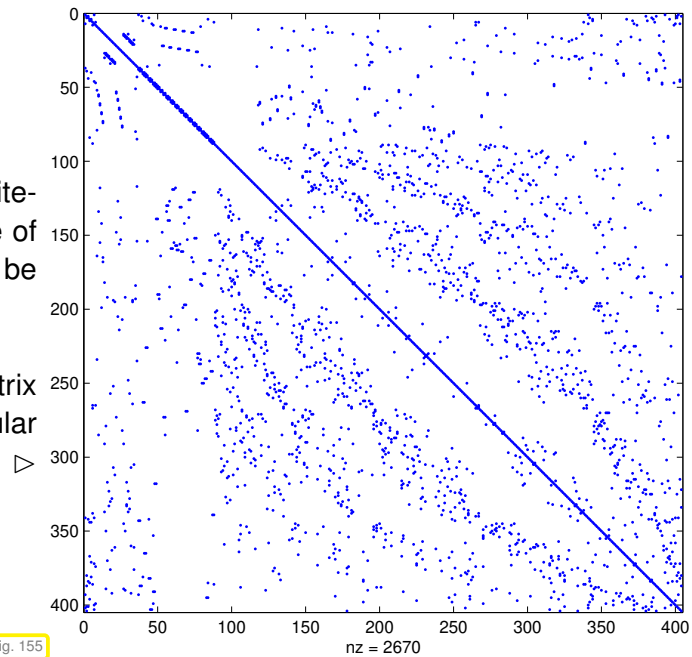
$$\text{supp}(b_h^i) = \bigcup \{ \bar{K} : K \in \mathcal{M}, x^i \in \bar{K} \}, \tag{4.1.1.19}$$

that is, the support of a basis function is the union of the (closed) triangles adjacent to the associated vertex, the $\mathcal{S}_{1,0}^0(\mathcal{M})$ -Galerkin matrix $\mathbf{A} \in \mathbb{R}^{N,N}$ for the bilinear form $a(\cdot, \cdot)$ from (4.1.1.8) satisfies:

$$\left\{ \begin{array}{l} \text{Nodes } x^i, x^j \in \mathcal{V}(\mathcal{M}) \\ \text{not connected by an edge} \end{array} \right\} \Leftrightarrow \text{Vol}(\text{supp}(b_h^i) \cap \text{supp}(b_h^j)) = 0 \Rightarrow (\mathbf{A})_{ij} = 0. \tag{4.1.1.20}$$

This means that for a “nice” mesh \mathcal{M} the finite-element Galerkin matrix \mathbf{A} is **sparse** in the sense of [NumPDE Notion 2.4.4.3]: most of its entries will be zero.

Non-zero entries of the $\mathcal{S}_{1,0}^0(\mathcal{M})$ -Galerkin matrix arising from discretizing (4.1.1.8) on the triangular mesh displayed in Fig. 153.



Here “nice” means that the **shape-regularity measure** $\rho_{\mathcal{M}}$ of \mathcal{M} according to [NumPDE Def. 3.3.2.20] is small.

Definition [NumPDE Def. 3.3.2.20]. Shape regularity measure

For a simplex $K \in \mathbb{R}^d$ we define its **shape regularity measure** as the ratio

$$\rho_K := h_K^d : |K|, \quad h_K := \text{diam}(K),$$

and the shape regularity measure of a simplicial mesh $\mathcal{M} = \{K\}$ as

$$\rho_{\mathcal{M}} := \max_{K \in \mathcal{M}} \rho_K.$$

As a consequence, the solid of cells of \mathcal{M} are bounded from below, which means that only a small number of cells (depending on $\rho_{\mathcal{M}}$) can abut a node. This limits the number of edges emanating from a node.

Sparsity of finite element Galerkin matrices

$N \times N$ Galerkin matrices for low-order finite element methods have $O(N)$ non-zero entries for $N \rightarrow \infty$.

As a consequence

- ◆ It takes only $O(N)$ memory to store an $N \times N$ finite element Galerkin matrix (\triangleright **data-sparse** matrices),
- ◆ the computational effort for the multiplication of an $N \times N$ finite element Galerkin matrix with a vector scales like $O(N)$ for $N \rightarrow \infty$.

┘

EXAMPLE 4.1.1.22 (Poisson matrix)

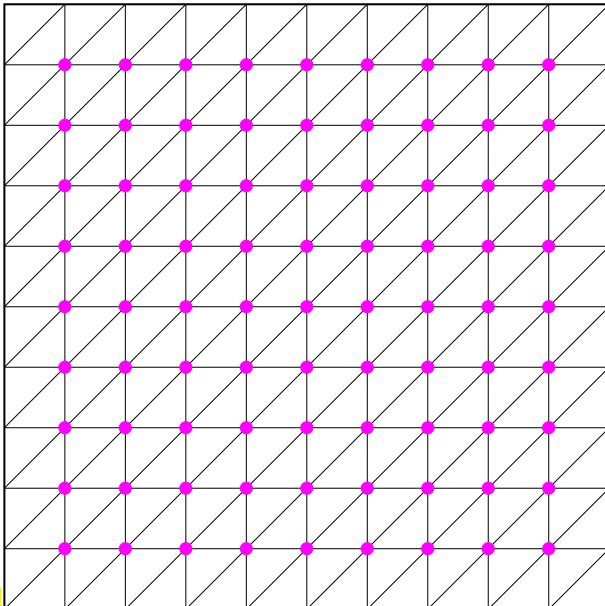


Fig. 156

As in [NumPDE § 4.1.3.3] we consider the finite element Galerkin discretization of the Poisson equation (4.1.1.4) on the unit square $\Omega =]0, 1[^2$ using linear finite elements on the “equidistant triangular tensor-product mesh” \mathcal{M} displayed beside.

Line-by-line lexicographic numbering of the interior nodes (\bullet) is assumed, cf. [NumPDE Section 4.1].

If there are $M + 1$ cells in each direction, the total number of interior nodes will be $N := M^2$, which agrees with $\dim \mathcal{S}_{1,0}^0(\mathcal{M})$.

As explained in [NumPDE § 4.1.2.3], we end up with an $N \times N$ block-tridiagonal Galerkin matrix, known as **Poisson matrix**

$$\mathbf{A} := \begin{pmatrix} \mathbf{T} & -\mathbf{I} & 0 & \cdots & \cdots & 0 \\ -\mathbf{I} & \mathbf{T} & -\mathbf{I} & & & \vdots \\ 0 & -\mathbf{I} & \mathbf{T} & -\mathbf{I} & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & 0 \\ \vdots & & & -\mathbf{I} & \mathbf{T} & -\mathbf{I} \\ 0 & \cdots & \cdots & 0 & -\mathbf{I} & \mathbf{T} \end{pmatrix}, \quad \mathbf{T} := \begin{pmatrix} 4 & -1 & 0 & & & 0 \\ -1 & 4 & -1 & & & \vdots \\ 0 & -1 & 4 & -1 & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & -1 & 4 & -1 \\ 0 & \cdots & \cdots & 0 & -1 & 4 \end{pmatrix} \in \mathbb{R}^{M,M} \quad (4.1.1.23)$$

We are going to rely on this matrix in several numerical experiments.

┘

EXAMPLE 4.1.1.24 (Bilinear FE for scalar elliptic Dirichlet BVPs) We start from the scalar elliptic boundary value problem with constant diffusion tensor and homogeneous Dirichlet boundary conditions:

$$-\operatorname{div} \left(\begin{bmatrix} \alpha & 0 \\ 0 & \beta \end{bmatrix} \operatorname{grad} u \right) = f \quad \text{in } \Omega :=]0, 1[^2, \quad u = 0 \quad \text{on } \partial\Omega, \quad (4.1.1.25)$$

with $\alpha, \beta > 0$, $f \in C^0(\overline{\Omega})$. Obviously, the diffusion tensor is symmetric and positive definite, which ensures the existence of a unique solution $u \in H^2(\Omega)$ [NumPDE Thm. 3.4.0.10].

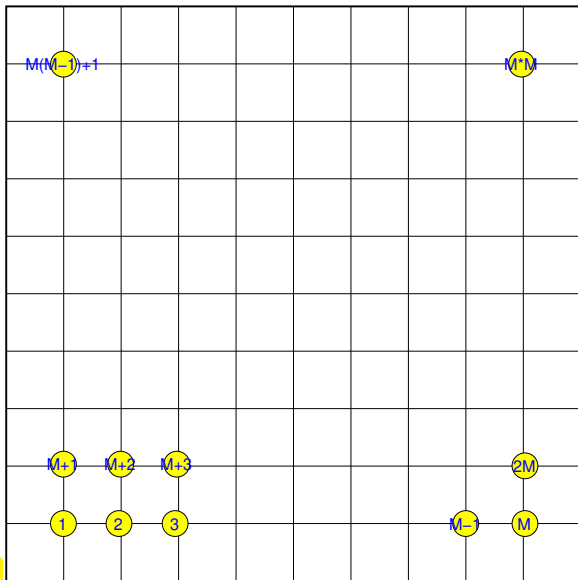


Fig. 157

▷ We use an equidistant quadrilateral tensor product mesh \mathcal{M} with $M \in \mathbb{N}$ interior nodes in each direction and meshwidth $h := \frac{1}{M+1}$. We number the M^2 interior mesh nodes lexicographically.

We rely on the piecewise bilinear Lagrangian finite element space $\mathcal{S}_{1,0}^0(\mathcal{M})$ introduced in [NumPDE Ex. 2.6.2.1] for the Galerkin finite-element discretization of the linear variational problem arising from (4.1.1.25): seek $u \in H_0^1(\Omega)$ such that

$$\int_{\Omega} \begin{bmatrix} \alpha & 0 \\ 0 & \beta \end{bmatrix} \mathbf{grad} u(x) \cdot \mathbf{grad} v(x) \, dx = \int_{\Omega} f(x)v(x) \, dx$$

for all $v \in H_0^1(\Omega)$.

The vertex-associated **local shape functions** on the reference element \hat{K} , the unit square, are, cf. [NumPDE Eq. (2.6.2.3)],

$$\begin{aligned} \hat{b}^1(x) &= (1 - x_1)(1 - x_2), \\ \hat{b}^2(x) &= x_1(1 - x_2), \\ \hat{b}^3(x) &= x_1x_2, \\ \hat{b}^4(x) &= (1 - x_1)x_2. \end{aligned} \tag{4.1.1.26}$$

▶ $\hat{b}^i(\mathbf{a}^j) = \delta_{ij}, \quad 1 \leq i, j \leq 4,$

that is, these basis functions satisfy the cardinal basis property with respect to the vertices of the unit square.

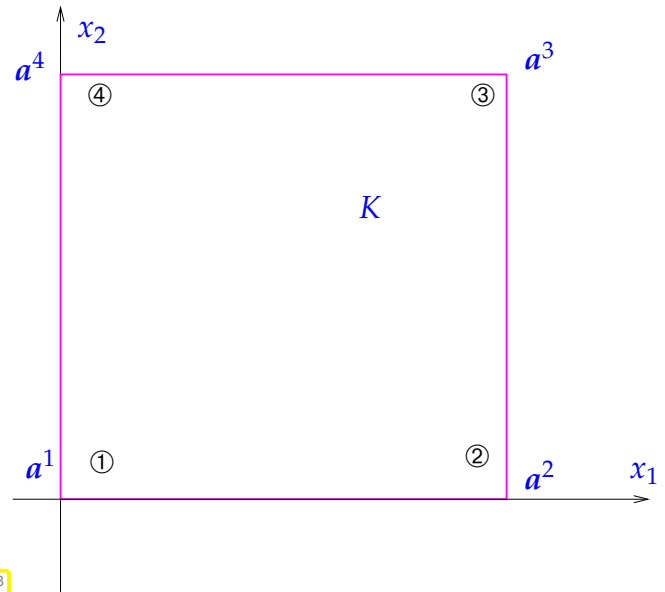


Fig. 158

Affine pullback then provides the local shape functions $b_K^i, i = 1, 2, 3, 4$, for any square K of the mesh \mathcal{M} . Using the vertex numbering from Fig. 158 plus the local trapezoidal quadrature rule

$$\int_K \varphi(x) \, dx \approx \frac{1}{4}|K| (\varphi(\mathbf{a}^1) + \varphi(\mathbf{a}^2) + \varphi(\mathbf{a}^3) + \varphi(\mathbf{a}^4)), \tag{4.1.1.27}$$

the element matrix \mathbf{A}_K has the entries

$$\mathbf{A}_K = \left[\int_K \mathbf{grad} b_K^j(x) \cdot \mathbf{grad} b_K^i(x) \, dx \right]_{i,j=1}^4 \approx \begin{bmatrix} 1/2(\alpha + \beta) & -1/2\alpha & 0 & -1/2\beta \\ -1/2\alpha & 1/2(\alpha + \beta) & -1/2\beta & 0 \\ 0 & -1/2\beta & 1/2(\alpha + \beta) & -1/2\alpha \\ -1/2\beta & 0 & -1/2\alpha & 1/2(\alpha + \beta) \end{bmatrix}. \tag{4.1.1.28}$$

Again using the local trapezoidal quadrature rule, the element load vector evaluates to

$$\varphi_K = \left[\int_{\Omega} f(x)b_K^i(x) \, dx \right]_{i=1}^4 \approx h^2 [f(\mathbf{a}^i)]_{i=1}^4. \tag{4.1.1.29}$$

Assembly then produces the $M^2 \times M^2$ linear system of equations

$$\mathbf{A}\vec{\mu} = \left[f(\mathbf{p}^i) \right]_{i=1}^{M^2}, \quad \{ \mathbf{p}^1, \mathbf{p}^2, \dots, \mathbf{p}^{M \cdot M} \} \triangleq \text{interior nodes of } \mathcal{M},$$

where the matrix \mathbf{A} has a structure similar to the one in (4.1.1.23):

$$\mathbf{A} := \begin{pmatrix} \mathbf{T} & -\beta\mathbf{I} & 0 & \cdots & \cdots & 0 \\ -\beta\mathbf{I} & \mathbf{T} & -\beta\mathbf{I} & & & \vdots \\ 0 & -\beta\mathbf{I} & \mathbf{T} & -\beta\mathbf{I} & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & 0 \\ \vdots & & & -\beta\mathbf{I} & \mathbf{T} & -\beta\mathbf{I} \\ 0 & \cdots & \cdots & 0 & -\beta\mathbf{I} & \mathbf{T} \end{pmatrix} \in \mathbb{R}^{M^2, M^2},$$

$$\mathbf{T} := \begin{pmatrix} 2(\alpha + \beta) & -\alpha & 0 & & & 0 \\ -\alpha & 2(\alpha + \beta) & -\alpha & & & \vdots \\ 0 & -\alpha & 2(\alpha + \beta) & -\alpha & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \\ \vdots & & & -\alpha & 2(\alpha + \beta) & -\alpha \\ 0 & \cdots & \cdots & 0 & -\alpha & 2(\alpha + \beta) \end{pmatrix} \in \mathbb{R}^{M, M}.$$
(4.1.1.30)

┘

§4.1.1.31 (Stencil notation [NumPDE § 4.1.2.10]) In Ex. 4.1.1.22 and Ex. 4.1.1.24 we saw rather special finite-element meshes that can be obtained by truncating an infinite *periodic lattice* to finite domain. Meshes of this special type are often called **(finite-difference) grids**. **Grid functions** are mappings from nodes/edges/cells of such grids into the real numbers.

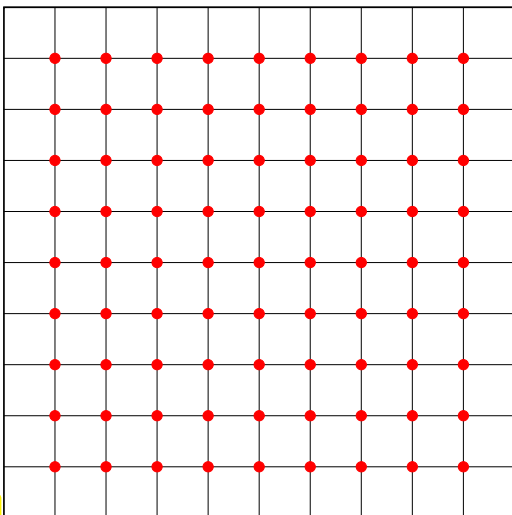


Fig. 159

Translation-invariant local linear operators acting on grid functions can be described by means of (finite-difference) stencils. Consider the $M \times M$ 2D tensor-product grid displayed beside with $M + 1$ cells in each direction and meshwidth $h := \frac{1}{M+1}$. Its interior nodes can be indexed by pairs (i, j) , $i, j \in \{1, \dots, M\}$, i being the “row index”. Thus a node-based grid function $\underline{\mu}$ corresponds to an $M \times M$ -matrix with entries $\mu_{i,j}$.

A translation-invariant local linear operator \mathbf{L} providing an endomorphism on the space of node-based grid functions can be written as

$$(\mathbf{L}\vec{\mu})_{i,j} = \sum_{\ell=-1}^1 \sum_{k=-1}^1 s_{\ell,k} \mu_{i+\ell, j+k}, \quad i, j \in \{1, \dots, M\}.$$
(4.1.1.32)

($\mu_{i,j} = 0$ assumed for “out-of-range” indices.)

The **(difference) stencil** notation for this operator \mathbf{L} is denoted as

$$\text{Stencil of } \mathbf{L} \text{ from (4.1.1.32)} = \begin{bmatrix} s_{1,-1} & s_{1,0} & s_{1,1} \\ s_{0,-1} & s_{0,0} & s_{0,1} \\ s_{-1,-1} & s_{-1,0} & s_{-1,1} \end{bmatrix}_h.$$
(4.1.1.33)

4.1.2 Sparse Elimination Solvers

Recall Gaussian elimination (GE) and its rewriting through the LU -decomposition of matrices, [NumCSE Section 2.3]. Gaussian elimination does not mesh smoothly with the sparse matrices obtained from finite element discretization:



[NumCSE Ex. 2.7.4.1]: LU -factors of a sparse matrix need not be sparse

fill-in [NumCSE Def. 2.7.4.3]

Let $\mathbf{A} \in \mathbb{R}^{N,N}$ be a large sparse finite element Galerkin matrix for a 2D or 3D BVP with “ $O(N)$ ” non-zero entries:

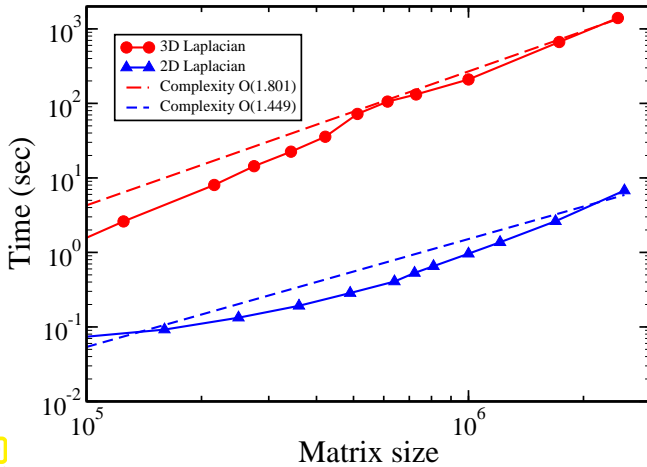
Dream: Cost for solving $\mathbf{A}\vec{\mu} = \vec{\varphi} = O(N)$

Reality Cost for solving $\mathbf{A}\vec{\mu} = \vec{\varphi} = O(N^\alpha)$, $1.5 \leq \alpha \leq 2$.

The exponent α depends on the details of the method and tends to be bigger for 3D problems. In practice one observes

- $\alpha \approx 1.5$ for **2D** finite element Galerkin matrices,
- $\alpha \approx 2$ for finite element Galerkin matrices arising from **3D** problems.

EXPERIMENT 4.1.2.1 (Cost of direct elimination solvers)



Runtime measurements for direct solution of FE linear systems, courtesy of Prof. O. Schenk, USI Lugano

- ◆ Sparse solver code **PARDISO 6.1** [SG06]
- ◆ Domain $\Omega =]0, 1[^d$, $\mathbf{A} \hat{=}$ Poisson-Galerkin matrix on uniform 2D/3D tensor product mesh (5-point/9-point stencil)
- ◆ OS: Ubuntu Linux 18.04, Compiler: gcc-7, -O3, single core, CPU: Intel Xeon CPU E7-4880@2.50GHz

Fig. 160

4.1.3 Stationary Linear Iterations (SLIs)

The $O(N^\alpha)$, $\alpha \approx 2$ asymptotic computational cost of direct elimination solvers becomes prohibitive for $N \approx 10^7$ even on HPC systems. Is there an alternative?

§4.1.3.1 (Iterative solution of linear systems of equations) As an alternative to the direct solution of $\mathbf{A}\vec{\mu} = \vec{\varphi}$, $\mathbf{A} \in \mathbb{R}^{N,N}$ sparse, we could try **iterative methods** that produce **sequences** $(\vec{\mu}^{(k)})_{k \in \mathbb{N}_0}$ of **approximate solutions** that, ideally, fast converge to the exact solution

$$\lim_{k \rightarrow \infty} \|\vec{\mu}^{(k)} - \vec{\mu}^*\| = 0, \quad \mathbf{A}\vec{\mu}^* = \vec{\varphi}.$$

Iterative methods may be preferred for several reasons often relevant in the context of finite element computations:

- (I) The sheer size of the linear system of equations rules out the use of methods whose memory requirements scale like $O(N^\alpha)$ as $N \rightarrow \infty$ for some $\alpha > 1$.
- (II) In light of inevitable discretization errors highly accurate solutions of the linear systems are not needed; early termination of the iteration may be possible.
- (III) If a rather good approximation of the solution is available already as initial guess, a sufficiently accurate solution may be obtained after only a few iterations.

┘

§4.1.3.2 (Gauss-Seidel method) The Gauss-Seidel method is an **iterative solution method** for general square linear systems of equations: Given

- ◆ the coefficient matrix $\mathbf{A} \in \mathbb{R}^{N,N}$, $N \in \mathbb{N}$, with non-zero diagonal elements, $(\mathbf{A})_{i,i} \neq 0$,
- ◆ any right-hand-side vector $\vec{\varphi} \in \mathbb{R}^N$,
- ◆ and an **initial guess** $\vec{\mu}_0 \in \mathbb{R}^N$,

it can be implemented as follows (the argument $\vec{\mu}$ both passes the initial guess and serves to return the approximate solution):

Pseudocode 4.1.3.3: Gauss-Seidel method for $\mathbf{A}\vec{\mu} = \vec{\varphi}$

```

1 void GaussSeidel(const  $\mathbf{A} \in \mathbb{R}^{N,N}$ , const  $\vec{\varphi} \in \mathbb{R}^N$ , ref  $\vec{\mu} \in \mathbb{R}^N$ , double TOL) {
2   do {
3     double deltanorm = 0; // squared norm of update in one step
4     // Update all components of the approximate solution
5     for (int i=0; i<N; i++) { \
6       double  $\delta\mu := \frac{1}{(\mathbf{A})_{i,i}} \left( (\vec{\varphi})_i - \sum_{j=1}^N (\mathbf{A})_{i,j} (\vec{\mu})_j \right)$ ;
7        $(\vec{\mu})_i += \delta\mu$ ;
8       deltanorm +=  $(\delta\mu)^2$ ;
9     } \
10  }
11  while (sqrt(deltanorm) < TOL *  $\|\vec{\mu}\|$ ); // Termination criterion
12 }
```

The outer loop in Code 4.1.3.3 embodies the steps of the Gauss-Seidel method. At step i of the inner loop (Line 5–Line 9) the solution component $(\vec{\mu})_i$ is adjusted so that the i -th row of the LSE $\mathbf{A}\vec{\mu} = \vec{\varphi}$ is satisfied exactly.

Obviously, the **computational effort** for a single step of the Gauss-Seidel method is proportional to the number of non-zero entries of \mathbf{A} , hence $O(N)$ for finite element Galerkin matrices and $N \rightarrow \infty$, remember § 4.1.1.18.

┘

§4.1.3.4 (Gauss-Seidel method as stationary linear iteration) The operations in the inner loop of Code 4.1.3.3 from Line 5 through Line 9 boil down to

$$(\mathbf{A})_{i,i}(\vec{\mu})_i := (\vec{\varphi})_i - \sum_{\substack{j=1 \\ j \neq i}}^N (\mathbf{A})_{i,j}(\vec{\mu})_j, \quad i = 1, \dots, N. \quad (4.1.3.5)$$

Thus, the entire inner loop of the Gauss-Seidel method from Code 4.1.3.3 can be rewritten as

$$\vec{\mu} \leftarrow \vec{\mu} + \text{tril}(\mathbf{A})^{-1} \vec{\rho} \quad \text{with residual } \vec{\rho} := \vec{\varphi} - \mathbf{A}\vec{\mu}, \quad (4.1.3.6)$$

where $\text{tril}(\mathbf{A}) \in \mathbb{R}^{N,N}$ extracts the lower-triangular part of the matrix \mathbf{A} . Assuming, $(\mathbf{A})_{i,i} \neq 0$ for all i ensures that $\text{tril}(\mathbf{A})$ is invertible.

Hence, the Gauss-Seidel method is an iteration generating the vector sequence $\vec{\mu}^{(0)}, \vec{\mu}^{(1)}, \vec{\mu}^{(2)}, \dots$ according to the rule

$$\vec{\mu}^{(0)} := \vec{\mu}_0, \quad \vec{\mu}^{(k+1)} = \vec{\mu}^{(k)} + \mathbf{M}(\vec{\varphi} - \mathbf{A}\vec{\mu}^{(k)}) \quad \text{with } \mathbf{M} := \text{tril}(\mathbf{A})^{-1}. \quad (4.1.3.7)$$

An iteration of the form $\vec{\mu}^{(k+1)} = \vec{\mu}^{(k)} + \mathbf{M}(\vec{\varphi} - \mathbf{A}\vec{\mu}^{(k)})$ is called a **stationary linear iteration** consistent with the linear system of equations $\mathbf{A}\vec{\mu} = \vec{\varphi}$.

Obviously, any solution of the LSE $\mathbf{A}\vec{\mu} = \vec{\varphi}$ provides a fixed point of the associated linear stationary iteration

$$\mathbf{A}\vec{\mu}^{(k)} = \vec{\varphi} \Rightarrow \vec{\mu}^{(k+1)} = \vec{\mu}^{(k)}. \quad (4.1.3.8)$$

This is the meaning of “*consistent*”. Moreover, every fixed point gives a solution of the LSE provided that \mathbf{M} is invertible

$$\vec{\mu} = \vec{\mu} + \mathbf{M}(\vec{\varphi} - \mathbf{A}\vec{\mu}) \xrightarrow{\mathbf{M} \text{ invertible}} \mathbf{A}\vec{\mu} = \vec{\varphi}. \quad (4.1.3.9)$$

┘

§4.1.3.10 (Jacobi method) The Jacobi method is a stationary linear iteration for the solution of $\mathbf{A}\vec{\mu} = \vec{\varphi}$ defined by

$$\vec{\mu}^{(k+1)} := \vec{\mu}^{(k)} + \omega \mathbf{D}^{-1}(\vec{\varphi} - \mathbf{A}\vec{\mu}^{(k)}), \quad \mathbf{D} := \text{diag}(\mathbf{A}), \quad (4.1.3.11)$$

that is, it is of the form (4.1.3.7) with \mathbf{M} replaced with the scaled diagonal part $\omega \text{diag}(\mathbf{A}) \in \mathbb{R}^{N,N}$ of \mathbf{A} , where $\omega > 0$ is a suitable **relaxation parameter**.

This method does not offer any advantages compared to the Gauss-Seidel method, except for being easier to analyze. ┘

§4.1.3.12 (Error recursion for stationary linear iterations) We consider a stationary linear iteration

$$\vec{\mu}^{(k+1)} = \vec{\mu}^{(k)} + \mathbf{M}(\vec{\varphi} - \mathbf{A}\vec{\mu}^{(k)}) \quad \text{with invertible } \mathbf{M} \in \mathbb{R}^{N,N}, \quad (4.1.3.13)$$

consistent with the $N \times N$ linear system of equations (LSE) $\mathbf{A}\vec{\mu} = \vec{\varphi}$, $\mathbf{A} \in \mathbb{R}^{N,N}$, $\vec{\varphi} \in \mathbb{R}^N$.

Assuming that \mathbf{A} is invertible, we write $\vec{\mu}^* \in \mathbb{R}^N$ for the unique solution of the LSE: $\mathbf{A}\vec{\mu}^* = \vec{\varphi}$. A one-line elementary calculation yields the **error recursion**

$$\vec{\epsilon}^{(k+1)} = (\mathbf{I} - \mathbf{M}\mathbf{A})\vec{\epsilon}^{(k)} \quad \text{for the } \text{iteration error } \vec{\epsilon}^{(k)} := \vec{\mu}^* - \vec{\mu}^{(k)}. \quad (4.1.3.14)$$

The matrix $\mathbf{E} := \mathbf{I} - \mathbf{M}\mathbf{A}$ is called the **error propagation matrix (EPM)** for the stationary linear iteration (4.1.3.13).

Corollary 4.1.3.15. Convergence of stationary linear iterations

Let $\|\cdot\|$ be a matrix norm induced by the vector norm $\|\cdot\|$ on \mathbb{R}^N . If $\rho := \|\mathbf{I} - \mathbf{M}\mathbf{A}\| < 1$, the stationary linear iteration (4.1.3.13) **converges** to $\vec{\mu}^* := \mathbf{A}^{-1}\vec{\varphi}$ **linearly** with rate ρ .

“Linear convergence” of an iteration is defined in [NumCSE Def. 8.2.2.1] and means that

$$\|\vec{\mu}^* - \vec{\mu}^{(k+1)}\| \leq \rho \|\vec{\mu}^* - \vec{\mu}^{(k)}\| \quad \text{for some } \rho < 1.$$

┘

Remark 4.1.3.16 (Asymptotic decay of iteration error) As in § 4.1.3.12 we consider the stationary linear iteration (4.1.3.13). From the error recursion (4.1.3.14) we learn that the sequence of error vectors $(\vec{e}^{(0)}, \vec{e}^{(1)}, \vec{e}^{(2)}, \dots)$ is generated by a **power iteration**. Therefore, we know

$$\lim_{k \rightarrow \infty} \frac{\|\vec{e}^{(k+1)}\|}{\|\vec{e}^{(k)}\|} = \lambda_{\max}(\mathbf{I} - \mathbf{MA}) := \max\{|\lambda| : \lambda \in \sigma(\mathbf{I} - \mathbf{MA})\}, \quad (4.1.3.17)$$

for *any* vector norm $\|\cdot\|$: Asymptotically the decay of the iteration error will be determined by the largest eigenvalue of the error propagation matrix. ┘

Remark 4.1.3.18 (Measuring rates of convergence of stationary linear iterations) Write $\lambda_{\max}(\mathbf{X})$ for the largest (in modulus) eigenvalue of the matrix $\mathbf{X} \in \mathbb{R}^{N,N}$:

$$\lambda_{\max}(\mathbf{X}) := \max\{|\lambda| : \lambda \in \sigma(\mathbf{X})\}, \quad \sigma(\mathbf{X}) := \text{spectrum of } \mathbf{X}. \quad (4.1.3.19)$$

Then, for any matrix norm $\|\cdot\|$ induced by a vector norm

$$\|\mathbf{X}^k\| \rightarrow \lambda_{\max}(\mathbf{X})^k \quad \text{for } k \rightarrow \infty. \quad (4.1.3.20)$$

Hence, the **spectral radius** $\lambda_{\max}(\mathbf{X})$ will give precise information about the so-called **asymptotic rate** of linear convergence, which, after several steps, is a good approximation of the actual rate.

