

ETH Lecture 401-0663-00L Numerical Methods for **Computer Science****End-Term Examination**

Autumn Term 2021

Thursday, Dec 23, 2021, 10:15, HG F 1 & TBA

**Don't
panic!**

Family Name		Grade
First Name		
Department		
Legi Nr.		
Date	Thursday, Dec 23, 2021	

Points:

Prb. No.	1	2	3	Total
max	15	18	20	53
achvd				

(100% = 50 pts. , $\approx 40\%$ (passed) = 20 pts.)

- Upon entering the exam room take a seat at a desk on which you find an exam paper with this cover page! Do not take a seat where you spot a red sticker "CSE".
- This is a **closed-book exam**, no aids are allowed.
- Keep only writing paraphernalia and your ETH ID card on the table.
- Turn off mobile phones, tablets, smartwatches, etc. and stow them away in your bag.
- When told to do so, take the exam paper out of the envelope, and fill in the cover sheet first. Do not turn pages yet!
- Make sure that your exam paper is for the course 401-0663-00L Numerical Methods for **Computer Science**, see the top of the front page.
- Turn the cover sheet only when instructed to do so.
- You will be given **10 minutes of advance reading** time to familiarize yourself with the topic areas of the problems. When told, drop any pen! Then you may turn the pages and start reading the problems.
- Start writing only when the start of the exam time proper is announced.
- **Write your answers in the appropriate (green) solution boxes on these problem sheets.**
- **Wrong ticks in multiple-choice boxes can lead to points being subtracted.** Hence, mere guessing is really dangerous! If you have no clue, leave all tickboxes empty.
- If you change your mind about an answer to a (multiple-choice) question, write a clear NO next to the old answer, draw fresh solution boxes/tickboxes and fill them.
- **Anything written outside the answer boxes will not be taken into account.**

- Do not write with red/green color or with pencil.
- **Duration: 30 minutes.**
- When the end of the exam is announced, make sure you have written your name on every sheet.
- The exam proctors will collect the filled exam papers. Remain seated until they have finished.

Special Covid-19 Safety Measures

- Only students in possession of a valid Covid Certificate are allowed to take the term exam.
- Protective masks covering nose and mouth have to be worn all the time.

Throughout the exam use the notations introduced in class, in particular [Lecture → Section 1.1.1]:

- $(\mathbf{A})_{i,j}$ to refer to the entry of the matrix $\mathbf{A} \in \mathbb{K}^{m,n}$ at position (i, j) .
- $(\mathbf{A})_{:,i}$ to designate the i -th-column of the matrix \mathbf{A} ,
- $(\mathbf{A})_{i,:}$ to denote the i -th row of the matrix \mathbf{A} ,
- $(\mathbf{A})_{i:j,k:\ell}$ to single out the sub-matrix $\left[(\mathbf{A})_{r,s} \right]_{\substack{i \leq r \leq j \\ k \leq s \leq \ell}}$ of the matrix \mathbf{A} ,
- $(\mathbf{x})_k$ to reference the k -th entry of the vector \mathbf{x} ,
- $\mathbf{e}_j \in \mathbb{R}^n$ to write the j -th Cartesian coordinate vector,
- \mathbf{I} to denote the identity matrix,
- \mathbf{O} to write a zero matrix,
- \mathcal{P}_n for the space of (univariate polynomials of degree $\leq n$),
- and superscript indices in brackets to denote iterates: $\mathbf{x}^{(k)}$, etc.

By default, vectors are regarded as column vectors.

Problem 0-1: Various Aspects of Interpolation by Global Polynomials

[Lecture → Chapter 5] teaches the mathematical foundations and algorithmic realizations of interpolation by means of global polynomials. This problem reviews some of these aspects.

This is a purely theoretical problem connected with [Lecture → Section 5.2]

Throughout this problem we write $\mathcal{P}_d(\mathbb{R})$ for the space of uni-variate polynomials of degree $\leq d$:

$$\mathcal{P}_k := \{t \mapsto \alpha_k t^k + \alpha_{k-1} t^{k-1} + \cdots + \alpha_1 t + \alpha_0 \cdot 1, \alpha_j \in \mathbb{R}\}. \quad [\text{Lecture} \rightarrow \text{Eq. (5.2.1.1)}]$$

(0-1.a) (6 pts.) What are the dimensions of the following subspaces of $\mathcal{P}_d(\mathbb{R})$, $d \in \mathbb{N}$?

