Numerische Mathematik

für Studiengang Rechnergestützte Wissenschaften

Profs. Ralf Hiptmair und Rolf Jeltsch

Draft version 15. Januar 2007, Subversion rev #1769

http://www.sam.math.ethz.ch/~hiptmair/tmp/NCSE_06.pdf

Inhaltsverzeichnis

Reporting errors

Fehler in den Vorlesungsunterlagen bitte per Wiki melden!

`http://elbanet.ethz.ch/wikifarm/rhiptmair/index.php?n=Main.NCSECourse`

Bitte folgende Angaben in den Wiki eintragen:

- Abschnitt, in dem der Fehler gefunden wurde.
- Genauer Ort des Fehlers (z.B. „nach Gleichung (4)")
- Kurzbeschreibung des Fehlers

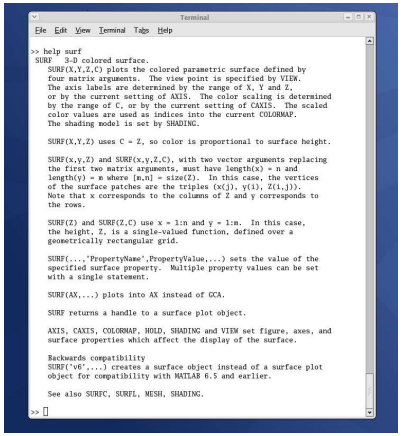
MATLAB

Beispielrechnungen und Übungen zur Vorlesung stützen sich auf die für numerische Berechnungen konzipierte Hochsprache MATLAB („Matrix Laboratory") von der Firma MATHWORKS.

Matlab ist eine Interpretersprache und verfügt über folgende Möglichkeiten:

- Mächtige Funktionen für Vektor- und Matrixarithmetik und -manipulationen
- Kontrollstrukturen, wie `if` ... `else` ... `end`, `for/while` ... `end`
- Modularisierung durch Unterprogramme
- Bibliotheken für komplexe numerische Aufgaben
- Umfangreiche Visualisierungsfunktionen

Einarbeitung in MATLAB: Am besten durch „Learning by doing" unter Zuhilfenahme der ausführlichen Online-Hilfsmenüs.



MATLAB-LINKS:
Matlab Online Documentation from the Mathworks
MATLAB guide
MATLAB Primer

0.1 Danksagung

Wertvolle Beitrage zur Entstehung und Verbesserung der vorliegenden Präsentationsfolien haben geleistet:

- Herr Martin Knoller Stocker (Student RW): Erstellung des Index und der Verzeichnisse im SS05
- Herr Christian Heitzmann (Student RW): Umfangreiche Fehlerlisten im WS 04/05
- Frau Gisela Widmer (Doktorandin RW): Korrekturlesen als Assistentin der Vorlesung im WS 04/05, WS 05/06, WS 06/07
- Herr Henning Avenhaus und Herr Christoph Scherrer (Stundenten RW): Aufarbeitung der vorlesungsbegleitenden MATLAB-Skripten

1 Computerarithmetik und Konsequenzen

[chapter-computerarithmetik-und-konsequenzen.tex, SVN: chapter-computerarithmetik-und-konsequenzen.tex 1035 2006-10-09 11:41:18Z kalai]

Viele Beispiele aus diesem Kapitel sind den ersten Kapiteln von [16] und [28] entnommen. Zum weiterführenden Studium sind Kapitel 2 von [12] und das Buch [29] empfohlen.

1.1 Beispiele

[File: section-beispiele.tex, SVN: chapter-computerarithmetik-und-konsequenzen.tex 1035 2006-10-09 11:41:18Z kalai]

Rechner rechnen falsch !

Numerik $\neq$ Analysis

Beispiel 1 (Berechnung der Eulerschen Zahl).

Bekannt aus Analysis I:

$$e = \lim_{n\to\infty} \left(1+\frac{1}{n}\right)^n$$

MATLAB-CODE: Näherung für e

```
Approximation of e
for i=1:15
  n = 10^i; r = (1+1/n)^n;
  fprintf('10^%2d %20.15f ...
        %20.15f\n', i,r,r-exp(1));
end
```

n	Näherung e_n	Fehler $e_n - e$
10^1	2.593742460100002	-0.124539368359044
10^2	2.704813829421529	-0.013467999037517
10^3	2.716923932235520	-0.001357896223525
10^4	2.718145926824356	-0.000135901634689
10^5	2.718268237197528	-0.000013591261517
10^6	2.718280469156428	-0.000001359302618
10^7	2.718281693980372	-0.000000134478673
10^8	2.718281786395798	-0.000000042063248
10^9	2.718282030814509	0.000000202355464
10^{10}	2.718282053234788	0.000000224775742
10^{11}	2.718282053357110	0.000000224898065
10^{12}	2.718523496037238	0.000241667578192
10^{13}	2.716110034086901	-0.002171794372145
10^{14}	2.716110034087023	-0.002171794372023
10^{15}	3.035035206549262	0.316753378090216

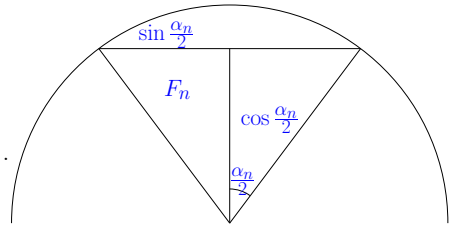
◇

Beispiel 2 („Quadratur des Kreises").

COMPUTE Approximation durch reguläres n-Eck, $n \in \mathbb{N}$:

Fläche des n-Ecks:

$$A_n = n\cos\frac{\alpha_n}{2}\sin\frac{\alpha_n}{2} = \frac{n}{2}\sin\alpha_n = \frac{n}{2}\sin\left(\frac{2\pi}{n}\right) .$$



REKURSION Rekursion für A_n abgeleitet von

$$\sin\frac{\alpha_n}{2} = \sqrt{\frac{1-\cos\alpha_n}{2}} = \sqrt{\frac{1-\sqrt{1-\sin^2\alpha_n}}{2}} ,$$

Startwert: $A_6 = \frac{3}{2}\sqrt{3}$.

◇

MATLAB-CODE: Instabile Rekursion für Pi

```
s = sqrt(3)/2; A=3*s; n = 6;   % initialization
z = [n A A-pi s];              % matrix for storing results
while (s > 1e-10)              % terminate when s is small
  s = sqrt((1-sqrt(1-s*s))/2); % Recursion
  n=2*n; A = n/2*s;
  z = [z; n A A-pi s];         % store results
end
for i=1:length(z)
  fprintf('%10d %20.15f %20.15f %20.15f\n', ...
          z(i,1),z(i,2),z(i,3),z(i,4));
end
```

n	A_n	$A_n - \pi$	$\sin \alpha_n$
6	2.598076211353316	-0.543516442236477	0.866025403784439
12	3.000000000000000	-0.141592653589794	0.500000000000000
24	3.105828541230250	-0.035764112359543	0.258819045102521
48	3.132628613281237	-0.008964040308556	0.130526192220052
96	3.139350203046872	-0.002242450542921	0.065403129230143
192	3.141031950890530	-0.000560702699263	0.032719082821776
384	3.141452472285344	-0.000140181304449	0.016361731626486
768	3.141557607911622	-0.000035045678171	0.008181139603937
1536	3.141583892148936	-0.000008761440857	0.004090604026236
3072	3.141590463236762	-0.000002190353031	0.002045306291170
6144	3.141592106043048	-0.000000547546745	0.001022653680353
12288	3.141592516588155	-0.000000137001638	0.000511326906997
24576	3.141592618640789	-0.000000034949004	0.000255663461803
49152	3.141592645321216	-0.000000008268577	0.000127831731987
98304	3.141592645321216	-0.000000008268577	0.000063915865994
196608	3.141592645321216	-0.000000008268577	0.000031957932997
393216	3.141592645321216	-0.000000008268577	0.000015978966498
786432	3.141593669849427	0.000001016259634	0.000007989485855
1572864	3.141592303811738	-0.000000349778055	0.000003994741190
3145728	3.141608696224804	0.000016042635011	0.000001997381017
6291456	3.141586839655041	-0.000005813934752	0.000000998683561
12582912	3.141674265021758	0.000081611431964	0.000000499355676
25165824	3.141674265021758	0.000081611431964	0.000000249677838
50331648	3.143072740170040	0.001480086580246	0.000000124894489
100663296	3.159806164941135	0.018213511351342	0.000000062779708
201326592	3.181980515339464	0.040387861749671	0.000000031610136
402653184	3.354101966249685	0.212509312659892	0.000000016660005
805306368	4.242640687119286	1.101048033529493	0.000000010536712
1610612736	6.000000000000000	2.858407346410207	0.000000007450581

Beispiel 3 (Auswertung der Exponentialfunktion).

EXPONENTIALREIHE

Global konvergente Exponentialreihe :

$$\exp(x) = \sum_{k=0}^{\infty} \frac{x^k}{k!} = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \ldots .$$

MATLAB: Summation Exp.-Reihe

```
function y = expeval(x,tol)
% Evaluation of exponential series
s=1; term=1; k=1; % Initialization
while (abs(term)>tol*abs(s))
  term = term*x/k;
  s = s + term;
  k = k+1;
end
y = s;
```

Numerisches *Experiment*: Untersuchung des relativen Fehlers im Ergebnis für `tol`$= 10^{-8}$

Definition 1.1.1 (Relativer Fehler). $f(x) \in \mathbb{C} \hat{=}$ *(theoretisch) exaktes Resultat einer Berechnung,* $\widetilde{f}(x) \in \mathbb{C} \hat{=}$ *Näherung. Falls* $f(x) \neq 0$ ➤ RELFEHLER *relativer Fehler (engl.: relative error) gegeben durch*

$$\epsilon_r := \frac{|f(x) - \widetilde{f}(x)|}{|f(x)|}$$

Warum betrachten wir den relative Fehler?

Relativer Fehler eines Resultats $\leftrightarrow$ Anzahl der gültigen Stellen

x	Approximation $\widetilde{\exp}(x)$	$\exp(x)$	$\frac{\lvert\exp(x)-\widetilde{\exp}(x)\rvert}{\exp(x)}$
-20	5.6218844674e-09	2.0611536224e-09	1.727542676201181
-18	1.5385415977e-08	1.5229979745e-08	0.010205938187564
-16	1.1254180496e-07	1.1253517472e-07	0.000058917020257
-14	8.3152907681e-07	8.3152871910e-07	0.000000430176956
-12	6.1442133148e-06	6.1442123533e-06	0.000000156480737
-10	4.5399929556e-05	4.5399929762e-05	0.000000004544414
-8	3.3546262817e-04	3.3546262790e-04	0.000000000788902
-6	2.4787521758e-03	2.4787521767e-03	0.000000000333306
-4	1.8315638879e-02	1.8315638889e-02	0.000000000530694
-2	1.3533528320e-01	1.3533528324e-01	0.000000000273603
0	1.0000000000e+00	1.0000000000e+00	0.000000000000000
2	7.3890560954e+00	7.3890560989e+00	0.000000000479969
4	5.4598149928e+01	5.4598150033e+01	0.000000001923058
6	4.0342879295e+02	4.0342879349e+02	0.000000001344248
8	2.9809579808e+03	2.9809579870e+03	0.000000002102584
10	2.2026465748e+04	2.2026465795e+04	0.000000002143800
12	1.6275479114e+05	1.6275479142e+05	0.000000001723845
14	1.2026042798e+06	1.2026042842e+06	0.000000003634135
16	8.8861105010e+06	8.8861105205e+06	0.000000002197990
18	6.5659968911e+07	6.5659969137e+07	0.000000003450972
20	4.8516519307e+08	4.8516519541e+08	0.000000004828738

Beispiel 4 (Summation der Harmonischen Reihe). $s := \sum_{k=1}^{100000} \frac{1}{k}$

MATLAB-CODE: Summation vorwärts

```
s = 0;
for i = 1:1e6
  s = s+1/i;
end
```

← ? →

„+" assoziativ !?

MATLAB-CODE: Summation rückwärts

```
s = 0;
for i = 1e6:-1:1
  s = s+1/i;
end
```

➢ $s = 14.392726722864\mathbf{98889}$ ➢ $s = 14.392726722865\mathbf{77226}$

[„Exakte" Auswertung durch MAPLE-Code: ➢ $s = 14.392726722865723631381127\mathbf{49}\ldots$]

Beispiel 5 (Probleme bei Polynomauswertung).

p: Polynom, Grad 19

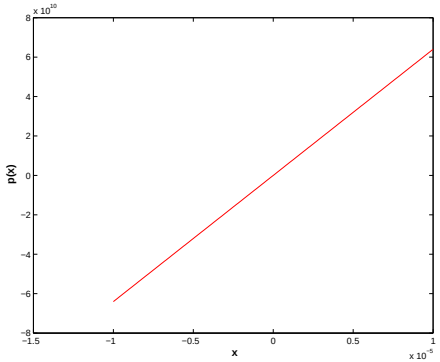
$$p(x) = x^{19} + \alpha_{18}x^{18} + \cdots + \alpha_1 x + \alpha_0$$

mit Nullstellen $0, 1, 2, \ldots, 18$

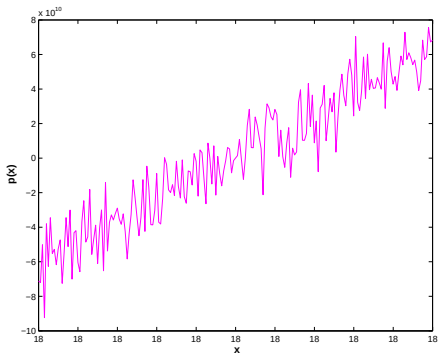
➢ Symmetrie $p(9-x) = p(9+x)$

MATLAB-CODE: Polynomauswertung

```
p = poly(0:18);
x = -1E-5:1E-7:1E-5;
y = polyval(p,x);
plot(x,y,'r-');
y = polyval(p,x+18);
plot(x+18,y,'m-');
```



Erwartet



"Numerisches Rauschen"

1.2 Zahldarstellung

[File: section-zahldarstellung.tex, SVN: section-zahldarstellung.tex 1371 2007-04-04 10:42:11Z hiptmair]

ZAHLDARSTELLUNG

Computer = endlicher Automat ➢ rechnet mit endlich vielen Zahlen

Bemerkung 6 (Zahldarstellung durch Exponent und Mantisse: "wissenschaftliche Notation").