The computation of $\lambda_{\max}(\mathbf{X})$ relies on the **power iteration**, see [NumCSE Section 9.3.1] and Code 4.1.3.21

Pseudocode 4.1.3.21: Power method for computing $\lambda_{\max}(\mathbf{X})$, $\mathbf{X} \in \mathbb{R}^{N,N}$

```

1  real comp_lmax(const Matrix X ∈ ℝN,N, real TOL) {
2    Vector v̄ ∈ ℝN := random vector;
3    λnew := 0
4    do {
5      λold := λnew;
6      v̄ :=  $\frac{\vec{v}}{\|\vec{v}\|}$ ; // normalization
7      v̄ := Xv̄;
8      λnew :=  $\|\vec{v}\|$ ; // new guess for largest eigenvalue
9    }
10  while (  $\frac{|\lambda_{\text{new}} - \lambda_{\text{old}}|}{|\lambda_{\text{new}}|} > \text{TOL}$  ); // Terminate in case small relative change
11  return (λnew);
12 }
```

Next note that a stationary linear iteration (4.1.3.13) with $\vec{\varphi} = \mathbf{0}$ boils down to a power iteration with iteration matrix $\mathbf{I} - \mathbf{MA}$. Thus we can adapt Code 4.1.3.21 to compute the asymptotic rate of convergence of a stationary linear iteration for solving $\mathbf{A}\vec{\mu} = \vec{\varphi}$, a single step of which $(\vec{\mu}^{(k)} \rightarrow \vec{\mu}^{(k+1)})$ can be carried out by calling the function

Vector statLinIt(const Vector $\vec{\varphi} \in \mathbb{R}^N$, const Vector $\vec{\mu} \in \mathbb{R}^N$);

Pseudocode 4.1.3.22: Computing the asymptotic convergence rate of a stationary linear iteration available through `statLinIt()`.

```

1  real compAsympCvgRate(real TOL) {
2    Vector  $\vec{v} \in \mathbb{R}^N$  := random vector;
3     $\lambda_{\text{new}} := 0$ 
4    do {
5       $\lambda_{\text{old}} := \lambda_{\text{new}}$ ;
6       $\vec{v} := \frac{\vec{v}}{\|\vec{v}\|}$ ; // normalization
7       $\vec{v} := \text{statLinIt}(\mathbf{0}, \vec{v})$ ;
8       $\lambda_{\text{new}} := \|\vec{v}\|$ ; // new guess for largest eigenvalue
9    }
10   while (  $\frac{|\lambda_{\text{new}} - \lambda_{\text{old}}|}{|\lambda_{\text{new}}|} > \text{TOL}$  ); // Terminate in case small relative change
11   return ( $\lambda_{\text{new}}$ );
12 }
```

EXPERIMENT 4.1.3.23 (Convergence of Gauss-Seidel method for Poisson matrix) We measure the (asymptotic) rate of linear convergence of the Gauss-Seidel method from Code 4.1.3.3 when applied to the linear system of equations $\mathbf{A}\vec{\mu} = \vec{\varphi}$, where \mathbf{A} is the $N \times N$ Poisson matrix from (4.1.1.23) and $\vec{\varphi} = \mathbf{1}$ is the vector of all ones.

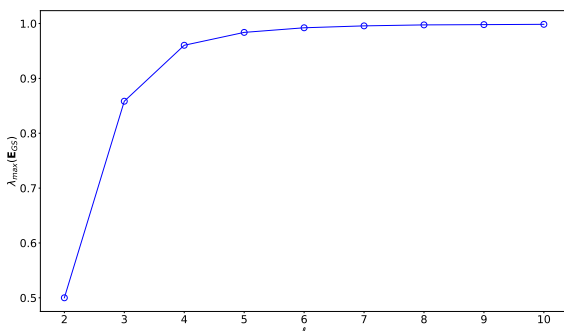


Fig. 161

We investigate the matrix sizes $N := M^2$, $M = 2^\ell - 1$, $\ell = 2, \dots, 10$, and compute a guess for the rate of linear convergence by means of the power iteration with $\text{TOL} = 10^{-3}$.

As initial guess we used $\mathbf{0}$.



We observe a massive deterioration of the rate of convergence for increasing matrix size.

A simple heuristic argument can make this observation plausible: If $\vec{v}_{\min} \in \mathbb{R}^N$, $\|\vec{v}_{\min}\| = 1$, is an eigenvector of the Poisson matrix \mathbf{A} belonging to the smallest eigenvalue λ_{\min} , then we expect ($\|\cdot\|$ the Euclidean norm)

$$\|(\mathbf{I} - \text{tril}(\mathbf{A})^{-1}\mathbf{A})\vec{v}_{\min}\| \approx 1 - \lambda_{\min}.$$

The eigenvectors and eigenfunctions of the Poisson matrix are well-known [Stü99, Ex. 3.1], [Hac94, Sect. 4.1]: for $k, m \in \{1, \dots, n-1\}$ we find

$$\begin{aligned} \text{eigenvectors : } & \left[\sin\left(\frac{i}{n}k\right) \right]_{i=1}^{n-1} \otimes \left[\sin\left(\frac{j}{n}m\right) \right]_{j=1}^{n-1} \in \mathbb{R}^N, \\ \text{eigenvalues : } & \lambda_{k,m} = 4 - 2 \cos\left(\frac{k\pi}{n}\right) - 2 \cos\left(\frac{m\pi}{n}\right) = 4 \sin^2\left(\frac{k\pi}{2n}\right) + 4 \sin^2\left(\frac{m\pi}{2n}\right). \end{aligned} \quad (4.1.3.24)$$

As a consequence we have $\lambda_{\min} = O(N^{-1})$ for $N \rightarrow \infty$ and the asymptotic rate of convergence of the Gauss-Seidel iteration will behave like $1 - O(N^{-1})$. Hence, writing $\vec{\mu}^{(k)}$ for the Gauss-Seidel iterates,

from (4.1.3.14) we can expect that after a few steps and for large N

$$\exists C > 0: \quad \left\| \vec{\mu}^* - \vec{\mu}^{(k+1)} \right\| \leq \left(1 - \frac{C}{N}\right) \left\| \vec{\mu}^* - \vec{\mu}^{(k)} \right\|.$$

In order to achieve error reduction by a factor of $\epsilon < 1$, we have to carry out at least

$$K \geq \frac{\log \epsilon}{\log(1 - \frac{C}{N})} \geq \frac{|\log \epsilon|}{C} \cdot N = O(N) \quad \text{for } N \rightarrow \infty$$

Gauss-Seidel steps. Since each step involves computational cost $O(N)$, we arrive at an asymptotic effort of $O(N^2)$ for solving $\mathbf{A}\vec{\mu} = \vec{\varphi}$ approximately up to a prescribed error level. This does not compare favorably with the effort required by a modern sparse direct solver, see Section 4.1.2. ┘

§4.1.3.25 (Composition of stationary linear iterations) Let us consider two interleaved stationary linear iterations

$$\begin{aligned} \vec{\mu}_{\text{tmp}} &= \vec{\mu}^{(k)} + \mathbf{M}_1(\vec{\varphi} - \mathbf{A}\vec{\mu}^{(k)}) \quad \text{with invertible } \mathbf{M}_1 \in \mathbb{R}^{N,N}, \\ \vec{\mu}^{(k+1)} &= \vec{\mu}_{\text{tmp}} + \mathbf{M}_2(\vec{\varphi} - \mathbf{A}\vec{\mu}_{\text{tmp}}) \quad \text{with invertible } \mathbf{M}_2 \in \mathbb{R}^{N,N}. \end{aligned} \tag{4.1.3.26}$$

By elementary algebra, this yields another stationary linear iteration

$$\vec{\mu}^{(k+1)} = \vec{\mu}^{(k)} + \mathbf{M}(\vec{\varphi} - \mathbf{A}\vec{\mu}^{(k)}), \quad \mathbf{M} := \mathbf{M}_1 + \mathbf{M}_2 - \mathbf{M}_2\mathbf{A}\mathbf{M}_1. \tag{4.1.3.27}$$

Naturally, its error propagation matrix must be the product of the two error propagation matrices of the involved stationary linear iterations:

$$\mathbf{I} - \mathbf{M}\mathbf{A} = (\mathbf{I} - \mathbf{M}_2\mathbf{A})(\mathbf{I} - \mathbf{M}_1\mathbf{A}). \tag{4.1.3.28}$$

4.1.4 Conjugate Gradient Method (CG)

From Lemma 4.1.1.13 we learn that finite element discretizations will lead to linear systems of equations with *large sparse symmetric positive definite* coefficient matrices. For this class of linear systems, the *conjugate gradient method* (CG) [NumCSE Section 10.2] is the most important iterative method.

§4.1.4.1 (CG iteration) The guiding idea of the CG method is minimization over Krylov spaces.

Definition 4.1.4.2. Krylov space

For $\mathbf{A} \in \mathbb{R}^{N,N}$, $\vec{\zeta} \in \mathbb{R}^N$, $\mathbf{z} \neq 0$, the ℓ -th **Krylov space**, $\ell \in \mathbb{N}$, is defined as

$$\mathcal{K}_\ell(\mathbf{A}, \vec{\zeta}) := \text{Span}\{\vec{\zeta}, \mathbf{A}\vec{\zeta}, \dots, \mathbf{A}^{\ell-1}\vec{\zeta}\}.$$

Now assume that \mathbf{A} is *symmetric positive definite*, that is,

$$\mathbf{A} = \mathbf{A}^\top \quad \text{and} \quad \vec{\zeta}^\top \mathbf{A} \vec{\zeta} > 0 \quad \forall \vec{\zeta} \in \mathbb{R}^N \setminus \{0\}. \tag{4.1.4.3}$$

Then the k -iterate of the CG iterative method for solving $\mathbf{A}\vec{\mu} = \vec{\varphi}$ with initial guess $\vec{\mu}^{(0)} \in \mathbb{R}^N$ is the unique vector minimizing the \mathbf{A} -norm/energy norm of the iteration error over $\vec{\mu}^{(0)} + \mathcal{K}_k(\mathbf{A}, \vec{\rho}_0)$, $\vec{\rho}_0 := \vec{\varphi} - \mathbf{A}\vec{\mu}^{(0)}$ [NumPDE ??]:

$$\vec{\mu}^{(k)} := \underset{\vec{v} \in \vec{\mu}^{(0)} + \mathcal{K}_k(\mathbf{A}, \vec{\rho}_0)}{\text{argmin}} \quad (\vec{v} - \vec{\mu}^*)^\top \mathbf{A} (\vec{v} - \vec{\mu}^*), \quad \vec{\rho}_0 := \vec{\varphi} - \mathbf{A}\vec{\mu}^{(0)}, \quad \vec{\mu}^* := \mathbf{A}^{-1}\vec{\varphi}. \tag{4.1.4.4}$$

This defines a *non-linear* stationary iterative method, because the sequence $(\vec{\mu}^{(k)})$ cannot be generated by an iteration of type (4.1.3.13).

It can be shown that $\vec{\mu}^{(N)} = \vec{\mu}^*$; the CG method, when executed in exact arithmetic, will yield the exact solution after at most N steps. ┘

§4.1.4.5 (CG algorithm [NumCSE Section 10.2.2]) Amazingly, the computation of the iterate $\vec{\mu}^{(k)}$ according to (4.1.4.4) requires only k matrix×vector multiplications (with \mathbf{A}) in addition to $\sim 6kN$ arithmetic operations!

The next pseudocode gives a mathematical definition of the conjugate gradient method applied to the LSE $\mathbf{A}\vec{\mu} = \vec{\varphi}$. For the derivation refer to [NumCSE Section 10.2].

```

Pseudocode 4.1.4.6: Conjugate gradient method
1  Vector cg( $\mathbf{A} \in \mathbb{R}^{N,N}$ ,  $\vec{\varphi} \in \mathbb{R}^N$ ,  $\vec{\mu}^{(0)}$ ) {
2     $\vec{\zeta}_1 = \vec{\rho}_0 := \vec{\varphi} - \mathbf{A}\vec{\mu}^{(0)}$ ;
3    for(  $j=1$ ;  $j < \text{maxit}$ ;  $++j$ ) {
4       $\vec{\mu}^{(j)} := \vec{\mu}^{(j-1)} + \frac{\vec{\zeta}_j^\top \vec{\rho}_{j-1}}{\vec{\zeta}_j^\top \mathbf{A}\vec{\zeta}_j} \vec{\zeta}_j$ ;
5       $\vec{\rho}_j = \vec{\rho}_{j-1} - \frac{\vec{\zeta}_j^\top \vec{\rho}_{j-1}}{\vec{\zeta}_j^\top \mathbf{A}\vec{\zeta}_j} \mathbf{A}\vec{\zeta}_j$ ;
6       $\vec{\zeta}_{j+1} = \vec{\rho}_j - \frac{(\mathbf{A}\vec{\zeta}_j)^\top \vec{\rho}_j}{\vec{\zeta}_j^\top \mathbf{A}\vec{\zeta}_j} \vec{\zeta}_j$ ;
7      if ( $\|\vec{\rho}_j\| < \text{TOL} \cdot \|\vec{\rho}_0\|$ ) return ( $\vec{\mu}^{(j)}$ );
8    }
9  }
    
```

◁ CG-Algorithm for solving LSE $\mathbf{A}\vec{\mu} = \vec{\varphi}$

Input:

- ◆ S.p.d. matrix $\mathbf{A} \in \mathbb{R}^{N,N}$,
- ◆ right-hand-side vector $\vec{\varphi}$,
- ◆ initial guess $\vec{\mu}^{(0)} \in \mathbb{R}^N$,
- ◆ tolerance TOL for termination criterion.

Return value: approximate solution.

Cost of CG step

A single CG step requires **one** \mathbf{A} ×vector multiply plus a small number of vector operations.

▶ The cost for a single CG step applied to an $N \times N$ finite element linear system is $O(N)$ asymptotically for $N \rightarrow \infty$. ┘

§4.1.4.8 (Convergence of CG [NumCSE Section 10.2.3]) For a symmetric positive definite matrix $\mathbf{A} \in \mathbb{R}^{N,N}$ we denote by $\|\cdot\|_{\mathbf{A}}$ the **energy norm** induced by \mathbf{A} :

$$\|\vec{v}\|_{\mathbf{A}}^2 := \vec{v}^\top \mathbf{A} \vec{v}, \quad \vec{v} \in \mathbb{R}^N. \tag{4.1.4.9}$$

This energy norm is fundamental in the theory of the CG method [NumCSE Cor. 10.2.3.3] and it is in this energy norm that convergence estimates are stated. We also need the notion of the **spectral condition number** of an invertible matrix

$$\kappa(\mathbf{A}) := \frac{\lambda_{\max}(\mathbf{A})}{\lambda_{\min}(\mathbf{A})} = \frac{\max\{|\lambda| : \lambda \in \sigma(\mathbf{A})\}}{\min\{|\lambda| : \lambda \in \sigma(\mathbf{A})\}}. \tag{4.1.4.10}$$

Theorem 4.1.4.11. Convergence of the CG method [NumCSE Thm. 10.2.3.5]

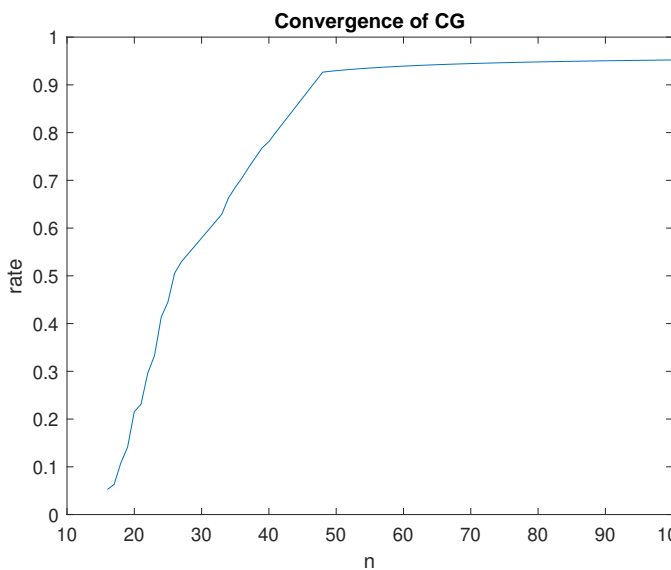
The iterates of the CG method for solving $\mathbf{A}\vec{\mu} = \vec{\varphi}$ (see Code 4.1.4.6) with $\mathbf{A} = \mathbf{A}^\top \in \mathbb{R}^{N,N}$ s.p.d. satisfy

$$\|\vec{\mu}^* - \vec{\mu}^{(l)}\|_A \leq 2\rho^l \|\vec{\mu}^* - \vec{\mu}^{(0)}\|_A, \quad \rho := \frac{\sqrt{\kappa(\mathbf{A})} - 1}{\sqrt{\kappa(\mathbf{A})} + 1}, \quad l \in \mathbb{N},$$

where $\mathbf{A}\vec{\mu}^* = \vec{\varphi}$.

► The larger $\kappa(\mathbf{A})$ the slower the convergence of CG! ◄

EXPERIMENT 4.1.4.12 (Convergence of CG for the Poisson matrix) We apply the CG method to a linear system with the Poisson matrix (4.1.1.23) as coefficient matrix.



We record the “approximate asymptotic convergence rates”

$$\text{rate} \approx \sqrt[10]{\frac{\|\vec{\mu}^{(30)} - \vec{\mu}^*\|_A}{\|\vec{\mu}^{(20)} - \vec{\mu}^*\|_A}},$$

for $\vec{\mu}^* = \mathbf{1}$ and $\vec{\mu}^{(0)} = \mathbf{0}$.

We measure these rates of convergence for $N = (n - 1)^2, n = 5, 6, \dots, 30$.

◀ We observe a pronounced deterioration of CG convergence for larger N .

Fig

§4.1.4.13 (CG convergence for FE linear systems) The observation made in the previous experiment can be concluded from Thm. 4.1.4.11 and [NumPDE Lemma 9.2.7.30]. That theorem told us that for finite element Galerkin matrices \mathbf{A} for second-order scalar elliptic boundary value problems (4.1.1.8) and trial/test spaces $\mathcal{S}_{1,0}^0(\mathcal{M})$ we have

$$0 < \lambda_{\min}(\mathbf{A}) \leq C, \quad \lambda_{\max}(\mathbf{A}) \geq C'h_{\mathcal{M}}^{-2}, \tag{4.1.4.14}$$

with constants $C, C' > 0$ depending only on the shape-regularity measure (\rightarrow [NumPDE Def. 3.3.2.20]) and quasi-uniformity of the mesh \mathcal{M} . As a consequence

$$\kappa(\mathbf{A}) \geq Ch_{\mathcal{M}}^{-2}. \tag{4.1.4.15}$$

Hence, by Thm. 4.1.4.11 we expect **slower convergence on finer meshes**, exactly what we have observed in Exp. 4.1.4.12. In fact, $\kappa(\mathbf{A}) \approx h_{\mathcal{M}}^{-2}$, which gives, asymptotically on sequences of uniformly and regularly refined meshes

$$\|\vec{\mu}^* - \vec{\mu}^{(k)}\|_A \leq 2(1 - O(h_{\mathcal{M}}))^{2k} \|\vec{\mu}^* - \vec{\mu}^{(0)}\|_A \quad \text{for meshwidth } h_{\mathcal{M}} \rightarrow 0. \tag{4.1.4.16}$$

In two dimensions we have $N = O(h_{\mathcal{M}}^{-2})$, which means that we get an asymptotic reduction of the energy norm of the CG iteration error by a factor of $\epsilon < 1$, if we carry out at least

$$K \geq \frac{\log \epsilon}{\log(1 - CN^{-\frac{1}{2}})} \geq \frac{\log \epsilon}{C} N^{\frac{1}{2}} = O(\sqrt{N}) \quad \text{for } N \rightarrow \infty$$

CG steps. We conclude an asymptotic computational effort of $O(N^{\frac{3}{2}})$ for solving $A\vec{\mu} = \vec{\varphi}$ up to a prescribed relative accuracy. This is superior to the Gauss-Seidel method, but not better than the advanced sparse direct solvers mentioned in Section 4.1.2. ┘

4.2 Geometric Multigrid Method

Recall the Gauss-Seidel iteration for solving the linear system of equations $A\vec{\mu} = \vec{\varphi}$,

$$\vec{\mu}^{(0)} := \vec{\mu}_0, \quad \vec{\mu}^{(k+1)} = \vec{\mu}^{(k)} + \mathbf{M}(\vec{\varphi} - A\vec{\mu}^{(k)}) \quad \text{with} \quad \mathbf{M} := \text{tril}(A)^{-1}, \quad (4.1.3.7)$$

for which we found the error recursion

$$\vec{\epsilon}^{(k+1)} = (\mathbf{I} - \mathbf{M}\mathbf{A})\vec{\epsilon}^{(k)} \quad \text{for the iteration error} \quad \vec{\epsilon}^{(k)} := \vec{\mu}^* - \vec{\mu}^{(k)}. \quad (4.1.3.14)$$



Idea: Study the eigenvector belonging to the largest (in modulus) eigenvalue of $\mathbf{I} - \mathbf{M}\mathbf{A}$

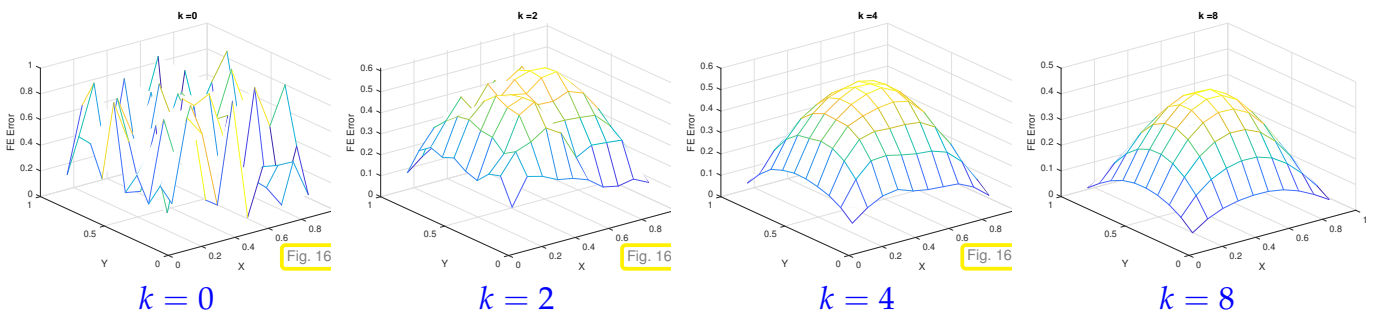
= most slowly converging error component!

EXPERIMENT 4.2.0.1 (Convergence of Gauss-Seidel II, see also Exp. 4.1.3.23) As in Exp. 4.1.3.23 we study the Gauss-Seidel iteration

$$\vec{\mu}^{(k+1)} = \vec{\mu}^{(k)} + \text{tril}(A)^{-1}(\vec{\varphi} - A\vec{\mu}^{(k)}),$$

for the 2D Poisson matrix A as defined in (4.1.1.23). We choose $\vec{\varphi} := A\vec{\mu}^*$ with a random vector $\vec{\mu}^* \in \mathbb{R}^N$ (entries equidistributed in $[0, 1]$), and initial guess $\vec{\mu}^{(0)} = \mathbf{0}$.

- 1 For $N = 100$ we we plot the finite element “error” functions $e_h^{(k)} \in S_{1,0}^0(\mathcal{M})$ with nodal coefficient vectors $\vec{\mu}^* - \vec{\mu}^{(k)}$ generated by the Gauss-Seidel iteration (4.1.3.7).



We observe that after several steps of the Gauss-Seidel iteration the iteration error viewed as a finite element function becomes **smooth**.

- 2 For the Poisson matrix A given in (4.1.1.23) we inspect the finite element functions defined by the eigenvectors of the error propagation matrix $\mathbf{E} := \mathbf{I} - \text{tril}(A)^{-1}A$ belonging to the largest eigenvalue.

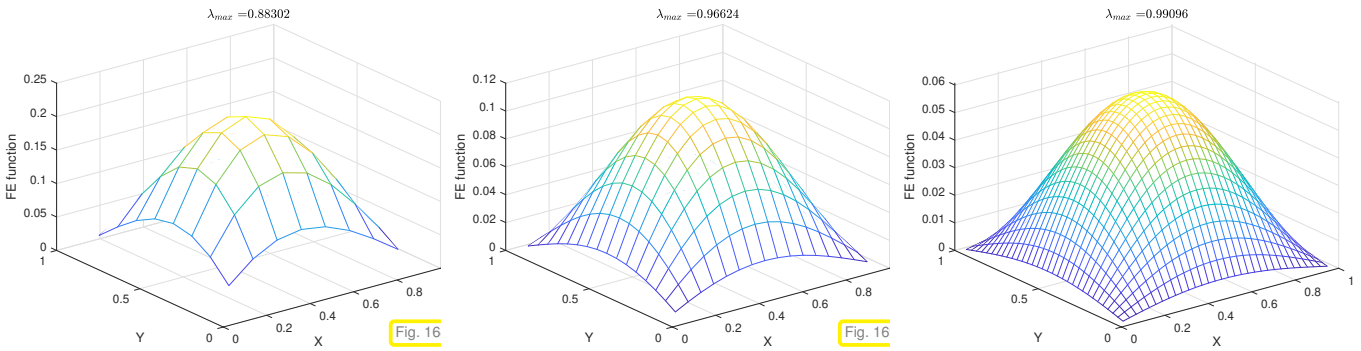


Fig. 16

Fig. 16

Fig. 16

$N = 64, \lambda_{\max}(\mathbf{E}) = 0.88302 \quad N = 256, \lambda_{\max}(\mathbf{E}) = 0.96624 \quad N = 1024, \lambda_{\max}(\mathbf{E}) = 0.99096$

We observe that the “most slowly converging” error functions are **smooth** and their per-step reduction as measured by $\lambda_{\max}(\mathbf{E})$ becomes smaller with increasing N : $\lambda_{\max}(\mathbf{E}) \rightarrow 1$ as $N \rightarrow \infty$.

- Now we examine the finite element function defined by the eigenfunction of the Gauss-Seidel error propagation matrix \mathbf{E} belonging to the smallest (in modulus) eigenvalue.

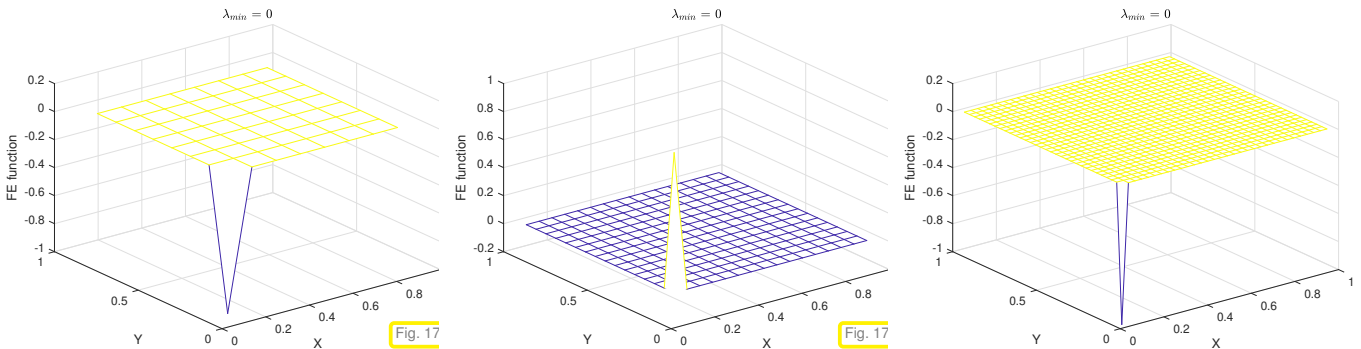


Fig. 17

Fig. 17

Fig. 17

$N = 64, \lambda_{\min}(\mathbf{E}) = 0 \quad N = 256, \lambda_{\min}(\mathbf{E}) = 0 \quad N = 1024, \lambda_{\min}(\mathbf{E}) = 0$

Obviously, the “fastest converging” error functions are **highly localized** and they experience an (almost) N -independent per-step reduction given by $\lambda_{\min}(\mathbf{E})$. ┘

Behavior of the Gauss-Seidel iteration error

When applied to LSE arising from the finite element discretization of scalar 2nd-order elliptic boundary value problems **on fine meshes** (large N , small h_M), the Gauss-Seidel iteration

- ◆ effects a fast reduction of highly-oscillatory error components,
- ◆ fails to reduce smooth error components significantly.

4.2.1 Subspace Correction Methods

The relationship between variational problems, linear systems of equations, and minimization problems, hinted at in § 4.1.1.9 suggests an abstract approach to the construction of iterative solution methods for finite element linear systems of equations with s.p.d. coefficient matrix.

Let us assume that \mathbf{A} is spawned by the Galerkin discretization of a linear variational problem

$$u_h \in V_h: \quad a(u_h, v_h) = \ell(v_h) \quad \forall v_h \in V_h, \tag{4.2.1.1}$$

using the finite-dimensional trial/test space V_h and its basis $\{b_h^1, \dots, b_h^N\} \subset V_h, N := \dim V_h$. Thus, we assume $(\mathbf{A})_{i,j} := a(b_h^i, b_h^j), 1 \leq i, j \leq N$.

If the bilinear form $\mathbf{a}(\cdot, \cdot)$ is symmetric and positive definite, then (4.2.1.1) is equivalent to the quadratic minimization problem

$$u_h = \underset{v_h \in V_h}{\operatorname{argmin}} J(v) \quad , \quad J(v) := \frac{1}{2} \mathbf{a}(v, v) - \ell(v) . \quad (4.2.1.2)$$

The scheme outlined next is a natural iterative approach to solving (4.2.1.2).