(i) $V_1 := \{p \in \mathcal{P}_d(\mathbb{R}) : \int_{-1}^1 p(t) dt = 0\}$: $\dim V_1 =$.

(ii) $V_2 := \{p \in \mathcal{P}_d(\mathbb{R}) : p(1) = p(-1) = 0\}$: $\dim V_2 =$.

(iii) $V_3 := \{p \in \mathcal{P}_d(\mathbb{R}) : p^{(3)}(t) = 0 \forall t \in [-1, 1]\}$: $\dim V_3 =$ $\begin{cases} \text{ } & \text{for } \text{ } \\ \text{ } & \text{for } \text{ } \end{cases}$.

Here $p^{(3)}$ stands for the third derivative of p .

(iv) $V_4 := \{p \in \mathcal{P}_d(\mathbb{R}) : p(t) = p(-t)\}$: $\dim V_4 =$ $\begin{cases} \text{ } & \text{for } \text{ } \\ \text{ } & \text{for } \text{ } \end{cases}$.

SOLUTION of (0-1.a):

In this problem we can sometimes appeal to the heuristics that

$$\dim V_i = \dim \mathcal{P}_d(\mathbb{R}) - \#\{\text{linear constraints defining } V_i\}.$$

From [Lecture → Thm. 5.2.1.2] we know that $\dim \mathcal{P}_d = d + 1$.

1. V_1 is defined by a single linear constraint, which means that

$$\dim V_1 = d + 1 - 1 = d.$$

2. V_2 is defined by fixing two zeros, which amounts to two linearly independent linear constraints.

$$\blacktriangleright \dim V_2 = \dim \mathcal{P}_d(\mathbb{R}) - 2 = d - 1.$$

Remark. In fact we can write

$$V_2 := \{t \mapsto (1 - t)^2 q(t), q \in \mathcal{P}_{d-2}(\mathbb{R})\}.$$

3. If $p \in \mathcal{P}_d(\mathbb{R})$, then for the third derivative $p^{(3)} \in \mathcal{P}_{d-3}(\mathbb{R})$. Moreover, if a polynomial vanishes on an interval ("has infinitely many zeros"), then it must be zero everywhere. We conclude that $p^{(3)} \equiv 0$. As a consequence $p \in \mathcal{P}_2$.

$$\blacktriangleright \dim V_3 = \begin{cases} 2 & \text{for } d = 1, \\ 3 & \text{for } d \geq 2. \end{cases}$$


4. The polynomials in V_4 are *even functions*, which means that they are of the form

$$V_4 \subset \mathcal{P}_n^{\text{even}} := \{t \mapsto a_0 + a_2 t^2 + a_4 t^4 + \dots + a_{2n} t^{2n}\}, \quad n \in \mathbb{N}.$$

Elements of space $\mathcal{P}_n^{\text{even}}$ are defined by $n+1$ parameters, which means $\dim \mathcal{P}_n^{\text{even}} = n+1$.

Note that $n = \frac{d}{2}$ for even d and $n = \frac{d-1}{2}$ for odd d .

$$\blacktriangleright \quad \dim V_4 = \begin{cases} \frac{d}{2} + 1 & \text{for even } d, \\ \frac{d-1}{2} + 1 & \text{for odd } d. \end{cases}$$

(0-1.b)  (3 pts.)

Let $\{t_0, t_1, \dots, t_n\} \subset \mathbb{R}$, $n \in \mathbb{N}$. The following sets of functions provide bases for \mathcal{P}_d :

$$\mathfrak{B}_1 := \left\{ t \mapsto \prod_{\substack{j=0 \\ j \neq i}}^n \frac{t - t_j}{t_i - t_j}, \quad i = 0, \dots, n \right\}, \quad \square$$

$$\mathfrak{B}_2 := \left\{ t \mapsto \prod_{j=0}^{i-1} (t - t_j), \quad i = 0, \dots, n \right\}, \quad \square$$

$$\mathfrak{B}_3 := \left\{ t \mapsto t^i, \quad i = 0, \dots, n \right\}. \quad \square$$

Note that an empty product is defined to be $\equiv 1$.

Put the right letter in the box indicating the name of the basis:

A \triangleq monomial basis, **B** \triangleq Newton basis, **C** \triangleq Lagrangian basis.