Basis $B \in \mathbb{N} \setminus \{1\}$ fest: $\forall x \in \mathbb{R}$: $\exists_1 B^{-1} \le d < 1, \exists_1 E \in \mathbb{Z}$: $x = d \cdot B^E$. (Mantisse, Exponent)

△

MSTELLDEZZAHL

Beispiel 7 (m-stellige Dezimalzahlen).

3-stellige Dezimalzahlen $B = 10, 0.1 \le d < 1$:

Gültig : $0.145 \cdot 10^5$, $-0.100 \cdot 10^7$, $0.450 \cdot 10^{-20}$

Ungültig : $1.345 \cdot 10^2$, $0.001 \cdot 10^4$

◇

☞ Computer speichern Mantissen und Exponenten nur mit *endlich vielen Stellen*:

MASCHNR MANTISSE EXPONENT

Definition 1.2.1 (Maschinenzahlen). *Die Menge $\mathbb{M}$ der Maschinenzahlen (engl. machine numbers) zur Basis $B \in \mathbb{N} \setminus \{1\}$, mit m-stelliger Mantisse d, $m \in \mathbb{N}$, und Exponentenbereich $\{e_{\min}, \ldots, e_{\max}\}$, $e_{\min}, e_{\max} \in \mathbb{Z}$, ist gegeben durch*

$$\mathbb{M} := \{d \cdot B^E : d = i \cdot B^{-m}, i = B^{m-1}, \ldots, B^m - 1, E \in \{e_{\min}, \ldots, e_{\max}\}\}$$

Maschinenzahl $\in \mathbb{M}$: $x = \pm \boxed{0} . \boxed{}\boxed{}\boxed{}\boxed{} \ldots \boxed{}\boxed{} \cdot B \; \boxed{}\boxed{} \ldots \boxed{}\boxed{}$

Niemals $= 0$!

m Mantissenstellen

Exponentenstellen

➢ Betragsgrösste Maschinenzahl : $x_{\max} = (1 - B^{-m}) \cdot B^{e_{\max}}$

Betragskleinste Maschinenzahl : $x_{\min} = B^{-1} \cdot B^{e_{\min}}$

Bemerkung 8. Endlichkeit von $\mathbb{M}$ ➢ (Unvermeidliche) Eingabefehler (z.B. für $x = \pi$)

△

IEEESTANDARD GLEITKOMMAZAHL

Beispiel 9 (IEEE Standard 754 für Gleitkommazahlen*). (engl. *floating point numbers*) → OVE01 [35]

Natürlich: $B = 2$ (Binärsystem)

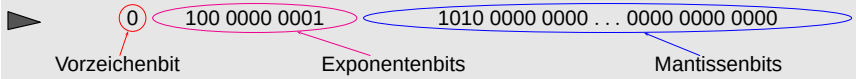
EINFACHEGENAUIGKEIT
Einfache Genauigkeit (*single precision*) : $m = 24^*, E \in \{-125, \ldots, 128\}$ ➢ 4 Byte
DOPPELTEGENAUIGKEIT
Doppelte Genauigkeit (*double precision*) : $m = 53^*, E \in \{-1021, \ldots, 1024\}$ ➢ 8 Byte

⋆: Konsequent wird in dieser Vorlesung ein Dezimalpunkt verwendet.
∗: Ein Vorzeichenbit ist inbegriffen.

◇

Bemerkung 10 (Interne Binärdarstellung einer Gleitkommazahl doppelter Genauigkeit (MATLAB)).

```
>> format hex
>> 6.5
ans = 401a000000000000
```



$$x = (-1)^0 \cdot (\frac{1}{2} + \frac{1}{4} + \frac{1}{16}) \times 2^{1026-1023} = 6.5\ .$$

△

Bemerkung 11 (Sonderfälle im IEEE Standard).

```
>> x = exp(1000), y = 3/x, z = x*sin(pi), w = x*log(1)
x = Inf
y = 0
z = Inf
w = NaN
```

NAN NOTANUMBER
$E = e_{\max}, M \neq 0 \mathrel{\hat{=}}$ NaN = *Not a number* → Ausnahme (engl. *exception*)
INF
$E = e_{\max}, M = 0 \mathrel{\hat{=}}$ Inf = *Infinity* → Überlauf (→ Abschnitt 1.5) sec:ueber-und-unterlauf
UNNORMZAHLEN
$E = 0 \mathrel{\hat{=}}$ Unnormalisierte Zahlen → Unterlauf (→ Abschnitt 1.5) sec:ueber-und-unterlauf
$E = 0, M = 0 \mathrel{\hat{=}}$ Gleitkommazahl 0

△

Beispiel 12. Charakteristische Grössen der IEEE Gleitpunktarithmetik (doppelte Genauigkeit)

☞ Relevant für Rechnungen mit MATLAB

Parameter der Gleitkommaarithmetik in MATLAB

```
>> format hex; realmin, format long; realmin
ans = 0010000000000000
ans = 2.225073858507201e-308
```



```
>> format hex; realmax, format long; realmax
ans = 7fefffffffffffff
ans = 1.797693134862316e+308
```

◇

Konsequenzen der Gleitpunktarithmetik sind aktuelles Forschungsthema: BPZ07 [5], siehe auch die Monographie HIG96 [29]

1.3 Gleitpunktarithmetik und Rundungsfehler

[File: section-gleitpunktarithmetik-und-rundungsfehler.tex, SVN: section-zahldarstellung.tex 1371 2007-04-04 10:42:11Z hiptmair]

GLEITPUNKTARITHMETIK

1. Beobachtung: Maschinenzahlenmenge $\mathbb{M}$ (→ Def. def:MZ 1.2.1) ist endlich !

Beispiel 13. Abbruchkriterium von Iterationen unter Ausnutzung von Gleitpunktarithmetik:
WURZELITERATION
Wurzeliteration für $a > 0$:

$$x^{(k+1)} = \frac{1}{2}(x^{(k)} + \frac{a}{x^{(k)}}) \quad \to \sqrt{a} \quad \text{for } k \to \infty\ .$$

Monotonie:
$x^{(0)} > \sqrt{a} \succ \sqrt{a} < x^{(k)} < x^{(k-1)}\ \forall k \in \mathbb{N}$.
MASCHUNABGABKRIT
Maschinenunabhängiges Abbruchkriterium

MATLAB-CODE: Wurzeliteration

```
function y=squareroot(a)
xo = 0.5*(1+a);
xn = (xo+a/xo)/2;
while (xn<xo)
  xo =xn; xn = (xo+a/xo)/2;
end
y = (xo+xn)/2;
```

◇

2. Beobachtung: Maschinenzahlenmenge $\mathbb{M}$ (→ Def. def:MZ 1.2.1) nicht abgeschlossen unter den elementaren arithmetischen Operationen $+, -, \cdot, /$.

RUNDFEHLER
Unvermeidlich: Rundungsfehler

Rundung:

$$\mathrm{rd}(x) = \arg\min_{\widetilde{x} \in \mathbb{M}} |x - \widetilde{x}|$$

(grössere Gleitkommazahl bei Nichteindeutigkeit, “Aufrunden”)

Beispiel 14 (Eingabefehler und Rundungsfehler).

MATLAB-CODE: Demonstration von Rundungsfehlern

```
>> format long;
>> a = 4/3; b = a-1; c = 3*b; e = 1-c
e = 2.220446049250313e-16
```

◇

Notation: Realisierung von $\star \in \{+,-,\cdot,/\}$ in Gleitpunktarithmetik $\rightarrow$ $\widetilde{\star}$

Definition 1.3.1 (Maschinengenauigkeit)**.** *Die Maschinengenauigkeit (engl. machine precision) eines Maschinenzahlensystems nach Def. 1.2.1 ist gegeben durch*

$$\texttt{eps} := \tfrac{1}{2}B^{-m+1} .$$

$x_{\min} \le x \le x_{\max} \quad \Rightarrow$ Relativer Rundungsfehler $\dfrac{|\mathrm{rd}(x)-x|}{|x|} \le \texttt{eps}$.

Beispiel 15 (Abfrage der Maschinengenauigkeit in MATLAB). (Prozessor Intel Pentium)

Abfrage der Maschinengenauigkeit in MATLAB

```
>> format hex; eps, format long; eps
ans = 3cb0000000000000
ans = 2.220446049250313e-16
```

◇

„Axiom" der Gleitpunktarithmetik: Für die elementaren arithmetischen Operationen $\star \in \{+,-,\cdot,/\}$ und elementare Funktionen $f \in \{\exp,\sin,\cos,\log,\ldots\}$ gilt

$$x \,\widetilde{\star}\, y = (x \star y)(1+\delta) \quad , \quad \widetilde{f}(x) = f(x)(1+\delta) \quad \forall x,y \in \mathbb{M} ,$$

mit $|\delta| < \texttt{eps}$, eps = Maschinengenauigkeit (→ Def. 1.3.1)

► Relative Rundungsfehler elementarer Rechenschritte beschränkt durch Maschinengenauigkeit

Bemerkung 16 (Realisierung von $\widetilde{+},\widetilde{-},\widetilde{\cdot},\widetilde{/}$).

$$\star \in \{+,-,\cdot,/\}: \qquad x \,\widetilde{\star}\, y := \mathrm{rd}(x \star y) \tag{1.3.1}$$

△

! Warum sollen uns diese winzigen Rundungsfehler kümmern ?

Sie sollten (→ siehe Abschnitt 1.1): • Akkumulation von Rundungsfehlern
• Verstärkung von Rundungsfehlern

Bemerkung 17. Rundung ➤ Operationen $\widetilde{+},\widetilde{\cdot}: \mathbb{M}\times\mathbb{M} \mapsto \mathbb{M}$ nicht assoziativ ! (→ Beispiel 4)

△

Beispiel 18 (Nichtassoziativität der Maschinenaddition). 3-stellig Dezimalarithmetik, $\widetilde{+}$ gemäss (1.3.1)

➢ $0.100\cdot 10^4 \widetilde{+} 1 = \mathrm{rd}(0.1001\cdot 10^4) = 0.100\cdot 10^4$!

► $1000\widetilde{+}\underbrace{1\widetilde{+}\cdots\widetilde{+}1}_{1000\times} = 1000 \quad \leftrightarrow \quad \underbrace{1\widetilde{+}\cdots\widetilde{+}1}_{1000\times}\widetilde{+}1000 = 2000$.

◇

Bemerkung 19 (Numerische Nullabfrage).

Eine Konsequenz von Rundungsfehlern:

Abfragen wie `if (x == 0)` oft sinnlos/gefährlich, falls x Rechenergebnis

► Besser: Abfrage `if (abs(x) < eps*s) ...`,
s = positive Zahl, relativ zu der x „klein" sein soll.

△

Übung 1.4. Gauge the relative error in the function evaluation `x*log(x)` (can be plotted in MATLAB with `ezplot(@(x) x.*log(x),[0,5])`)

1.5 Über- und Unterlauf

[File: section-ueber-und-unterlauf.tex, SVN: section-gleitpunktarithmetik-und-rundungsfehler.tex 1081 2006-10-27 14:13:07Z hiptmair]

Überlauf (engl. *overflow*) $\widehat{=}$ |Resultat einer elementaren Operation| $> \max\{\mathbb{M}\}$
$\widehat{=}$ IEEE Standard $\Rightarrow$ `Inf`

Unterlauf (engl. *underflow*) $\widehat{=}$ $0 <$ |Resultat einer elementaren Operation| $< \min\{|\mathbb{M}\setminus\{0\}|\}$
$\widehat{=}$ IEEE Standard $\Rightarrow$ unnormalisierte Zahlen (!)

Bemerkung 20. Axiom der Gleitpunktarithmetik gilt nicht bei Unterlauf: MATLAB Beispiel

MATLAB-CODE: Unterlauf

```
>> format long; res=pi*realmin/123456789101112
res = 5.681754927174335e-322
>> res=res*123456789101112/realmin
res = 3.15248510554597
```

Vermeide Über- und Unterlauf wann immer möglich !

Beispiel 21 (Vermeidung von Überlauf).

$$r = \sqrt{x^2 + y^2}$$

Naive Auswertung: Überlauf für $x \gg 1$, $y \gg 1$

$$r = \begin{cases} |x|\sqrt{1+(y/x)^2} & \text{wenn } |x| \geq |y| \,, \\ |y|\sqrt{1+(x/y)^2} & \text{wenn } |y| > |x| \,. \end{cases}$$

➢ Überlauf vermieden

1.6 Kondition

[File: section-kondition.tex, SVN: section-kondition.tex 1037 2006-10-09 13:23:05Z kalai]

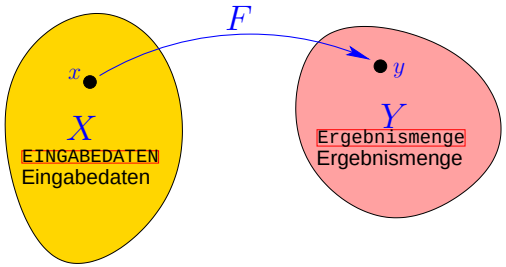
KONDITION

Die Kondition ist eine intrinsische Eigenschaft eines Problems

Hier:

KONDPROBLEM
Problem = Mathematisch wohldefinierte Vorschrift, wie aus Eingabedaten (Input) ein Ergebnis (Output) erhalten werden kann.

X : Datenmenge
Y : Ergebnismenge
F : „Problemfunktion" $X \mapsto Y$
d_X : Metrik auf X
d_Y : Metrik auf Y
Die Metriken (Abstandsmasse) auf X und Y sind „sinnvoll" zu wählen.