Definition 4.2.1.3. (Successive) subspace correction method

Given an additive decomposition (not necessarily direct)

$$V_h = \sum_{m=1}^M V_m \quad , \quad \text{with subspaces } V_m \subset V_h \quad M \in \mathbb{N} \quad , \quad (4.2.1.4)$$

a single step $u_h^{(k)} \rightarrow u_h^{(k+1)}$ of the induced (successive) subspace correction iteration is defined as

$$u_h^{(k+1)} := u_h^{(k)} \quad , \quad u_h^{(k+1)} \leftarrow u_h^{(k+1)} + \underset{w_m \in V_m}{\operatorname{argmin}} J(u_h^{(k+1)} + w_m) \quad , \quad m = 1, \dots, n . \quad (4.2.1.5)$$

Remember from [NumPDE Section 1.4.1] that the necessary and sufficient optimality conditions for a quadratic minimization problem with s.p.d. bilinear form amount to a linear variational problem. Thus, any successive subspace correction method can also be reformulated in terms of linear variational problems restricted to the subspaces V_m , because

$$J(u_h + w_m) = \frac{1}{2} \mathbf{a}(w_m, w_m) - (\ell(w_m) - \mathbf{a}(u_h, w_m)) + \mathbf{a}(u_h, u_h) - \ell(u_h)$$

is a quadratic functional in $w_m \in V_m$. Hence, by the equivalence of linear variational problems with s.p.d. bilinear forms and quadratic minimization problems,

$$\begin{aligned} v_m &= \underset{w_m \in V_m}{\operatorname{argmin}} J(u_h^{(k+1)} + w_m) \\ &\Updownarrow \\ v_m \in V_m: \quad \mathbf{a}(v_m, w_m) &= r(u_h; w_m) := \ell(w_m) - \mathbf{a}(u_h, w_m) \quad \forall w_m \in V_m \quad , \end{aligned}$$

with the residual linear form $w \mapsto r(u_h; w) := \ell(w) - \mathbf{a}(u_h, w)$, $w \in V_h$. As a consequence, the subspace correction iteration (4.2.1.5) can be recast as

$$\begin{cases} u_h^{(k+1)} := u_h^{(k)} \quad , \\ \left\{ \begin{array}{l} v_m \in V_m: \quad \mathbf{a}(v_m, w_m) = r(u_h^{(k+1)}; w_m) \quad \forall w_m \in V_m \quad , \\ u_h^{(k+1)} \leftarrow u_h^{(k+1)} + v_m \quad , \end{array} \right. \quad m = 1, \dots, M . \end{cases} \quad (4.2.1.6)$$

We switch to an algebraic perspective: Assume that we are given a basis $\{b_m^1, \dots, b_m^{N_m}\}$ of V_m , $N_m := \dim V_m$. Then we can express v_m from (4.2.1.6) as a linear combination

$$v_m = \sum_{k=1}^{N_m} (\tilde{v}_m)_k b_m^k \quad \text{for some } \tilde{v}_m \in \mathbb{R}^{N_m} .$$

The coefficient vector $\tilde{v}_m \in \mathbb{R}^{N_m}$ can be computed as the solution of the $N_m \times N_m$ linear system of equations

$$\mathbf{A}_m \tilde{v}_m = \tilde{\rho}_m(u_h) \quad \text{with} \quad \begin{aligned} (\mathbf{A}_m)_{ij} &= \mathbf{a}(b_m^j, b_m^i) \quad , \quad i, j \in \{1, \dots, N_m\} \quad , \\ (\tilde{\rho}_m(u_h))_i &:= r(u_h, b_m^i) \quad , \quad i \in \{1, \dots, N_m\} . \end{aligned} \quad (4.2.1.7)$$

Since $V_m \subset V_h$, the basis functions b_m^i are linear combinations of the basis functions b_h^k of V_h :

$$\blacktriangleright \quad \exists \mathbf{P}_m \in \mathbb{R}^{N_m \times N_m}: \quad b_m^i = \sum_{k=1}^{N_m} (\mathbf{P}_m)_{k,i} b_h^k, \quad i \in \{1, \dots, N_m\}, \quad m = 1, \dots, M. \quad (4.2.1.8)$$

Exploiting the bilinearity of $\mathbf{a}(\cdot, \cdot)$ and the linearity of $r(u_h, \cdot)$, we find

$$\mathbf{A}_m = \mathbf{P}_m^\top \mathbf{A} \mathbf{P}_m, \quad (4.2.1.9)$$

$$\tilde{\rho}_m(u_h) = \mathbf{P}_m^\top (\vec{\phi} - \mathbf{A} \vec{\mu}), \quad (4.2.1.10)$$

where $\vec{\mu} \in \mathbb{R}^N$ is the coefficient vector of $u_h \in V_h$ with respect to the basis $\{b_h^1, \dots, b_h^N\}$ of V_h . This implies that the solution of (4.2.1.7) reads

$$\tilde{v}_m = \left(\mathbf{P}_m^\top \mathbf{A} \mathbf{P}_m \right)^{-1} \mathbf{P}_m^\top (\vec{\phi} - \mathbf{A} \vec{\mu}^{(k+1)}),$$

with the coefficient vector $\vec{\mu}^{(k+1)}$ of $u_h^{(k+1)}$. Also \tilde{v}_m describes a function in $V_m \subset V_h$ and this function is represented by a coefficient vector $\vec{v}_m \in \mathbb{R}^{N_m}$, too:

$$\vec{v}_m = \mathbf{P}_m \tilde{v}_m = \mathbf{P}_m \left(\mathbf{P}_m^\top \mathbf{A} \mathbf{P}_m \right)^{-1} \mathbf{P}_m^\top (\vec{\phi} - \mathbf{A} \vec{\mu}^{(k+1)}). \quad (4.2.1.11)$$

This gives the final algebraic version of (4.2.1.6):

$$\begin{cases} \vec{\mu}^{(k+1)} := \vec{\mu}^{(k)}, \\ \vec{v}_m := \mathbf{P}_m \left(\mathbf{P}_m^\top \mathbf{A} \mathbf{P}_m \right)^{-1} \mathbf{P}_m^\top (\vec{\phi} - \mathbf{A} \vec{\mu}^{(k+1)}), \quad m = 1, \dots, M, \\ \vec{\mu}^{(k+1)} \leftarrow \vec{\mu}^{(k+1)} + \vec{v}_m, \end{cases} \quad (4.2.1.12)$$

The highlighted formula provides the **subspace correction** in the direction of V_m . Actually, matching (4.2.1.12) with the general recursion formula

$$\vec{\mu}^{(k+1)} = \vec{\mu}^{(k)} + \mathbf{M} (\vec{\phi} - \mathbf{A} \vec{\mu}^{(k)}) \quad \text{with } \mathbf{M} \in \mathbb{R}^{N, N}, \quad (4.1.3.13)$$

for a generic stationary linear iteration (for solving the LSE $\mathbf{A} \vec{\mu} = \vec{\phi}$), we see that a single subspace correction step amounts to carrying out one step of a stationary linear iteration with $\mathbf{M} := \mathbf{P}_m \left(\mathbf{P}_m^\top \mathbf{A} \mathbf{P}_m \right)^{-1} \mathbf{P}_m^\top$.

The following pseudocode implements a subspace correction iteration for the linear system of equations $\mathbf{A} \vec{\mu} = \vec{\phi}$. The function takes the right-hand side vector $\vec{\phi} \in \mathbb{R}^N$ and the initial guess $\vec{\mu}$ as arguments and returns the final approximation in $\vec{\mu}$. The codes assumes that the basis transformation matrices \mathbf{P}_m , $m = 1, \dots, M$, are known. Termination triggered when the relative size of the update of $\vec{\mu}$ drops below a specified threshold **TOL**.

Pseudocode 4.2.1.13: Algebraic (successive) subspace correction method

```

1 void ssc(const Vector φ ∈ ℝN, ref Vector μ, real TOL) {
2   // Precompute Galerkin matrices in subspaces
3   Compute Am := PmA Pm⊤ ∈ ℝNm × Nm, m = 1, ..., M;
4   do {
5     μold := μ;
6     for (int m = 1; m < M; m++) {
```

```

7   Compute  $\vec{\rho}_m := \mathbf{P}_m^\top (\vec{\varphi} - \mathbf{A}\vec{\mu})$ ;
8   Solve  $\mathbf{A}_m \vec{\gamma} = \vec{\rho}$ ; //  $N_m \times N_m$  LSE
9    $\vec{\mu} \leftarrow \vec{\mu} + \vec{\gamma}$ ; // Update in the direction of  $V_m$ 
10  }
11  }
12  while ( $\|\vec{\mu} - \vec{\mu}_{\text{old}}\| > \text{TOL} \cdot \|\vec{\mu}\|$ ); // Termination test
13  }
```

From (4.2.1.12) it is clear that

- (i) the correction in the direction of V_m already defines a stationary linear iteration of the form (4.1.3.13) with $\mathbf{M} = \mathbf{P}_m (\mathbf{P}_m^\top \mathbf{A} \mathbf{P}_m)^{-1} \mathbf{P}_m^\top$,
- (ii) the whole subspace correction iteration is the composition of subspace corrections in individual directions as introduced in § 4.1.3.25.

Hence, from § 4.1.3.25 and, in particular (4.1.3.28), we learn that the whole subspace correction iteration is a stationary linear iterative method, whose error propagation matrix is

$$\mathbf{E}_{\text{SSC}} := \mathbf{I} - \mathbf{M}_{\text{SSC}} \mathbf{A} = (\mathbf{I} - \mathbf{P}_M (\mathbf{P}_M^\top \mathbf{A} \mathbf{P}_M)^{-1} \mathbf{P}_M^\top \mathbf{A}) \cdots (\mathbf{I} - \mathbf{P}_1 (\mathbf{P}_1^\top \mathbf{A} \mathbf{P}_1)^{-1} \mathbf{P}_1^\top \mathbf{A}). \quad (4.2.1.14)$$

§4.2.1.15 (Gauss-Seidel as a subspace correction method) Now we view the Gauss-Seidel stationary linear iteration for a s.p.d. finite element Galerkin matrix $\mathbf{A} \in \mathbb{R}^{N,N}$ as defined in Code 4.1.3.3/(4.1.3.7) from a new angle and identify it as a particular subspace correction method.

To that end, we consider the very special situation

$$M = N \quad , \quad N_m = 1 \quad , \quad b_m^1 = b_h^m \quad ,$$

which yields a subspace correction method with one-dimensional subspaces spanned by a single basis function of V_h each. In this case we have

$$\mathbf{P}_m = \vec{\varepsilon}_m \quad \hat{=} \quad m\text{-th coordinate vector} \quad , \quad \mathbf{A}_m = (\mathbf{A})_{m,m} \quad , \quad m = 1, \dots, N.$$

This means

$$\vec{\mu} \leftarrow \vec{\mu} + \mathbf{P}_m \mathbf{A}_m^{-1} \mathbf{P}_m^\top (\vec{\varphi} - \mathbf{A}\vec{\mu}) \iff (\vec{\mu})_m \leftarrow (\vec{\mu})_m + \frac{1}{(\mathbf{A})_{m,m}} \left((\vec{\varphi})_m - \sum_{j=1}^N (\mathbf{A})_{m,j} (\vec{\mu})_j \right). \quad (4.2.1.16)$$

This perfectly agrees with what is done in the inner loop body of the Gauss-Seidel implementation Code 4.1.3.3. Carrying out (4.2.1.16) sequentially for $m = 1, \dots, N$, we recover one step of the Gauss-Seidel method for the LSE $\mathbf{A}\vec{\mu} = \vec{\varphi}$! Hence, Gauss-Seidel is a subspace correction iteration based on the special type of splitting (4.2.1.4)

$$V_h = \sum_{m=1}^N \text{Span}\{b_h^m\}. \quad (4.2.1.17)$$

Gauss-Seidel for FE LSEs = local subspace correction

Gauss-Seidel for a finite-element linear system of equations realizes a successive subspace correction in the directions of finite elements basis functions.

Since finite element basis functions invariably have localized supports, it is not surprising that, when applied on fine meshes, the Gauss-Seidel iteration cannot cope with smooth, that is, **long-range** error components. \lrcorner

4.2.2 Convergence of SSC Methods

§4.2.2.1 (SSC = Method of successive projections) Recall a concept from linear algebra.

Definition 4.2.2.2. Orthogonal projection in finite dimensions

Let V be a finite-dimensional real vector space equipped with an inner product, a *symmetric positive definite* bilinear form $a : V \times V \rightarrow \mathbb{R}$. For a subspace $U \subset V$ the mapping $Q : V \mapsto U$ defined by

$$Qv \in U: \quad a(Qv, u) = a(v, u) \quad \forall u \in U \quad (4.2.2.3)$$

is an **a-orthogonal projection**.

Corollary 4.2.2.4. Properties of orthogonal projections

The orthogonal projection $Q : V \rightarrow U$ from Def. 4.2.2.2

- (i) is a *linear mapping*,
- (ii) is *idempotent*, that is $Q \circ Q = Q$,
- (iii) is *a-selfadjoint*, $a(Qv, w) = a(v, Qw)$ for all $v, w \in V$,
- (iv) and satisfies $\|Qv\|_A \leq \|v\|_A$, where $\|\cdot\|_A$ is the (energy) norm induced by $a(\cdot, \cdot)$.

Proof.

- (i) Linearity of Q is an immediate consequence of the linearity of a .
- (ii) $Q^2 = Q$ follows by inserting $v := Qw$, $w \in V$, into (4.2.2.3).
- (iii) Directly from the definition, since $Qv \in U$:

$$a(Qv, w) = a(Qv, Qw) = a(v, Qw) \quad \forall v, w \in V.$$

- (iv) The contraction property is immediate from

$$\|Qv\|_A^2 = a(Qv, Qv) \stackrel{(4.2.2.3)}{=} a(v, Qv) \leq \|v\|_A \|Qv\|_A,$$

where we use the Cauchy-Schwarz inequality in the last step. \square

Let us return to the initial linear variational problem

$$u_h \in V_h: \quad a(u_h, v_h) = \ell(v_h) \quad \forall v_h \in V_h, \quad (4.2.1.1)$$

and the linear variational problem (4.2.1.6) to be solved to compute the subspace correction v_m of $u_h^{(k)} \in V_h$ in direction $V_m \subset V_h$:

$$v_m \in V_m: \quad a(v_m, w_m) = \ell(w_m) - a(u_h^{(k)}, w_m) \quad \forall w_m \in V_m. \quad (4.2.2.5)$$

Writing $u_h^* \in V_h$ for the solution of (4.2.1.1), that is, $a(u_h^*, w_h) = \ell(w_h)$ for all $w_h \in V_h$, (4.2.2.5) is equivalent to

$$v_m \in V_m: a(v_m, w_m) = a(u_h^* - u_h^{(k)}, w_m) \quad \forall w_m \in V_m. \tag{4.2.2.6}$$

Assumption 4.2.2.7. a defines an inner product

The bilinear form $a : V_h \times V_h \rightarrow \mathbb{R}$ is symmetric and positive definite (s.p.d.).

Under this assumption and in light of Def. 4.2.2.2 from (4.2.2.6) we conclude that the correction $v_m \in V_m$ is the a -orthogonal projection of the iteration error $u_h^* - u_h^{(k)}$ onto V_m :

$$v_m = Q_m(u_h^* - u_h^{(k)}), \quad Q_m : V_h \rightarrow V_m \hat{=} a\text{-orthogonal projection onto } V_m. \tag{4.2.2.8}$$

The error after the correction is

$$u_h^* - (u_h^{(k)} + v_m) = (\text{Id} - Q_m)(u_h^* - u_h^{(k)}). \tag{4.2.2.9}$$

Error propagation in SSC

Under Ass. 4.2.2.7, subspace correction in the direction of V_m amounts to the a -orthogonal projection of the iteration error onto the a -orthogonal complement of V_m .

Thus, the error propagation of one step of the SSC method of ?? can be described by the action of the operator product

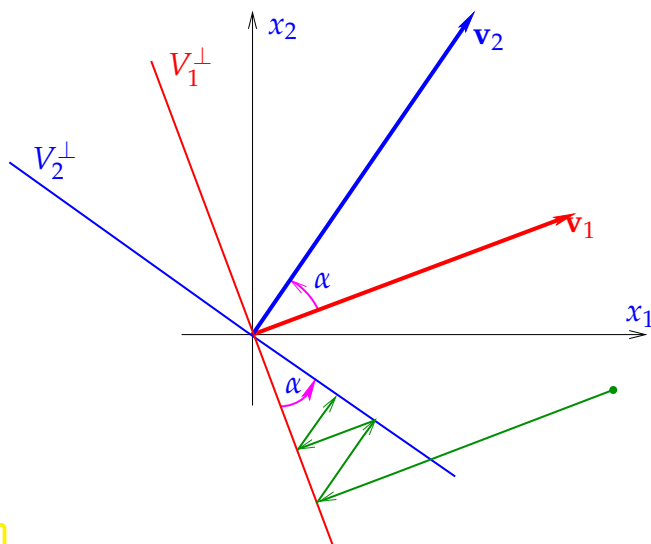
$$E_{\text{SSC}} = (\text{Id} - Q_M) \circ (\text{Id} - Q_{M-1}) \circ \dots \circ (\text{Id} - Q_2) \circ (\text{Id} - Q_1). \tag{4.2.2.11}$$

We may call $E_{\text{SSC}} : V_h \rightarrow V_h$ the **error propagation operator** of the SSC method of Def. 4.2.1.3. The error propagation matrix \mathbf{E}_{SSC} from (4.2.1.14) is its representation with respect to the chosen basis of V_h . ┘

Remark 4.2.2.12 (The geometry of SSC) We study SSC in Euclidean space \mathbb{R}^2 with $M = 2$:

$$\mathbb{R}^2 = V_1 + V_2, \quad V_m := \text{Span}\{\mathbf{v}_m\}, \quad m = 1, 2, \quad \mathbf{v}_m \in \mathbb{R}^2. \tag{4.2.2.13}$$

The role of the bilinear form $a(\cdot, \cdot)$ is played by the Euclidean inner product.



◁ $\bullet \hat{=} \text{initial error } \mathbf{e}^{(0)}$

In turns, the error vector is projected onto $V_1^\perp = \text{Span}\{\mathbf{v}_1\}^\perp$ and $V_2^\perp = \text{Span}\{\mathbf{v}_2\}^\perp$.

$$\|\mathbf{e}^{(k+1)}\|_2 = \cos^2 \alpha \cdot \|\mathbf{e}^{(k)}\|_2, \tag{4.2.2.14}$$

where $\alpha \in [0, \pi/2]$ is the angle enclosed by the subspaces V_1 and V_2 .

The larger the angle $\angle(V_1, V_2)$ the faster the convergence of iterated SSC.

Fig. 173

Remark 4.2.2.15 (SSC with $M = 2$: Method of alternating projections) Ass. 4.2.2.7 still applies. We consider the SSC iteration for $M = 2$ in the special setting

$$V_h = V_1 + V_2 \quad , \quad V_1 \cap V_2 = \{0\} \quad , \quad (4.2.2.16)$$

which means that V_h is the **direct sum** of V_1 and V_2 . The angle between V_1 and V_2 , $\angle(V_1, V_2) \in [0, \pi/2]$, is defined as

$$\delta := \cos \angle(V_1, V_2) = \sup \left\{ \frac{|\mathfrak{a}(v_1, v_2)|}{\|v_1\|_A \|v_2\|_A} : v_1 \in V_1 \setminus \{0\}, v_2 \in V_2 \setminus \{0\} \right\} . \quad (4.2.2.17)$$

We denote by $Z_i := \text{Id} - Q_i$, $i = 1, 2$, the \mathfrak{a} -orthogonal projection onto the orthogonal complement V_i^\perp . Then, appealing to (4.2.1.14),

$$\begin{aligned} \|E_{\text{SSC}}\|_A^2 &= \|Z_2 Z_1\|_A^2 = \sup \left\{ \|Z_2 Z_1 v\|_A^2 : \|v\|_A = 1 \right\} = \sup \{ \mathfrak{a}(Z_2 Z_1 v, Z_2 Z_1 v) : \|v\|_A = 1 \} \\ &= \sup \left\{ \mathfrak{a}(Z_1 v, Z_2^2 Z_1 v) : \|v\|_A = 1 \right\} \\ &\leq \delta \sup \{ \|Z_1 v\|_A \|Z_2 Z_1 v\|_A : \|v\|_A = 1 \} \leq \delta \|E_{\text{SSC}}\|_A . \end{aligned} \quad (4.2.2.18)$$

We used that orthogonal projections are selfadjoint, idempotent, contracting, together with (4.2.2.17).

Thus, also in this case the angle enclosed by V_1 and V_2 directly determines the rate of linear convergence (w.r.t. the energy norm). \lrcorner

§4.2.2.19 (SSC with orthogonal subspaces) Obviously, in the setting of Rem. 4.2.2.12, if V_1 and V_2 are orthogonal, then the exact solution is recovered after only one step of SSC. This reflects a general fact that we want to state and prove now.

Theorem 4.2.2.20. SSC with orthogonal subspaces

In the context of the SSC method of Def. 4.2.1.3, if the spaces $V_m \subset V_h$ are mutually \mathfrak{a} -orthogonal, then the iteration produces the exact solution of (4.2.1.1) after only one step.

The theorem is a consequence of the following auxiliary result.

Lemma 4.2.2.21. Orthogonal projections onto orthogonal subspaces

Let V be equipped with an inner product $\mathfrak{a} : V \times V \rightarrow \mathbb{R}$, and let $Q_1 : V \rightarrow V_1$ and $Q_2 : V \rightarrow V_2$ be two \mathfrak{a} -orthogonal projections onto subspaces V_1 and V_2 , respectively.

If V_1 and V_2 are \mathfrak{a} -orthogonal in the sense that

$$\mathfrak{a}(v_1, v_2) = 0 \quad \forall v_1 \in V_1, v_2 \in V_2 \quad ,$$

then

- (i) *the composition of Q_1 and Q_2 vanishes, $Q_1 \circ Q_2 = 0$, and*
- (ii) *$Q_1 + Q_2$ is the \mathfrak{a} -orthogonal projection onto $V_1 + V_2$*

Proof.

- (i) By the very definition of an orthogonal projection $T := Q_1 \circ Q_2$ is defined by

$$T v \in V_1 : \quad \mathfrak{a}(T v, v_1) = \mathfrak{a}(Q_2 v, v_1) \quad \forall v_1 \in V_1 .$$

By orthogonality, since $Q_2 v \in V_2$, the right-hand side vanishes for all $v_1 \in V_2$, which implies $T v = 0$.

(ii) It is clear that $(Q_1 + Q_2)v \in V_1 + V_2$ for any $v \in V$. Besides,

$$\begin{aligned} a((Q_1 + Q_2)v, v_1) &= a(Q_1v, v_1) + \underbrace{a(Q_2v, v_1)}_{=0 \text{ by orthogonality}} = a(v, v_1) \quad \forall v_1 \in V_1, \\ a((Q_1 + Q_2)v, v_2) &= a(Q_2v, v_2) + \underbrace{a(Q_1v, v_2)}_{=0 \text{ by orthogonality}} = a(v, v_2) \quad \forall v_2 \in V_2, \end{aligned}$$

which means

$$a((Q_1 + Q_2)v, v_1 + v_2) = a(v, v_1 + v_2) \quad \forall v_1 \in V_1, v_2 \in V_2.$$

Compare this with Def. 4.2.2.2.

□

Proof. (of Thm. 4.2.2.20) Apply Lemma 4.2.2.21 repeatedly to the product formula

$$E_{SSC} = (Id - Q_M) \circ (Id - Q_{M-1}) \circ \dots \circ (Id - Q_2) \circ (Id - Q_1) \tag{4.2.2.11}$$

for the error propagation operator E_{SSC} .

$$\blacktriangleright \quad E_{SSC} = Id - \sum_{m=1}^M Q_m = Id - Id = 0,$$

owing to Lemma 4.2.2.21, Item (ii). A vanishing error propagation operator implies that already after one step of the iteration the exact solution has been reached.

□

┘

Remark 4.2.2.22 (Abstract convergence theory for SSC) Again, we rely on Ass. 4.2.2.7 and study convergence of SSC according to Def. 4.2.1.3 in the energy norm $\|\cdot\|_A$ induced by $a(\cdot, \cdot)$. The difficulty is to find the appropriate generalization of the angle between two subspaces in the case $M > 2$. This “appropriate generalization” is expressed through the assumptions of the main convergence theorem for SSC.

Theorem 4.2.2.23. Main convergence theorem for SSC

In the setting of Def. 4.2.1.3 and using Ass. 4.2.2.7, if the subspace splitting $V_h = \sum_{m=1}^M v_m$ satisfies

- the **stability estimate**

$$\exists c_0 > 0: \quad \inf \left\{ \sum_{m=1}^M \|v_m\|_A^2 : v_m \in V_m, \sum_{\ell=1}^M v_\ell = v \right\} \leq c_0 \|v\|_A^2 \quad \forall v \in V_h, \tag{4.2.2.24}$$

- and the **strengthened Cauchy-Schwarz inequalities**,

$$\exists \delta_{i,j} = \delta_{j,i} \in [0, 1]: \quad |a(v_i, v_j)| \leq \delta_{i,j} \|v_i\|_A \|v_j\|_A \quad \forall v_i \in V_i, v_j \in V_j, \quad , \tag{4.2.2.25}$$

for all $i, j \in \{1, \dots, M\}$, then

$$\|E_{SSC}\|_A \leq \sqrt{1 - \frac{1}{c_0 \lambda_{\max}(\mathbf{D})^2}}, \quad \mathbf{D} := [\delta_{i,j}]_{i,j=1}^M \in \mathbb{R}^{M,M}. \tag{4.2.2.26}$$

The proof of this main theorem requires some preparation. We introduce the auxiliary operator

$$\mathbf{T} := \sum_{m=1}^M \mathbf{Q}_m : V_h \rightarrow V_h, \quad \mathbf{Q}_m : V_h \rightarrow V_m \quad \hat{=} \text{a-orthogonal projection onto } V_m. \quad (4.2.2.27)$$

Lemma 4.2.2.28. Lower bound for \mathbf{T}

Assuming (4.2.2.24) the operator \mathbf{T} is “bounded from below”:

$$\|v\|_A^2 \leq c_0 \mathbf{a}(\mathbf{T}v, v) \quad \forall v \in V_h.$$

Proof. An elementary but central tool is the Cauchy-Schwarz inequality in \mathbb{R}^M :

$$\left| \sum_{m=1}^M \alpha_m \beta_m \right| \leq \left(\sum_{m=1}^M |\alpha_m|^2 \right)^{\frac{1}{2}} \cdot \left(\sum_{m=1}^M |\beta_m|^2 \right)^{\frac{1}{2}} \quad \forall \alpha_i, \beta_i \in \mathbb{C}. \quad (4.2.2.29)$$

Pick $v \in V_h$ and assume that the decomposition $v_h = \sum_{m=1}^M v_m, v_m \in V_m$, realizes the stability estimate (4.2.2.24).

$$\begin{aligned} \blacktriangleright \quad \|v\|_A^2 &= \sum_{m=1}^M \mathbf{a}(v_m, v) = \sum_{m=1}^M \mathbf{a}(\mathbf{Q}_m v_m, v) = \sum_{m=1}^M \mathbf{a}(v_m, \mathbf{Q}_m v) \\ &\leq \left(\sum_{m=1}^M \|v_m\|_A^2 \right)^{\frac{1}{2}} \cdot \left(\sum_{m=1}^M \|\mathbf{Q}_m v\|_A^2 \right)^{\frac{1}{2}} \\ &\leq c_0^{\frac{1}{2}} \|v\|_A \cdot \left(\sum_{m=1}^M \mathbf{a}(\mathbf{Q}_m v, v) \right)^{\frac{1}{2}} \leq c_0^{\frac{1}{2}} \|v\|_A \cdot (\mathbf{a}(\mathbf{T}v, v))^{\frac{1}{2}}, \end{aligned}$$

which, after canceling $\|v\|_A$, amounts to the assertion of the lemma. □

Note that Lemma 4.2.2.28 implies that \mathbf{T} is injective with closed range. As \mathbf{T} is also selfadjoint, we can conclude that \mathbf{T} is surjective.

Lemma 4.2.2.30. Upper bound for \mathbf{T}

If (4.2.2.24) and (4.2.2.25) hold true, the operator \mathbf{T} from (4.2.2.27) is “bounded from above”:

$$\mathbf{a}(\mathbf{T}v, v) \leq \lambda_{\max}(\mathbf{D})^2 c_0 \|v\|_A^2 \quad \forall v \in V_h.$$

Proof. Since the matrix \mathbf{D} is symmetric, we have $\|\mathbf{D}\|_2 = \lambda_{\max}(\mathbf{D})$, which implies

$$\mathbf{x}^T \mathbf{D} \mathbf{y} \leq \lambda_{\max}(\mathbf{D}) \|\mathbf{x}\|_2 \|\mathbf{y}\|_2 \quad \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^M. \quad (4.2.2.31)$$

Again, fix $v \in V_h$ and choose a splitting $v = \sum_{m=1}^M v_m$ that realizes (4.2.2.24).

$$\begin{aligned} \mathbf{a}(\mathbf{T}v, v) &= \sum_{j=1}^M \mathbf{a}(\mathbf{Q}_j v, v) = \sum_{m=1}^M \sum_{j=1}^M \mathbf{a}(\mathbf{Q}_j v, v_m) \stackrel{(4.2.2.25)}{\leq} \sum_{m=1}^M \sum_{j=1}^M \delta_{m,j} \|\mathbf{Q}_j v\|_A \|v_m\|_A \\ &\stackrel{(4.2.2.31)}{\leq} \lambda_{\max}(\mathbf{D}) \left(\sum_{j=1}^M \|\mathbf{Q}_j v\|_A^2 \right)^{\frac{1}{2}} \cdot \left(\sum_{m=1}^M \|v_m\|_A^2 \right)^{\frac{1}{2}} \leq \lambda_{\max}(\mathbf{D}) \cdot \mathbf{a}(\mathbf{T}v, v)^{\frac{1}{2}} \cdot c_0^{\frac{1}{2}} \|v\|_A. \end{aligned}$$

Cancel $a(\mathbb{T}v, v)^{\frac{1}{2}}$ and square the resulting inequality. □

Proof. (of Thm. 4.2.2.23) We define the operators

$$\mathbf{E}_i := (\text{Id} - \mathbf{Q}_i) \circ \cdots \circ (\text{Id} - \mathbf{Q}_1), \quad i = 1, \dots, M, \quad \mathbf{E}_0 := \text{Id}. \quad (4.2.2.32)$$

Note that $\mathbf{E}_M = \mathbf{E}_{\text{SSC}}$, which means that we have to find a bound for $\|\mathbf{E}_M\|_A$. From the very definition (4.2.2.32) of the \mathbf{E}_i s we infer

$$\mathbf{E}_i = (\text{Id} - \mathbf{Q}_i)\mathbf{E}_{i-1} \iff \mathbf{E}_{i-1} - \mathbf{E}_i = \mathbf{Q}_i\mathbf{E}_{i-1} \blacktriangleright \mathbf{E}_i = \text{Id} - \sum_{j=0}^i \mathbf{Q}_j\mathbf{E}_{j-1}, \quad (4.2.2.33)$$

by a telescopic sum. Another telescopic sum yields for any $v \in V_h$

$$\|\mathbf{E}_M v\|_A^2 - \|v\|_A^2 = - \sum_{j=0}^M a(\mathbf{Q}_j\mathbf{E}_{j-1}v, \mathbf{E}_{j-1}v), \quad (4.2.2.34)$$

$$\text{because } \|\mathbf{E}_i v\|_A^2 = a((\text{Id} - \mathbf{Q}_i)\mathbf{E}_{i-1}v, \mathbf{E}_{i-1}v) = \|\mathbf{E}_{i-1}v\|_A^2 - a(\mathbf{Q}_i\mathbf{E}_{i-1}v, \mathbf{E}_{i-1}v).$$

Expressing Id by means of (4.2.2.33) gives us

$$a(\mathbf{Q}_i v, v) = a(\mathbf{Q}_i v, \mathbf{E}_{i-1}v + \sum_{j=0}^{i-1} \mathbf{Q}_j\mathbf{E}_{j-1}v) = \sum_{j=0}^i a(\mathbf{Q}_i v, \mathbf{Q}_j\mathbf{E}_{j-1}v). \quad (4.2.2.35)$$

We sum up these identities for $i = 0, \dots, M$ and then invoke the the strengthened Cauchy-Schwarz inequality (4.2.2.25):

$$\begin{aligned} a(\mathbb{T}v, v) &= \sum_{i=0}^M a(\mathbf{Q}_i v, v) = \sum_{i=0}^M \sum_{j=0}^i a(\mathbf{Q}_i v, \mathbf{Q}_j\mathbf{E}_{j-1}v) \leq \sum_{i=0}^M \sum_{j=0}^i \delta_{i,j} \|\mathbf{Q}_i v\|_A \|\mathbf{Q}_j\mathbf{E}_{j-1}v\|_A \\ &\leq \lambda_{\max}(\mathbf{D}) \cdot a(\mathbb{T}v, v)^{\frac{1}{2}} \left(\sum_{j=0}^M a(\mathbf{Q}_j\mathbf{E}_{j-1}v, \mathbf{E}_{j-1}v) \right)^{\frac{1}{2}} \\ &\blacktriangleright a(\mathbb{T}v, v) \leq \lambda_{\max}(\mathbf{D})^2 \cdot \sum_{j=0}^M a(\mathbf{Q}_j\mathbf{E}_{j-1}v, \mathbf{E}_{j-1}v). \end{aligned} \quad (4.2.2.36)$$

As if by magic, the term from (??) has popped up! Thus, (4.2.2.36) implies

$$a(\mathbb{T}v, v) \leq \lambda_{\max}(\mathbf{D})^2 \cdot (\|v\|_A^2 - \|\mathbf{E}_M v\|_A^2) \Rightarrow \|\mathbf{E}_M v\|_A^2 \leq \|v\|_A^2 - \lambda_{\max}(\mathbf{D})^{-2} a(\mathbb{T}v, v).$$

In the last step we resort to the estimate $\|v\|_A^2 \leq c_0 a(\mathbb{T}v, v)$ of Lemma 4.2.2.28 and get

$$\|\mathbf{E}_M v\|_A^2 \leq \left(1 - \frac{1}{c_0 \lambda_{\max}(\mathbf{D})^2} \right) \|v\|_A^2,$$

which is the assertion of the theorem. □

□

┘

4.2.3 Coarse-Grid Correction (CGC)

Now we discuss a remedy for the failure of the Gauss-Seidel iteration from Code 4.1.3.3 to reduce smooth/long-range error components effectively. This remedy is suggested by the subspace correction interpretation of the Gauss-Seidel method elaborated in § 4.2.1.15.