SOLUTION of (0-1.b):

- \mathfrak{B}_1 is the Lagrangian basis **C** [Lecture \rightarrow Eq. (5.2.2.4)], a cardinal basis for polynomial interpolation in $\{t_0, t_1, \dots, t_n\}$,
- \mathfrak{B}_2 is the Newton basis **B** [Lecture \rightarrow Eq. (5.2.3.23)], used for update-friendly polynomial interpolation
- \mathfrak{B}_3 is the monomial basis **A** [Lecture \rightarrow Section 5.2.1].

(0-1.c)  (6 pts.)

Given data points (t_i, y_i) , $i = 0, \dots, n$, $n \in \mathbb{N}$, we write $p_{k,\ell}$, $0 \leq k \leq \ell \leq n$, for the polynomial $p_{k,\ell} \in \mathcal{P}_{\ell-k}$ interpolating through $(t_i, y_i)_{i=k}^{\ell}$: $p_{k,\ell}(t_i) = y_i$, $i = k, \dots, \ell$. Supplement the missing parts of the following recursion:

$$p_{k,\ell}(t) = \frac{\left(\square \right) p_{k+1,\ell}(t) + \left(\square \right) p_{k,\ell-1}(t)}{\square - \square}, \quad t \in \mathbb{R}, \quad 0 \leq k < \ell \leq n.$$

SOLUTION of (0-1.c):

This is the recursion [Lecture → Eq. (5.2.3.9)] underlying the **Aitken-Neville** scheme:

$$p_{k,\ell}(t) = \frac{\left(\boxed{t - t_k} \right) p_{k+1,\ell}(t) + \left(\boxed{t_\ell - t} \right) p_{k,\ell-1}(t)}{\boxed{t_\ell} - \boxed{t_k}}, \quad t \in \mathbb{R}.$$

The correctness of this recursion is shown by induction, verifying the interpolation conditions.



End Problem 0-1 , 15 pts.

Problem 0-2: Local Representations of Splines

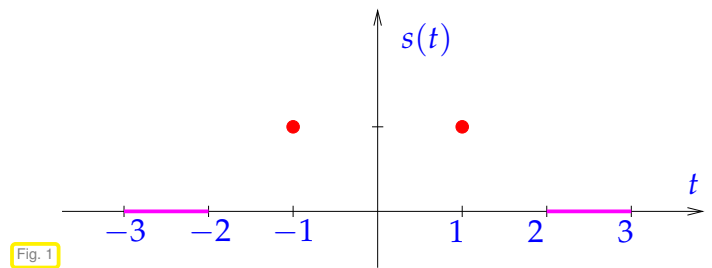
Locally on knot intervals splines coincide with polynomials of a fixed degree. In this problem we compute those local polynomial representations in special cases.

This is a purely theoretical problem based on [Lecture → Section 5.4.1] and [Lecture → Section 5.4.2].

(0-2.a) (6 pts.)

Let s denote a *quadratic spline*, $s \in \mathcal{S}_{2,\mathcal{M}}$ with respect to the knot set $\mathcal{M} := \{-3, -2, -1, 1, 2, 3\}$ and satisfying

$$\begin{aligned} s &\equiv 0 \quad \text{on} \quad [-3, -2] \cup [2, 3], \\ s(-1) &= s(1) = 1. \end{aligned}$$



Determine the unknown real coefficients in the local representations

$$\begin{aligned} s(t) &= \boxed{} t^2 + \boxed{} t + \boxed{} \quad \text{for } t \in [1, 2], \\ s(t) &= \boxed{} t^2 + \boxed{} t + \boxed{} \quad \text{for } t \in [-1, 1]. \end{aligned}$$

SOLUTION of (0-2.a):

Since $s \in C^1([-3, 3])$ and $s \equiv 0$ on $[2, 3]$, we conclude $s(2) = s'(2) = 0$, which means

$$s(t) = \alpha(t-2)^2, \quad \alpha \in \mathbb{R} \quad \text{on} \quad [1, 2].$$