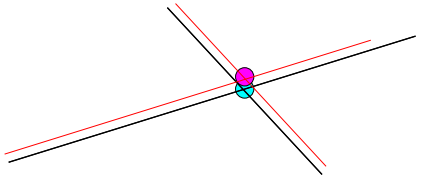


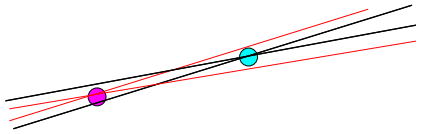
Definition 1.6.1 (Gut konditioniertes Problem)**.** *Ein Problem $F : X \mapsto Y$ heisst gut konditioniert zum Input $x \in X$ (engl.* well conditioned*), wenn „kleine Störungen in den Eingabedaten ebenso kleine Störungen im Ergebnis hervorrufen", das heisst*

$$\exists \kappa_{\text{abs}} \approx 1: \quad d_Y(F(x), F(y)) \leq \kappa_{\text{abs}} d_X(x, y) \quad \forall y \in \mathcal{U}(x)$$

für eine Umgebung $\mathcal{U}(x)$ von x in X.
ABSCONDNR
κ_{abs} = *absolute Konditionszahl, misst Verstärkung von Fehlern in den Eingabedaten.*

Beispiel 22. Schnitt von Geraden (mit Abstandsmetrik)





"Schleifender" Schnitt: schlecht konditioniert

DIFFKONDANAL
Differentielle Konditionsanalyse: betrachte Spezialfall: $X, Y \subset \mathbb{R}$, Betragsmetrik

Falls f differenzierbar: $\delta y := f(x + \delta x) - f(x) = f'(\xi)\delta x$, mit $\xi \in]x, x + \delta x[$

Falls f stetig differenzierbar: $\delta y = f'(x)\delta x + O(\delta x^2)$

Falls $x \neq 0$, $y := f(x) \neq 0$, f stetig differenzierbar

$$\frac{\delta y}{y} \approx f'(x)\frac{\delta x}{f(x)} = \left(f'(x)\frac{x}{f(x)}\right)\frac{\delta x}{x} \,.$$

RELCONDNR
➢ $\kappa_{\text{rel}} := \left| f'(x)\frac{x}{f(x)} \right|$ = relative Konditionszahl, misst Verstärkung relativer Fehler

Beispiel 23 (Kondition von Nullstellen).

$f : I \subset \mathbb{R} \mapsto \mathbb{R}$ 2-fach stetig differenzierbar mit Nullstelle (engl. *zero*) $s \in I$: $f(s) = 0$.

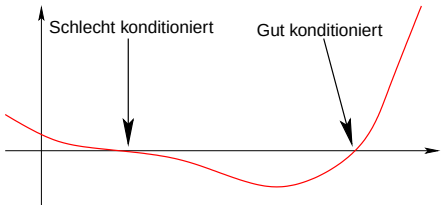
Additive Störung: $\widetilde{f} := f + \epsilon$ ➢ $s = s(\epsilon)$, $\widetilde{f}(s(\epsilon)) = 0$

Implizites Differenzieren von $f(s(\epsilon)) = -\epsilon$

$$f'(s(\epsilon))\frac{ds}{d\epsilon}(\epsilon) = -1 \quad \Rightarrow \quad \frac{ds}{d\epsilon}_{|\epsilon=0} = -\frac{1}{f'(s)} \,.$$

Je "flacher" die Funktion desto schlechter die Kondition der Nullstelle bzgl. additiver Störungen

(betrifft **nicht** Funktions*auswertung* !)



Kondition elementarer Operationen:

• Multiplikation $f(x) = a \cdot x$, $a \in \mathbb{R}$

$$\kappa_{\rm rel} = \left| f'(x) \frac{x}{f(x)} \right| = 1 \; ! \qquad (1.6.1)$$ Condmulti

➢ Keine Verstärkung relativer Fehler durch Multiplikation

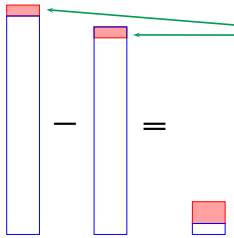
• Subtraktion $f(x) = x - a$, $a \in \mathbb{R}$. Relative Konditionszahl für $x \notin \{0, a\}$

$$\kappa_{\rm rel} = \left| f'(x) \frac{x}{f(x)} \right| = \left| \frac{x}{x-a} \right| \quad \text{➢} \quad \boxed{\kappa_{\rm rel} \gg 1, \text{ falls } x \approx a} \qquad (1.6.2)$$ Condsubi

1.7 Auslöschung

[File: section-ausloeschung.tex, SVN: section-ausloeschung.tex 1073 2006-10-24 19:22:33Z hiptmair]

AUSLOESCHUNG



Auslöschung (engl. *cancellation*)

$\hat{=}$

Subtraktion zweier ungefähr gleich grosser Zahlen

(➢ extreme Verstärkung relativer Fehler, → (1.6.2)) condsubtract

Subtraktionen mit Auslöschung geschehen selbst rundungsfehlerfrei !

Beispiel 24 (Auslöschung). Rechnung mit 2-stelliger Dezimalarithmetik:

$$a = 1.3 \quad , \quad b = 1.2$$
$$a^2 - b^2 = (a+b)(a-b)$$

ALGA

Algorithmus A	Algorithmus B
$x := a \tilde{\cdot} a = 1.7$ (gerundet)	$x := a \tilde{+} b = 2.5$ (exakt)
$y := b \tilde{\cdot} b = 1.4$ (gerundet)	$y := a \tilde{-} b = 0.1$ (exakt)
$x \tilde{-} y = 0.30$ (exakt)	$x * y = 0.25$ (exakt)

◣ Subtraktion im letzten Schritt ➢ grosse Verstärkung der (relativen) Rundungsfehler aus den ersten beiden Schritten

„Auslöschungsverdächtige" Subtraktionen möglichst zu Beginn eines Algorithmus !
(Allgemein: Schlecht konditionierte Elementarschritte möglichst früh !)

Formale Analyse für $a, b, \in \mathbb{R}$: $f = a^2 - b^2$

Mit $\delta_1, \delta_2, \delta_3 \in \mathbb{R}$, $|\delta_i| < \texttt{eps}$ (→ Axiom der Gleitpunktarithmetik)
+ Vernachlässigung von Termen höherer Ordnung in eps ($\hat{=}$ Linearisierung)

Algorithmus A:

$$x = a^2(1+\delta_1)\,, \; y = b^2(1+\delta_2)$$
$$\widetilde{f} = (a^2(1+\delta_1) - b^2(1+\delta_2))(1+\delta_3) = f + a^2\delta_1 + b^2\delta_2 + (a^2-b^2)\delta_3 + O(\texttt{eps}^2)$$

$$\frac{|\widetilde{f} - f|}{|f|} \leq \texttt{eps} \frac{a^2 + b^2 + |a^2 - b^2|}{|a^2 - b^2|} + O(\texttt{eps}^2) = \texttt{eps}\left(1 + \frac{|a^2+b^2|}{|a^2-b^2|}\right) + O(\texttt{eps}^2)\,. \qquad (1.7.1)$$ ALGA

Vernachlässigbar klein

LANDAUO

Notation („Landau-O"): $f(h) = O(h^\alpha) \Leftrightarrow \exists C > 0, h_0 > 0 \colon |f(x)| \leq C|h|^\alpha$ für alle $|h| < h_0$.

Algorithmus B:

$$x = (a+b)(1+\delta_1)\,, \; y = (a-b)(1+\delta_2)$$
$$\widetilde{f} = (a+b)(a-b)(1+\delta_1)(1+\delta_2)(1+\delta_3) = f + (a^2-b^2)(\delta_1+\delta_2+\delta_3) + O(\texttt{eps}^2)$$

$$\frac{|\widetilde{f} - f|}{|f|} \leq |\delta_1 + \delta_2 + \delta_3| + O(\texttt{eps}^2) \leq 3\texttt{eps} + O(\texttt{eps}^2)\,. \qquad (1.7.2)$$ ALGB

◣ Relativer Ergebnisfehler immer $\approx$ eps **!**

◇

Beispiel 25 (Fortsetzung von „Berechnung der Exponentialfunktion", Beispiel 3). ex:exp

Für $x > 0$:

$$\exp(-x) = \sum_{k=0}^{\infty} (-1)^k \frac{x^k}{k!}$$
$$= 1 - x + \frac{x^2}{2} - \frac{x^3}{6} + \frac{x^4}{24} - \ldots .$$

Alternierende Reihe

➤ Ständige Auslöschung bei Summation

Partialsummen für $x = 20 \rightarrow$ partsum

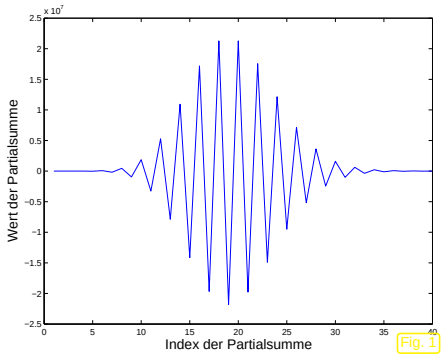


Fig. 1

Beispiel 26 (Fortsetzung „Quadratur des Kreises", Beispiel 2). ex:circarea

Rekursionsformel aus Beispiel 2: ex:circarea

$$\sin\frac{\alpha_n}{2} = \sqrt{\frac{1-\cos\alpha_n}{2}} = \sqrt{\frac{1-\sqrt{1-\sin^2\alpha_n}}{2}}$$

Für $\alpha_n \ll 1$: $\sqrt{1-\sin^2\alpha_n} \approx 1$

Auslöschung !

Umformung in auslöschungsfreie Formel:

$$\sin\frac{\alpha_n}{2} = \sqrt{\frac{1-\sqrt{1-\sin^2\alpha_n}}{2}} = \sqrt{\frac{1-\sqrt{1-\sin^2\alpha_n}}{2} \cdot \frac{1+\sqrt{1-\sin^2\alpha_n}}{1+\sqrt{1-\sin^2\alpha_n}}}$$
$$= \sqrt{\frac{1-(1-\sin^2\alpha_n)}{2(1+\sqrt{1-\sin^2\alpha_n})}} = \frac{\sin\alpha_n}{\sqrt{2\left(1+\sqrt{1-\sin^2\alpha_n}\right)}}$$

MATLAB Code: (mit maschinenunabhängigem Abbruchkriterium)

MATLAB-Code: Stabile Rekursion für Pi

```
s = sqrt(3)/2; Ao = 0; An=3*s; n = 6;   % initialization
z = [n A An-pi s];           % matrix for storing results
while (An < Ao) % terminate when machine accuracy is reached
  s = s/sqrt(2*(1+sqrt((1+s)*(1-s)))); % Stabilized recursion
  n=2*n; Ao = An; An = n/2*s;
  z = [z; n An An-pi s];          % store results
end
```


n	A_n	$A_n - \pi$	$\sin\alpha_n$
6	2.598076211353316	-0.543516442236477	0.866025403784439
12	3.000000000000000	-0.141592653589793	0.500000000000000
24	3.105828541230249	-0.035764112359544	0.258819045102521
48	3.132628613281238	-0.008964040308555	0.130526192220052
96	3.139350203046867	-0.002242450542926	0.065403129230143
192	3.141031950890509	-0.000560702699284	0.032719082821776
384	3.141452472285462	-0.000140181304332	0.016361731626487
768	3.141557607911857	-0.000035045677936	0.008181139603937
1536	3.141583892148318	-0.000008761441475	0.004090604026235
3072	3.141590463228050	-0.000002190361744	0.002045306291164
6144	3.141592105999271	-0.000000547590522	0.001022653680338
12288	3.141592516692156	-0.000000136897637	0.000511326907014
24576	3.141592619365383	-0.000000034224410	0.000255663461862
49152	3.141592645033690	-0.000000008556103	0.000127831731976
98304	3.141592651450766	-0.000000002139027	0.000063915866118
196608	3.141592653055036	-0.000000000534757	0.000031957933076
393216	3.141592653456104	-0.000000000133690	0.000015978966540
786432	3.141592653556371	-0.000000000033422	0.000007989483270
1572864	3.141592653581438	-0.000000000008355	0.000003994741635
3145728	3.141592653587705	-0.000000000002089	0.000001997370818
6291456	3.141592653589271	-0.000000000000522	0.000000998685409
12582912	3.141592653589663	-0.000000000000130	0.000000499342704
25165824	3.141592653589761	-0.000000000000032	0.000000249671352
50331648	3.141592653589786	-0.000000000000008	0.000000124835676
100663296	3.141592653589791	-0.000000000000002	0.000000062417838
201326592	3.141592653589794	0.000000000000000	0.000000031208919
402653184	3.141592653589794	0.000000000000001	0.000000015604460
805306368	3.141592653589794	0.000000000000001	0.000000007802230
1610612736	3.141592653589794	0.000000000000001	0.000000003901115

NUMDIFF

Beispiel 27 (Numerische Differentiation).

DIFFQUOT

Approximation der Ableitung durch Differenzenquotienten: $f'(x) \approx \dfrac{f(x+h)-f(x)}{h}$.

Je kleiner h desto besser, oder **?**

MATLAB-CODE: Numerische Diff. von exp(x)

```
h = 0.1; x = 0.0;
for i = 1:16
   df = (exp(x+h)-exp(x))/h;
   fprintf('%d %16.14f\n',i,df-1);
   h = h*0.1;
end
```

Offensichtliche Auslöschung

$\log_{10}(h)$	Relativer Fehler
-1	0.05170918075648
-2	0.00501670841679
-3	0.00050016670838
-4	0.00005000166714
-5	0.00000500000696
-6	0.00000049996218
-7	0.00000004943368
-8	-0.00000000607747
-9	0.00000008274037
-10	0.00000008274037
-11	0.00000008274037
-12	0.00000008890058234
-13	-0.00079927783736
-14	-0.00079927783736
-15	0.11022302462516
-16	-1.00000000000000

$$\left.\begin{array}{l} f'(x) - \dfrac{f(x+h)-f(x)}{h} \to 0 \\ \text{Rundungsfehlereinfluss} \to \infty \end{array}\right\} \quad \text{für } h \to 0 \,.$$

Analyse für $f(x) = \exp(x)$:

$$\texttt{df} = \frac{e^{x+h}\boxed{(1+\delta_1)} - e^x\boxed{(1+\delta_2)}}{h}$$

Korrekturfaktoren berücksichtigen Rundungsfehlereinfluss:
(→ „Axiom der Gleitpunktarithmetik", siehe 1.3)

sec:rundungsf

$$= e^x\left(\frac{e^h-1}{h} + \frac{\delta_1 e^h - \delta_2}{h}\right)$$

$|\delta_1|, |\delta_2| \le \texttt{eps}$, eps= Maschinengenauigkeit

$$\Rightarrow \quad |\texttt{df}| \le e^x\left(e^h + \texttt{eps}\,\frac{1+e^h}{h}\right)$$

$1 + O(h)$ $\quad O(h^{-1})$ für $h \to 0$

Relativer Fehler: $\left|\dfrac{e^x - \texttt{df}}{e^x}\right| \approx h + \dfrac{2\texttt{eps}}{h} \to \min$ für $h = \sqrt{2\,\texttt{eps}}$.