Idea: Augment the subspace splitting $V_h = \sum_{m=1}^N \text{Span}\{b_h^m\}$ defining the Gauss-Seidel iteration by another subspace $V_H \subset V_h$ capable of representing smooth functions with global support.

Of course, the dimension of this extra subspace must not be too large, in order to keep the cost of computing the subspace correction affordable.



Idea: Choose V_H as finite element space on a coarse mesh \mathcal{M}_H of the computational domain Ω with significantly fewer cells than \mathcal{M} , e.g., $V_H := \mathcal{S}_{1,0}^0(\mathcal{M}_H)$.



For unrelated $\mathcal{M}, \mathcal{M}_H$ the requirement $V_H \subset V_h$ will not be met in general.

Fortunately, [NumPDE § 3.1.4.2] discusses a special situation, in which $V_H \subset V_h$ is guaranteed for Lagrangian finite elements: the case of nested meshes.

Definition 4.2.3.1. Nested finite element meshes

Two finite element meshes $\mathcal{M}_h, \mathcal{M}_H$ (\rightarrow [NumPDE Def. 2.5.1.1]) of a computational domain $\Omega \subset \mathbb{R}^d$ are nested, $\mathcal{M}_H \prec \mathcal{M}_h$, if every (closed) cell of \mathcal{M}_H is the union of closed cells of \mathcal{M}_h .

Lemma 4.2.3.2. Nesting of meshes implies nesting of finite element spaces

In the case of nested meshes $\mathcal{M}_H \prec \mathcal{M}_h$ we have $\mathcal{S}_{1,0}^0(\mathcal{M}_H) \subset \mathcal{S}_{1,0}^0(\mathcal{M}_h)$.

Proof. The assertion is immediate from the definition [NumPDE Def. 2.6.1.1] of the Lagrangian finite element space $\mathcal{S}_1^0(\mathcal{M})$: thanks to the nesting property $\mathcal{M}_H \prec \mathcal{M}_h$ every function in $\mathcal{S}_{1,0}^0(\mathcal{M}_H)$ is affine linear on every cell of \mathcal{M}_h . Continuity and boundary conditions are immediate. \square

On pairs of nested meshes we can thus defined an enhanced Gauss-Seidel method supplemented with a so-called coarse grid correction. The resulting subspace correction method is known as two-grid iteration.

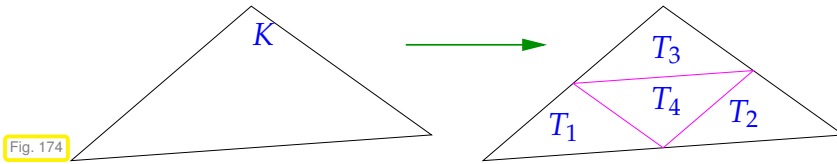
Two-grid iteration

The two-grid method based on nested meshes $\mathcal{M}_H \prec \mathcal{M}_h$ carrying nested finite element spaces $V_H \subset V_h$ is the successive subspace correction method according to Def. 4.2.1.3 using the subspace decomposition

$$V_h = \sum_{j=1}^N \text{Span}\{b_h^j\} + V_H, \tag{4.2.3.4}$$

where $\{b_h^1, \dots, b_h^N\}$, $N := \dim V_h$, is the nodal basis of V_h .

In finite element applications nested meshes are usually generated by means of local or global refinement.



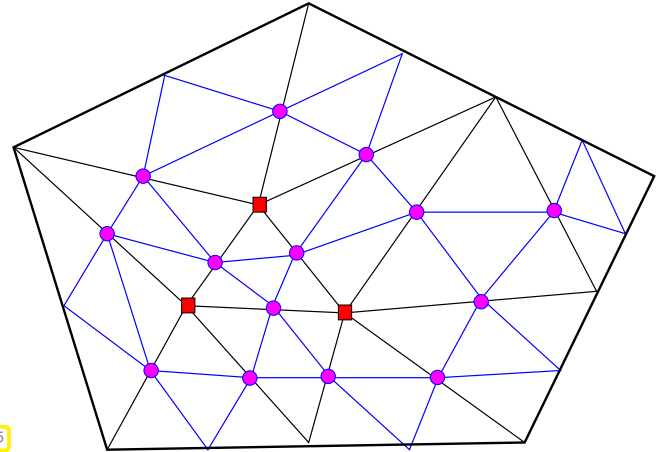
We focus on the global regular refinement of 2D triangular meshes as achieved by splitting every triangle into four smaller ones, see figure beside.

◁ Regular refinement of triangle K into four congruent triangles T_1, T_2, T_3, T_4

Two nested triangular mesh created by uniform regular refinement

- : edges of coarse mesh \mathcal{M}_H
 - : new edges of fine mesh \mathcal{M}_h
 - : interior nodes of coarse mesh \mathcal{M}_H
 - : new interior nodes of fine mesh \mathcal{M}_h
- $\dim \mathcal{S}_{1,0}^0(\mathcal{M}_H) = 3,$
 - $\dim \mathcal{S}_{1,0}^0(\mathcal{M}_h) = 17,$

Fig. 175



For two given nested triangular meshes $\mathcal{M}_H \prec \mathcal{M}_h$ with associated linear Lagrangian finite element spaces $V_h := \mathcal{S}_{1,0}^0(\mathcal{M}_h)$ and $V_H := \mathcal{S}_{1,0}^0(\mathcal{M}_H)$ we now explain the computation of the so-called **prolongation matrix** $\mathbf{P}_H \in \mathbb{R}^{N,N_H}$, $N := \dim V_h$, $N_H := \dim V_H$ with respect to the nodal bases $\{b_h^1, \dots, b_h^N\}$ and $\{b_H^1, \dots, b_H^{N_H}\}$ of V_h and V_H , respectively. Remember that \mathbf{P}_H is a basis transformation matrix and, thus, the entries of \mathbf{P}_H are defined by the relationship

$$b_H^i = \sum_{j=1}^N (\mathbf{P}_H)_{j,i} b_h^j, \quad i = 1, \dots, N_H. \tag{4.2.3.5}$$

We number the interior nodes/vertices of meshes:

- $\{x_h^1, \dots, x_h^N\} \hat{=}$ interior nodes of the fine mesh \mathcal{M}_h ,
- $\{x_H^1, \dots, x_H^{N_H}\} \hat{=}$ interior nodes of the coarse mesh \mathcal{M}_H .

Since the nodal basis functions are one-on-one associated with interior nodes, we assume that the numbering of both matches. Therefore,

$$b_h^i(x_h^j) = \delta_{i,j}, \quad i, j \in \{1, \dots, N\}, \quad b_H^i(x_H^j) = \delta_{i,j} \quad i, j \in \{1, \dots, N_H\}. \tag{4.2.3.6}$$

From this **cardinal basis property** we conclude for the prolongation matrix

$$(\mathbf{P}_H)_{j,i} = b_H^i(x_h^j), \quad 1 \leq i \leq N_H, \quad 1 \leq j \leq N. \tag{4.2.3.7}$$

Notice that the new nodes of \mathcal{M}_h , those that do not coincide with nodes of \mathcal{M}_H are midpoints of edges of \mathcal{M}_H , see Fig. 175. The function b_H^i is linear on all edges of the coarse mesh and attains the value $\frac{1}{2}$ at all midpoints of edges adjacent to x_H^i . From this observation and (4.2.3.6) we infer

$$(\mathbf{P}_H)_{i,j} = \begin{cases} 1 & , \text{ if } x_h^j = x_H^i, \\ \frac{1}{2} & , \text{ if } x_h^j \text{ is midpoint of an edge of } \mathcal{M}_H \text{ adjacent to } x_H^i, \\ 0 & , \text{ otherwise,} \end{cases} \quad \begin{matrix} 1 \leq i \leq N_H, \\ 1 \leq j \leq N. \end{matrix} \tag{4.2.3.8}$$

EXAMPLE 4.2.3.9 (A concrete basis transformation matrix)

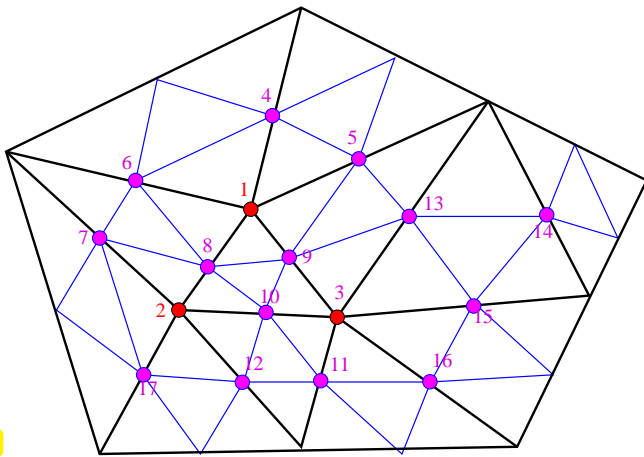


Fig. 176

We examine the two nested meshes $\mathcal{M}_H \prec \mathcal{M}_h$ sketched beside, see also Fig. 175.

The interior nodes of both meshes are numbered as indicated, with the coinciding nodes numbered first on the fine mesh.

We use piecewise linear Lagrangian finite elements on both meshes: $V_h := \mathcal{S}_{1,0}^0(\mathcal{M}_h)$, $V_H := \mathcal{S}_{1,0}^0(\mathcal{M}_H)$.

According to the rule (4.2.3.8), we have

$$\mathbf{P}_H^\top = \begin{bmatrix} 1 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 \end{bmatrix} \in \mathbb{R}^{3,17}. \quad (4.2.3.10)$$

┘

Evidently, the prolongation matrix \mathbf{P}_H is a **sparse matrix**, with important consequences:

Applying the basis transformation matrix

The asymptotic cost of multiplying a vector with \mathbf{P}_H or \mathbf{P}_H^\top is $O(N)$ for $N \rightarrow \infty$.

Assuming that the basis transform matrix \mathbf{P}_H is available, the two-grid iteration for solving the linear system of equations $\mathbf{A}\vec{\mu} = \vec{\varphi}$ can be implemented as follows on the algebraic level:

Pseudocode 4.2.3.12: Two-grid iteration algorithm

```

1 void two_grid_iteration(const Matrix A ∈ ℝN,N, const Vector φ ∈ ℝN,
2                       ref Vector μ, double TOL) {
3   AH := PH⊤A PH; // build Galerkin matrix on MH
4   do {
5     μold := μ;
6     for (i=1 ; i<N ; i++) { // Inner Gauss-Seidel loop
7       (μ)i = 1 / (A)ii ( (φ)i - ∑j=1, j≠iN (A)ij (μ)j );
8     } //
9     ρh := φ - A μ; // Residual vector ∈ ℝN
10    ρH := PH⊤ ρh; // Residual vector ∈ ℝNH by restriction
11    Solve AH vH = ρH; // Correction in VH
12    μ ← μ + PH vH; // Prolongation and update of approximate solution
13  }
14  while ( ||μ - μold|| > TOL · ||μ|| ); // Termination test
15 }
```

Here, the argument $\vec{\mu}$ both passes the initial guess and serves as variable to return the final approximate solution. As has already been mentioned, the operations in lines 9–8 of Code 4.2.3.12 are usually called **coarse-grid correction**. The Gauss-Seidel loop comprising lines 6–8 is often dubbed the **smoother**.

What is implemented in Code 4.2.3.12 is **pre-smoothing**, because the smoother comes before the coarse-grid correction. Of course, the coarse-grid correction and the smoother can also be swapped and this will result in **post-smoothing**.

A simple inspection of the algorithm reveals its computational cost:

Cost of two-grid method

Apart from solving the linear system $\mathbf{A}_H \vec{v}_H = \vec{\rho}_H$ the asymptotic computational cost of the two-grid method from Code 4.2.3.12 is $O(N)$ provided that \mathbf{A} is a sparse finite element matrix.

Moreover, the two grid method is the composition in the sense of § 4.1.3.25 of the Gauss-Seidel iteration and a subspace correction in the direction of V_H . Hence, from (4.1.3.28) and (4.1.3.7) we draw the following conclusions:

Corollary 4.2.3.14. Two-grid method as stationary linear iteration

The two-grid method from Code 4.2.3.12 is a **stationary linear iteration** with error propagation matrix

$$\mathbf{E}_{\text{TGM}} = (\mathbf{I} - \mathbf{P}_H \mathbf{A}_H^{-1} \mathbf{P}^T \mathbf{A})(\mathbf{I} - \text{tril}(\mathbf{A})^{-1} \mathbf{A}) . \tag{4.2.3.15}$$

EXPERIMENT 4.2.3.16 (Two-grid method for the Poisson matrix)

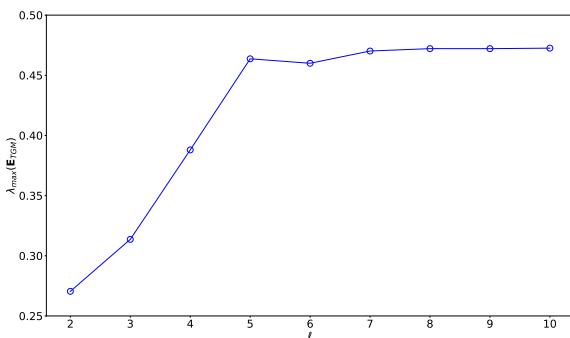


Fig. 177

Apply the two-grid method to the Poisson matrix $\mathbf{A} \in \mathbb{R}^{N,N}$, $N = (n - 1)^2$, from (4.1.1.23).

We investigate the matrix sizes $N := (n - 1)^2$, $n = 2^\ell$, $\ell = 2, \dots, 10$, and compute a guess for the rate of linear convergence by means of the power iteration with $\text{TOL} = 10^{-3}$.



In sharp contrast to the behavior of the Gauss-Seidel and CG iterations, the convergence of the two-grid method does not deteriorate on fine meshes; it is **h-uniform**.

┘

4.2.4 Multigrid Iteration



The coarse grid linear system $\mathbf{A}_H \vec{v}_H = \vec{\rho}_H$ may still be too big for direct elimination solvers.



Idea: (**Recursion**)

If also \mathcal{M}_H arises from refining an even coarser mesh, iteratively solve $\mathbf{A}_H \vec{v}_H = \vec{\rho}_H$ approximately by another two-grid iteration.

Assumption 4.2.4.1. Mesh hierarchy

We assume that a hierarchy of nested meshes

$$\mathcal{M}_0 \prec \mathcal{M}_1 \prec \dots \prec \mathcal{M}_L, \quad L \in \mathbb{N},$$

is available.

The subscript ℓ of \mathcal{M}_ℓ is called the **level** of a mesh.

This gives us a sequence of nested finite element spaces

$$V_0 \subset V_\ell \subset \dots \subset V_h := V_L, \quad \text{e.g.,} \quad V_\ell := \mathcal{S}_{1,0}^0(\mathcal{M}_\ell). \quad (4.2.4.2)$$

All these spaces are equipped with (nodal) finite element bases:

$$V_\ell = \text{Span}\{b_\ell^1, \dots, b_\ell^{N_\ell}\}, \quad N_\ell := \dim V_\ell. \quad (4.2.4.3)$$

This fixes the finite element Galerkin matrices $\mathbf{A}_\ell \in \mathbb{R}^{N_\ell \times N_\ell}$ for all levels $\ell = 0, \dots, L$. We can also compute the prolongation matrices $\mathbf{P}_{\ell-1, \ell} \in \mathbb{R}^{N_\ell \times N_{\ell-1}}$ through, cf. (4.2.3.5)

$$b_{\ell-1}^i = \sum_{j=1}^{N_\ell} (\mathbf{P}_{\ell-1, \ell})_{j,i} b_\ell^j. \quad (4.2.4.4)$$

At this point we have all ingredients ready for the **(geometric) multigrid iteration**, whose **recursive implementation** is given next:

Pseudocode 4.2.4.5: Multigrid iteration: recursive algorithm (adaptive cycle)

```

1 void multi_grid_iteration(const Vector  $\vec{\varphi} \in \mathbb{R}^{N_\ell}$ , ref Vector  $\vec{\mu}$ ,
2                           int  $\ell$ , double TOL, int max_n_steps) {
3   if ( $\ell == 0$ ) { Directly solve  $\mathbf{A}_0 \vec{\mu} = \vec{\varphi}$ ; }
4   else {
5     for (nsteps = 0; nsteps < max_n_steps; nsteps++) {
6        $\vec{\mu}_{\text{old}} := \vec{\mu}$ ;
7        $\vec{\mu} \leftarrow \vec{\mu} + \text{tril}(\mathbf{A}_\ell)^{-1}(\vec{\varphi} - \mathbf{A}_\ell \vec{\mu})$ ; // Gauss-Seidel step, pre-smoothing
8        $\vec{\rho}_h := \vec{\varphi} - \mathbf{A}_\ell \vec{\mu}$ ; // Residual vector  $\in \mathbb{R}^{N_\ell}$ 
9        $\vec{\rho}_H := \mathbf{P}_{\ell-1, \ell}^\top \vec{\rho}_h$ ; // Residual vector  $\in \mathbb{R}^{N_{\ell-1}}$ 
10       $\vec{v}_H := \mathbf{0}$ ; // Natural initial guess for correction
11      multi_grid_iteration( $\vec{\rho}_H$ ,  $\vec{v}_H$ ,  $\ell-1$ , 0.0, 1); // Recursion
12       $\vec{\mu} \leftarrow \vec{\mu} + \mathbf{P}_{\ell-1, \ell} \vec{v}_H$ ; // Update approximate solution
13      if ( $\|\vec{\mu} - \vec{\mu}_{\text{old}}\|_A \leq \text{TOL} \cdot \|\vec{\mu}\|_A$ ) break; // Termination test
14    }
15    error("No convergence");
16  }
17 }

```

The algorithm assumes that all Galerkin matrices $\mathbf{A}_\ell \in \mathbb{R}^{N_\ell \times N_\ell}$ on all levels $\ell = 1, \dots, L$, have been precomputed. Again, the code in lines 8–12 represents the **coarse-grid correction** and Line 7 is a compact way to express Gauss-Seidel **pre-smoothing**. The corresponding variant with post-smoothing should be clear.

In practice, one prefers to apply both pre- and post-smoothing together and in a symmetric fashion. In Code 4.2.4.5 this can be realized by inserting the backward Gauss-Seidel smoothing step

$$\vec{\mu} \leftarrow \vec{\mu} + \text{triu}(\mathbf{A})^{-1}(\vec{\varphi} - \mathbf{A}\vec{\mu}),$$

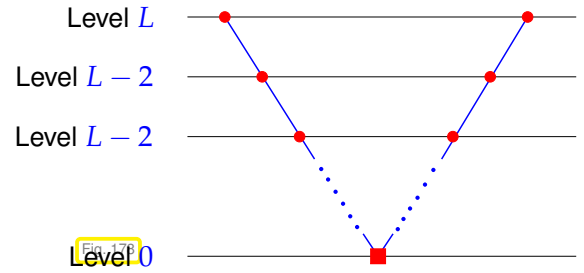
after Line 12. Here `triu` designates the upper triangular part of the matrix A .

Remark 4.2.4.6 (Multigrid cycles)

Note that only that in Code 4.2.4.5 coarse grid correction make s use of only a single multigrid iteration. This is called a **multigrid V-cycle**.

The name is due to a popular visualization of the control flow in the form of a 'V', descending down to the coarsest mesh and ascending on the way back. ▷

In the figure ● stands for an application of a pre-/post-smoother, ■ designates the use of a direct solver.



In the '1' in Line 11 of Code 4.2.4.5 is replaced with a '2', the resulting scheme is known as **multigrid W-cycle**. ┘

§4.2.4.7 (Cost of multigrid iteration) Let us supplement Ass. 4.2.4.1 with the additional requirement that the number of cells on coarser meshes decreases geometrically

$$\#\mathcal{M}_{\ell-1} = q\#\mathcal{M}_{\ell} \text{ for } 0 < q < 1, \ell = 1, \dots, L. \tag{4.2.4.8}$$

This is the case, e.g., if the sequence of nested meshes $\mathcal{M}_0 \prec \mathcal{M}_1 \prec \dots \prec \mathcal{M}_L$ is generated by repeated global regular refinement, recall Section 4.2.3. In 2D in this case we obtain $q = \frac{1}{4}$. A consequence of (4.2.4.8) is that

$$N_{\ell} := \dim \mathcal{S}_{1,0}^0(\mathcal{M}_{\ell}) \approx q^{-\ell} N_0, \ell = 1, \dots, L. \tag{4.2.4.9}$$



As we have already noted, apart from Line 11 the cost of a function call in Code 4.2.4.5 is $\approx N_{\ell}$. Summing the geometric series, we conclude that the total cost for all recursive calls of `multi_grid_iteration()` is $O(N_L)$!

Remark 4.2.4.10 (Multigrid iteration as successive subspace correction method) It was a major discovery that the complete multigrid iteration as implemented in Code 4.2.4.5 is a genuine successive subspace correction method according to Def. 4.2.1.3, see [TOS00, Appendix B].

Theorem 4.2.4.11. Multigrid = multi-level subspace correction [Xu92]

The multigrid iteration from Code 4.2.4.5 with `max_n_steps=1` is a successive subspace correction method based on the space decomposition

$$V_h = V_0 + \sum_{\ell=1}^L \sum_{j=1}^{N_{\ell}} \text{Span}\{b_{\ell}^j\}. \tag{4.2.4.12}$$

This interpretation of the geometric multigrid method made it possible to establish h -uniform convergence for finite element linear systems.

Theorem 4.2.4.13. Convergence of geometric multigrid [BY93]

Consider the Galerkin discretization of (4.1.1.8) by means of linear Lagrangian finite elements. Let the multigrid iteration from Code 4.2.4.5 with `max_n_steps=1` be based on a uniformly shape-regular and quasi-uniform family of nested triangular meshes. Then the energy operator norm of the error propagation operator of the multigrid iteration is bounded by a constant $0 < \rho < 1$ that depends only on the shape-regularity and quasi-uniformity of the meshes and the coefficient functions A and γ .

In particular, geometric multigrid enjoys a rate of linear convergence, which does not depend on the number L of levels involved. ┘

§4.2.4.14 (Nested iteration (NI)) One crucial issue remains: How do we choose the initial guess?



Idea: (Recursion) Use “low-accuracy” solution obtained by multigrid iteration on next coarser level as initial guess.

This policy is known as **nested iteration** and a **recursive implementation** is given next. Again, the Galerkin matrices $\mathbf{A}_\ell \in \mathbb{R}^{N_\ell, N_\ell}$ are supposed to be available.

Pseudocode 4.2.4.15: Nested multigrid iteration: recursive algorithm

```

1   $\mathbb{R}^{N_\ell}$ -vector mg_solve(const  $\vec{\varphi} \in \mathbb{R}^{N_\ell}$ , int  $\ell$ , double TOL, int max_n_steps) {
2      if ( $\ell == 0$ ) { Directly solve  $\mathbf{A}_0 \vec{\mu} = \vec{\varphi}$ ; }
3      else {
4           $\vec{\varphi}_H := \mathbf{P}_{\ell-1, \ell}^\top \vec{\varphi}$ ;
5           $\vec{\mu}_H :=$  mg_solve( $\vec{\varphi}_H$ ,  $\ell - 1$ ,  $\rho \cdot \text{TOL}$ , max_n_steps);
6           $\vec{\mu}_h := \mathbf{P}_{\ell-1, \ell} \vec{\mu}_H$ ; // Transfer solution to the next level
7          multi_grid_iteration( $\vec{\varphi}$ ,  $\vec{\mu}$ ,  $\ell$ , TOL, max_n_steps); // Code Code 4.2.4.5
8      }
9      return ( $\vec{\mu}$ );
10 }
```

Here the factor $\rho > 1$ takes into account that on coarser meshes we expect a larger discretization error, which justifies relaxed accuracy requirements there. The concrete choice of ρ can be guided by asymptotic a-priori error estimates for finite element Galerkin solutions: If we expect an asymptotic convergence like $O(h_{\mathcal{M}}^\alpha)$ for some $\alpha > 0$ in a norm of interest, and assume regular global refinement, then choosing $\rho = 2^\alpha$ is the proper value. ┘

§4.2.4.16 (Nested iteration with fixed number of iterations) We consider the $\mathcal{S}_1^0(\mathcal{M})$ -finite element Galerkin discretization of a 2nd-order elliptic boundary value problem on $\Omega \subset \mathbb{R}^2$, cf. the model problems discussed in Section 4.1.1. A hierarchy of nested meshes finite-element meshes $\mathcal{M}_0 \prec \mathcal{M}_1 \prec \dots \prec \mathcal{M}_L$, $L \in \mathbb{N}$, created by **uniform regular refinement** [NumPDE Ex. 3.1.4.3], Fig. 174, is supposed to be available. In particular, for the meshwidths h_ℓ of these meshes holds

$$h_\ell = \frac{1}{2} h_{\ell-1}, \quad \ell = 1, \dots, L. \quad (4.2.4.17)$$

We write

- $u \in H^1(\Omega)$ for the exact (weak) solution,
- $u_\ell \in \mathcal{S}_1^0(\mathcal{M}_\ell)$ for the finite-element Galerkin solution on \mathcal{M}_ℓ .

We assume that $u \in H^2(\Omega)$, so that [NumPDE Thm. 3.3.5.6] plus the quasi-optimality of Galerkin solutions in energy norm [NumPDE Thm. 3.1.3.7] implies the a priori finite-element error estimate ($h_\ell \hat{=}$ mesh width of \mathcal{M}_ℓ)

$$\exists C > 0: \|u - u_\ell\|_a \leq C h_\ell \|u\|_{H^2(\Omega)} \quad \forall \ell \in \{0, \dots, L\}, \quad (4.2.4.18)$$

where \cdot_a is the energy norm induced by the bilinear form of the variational problem [NumPDE Section 3.1.1]. If C is chosen as small as possible, the estimates (4.2.4.18) will be rather sharp.

Assumption 4.2.4.19. ℓ -uniform linear convergence of multigrid V-cycle

The multigrid V-cycle for solving $\mathbf{A}_\ell \vec{\mu}_\ell = \vec{\varphi}_\ell$, implemented in `multi_grid_iteration()` in Code 4.2.4.5, converges linearly in energy norm with rate $\rho < 1$ for all ℓ .

Consequently, the function call

multi_grid_iteration($\vec{\varphi}_\ell, \vec{\mu}_\ell^{(0)}, \ell, 0.0, 1$)

will overwrite $\vec{\mu}_\ell^{(0)} \in \mathbb{R}^{N_\ell}$ with a vector $\vec{\mu}_\ell^{(1)} \in \mathbb{R}^{N_\ell}$ satisfying

$$\|u_\ell^{(1)} - u_\ell\|_a = \|\vec{\mu}_\ell^{(1)} - \vec{\mu}_\ell\|_{A_\ell} \leq \rho \|\vec{\mu}_\ell^{(0)} - \vec{\mu}_\ell\|_{A_\ell} = \|u_\ell^{(0)} - u_\ell\|_a. \quad (4.2.4.20)$$

Here, the basis expansion coefficient vectors $\vec{\mu}_\ell, \vec{\mu}_\ell^{(0)}, \vec{\mu}_\ell^{(1)}$ belong to the functions $u_\ell, u_\ell^{(0)}, u_\ell^{(1)} \in \mathcal{S}_1^0(\mathcal{M})$. In light of this correspondence, a “function-centered” perspective is adopted below.

We employ `multi_grid_iteration()`-based nested iteration with a fixed number m of multigrid V-cycles as iterative solvers on each level.

Pseudocode 4.2.4.21: Fixed-steps nested multigrid iteration

```

1   $\mathbb{R}^{N_L}$ -vector nested_mg_fixed(int m) {
2    Solve  $\mathbf{A}_0 \vec{\mu}_0 = \vec{\varphi}_0$ ;
3    for (int l = 1; l <= L; ++l) {
4       $\vec{\mu}_l := \mathbf{P}_{l-1, l} \vec{\mu}_{l-1}$ ; // Transfer solution to the next level
5      for (int k = 0; k < m; ++k) {
6        multi_grid_iteration( $\vec{\varphi}_l, \vec{\mu}_l, l, 0.0, 1$ ); // Multigrid V-cycle SLI
7      }
8    }
9    return ( $\vec{\mu}_L$ );
10 }
```

(The $\mathcal{S}_1^0(\mathcal{M}_\ell)$ Galerkin matrices \mathbf{A}_ℓ , right-hand side vectors $\vec{\varphi}_\ell$, and prolongation matrices $\mathbf{P}_{\ell-1, \ell}$ are assumed to be available.)

We choose

$$m := \lceil -\frac{\log 4}{\log \rho} \rceil \iff \rho^m \leq \frac{1}{4}. \quad (4.2.4.22)$$

From (4.2.4.18) we conclude

$$\|u_0 - u\|_a \leq Ch_0 \|u\|_{H^2(\Omega)}. \quad (4.2.4.23)$$

Since u_0 is the initial guess $u_1^{(0)}$ on level 1, by the triangle inequality

$$\begin{aligned} \|u_1^{(m)} - u_1\|_a &\leq \frac{1}{4} \|u_1^{(0)} - u_1\|_a \leq \frac{1}{4} (\|u_0 - u\|_a + \|u_1 - u\|_a) \\ &\stackrel{(4.2.4.18)}{\leq} \frac{1}{4} (Ch_0 + Ch_1) \|u\|_{H^2(\Omega)} \leq \frac{3}{4} Ch_1 \|u\|_{H^2(\Omega)}. \end{aligned} \quad (4.2.4.24)$$

Induction in ℓ based on (4.2.4.24) with $1 \rightarrow \ell, 0 \rightarrow \ell - 1$, leads to the estimate

$$\begin{aligned} \|u_\ell^{(m)} - u_\ell\|_a &\leq \frac{1}{4} \|u_\ell^{(0)} - u_\ell\|_a \leq \frac{1}{4} (\|u_{\ell-1} - u\|_a + \|u_\ell - u\|_a) \\ &\stackrel{(4.2.4.18)}{\leq} \frac{1}{4} (Ch_{\ell-1} + Ch_\ell) \|u\|_{H^2(\Omega)} \leq Ch_\ell \|u\|_{H^2(\Omega)}, \end{aligned} \quad \ell = 1, \dots, L. \quad (4.2.4.25)$$

Under Ass. 4.2.4.19 and choosing m according to (4.2.4.22), fixed-step nested multigrid iteration achieves an iteration/solver error smaller than the discretization error (in energy norm)

$$\|u_L^{(m)} - u_L\|_a \lesssim \|u_L - u\|_a.$$

This leaves nothing wanting, because any further reduction of the iteration/solver error would not gain anything in terms of overall accuracy.

Since the dimensions of the finite element spaces $\mathcal{S}_1^0(\mathcal{M}_\ell)$ decrease exponentially as ℓ decreases and the cost of `multi_grid_iteration()` on level ℓ is proportional to $N_\ell := \dim \mathcal{S}_1^0(\mathcal{M}_\ell)$ (\rightarrow § 4.2.4.7), a geometric sum argument shows that

$$\text{cost}(\text{nested_mg_fixed}()) = O(N_L), \quad N_L := \dim \mathcal{S}_1^0(\mathcal{M}_L), \quad \text{for } L \rightarrow \infty. \quad (4.2.4.26)$$

┘

4.2.5 Multigrid Preconditioning

§4.2.5.1 (Preconditioned conjugate gradient method (PCG)) In § 4.1.4.13 we saw that the conjugate gradient (CG) iterative solvers is haunted by a similar degradation of performance for large finite element linear systems $\mathbf{A}\vec{\mu} = \vec{\phi}$, $\mathbf{A} \in \mathbb{R}^{N,N}$ s.p.d., as the Gauss-Seidel method. Fortunately, there is a powerful technique for accelerating the convergence of CG known as **preconditioning**, cf. [NumCSE Section 10.3]. It relies on the availability of a linear operator $\mathbb{R}^N \rightarrow \mathbb{R}^N$, henceforth incarnated by an s.p.d. matrix $\mathbf{B} \in \mathbb{R}^{N,N}$. The resulting algorithm for the **preconditioned conjugate gradient method (PCG)** is given next.