From the condition $s(1) = 1$ we get $\alpha = 1$, which implies

$$s(t) = 1 \cdot t^2 - 4t + 4 \quad \text{on} \quad [1, 2].$$

As consequence $s'(1) = -2$. Thus the quadratic polynomial p describing s on $[-1, 1]$ has to satisfy

$$p(-1) = 1, \quad p(1) = 1, \quad p'(1) = -2.$$

This uniquely defines the parabola

$$p(t) = 2 - t^2, \quad t \in \mathbb{R},$$

from which we infer

$$s(t) = -1 \cdot t^2 + 0t + 2 \quad \text{on} \quad [-1, 1].$$

Remark. Here the solution of the problem can stop, but we have not yet verified that the partial s as found so far can be extended to a spline in $\mathcal{S}_{2,\mathcal{M}}$. We conjecture that s is an even function $s(t) = s(-t)$ and, thus, set

$$s(t) = t^2 + 4t + 4 \quad \text{on} \quad [-2, -1].$$

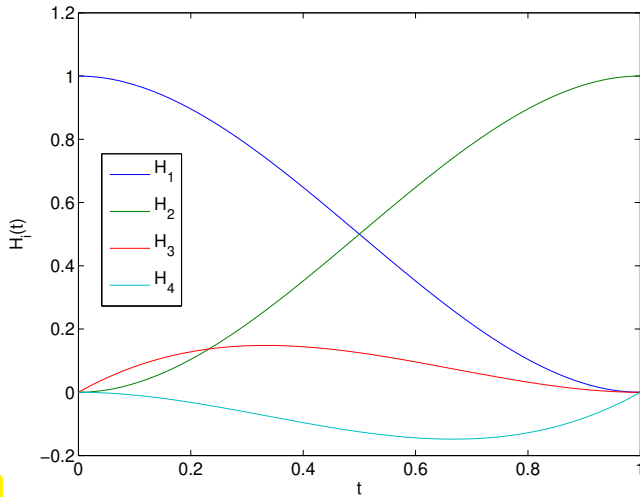
Symmetry arguments readily demonstrate that this completes the definition of the spline s .



(0-2.b) (12 pts.) Let the knot set $\mathcal{M} := \{t_0 < t_1 < t_2 < \dots < t_n\} \subset \mathbb{R}$ be given. The local representation of a cubic spline $s \in \mathcal{S}_{3,\mathcal{M}}$ is given by $(h_j := t_j - t_{j-1}, j \in \{1, \dots, n\})$

$$s|_{[t_{j-1}, t_j]}(t) = \alpha_j H_1(\tau) + \beta_j H_2(\tau) + h_j \gamma_j H_3(\tau) + h_j \delta_j H_4(\tau), \quad \tau := \frac{t - t_{j-1}}{h_j}, \quad \alpha_j, \beta_j, \gamma_j, \delta_j \in \mathbb{R},$$

where H_1, H_2, H_3, H_4 are the cardinal basis functions for Hermite interpolation on $[0, 1]$,



$$\begin{aligned} H_1(\tau) &= 1 - 3\tau^2 + 2\tau^3, \\ H_2(\tau) &= 3\tau^2 - 2\tau^3, \\ H_3(\tau) &= \tau - 2\tau^2 + \tau^3, \\ H_4(\tau) &= -\tau^2 + \tau^3, \end{aligned}$$

Fig. 2

which satisfy

$$\begin{aligned} H_1(0) &= 1, \quad H_1(1) = 0, \quad H_1'(0) = 0, \quad H_1'(1) = 0, \quad H_1(\tfrac{1}{2}) = \tfrac{1}{2}, \quad H_1'(\tfrac{1}{2}) = -\tfrac{3}{2}, \\ H_2(0) &= 0, \quad H_2(1) = 1, \quad H_2'(0) = 0, \quad H_2'(1) = 0, \quad H_2(\tfrac{1}{2}) = \tfrac{1}{2}, \quad H_2'(\tfrac{1}{2}) = \tfrac{3}{2}, \\ H_3(0) &= 0, \quad H_3(1) = 0, \quad H_3'(0) = 1, \quad H_3'(1) = 0, \quad H_3(\tfrac{1}{2}) = \tfrac{1}{8}, \quad H_3'(\tfrac{1}{2}) = -\tfrac{1}{4}, \\ H_4(0) &= 0, \quad H_4(1) = 0, \quad H_4'(0) = 0, \quad H_4'(1) = 1, \quad H_4(\tfrac{1}{2}) = -\tfrac{1}{8}, \quad H_4'(\tfrac{1}{2}) = -\tfrac{1}{4}. \end{aligned} \quad (0.2.1)$$