IEEE-Arithmetik doppelter Genauigkeit: $\sqrt{2\texttt{eps}} = 2.107342425544702 \cdot 10^{-8}$

1.8 Stabilität

[File: section-stabilitaet.tex, SVN: section-stabilitaet.tex 1073 2006-10-24 19:22:33Z hiptmair]

STABILITAET

Stabilität ist eine Eigenschaft eines Algorithmus für ein Problem

NUMALGO

Numerischer Algorithmus = Spezifizierte Folge elementarer Operationen (→ C(++)- oder FORTRAN-Programm)

Im folgenden: X, Y = normierte Vektorräume, z.B. $X = \mathbb{R}^n$, $Y = \mathbb{R}^m$

Definition 1.8.1 (Stabiler Algorithmus)**.** *Ein Algorithmus $\widetilde{F}$ zu einem Problem $F : X \mapsto Y$ heisst numerisch stabil (engl. stable), wenn sich für alle $x \in X$ sein (durch Rundungsfehlereinfluss gestörtes Ergebnis) $\widetilde{F}(x)$ als exaktes Ergebnis zu leicht gestörten Eingabedaten interpretieren lässt, d.h.*

NUMSTAB

$$\exists C \approx 1\!: \quad \forall x \in X\!: \quad \exists \widetilde{x} \in X\!: \quad \|x - \widetilde{x}\| \le C\,\texttt{eps}\,\|x\| \quad \wedge \quad \widetilde{F}(x) = F(\widetilde{x}) \,.$$

Je besser die Kondition eines Problems, desto kritischer die Stabilität von Algorithmen !

Bemerkung: Def. 1.8.1 definiert Stabilität im Sinne der Rückwärtsfehleranalyse (engl. *backward error analysis*)

def:stabalg

RUECKWANALYSE

Theorem 1.8.2 (Stabilität elementarer Operationen)**.**
Unter dem „Axiom der Gleitpunktarithmetik" sind die elementaren Operationen $\widetilde{+}, \widetilde{-}, \widetilde{\cdot}, \widetilde{/}$ numerisch stabil.

Beweis. (für $+$) $\quad x\widetilde{+}y = (x+y)(1+\delta) = \underbrace{x(1+\delta)}_{=:\widetilde{x}} + \underbrace{y(1+\delta)}_{=:\widetilde{y}}$. □

Beispiel 28. Stabilität von Algorithmus A aus Bsp. 24:

ex:cancel

$$\widetilde{f} = \underbrace{a^2(1+\delta_1)(1+\delta_3)}_{=:\widetilde{a}^2} - \underbrace{b^2(1+\delta_2)(1+\delta_3)}_{=:\widetilde{b}^2} \quad \text{mit} \quad \begin{array}{l}\widetilde{a} = a\sqrt{1+\delta_a} \\ \widetilde{b} = b\sqrt{1+\delta_b}\end{array} \,, \quad |\delta_a|, |\delta_b| < 3\texttt{eps} \,.$$

(Schlechter) Algorithmus ist trotzdem numerisch stabil ! Warum **?**

Betrachte (relative) Kondition von $(a,b) \mapsto f = a^2 - b^2$, z.B.

$$\frac{|\partial_a f(a,b)|}{|f(a,b)|} = \frac{2|a|^2}{|a^2-b^2|} \gg 1 \quad \text{falls} \quad a \approx b\,.$$

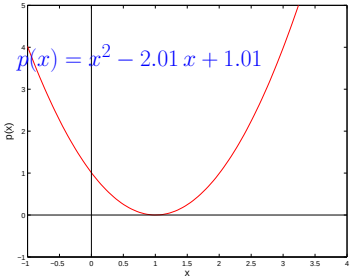
ALGA

Algorithmus A, (1.7.1): grosser Rundungsfehlereinfluss nur im Fall schlechter Kondition

NULLSTELLEN

Beispiel 29 (Reelle Nullstellen eines quadratischen Polynoms).

$p(x) = x^2 + px + q\,, \quad p,q \in \mathbb{R} \quad \Rightarrow$ Nullstellen $x_\pm = -\frac{p}{2} \pm \sqrt{(p/2)^2 - q}$, falls $p^2 > 4q$.



Fall: $p^2 \approx 4q$

HILFE! Auslöschung unter der Wurzel

Don't panic:

NULLSTELLENBEST

Nullstellenbestimmung schlecht konditioniert $(*)$

!

Formel auch mit Auslöschung stabil

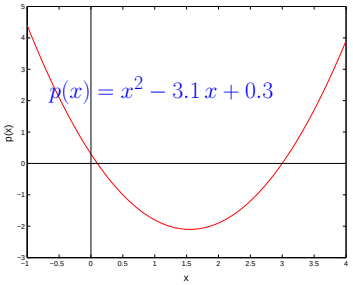
$(*)$ Absolute Konditionszahl bzgl. q: $\kappa_{\rm abs} = \dfrac{1}{\sqrt{p^2-4q}}$.

Fall: $p^2 \gg 4q$

[$x_\pm$ gut konditioniert ➢ Auslöschung gefährlich]

$$\left|\frac{p}{2}\right| \approx \sqrt{(p/2)^2 - q}$$

Auslöschung bei x_- , falls $p < 0$,
x_+ , falls $p > 0$.



Benutze $x_- x_+ = q$: numerisch stabile Nullstellenformel

NULLSTELLENFORM

$$\begin{cases} x_- = -\frac{p}{2} - \sqrt{(p/2)^2 - q}\,, & x_+ = \frac{q}{x_-} \quad \text{, falls } p > 0\,, \\ x_+ = -\frac{p}{2} + \sqrt{(p/2)^2 - q}\,, & x_- = \frac{q}{x_+} \quad \text{, falls } p < 0\,. \end{cases}$$

MATLAB-CODE: Nullstellen einer quadr. Fkt.

```
function [xm,xp] = quadzero(p,q)
D = 0.25*p^2-q;
if (D < 0), error('No zeros'); end;
if (p>0), xm = -0.5*p-sqrt(D); xp = q/xm;
else xp = -0.5*p+sqrt(D); xm = q/xp;
end
```

Stabile MATLAB-Implementierung

Resultate: Instabile Imlementierung ↓ für $p = -1, q = a(1-a)$

MATLAB-CODE: Nullstellen einer quadr. Fkt.

```
function [xm,xp] = qzfool(p,q)
D = 0.25*p^2-q;
if (D < 0), error('No zeros'); end;
xp = -0.5*p+sqrt(D);
xm = -0.5*p-sqrt(D);
```

Instabile MATLAB-Implementierung

a	Relativer Fehler in x_-
10^{-5}	0.0000000000009995
10^{-6}	0.0000000000267561
10^{-7}	0.0000000000287549
10^{-8}	0.0000000005263567
10^{-9}	0.0000000272292189
10^{-10}	0.0000000827403699
10^{-11}	0.0000000827403699
10^{-12}	0.0000333894311084
10^{-13}	0.0003109451872646
10^{-14}	0.0007992778373605
10^{-15}	0.0007992778373608
10^{-16}	0.1102230246251547
10^{-17}	1.0000000000000000

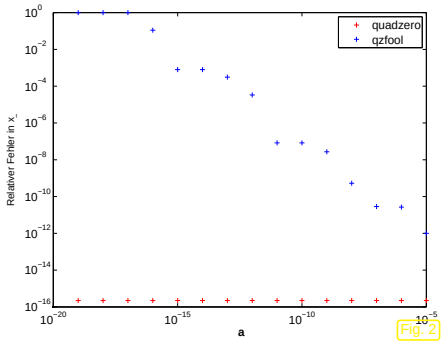


Fig. 2

◁ Relativer Fehler der Nullstellenbestimmung von

$$p(x) = x^2 - x + a(1-a) = (x-a)(x-1+a)$$

mittels `qzfool(-1.a*(1-a));` `quadzero(-1.a*(1-a));`

(Relative Fehler $<$ `eps` ersetzt durch `eps` !)

2 Nichtlineare Gleichungen

[File: chapter-nichtlineare-gleichungen.tex, SVN: chapter-nichtlineare-gleichungen.tex 1054 2006-10-18 11:43:04Z kalai]

Gegeben: Funktion $F : D \subset \mathbb{R}^n \mapsto \mathbb{R}^n, \quad n \in \mathbb{N}$

$\Updownarrow$

Kann bedeuten: Wir verfügen über eine Prozedur `function y=F(x)` zur Auswertung von F

Gesucht: Lösung der NICHTLINGLEICHUNG nichtlinearen Gleichung $F(\mathbf{x}) = 0$

Beachte: "Gleiche Anzahl von Gleichungen (engl. *equations*) und Unbekannten (engl. *unknowns*)"

Existenz und Eindeutigkeit von Lösungen ist für konkretes F jeweils zu klären !

2.1 Iterationsverfahren

[File: section-iterationsverfahren.tex, SVN: chapter-nichtlineare-gleichungen.tex 1054 2006-10-18 11:43:04Z kalai]

ITMETH
Iterationsverfahren zur (approximativen) Lösung der nichtlinearen Gleichung $F(\mathbf{x}) = 0$ = Algorithmus, der eine Folge $(\mathbf{x}^{(k)})_{k\in\mathbb{N}_0}$ von Näherungslösungen erzeugt.

ITS
- Iterierte $\mathbf{x}^{(k)}$ abhängig von F und (von einigen) $\mathbf{x}^{(n)}$, $n < k$, z.B.

$$\mathbf{x}^{(k)} = \underbrace{\mathbf{x}^{(k)}(F, \mathbf{x}^{(k-1)}, \ldots, \mathbf{x}^{(k-m)})}_{\text{ITRULE, MPIT}} \qquad (2.1.1)$$

eq:itmeth

Iterationsvorschrift für m-Punkt-Verfahren

INITGUESS
- $\mathbf{x}^{(0)}$ = Startwert (engl. *initial guess*)

Wir kennen bereits Iterationsverfahren: → Beispiele 2 & 13.

ex:extarea ex:extcarea

Definition 2.1.1 (Konvergenz von Iterationsverfahren).
CONVERGENT
Iterationsverfahren konvergent $:\Leftrightarrow \quad \mathbf{x}^{(k)} \to \mathbf{x}^*$ *und* $F(\mathbf{x}^*) = 0$.

Definition 2.1.2 (Konsistenz von Iterationsverfahren).
CONSISTENT
Iterationsverfahren konsistent mit $F(\mathbf{x}) = 0 \quad :\Leftrightarrow \quad \{\mathbf{x}^{(k)} \to \mathbf{x}^* \quad \Rightarrow \quad F(\mathbf{x}^*) = 0\}$

Terminologie: ITERR Fehler der Iterierten $\mathbf{x}^{(k)}$: $\mathbf{e}^{(k)} := \mathbf{x}^{(k)} - \mathbf{x}^*$

Definition 2.1.3 (Lokale und globale Konvergenz).
LOCCVG
Ein Iterationsverfahren konvergiert lokal (engl. converges locally*) gegen* $\mathbf{x}^* \in \mathbb{R}^n$*, falls eine Umgebung* $U \subset D$ *von* $\mathbf{x}^*$ *existiert, so dass*

$$\mathbf{x}^{(0)} \in U \quad \Rightarrow \quad \mathbf{x}^{(k)} \text{ wohldefiniert} \quad \wedge \quad \lim_{k\to\infty} \mathbf{x}^{(k)} = \mathbf{x}^*$$

für die erzeugten Iterationsfolgen $(\mathbf{x}^{(k)})_k$ GLOBCVG *gilt.*
Falls $U = D$*, so heisst die Iteration global konvergent.*

Ziel: Finde Iterationsverfahren, die (lokal) gegen (eine) Lösung von $F(\mathbf{x}) = 0$ konvergieren.

"*Konvergenzgeschwindigkeit*" **?**

Nötig: Norm $\|\cdot\|$ auf $\mathbb{R}^n$ (→ Grundvorlesungen)
(Allgemeiner: Metrik auf $\mathbb{R}^n$)
Fest gewählte Norm auf $\mathbb{R}^n$ wird im Folgenden vorausgesetzt.

Definition 2.1.4 (Norm). def:norm
X = $\mathbb{K}$*-Vektorraum,* $\mathbb{K} = \mathbb{C}, \mathbb{R}$*. Eine Abbildung* $\|\cdot\| : X \mapsto \mathbb{R}_0^+$ *ist eine* NORM *Norm auf* X*, falls*

NDEFINIT
(i) $\forall \mathbf{x} \in X: \quad \mathbf{x} \neq 0 \quad \Leftrightarrow \quad \|\mathbf{x}\| > 0$ *(Definitheit),*

NHOM
(ii) $\|\lambda\mathbf{x}\| = |\lambda|\|\mathbf{x}\| \quad \forall \mathbf{x} \in X, \lambda \in \mathbb{K}$ *(Homogenität),*

NTRIINEQ
(iii) $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\| \quad \forall \mathbf{x}, \mathbf{y} \in X$ *(Dreiecksungleichung).*

Beispiele: (für Vektor $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T \in \mathbb{K}^n$)

Bezeichnung	:	Definition	MATLAB-Funktion
EUKNORM Euklidische Norm	:	$\|\mathbf{x}\|_2 := \sqrt{\lvert x_1\rvert^2 + \cdots + \lvert x_n\rvert^2}$	`norm(x)`
1NORM 1-Norm	:	$\|\mathbf{x}\|_1 := \lvert x_1\rvert + \cdots + \lvert x_n\rvert$	`norm(x,1)`
MAXNORM Maximumnorm	:	$\|\mathbf{x}\|_\infty := \max\{\lvert x_1\rvert, \ldots, \lvert x_n\rvert\}$	`norm(x,inf)`

Wahl der Norm "nicht so wichtig" im Endlichdimensionalen:

Theorem 2.1.5 (Normäquivalenz im Endlichdimensionalen). thm:normeq
NORMEQIV
Beliebige zwei Normen $\|\cdot\|, \|\cdot\|_0$ auf $\mathbb{K}^n$ sind äquivalent

$$\exists C > 0: \quad C^{-1}\|\mathbf{x}\| \le \|\mathbf{x}\|_0 \le C\|x\| \quad \forall \mathbf{x} \in \mathbb{K}^n .$$

Definition 2.1.6 (Lineare Konvergenz). def:cvglin
LINCVG
Eine Folge $\mathbf{x}^{(k)}$, $k = 0, 1, 2, \ldots$, in $\mathbb{R}^n$ konvergiert linear gegen $x^* \in \mathbb{K}^n$, falls

$$\exists L < 1: \quad \left\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\right\| \le L \left\|\mathbf{x}^{(k)} - \mathbf{x}^*\right\| \quad \forall k \in \mathbb{N}_0 .$$

Terminologie: Kleinste obere Schranke für L → CVGRATE Konvergenzrate (engl. *rate of convergence*)

Erinnerung an zentralen Satz der Analysis:

Theorem 2.1.7 (Banachscher Fixpunktsatz).
Falls $D \subset \mathbb{K}^n$ abgeschlossen und für $\Phi : D \mapsto D$ gilt

$$\exists L < 1: : \quad \|\Phi(\mathbf{x}) - \Phi(\mathbf{y})\| \le L\|\mathbf{x} - \mathbf{y}\| \quad \forall \mathbf{x}, \mathbf{y} \in D ,$$

dann existiert genau ein Fixpunkt $\mathbf{x}^* \in D$ mit $\Phi(\mathbf{x}^*) = \mathbf{x}^*$, gegen den die Folge $\mathbf{x}^{(k+1)} := \Phi(x^{(k)})$ für beliebiges $\mathbf{x}^{(0)} \in D$ konvergiert.