Pseudocode 4.2.5.2: PCG method

```

1 void pcg(A ∈ ℝN,N, φ ∈ ℝN, ref μ̄,
2         B ∈ ℝN,N, double TOL) {
3   r̄ := φ - Aμ̄; // Residual vector
4   π := B r̄; η̄ := π; τ₀ = πT r̄;
5   for( j=1; j < maxit; ++j) {
6     β := r̄T η̄;
7     γ̄ := A η̄;
8     α := β / πT γ̄;
9     μ̄ ← μ̄ + α π; // update solution
10    r̄ ← r̄ - α γ̄; // update residual
11    η̄ ← B r̄; // Apply preconditioner
12    β ← r̄T η̄ / β;
13    if (|η̄T r̄| < TOL · τ₀) break;
14    π ← η̄ + β π;
15  }
16 }
```

◁ Preconditioned conjugate gradient method for solving $\mathbf{A}\vec{\mu} = \vec{\phi}$ with preconditioner \mathbf{B} . ($\vec{\mu}$ passes the initial guess and also returns the result.)

Computational effort per step:

- One $\mathbf{A} \times$ vector operation
- One $\mathbf{A} \times$ vector operation
- 3 dot products
- 3 AXPY operations

▶ PCG requires only the application of the linear operators described by \mathbf{A} and \mathbf{B} to a vector.

Cost of PCG step

If \mathbf{A} and \mathbf{B} are sparse matrices with “ $O(N)$ number of non-zero entries”, then the computational cost per PCG step is $O(N)$ for $N \rightarrow \infty$.

The assertion of Thm. 4.1.4.11 remains valid for PCG, provided that $\kappa(\mathbf{A})$ is replaced with $\kappa(\mathbf{BA})$:

Theorem 4.2.5.4. Convergence of the PCG method [NumCSE Thm. 10.2.3.5]

The iterates of the PCG method with preconditioner $\mathbf{B} \in \mathbb{R}^{N,N}$ for solving $\mathbf{A}\vec{\mu} = \vec{\varphi}$ (see Code 4.2.5.2) with $\mathbf{A} = \mathbf{A}^\top, \mathbf{B} = \mathbf{B}^\top \in \mathbb{R}^{N,N}$ s.p.d. satisfy

$$\|\vec{\mu}^* - \vec{\mu}^{(l)}\|_{\mathbf{A}} \leq 2 \left(\frac{\sqrt{\kappa(\mathbf{BA})} - 1}{\sqrt{\kappa(\mathbf{BA})} + 1} \right)^l \|\vec{\mu}^* - \vec{\mu}^{(0)}\|_{\mathbf{A}}, \quad l \in \mathbb{N},$$

where $\mathbf{A}\vec{\mu}^* = \vec{\varphi}$.

Summing up, a good preconditioner \mathbf{B} must satisfy that

- (I) \mathbf{B} is symmetric and positive definite,
- (II) the cost of $\mathbf{B} \times$ vector is proportional to N , and
- (III) the spectral condition number $\kappa(\mathbf{BA})$ is small independently of N .

┘

How to build preconditioners? The good news is that stationary linear iterations for solving $\mathbf{A}\vec{\mu} = \vec{\varphi}$, $\mathbf{A} \in \mathbb{R}^{N,N}$

$$\vec{\mu}^{(k+1)} = \vec{\mu}^{(k)} + \mathbf{M}(\vec{\varphi} - \mathbf{A}\vec{\mu}^{(k)}), \quad \mathbf{M} = \mathbf{M}^\top \in \mathbb{R}^{N,N} \text{ regular,} \quad (4.1.3.13)$$

are a source for preconditioners:

Theorem 4.2.5.5. Preconditioners from stationary linear iterations

If the stationary linear iteration (4.1.3.13) enjoys an asymptotic rate of convergence $\rho < 1$, then

$$\kappa(\mathbf{MA}) \leq \frac{1 + \rho}{1 - \rho}.$$

Proof. As explained in § 4.1.3.12 we have $\lambda_{\max}(\mathbf{I} - \mathbf{MA}) \leq \rho$, which implies

$$\begin{aligned} |1 - \lambda| &\leq \rho \\ \iff & \text{for all eigenvalues } \lambda \in \sigma(\mathbf{MA}). \\ 1 - \rho &\leq \lambda \leq 1 + \rho \end{aligned}$$

The claim follows from the definition of $\kappa(\mathbf{MA}) := \lambda_{\max}(\mathbf{MA})\lambda_{\min}^{-1}(\mathbf{MA})$. □

Thus, the stationary linear iteration induced by the multigrid method is a promising candidate for a preconditioner, provided that it supplies a symmetric \mathbf{M} ! Thm. 4.2.4.11 together with the following lemma tell us how to achieve this.

Lemma 4.2.5.6. Symmetric successive subspace correction

The error propagation matrix \mathbf{E}_{SSC} of a successive subspace correction method according to Def. 4.2.1.3 satisfies

$$\mathbf{A}\mathbf{E}_{\text{SSC}} = \mathbf{E}_{\text{SSC}}^\top \mathbf{A},$$

if $\mathbf{A} = \mathbf{A}^\top$ and $V_{M-j+1} = V_j, j = 1, \dots, M$.

Proof. Using (4.2.1.14) we conclude

$$\begin{aligned} \mathbf{E}^\top \mathbf{A} &= (\mathbf{I} - \mathbf{A}^\top \mathbf{P}_1 (\mathbf{P}_1^\top \mathbf{A} \mathbf{P}_1)^{-1} \mathbf{P}_1^\top \mathbf{A}) \cdots (\mathbf{I} - \mathbf{A}^\top \mathbf{P}_m (\mathbf{P}_m^\top \mathbf{A} \mathbf{P}_m)^{-1} \mathbf{P}_m^\top \mathbf{A}) \\ &= (\mathbf{I} - \mathbf{A}^\top \mathbf{P}_m (\mathbf{P}_m^\top \mathbf{A} \mathbf{P}_m)^{-1} \mathbf{P}_m^\top \mathbf{A}) \cdots (\mathbf{I} - \mathbf{A}^\top \mathbf{P}_1 (\mathbf{P}_1^\top \mathbf{A} \mathbf{P}_1)^{-1} \mathbf{P}_1^\top \mathbf{A}) \\ &= \mathbf{A} \mathbf{E}, \end{aligned}$$

because $\mathbf{A} = \mathbf{A}^\top$. □

If the assumptions of the lemma are satisfied, we have

$$\mathbf{M}_{\text{SSC}}^\top = \mathbf{A}^{-1} (\mathbf{I} - \mathbf{E}_{\text{SSC}}^\top) = (\mathbf{I} - \mathbf{A}^{-1} \mathbf{E}_{\text{SSC}}^\top \mathbf{A}) \mathbf{A}^{-1} \stackrel{\text{Lemma 4.2.5.6}}{=} (\mathbf{I} - \mathbf{A}^{-1} \mathbf{A} \mathbf{E}_{\text{SSC}}) \mathbf{A}^{-1} = \mathbf{M}_{\text{SSC}}.$$

The symmetry of the subspace splitting, $V_{M-j+1} = V_j, j = 1, \dots, M$, can be ensured by using symmetric pre- and post-smoothing steps, that is, we employ Gauss-Seidel iterations with opposite directions. This results in the following algorithm:

Pseudocode 4.2.5.7: Multigrid iteration: recursive algorithm (symmetric V-cycle)

```

1 void mgsym_iteration(const  $\vec{\varphi} \in \mathbb{R}^{N_\ell}$ , ref  $\vec{\mu}$ , int  $\ell$ ) {
2   if ( $\ell == 0$ ) { Directly solve  $\mathbf{A}_0 \vec{\mu} = \vec{\varphi}$ ; }
3   else {
4      $\vec{\mu}_{\text{old}} := \vec{\mu}$ ;
5      $\vec{\mu} \leftarrow \vec{\mu} + \text{tril}(\mathbf{A})^{-1} (\vec{\varphi} - \mathbf{A} \vec{\mu})$ ; // Gauss-Seidel step, pre-smoothing
6      $\vec{\rho}_h := \vec{\varphi} - \mathbf{A} \vec{\mu}$ ; // Residual vector  $\in \mathbb{R}^{N_\ell}$ 
7      $\vec{\rho}_H := \mathbf{P}_H^\top \vec{\rho}_h$ ; // Residual vector  $\in \mathbb{R}^{N_{\ell-1}}$ 
8      $\vec{v}_H := \mathbf{0}$ ; // Natural initial guess for correction
9     mgsym_iteration( $\vec{\rho}_H, \vec{v}_H, \ell-1$ ); // Recursion
10     $\vec{\mu} \leftarrow \vec{\mu} + \mathbf{P}_H \vec{v}_H$ ; // Update approximate solution
11     $\vec{\mu} \leftarrow \vec{\mu} + \text{triu}(\mathbf{A})^{-1} (\vec{\varphi} - \mathbf{A} \vec{\mu})$ ; // Gauss-Seidel step, post-smoothing
12  }
13 }
```

Then the multigrid preconditioner can be realized as follows. Line 11 of Code 4.2.5.2 becomes

$$\vec{\eta} := \mathbf{B} \vec{\rho} \iff \vec{\eta} := \mathbf{0}; \text{ mgsym_iteration}(\vec{\rho}, \vec{\eta}, L), \quad (4.2.5.8)$$

where L is the refinement level of the finest mesh in the hierarchy, cf. Ass. 4.2.4.1.

4.2.6 Full Approximation Storage Multigrid (FAS)

Multigrid iterative solvers can be extended to non-linear boundary value problems, but the adaptation is not straightforward. It will be discussed in this section.

§4.2.6.1 (Non-linear variational boundary-value problem) On a (polygonally bounded) domain $\Omega \subset \mathbb{R}^2$ we consider a second-order scalar **non-linear** elliptic **variational problem** (NLVP) with homogeneous Dirichlet boundary conditions:

$$u \in V: \quad a(u; v) = \ell(v) \quad \forall v \in V, \quad V \subset H_0^1(\Omega), \quad (4.2.6.2)$$

where

- ◆ V is a subspace of $H_0^1(\Omega)$, which may also coincide with $H_0^1(\Omega)$,
- ◆ $a: V \times V \rightarrow \mathbb{R}$ is continuous and *linear* in the second argument,

- ◆ $\ell : H_0^1(\Omega) \rightarrow \mathbb{R}$ is a continuous/bounded linear form.
- The notation $\mathbf{a}(u; v)$ with a “;” is supposed to hint at the non-linear dependence on the first argument.
- Concrete examples are presented in [NumPDE Section 5.2.3]. The need to switch to a subspace of $H_0^1(\Omega)$ is due to the fact that \mathbf{a} may not be well-defined on the whole Sobolev space $H_0^1(\Omega)$.
- Of course, existence of solutions of (4.2.6.2) crucially depends on the properties of \mathbf{a} and will just be assumed in the sequel.

┘

§4.2.6.3 (Finite-element Galerkin discretization) We follow the approach of [NumPDE Section 5.3]. We equip Ω with a triangular finite-element mesh \mathcal{M} as defined in [NumPDE Section 2.5.1]. As trial and test space we opt for the lowest-order Lagrangian finite-element space $V_h := \mathcal{S}_{1,0}^0(\mathcal{M})$, assuming that $\mathcal{S}_{1,0}^0(\mathcal{M}) \subset V$. This yields the **nonlinear discrete variational problem**:

$$u_h \in V_h: \quad \mathbf{a}(u_h; v_h) = \ell(v_h) \quad \forall v_h \in V_h. \tag{4.2.6.4}$$

We write $\mathfrak{B}_h := \{b_h^1, \dots, b_h^N\}$ for the tent-function basis of V_h as introduced in [NumPDE Section 2.4.3]. Here $N := \dim V_h$, that is, N is the number of interior nodes of \mathcal{M} . Plugging basis expansions for u_h and v_h into (4.2.6.4) we obtain an $N \times N$ non-linear system of (algebraic) equations:

$$\vec{\mu} = [\mu_1, \dots, \mu_N]^\top \in \mathbb{R}^N: \quad \mathbf{a}\left(\sum_{j=1}^N \mu_j b_h^j; b_h^i\right) = \ell(b_h^i) \quad \forall i = 1, \dots, N. \tag{4.2.6.5}$$

For it we adopt the shorthand notation

$$\vec{\mu} = [\mu_1, \dots, \mu_N]^\top \in \mathbb{R}^N: \quad \mathbf{A}_h(\vec{\mu}) = \vec{\varphi} := \left[\ell(b_h^i)\right]_{i=1}^N, \tag{4.2.6.6}$$

with a non-linear mapping $\mathbf{A}_h : \mathbb{R}^N \rightarrow \mathbb{R}^N$, $\mathbf{A}_h(\vec{\xi}) := \left[\mathbf{a}\left(\sum_{j=1}^N \xi_j b_h^j; b_h^i\right)\right]_{i=1}^N$. ┘

§4.2.6.7 (Setting: nested meshes) We retain the setting of Section 4.2.4, in particular Ass. 4.2.4.1, which provides a hierarchy of *nested* triangular meshes of Ω :

$$\mathcal{M}_0 \prec \mathcal{M}_1 \prec \mathcal{M}_2 \prec \dots \prec \mathcal{M}_L, \quad L \in \mathbb{N}. \tag{4.2.6.8}$$

Remember from [NumPDE Ex. 3.1.4.3] that nesting of finite-element meshes $\mathcal{M}_{\ell-1} \prec \mathcal{M}_\ell$ means that

$$\forall K \in \mathcal{M}_{\ell-1}: \quad \exists K' \subset \mathcal{M}_\ell: \quad \bar{K} = \bigcup_{K' \in \mathcal{K}_K} \bar{K}', \tag{4.2.6.9}$$

that is, every (closed) triangle of $\mathcal{M}_{\ell-1}$ is the union of (closed) triangles of \mathcal{M}_ℓ . Recall that on nested meshes also the Lagrangian finite-element spaces are nested:

$$\mathcal{M}_{\ell-1} \prec \mathcal{M}_\ell \quad \Rightarrow \quad \mathcal{S}_{1,0}^0(\mathcal{M}_{\ell-1}) \subset \mathcal{S}_{1,0}^0(\mathcal{M}_\ell). \tag{4.2.6.10}$$

┘

§4.2.6.11 (Failure of stationary linear iterations for (4.2.6.6)) The coarse grid correction of geometric multigrid as introduced in Section 4.2.3 is a stationary linear iteration (4.1.3.13). We could try to generalize (4.1.3.13) to (4.2.6.6) as follows:

$$\vec{\mu}^{(k+1)} := \vec{\mu}^{(k)} + \mathbf{M}_h(\vec{\varphi} - \mathbf{A}_h(\vec{\mu}^{(k)})), \quad k = 0, 1, 2, \dots \tag{4.2.6.12}$$

Here $M_h : \mathbb{R}^N \rightarrow \mathbb{R}^N$ should be some approximate inverse of A_h . A simple “thought experiment” discloses That (4.2.6.12) is pointless. Assume that $A_h : \mathbb{R}^N \rightarrow \mathbb{R}^N$ is bijective and that $M_h = A_h^{-1}$ is its exact inverse. Not even in this case, we can say anything about the convergence of (4.2.6.12), let alone claim that it will converge to the solution in one step, as it happens in the linear case. \lrcorner

§4.2.6.13 (Non-linear Gauss-Seidel iteration) Though the use of stationary linear iterations is doomed, the subspace correction idea developed in Section 4.2.1 remains effective. Inspired by § 4.2.1.15 we can use it to define the **non-linear Gauss-Seidel iteration** for (4.2.6.6) as a non-linear successive subspace correction method in the direction of the one-dimensional spaces spanned by the basis functions b_h^i . Given an approximate solution $u_h \in V_h$ one sweep of the non-linear Gauss-Seidel iteration corrects it according to

$$\forall j \in \{1, \dots, N\}: \{ \zeta \in \mathbb{R}: a(u_h + \zeta b_h^j; b_h^j) = \ell(b_h^j); \quad u_h \leftarrow u_h + \zeta b_h^j; \} .$$

Pseudocode 4.2.6.14: Single Non-linear Gauss-Seidel sweep for $A_h(\vec{\mu}) = \vec{\varphi}$

```

1 void nIGS( $A_h : \mathbb{R}^N \rightarrow \mathbb{R}^N$ , const  $\vec{\varphi} \in \mathbb{R}^N$ , ref  $\vec{\mu} \in \mathbb{R}^N$ ) {
2   // Update all components of the approximate solution
3   for (int j=0; j<N; j++) {
4     Solve  $\zeta \in \mathbb{R}: (A_h(\vec{\mu} + \zeta \vec{e}_j))_j = (\vec{\varphi})_j$ ; //  $\vec{e}_i \triangleq j$ -th Cartesian basis
5      $(\vec{\mu})_j += \zeta$ ; \ \ Local update
6   }
7 }
```

This function assumes that, initially, its argument $\vec{\mu} \in \mathbb{R}^N$ already provides a *good approximation* of the solution of $A_h(\vec{\mu}) = \vec{\varphi}$.

Note that Line 4 amounts to solving a *scalar* non-linear equation of the form $F(\zeta) = 0$ for a function $F : \mathbb{R} \rightarrow \mathbb{R}$, here concretely given by $F(\zeta) := (A_h(\vec{\mu} + \zeta \vec{e}_j) - \vec{\varphi})_j$. We can use any method from [NumCSE Section 8.4] to determine ζ , Newton’s method, the secant method, *regula falsi*, etc. \lrcorner

§4.2.6.15 (FAS coarse grid correction) We adopt a two-grid perspective, fix $\ell \in \{1, \dots, N\}$, and abbreviate $\mathcal{M}_h := \mathcal{M}_\ell$, $\mathcal{M}_H := \mathcal{M}_{\ell-1}$, $V_h := \mathcal{S}_{1,0}^0(\mathcal{M}_h)$, $V_H := \mathcal{S}_{1,0}^0(\mathcal{M}_H)$. We start from an approximate solution $u_h \in V_h$ of (4.2.6.4). We try subspace correction in the direction of V_H :

$$c_H \in V_H: a(u_h + c_H; v_H) = \ell(v_H) \quad \forall v_H \in V_H; \quad u_h \leftarrow u_h + c_H . \tag{4.2.6.16}$$

Problem: We cannot determine c_H by solving a non-linear variational problem (4.2.6.4) on V_H .

However, finding a correction by solving the same problem “on the coarse grid”, that is, on V_H , is the gist of coarse grid correction. In this respect (4.2.6.16) disappoints.

An equivalent way to write the discrete variational equation of (4.2.6.16) is

$$c_H \in V_H: a(u_h + c_H; v_H) - a(u_h; v_H) = r(u_h; v_H) \quad \forall v_H \in V_H , \tag{4.2.6.17}$$

with **residual linear form** $r(u_h; v) := \ell(v) - a(u_h; v)$, $v \in V$.



Idea: Replace u_h on the left-hand side of (4.2.6.17) with an approximation $\tilde{u}_H \in V_H$.

This converts (4.2.6.17) into

$$c_H \in V_H: a(\tilde{u}_H + c_H; v_H) - a(\tilde{u}_H; v_H) = r(u_h; v_H) \quad \forall v_H \in V_H , \tag{4.2.6.18}$$

which, assuming that the residual linear form can be evaluated on V_H , is a non-linear variational problem completely set in V_H . To see this, we recast (4.2.6.18) as a non-linear system of equations. We set $N_H := \dim V_H$, write $\vec{\mu}_H \in \mathbb{R}^{N_H}$ for the basis expansion coefficient vector of $\tilde{u}_H \in V_H$, and obtain the basis expansion coefficient vector $\vec{v}_H \in \mathbb{R}^{N_H}$ of $c_H + \tilde{u}_H \in V_H$ from

$$\vec{\zeta}_H \in \mathbb{R}^{N_H}: \quad \mathbf{A}_H(\vec{\zeta}_H) = \mathbf{A}_H(\vec{\mu}_H) + \vec{\rho}_H; \quad \vec{v}_H := \vec{\zeta}_H - \vec{\mu}_H, \quad (4.2.6.19)$$

where $\vec{\rho}_H := [r(u_h; b_H^1), \dots, r(u_h; b_H^{N_H})] \in \mathbb{R}^{N_H}$. As in Section 4.2.3, that vector $\vec{\rho}_H$ can be computed as the restriction of the residual vector from the fine grid

$$\vec{\rho}_H := \mathbf{P}_H^\top \vec{\rho}_h, \quad \vec{\rho}_h := [r(u_h; b_h^1), \dots, r(u_h; b_h^N)] \in \mathbb{R}^N, \quad (4.2.6.20)$$

and the matrix $\mathbf{P}_H \in \mathbb{R}^{N, N_H}$ defined in (4.2.3.7), (4.2.3.8).

The standard choice for \tilde{u}_H is $\tilde{u}_H := I_H u_h$, where $I_H : C^0(\bar{\Omega}) \rightarrow \mathcal{S}_1^0(\mathcal{M}_H)$ is the linear interpolation operator [NumPDE Def. 3.3.2.1] defined by

$$(I_H f)(p) = f(p) \quad \forall p \in \mathcal{V}(\mathcal{M}_H) := \{\text{nodes of } \mathcal{M}_H\}. \quad (4.2.6.21)$$

On the algebraic level, with $\vec{\mu}_*$ standing for the basis expansion coefficient vectors of $u_* \in V_*$, this can be expressed as

$$\vec{\mu}_H = \mathbf{N}_H \vec{\mu}_h, \quad \mathbf{N}_H \in \{0, 1\}^{N_H, N}, \quad (\mathbf{N})_{ij} := \begin{cases} 1 & \text{, if node } i \text{ of } \mathcal{M}_H = \text{node } j \text{ of } \mathcal{M}_h, \\ 0 & \text{else.} \end{cases} \quad (4.2.6.22)$$

┘

§4.2.6.23 (FAS two-grid iteration) As in Section 4.2.3 non-linear Gauss-Seidel sweeps as smoothing steps can be combined with the FAS coarse grid correction to obtain a viable two-grid iterative solver. The following code illustrates the FAS counterpart of Code 4.2.3.12.

Pseudocode 4.2.6.24: FAS Two-grid iteration

```

1 void two_grid_fas(const  $\vec{\varphi} \in \mathbb{R}^N$ , ref  $\vec{\mu}$ , double TOL) {
2   do {
3      $\vec{\mu}_{\text{old}} := \vec{\mu}$ ;
4      $\vec{\mu}_H := \mathbf{N}_H \vec{\mu}$ ; //  $\mathbf{N}_H$  as in (4.2.6.22)
5     nIGS( $\mathbf{A}_h$ ,  $\vec{\varphi}$ ,  $\vec{\mu}$ ); // Pre-smoothing, see Code 4.2.6.14
6      $\vec{\rho}_h := \vec{\varphi} - \mathbf{A}_h(\vec{\mu})$ ; // Residual vector  $\in \mathbb{R}^N$ 
7      $\vec{\rho}_H := \mathbf{P}_H^\top \vec{\rho}_h$ ; // Residual vector  $\in \mathbb{R}^{N_H}$  by restriction
8     Solve  $\mathbf{A}_H(\vec{\zeta}_H) = \mathbf{A}_H(\vec{\mu}_H) + \vec{\rho}_H$ ; // Coarse grid problem
9      $\vec{v}_H := \vec{\zeta}_H - \vec{\mu}_H$ ; // Correction
10     $\vec{\mu} \leftarrow \vec{\mu} + \mathbf{P}_H \vec{v}_H$ ; // Prolongation and update of approximate solution
11    nIGS( $\mathbf{A}_h$ ,  $\vec{\varphi}$ ,  $\vec{\mu}$ ); // Post-smoothing, see Code 4.2.6.14
12  }
13  while ( $\|\vec{\mu} - \vec{\mu}_{\text{old}}\| > \text{TOL} \cdot \|\vec{\mu}\|$ ); // Termination test
14 }
```

┘

§4.2.6.25 (Full FAS multigrid method) Of course, the two key ideas of Section 4.2.4 can also be applied with for the FAS scheme:

- **Recursion:** The coarse grid problem in Line 8 of Code 4.2.6.24 can be solved approximately by calling

```
1 two_grid_fas(AH( $\vec{\mu}_H$ ) +  $\vec{\rho}_H$ ,  $\vec{\zeta}_H$ , TOL);
```

- **Nested iteration:** An initial guess on \mathcal{M}_ℓ is obtained by prolongating the approximate solution from $\mathcal{M}_{\ell-1}$, refer to § 4.2.4.14.

┘

Remark 4.2.6.26. It goes without saying that the FAS multigrid method as introduced above can be abstractly defined for a non-linear variational problem like (4.2.6.2) and its Galerking discretization based on a sequence of *nested* trial and test spaces $V_0 \subset V_1 \subset \dots \subset V_L$.

┘

Review question(s) 4.2.6.27 (Geometric multigrid methods)

(Q4.2.6.27.A) Consider an SSC method based on sub-spaces $V_m \subset V_h$, $m = 1, \dots, M$ for iteratively solving a linear discrete variational problem posed on V_h .

- Show that the sub-space correction in the direction of V_m gives the exact solution, if the iteration error was in V_m before that sub-space correction.
- In light of (i), why do SSC methods not converge in one step generally, though the sum of the underlying sub-spaces spans the whole space V_h ?

△

4.3 Algebraic Multigrid (AMG): Matrix-Based Multigrid

§4.3.0.1 (Need for black-box plug-in iterative solvers) For a large array of discretized (by means of finite elements or finite differences) boundary value problems for PDEs full **geometric multigrid** iterative solvers provide sufficiently accurate solutions with $O(N)$ asymptotic effort for $N \rightarrow \infty$, N the number of unknowns/degrees of freedom. What else could we desire?

Yet, geometric multigrid relies on a hierarchy of (nested) meshes, remember Ass. 4.2.4.1, whereas in practical finite-element computations are usually done on a single fine mesh output by a mesh generator (like GMSH [NumPDE Section 2.7.1]).

► In practice, when solving boundary value problems on complicated domains, in general *no hierarchy of nested meshes* is available.

In addition, non-modular “legacy” simulation codes are still widely used and maintainers are wary of rewriting them in order to incorporate a geometric multigrid module, because this will entail a highly *intrusive* modification of the computational kernel.

“Network-type” systems of equations that share many properties with those arising from the discretization of boundary value problems for PDEs, but lack a PDE background, are faced in many applications. The missing context of a spatial discretization makes generalizing geometric multigrid difficult.

The dream (at least for the case of sparse linear systems of equations) is,

- an iterative solution method that uses only information contained in the system matrix and the right-hand side vector (“**black-box**”)
- that can be implemented as a function (“**plug-in**”)

```
Vector solve(const Matrix A, const Vector rhs, double tolerance)
```

- that delivers fast convergence independent of the size of the linear system,
- and with the cost of one iteration proportional for the system size.

Of, course for general sparse linear systems of equations this will remain a dream. However, the **algebraic multigrid method (AMG)** presented in this section works this magic at least for linear systems of equations arising from the finite-element discretization of 2nd-order scalar elliptic BVPs. \lrcorner

4.3.1 AMG Framework

We focus on the two-grid method. Extension to many grids/levels is immediate by recursion, see Code 4.2.4.5.

The goal is to solve a linear system of equations $\mathbf{A}\vec{\mu} = \vec{\varphi}$, $\mathbf{A} \in \mathbb{R}^{N,N}$ large and *sparse*, iteratively.

Assumption 4.3.1.1.

The sparse matrix $\mathbf{A} \in \mathbb{R}^{N,N}$ is given in a sparse matrix format, e.g., CRS/CCS [NumCSE Section 2.7.1], permitting access to the non-zero entries in rows and columns.

§4.3.1.2 (Building multigrid components algebraically) Based on the matrix $\mathbf{A} \in \mathbb{R}^{N,N}$ we have to find the following components that we identified as key building blocks for the geometric multigrid two-grid iteration of Code 4.2.3.12:

- (I) The **prolongation matrix**/transfer matrix $\mathbf{P} \in \mathbb{R}^{N,N_H}$, and
- (II) and a **smoother**, an affine linear operator $\mathbb{R}^N \rightarrow \mathbb{R}^N$.

What about the course-grid matrix \mathbf{A}_H that plays a key role in Code 4.2.3.12? We can recover it through the so-called **Galerkin construction** as $\mathbf{A}_H = \mathbf{P}^\top \mathbf{A} \mathbf{P} \in \mathbb{R}^{N_H,N_H}$. The name can be explained as follows assuming that \mathbf{A} stems from a Galerkin discretization and using the notations of Section 4.2.3: for all $i, j \in \{1, \dots, N_H\}$

$$(\mathbf{A}_H)_{ij} = a(b_H^j, b_H^i) = \sum_{\ell=1}^N \sum_{k=1}^N a((\mathbf{P})_{\ell,j} b_{h'}^\ell, (\mathbf{P})_{k,i} b_h^k) = \sum_{\ell=1}^N \sum_{k=1}^N (\mathbf{P})_{\ell,j} (\mathbf{P})_{k,i} a(b_{h'}^\ell, b_h^k) = (\mathbf{P}^\top \mathbf{A} \mathbf{P})_{ij}.$$

In fact, this construction was also used in Code 4.2.3.12.

Note that \mathbf{A}_H will be regular, provided that \mathbf{A} is regular and the nullspace of \mathbf{P} is trivial: $\mathcal{N}(\mathbf{P}) = \{\mathbf{0}\}$. \lrcorner

§4.3.1.3 (AMG two-grid iteration) Almost all AMG algorithms rely on simple point smoothers, in particular Gauss-Seidel sweeps. This settles Item (II) from § 4.3.1.2. Then the AMG two-grid iteration is essentially the same as the method outlined in Code 4.2.3.12:

Pseudocode 4.3.1.4: AMG two-grid iteration step, coarse-grid-correction highlighted

```

1 void AMG_TG_step(const Matrix A ∈ ℝN,N, const Vector φ ∈ ℝN,
2                 ref Vector μ) {
3     Matrix A_H := P_H⊤ A P_H; // Galerkin construction, precomputed once
4     μ ← μ + triu(A)-1 (φ - Aμ); // (forward) Gauss-Seidel pre-smoothing
5     ρ_h := φ - Aμ; // Residual vector ∈ ℝN
6     ρ_H := P_H⊤ ρ_h; // Residual vector ∈ ℝN_H by restriction
7     Solve A_H v_H = ρ_H; // Coarse grid correction equation
8     μ ← μ + P_H v_H; // Prolongation and update of approximate solution
9     μ ← μ + tril(A)-1 (φ - Aμ); // (backward) Gauss-Seidel post-smoothing
10 }

```

This is a stationary linear iteration (4.1.3.14) with error propagation matrix, cf. Cor. 4.2.3.14,

$$\mathbf{E}_{ATG} := \underbrace{(\mathbf{I}_N - \text{tril}(\mathbf{A})^{-1}\mathbf{A})}_{=\mathbf{E}_{GS}^\top} \underbrace{(\mathbf{I}_N - \mathbf{P}\mathbf{A}_H^{-1}\mathbf{P}^\top\mathbf{A})}_{=\mathbf{E}_{GCC}} \underbrace{(\mathbf{I}_N - \text{triu}(\mathbf{A})^{-1}\mathbf{A})}_{=\mathbf{E}_{GS}}. \tag{4.3.1.5}$$

┘

Remark 4.3.1.6 (AMG terminology) The inventors of AMG methods always imagined that the matrix $\mathbf{A} \in \mathbb{R}^{N,N}$ was generated by the discretization of a scalar elliptic boundary value problem by means of linear Lagrangian finite elements. This is why they referred to indices $\in \{1, \dots, N\}$ as **nodes**, a terminology that has persisted. ┘

§4.3.1.7 (Abstract AMG prolongation matrix) At this point the design of an AMG method is reduced to the single task of finding a suitable matrix $\mathbf{P} \in \mathbb{R}^{N,N_H}$!