Inserting the midpoints of knot intervals gives us the extended knot set

$$\widetilde{\mathcal{M}} := \{\tilde{t}_0 < \tilde{t}_1 < \tilde{t}_2 < \dots < \tilde{t}_{2n}\}, \quad \begin{aligned} \tilde{t}_{2j} &:= t_j, & j &\in \{0, \dots, n\}, \\ \tilde{t}_{2j-1} &:= \tfrac{1}{2}(t_j + t_{j-1}), & j &\in \{1, \dots, n\}. \end{aligned}$$

On the knot intervals of $\widetilde{\mathcal{M}}$ the spline s has the local representation $(\tilde{h}_\ell := \tilde{t}_\ell - \tilde{t}_{\ell-1}, \ell \in \{1, \dots, 2n\})$

$$s|_{[\tilde{t}_{\ell-1}, \tilde{t}_\ell]}(t) = \tilde{\alpha}_\ell H_1(\tau) + \tilde{\beta}_\ell H_2(\tau) + \tilde{h}_\ell \tilde{\gamma}_\ell H_3(\tau) + \tilde{h}_\ell \tilde{\delta}_\ell H_4(\tau), \quad \tau := \frac{t - \tilde{t}_{\ell-1}}{\tilde{h}_\ell}, \quad \tilde{\alpha}_\ell, \tilde{\beta}_\ell, \tilde{\gamma}_\ell, \tilde{\delta}_\ell \in \mathbb{R}.$$

Express the coefficients $\tilde{\alpha}_\ell, \tilde{\beta}_\ell, \tilde{\gamma}_\ell, \tilde{\delta}_\ell$ in terms of $\alpha_j, \beta_j, \gamma_j, \delta_j$:

$$\begin{aligned}
 \tilde{\alpha}_\ell &= \begin{cases} \boxed{} & \text{for even } \ell = 2j, \\ \boxed{} & \text{for odd } \ell = 2j - 1, \end{cases} \\
 \tilde{\beta}_\ell &= \begin{cases} \boxed{} & \text{for even } \ell = 2j, \\ \boxed{} & \text{for odd } \ell = 2j - 1, \end{cases} \\
 \tilde{\gamma}_\ell &= \begin{cases} \boxed{} & \text{for even } \ell = 2j, \\ \boxed{} & \text{for odd } \ell = 2j - 1, \end{cases} \\
 \tilde{\delta}_\ell &= \begin{cases} \boxed{} & \text{for even } \ell = 2j, \\ \boxed{} & \text{for odd } \ell = 2j - 1. \end{cases}
 \end{aligned}$$

SOLUTION of (0-2.b):

The key insight is that from (0.2.1) and the chain rule we conclude for the spline s

$$s(t_{j-1}) = \alpha_j, \quad s(t_j) = \beta_j, \quad s'(t_{j-1}) = \gamma_j, \quad s'(t_j) = \delta_j, \quad j = 1, \dots, n. \quad (0.2.2)$$

This means that the coefficients have a concrete meaning as the point values and derivative values of the spline in the endpoints of the knot intervals.

The same argument yields, now using the local representation on the knot intervals of $\tilde{\mathcal{M}}$:

$$s(\tilde{t}_{\ell-1}) = \tilde{\alpha}_\ell, \quad s(\tilde{t}_\ell) = \tilde{\beta}_\ell, \quad s'(\tilde{t}_{\ell-1}) = \tilde{\gamma}_\ell, \quad s'(\tilde{t}_\ell) = \tilde{\delta}_\ell, \quad \ell = 1, \dots, 2n. \quad (0.2.3)$$

Next, we have to compute the values of s and s' at midpoints of knot intervals of \mathcal{M} . We start with

$$\begin{aligned}
 H_1(1/2) &= \frac{1}{2}, & H'_1(1/2) &= -\frac{3}{2}, \\
 H_2(1/2) &= \frac{1}{2}, & H'_2(1/2) &= \frac{3}{2}, \\
 H_3(1/2) &= \frac{1}{8}, & H'_3(1/2) &= -\frac{1}{4}, \\
 H_4(1/2) &= -\frac{1}{8}, & H'_4(1/2) &= -\frac{1}{4},
 \end{aligned} \quad (0.2.4)$$

which implies

$$s\left(\frac{1}{2}(t_j + t_{j-1})\right) = \frac{1}{2}(\alpha_j + \beta_j) + \frac{1}{8}(\gamma_j - \delta_j), \quad (0.2.5)$$

$$s'\left(\frac{1}{2}(t_j + t_{j-1})\right) = \frac{3}{2}(-\alpha_j + \beta_j) - \frac{1}{4}(\gamma_j + \delta_j). \quad (0.2.6)$$