Proof. Beweis verwendet 1-Punkt-Iteration $\mathbf{x}^{(k)} = \Phi(\mathbf{x}^{(k-1)})$, $\mathbf{x}^{(0)} \in D$:

$$\left\|\mathbf{x}^{(k+N)} - \mathbf{x}^{(k)}\right\| \le \sum_{j=k}^{k+N-1} \left\|\mathbf{x}^{(j+1)} - \mathbf{x}^{(j)}\right\| \le \sum_{j=k}^{k+N-1} L^j \left\|\mathbf{x}^{(1)} - \mathbf{x}^{(0)}\right\|$$

$$\le \frac{L^k}{1-L}\left\|\mathbf{x}^{(1)} - \mathbf{x}^{(0)}\right\| \xrightarrow{k\to\infty} 0 .$$

$(\mathbf{x}^{(k)})_{k\in\mathbb{N}_0}$ Cauchy-Folge ➤ konvergent $\mathbf{x}^{(k)} \xrightarrow{k\to\infty} \mathbf{x}^*$.
Stetigkeit von Φ ➤ $\Phi(\mathbf{x}^*) = \mathbf{x}^*$. Eindeutigkeit des Fixpunktes ist klar □

Beispiel 30 (Linear Konvergente Iteration).

Iteration ($n = 1$):

$$x^{(k+1)} = x^{(k)} + \frac{\cos x^{(k)} + 1}{\sin x^{(k)}} .$$

```
y  = [ ];
for i = 1:15
  x = x + (cos(x)+1)/sin(x);
  y = [y,x];
end
err = y - x;
rate = err(2:15)./err(1:14);
```


$x^{(15)}$ als Ersatz für x^* bei der Berechnung der Konvergenzrate:

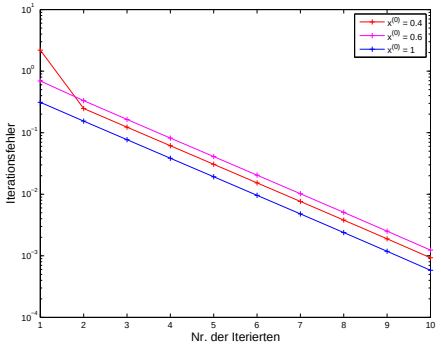
k	$x^{(0)} = 0.4$		$x^{(0)} = 0.6$		$x^{(0)} = 1$	
	$x^{(k)}$	$\frac{\lvert x^{(k)}-x^{(15)}\rvert}{\lvert x^{(k-1)}-x^{(15)}\rvert}$	$x^{(k)}$	$\frac{\lvert x^{(k)}-x^{(15)}\rvert}{\lvert x^{(k-1)}-x^{(15)}\rvert}$	$x^{(k)}$	$\frac{\lvert x^{(k)}-x^{(15)}\rvert}{\lvert x^{(k-1)}-x^{(15)}\rvert}$
2	3.3887	0.1128	3.4727	0.4791	2.9873	0.4959
3	3.2645	0.4974	3.3056	0.4953	3.0646	0.4989
4	3.2030	0.4992	3.2234	0.4988	3.1031	0.4996
5	3.1723	0.4996	3.1825	0.4995	3.1224	0.4997
6	3.1569	0.4995	3.1620	0.4994	3.1320	0.4995
7	3.1493	0.4990	3.1518	0.4990	3.1368	0.4990
8	3.1454	0.4980	3.1467	0.4980	3.1392	0.4980

➤ Konvergenzrate ≈ 0.5

Lineare Konvergenz → Def. 2.1.6 def:lincvg

⇕

Fehlerkurven = Geraden im lin-log Plot



◇

Abkürzung für Fehlernorm im k-ten Schritt $\epsilon_k := \left\|\mathbf{x}^{(k)} - \mathbf{x}^*\right\|$. Im Falle linearer Konvergenz (→ Def. 2.1.6) nimmt man an ($0 < L < 1$) def:lincvg

$$\epsilon_{k+1} \approx L\epsilon_k \quad \Rightarrow \quad \log \epsilon_{k+1} \approx \log L + \log \epsilon\epsilon_k \quad \Rightarrow \quad \log \epsilon_{k+1} \approx k \log L + \log \epsilon_0 .$$

➤ $\log L < 0$ gibt die Steigung der Geraden im lin-log-Plot.

Im Fall von Konvergenz von p-ter Ordnung ($p > 1$) (→ Def. 2.1.8): def:cvgord

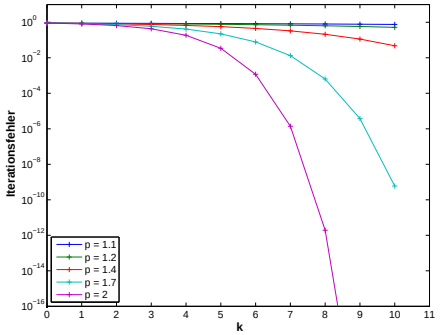
$$\epsilon_{k+1} \approx C\epsilon_k^p \quad \Rightarrow \quad \log \epsilon_{k+1} = \log C + p \log \epsilon_k \quad \Rightarrow \quad \log \epsilon_{k+1} = \log C \sum_{l=0}^{k} p^k + p^{k+1} \log \epsilon_0$$

$$\Rightarrow \quad \log \epsilon_{k+1} = \left(\frac{\log C}{p-1} + \log \epsilon_0 \right) p^{k+1} .$$

➤ Kurve ist negative Exponentialkurve im lin-log-Plot.

Definition 2.1.8 (Konvergenzordnung)**.**
Eine ***konvergente*** *Folge* $\mathbf{x}^{(k)}$, $k = 0, 1, 2, \ldots$, *in* $\mathbb{R}^n$ *konvergiert in* p*-ter Ordnung* *gegen* $\mathbf{x}^* \in \mathbb{K}^n$, *falls* $p > 1$ *und* PCVG

$$\exists C > 0: \quad \left\| \mathbf{x}^{(k+1)} - \mathbf{x}^* \right\| \leq C \left\| \mathbf{x}^{(k)} - \mathbf{x}^* \right\|^p \quad \forall k \in \mathbb{N}_0 .$$



◁ Qualitative Fehlerkurven für Konvergenz der Ordnung p
(lin-log-Plot)

Beispiel 31 (Quadratische Konvergenz einer Iteration).

Wurzeliteration für $a > 0$, → Bsp. 13, Monotonie & Beschränktheit ⇒ Konvergenz ex:termcrit

$$x^{(k+1)} = \frac{1}{2}(x^{(k)} + \frac{a}{x^{(k)}}) \quad \Rightarrow \quad |x^{(k+1)} - \sqrt{a}| = \frac{1}{2x^{(k)}} |x^{(k)} - \sqrt{a}|^2 .$$

AGM: $x^{(k)} > \sqrt{a}$ für $k \geq 1$ ► Konvergenzordnung 2 (Quadratische Konvergenz) QUADRKONV
(AGM ≙ Ungleichung zwischen arithmethischem und geometrischem Mittel: $\sqrt{ab} \leq \frac{1}{2}(a+b)$)

Kürze ab: $\epsilon_k := \left\| \mathbf{x}^{(k)} - \mathbf{x}^* \right\|$

$$\epsilon_{k+1} \approx C\epsilon_k^p \quad \Rightarrow \quad \log \epsilon_{k+1} \approx \log C + p \log \epsilon_k \quad \Rightarrow \quad \frac{\log \epsilon_{k+1} - \log \epsilon_k}{\log \epsilon_k - \log \epsilon_{k-1}} \approx p .$$

Iterierte für $a = 2$:

k	$x^{(k)}$	$e^{(k)} := x^{(k)} - \sqrt{2}$	$\log \frac{\lvert e^{(k)} \rvert}{\lvert e^{(k-1)} \rvert} : \log \frac{\lvert e^{(k-1)} \rvert}{\lvert e^{(k-2)} \rvert}$
0	2.00000000000000000	0.58578643762690485	
1	1.50000000000000000	0.08578643762690485	
2	1.41666666666666652	0.00245310429357137	1.850
3	1.41421568627450966	0.00000212390141452	1.984
4	1.41421356237468987	0.00000000000159472	2.000
5	1.41421356237309492	0.00000000000000022	0.630

Verdopplung der Anzahl der gültigen Stellen in jeder Iteration ! [Rundungsfehlereinfluss!]

Relativer Fehler: $x^{(k)} = x^*(1 + \delta_k)$ ➤ $x^{(k)} - x^* = \delta_k x^*$

$$|x^* \delta_{k+1}| = |x^{(k+1)} - x^*| \leq C|x^{(k)} - x^*|^2 = C|x^* \delta_k|^2$$

$$\blacktriangleright \quad |\delta_{k+1}| \leq C|x^*| \delta_k^2 .$$

◇

TERMKRIT
Abbruchkriterium (engl. *termination criterion*) **?**

Einfach: Abbruch konvergenter Iteration $\{\mathbf{x}^{(k)}\}_{k \in \mathbb{N}_0}$ falls

$\left\| F(\mathbf{x}^{(k)}) \right\| \leq \tau$ = vorgegebene TOL Toleranz > 0 .

= RESTERMCRIT Residuenbasiertes Abbruchkriterium

Schwieriger: Abbruch konvergenter Iteration $\{\mathbf{x}^{(k)}\}_{k\in\mathbb{N}_0}$ mit Grenzwert $\mathbf{x}^*$ falls

$$\left\|\mathbf{x}^{(k)}-\mathbf{x}^*\right\| \le \tau \quad = \text{vorgegebene Toleranz } > 0 \text{ .}$$

TOL

☹ $\mathbf{x}^*$ leider unbekannt !

Strategien:

- ☞ Maschinenunabhängiges Abbruchkriterium $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)}$ in $\mathbb{M}$ (→ overkill ?)
  APRIOTERMCRIT
- ☞ A priori Abbruchkriterium: fixiere Anzahl der Iterationen im Voraus (→ problemspezifisch !)
  APOSTTERMCRIT
- ☞ A posteriori Abbruchkriterium: STOP, falls $\left\|\mathbf{x}^{(k)}-\mathbf{x}^{(k-1)}\right\| \le \tau'$ (→ zuverlässig ?)
- ☞ Ersatzlösung: Residuenbasiertes Abbruchkriterium:
  STOP, falls $\left\|F(\mathbf{x}^{(k)})\right\| \le \tau''$ (→ zuverlässig ?)

A posteriori Abbruchkriterium für linear konvergente Iterationen mit Konvergenzrate $0 < L < 1$:

$$\left\|\mathbf{x}^{(k)}-\mathbf{x}^*\right\| \overset{\triangle\text{-Ungl.}}{\le} \left\|\mathbf{x}^{(k+1)}-\mathbf{x}^{(k)}\right\| + \left\|\mathbf{x}^{(k+1)}-\mathbf{x}^*\right\| \le \left\|\mathbf{x}^{(k+1)}-\mathbf{x}^{(k)}\right\| + L\left\|\mathbf{x}^{(k)}-\mathbf{x}^*\right\| \text{ .}$$

$$\blacktriangleright \quad \boxed{\left\|\mathbf{x}^{(k+1)}-\mathbf{x}^*\right\| \le \frac{L}{1-L}\left\|\mathbf{x}^{(k+1)}-\mathbf{x}^{(k)}\right\|} \qquad (2.1.2)$$ eq:apost

2.1 p. 65

Beispiel 32 (Fehlerschätzung für linear konvergente Iteration). → Bsp. 30

EX:CURZERO

$$x^{(k+1)} = x^{(k)} + \frac{\cos x^{(k)} + 1}{\sin x^{(k)}} \quad \Rightarrow x^{(k)} \to \pi \quad \text{für } x^{(0)} \text{ nahe bei } \pi \text{ .}$$

Beobachtete Konvergenzrate: $L = 1/2$

Fehler und geschätzter Fehler für $x^{(0)} = 0.4$:

k	$\lvert x^{(k)}-\pi\rvert$	$\frac{L}{1-L}\lvert x^{(k)}-x^{(k-1)}\rvert$	Schätzfehler
1	2.191562221997101	4.933154875586894	2.741592653589793
2	0.247139097781070	1.944423124216031	1.697284026434961
3	0.122936737876834	0.124202359904236	0.001265622027401
4	0.061390835206217	0.061545902670618	0.000155067464401
5	0.030685773472263	0.030705061733954	0.000019288261691
6	0.015341682696235	0.015344090776028	0.000002408079792
7	0.007670690889185	0.007670991807050	0.000000300917864
8	0.003835326638666	0.003835364250520	0.000000037611854
9	0.001917660968637	0.001917665670029	0.000000004701392
10	0.000958830190489	0.000958830778147	0.000000000587658
11	0.000479415058549	0.000479415131941	0.000000000073392
12	0.000239707524646	0.000239707533903	0.000000000009257
13	0.000119853761949	0.000119853762696	0.000000000000747
14	0.000059926881308	0.000059926880641	0.000000000000667
15	0.000029963440745	0.000029963440563	0.000000000000181

➢ A posteriori Schätzung sehr genau !

◇

Bemerkung. L nicht genau bekannt ➢ $\widetilde{L} > L$ für Fehlerschätzung garantiert Genauigkeit

2.2 p. 67

2.2 Fixpunktiteration

[File: section-fixpunktiteration.tex, SVN: section-fixpunktiteration.tex 1103 2006-11-07 05:33:57Z hiptmair]

FIXPIT

Eine Fixpunktiteration ist definiert durch eine FITFN Iterationsfunktion $\Phi : \mathrm{dom}(\Phi) \subset \mathbb{R}^n \mapsto \mathbb{R}^n$.

Iterationsfunktion $\Phi : \mathrm{dom}(\Phi) \subset \mathbb{R}^n \mapsto \mathbb{R}^n$
Startwert $\mathbf{x}^{(0)} \in \mathrm{dom}(\Phi)$

$\Rightarrow$ Iterationsfolge $(\mathbf{x}^{(k)})_{k\in\mathbb{N}_0}$: $\mathbf{x}^{(k+1)} := \Phi(\mathbf{x}^{(k)})$.