This mission is split into two steps:

- ① Definition of a suitable **coarse-fine splitting** of the index set $\{1, \dots, N\}$, a partition

$$\{1, \dots, N\} = \mathcal{C} \cup \mathcal{F}, \quad \mathcal{C} \cap \mathcal{F} = \emptyset. \tag{4.3.1.8}$$

The indices in \mathcal{C} are called the **coarse nodes** (C-nodes), those in \mathcal{F} the F-nodes, and we have $N_H := \#\mathcal{C}$. The indices in \mathcal{C} are supposed to be ordered and, then, are associated with the columns of the prolongation matrix $\mathbf{P} \in \mathbb{R}^{N,N_H}$.

Fixing \mathcal{C} entails the following constraint on \mathbf{P}

$$(\mathbf{P})_{ij} = \begin{cases} 1 & , \text{ if } i \text{ is the } j\text{-th index in } \mathcal{C}, \\ 0 & \text{ for all other indices } i \in \mathcal{C}. \end{cases} \tag{4.3.1.9}$$

This means, that the prolonged value of a C-node is preserved for that C-node, solely spreads to F-nodes, and does not affect any other C-nodes:

Application of $\mathbf{P} \iff$ *interpolation* from C-nodes into F-nodes.

Furthermore, after reordering the basis vectors of \mathbb{R}^N according to the coarse-fine splitting (C-indices first, F-indices last), we can bring \mathbf{P} in the form

$$\mathbf{P} = \begin{bmatrix} \mathbf{I}_{N_H} \\ \mathbf{P}_F \end{bmatrix}, \quad \begin{array}{l} \mathbf{I}_{N_H} \hat{=} N_H \times N_H \text{ identity matrix,} \\ \mathbf{P}_F \in \mathbb{R}^{N_F, N_H}, \quad N_F := N - N_H. \end{array} \tag{4.3.1.10}$$

- ② Determination of the entries of \mathbf{P}_F , known as **prolongation weights**.

For the sake of efficiency – we strive for $O(N)$ asymptotic computational cost for a single two-grid iteration step except for solving $\mathbf{A}_H \vec{v}_H = \vec{\rho}_H$ — the matrix \mathbf{P}_F must be *sparse* with only a small number of non-zero entries per row: \mathbf{P} should distribute the value of a \mathcal{C} -node to only a few \mathcal{F} -nodes. ┘

Remark 4.3.1.11 (Relationship to prolongation for geometric multigrid) We return to the geometric multigrid setting of Section 4.2.3 and relate it to the previous §.

- The C-nodes of AMG clearly correspond to the vertices of the coarse mesh.
- The requirement (4.3.1.9) is satisfied by the formula (4.2.3.8).

┘

§4.3.1.12 (Matrix (adjacency) graph and mesh graph)

Definition 4.3.1.13. Matrix graph

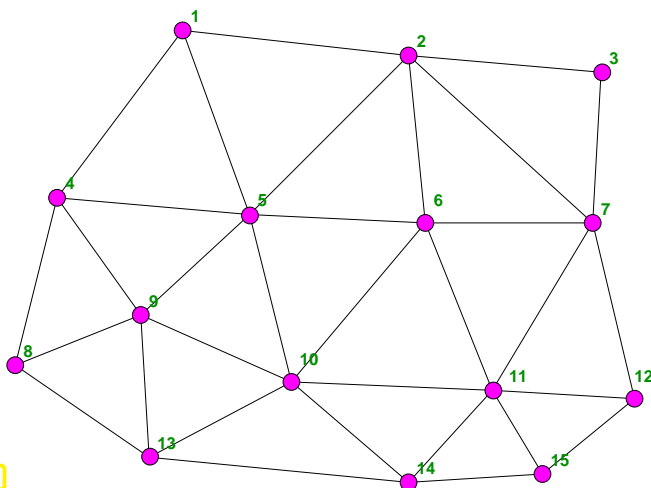
Given a square matrix $\mathbf{A} \in \mathbb{R}^{N,N}$, its associated **matrix (adjacency) graph** is the directed graph $\mathcal{G}_A = (\mathcal{V}, \mathcal{E})$ with

- node/vertex set $\mathcal{V} := \{1, \dots, N\}$, and
- edge set $\mathcal{E} := \{(i, j) \in \{1, \dots, N\}^2 : (\mathbf{A})_{i,j} \neq 0\}$.

Sloppily speaking, the edges of a matrix graph correspond to the non-zero entries of the matrix. If the matrix \mathbf{A} is symmetric, then \mathcal{G}_A can be regarded as an undirected graph. This is the perspective adopted in the sequel.

If \mathbf{A} is regarded as a $\mathcal{S}_1^0(\mathcal{M})$ Galerkin matrix, then the matrix graph of \mathbf{A} agrees with the **vertex-edge graph** of the finite-element mesh \mathcal{M} , unless some entries of \mathbf{A} vanish “by coincidence”. Refer to [NumPDE Section 2.4.4] for more explanations.

◁ planar visualization of matrix graph of



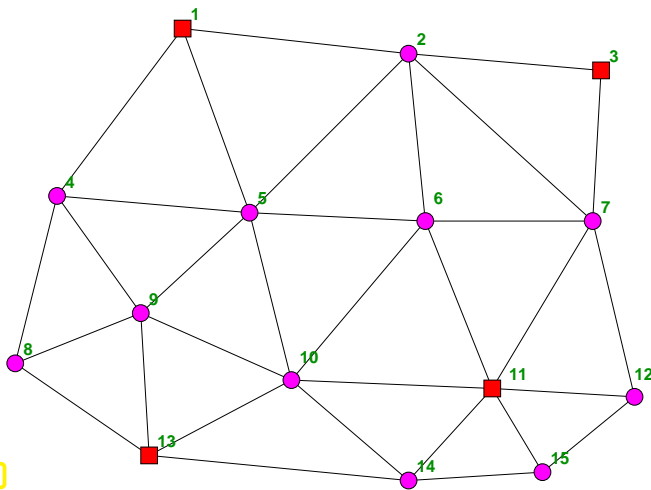
$$\mathbf{A} = \begin{bmatrix} * & * & 0 & * & * & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ * & * & * & 0 & * & * & * & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & * & * & 0 & 0 & 0 & * & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ * & 0 & 0 & * & * & 0 & 0 & * & * & 0 & 0 & 0 & 0 & 0 & 0 \\ * & * & 0 & 0 & * & * & 0 & 0 & * & * & 0 & 0 & 0 & 0 & 0 \\ 0 & * & 0 & 0 & * & * & * & 0 & 0 & * & * & 0 & 0 & 0 & 0 \\ 0 & * & * & 0 & 0 & * & * & 0 & 0 & 0 & * & * & 0 & 0 & 0 \\ 0 & 0 & 0 & * & 0 & 0 & 0 & * & * & 0 & 0 & 0 & * & 0 & 0 \\ 0 & 0 & 0 & * & * & 0 & 0 & * & * & * & 0 & 0 & * & 0 & 0 \\ 0 & 0 & 0 & 0 & * & * & 0 & 0 & * & * & * & 0 & 0 & * & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & * & * & 0 & 0 & 0 & * & * & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & * & * & 0 & * & * & * \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & * & * & 0 & * & * \end{bmatrix}$$

Fig. 179

┘

§4.3.1.14 (Matrix graph and prolongation) Interpreting vectors $\in \mathbb{R}^{N_H}$ as values attached to C-nodes, the action of the F-part \mathbf{P}_F of the prolongation matrix \mathbf{P} amounts to distributing values from C-nodes to F-nodes.

This action can be regarded as *local*, if $(\mathbf{P}_F)_{i,k} \neq 0$ only if $(\mathbf{A})_{i,j} \neq 0$, where the k -th C-node has the index j : Values are distributed along edges of the matrix graph.



Small example related to Fig. 179:

◁ ■ $\hat{=}$ nodes $\in \mathcal{C}$: C-nodes

● $\hat{=}$ nodes $\in \mathcal{F}$: F-nodes

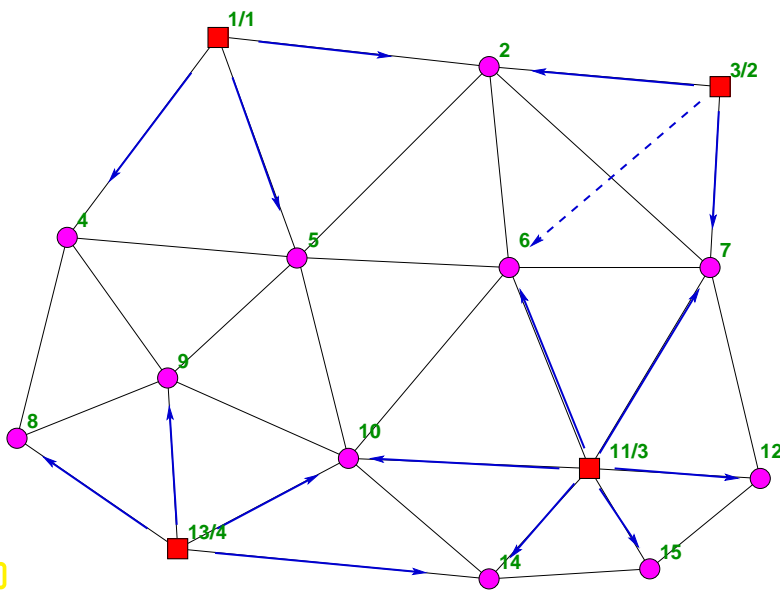
— $\hat{=}$ edges of the mesh

$$\mathcal{C} = \{1, 3, 11, 13\},$$

$$\mathcal{F} = \{1, 2, 4, 6, 7, 8, 9, 10, 12, 14, 15\}.$$

Fig. 180

In the figure below solid blue arrows indicate local distribution of values from C-nodes, the single dashed arrow marks a non-local assignment, node numbers “ i/j ” give indices “on the fine/coarse mesh”.



$$\mathbf{P} = \begin{bmatrix} * & 0 & 0 & 0 \\ * & * & 0 & 0 \\ 0 & * & 0 & 0 \\ * & 0 & 0 & 0 \\ * & 0 & 0 & 0 \\ 0 & * & 0 & 0 \\ 0 & * & * & 0 \\ 0 & 0 & 0 & * \\ 0 & 0 & 0 & * \\ 0 & 0 & * & * \\ 0 & 0 & * & 0 \\ 0 & 0 & * & 0 \\ 0 & 0 & 0 & * \\ 0 & 0 & * & * \\ 0 & 0 & * & 0 \end{bmatrix}.$$

Fig. 181

Remark 4.3.1.15 (The AMG fill-in challenge) The embedding of the algebraic two-grid method into a recursive algebraic multigrid method hinges on the coarse grid matrix \mathbf{A}_H being sparse with a small fixed number of non-zero entries per row and column. Otherwise the smoothing steps will become too costly and the algorithms constructing coarse-fine splittings will fail.

The coarse grid matrix is obtained by the Galerkin construction $\mathbf{A}_H := \mathbf{P}^\top \mathbf{A} \mathbf{P}$. Thus its sparsity pattern is completely determined by that of \mathbf{P} and \mathbf{A} : For $k, \ell \in \{1, \dots, N_H\}$,

$$(\mathbf{A}_H)_{k,\ell} \neq 0 \implies \exists i, j \in \{1, \dots, N\}: (\mathbf{P})_{i,k} \neq 0 \wedge (\mathbf{A})_{ij} \neq 0 \wedge (\mathbf{P})_{j,\ell} \neq 0. \quad (4.3.1.16)$$

Thus, necessary (but not sufficient!) condition for the sparsity of \mathbf{A}_H is that \mathbf{P} is also sparse; each C-node is connected with a small number of F-nodes only. A stronger condition is that \mathbf{P} is local,

$$(\mathbf{P})_{i,k} \neq 0 \text{ only if F-node } i \text{ and C-node } k \text{ are connected in the matrix graph of } \mathbf{A}. \quad (4.3.1.17)$$

The following example based on Fig. 181 displays the matrix graph of \mathbf{A}_H for a local \mathbf{P} .

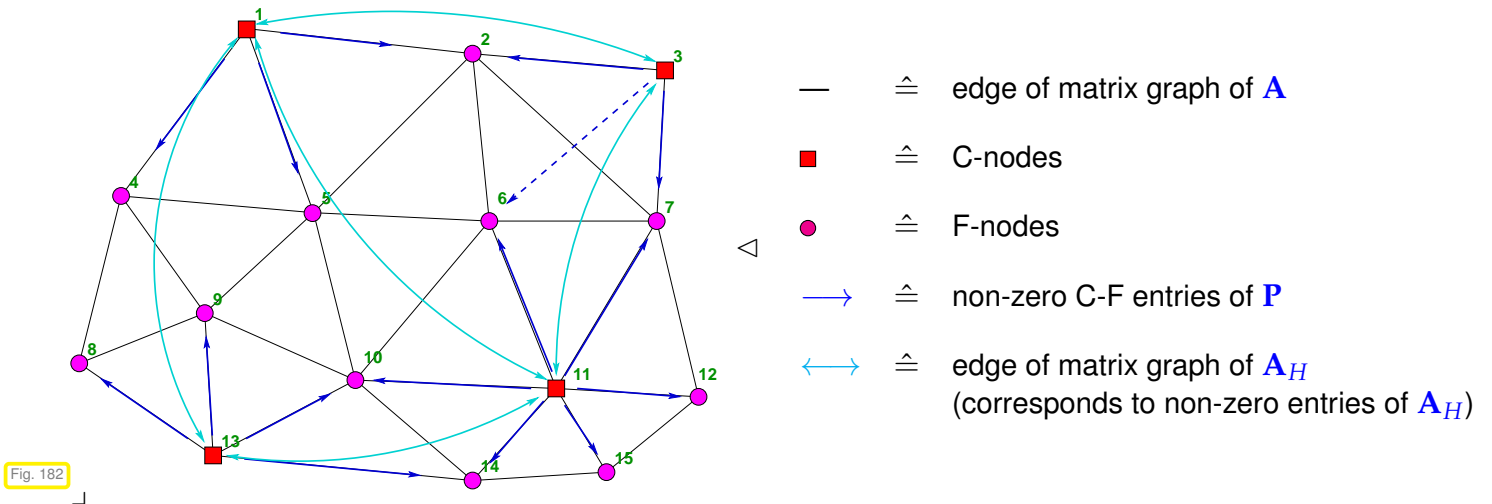


Fig. 182

4.3.2 AMG Heuristics

We develop a *recipe* for building \mathbf{P} given a coarse-fine splitting.

§4.3.2.1 (Assumptions on \mathbf{A}) We focus on the Galerkin finite element discretization of a second-order scalar elliptic boundary value problem (4.1.1.1) using piecewise linear Lagrangian finite elements and the tent function basis. In particular $\mathbf{A} \in \mathbb{R}^{N,N}$ is supposed to be *s.p.d.* (symmetric positive definite):

$$\mathbf{A} = \mathbf{A}^\top \quad \text{and} \quad \vec{v}^\top \mathbf{A} \vec{v} > 0 \quad \forall \vec{v} \in \mathbb{R}^N \setminus \{0\}. \tag{4.3.2.2}$$

Temporarily we make the additional assumption that \mathbf{A} is an **M-matrix** [NumPDE Rem. 3.7.2.11], [NumPDE Def. 3.7.2.18]:

- $(\mathbf{A})_{ii} > 0$ (positive diagonal) , [NumPDE Eq. (3.7.2.12)]
- $(\mathbf{A})_{ij} \leq 0$ for $j \neq i$ (non-positive off-diagonal entries) , [NumPDE Eq. (3.7.2.13)]
- $\sum_j (\mathbf{A})_{ij} \geq 0$ (non-negative row sums) . [NumPDE Eq. (3.7.2.14)]

From [NumPDE Rem. 3.7.2.21] we know that for $d = 2$ and triangular meshes \mathcal{M} the $\mathcal{S}_{1,0}^0(\mathcal{M})$ finite-element Galerkin matrix for (4.1.1.12) will be an M-matrix, if none of the triangles has an obtuse angle $> \pi/2$.

§4.3.2.3 (Assumptions on $\vec{\varphi}$) Note that the right-hand side vector $\vec{\varphi}$ of the linear system of equations $\mathbf{A}\vec{\mu} = \vec{\varphi}$ tackled by a stationary linear iteration (4.1.3.13) does not influence the error propagation matrix. Hence, the convergence of a stationary linear iteration can be fully understood by studying the case $\vec{\varphi} = \mathbf{0}$.

Since the algebraic multigrid method is a special stationary linear iteration we need only consider the case $\vec{\varphi} = \mathbf{0}$ in its derivation.

§4.3.2.4 (Effect of coarse grid correction) We study the error propagation operator associated with coarse grid correction derived in also Section 4.2.3:

$$\mathbf{E}_{CGC} := \mathbf{I}_N - \mathbf{P}\mathbf{A}_H^{-1}\mathbf{P}^\top \mathbf{A} \tag{4.3.2.5}$$

We reconnect to some insights already gained in § 4.2.2.1.

Lemma 4.3.2.6. Error propagation operator of coarse grid correction

Assume that $\mathbf{A} = \mathbf{A}^\top \in \mathbb{R}^{N,N}$ is s.p.d. and that the prolongation matrix $\mathbf{P} \in \mathbb{R}^{N,N_H}$, $N_H < N$, has full rank N_H .

Then, in the case of the Galerkin construction of the coarse grid matrix, $\mathbf{A}_H := \mathbf{P}^\top \mathbf{A} \mathbf{P}$, the error propagation operator of the coarse grid correction is

- an **A-orthogonal** projection (\rightarrow Def. 4.2.2.2),
- with kernel $\mathcal{N}(\mathbf{E}_{\text{CGC}}) = \mathcal{R}(\mathbf{P})$.

📎 Notations: We write $\mathcal{N}(\mathbf{M})$ for the kernel/nullspace of a matrix \mathbf{M} and $\mathcal{R}(\mathbf{M})$ for its range space/image space.

Recall that an s.p.d. matrix $\mathbf{A} \in \mathbb{R}^{N,N}$ induces an inner product on \mathbb{R}^N . “A-orthogonal” refers to that inner product.

Proof. (of Lemma 4.3.2.6) From linear algebra you know that a linear mapping described by a matrix \mathbf{M} is an A-orthogonal projection, if

- \mathbf{M} is idempotent, $\mathbf{M}^2 = \mathbf{M}$,
- \mathbf{M} is A-selfadjoint,

$$(\mathbf{A}\mathbf{M}\vec{\xi}) \cdot \vec{v} = (\mathbf{A}\vec{\xi}) \cdot (\mathbf{M}\vec{v}) \iff \mathbf{M}^\top \mathbf{A} = \mathbf{A}\mathbf{M}. \quad (4.3.2.7)$$

We directly verify these properties for \mathbf{E}_{CGC} :

$$\begin{aligned} \mathbf{E}_{\text{CGC}}^2 &= \mathbf{I}_N - 2\mathbf{P}\mathbf{A}_H^{-1}\mathbf{P}^\top + \mathbf{P} \underbrace{(\mathbf{P}^\top \mathbf{A} \mathbf{P})^{-1} \mathbf{P}^\top \mathbf{A} \mathbf{P}}_{=\mathbf{I}_{N_H}} \mathbf{A}_H^{-1} \mathbf{P}^\top \mathbf{A} = \mathbf{E}_{\text{CGC}}, \\ \mathbf{E}_{\text{CGC}}^\top \mathbf{A} &= \mathbf{A} - \mathbf{A} \mathbf{P} \mathbf{A}_H^{-1} \mathbf{P}^\top \mathbf{A} = \mathbf{A} (\mathbf{I}_N - \mathbf{P} \mathbf{A}_H^{-1} \mathbf{P}^\top \mathbf{A}) = \mathbf{A} \mathbf{E}_{\text{CGC}}. \end{aligned}$$

The kernel of \mathbf{E}_{CGC} is found as follows:

$$\begin{aligned} \vec{\xi} \in \mathcal{N}(\mathbf{E}_{\text{CGC}}) &\iff \vec{\xi} = \mathbf{P} \mathbf{A}_H^{-1} \mathbf{P}^\top \vec{\xi} \in \mathcal{R}(\mathbf{P}), \\ \vec{\xi} \in \mathcal{R}(\mathbf{P}) &\iff \exists \vec{v} \in \mathbb{R}^{N_H} : \vec{\xi} = \mathbf{P}\vec{v} \implies \mathbf{E}_{\text{TGM}} \vec{\xi} = \mathbf{P}\vec{v} - \mathbf{P} \underbrace{(\mathbf{P}^\top \mathbf{A} \mathbf{P})^{-1} \mathbf{P}^\top \mathbf{A} \mathbf{P}}_{=\mathbf{I}_{N_H}} \vec{v} = \mathbf{0}. \end{aligned}$$

Remember that the assumption that \mathbf{P} has full rank/trivial kernel ensures that \mathbf{A}_H is invertible. □

We have learned that the coarse grid correction completely eliminates error components in $\mathcal{R}(\mathbf{P})$.

Error components that are insufficiently reduced by the smoother must be captured by $\mathcal{R}(\mathbf{P})$.

More formally, writing \mathbf{E}_S for the error propagation operator of the smoothing step, we desire

$$\vec{\eta} \in \mathbb{R}^N, \quad \mathbf{E}_S \vec{\eta} \approx \vec{\eta} \implies \vec{\eta} \in \mathcal{R}(\mathbf{P}) \text{ desired.}$$

For Gauss-Seidel smoothing, as used in Code 4.3.1.4, we have $\mathbf{E}_S = \mathbf{I}_N - \text{tril}(\mathbf{A})^{-1} \mathbf{A}$. Thus, for vanishing right-hand side vector

$$\vec{\eta} \approx (\mathbf{I}_N - \text{tril}(\mathbf{A}))^{-1} \mathbf{A} \vec{\eta} \iff (\vec{\eta})_i \approx -\frac{1}{(\mathbf{A})_{ii}} \sum_{i \neq j} (\mathbf{A})_{ij} (\vec{\eta})_j, \quad i = 1, \dots, N. \quad (4.3.2.8)$$

Hence, if \mathbf{A} is an M-matrix, the components error vectors that the Gauss-Seidel smoother just passes on are **A-weighted averages** of the components associated with neighboring nodes (in the matrix graph of \mathbf{A}).

Ideally, one would like to use (4.3.2.8) with “ \approx ” replaced with “ $=$ ” to define \mathbf{P} . However, this amounts to too many constraints and it clashes with the requirement (4.3.1.9) that values in C-nodes are preserved. Yet, for F-nodes our wish can be fulfilled.



Choose \mathbf{P} such that for $i \in \mathcal{F}$ $(\mathbf{P}\vec{\eta})_i$ is an A-weighted average:

$$(\mathbf{P}\vec{\eta})_i = -\frac{1}{(\mathbf{A})_{i,i}} \sum_{j \neq i} (\mathbf{A})_{i,j} \cdot \eta\text{-value at node } j \quad (4.3.2.9)$$

In this crude form, this idea is doomed, because \mathbf{P} acts only on the values in the C-nodes, whereas the sums on the right-hand side of (4.3.2.9) also cover F-nodes. We will resume the discussion and resolve this problem in Section 4.3.4.

4.3.3 C/F Splitting Algorithm

F-node prolongation by averaging, the idea floated in the previous section for the construction of the prolongation matrix \mathbf{P} , can only work if every F-node is “sufficiently strongly connected” (w.r.t. \mathbf{A}) to at least one C-node. In this section we want to make rigorous this fuzzy requirement and let it inspire an algorithm for building a coarse-fine splitting (4.3.1.8).

Also in this section $\mathbf{A} \in \mathbb{R}^{N,N}$ should be thought of as a symmetric M-matrix; we retain the assumptions of § 4.3.2.1.

§4.3.3.1 (Strong coupling) We quantify the notion of coupling between two indices/nodes/degrees of freedom. Of course, this will rely on the relative sizes and signs of entries of \mathbf{A} .

Definition 4.3.3.2. Strongly coupled nodes

Given $\mathbf{A} = \mathbf{A}^\top \in \mathbb{R}^{N,N}$ a pair (i, j) of different nodes/indices $i, j \in \{1, \dots, N\}$, $i \neq j$, is **(τ_C -strongly (negatively) coupled** (w.r.t. to \mathbf{A}), if

$$-(\mathbf{A})_{i,j} \geq \tau_C \max\{ |(\mathbf{A})_{i,k}| : (\mathbf{A})_{i,k} < 0, k \in \{1, \dots, N\} \setminus \{i\} \} \quad (4.3.3.3)$$

for some fixed $\tau_C \in]0, 1[$.

A commonly used value for τ_C is $\tau_C = \frac{1}{4}$. From now we fix τ_C .

For the sake of concise notation we introduce two sets of strongly coupled neighboring nodes:

$$\begin{aligned} \mathcal{S}(i) &:= \{j \in \{1, \dots, N\} : (i, j) \text{ is strongly coupled}\} \\ &:= \left\{ j \in \{1, \dots, N\} : -(\mathbf{A})_{i,j} \geq \tau_C \max\{ |(\mathbf{A})_{i,k}| : (\mathbf{A})_{i,k} < 0, k \in \{1, \dots, N\} \setminus \{i\} \} \right\}, \\ \mathcal{S}(j)^* &:= \{i \in \{1, \dots, N\} : j \in \mathcal{S}(i)\} \\ &= \left\{ i \in \{1, \dots, N\} : -(\mathbf{A})_{i,j} \geq \tau_C \max\{ |(\mathbf{A})_{i,k}| : (\mathbf{A})_{i,k} < 0, k \in \{1, \dots, N\} \setminus \{i\} \} \right\}. \end{aligned}$$

- $\mathcal{S}(i)$ captures the strong couplings **to** node i , the set of nodes from which node i could receive values during prolongation.
- $\mathcal{S}(j)^*$ is the set of nodes to which node j could send values during prolongation.

In general $\mathcal{S}(i) \neq \mathcal{S}(i)^*$. ┘

EXAMPLE 4.3.3.4 (Strong couplings defined by Poisson matrix) We consider the Poisson matrix $\mathbf{A} \in \mathbb{R}^{N,N}$, $N = (M - 1)^2$, from Ex. 4.1.1.22. We clearly have

$$\mathcal{S}(i) = \{j \in \{1, \dots, N\} \setminus \{i\} : (\mathbf{A})_{i,j} \neq 0\}.$$

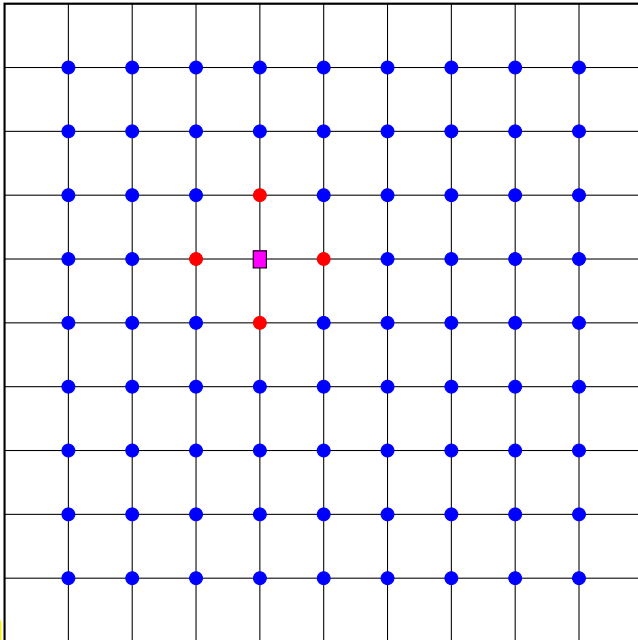


Fig. 183

Remember that the indices/nodes are in one-to-one correspondence with the nodes of a tensor-product mesh of $]0, 1[^2$ and that \mathbf{A} can be represented by a five-point stencil, see § 4.1.1.31, [NumPDE § 4.1.2.10]:

$$\mathbf{A} \sim \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}_h. \tag{4.3.3.5}$$

Thus, the set $\mathcal{S}(i)$ can be visualized as a subset of the nodes of the tensor-product mesh.

- ◁ ■ $\hat{=}$ node i
- $\hat{=}$ set $\mathcal{S}(i)$ of nodes
- $\hat{=}$ nodes $\notin \mathcal{S}(i)$

▶ The set $\mathcal{S}(i)$ comprises the four neighbors of a node in horizontal and vertical direction. ┘

EXAMPLE 4.3.3.6 (Anisotropic diffusion matrix) Next we consider the matrix \mathbf{A} from Ex. 4.1.1.24, (4.1.1.30) with $\alpha = 1, 0 < \beta \ll 1$, which has the 5-point difference stencil representation

$$\mathbf{A} \sim \begin{bmatrix} 0 & -\beta & 0 \\ -\alpha & 2(\alpha + \beta) & -\alpha \\ 0 & -\beta & 0 \end{bmatrix}_h. \tag{4.3.3.7}$$

- $\hat{=}$ node i
- $\hat{=}$ sets $\mathcal{S}(i), \mathcal{S}(i)^*$ of nodes
- $\hat{=}$ nodes $\notin \mathcal{S}(i)$

Strong coupling in horizontal direction only!

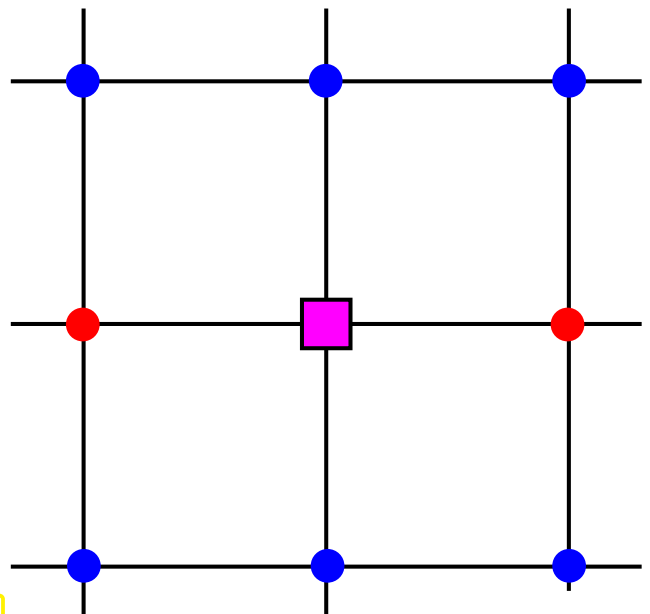


Fig. 184

§4.3.3.8 (“Smoothable” variables) Let us consider a “strongly diagonally dominant” row of $\mathbf{A} \in \mathbb{R}^{N,N}$:

$$|(\mathbf{A})_{i,i}| \geq \sigma \sum_{k \neq i} |(\mathbf{A})_{i,k}|, \quad \sigma > 1, \text{ e.g. } \sigma = \frac{3}{2}, \quad i \in \{1, \dots, N\}. \tag{4.3.3.9}$$

Then a Gauss-Seidel sweep on $\mathbf{A}\vec{\mu} = \mathbf{0}$ will substantially reduce the i -th component of the error. If (4.3.3.9) held true for all i , then a Gauss-Seidel iterative solver would enjoy N -independent (robust) fast linear convergence.

Thus, in the context of AMG, nodes satisfying (4.3.3.9) can be left to the smoother and need not be included in the coarse grid correction; they can all be classified as F-nodes. ┘

§4.3.3.10 (A greedy algorithm) The algorithm from [Stü99] for constructing the coarse-fine splitting (4.3.1.8) of the node set relies on three dynamic sets

$\mathcal{F} \triangleq$ nodes that have already been identified as suitable F-nodes

$\mathcal{C} \triangleq$ nodes that will become C-nodes

$\mathcal{U} \triangleq$ “unassigned nodes”, not yet assigned to either \mathcal{F} or \mathcal{C} .