I. For **even** $\ell = 2j$ we have $\tilde{t}_\ell = t_j, j = 1, \dots, n$ and the associated knot interval for $\ell, j > 0$ is

$$[\tilde{t}_{\ell-1}, \tilde{t}_\ell] = \left[\frac{1}{2}(t_j + t_{j-1}), t_j\right].$$

So its right endpoint is a knot of \mathcal{M} , the left endpoint a midpoint of a knot interval of \mathcal{M} . We conclude from (0.2.2), (0.2.3), (0.2.5)

$$\begin{aligned}\tilde{\alpha}_\ell &= s(\tilde{t}_{\ell-1}) = s(\tfrac{1}{2}(t_j + t_{j-1})) = \tfrac{1}{2}(\alpha_j + \beta_j) + \tfrac{1}{8}h_j(\gamma_j - \delta_j) , \\ \tilde{\beta}_\ell &= s(\tilde{t}_\ell) = s(t_j) = \beta_j , \\ \tilde{\gamma}_\ell &= s'(\tilde{t}_{\ell-1}) = s'(\tfrac{1}{2}(t_j + t_{j-1})) = \tfrac{3}{2}h_j^{-1}(-\alpha_j + \beta_j) - \tfrac{1}{4}(\gamma_j + \delta_j) , \\ \tilde{\delta}_\ell &= s'(\tilde{t}_\ell) = s'(t_j) = \delta_j .\end{aligned}$$

The factor h_j^{-1} is introduced by applying the chain rule.

II. For **odd** $\ell = 2j - 1$, $j = 1, \dots, n$, we consider the $\tilde{\mathcal{M}}$ knot interval

$$[\tilde{t}_{\ell-1}, \tilde{t}_\ell] = [t_{j-1}, \tfrac{1}{2}(t_j + t_{j-1})] .$$

Now the left endpoint is a midpoint of a knot interval of \mathcal{M} , and the right endpoint is a knot of \mathcal{M} . As before we conclude

$$\begin{aligned}\tilde{\alpha}_\ell &= s(\tilde{t}_{\ell-1}) = s(t_{j-1}) = \alpha_j , \\ \tilde{\beta}_\ell &= s(\tilde{t}_\ell) = s(\tfrac{1}{2}(t_j + t_{j-1})) = \tfrac{1}{2}(\alpha_j + \beta_j) + \tfrac{1}{8}h_j(\gamma_j - \delta_j) , \\ \tilde{\gamma}_\ell &= s'(\tilde{t}_{\ell-1}) = s'(t_{j-1}) = \gamma_j , \\ \tilde{\delta}_\ell &= s'(\tilde{t}_\ell) = s'(\tfrac{1}{2}(t_j + t_{j-1})) = \tfrac{3}{2}h_j^{-1}(-\alpha_j + \beta_j) - \tfrac{1}{4}(\gamma_j + \delta_j) .\end{aligned}$$

Summing up

$$\begin{aligned}\tilde{\alpha}_\ell &= \begin{cases} \tfrac{1}{2}(\alpha_j + \beta_j) + \tfrac{1}{8}h_j(\gamma_j - \delta_j) & \text{for even } \ell , \\ \alpha_j & \text{for odd } \ell , \end{cases} & \tilde{\beta}_\ell &= \begin{cases} \beta_j & \text{for even } \ell , \\ \tfrac{1}{2}(\alpha_j + \beta_j) + \tfrac{1}{8}h_j(\gamma_j - \delta_j) & \text{for odd } \ell , \end{cases} \\ \tilde{\gamma}_\ell &= \begin{cases} \tfrac{3}{2}h_j^{-1}(-\alpha_j + \beta_j) - \tfrac{1}{4}(\gamma_j + \delta_j) & \text{for even } \ell , \\ \gamma_j & \text{for odd } \ell , \end{cases} & \tilde{\delta}_\ell &= \begin{cases} \delta_j & \text{for even } \ell , \\ \tfrac{3}{2}h_j^{-1}(-\alpha_j + \beta_j) - \tfrac{1}{4}(\gamma_j + \delta_j) & \text{for odd } \ell . \end{cases}\end{aligned}$$



End Problem 0-2 , 18 pts.