Iterationsfolge nicht unbedingt wohldefiniert: $\mathbf{x}^{(k)} \notin \mathrm{dom}(\Phi)$ möglich !

Definition 2.2.1 (Spezialisierung von Def. 2.1.2). *Eine Fixpunktiteration* $\mathbf{x}^{(k+1)} = \Phi(\mathbf{x}^{(k)})$ *ist konsistent (engl.* consistent*) mit* $F(\mathbf{x}) = 0$*, falls*

def:consit

KONSISTENTFIXP

$$F(\mathbf{x}) = 0 \quad \wedge \quad x \in \mathrm{dom}(\Phi) \cap D \quad \Leftrightarrow \quad \Phi(\mathbf{x}) = \mathbf{x} \text{ .}$$

2.1 p. 66

Beachte: Φ stetig & Fixpunktiteration (lokal) konvergent gegen Nullstelle $\mathbf{x}^*$ ➢ $\mathbf{x}^*$ ist FIXPKT Fixpunkt (engl. *fixed point*) der Iterationsfunktion.

2.2 p. 68

Konstruktion von Fixpunktiteration für $F(\mathbf{x}) = 0$:

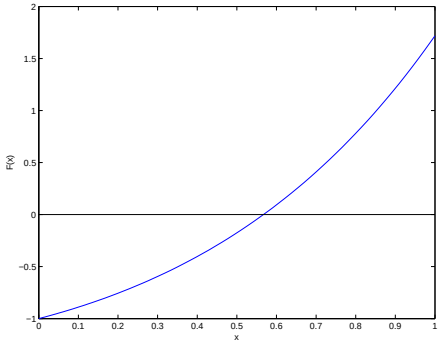
$$F(\mathbf{x}) = 0 \quad \Leftrightarrow \quad \Phi(\mathbf{x}) = \mathbf{x} \quad \blacktriangleright \quad \text{Fixpunktiteration} \quad \mathbf{x}^{(k+1)} := \Phi(\mathbf{x}^{(k)}) \,. \qquad (2.2.1)$$

fpit

☛ Transformation in Fixpunktform (nicht eindeutig):

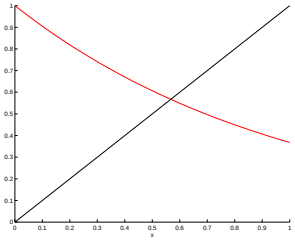
Beispiel 33 (Fixpunktiteration).

$$F(x) = xe^x - 1 \,, \quad x \in [0,1] \,.$$

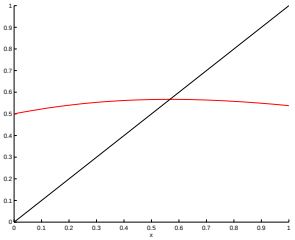


Verschiedene Fixpunktformen:

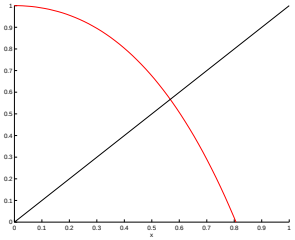
$$\begin{aligned} \Phi_1(x) &= e^{-x} \,, \\ \Phi_2(x) &= \frac{1+x}{1+e^x} \,, \\ \Phi_3(x) &= x + 1 - xe^x \,. \end{aligned}$$



Funktion Φ_1



Funktion Φ_2



Funktion Φ_3

k	$x^{(k+1)} := \Phi_1(x^{(k)})$	$x^{(k+1)} := \Phi_2(x^{(k)})$	$x^{(k+1)} := \Phi_3(x^{(k)})$
0	0.500000000000000	0.500000000000000	0.500000000000000
1	0.606530659712633	0.566311003197218	0.675639364649936
2	0.545239211892605	0.567143165034862	0.347812678511202
3	0.579703094878068	0.567143290409781	0.855321409174107
4	0.560064627938902	0.567143290409784	-0.156505955383169
5	0.571172148977215	0.567143290409784	0.977326422747719
6	0.564862946980323	0.567143290409784	-0.619764251895580
7	0.568438047570066	0.567143290409784	0.713713087416146
8	0.566409452746921	0.567143290409784	0.256626649129847
9	0.567559634262242	0.567143290409784	0.924920676910549
10	0.566907212935471	0.567143290409784	-0.407422405542253

k	$\lvert x_1^{(k+1)} - x^* \rvert$	$\lvert x_2^{(k+1)} - x^* \rvert$	$\lvert x_3^{(k+1)} - x^* \rvert$
0	0.067143290409784	0.067143290409784	0.067143290409784
1	0.039387369302849	0.000832287212566	0.108496074240152
2	0.021904078517179	0.000000125374922	0.219330611898582
3	0.012559804468284	0.000000000000003	0.288178118764323
4	0.007078662470882	0.000000000000000	0.723649245792953
5	0.004028858567431	0.000000000000000	0.410183132337935
6	0.002280343429460	0.000000000000000	1.186907542305364
7	0.001294757160282	0.000000000000000	0.146569797006362
8	0.000733837662863	0.000000000000000	0.310516641279937
9	0.000416343852458	0.000000000000000	0.357777386500765
10	0.000236077474313	0.000000000000000	0.974565695952037

Beobachtung: Lineare Konvergenz von $x_1^{(k)}$, quadratische Konvergenz von $x_2^{(k)}$,
Keine Konvergenz (chaotisches Verhalten) von $x_3^{(k)}$), $x_i^{(0)} = 0.5$. ◇

KONVANALYSEFIXP

Konvergenzanalyse, Fall $n = 1$ ($\Phi : \mathrm{dom}(\Phi) \subset \mathbb{R} \mapsto \mathbb{R}$)

TAYLORFORMEL

Theorem 2.2.2 (Taylorformel). *Falls $\Phi : \mathrm{dom}(\Phi) \subset \mathbb{R} \mapsto \mathbb{R}$ $(m+1)$-mal stetig differenzierbar, $x \in \mathrm{dom}(\Phi)$*

$$\Phi(y) - \Phi(x) = \sum_{k=1}^{m} \frac{1}{k!} \Phi^{(k)}(x)(y-x)^k + O(|y-x|^{m+1}) \quad \forall y \in \mathcal{U}(x) \,.$$

Anwendung auf Fixpunktiteration mit Iterationsfunktion Φ:

Falls $\Phi(x^*) = x^*$, $\Phi : \mathrm{dom}(\Phi) \subset \mathbb{R} \mapsto \mathbb{R}$ "hinreichend glatt"

$$x^{(k+1)} - x^* = \Phi(x^{(k)}) - \Phi(x^*) = \sum_{l=1}^{m} \frac{1}{l!} \Phi^{(l)}(x^*)(x^{(k)} - x^*)^l + O(|x^{(k)} - x^*|^{m+1}) \,. \qquad (2.2.2)$$

eq:FPT

Lemma 2.2.3. *Falls $\Phi : \mathrm{dom}(\Phi) \subset \mathbb{R} \mapsto \mathbb{R}$ $(m+1)$-mal stetig differenzierbar, $\Phi(x^*) = x^*$ für ein x^* im Innern von $\mathrm{dom}(\Phi)$, und $\Phi^{(l)}(x^*) = 0$ für $l = 1, \ldots, m$, $m \geq 1$, dann konvergiert die Fixpunktiteration (2.2.1) lokal mit Ordnung $m+1$ (→ Def. 2.1.8) gegen x^*.*

fpit def:cvgord

Beweis. Für Umgebung $\mathcal{U}$ von x^*

$$\overset{\text{(2.2.2)}}{\Rightarrow} \quad \exists C > 0: \quad |\Phi(y) - \Phi(x^*)| \leq C\,|y - x^*|^{m+1} \quad \forall y \in \mathcal{U}\ .$$

$$\delta^m C < 1/2: \quad |x^{(0)} - x^*| < \delta \quad \Rightarrow \quad |x^{(k)} - x^*| < 2^{-k}\delta \quad \succ \text{ lokale Konvergenz .} \quad \square$$

Fortsetzung Bsp. 33:

$$\Phi_2'(x) = \frac{1 - xe^x}{(1+e^x)^2} = 0 \quad \text{wenn} \quad xe^x - 1 = 0 \quad \blacktriangleright \text{ Quadratische Konvergenz .}$$

Lemma 2.2.4 (Taylorformel im Mehrdimensionalen). *Falls $\Phi : \mathrm{dom}(\Phi) \subset \mathbb{R}^n \mapsto \mathbb{R}^n$ 2-fach stetig differenzierbar*

$$\Phi(\mathbf{y}) - \Phi(\mathbf{x}) = D\Phi(\mathbf{x})(\mathbf{y} - \mathbf{x}) + O(\|\mathbf{y} - \mathbf{x}\|^2) \quad \forall \mathbf{y} \in \mathcal{U}(\mathbf{x})\ .$$

Terminologie: $D\Phi(\mathbf{x}) \in \mathbb{R}^{n,n}$ = Jacobimatrix von Φ an der Stelle $\mathbf{x}$

Lemma 2.2.5. *Falls $\Phi : \mathrm{dom}(\Phi) \subset \mathbb{R}^n \mapsto \mathbb{R}^n$ 2-fach stetig differenzierbar, $\Phi(\mathbf{x}^*) = \mathbf{x}^*$, $\|D\Phi(\mathbf{x}^*)\| < 1$, dann konvergiert die Fixpunktiteration (2.2.1) lokal linear.*

Matrixnorm !

Definition 2.2.6. *Der Norm $\|\cdot\|$ auf $\mathbb{R}^n$ zugeordnete Matrixnorm*

$$\mathbf{M} \in \mathbb{R}^{n,n}: \quad \|\mathbf{M}\| := \sup_{x \in \mathbb{R}^n \setminus \{0\}} \frac{\|\mathbf{M}x\|}{\|x\|}\ .$$

Notation: $\|\mathbf{x}\|_2 \to \|\mathbf{M}\|_2\,, \quad \|\mathbf{x}\|_1 \to \|\mathbf{M}\|_1\,, \quad \|\mathbf{x}\|_\infty \to \|\mathbf{M}\|_\infty$

Beispiel 34 (Matrixnorm zur Maximumnorm und 1-Norm).

z.B. für $\mathbf{M} \in \mathbb{K}^{2,2}$:

$$\begin{aligned} \|\mathbf{Mx}\|_\infty &= \max\{|m_{11}x_1 + m_{12}x_2|, |m_{21}x_1 + m_{22}x_2|\} \\ &\leq \max\{|m_{11}| + |m_{12}|, |m_{21}| + |m_{22}|\}\ \|x\|_\infty\ , \\ \|\mathbf{Mx}\|_1 &= |m_{11}x_1 + m_{12}x_2| + |m_{21}x_1 + m_{22}x_2| \\ &\leq \max\{|m_{11}| + |m_{21}|, |m_{12}| + |m_{22}|\}(|x_1| + |x_2|)\ . \end{aligned}$$

➢ Matrixnorm zu $\|\cdot\|_\infty$ = ROWSUMNORM Zeilensummennorm $\|\mathbf{M}\|_\infty := \max_{i=1,\ldots,n} \sum_{j=1}^{n} |m_{ij}|$, (2.2.3)

➢ Matrixnorm zu $\|\cdot\|_1$ = COLSUMNORM Spaltensummennorm $\|\mathbf{M}\|_1 := \max_{j=1,\ldots,m} \sum_{i=1}^{n} |m_{ij}|$. (2.2.4)

$\diamond$

Beweis von Lemma 2.2.5. Für Umgebung $\mathcal{U}(\mathbf{x}^*)$:

$$\exists C > 0: \quad \|\Phi(\mathbf{y}) - \Phi(\mathbf{x}) - D\Phi(\mathbf{x})(\mathbf{y} - \mathbf{x})\| \leq C\,\|\mathbf{y} - \mathbf{x}\|^2 \quad \forall \mathbf{y} \in \mathcal{U}(\mathbf{x}^*)\ .$$

Wähle $\delta > 0$ so dass $L := \delta C + \|D\Phi(\mathbf{x}^*)\| < 1$, $\{\mathbf{y} \in \mathbb{R}^n\colon \|\mathbf{y} - \mathbf{x}^*\| < \delta\} \subset \mathcal{U}(\mathbf{x}^*)$

$$\blacktriangleright \quad \|\mathbf{x} - \mathbf{x}^*\| < \delta \quad \Rightarrow \quad \|\Phi(\mathbf{x}) - \mathbf{x}^*\| \leq \underbrace{(\|D\Phi(\mathbf{x}^*)\| + C\,\|\mathbf{x} - \mathbf{x}^*\|)}_{\leq L}\ \|\mathbf{x} - \mathbf{x}^*\|\ . \quad \square$$

Bemerkung 35.

Falls $\|D\Phi(\mathbf{x}^*)\| < 1$, $\mathbf{x}^{(k)} \approx \mathbf{x}^*$ ➢ ASYMPKONVRATE Asymptotische Konvergenzrate $L = \|D\Phi(x^*)\|$

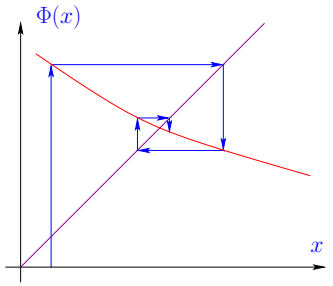
$\triangle$

Fortsetzung Bsp. 33: Da $x^*e^{x^*} - 1 = 0$

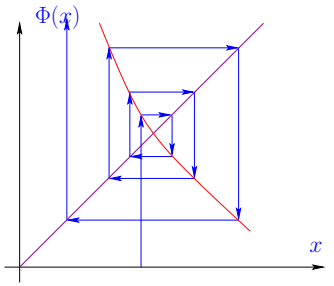
$$\Phi_1'(x) = -e^{-x} \quad \Rightarrow \quad \Phi_1'(x^*) = -x^* \approx -0.56 \quad \blacktriangleright \text{ Lokal lineare Konvergenz .}$$

$$\Phi_3'(x) = 1 - xe^x - e^x \quad \Rightarrow \quad \Phi_3'(x^*) = -\frac{1}{x^*} \approx -1.79 \quad \blacktriangleright \text{ Keine Konvergenz .}$$

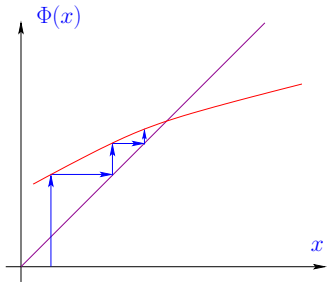
Definition 2.2.7. *Für stetig differenzierbares $\Phi : \mathrm{dom}(\Phi) \subset \mathbb{R} \mapsto \mathbb{R}$ heisst ein Fixpunkt $x^* \in \mathrm{dom}(\Phi)$ FIXPANZ anziehend (engl.* attractive*), falls $|\Phi'(x^*)| < 1$ und FIXPABST abstossend, falls $|\Phi'(x^*)| > 1$*



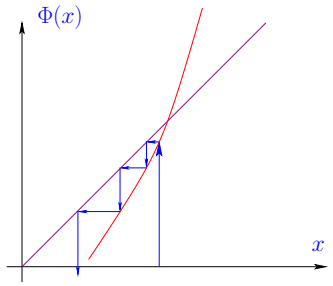
$-1 < \Phi'(x^*) \leq 0$ ➢ Konvergenz



$\Phi'(x^*) < -1$ ➢ Divergenz



$0 \le \Phi'(x^*) < 1$ ➢ Konvergenz



$1 < \Phi'(x^*)$ ➢ Divergenz

Beispiel 36 (Mehrdimensionale Fixpunktiteration).