The algorithm also makes use of the **connectivity measure**

$$\lambda(i) := \#(\mathcal{S}(i)^* \cap \mathcal{U}) + 2 \cdot \#(\mathcal{S}(i)^* \cap \mathcal{F}), \quad i \in \{1, \dots, N\}. \quad (4.3.3.11)$$

It tries to reflect the suitability of a node to be promoted to a C-node. The more strongly connected F-nodes, the larger $\lambda(i)$.

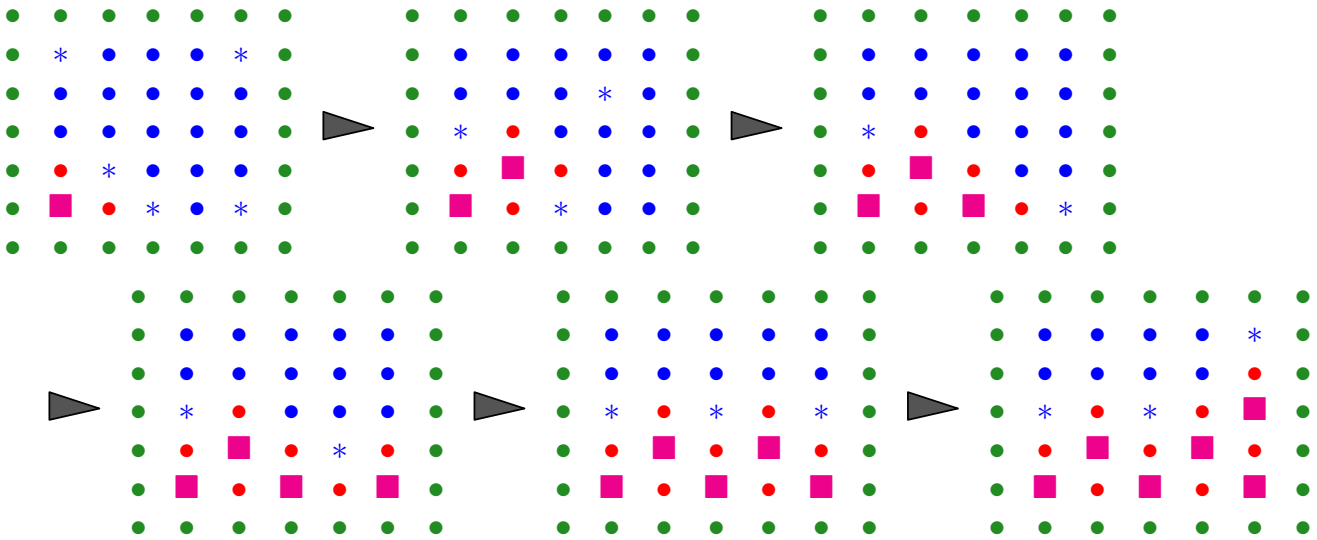
Pseudocode 4.3.3.12: Construction of coarse-fine splitting based on matrix \mathbf{A} [ZAS17]

```

1  [ $\mathcal{C}, \mathcal{F}$ ] = CFsplit(matrix  $\mathbf{A} \in \mathbb{R}^{N,N}$ ) {
2     $\mathcal{F} := \{i \in \{1, \dots, N\} : |(\mathbf{A})_{i,i}| \geq \sigma \sum_{k \neq i} |(\mathbf{A})_{i,k}|\}$ ; // See § 4.3.3.8
3
4     $\mathcal{C} := \emptyset$ ; // no C-nodes yet
5     $\mathcal{U} := \{1, \dots, N\} \setminus \mathcal{F}$ ; // still unassigned nodes
6    Compute  $\lambda(i)$  for all  $i \in \mathcal{U}$ ; // Connectivity measure (4.3.3.11)
7    while (  $\exists j \in \mathcal{U} : \lambda(j) \neq 0$  ) {
8       $i \in \operatorname{argmax}_{j \in \mathcal{U}} \{\lambda(j)\}$ ; // Most suitable C-node
9       $\mathcal{C} \leftarrow \mathcal{C} \cup \{i\}$ ; // Grow set of C-nodes
10      $\mathcal{F} \leftarrow \mathcal{F} \cup \mathcal{S}(i)^*$ ; // Eligible "receiver nodes" as F-nodes
11      $\mathcal{U} \leftarrow \mathcal{U} \setminus \{\{i\} \cup \mathcal{S}(i)^*\}$ ;
12     Update  $\lambda(i)$ ,  $i \in \mathcal{U}$ ; // Necessary for only a small number of  $\lambda(i)$ s
13   }
14    $\mathcal{F} \leftarrow \mathcal{F} \cup \mathcal{U}$ ; // Make all remaining nodes F-nodes
15   return {  $\mathcal{C}, \mathcal{F}$  };
}
```

The algorithm is called greedy because it picks the “next best” C-node. The aggressive relegation of nodes to F-nodes in Line 9 ensures that there is hardly any strong coupling between C-nodes. ┘

EXAMPLE 4.3.3.13 (Coarse-fine splitting for the Poisson matrix) As in Ex. 4.3.3.4 we consider the Poisson matrix $\mathbf{A} \in \mathbb{R}^{N,N}$ as introduced in Ex. 4.1.1.22 and visualize the advance of the greedy algorithm of Code 4.3.3.12 (executions of the **while**-loop body) on the underlying tensor-product mesh.



Symbols: $\bullet \hat{=}$ node in \mathcal{U} , $*$ $\hat{=}$ candidate for next C-node, $\bullet \hat{=}$ F-node $\in \mathcal{F}$, $\blacksquare \hat{=}$ C-node $\in \mathcal{C}$, $\bullet \hat{=}$ smoothable node

We see that about half of the nodes will be promoted to C-nodes. The emerging pattern is called **red-black refinement** in the context of geometric multigrid. ┘

Remark 4.3.3.14. In the construction of a coarse-fine splitting we have to balance two conflicting goals:

1. Keep the number of C-nodes as low a possible.
2. Ensure that every F-node is sufficiently strongly connected to C-nodes (F-nodes have to be “surrounded” by C-nodes).

┘

4.3.4 AMG Prolongation

After having fixed the coarse-fine splitting $\{1, \dots, N\} = \mathcal{C} \cup \mathcal{F}$ the only remaining unspecified component of the AMG two-grid algorithm is the prolongation matrix $\mathbf{P} \in \mathbb{R}^{N, N+H}$ in the form (C-nodes numbered first)

$$\mathbf{P} = \begin{bmatrix} \mathbf{I}_{N_H} \\ \mathbf{P}_F \end{bmatrix}, \quad \begin{array}{l} \mathbf{I}_{N_H} \hat{=} N_H \times N_H \text{ identity matrix, } N_H := \#\mathcal{C}, \\ \mathbf{P}_F \in \mathbb{R}^{N_F, N_H}, \quad N_F := N - N_H = \#\mathcal{F}. \end{array} \quad (4.3.1.10)$$

We resume the considerations of Section 4.3.2, which suggested the use of **A-weighted averaging** to determine \mathbf{P}_F , remember the tentative formula

$$\text{“ } (\mathbf{P}\vec{\eta})_i = -\frac{1}{(\mathbf{A})_{i,i}} \sum_{j \neq i} (\mathbf{A})_{i,j} \cdot \eta\text{-value at node } j \text{ ”, } i \in \mathcal{F}. \quad (4.3.2.9)$$

The remaining challenge is to devise an A-weighted averaging relying on values in C-nodes alone. As indicated by the definition of strong coupling, Def. 4.3.3.2, this will entail taking into account the signs of off-diagonal entries of \mathbf{A} in order to be able to deal with matrices that are not M-matrices. We describe the scheme of “standard interpolation” [TOS00, A.7.2.1] implemented in many AMG packages, e.g. [FJY06].

The first step consists of restricting the summation in (4.3.2.9) to nodes strongly connected to $i \in \mathcal{F}$, that is, to the set $\mathcal{S}(i)$:

$$(\mathbf{P}\vec{\eta})_i := -\frac{1}{(\mathbf{A})_{i,i}} \sum_{j \in \mathcal{S}(i)} (\mathbf{A})_{i,j} (\vec{\eta})_j, \quad i \in \mathcal{F}. \quad (4.3.4.1)$$

The tacit assumption $(\mathbf{A})_{i,i} \neq 0$ for all $i \in \{1, \dots, N\}$ is made throughout.

§4.3.4.2 (Locally preserving constants (LPC)) Let us assume that \mathbf{A} has vanishing row sums, $\mathbf{A}\mathbf{1} = \mathbf{0}$, $\mathbf{1} = [1, \dots, 1]^\top$, which is satisfied, for instance, for the $\mathcal{S}_1^0(\mathcal{M})$ Galerkin matrix for pure Neumann BVPs. In this case the Gauss-Seidel smoother will not be able to effect any correction in the direction of $\mathbf{1}$, which means that $\mathbf{1} \in \mathcal{R}(\mathbf{P})$ is mandatory.

In light of (4.3.1.10), this is equivalent to demanding $\mathbf{P}_F\mathbf{1} = \mathbf{1}$:

$$\mathbf{A}\mathbf{1} = \mathbf{0} \implies \mathbf{P}_F\mathbf{1} = \mathbf{1}. \quad (4.3.4.3)$$

To repeat, interpreting \mathbf{A} as the $\mathcal{S}_1^0(\mathcal{M})$ finite-element Galerkin matrix of a scalar second-order elliptic BVP, the property $\mathbf{A}\mathbf{1} = \mathbf{0}$ will hold for the pure Neumann problem [NumPDE Ex. 1.8.0.10].

The requirement (4.3.4.3) is extended to general matrices in the form of a localized version:

$$(\mathbf{A}\mathbf{1})_\ell = 0 \implies (\mathbf{P}_F\mathbf{1})_\ell = 1. \quad (4.3.4.4)$$

We call this property of \mathbf{P} a “local preservation of constants”.

In light of (4.3.4.4) we convert (4.3.4.1) by *re-weighting* to

$$(\mathbf{P}\vec{\eta})_i := -\frac{\alpha_i}{(\mathbf{A})_{i,i}} \sum_{j \in \mathcal{S}(i)} (\mathbf{A})_{i,j} (\vec{\eta})_j, \quad \alpha_i := \frac{\sum_{k \neq i} (\mathbf{A})_{i,k}}{\sum_{k \in \mathcal{S}(i)} (\mathbf{A})_{i,k}}, \quad i \in \mathcal{F}. \quad (4.3.4.5)$$

Obviously, $\vec{\eta} = \mathbf{1}$ and $(\mathbf{P}\vec{\eta})_i = 1$ satisfies this equation, if $(\mathbf{A}\mathbf{1})_i = 0$:

$$(\mathbf{A})_{i,i} = -\frac{\sum_{k \neq i} (\mathbf{A})_{i,k}}{\sum_{k \in \mathcal{S}(i)} (\mathbf{A})_{i,k}} \cdot \sum_{j \in \mathcal{S}(i)} (\mathbf{A})_{i,j} = -\sum_{k \neq i} (\mathbf{A})_{i,k}.$$

┘

§4.3.4.6 (Positive and negative connections) Heuristic arguments [TOS00, A.4.2.2] suggest a distinction between negative and positive off-diagonal matrix entries. Keeping in mind the local preservation of constants, we refine (4.3.4.5) into¹

$$(\mathbf{A})_{i,i} (\mathbf{P}\vec{\eta})_i := -\alpha_i \sum_{j \in \mathcal{S}(i)} (\mathbf{A})_{i,j}^- (\vec{\eta})_j - \beta_i \sum_{j \in \mathcal{S}(i)} (\mathbf{A})_{i,j}^+ (\vec{\eta})_j, \quad (4.3.4.7)$$

$$\alpha_i := \frac{\sum_{k \neq i} (\mathbf{A})_{i,k}^-}{\sum_{k \in \mathcal{S}(i)} (\mathbf{A})_{i,k}^-}, \quad \beta_i := \frac{\sum_{k \neq i} (\mathbf{A})_{i,k}^+}{\sum_{k \in \mathcal{S}(i)} (\mathbf{A})_{i,k}^+}$$

for $i \in \mathcal{F}$, which still ensures (4.3.4.3): if $(\mathbf{A}\mathbf{1})_i = 1$, then

$$\begin{aligned} (\mathbf{P}_F\mathbf{1})_i &= \frac{1}{(\mathbf{A})_{i,i}} \cdot \left(-\frac{\sum_{k \neq i} (\mathbf{A})_{i,k}^-}{\sum_{k \in \mathcal{S}(i)} (\mathbf{A})_{i,k}^-} \cdot \sum_{j \in \mathcal{S}(i)} (\mathbf{A})_{i,j}^- - \frac{\sum_{k \neq i} (\mathbf{A})_{i,k}^+}{\sum_{k \in \mathcal{S}(i)} (\mathbf{A})_{i,k}^+} \cdot \sum_{j \in \mathcal{S}(i)} (\mathbf{A})_{i,j}^+ \right) \\ &= \frac{1}{(\mathbf{A})_{i,i}} \cdot \left(-\sum_{k \neq i} (\mathbf{A})_{i,k}^- - \sum_{k \neq i} (\mathbf{A})_{i,k}^+ \right) = 1. \end{aligned}$$

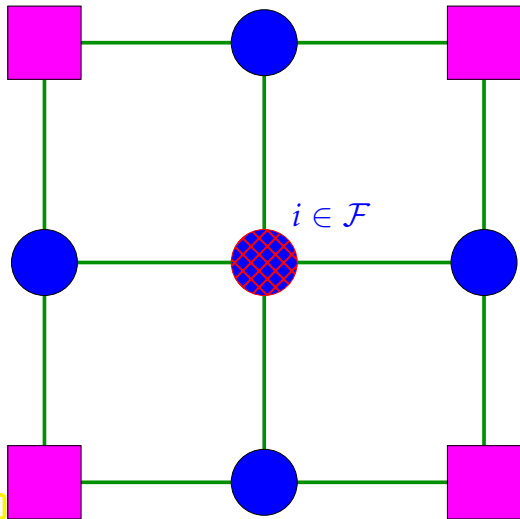
¹Throughout, collapsing sums will be set to zero, even after “division by zero”.

Above we wrote

$$(\mathbf{A})_{ij}^+ := \begin{cases} (\mathbf{A})_{ij} & \text{if } (\mathbf{A})_{ij} > 0, \\ 0 & \text{else,} \end{cases}, \quad (\mathbf{A})_{ij}^- := \begin{cases} (\mathbf{A})_{ij} & \text{if } (\mathbf{A})_{ij} < 0, \\ 0 & \text{else.} \end{cases},$$

This realizes a purely *local prolongation* based on node connections corresponding to edges in the matrix adjacency graph of \mathbf{A} . However, (4.3.4.7) still fails to make sense, because the right-hand side sums still covers F-nodes, for which $(\vec{\eta})_j$ is not defined. ┘

§4.3.4.8 (Reduction to C-nodes)



We may face the situation that an F-node i is connected to only other F-nodes: $\mathcal{S}(i) \subset \mathcal{F}$.

- ◁ ■ $\hat{=}$ C-nodes
- $\hat{=}$ F-nodes
- $\hat{=}$ edge of matrix graph of \mathbf{A}



Idea: Prolongate values into i from strongly connected C-neighbors of F-nodes $\in \mathcal{S}(i)$.

Thus we can remove the ill-defined values $(\vec{\eta})_j$ from the right-hand side of (4.3.4.7),

$$(\mathbf{P}\vec{\eta})_i := -\frac{1}{(\mathbf{A})_{ii}} \left(\alpha_i \sum_{j \in \mathcal{S}(i)} (\mathbf{A})_{ij}^- (\vec{\eta})_j + \beta_i \sum_{j \in \mathcal{S}(i)} (\mathbf{A})_{ij}^+ (\vec{\eta})_j \right), \quad i \in \mathcal{F}, \tag{4.3.4.7}$$

by pursuing the following strategy. In (4.3.4.7) for $j \in \mathcal{S}(i) \cap \mathcal{F}$ replace

$$(\vec{\eta})_j \leftarrow (\vec{\eta}')_j := \frac{1}{(\mathbf{A})_{jj}} \hat{\alpha}_j \cdot \sum_{k \in \mathcal{S}(j) \cap \mathcal{C}} (\mathbf{A})_{jk} (\vec{\eta})_k, \quad \hat{\alpha}_j := \frac{\sum_{k \neq j} (\mathbf{A})_{jk}}{\sum_{k \in \mathcal{S}(j) \cap \mathcal{C}} (\mathbf{A})_{jk}}. \tag{4.3.4.9}$$

Note that we have employed the same re-weighting approach as in the transition from (4.3.2.9) to (4.3.4.1) in order to ensure the local preservation of constants.

Then define $(\mathbf{P}\vec{\eta})_i$ through

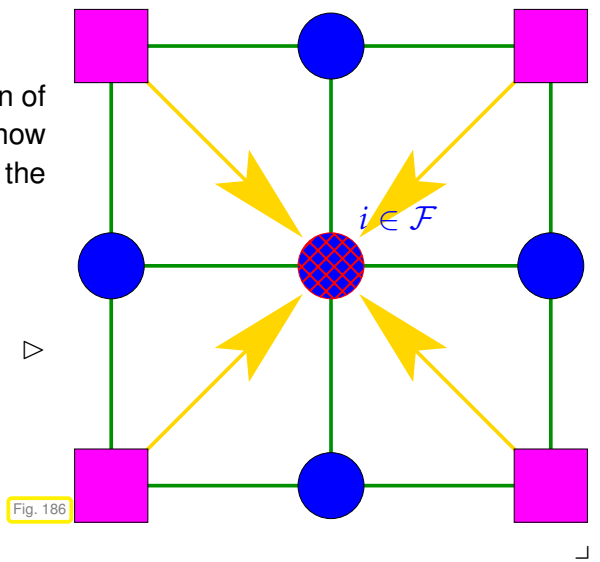
$$(\mathbf{P}\vec{\eta})_i := -\frac{1}{(\mathbf{A})_{ii}} \left(\alpha_i \left(\sum_{j \in \mathcal{S}(i) \cap \mathcal{C}} (\mathbf{A})_{ij}^- (\vec{\eta})_j + \sum_{j \in \mathcal{S}(i) \cap \mathcal{F}} \frac{(\mathbf{A})_{ij}^-}{(\mathbf{A})_{jj}} \cdot \hat{\alpha}_j \cdot \sum_{k \in \mathcal{S}(j) \cap \mathcal{C}} (\mathbf{A})_{jk} (\vec{\eta})_k \right) + \right. \tag{4.3.4.10}$$

$$\left. \hat{\beta}_i \sum_{j \in \mathcal{C} \cap \mathcal{S}(i)} (\mathbf{A})_{ij}^+ (\vec{\eta})_j \right), \quad \hat{\beta}_i := \frac{\sum_{k \neq i} (\mathbf{A})_{ik}^+}{\sum_{j \in \mathcal{C} \cap \mathcal{S}(i)} (\mathbf{A})_{ij}^+}.$$

In this formula, only values in C-nodes are accessed and those are available from components of $\vec{\eta}$.

In the situation of Fig. 185 during prolongation ($\hat{=}$ application of \mathbf{P} to the vector of values on the C-nodes) F-node i would now receive values from four C-nodes located at distance 2 in the matrix graph.

- $\hat{=}$ C-nodes
- $\hat{=}$ F-nodes
- $\hat{=}$ edge of matrix graph of \mathbf{A}
- $\hat{=}$ transfer of values during prolongation



Bibliography

- [BY93] F. Bornemann and H. Yserentant. “A basic norm equivalence for the theory of multilevel methods”. In: *Numer. Math.* 64.4 (1993), pp. 455–476 (cit. on p. 391).
- [Hac94] Wolfgang Hackbusch. *Iterative solution of large sparse systems of equations*. Vol. 95. Applied Mathematical Sciences. New York: Springer-Verlag, 1994, pp. xxii+429 (cit. on p. 370).
- [RS87] J. Ruge and K. Stüben. “Algebraic multigrid”. In: *Multigrid methods*. Ed. by S. McCormick. Frontiers in Applied Mathematics. Philadelphia: SIAM, 1987. Chap. 4, pp. 73–130 (cit. on p. 358).
- [SG06] Olaf Schenk and Klaus Gärtner. “On fast factorization pivoting methods for sparse symmetric indefinite systems”. In: *Electron. Trans. Numer. Anal.* 23 (2006), pp. 158–179 (cit. on p. 365).
- [Stü99] K. Stüben. *Algebraic Multigrid (AMG): An introduction with applications*. Report 70. St. Augustin, Germany: GMD – German National Research Center for Information Technology, 1999 (cit. on pp. 370, 409).
- [TOS00] U. Trottenberg, C.W. Oosterlee, and A. Schüller. *Multigrid*. London: Academic Press, 2000 (cit. on pp. 358, 390, 410, 411).
- [Wag99] C. Wagner. “Introduction to Algebraic Multigrid”. 1999 (cit. on p. 358).
- [Xu92] J. Xu. “Iterative methods by space decomposition and subspace correction”. In: *SIAM Review* 34 (1992), pp. 581–613 (cit. on p. 391).

Chapter 5

Reduced Bases Methods (RBM)



Supplementary literature. [[HRS16](#); [QMN16](#)] are two textbooks on reduced basis methods, both with a focus on algorithms. This chapter mainly follows these works.

5.1 Parameterized Boundary Value Problems

5.1.1 Coefficients as Parameters

§5.1.1.1 (Stationary heat conduction in composite solids)

┘

§5.1.1.2 (Quantity of interest (QoI): Heat flux)

┘

§5.1.1.3 (Unvertainty quantification)

┘

5.1.2 Parameter-Dependent Domains

§5.1.2.1 (Model problem with variable computational domain)

┘

§5.1.2.2 (Pullback to reference domain)

┘

5.1.3 Abstract Framework

§5.1.3.1 (Parameter-dependent variational problems)

┘

§5.1.3.2 (Reduced Bases Methods: Objectives and Policy)

┘

§5.1.3.3 (π -Uniformity assumptions)

┘

§5.1.3.4 (Trusted approximation: “Truth solution”)

┘

5.2 Reduced Bases Methods: Ideas and Algorithms

5.2.1 Prelude: Polynomial Interpolation

5.2.2 Projected Variational Problem

5.2.3 Generation of Reduced Bases

5.2.3.1 Proper Orthogonal Decomposition (POD)

5.2.4 Special Case: Separable Decomposition

§5.2.4.1 ($O(N)$ trap)

┘

§5.2.4.2 (Separable variational problems)

┘

§5.2.4.3 (Setup and evaluation phases)

┘

5.3 Error Estimation

5.3.1 Residual-Based Estimator

5.3.2 Computation of Residual Norm

5.3.3 Lower Bound for $\gamma_h(\pi)$

5.4 Separable Approximation

5.4.1 Interpolation on Parameter Space

5.4.2 Adaptive Cross Approximation (ACA)

Main Index: Terms and Keywords

- L^2 -projections, 134
- \mathcal{H}^2 -matrix, 249
- \mathcal{H} -matrix, 215
- z -transform, 269
- \mathcal{H}^2 -matrices, 249
- (Layer) potentials, 42
- (Successive) subspace correction method, 357
- 1D Quadrature formula/quadrature rule, 100
- A-stability
 - of multi-step methods, 302
- A-Stability of linear multi-step methods [DB02, Sect. 7.2.2], 302
- A-stable, 302
- Abel integral equation, 277
- Abel integral operator, 277
- Abstract admissibility condition, 198
- acoustic scattering, 281
- acoustic wave equation, 281
- adjoint
 - differential operator, 35
- admissibility condition, 154, 215
 - abstract, 198
- admissibility measure, 171, 173, 179
- Admissibility of index sets, 183
- admissible
 - index sets, 183
- algebraic convergence, 104
- alternating directional splitting, 193
- analytic extension, 107
- analytic function, 107
- Analyticity of a function in \mathbb{C} , 107
- arclength derivative, 64
- arclength parameterization, 114
- Ass: ℓ -uniform linear convergence of multigrid V-cycle, 373
- Ass: a defines an inner product, 361
- Ass: Analytic parameterization, 112
- Ass: Analyticity of local parameterizations, 126
- Ass: Availability of low-rank factor matrices, 216
- Ass: Binary cluster trees, 216
- Ass: Connected domains, 16
- Ass: Data in procedural form, 83
- Ass: Mesh compatible with partition, 139
- Ass: Mesh hierarchy, 370
- Ass: Near-field diagonal blocks, 241
- Ass: Polynomial growth of F , 295
- Ass: Properties of linear multi-step method underlying CQ, 305
- Ass: Properties of transfer function, 284
- Ass: Rank- q separable approximation on admissible boxes, 178
- Ass: Requirements on implicit RK-SSM for CQ, 318
- Ass: Structure of result matrix, 229
- Ass: Uniform distribution, 155
- Ass: Unisolvence of interpolation nodes, 163
- assembly
 - of Galerkin matrices, 88
- Asymptotic convergence of quadrature rules, 104
- attenuation equation, 276
- average
 - of traces, 58
- barycentric coordinate functions, 125
- barycentric interpolation formula, 206
- BDF-2, 299
- Bernstein ellipse, 108, 174
- BIE $\hat{=}$ boundary integral equations, 48
- bilinear form, 23
 - elliptic, 131
- BIO $\hat{=}$ boundary integral operator, 48
- BIO $\hat{=}$ boundary integral equations, 67
- block
 - of a hierarchical matrix, 216
- Block tree underlying a hierarchical matrix, 217
- boundary conditions
 - mirror symmetry, 20
- boundary element space, 77
- boundary integral equations, 48, 67
- boundary integral operators, 48
- Boundary integral operators for $-\Delta$, 61
- boundary value problem

- elliptic, 340
- boundary-value problem
 - two-point, 341
- bounding box, 191
 - of a cluster, 191
- Bounding box of an index set, 183
- Bounding boxes of clusters, 191
- Bromwich integral, 264
- Butcher scheme, 317
- cardinal basis property, 367
- cardinal function, 164
- Cauchy integral formula, 289, 309
- Cauchy integral theorem, 264
- Cauchy product, 269, 291
- causal function, 257
- Causal functions, 257
- causal polynomially bounded function, 262
- Causal polynomially bounded functions, 262
- causal sequence, 260
- Cea's lemma, 131
- CG, 352
- channel, 258
 - time-invariant, 258
- characteristic function, 81
- characteristic polynomial, 299
- charge density, 21
- Chebyshev interpolation, 86, 174
 - error estimates, 172
- Chebyshev nodes, 165, 173
- circulant augmentation, 272
- circulant matrix, 270
- Clenshaw-Curtis quadrature rule, 102
- cluster, 191
- cluster bases
 - for \mathcal{H}^2 -matrix, 249
- Cluster tree, 191
- cluster tree, 191
- co-axial cable, 278
- co-normal trace, 56
- coarse-fine splitting, 382
- coarse-grid correction, 368
- coarse-grid correction, 366
- coefficient vector, 77
- collocation
 - of a kernel, 156
- column cluster bases
 - for \mathcal{H}^2 -matrix, 249
- column tree, 215
- comb function, 292
- compatibility conditions
 - for $H^1(\Omega)$, 19
- complex contour integral, 263
- Concepts connected with trees, 189
- configuration space, 20
- conjugate gradient method, 352
- consistent, 349
- convergence
 - algebraic, 104
 - exponential, 104
- convolution, 39, 255
 - associativity, 256
 - of causal functions, 258
 - of distributions, 256
 - of operators, 257
 - of sequences, 259
- convolution equation, 258, 285
 - discrete, 260
- Convolution of functions in \mathbb{R}^d , 39
- Convolution of sequences, 259
- Convolution on the real line, 255
- convolution quadrature, 284
- convolution quadrature weights, 284
- Corollary: Associativity of convolution, 256
- Corollary: Cauchy differentiation formula, 290
- Corollary: Continuous, piecewise- C^1 functions in $H^{\frac{1}{2}}(\Gamma)$, 52
- Corollary: Convergence of stationary linear iterations, 349
- Corollary: Direct 1st-kind variational BIE for transmission problem, 143
- Corollary: Embedding of $H^{\frac{1}{2}}(\Gamma)$, 51
- Corollary: Embeddings of boundary element spaces, 80
- Corollary: Green's function integral representations, 46
- Corollary: Mapping properties of Dirichlet trace, 49
- Corollary: Mapping properties of the Newton potential, 39
- Corollary: Properties of orthogonal projections, 360
- Corollary: Two-grid method as stationary linear iteration, 369
- Coulomb force, 14, 32
- coupling matrix
 - for \mathcal{H}^2 -matrix, 249
- curl operator, 15
- curl-free, 15
- curved polygon, 31, 78
- curvilinear polyhedron, 31
- d.o.f. mapper, 89

- data sparse, 158
- decay conditions, 27
- delta distribution, 35
- density, 49
- density unknowns, 74
- DFT, 315
- DFT (discrete Fourier transform), 271
- diffusion coefficient, 340
- Dirichlet BVP, 68, 83
- Dirichlet trace operator, 49
- Dirichlet trace space, 49
- discrete Fourier transform, 315
- discrete variational problem, 76, 77
- distributions, 35
- Double layer potential, 44
- Doxygen, 75
- Dual norm for source charge distributions, 29
- duality, 29, 55

- eddy current model, 278
- edge, 120
- edge set
 - of a tree, 189
- Electrostatic field energy [\[NumPDE Eq. \(1.2.2.6\)\]](#), 14
- elliptic, 131
 - bilinear form, 69
- elliptic boundary value problem, 340
- energy norm
 - for Neumann trace space, 53
- equilibrium principle, 22
- error propagation matrix, 349
- error recursion
 - for stationary linear iterations, 349
- expand-from-cluster, 250
- explicit midpoint method, 298
- exponential convergence, 104
- exterior Dirichlet problem, 149

- face, 120
- Far field, 181
- far field, 180, 183
- Far-field blocks of index pairs, 184
- FFT (fast Fourier transform), 271
- fill-in, 347
- filter, 260
- finite-difference grid, 346
- First-kind BIE, 71
- forward transformation, 251
- Fourier matrix, 271
- Fourier transform, 261
 - discrete, 315
- full approximation storage (FAS), 376
- Fundamental solution, 35
- fundamental solution, 35, 45

- Galerkin approximation, 76
- Galerkin discretization, 75, 342
- Galerkin matrix, 342
- Gauss quadrature, 101
 - generalized, 103
- Gauss-Seidel method, 348
 - non-linear, 378
- Gaussian elimination, 347
- generating function, 290, 309
- global shape functions (GSF), 124
- Gram determinant, 31
- Green's first formula, 24
- Green's function, 45
 - for disk, 46
 - for half space, 47
- grid, 346
- grid function, 346

- h-refinement, 131
- h-uniform convergence, 369
- half space, 47
- hat function, 81
- Hierarchical matrix, 215
- hierarchical matrix, 215
- Hilbert BEM library, 74
- Hilbert space of square integrable functions
 - [\[NumPDE Def. 1.3.2.3\]](#), 17
- horizontal concatenation
 - of matrices, 225

- idempotent, 360
- impedance boundary conditions, 279
- impedance condition, 280
- Implicit Euler convolution quadrature (IE-CQ), 288
- implicit Euler method, 287
- Implicit Runge-Kutta single-step method,
 - [\[NumCSE Def. 12.3.3.1\]](#), 317
- impulse response, 260
- initial guess, 348
- integration by parts
 - multidimensional, 24
- interpolation, 163
 - bi-directional, 166
 - polynomial, 86
- interpolation conditions, 163
- interpolation nodes, 163
- interpolation operator, 163