Problem 0-3: Modified Cosine Transform

Many so-called trigonometric transformations can be reduced to the discrete Fourier transform (DFT). One such example is studied in this problem.

Assumes familiarity with the discrete Fourier transform, C++, and the complex exponential

For a given vector $\mathbf{a} = [a_0, \dots, a_{n-1}]^\top \in \mathbb{R}^n$, $n \in \mathbb{N}$, we want to compute another vector $\mathbf{f} := [f_0, \dots, f_{n-1}]^\top \in \mathbb{R}^n$ according to the formula

$$f_k := \sum_{j=0}^{n-1} a_j \cos\left(\pi \frac{jk}{n}\right), \quad k = 0, \dots, n-1. \quad (0.3.1)$$

Using the identity $\cos x = \frac{1}{2}(\exp(-ix) + \exp(ix))$, we find the alternative formula

$$f_k = \sum_{\ell=0}^{n-1} \frac{1}{2} a_\ell \exp(-2\pi i \frac{k\ell}{2n}) + \exp(\pi i \frac{k(n-1)}{n}) \cdot \sum_{\ell=0}^{n-1} \frac{1}{2} a_{n-1-\ell} \exp(-2\pi i \frac{k\ell}{2n}) \quad (0.3.2)$$

for $k = 0, \dots, n-1$.

Recall the definition of the discrete Fourier transform (DFT) as multiplication of a vector with the Fourier matrix \mathbf{F}_n :

Definition [Lecture \rightarrow Def. 4.2.1.18].

The linear map $\text{DFT}_n : \mathbb{C}^n \mapsto \mathbb{C}^n$, $\text{DFT}_n(\mathbf{y}) := \mathbf{F}_n \mathbf{y}$, $\mathbf{y} \in \mathbb{C}^n$, is called **discrete Fourier transform** (DFT), i.e. for $[c_0, \dots, c_{n-1}] := \text{DFT}_n(\mathbf{y})$

$$c_k := \sum_{j=0}^{n-1} y_j \exp(-2\pi i \frac{kj}{n}), \quad k = 0, \dots, n-1. \quad [\text{Lecture} \rightarrow \text{Eq. (4.2.1.19)}]$$

(0-3.a) (10 pts.) Taking the cue from (0.3.2) we can write ("C++ indexing" throughout)

$$\mathbf{f} = (\text{DFT}_{2n}(\mathbf{x}))_{0:n-1} + \mathbf{D}(\text{DFT}_{2n}(\mathbf{y}))_{0:n-1}, \quad (0.3.3)$$

with suitable vectors $\mathbf{x}, \mathbf{y} \in \mathbb{C}^{2n}$ and a **diagonal** matrix $\mathbf{D} \in \mathbb{C}^{n,n}$. Characterize the vectors \mathbf{x}, \mathbf{y} (in terms of \mathbf{a}) and the matrix \mathbf{D} .

$$(\mathbf{x})_\ell = \begin{cases} \boxed{} & \text{for } \ell \in \{ \boxed{} \}, \\ \boxed{} & \text{for } \ell \in \{ \boxed{} \}, \end{cases}, \quad (\mathbf{y})_\ell = \begin{cases} \boxed{} & \text{for } \ell \in \{ \boxed{} \}, \\ \boxed{} & \text{for } \ell \in \{ \boxed{} \}, \end{cases},$$

$$(\mathbf{D})_{k,k} = \boxed{}, \quad k = 0, \dots, n-1.$$

SOLUTION of (0-3.a):

Remark. The formula (0.3.2) is derived as follows

$$\begin{aligned}
 f_k &= \sum_{j=0}^{n-1} a_j \cos\left(\pi \frac{kj}{n}\right) = \frac{1}{2} \sum_{j=0}^{n-1} a_j \left(e^{-\pi i \frac{kj}{n}} + e^{\pi i \frac{kj}{n}} \right) \\
 &= \sum_{j=0}^{n-1} \frac{1}{2} a_j e^{-2\pi i \frac{kj}{2n}} + \sum_{\ell=0}^{n-1} \frac{1}{2} a_{n-1-\ell} e^{2\pi i \frac{k(n-1-\ell)}{2n}} \\
 &= \sum_{j=0}^{n-1} \frac{1}{2} a_j e^{-2\pi i \frac{kj}{2n}} + e^{\pi i \frac{k(n-1)}{n}} \sum_{\ell=0}^{n-1} \frac{1}{2} a_{n-1-\ell} e^{-2\pi i \frac{k\ell}{2n}}
 \end{aligned}$$