Gleichungssystem	➢	Fixpunktform
$x_1 - c(\cos x_1 - \sin x_2) = 0$	➢	$c(\cos x_1 - \sin x_2) = x_1$
$(x_1 - x_2) + c \sin x_2 = 0$	➢	$c(\cos x_1 - 2\sin x_2) = x_2$

$$\Phi\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = c\begin{pmatrix} \cos x_1 - \sin x_2 \\ \cos x_1 - 2\sin x_2 \end{pmatrix} \Rightarrow D\Phi\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = -c\begin{pmatrix} \sin x_1 & \cos x_2 \\ \sin x_1 & 2\cos x_2 \end{pmatrix}$$

Wahl einer geeigneten Vektornorm: $\|\cdot\|$ = Maximumnorm $\|\cdot\|_\infty$ (→ Bsp. 65) ex:inftymatrixnorm

Wenn $c < \frac{1}{3} \Rightarrow \|D\Phi(\mathbf{x})\|_\infty < 1 \quad \forall \mathbf{x} \in \mathbb{R}^2$ ➢ Lineare Konvergenz der Fixpunktiteration

◇

Abbruchkriterien für kontrahierende Iterationen mit Kontraktionsrate $0 < L < 1$:

$$\left\|\mathbf{x}^{(k+m)} - \mathbf{x}^{(k)}\right\| \overset{\triangle\text{-Ungl.}}{\le} \sum_{j=k}^{k+m-1} \left\|\mathbf{x}^{(j+1)} - \mathbf{x}^{(j)}\right\| \le \sum_{j=k}^{k+m-1} L^{j-k} \left\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\right\|$$
$$= \frac{1-L^m}{1-L} \left\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\right\| \le \frac{1-L^m}{1-L} L^{k-l} \left\|\mathbf{x}^{(l+1)} - \mathbf{x}^{(l)}\right\| .$$

➢ für $m \to \infty$, mit $\mathbf{x}^* := \lim_{k\to\infty} \mathbf{x}^{(k)}$:

$$\left\|\mathbf{x}^* - \mathbf{x}^{(k)}\right\| \le \frac{L^{k-l}}{1-L} \left\|\mathbf{x}^{(l+1)} - \mathbf{x}^{(l)}\right\| . \quad (2.2.5)$$ eq:lincvg

Setze $l = 1$ in (2.2.5) eq:lincvg

A priori Abbruchkriterium

$$\left\|\mathbf{x}^* - \mathbf{x}^{(k)}\right\| \le \frac{L^k}{1-L} \left\|\mathbf{x}^{(1)} - \mathbf{x}^{(0)}\right\| \quad (2.2.6)$$ eq:fitapri

Setze $l = k$ in (2.2.5) eq:lincvg

A posteriori Abbruchkriterium

$$\left\|\mathbf{x}^* - \mathbf{x}^{(k)}\right\| \le \frac{L}{1-L} \left\|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\right\| \quad (2.2.7)$$ eq:fitapost

2.3 Nullstellenbestimmung von Funktionen

[File: section-nullstellenbestimmung-von-funktionen.tex, SVN: section-nullstellenbestimmung-von-funktionen.tex 1183 2006-11-29 10:46:03Z hiptmair]

Spezialfall $n = 1$: $F : I \subset \mathbb{R} \mapsto \mathbb{R}$ **stetig**, I Intervall

Gesucht: $x^* \in I$: $F(x^*) = 0$

2.3.1 Bisektionsverfahren

Idee: Nutze Anordnung der reellen Zahlen aus (Intervallschachtelung) & Zwischenwertsatz

Input: $a, b \in I$ mit $F(a)F(b) < 0$ (unterschiedliche Vorzeichen)
➢ $\exists x^* \in]\min\{a,b\}, \max\{a,b\}[: \quad F(x^*) = 0$

MATLAB-Implementierung:

MATLAB-CODE: Bisektionsverfahren

```
function x = bisect(F,a,b,tol)
% Searching zero by bisection
if (a>b), t=b; a=b; b=t; end;
fa = F(a); fb = F(b);
if (fa*fb>0)
  error('f(a), f(b) same sign'); end;
fa > 0), v=-1; else v = 1; end
x = 0.5*(b+a);
while((b-a > tol) & ((a<x) & (x<b)) )
  if (v*F(x)>0), b=x; else a=x; end;
  x = 0.5*(a+b)
end
```

Handle auf MATLAB-Funktion, die F implementiert.

Vermeide Endlosschleife, falls `tol` < Abstand von Maschinenzahlen bei Nullstelle x^*.

- „narrensicher"
- Nur Auswertungen von F erforderlich

Nur lineare Konvergenz: $|x^{(k)} - x^*| \le 2^{-k}|b-a|$ ➢ $\log_2\left(\frac{|b-a|}{\texttt{tol}}\right)$ Schritte

Bemerkung 37. Bisektion implementiert in MATLAB-Funktion `fzero`.

△

2.3.2 Modellfunktionsverfahren

MODELLFKTVERF
≙ Iterationsverfahren zur Nullstellenbestimmung von F:

Idee: Gegeben approximative Nullstellen $x^{(k)}, x^{(k-1)}, \ldots, x^{(k-m)}$

❶ ersetze F durch MODELFNCT Modellfunktion $\widetilde{F}$
(verwende Funktionswerte/Ableitungen in $x^{(k)}, x^{(k-1)}, \ldots, x^{(k-m)}$)

❷ $x^{(k+1)} :=$ Nullstelle von $\widetilde{F}$
(muss "einfach" ↔ analytisch zu bestimmen sein)

Unterscheidung:

EINPKTVERFAHREN
Einpunktverfahren : $x^{(k+1)} = x^{(k+1)}(F, x^{(k)}), k \in \mathbb{N}$ (z.B. Fixpunktiteration → Abschn. 2.2) sec:fixpunkt

MEHRPKTVERF
Mehrpunktverfahren : $x^{(k+1)} = x^{(k+1)}(x^{(k)}, x^{(k-1)}, \ldots, x^{(k-m)}), \quad k \in \mathbb{N}, m = 2, 3, \ldots$.

2.3.2.1 Newton-Verfahren in 1D

NEWTONONED

Annahme: $F : I \mapsto \mathbb{R}$ stetig differenzierbar.

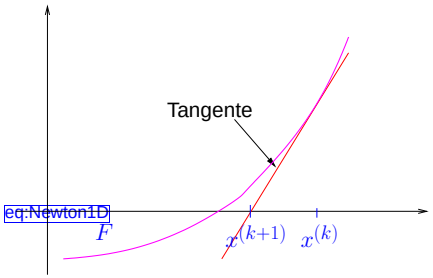
Modellfunktion = Tangente an F in $x^{(k)}$:

$$\widetilde{F}(x) := F(x^{(k)}) + F'(x^{(k)})(x - x^{(k)})$$

➤ $x^{(k+1)}$ als Nullstelle der Tangente

NEWTONIT
Newton-Iteration

$$x^{(k+1)} := x^{(k)} - \frac{F(x^{(k)})}{F'(x^{(k)})} , \tag{2.3.1}$$ eq:Newton1D



erforderlich $F'(x^{(k)}) \neq 0$.

Beispiel 38 (Newton-Verfahren in 1D). (→ Beispiel 33) ex:fixp

$$F(x) = xe^x - 1 \Rightarrow F'(x) = e^x(1+x) \Rightarrow x^{(k+1)} = x^{(k)} - \frac{x^{(k)}e^{x^{(k)}} - 1}{e^{x^{(k)}}(1+x^{(k)})} = \frac{(x^{(k)})^2 - e^{-x^{(k)}}}{1+x^{(k)}}$$

$$F(x) = x - e^{-x} \Rightarrow F'(x) = 1 + e^{-x} \Rightarrow x^{(k+1)} = x^{(k)} - \frac{x^{(k)} - e^{-x^{(k)}}}{1+e^{-x^{(k)}}} = \frac{1+x^{(k)}}{1+e^{x^{(k)}}}$$

Beispiel 33 ➢ quadratische Konvergenz! (→ Def. 2.1.8) def:cvgord

◇

Newton-Iteration (2.5.1) eq:Newton ≙ Fixpunktiteration (→ Abschnitt 2.2) mit Iterationsfunktion sec:fixpunktiteration

$$\Phi(x) = x - \frac{F(x)}{F'(x)} \Rightarrow \Phi'(x) = \frac{F(x)F''(x)}{(F'(x))^2} \Rightarrow \Phi'(x^*) = 0 , \quad \text{wenn } F(x^*) = 0,\ F'(x^*) \neq 0 .$$

Lemma 2.2.3 lem:fpcvg ➢ Vermutung (bestätigt durch Theorie, [12, Thm. 4.10], [10, Sect. 2.1]): DEH02 DEU04

LOKQUADKONV
Newton-Verfahren konvergiert *lokal* quadratisch (→ Def. 2.1.8) def:cvgord, falls $F'(x^*) \neq 0$

2.3.2.2 Spezielle Einpunktverfahren

Konstruktionsidee (für weitere Einpunktverfahren): Nichtlineare lokale Approximation

Voraussetzung: Glattheit von F: $F \in C^m(I)$ für ein $m > 1$

Beispiel 39 (Halleysche Iteration).

Gegeben $x^{(k)} \in I$, Modellfunktion $h(x^{(k+1)}) = 0$ wobei

$$h(x) = \frac{a}{x+b} + c \quad \text{(Rationale Funktion)} , \quad F^{(j)}(x^{(k)}) = h^{(j)}(x^{(k)}) , \quad j = 0, 1, 2 .$$

$$\frac{a}{x^{(k)}+b} + c = F(x^{(k)}) , \ -\frac{a}{(x^{(k)}+b)^2} = F'(x^{(k)}) , \ \frac{2a}{(x^{(k)}+b)^3} = F''(x^{(k)}) .$$

$$x^{(k+1)} = x^{(k)} - \frac{F(x^{(k)})}{F'(x^{(k)})} \cdot \frac{1}{1 - \frac{1}{2}\frac{F(x^{(k)})F''(x^{(k)})}{F'(x^{(k)})^2}} .$$

Halleysche Iteration für $F(x) = \frac{1}{(x+1)^2} + \frac{1}{(x+0.1)^2} - 1 , \quad x > 0 :$ und $x^{(0)} = 0$

k	$x^{(k)}$	$F(x^{(k)})$	$x^{(k)} - x^{(k-1)}$	$x^{(k)} - x^*$
1	0.19865959351191	10.90706835180178	-0.19865959351191	-0.84754290138257
2	0.69096314049024	0.94813655914799	-0.49230354697833	-0.35523935440424
3	1.02335017694603	0.03670912956750	-0.33238703645579	-0.02285231794846
4	1.04604398836483	0.00024757037430	-0.02269381141880	-0.00015850652965
5	1.04620248685303	0.00000001255745	-0.00015849848821	-0.00000000804145

Vergleich: Newton-Verfahren (2.3.1) eq:Newton1D für das gleiche Problem

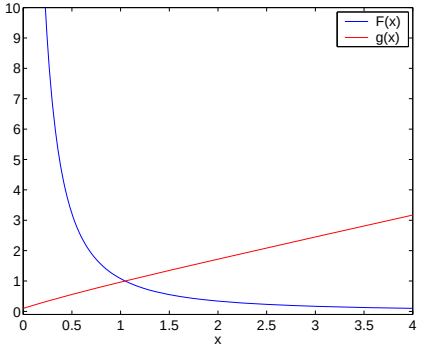
k	$x^{(k)}$	$F(x^{(k)})$	$x^{(k)} - x^{(k-1)}$	$x^{(k)} - x^*$
1	0.04995004995005	44.38117504792020	-0.04995004995005	-0.99625244494443
2	0.12455117953073	19.62288236082625	-0.07460112958068	-0.92165131536375
3	0.23476467495811	8.57909346342925	-0.11021349542738	-0.81143781993637
4	0.39254785728080	3.63763326452917	-0.15778318232269	-0.65365463761368
5	0.60067545233191	1.42717892023773	-0.20812759505112	-0.44552704256257
6	0.82714994286833	0.46286007749125	-0.22647449053641	-0.21905255202615
7	0.99028203077844	0.09369191826377	-0.16313208791011	-0.05592046411604
8	1.04242438221432	0.00592723560279	-0.05214235143588	-0.00377811268016
9	1.04618505691071	0.00002723158211	-0.00376067469639	-0.00001743798377
10	1.04620249452271	0.00000000058056	-0.00001743761199	-0.00000000037178

► Überlegenheit der Halley-Iteration, da F rationale Funktion.

! Newton-Verfahren langsamer, aber auch billiger (→ Abschn. 2.4) sec:effizienz ◇

Beispiel 40 (Massgeschneidertes Newton-Verfahren).