- intrinsic norm, 52
- irrotational, 15
- iteration error, 349
- Jacobi method, 349
- Jacobian, 15
- jump
 - of traces, 58
- jump relations, 60
- kernel
 - asymptotically smooth, 169
 - of an integral operator, 38, 42, 62
- Kernel collocation matrix, 156
- kernel collocation matrix, 156
- kernel function, 156
- Krylov space, 352
- Krylov subspace method, 214
- Lagrange polynomials, 164
- Lagrangian finite elements
 - bilinear, 345
- Lagrangian multiplier, 84
- Laplace inversion formula, 264
- Laplace transform, 262
 - inverse, 264
- Laplacian, 37
 - spherical coordinates, 36
- layer potential, 48
- leaf
 - of a tree, 189
- Lebesgue constant, 176
- Legendre polynomials, 82
- Lemma: A sign condition for A-stability of multi-step methods, 303
- Lemma: Arclength integration by parts, 64
- Lemma: Chebychev interpolation error estimate, 173
- Lemma: Circulant augmentation of Toeplitz matrix, 273
- Lemma: Convolution quadrature weights are Taylor expansion coefficients, 290
- Lemma: Convolution quadrature weights for multi-step CQ, 309
- Lemma: Ellipticity of c , 143
- Lemma: Error propagation operator of coarse grid correction, 386
- Lemma: Exact quadrature by equidistant trapezoidal rule, 314
- Lemma: Fundamental solution for $L := -\Delta + s^2$, 282
- Lemma: Generalized orthogonal polynomials, 103
- Lemma: Lower bound for T , 364
- Lemma: Nesting of meshes implies nesting of finite element spaces, 366
- Lemma: Orthogonal projections onto orthogonal subspaces, 362
- Lemma: Pointwise estimate for convolution, 268
- Lemma: Properties of I , 163
- Lemma: Quadrature error and best-approximation error, 106
- Lemma: Representation of low-rank matrices, 159
- Lemma: Smoothness of double layer potential, 45
- Lemma: Smoothness of single layer potential, 43
- Lemma: Symmetric positive definite Galerkin matrices, 342
- Lemma: Symmetric successive subspace correction, 375
- Lemma: Upper bound for T , 364
- Lemma: Variation of constants formula, 286
- level
 - of mesh, 370
 - of the nodes of a tree, 190
- Level of nodes of tree, 190
- Linear interpolation operator, 163
- Linear multi-step method, 299
- Linear variational problem, 23
- load vector, 342
- local parameterization, 82
- local shape functions (LSF), 124
- local→global index map, 89
- LU-dcomposition, 239
- LU-decomposition, 239
- magneto-quasistatic model, 278
- marching on in time (MOT), 261
- Matrix graph, 383
- matrix norm, 158
- mesh, 120, 341
 - of a curve, 78
- mesh hierarchy, 370
- Mesh/partitioning of a curve, 78
- Meshwidth, 132
- meshwidth, 132
- method of lines, 322
- Minimal angle, 132
- mirror symmetry, 20
- MOT = marching on in time, 261
- Multivariate polynomials, 122
- Near field, 181
- near field, 180

- Nested finite element meshes, 366
- nested iteration, 372
- nested meshes, 366
- Neumann BVP, 69, 73, 83
- Neumann trace, 52
- Neumann trace operator, 52
- Neumann trace space, 53
- Newton potential, 38
- nodal interpolation operator, 134
- nodal interpolation operators, 123
- node
 - quadrature, 101
- node set
 - of a tree, 189
- nodes
 - of a mesh, 79
- non-linear variational problem, 376
- non-local operator, 148
- Non-local operators on \mathbb{R}^N , 149
- Non-local operators on function spaces, 148
- normal component trace, 19
- numerical quadrature, 100

- offset function, 138
- Operational calculus, 267
- operational calculus, 267
- operator
 - non-local, 148
- order
 - of quadrature formula, 101
- Order of a quadrature rule, 101
- orthogonal projection, 360
- Orthogonal projection in finite dimensions, 360

- panels
 - of a mesh, 79
- parameterization, 78
- Pardiso, 347
- partitioning
 - of a curve, 78
- PEC boundary conditions, 20
- Piecewise Sobolev spaces on Γ , 132
- Plancherel theorem, 261
- plane wave, 281
- point charge, 32
- Poisson equation, 341
- Poisson integral formula, 47
- Poisson matrix, 344
- polygon
 - curved, 78
- polynomial interpolation, 86
- polynomials
 - degree, 122
 - multivariate, 122
- post-smoothing, 369
- potential
 - electrostatic, 16
- pre-smoothing, 369
- preconditioners, 245
- procedural form, 83
- prolongation matrix, 367
- pullback, 79
- Pullback from a curve, 79

- quadratic minimization problem, 341
- quadrature error, 104
- quadrature formula
 - order, 101
- quadrature node, 101
- quadrature weight, 101

- Rank of a matrix, 158
- reaction coefficient, 341
- Real [analytic functions](#), 107
- real analytic, 107
- recurrence relation, 301
- reference interval, 82
- reference shape function, 82
- reference shape functions, 82
- Region of stability for linear multi-step method, 302
- regularization
 - of a BIE, 150
- relative distance
 - of panels, 119
- relaxation parameter, 349
- residual linear form, 357
- restrict-to-cluster, 210, 250
- root
 - of a tree, 189
- Rotation invariance, 36
- rotation operator, 15
- rotation-invariant, 36
- row cluster bases
 - for \mathcal{H}^2 -matrix, 249
- row tree, 215
- Runge-Kutta single-step method, 317

- sampling, 259
- scattered field, 281
- Sectorial transfer function/parabolic symbol, 324
- separation of variables, 36
- shape function
 - reference, 82

- shape functions
 - global, 80, 124
 - local, 81
- shape regularity, 132
- shape regularity measure, 343
- shift operator, 293
- Single layer potential, 43
- single layer potential, 42
- singularity, 33
- skin effect, 280
- smoother, 368
- Sobolev norm, 132
- Sobolev space
 - higher-order, 132
 - on surfaces, 61
- Sobolev space $H^1(\Omega)$, [NumPDE Def. 1.3.4.8], 17
- solid-angle formula, 150
- sons
 - in a tree, 189
- sound-soft, 281
- source charge distribution, 22
- Space of function with square-integrable Laplacian, 54
- sparse matrix, 343
- sparsity measure
 - of a block partition, 208
- Sparsity measure of block partition, 208
- spectral condition number, 353
- spherical coordinates, 36, 130
- stability
 - of subspace splitting, 363
- stage
 - of RK-SSM, 317
- stage form
 - of RK-SSM, 318
- stationary linear iteration, 349
- stationary linear iterations
 - error recursion, 349
- stencil, 346
- strengthened Cauchy-Schwarz inequality, 363
- Strongly coupled nodes, 387
- sub-tree, 190
- Sub-trees, 190
- subspace correction method, 356, 359
- support
 - of a function, 148
- surface gradient, 66
- surface integral, 31
- surface mesh, 121
- symmetric positive definite (s.p.d.), 240
- tangent vector, 64
- tangential component trace, 19
- Taylor expansion, 15, 161
- tensor product polynomials, 110, 165
- tensor-product interpolation, 165
- tensor-product polynomial interpolation operator, 165
- Tensor-product polynomials, 110
- tensor-product quadrature, 110
- tent function, 81, 342
- test space
 - for Galerkin discretization, 76
- Theorem: z -Transform and discrete convolution, 269
- Theorem: “Higher” continuity of BIOs, 62
- Theorem: A stable and consistent linear MSM is convergent, 304
- Theorem: Analyticity of Laplace transforms, 262
- Theorem: Asymptotic interpolation/projection error estimates, 135
- Theorem: Best low rank approximation, 223
- Theorem: Cauchy integral formula, 289
- Theorem: Cauchy integral theorem, 264
- Theorem: Cea’s lemma, 131
- Theorem: Characterization of Cauchy data, 68
- Theorem: Chebychev interpolation of analytic functions, 174
- Theorem: Compatibility conditions for piecewise smooth functions in $H^1(\Omega)$, 19
- Theorem: Continuity of boundary integral operators, 61
- Theorem: Continuity of single layer potential in energy (trace) spaces, 57
- Theorem: Continuity of the double layer potential in energy trace spaces, 57
- Theorem: Continuity of the Neumann trace on $H(\Delta, \Omega)$, 54
- Theorem: Continuity of the single layer potential, 43
- Theorem: Convergence of geometric multigrid, 371
- Theorem: Convergence of IE-CQ, 297
- Theorem: Convergence of the CG method, 354
- Theorem: Convergence of the PCG method, 375
- Theorem: Convolution theorem for Fourier transform, 266
- Theorem: Convolution theorem for Laplace transform, 267
- Theorem: Decay of Newton potential, 39
- Theorem: Diagonalization of circulant matrices, 271

- Theorem: Differentiation formula for Laplace transform, 265
- Theorem: Dimensions of BE spaces on curves, 80
- Theorem: Dimensions of BE spaces on triangulated surfaces, 123
- Theorem: Economical QR-decomposition, 223
- Theorem: Electric fields are irrotational/curl-free, 15
- Theorem: Ellipticity of a_V in 2D, 70
- Theorem: Ellipticity of a_V in 3D, 69
- Theorem: Ellipticity of a_W , 71
- Theorem: Embedding of $H^{-\frac{1}{2}}(\Gamma)$, 55
- Theorem: Equivalence theorem for quadratic minimization problems, 23
- Theorem: Existence and uniqueness of energy minimizing potentials, 23
- Theorem: Existence of electrostatic potential, 16
- Theorem: Exponential convergence of trigonometric interpolation for analytic interpolands, [NumCSE Thm. 6.5.3.14], 314
- Theorem: First Dahlquist barrier [DB02, Thm. 7.16], 304
- Theorem: Gauss(-Legendre) quadrature, 101
- Theorem: Generalized Gauss quadrature, 104
- Theorem: Green's first formula, 24
- Theorem: Green's second formula, 30
- Theorem: Growth of solutions of linear recurrence relations, [DB02, Thm. 3.40], 301
- Theorem: Higher order trace theorem, 134
- Theorem: Independence of Galerkin solution of choice of basis, 77
- Theorem: Integral representation formula, 41
- Theorem: Integral representation formula for 3D exterior domains, 41
- Theorem: Integral representation of a_W in 2D, 66
- Theorem: Integral representation of a_W in 3D, 66
- Theorem: Inverse Laplace transform, 264
- Theorem: Jump relations for layer potentials, 60
- Theorem: Jump representation formula, 58
- Theorem: LU-decomposition of s.p.d. matrices, 240
- Theorem: Main approximation theorem for $\mathcal{S}_p^{-1}(\mathcal{G})$, 133
- Theorem: Main approximation theorem for $\mathcal{S}_p^0(\mathcal{G})$, 133
- Theorem: Main convergence theorem for SSC, 363
- Theorem: Multigrid = multi-level subspace correction, 371
- Theorem: Multiplicative trace inequality, 50
- Theorem: Pointwise estimate for convolution II, 268
- Theorem: Polynomial approximation of analytic functions, 108
- Theorem: Positivity of Clenshaw-Curtis weights, 102
- Theorem: Preconditioners from stationary linear iterations, 375
- Theorem: Properties of discrete convolution of sequences, 259
- Theorem: Quadrature error estimate for integrands with finite smoothness, 106
- Theorem: Second Dahlquist barrier, [DB02, Thm. 7.36], 303
- Theorem: Singular Value Decomposition (SVD), 222
- Theorem: SSC with orthogonal subspaces, 362
- Theorem: Uniqueness of fundamental solutions, 36
- Theorem: Validity of 1st-kind indirect BIE for Dirichlet problem, 73
- Theorem: Validity of 1st-kind indirect BIE for Neumann problem, 73
- Theorem: Young's inequality for convolutions, 256
- Theorem: $L^2(\Gamma)$ -duality between $H^{\frac{1}{2}}(\Gamma)$ and $H^{-\frac{1}{2}}(\Gamma)$, 55
- tiling, 180
- time-invariant channel, 258
- Toeplitz matrix, 272
- tomography, 276
- trace
 normal component, 19
 tangential component, 19
- Trace operator, 42
- trace operator, 48
- transfer function, 267
- transfer matrix, 249
- translation-invariant, 36
- transmission conditions, 140
- transmission problems, 139
- trapezoidal rule, 101, 314
- Tree, 189
- tree, 189
- trial space
 for Galerkin discretization, 76
- Triangular planar mesh/triangulation, 120
- Triangular surface mesh/surface triangulation, 121

triangulation, 120
trigonometric polynomial, 314
triple-factor low-rank factorization, 246
two-point boundary-value problem, 341

uniform cone condition, 115
uniformly positive definite, 340
unisolvence
 of interpolation nodes, 163

V-cycle, 371
 of multigrid, 376
variation of constants formula
 extended, 335
variational crimes, 136
variational formulation, 341
variational problem
 discrete, 76, 77
vertex, 120

vertex set
 of a tree, 189
vertical concatenation
 of matrices, 225
virtual work principle, 22
volume integral operator, 38
volume potential, 38

W-cycle, 371
weight
 quadrature, 101
weight function, 103
weights
 convolution quadrature, 284

Young's inequality
 for convolutions, 256

z-transform, 269
zero-stable, 302

Abbreviations and Acronyms

BDF $\hat{=}$ backward difference formula, 298

BEM $\hat{=}$ boundary element method, 11

BIE $\hat{=}$ boundary integral equation , 48

BIO $\hat{=}$ boundary integral operator , 60

BLF $\hat{=}$ bilinear form , 23

BVP $\hat{=}$ boundary value problem , 11

BVP $\hat{=}$ boundary-value problem, 340

c.i.t. $\hat{=}$ continuous-in-time, 292

CEQ $\hat{=}$ convolution equation, 258

CGC $\hat{=}$ coarse grid correction, 366

DFT $\hat{=}$ discrete Fourier transform, 315

EDP $\hat{=}$ exterior Dirichlet boundary value problem, 149

EPM $\hat{=}$ error propagation matrix, 349

FAS $\hat{=}$ full approximation storage, 376

FOCQ $\hat{=}$ fast and oblivious convolution quadrature, 322

FS $\hat{=}$ fundamental solution , 32

GalM $\hat{=}$ Galerkin matrix , 77

GE $\hat{=}$ Gaussian elimination, 347

GSF $\hat{=}$ global shape function , 80

IRK-SSM $\hat{=}$ implicit Runge-Kutta single-step method, 317

LF $\hat{=}$ linear form , 23

LPC $\hat{=}$ local preservation of constants, 391

LSE $\hat{=}$ linear system of equations, 151

LSE $\hat{=}$ linear system of equations , 11

LSF $\hat{=}$ local shape function , 81

MSCQ $\hat{=}$ multi-step convolution quadrature, 297

MSM $\hat{=}$ (linear) multi-step method, 297

NI $\hat{=}$ nested iteration, 372

NLVP $\hat{=}$ non-linear variational problem, 376

PDE $\hat{=}$ partial differential equation , 11

pwc $\hat{=}$ piecewise constant , 81

pwl $\hat{=}$ piecewise linear , 79

QF $\hat{=}$ quadratic functional or quadrature formula
(\rightarrow Def. 1.4.3.41), 100

QF $\hat{=}$ quadratic functional or quadrature formula
(\rightarrow Def. 1.4.3.41), 22

QMP $\hat{=}$ quadratic minimization problem , 22

QN $\hat{=}$ quadrature node (\rightarrow Def. 1.4.3.41), 100

QR $\hat{=}$ quadrature rule (\rightarrow Def. 1.4.3.41), 100

QW $\hat{=}$ quadrature weight (\rightarrow Def. 1.4.3.41), 100

RF $\hat{=}$ representation formula , 30

rhs $\hat{=}$ right-hand side , 38

RK-SSM $\hat{=}$ Runge-Kutta single-step method,
317

RK-SSM $\hat{=}$ Runge-Kutta single-step methods,
317

RKCQ $\hat{=}$ Runge-Kutta convolution quadrature,
317

s.p.d. $\hat{=}$ symmetric positive definite, 240

SLI $\hat{=}$ stationary linear iteration, 347

SSM $\hat{=}$ single-step method, 297

VF $\hat{=}$ variational formulation , 69

List of Symbols

- $C_{\text{pw}}^1(\overline{\Omega})$ $\hat{=}$ continuous, piecewise continuously differentiable functions, 19
 (x_ℓ) $\hat{=}$ sequence (usually on \mathbb{Z}), 259
 $*$ $\hat{=}$ convolution (binary operation), 255
 Div $\hat{=}$ divergence of a vector field, 24
 t_v^ℓ $\hat{=}$ interpolation nodes associated with cluster v , 203
 \mathcal{A} $\hat{=}$ set of transfer functions satisfying Ass. 3.3.1.1, 284
 $\mathcal{D}(\Omega)'$ $\hat{=}$ space of distributions on Ω , 35
 $\mathcal{L}_{\mathbb{I}}$ $\hat{=}$ set of leaves of a cluster tree $\mathcal{T}_{\mathbb{I}}$, 191
 $\mathcal{Q}_p(\mathbb{R}^d)$, 110
 FFT_n $\hat{=}$ discrete Fourier transform of length n , 315
 u, Fv, Fw $\hat{=}$ functions in $H^{\frac{1}{2}}(\Gamma)$, 49
 $H(\Delta, \Omega)$ $\hat{=}$ space of function with square-integrable Laplacian, 54
 $H^{\frac{1}{2}}(\Gamma)$ $\hat{=}$ Dirichlet trace space., 49
 $H^{-\frac{1}{2}}(\Gamma)$ $\hat{=}$ Neumann trace space on $\partial\Omega$, 53
 $H^1(\Omega)$ $\hat{=}$ Sobolev space, see Def. 1.1.2.14, 17
 $\mathcal{K}_l(\mathbf{A}, \mathbf{z})$ $\hat{=}$ Krylov subspace, 352
 $L^\infty(D)$ $\hat{=}$ space of bounded functions on D , 43
 $L^1(D)$ $\hat{=}$ space of integrable functions on D , 43
 $L^2(\Omega)$ $\hat{=}$ Hilbert space of square integrable functions, see Def. 1.1.2.15, 17
 $\|\cdot\|_{H^1(\Omega)}$ $\hat{=}$ norm of Sobolev space $H^1(\Omega)$, 17
 $\|\cdot\|_{L^2(\Omega)}$ $\hat{=}$ norm of $L^2(\Omega)$, 17
 $\|\cdot\|$ $\hat{=}$ Euclidean norm of a vector $\in \mathbb{R}^n$, 31
 $\|\cdot\|_1, \|\cdot\|_2, \|\cdot\|_\infty$ $\hat{=}$ vector norms and associated matrix norms, 158
 $\mathbf{A}, \mathbf{B}, \mathbf{C}, \dots$ (matrices), 77
 \mathbf{T}_v $\hat{=}$ transfer matrix in \mathcal{H}^2 -matrix format, 248
 adm $\hat{=}$ abstract admissibility condition, 198
 $\alpha_{\min}(\mathcal{G})$ $\hat{=}$ minimal angle of mesh \mathcal{G} , 132
 $\{\mathbf{T}\}_\Gamma$ $\hat{=}$ average of a trace, 58
 $B_r(\mathbf{x})$ $\hat{=}$ ball with center \mathbf{x} and radius $r > 0$, 27
 $\mathbb{C}^+ := \{z \in \mathbb{C} : \text{Re}(z) > 0\}$, 262
 $\mathbb{C}^+ := \{z \in \mathbb{C} : \text{Re}(z) > 0\}$, 262
 $\text{ID} := \{1, \dots, n\} \times \{1, \dots, m\}$ $\hat{=}$ index pairs for kernel collocation matrix, 181
 \mathbb{F} $\hat{=}$ matrix block partition for a hierarchical matrix, 215
 box $\hat{=}$ bounding box, 191
 box $\hat{=}$ bounding box for collocation points, 183
 grad_Γ $\hat{=}$ surface gradient, 66
 $\mathcal{CF}(X)$ $\hat{=}$ causal polynomially bounded functions, 262
 $\Omega' := \mathbb{R}^d \setminus \overline{\Omega}$ $\hat{=}$ complement of a domain $\Omega \subset \mathbb{R}^d$, 13
 $\text{CQ}_\tau^{\text{IE}}$ $\hat{=}$ implicit Euler convolution quadrature, 288
 D $\hat{=}$ total derivative operator, 15
 $\text{dist}(X; Y)$ $\hat{=}$ (Euclidean) distance of two set \mathbb{R}^d , 172
 $\text{dist}(X; Y)$ $\hat{=}$ distance of two sets $X, Y \subset \mathbb{R}^d$, 119
 $\dot{f}, \dot{\gamma}$ $\hat{=}$ derivative of a function depending on a single parameter (“time”), 31
 $\frac{df}{ds}$ $\hat{=}$ arclength derivative, 64
 $\frac{d}{ds}$ $\hat{=}$ arclength derivative, 64
 $G^\Delta(\mathbf{x}, \mathbf{y})$ $\hat{=}$ fundamental solutions, 34
 $\gamma^* f$ $\hat{=}$ pullback under parameterization γ , 79
 $[\mathbf{T}]_\Gamma$ $\hat{=}$ jump of a trace, 58
 $\kappa(\mathbf{A})$ $\hat{=}$ spectral condition number of the invertible matrix \mathbf{A} , 353
 L_ℓ $\hat{=}$ ℓ -th Lagrange polynomial for polynomial interpolation, 164
 $L(X, Y)$ $\hat{=}$ vector space of bounded (continuous) linear operators (mappings) $X \rightarrow Y$, 257
 $\mathcal{M}_H \prec \mathcal{M}_h$ $\hat{=}$ nesting of meshes, 366
 N_Δ $\hat{=}$ Newton potential operator, 38
 $\text{T}_{n, \Sigma}$ $\hat{=}$ normal component trace, 19
 T_N $\hat{=}$ Neumann trace on Γ , 52
 FFT $\hat{=}$ discrete Fourier transform, 271
 $\text{depth}(\mathcal{T})$ $\hat{=}$ depth of a tree \mathcal{T} , 190
 spm $\hat{=}$ sparsity measure of a cluster-based block partitioning, 208
 tril $\hat{=}$ lower-triangular part of a matrix, 349
 triu $\hat{=}$ upper triangular part of a matrix, 371
 \oplus $\hat{=}$ \mathcal{H} -addition of hierarchical matrices, 227
 $\mathcal{P}_p(\mathbb{R}^d)$ $\hat{=}$ space of d -variate polynomials, 122

$\mathcal{P}_p(\mathbb{R}^d) \hat{=}$ d -variate polynomials of total degree $\leq p$, 79

$H_{\text{pw}}^m(\partial\Omega) \hat{=}$ piecewise Sobolev space on $\Gamma := \partial\Omega$, 132

$\mathcal{S}_p^{-1}(\mathcal{G}) \hat{=}$ discontinuous, piecewise polynomial BE functions of degree p , 123

$\mathcal{S}_p^{-1}(\mathcal{G}) \hat{=}$ discontinuous, piecewise polynomial BE functions of degree p , 79

$\text{rank}(\mathbf{M}) \hat{=}$ rank of a matrix \mathbf{M} , 158

$\hat{I} \hat{=}$ reference interval $] -1, 1[$, 82

$\rho_K \hat{=}$ shape regularity measure of cell K , 343

$\rho_{\mathcal{M}} \hat{=}$ shape regularity measure of a mesh \mathcal{M} , 343

$\mathbf{M}|_{v \times w} \hat{=}$ matrix block belonging to a pair of clusters, 202

$\mathcal{S}_p^0(\mathcal{G}) \hat{=}$ continuous, piecewise polynomial BE functions of degree p , 123

$\mathcal{S}_p^0(\mathcal{G}) \hat{=}$ continuous, piecewise polynomial BE functions of degree p , 79

$\#\mathcal{M} \hat{=}$ cardinality (no. of elements) of the set \mathcal{M} , 80

$\mathbb{T}_{t,\Sigma} \hat{=}$ tangential component trace, 19

$\text{Tr} \hat{=}$ trace operator for matrices, 37

$\text{root}(\mathcal{T}) \hat{=}$ root of a tree \mathcal{T} , 189

$\vec{\mu}, \vec{\varphi}, \vec{\xi}, \dots$ (coefficient vectors), 77

$\mathcal{V}(\mathcal{G}) \hat{=}$ set of vertices of mesh \mathcal{G} , 81

$b_N^1, \dots, b_N^N \hat{=}$ basis function for BE space, 80

$b_\ell^v \hat{=}$ ℓ -th cardinal function belonging to cluster v , 203

$cqop \hat{=}$ convolution quadrature operator, 284

$diam \hat{=}$ diameter of a set in \mathbb{R}^d , 172

$h_{\mathcal{G}} \hat{=}$ meshwidth of mesh \mathcal{G} , 132

$sH^{\frac{1}{2}}(\partial\Omega) \hat{=}$ functions in $H^{\frac{1}{2}}(\partial\Omega)$ with vanishing mean, 71

$\mathcal{G}_{\Gamma} \hat{=}$ mesh of curve/surface Γ , 79

$\mathcal{E}(\mathcal{G}) \hat{=}$ edge set of a mesh, 121

$\mathcal{V}(\mathcal{G}) \hat{=}$ vertex set of a mesh, 121

$\mathbf{a}, \dots, \mathbf{x}, \mathbf{y}, Bz \hat{=}$ small vectors/points, 14

Examples and Remarks

\mathcal{H} -LU decomposition as preconditioner, 245

\mathcal{H} -matrix \times dense matrix, 220

δ for simple BDF multi-step schemes, 309

Backward difference formulas (BDF), [DB02, Sect. 7.3.2], 298

Global bi-directional interpolation of singular kernel, 168

" $L_y G^L = \delta_x$ ", 35

"Continuity" of functions in $H^{\frac{1}{2}}(\Gamma)$, 51

"Differentiation theorem" for convolution quadrature, 293

"First-kind", 71

"Second-kind", 72

A basis for $\mathcal{S}_0^{-1}(\mathcal{G})$, 81

A concrete basis transformation matrix, 367

A-stability and order of linear multi-step methods, 303

Abstract convergence theory for SSC, 363

Adaptive Clenshaw-Curtis quadrature, 109

Adaptive low-rank recompression, 225

Admissible source charge distributions, 22

Affine space V , 76

Algorithms for Runge-Kutta-based convolution quadrature, 322

AMG terminology, 382

Anisotropic diffusion matrix, 388

Approximately solving convolution equations by convolution quadrature, 285

Approximation of surfaces, 124

Approximations underlying (3.4.2.34), 310

Assembly of Galerkin matrix for double layer BIO K , 91

Asymptotic complexity of \mathcal{H} -multiplication, 239

Asymptotic decay of iteration error, 350

Asymptotically smooth kernels, 169

Behavior of quadrature errors for global quadrature rules, 104

Bi-directional interpolation of smooth kernel function, 168

Bi-directional polynomial interpolation, 206

BIEs for general second-order scalar differential operators, 68

Bilinear FE for scalar elliptic Dirichlet BVPs, 344

Binary cluster tree for $d = 1$, 195

Boundary integral equations related to scalar 2nd-order elliptic BVPs, 152

Bounding the sparsity measure, 209

Choice of integration radius r , 316

Co-normal trace, 56

Coarse-fine splitting for the Poisson matrix, 389

Complex contour integrals, 263

Compressing discrete BIEs with double layer kernels, 213

Computing G^Δ in 3D, 36

Convergence of CG for the Poisson matrix, 354

Convergence of CQ based on BDF-2, 311

Convergence of Gauss-Seidel II, 355

Convergence of Gauss-Seidel method for Poisson matrix, 351

Convergence of implicit Euler convolution quadrature, 294

Convergence of multi-step CQ, 311

Convolution evolution partial differential equations, 322

Convolution in $L^p(\mathbb{R})$ -spaces, 256

Convolution of distributions [Rud73, pp. 170], 256

Convolution quadrature based on explicit Euler timestepping?, 288

Cost of direct elimination solvers, 347

Data structure for \mathcal{H}^2 -matrices, 249

Density argument, 49

Derivation of impedance conditions, 280

Differential operators are strictly local, 148

Direct computation of convolution quadrature weights, 291

Electrostatic interpretation of Ψ_{SL} , 44

Electrostatic meaning of Ψ_{DL}^Δ , 45

Electrostatics in homogeneous isotropic media, 26

- Error incurred in multiplication of \mathcal{H} -matrices, 239
- Expand and restrict as adjoint operations, 210
- Explicit midpoint method, 298
- Explicit midpoint method for decay equation, Ex. 3.4.1.26 cont'd, 301
- Exponential sum approximation by quadrature, 324
- Families of sparse matrices, 158
- Fast & oblivious CQ: Properties & variants, 338
- Finite element discretization, 280
- Fixed potential boundary conditions, 21
- Fixing the potential, 21
- Fundamental solution for 2nd-order partial differential operator, 37
- Galerkin error estimates for 2nd-kind BIE, 131
- Gauss' law, 25
- General cluster tree, 191
- General layer potentials, 57
- Global quadrature of analytic integrand, 108
- Globally supported singular kernel functions, 156
- Green's function for a half space, 47
- Green's function for $-\Delta$ on a disk, 46
- Impact of kernel approximation on kernel matrix, 160
- Important stiffly accurate RK-SSMs, 318
- Instability of a 2-step method, 301
- Integral representation formula for exterior domains, 41
- Intrinsic norm of $H^{\frac{1}{2}}(\Gamma)$, 52
- Iterative solution methods for linear systems of equations, 213
- Kernel with known Laplace transform, 280
- Laplace transform of causal power function, 262
- Layer potentials and traces, 42
- local→global index map, 89
- Logarithmic kernel in 1D: Separable approximation by Taylor expansion, 162
- Matrix functions, 320
- Meaning of “density unknowns” ϕ and \mathbf{v} , 74
- Measuring rates of convergence of stationary linear iterations, 350
- More general surface meshes, 122
- Multigrid cycles, 371
- Multigrid iteration as successive subspace correction method, 371
- Multiplication of “simple” \mathcal{H} -matrices, 237
- Near- and far-field boxes constructed from cluster trees in 1D, 201
- Necessity of decay conditions, 28
- Nodal basis for $\mathcal{S}_1^0(\mathcal{G})$, 81
- Nodal interpolation operators, 123
- Nyström-discretized boundary integral equations of the second kind, 149
- Order of consistency of some multi-step methods, 300
- Pairing of traces, 53
- Partial blocks contributing to target block, 237
- Poisson equation, 341
- Poisson integral formula, 47
- Poisson matrix, 344
- Polynomially growing transfer functions, 294
- Potentials on unbounded domains, 18
- Precomputing complex quadrature formula, 129
- Preview: multiplication of hierarchical matrices, 220
- Properties of the potential due to a point charge, 32
- Properties of the potential of a point charge in 2D, 33
- Quadrature error of t -local trapezoidal rule quadrature, 328
- Quadrature over Talbot contour, 326
- Quadtree partition from cluster trees, 196
- Quadtree-based admissible tiling of unit square [Bör21, Sect. 2.4], 187
- Real-valued convolution quadrature weights, 290
- Reference shape functions for $\mathcal{S}_0^{-1}(\mathcal{G})$, 125
- Reference shape functions for $\mathcal{S}_1^0(\mathcal{G})$, 125
- Region of stability for BDF methods, 303
- Relationship to prolongation for geometric multigrid, 382
- S.p.d. boundary element Galerkin matrices, 240
- Scalar potentials and work, 16
- Scaling of electromagnetic field problems, cf. [NumPDE Rem. 1.2.1.25], 14
- Separable approximation by truncated power series, 161
- Sequences as distributions, 260
- Signal-processing background, 258
- Simplification of right-hand side, 142
- Some special convolutions, 256
- Sparsity measure for clustering in 1D, 209
- Special initial steps for multi-point methods, 299

- SSC with $M = 2$: Method of alternating projections, 362
- Stable evaluation of integrands, 118
- Stiffly accurate RK-SSMs, 318
- Storage requirements of double-factor and triple-factor representations, 246
- Strong couplings defined by Poisson matrix, 388
- Surface meshes as traces of volume meshes, 122
- Tensor-product Chebychev interpolation of singular kernel, 177
- The “magic” of the equidistant composite trapezoidal quadrature rule, [NumCSE Exp. 7.5.0.16], 313
- The AMG fill-in challenge, 384
- The geometry of SSC, 361
- The Neumann trace is not defined on $H^1(\Omega)$, 53
- The Newton potential from a physics perspective, 39
- The prototypical simple \mathcal{H} -matrix, 215
- The square of the Abel integral operator, 277
- Trapezoidal rule, 101
- Two-grid method for the Poisson matrix, 369
- Unbalanced cluster tree, 195
- Unbounded functions in $H^{\frac{1}{2}}(\Gamma)$, 52
- Well-defined IE-CQ, 288