Matching this with (0.3.3) we find

$$\begin{aligned}
 (\mathbf{x})_j &= \begin{cases} \frac{1}{2}a_j & \text{for } j \in \{0, \dots, n-1\}, \\ 0 & \text{for } j \in \{n, \dots, 2n-1\}, \end{cases}, \quad (\mathbf{y})_j = \begin{cases} \frac{1}{2}a_{n-1-j} & \text{for } j \in \{0, \dots, n-1\}, \\ 0 & \text{for } j \in \{n, \dots, 2n-1\}, \end{cases}, \\
 (\mathbf{D})_{k,k} &= \exp\left(\pi i k \frac{n-1}{n}\right), \quad \ell = 0, \dots, n-1.
 \end{aligned} \tag{0.3.4}$$



(0-3.b) (10 pts.) The C++ function `modcostrf()` realizes the mapping $\mathbf{a} \rightarrow \mathbf{f}$ according to (0.3.1). Supplement the missing parts of the following listing by writing valid C++ code in the boxes.

```

Eigen::VectorXcd modcostrf(const Eigen::VectorXcd &a) {
    using Comp = std::complex<double>;
    unsigned int n = a.size();
    Eigen::VectorXcd f(n);
    Eigen::FFT<double> fft;

    Eigen::VectorXcd tmp( );

    tmp << , Eigen::VectorXcd::Zero(n);

    Eigen::VectorXcd v1 = fft. ;
    tmp << 0.5 * a.reverse(), ;
    Eigen::VectorXcd v2 = fft. ;

    Comp fac = std::exp(((n - 1) * M_PI * Comp(0, 1)) / (double)n);
    Comp d(1.0, 0.0);
    for (int k = 0; k < n; ++k) {
        f[k] = (v1[ ] +  * v2[ ]) .real();
        *= ;
    }
    return f;
}

```

The constant `M_PI` is the number π .

HINT 1 for (0-3.b): The EIGEN helper class **Eigen::FFT<double>** provides the member functions `fwd()` for DFT and `inv()` for inverse DFT. ┘

HINT 2 for (0-3.b): The member function `reverse()` of **Eigen::VectorXd** reverses the order of the vector components. ┘

SOLUTION of (0-3.b):

The following code is a straightforward implementation of (??) making use of the fact that the DFT is available through the method `fwd()` of an **Eigen::DFT** object. The final loop also performs the multiplication with **D**, which is a simple component-wise scaling.

Note that (0.3.4) means that

$$(\mathbf{D})_{\ell,\ell} = \left(\exp\left(\pi i \frac{(n-1)}{n}\right) \right)^\ell,$$

such that the scaling factors can be obtained by successive multiplication of 1 with `fac = $\exp\left(\pi i \frac{(n-1)}{n}\right)$` .

C++ code 0.3.5: Function `modcostrf()`

```

2  Eigen::VectorXd modcostrf(const Eigen::VectorXd &a) {
3      using Comp = std::complex<double>;
4      unsigned int n = a.size();
5      Eigen::VectorXd f(n);
6      Eigen::FFT<double> fft;
7      Eigen::VectorXcd tmp(2 * n);
8      tmp << 0.5 * a, Eigen::VectorXcd::Zero(n);
9      Eigen::VectorXcd v1 = fft.fwd(tmp);
10     tmp << 0.5 * a.reverse(), Eigen::VectorXcd::Zero(n);
11     Eigen::VectorXcd v2 = fft.fwd(tmp);
12     Comp fac = std::exp(((n - 1) * M_PI * Comp(0, 1)) / (double)n);
13     Comp d(1.0, 0.0);
14     for (int k = 0; k < n; ++k) {
15         f[k] = (v1[k] + d * v2[k]).real();
16         d *= fac;
17     }
18     return f;
19 }
```



End Problem 0-3, 20 pts.