Wie in Bsp. 39: ex:Halley $F(x) = \frac{1}{(x+1)^2} + \frac{1}{(x+0.1)^2} - 1\,, \quad x > 0$:



Beobachtung:

$$F(x) + 1 \approx 2x^{-2} \text{ für } x \gg 1$$

➢ $g(x) := \frac{1}{\sqrt{F(x)+1}}$ fast linear für $x \gg 1$

► Anstelle von $F(x) \stackrel{!}{=} 0$ löse $g(x) \stackrel{!}{=} 1$ mit Newton-Verfahren (2.3.1). eq:Newton1D

$$x^{(k+1)} = x^{(k)} - \frac{g(x^{(k)}) - 1}{g'(x^{(k)})} = x^{(k)} + \left(\frac{1}{\sqrt{F(x^{(k)}) + 1}} - 1 \right) \frac{2(F(x^{(k)}) + 1)^{3/2}}{F'(x^{(k)})}$$

$$= x^{(k)} + \frac{2(F(x^{(k)}) + 1)(1 - \sqrt{F(x^{(k)}) + 1})}{F'(x^{(k)})}\,.$$

Konvergenzhistorie für $x^{(0)} = 0$:

k	$x^{(k)}$	$F(x^{(k)})$	$x^{(k)} - x^{(k-1)}$	$x^{(k)} - x^*$
1	0.91312431341979	0.24747993091128	0.91312431341979	-0.13307818147469
2	1.04517022155323	0.00161402574513	0.13204590813344	-0.00103227334125
3	1.04620244004116	0.00000008565847	0.00103221848793	-0.00000005485332
4	1.04620249489448	0.00000000000000	0.00000005485332	-0.00000000000000

◇

Weitere Idee: Konsistente Modifikation der Newton-Iterationsvorschrift:

► Fixpunktiteration : $\Phi(x) = x - \frac{F(x)}{F'(x)} H(x)$ mit „geeignetem" $H : I \mapsto \mathbb{R}$.

Ziel: Bestimme H so, dass Verfahren p-ter Ordnung entsteht, Werkzeug: Lemma 2.2.3. lem:fpcvg

Annahme: F „genügend" glatt, $\exists x^* \in I$: $F(x^*) = 0$, $F'(x^*) \neq 0$.

$$\Phi = x - uH \quad , \quad \Phi' = 1 - u'H - uH' \quad , \quad \Phi'' = -u''H - 2u'H - uH'' \,,$$

$$\text{mit } u = \frac{F}{F'} \quad \Rightarrow \quad u' = 1 - \frac{FF''}{(F')^2} \quad , \quad u'' = -\frac{F''}{F'} + 2\frac{F(F'')^2}{(F')^3} - \frac{FF'''}{(F')^2}\,.$$

$F(x^*) = 0 \qquad u(x^*) = 0 \;\; u'(x^*) = 1 \;\; u''(x^*) = -\frac{F''(x^*)}{F'(x^*)}$

► $\Phi'(x^*) = 1 - H(x^*) \quad , \quad \Phi''(x^*) = \frac{F''(x^*)}{F'(x^*)} H(x^*) - 2H'(x^*)\,.$ (2.3.2) MN:1

lem:fpcvg 2.2.3 p

$p = 2$ (quadratische Konvergenz): $H(x^*) = 1\,,$

KUBKONV $p = 3$ (kubische Konvergenz): $H(x^*) = 1 \quad \wedge \quad H'(x^*) = \frac{1}{2}\frac{F''(x^*)}{F'(x^*)}\,.$

$H(x) = G(1 - u'(x)) \qquad G$

► Fixpunktiteration $x^{(k+1)} = x^{(k)} - \frac{F(x^{(k)})}{F'(x^{(k)})} G\left(\frac{F(x^{(k)})F''(x^{(k)})}{(F'(x^{(k)}))^2} \right)\,.$ (2.3.3) eq:modN

ewt **Lemma 2.3.1.** *Falls $F \in C^2(I)$, $F(x^*) = 0$, $F'(x^*) \neq 0$, $G \in C^2(U)$ in einer Umgebung U von 0, $G(0) = 1$, $G'(0) = \frac{1}{2}$, dann konvergiert die Fixpunktiteration (2.3.3) lokal kubisch gegen x^*.* eq:modNewt

Beweis: Lemma 2.2.3, (2.3.2) und

$$H(x^*) = G(0) \quad , \quad H'(x^*) = -G'(0)u''(x^*) = G'(0)\frac{F''(x^*)}{F'(x^*)} \, .$$

Beispiel 41 (Beispiele für modifizierte Newton-Verfahren).

- $G(t) = \dfrac{1}{1-\frac{1}{2}t}$ ➡ Halleysche Methode (→ Beispiel 39)
- $G(t) = \dfrac{2}{1+\sqrt{1-2t}}$ ➡ Eulersche Methode
- $G(t) = 1 + \frac{1}{2}t$ ➡ Quadratische inverse Interpolation

Numerisches Experiment:

$$F(x) = xe^x - 1 \, , \quad x^{(0)} = 5$$

k	$e^{(k)} := x^{(k)} - x^*$		
	Halley	Euler	Quad. Inv.
1	2.81548211105635	3.57571385244736	2.03843730027891
2	1.37597082614957	2.76924150041340	1.02137913293045
3	0.34002908011728	1.95675490333756	0.28835890388161
4	0.00951600547085	1.25252187565405	0.01497518178983
5	0.00000024995484	0.51609312477451	0.00000315361454
6		0.14709716035310	
7		0.00109463314926	
8		0.00000000107549	

◇

2.3.2.3 Mehrpunktverfahren

Idee: ➡ Ersetze F durch Interpolationspolynom

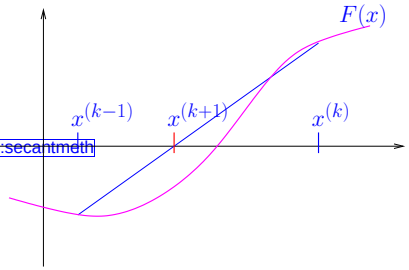
Interpolationsverfahren

Einfachster Vertreter: Sekantenverfahren

$x^{(k+1)}$ Nullstelle der Sekante

$$s(x) = F(x^{(k)}) + \frac{F(x^{(k)}) - F(x^{(k-1)})}{x^{(k)} - x^{(k-1)}}(x - x^{(k)}) \, ,$$

(2.3.4)



➡ $$x^{(k+1)} = x^{(k)} - \frac{F(x^{(k)})(x^{(k)} - x^{(k-1)})}{F(x^{(k)}) - F(x^{(k-1)})} \, .$$

(2.3.5)

Sekantenverfahren
(MATLAB-Implementierung)

- Nur eine Funktionsauswertung von F pro Schritt !
- Ableitungsfrei !

```
function x = secant(x0,x1,F,tol)
fo = feval(F,x0);
for i=1:MAXIT
    fn = feval(F,x1);
    s = fn*(x1-x0)/(fn-fo);
    x0 = x1; x1 = x1-s;
    if (abs(s) < tol), x = x1; return; end
    fo = fn;
end
```

MATLAB-Code Sekantenverfahren

Beispiel 42 (Sekantenverfahren). $F(x) = xe^x - 1 \, , \quad x^{(0)} = 0 \, , x^{(1)} = 5 \, .$

k	$x^{(k)}$	$F(x^{(k)})$	$e^{(k)} := x^{(k)} - x^*$	$\frac{\log\lvert e^{(k+1)}\rvert - \log\lvert e^{(k)}\rvert}{\log\lvert e^{(k)}\rvert - \log\lvert e^{(k-1)}\rvert}$
2	0.00673794699909	-0.99321649977589	-0.56040534341070	
3	0.01342122983571	-0.98639742654892	-0.55372206057408	24.43308649757745
4	0.98017620833821	1.61209684919288	0.41303291792843	2.70802321457994
5	0.38040476787948	-0.44351476841567	-0.18673852253030	1.48753625853887
6	0.50981028847430	-0.15117846201565	-0.05733300193548	1.51452723840131
7	0.57673091089295	0.02670169957932	0.00958762048317	1.70075240166256
8	0.56668541543431	-0.00126473620459	-0.00045787497547	1.59458505614449
9	0.56713970649585	-0.00000990312376	-0.00000358391394	1.62641838319117
10	0.56714329175406	0.00000000371452	0.00000000134427	
11	0.56714329040978	-0.00000000000001	-0.00000000000000	

◇

Konvergenzanalyse: Fehler $e^{(k)} := x^{(k)} - x^*$

$$e^{(k+1)} = \Phi(x^* + e^{(k)}, x^* + e^{(k-1)}) - x^* \quad \text{mit} \quad \Phi(x,y) := x - \frac{F(x)(x-y)}{F(x) - F(y)} \, . \qquad (2.3.6)$$

Taylorentwicklung mit MAPLE:

```
> Phi := (x,y) -> x-F(x)*(x-y)/(F(x)-F(y));
> F(s) := 0;
> e2 = normal(mtaylor(Phi(s+e1,s+e0)-s,[e0,e1],4));
```

➡ (Linearisierte) Fehlerrekursion: $e^{(k+1)} \doteq \frac{1}{2}\frac{F''(x^*)}{F'(x^*)}e^{(k)}e^{(k-1)} = Ce^{(k)}e^{(k-1)} \, .$

Heuristik zur Bestimmung der Konvergenzordnung: $e^{(k+1)} = K^2(e^{(k-1)})^{p^2}$, $e^{(k)} = K(e^{(k-1)})^p$ (vgl. Def. 2.1.8)

➡ $$(e^{(k-1)})^{p^2-p-1} = K^{-1}C \quad \Rightarrow \quad p^2 - p - 1 = 0 \quad \Rightarrow \quad p = \tfrac{1}{2}(1 \pm \sqrt{5}) \, .$$

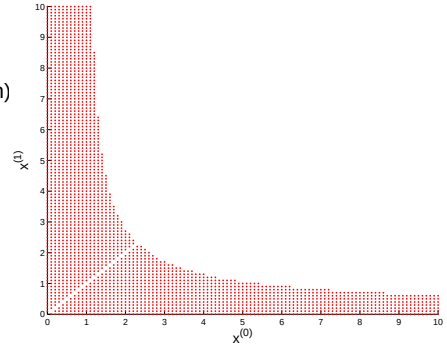
Da $e^{(k)} \to 0$ für $k \to \infty$ ➢ Konvergenzrate $p = \frac{1}{2}(1+\sqrt{5}) \approx 1.62$ (siehe Bsp. 42 !)

Beispiel 43 (Lokale Konvergenz, Sekantenverfahren)

$$F(x) = \arctan(x)$$

$\cdot$: Konvergenz des Sekantenverfahrens für ein Paar $(x^{(0)}, x^{(1)})$ von Startwerten. ▷

= lokale Konvergenz → Def. 2.1.3 def:cvgit



Variante von Mehrpunktverfahren: INVINTERP *Inverse Interpolation*

Annahme: $F : I \subset \mathbb{R} \mapsto \mathbb{R}$ umkehrbar

$$F(x^*) = 0 \quad \Rightarrow \quad F^{-1}(0) = x^* .$$

- Interpoliere F^{-1} durch Polynom p vom Grad d festgelegt durch

$$p(F(x^{(k-m)})) = x^{(k-m)} , \quad m = 0, \ldots, d .$$

- Neue Näherung $x^{(k+1)} := p(0)$

Fall $m = 1$ ➢ Sekantenverfahren

Fall $m = 2$: Quadratische inverse Interpolation

MAPLE-Code:
```
p := x-> a*x^2+b*x+c;
solve({p(f0)=x0,p(f1)=x1,p(f2)=x2},{a,b,c});
assign(%); p(0);
```

$$x^{(k+1)} = \frac{F_0^2(F_1x_2 - F_2x_1) + F_1^2(F_2x_0 - F_0x_2) + F_2^2(F_0x_1 - F_1x_0)}{F_0^2(F_1 - F_2) + F_1^2(F_2 - F_0) + F_2^2(F_0 - F_1)} .$$

$(F_0 := F(x^{(k-2)}), F_1 := F(x^{(k-1)}), F_2 := F(x^{(k)}), x_0 := x^{(k-2)}, x_1 := x^{(k-1)}, x_2 := x^{(k)})$

Beispiel 44 (Quadratische inverse Interpolation). $F(x) = xe^x - 1$, $x^{(0)} = 0$, $x^{(1)} = 2.5$,$x^{(2)} = 5$.

k	$x^{(k)}$	$F(x^{(k)})$	$e^{(k)} := x^{(k)} - x^*$	$\frac{\log\lvert e^{(k+1)}\rvert - \log\lvert e^{(k)}\rvert}{\log\lvert e^{(k)}\rvert - \log\lvert e^{(k-1)}\rvert}$
3	0.08520390058175	-0.90721814294134	-0.48193938982803	
4	0.16009252622586	-0.81211229637354	-0.40705076418392	3.33791154378839
5	0.79879381816390	0.77560534067946	0.23165052775411	2.28740488912208
6	0.63094636752843	0.18579323999999	0.06380307711864	1.82494667289715
7	0.56107750991028	-0.01667806436181	-0.00606578049951	1.87323264214217
8	0.56706941033107	-0.00020413476766	-0.00007388007872	1.79832936980454
9	0.56714331707092	0.00000007367067	0.00000002666114	1.84841261527097
10	0.56714329040980	0.00000000000003	0.00000000000001	

◇

2.4 Effizienz

[File: section-effizienz.tex, SVN: section-effizienz.tex 1054 2006-10-18 11:43:04Z kalai]

EFFIZIENZ	RECHAUFW
Effizienz eines Iterationsverfahrens (zur Lösung von $F(x) = 0$)	↔ Rechenaufwand um eine vorgegebene Anzahl gültiger Stellen zu erreichen.

Abstrakt: $W \mathrel{\hat{=}}$ Rechenaufwand pro Schritt

$$\text{(etwa } W \approx \frac{\sharp\{\text{Auswertungen von } F\}}{\text{Schritt}} + n \cdot \frac{\sharp\{\text{Auswertungen von } F'\}}{\text{Schritt}} + \cdots \text{)}$$

Quantitative Analyse für Iterationsverfahren der Konvergenzordnung p, $p \geq 1$ (→ Def. 2.1.8), d.h., def:cvgord

$$\exists C > 0: \quad \left\| e^{(k)} \right\| \leq C \left\| e^{(k-1)} \right\|^p \quad \forall k \geq 1 .$$

Anzahl Schritte k für $\left\|e^{(k)}\right\| \leq \rho \left\|e^{(0)}\right\|$, $\rho > 0$ vorgegeben **?**: (Annahme $C \left\|e^{(0)}\right\|^{p-1} < 1$)

$$p = 1: \quad \left\|e^{(k)}\right\| \leq C^k \left\|e^{(0)}\right\| \quad \Rightarrow \quad k \geq \frac{\log \rho}{\log C}, \tag{2.4.1}$$

eq:ISC

$$p > 1: \quad \left\|e^{(k)}\right\| \leq C^{\frac{p^k-1}{p-1}} \left\|e^{(0)}\right\|^{p^k} \quad \Rightarrow \quad p^k \geq 1 + \frac{\log \rho}{\log C/p-1 + \log(\left\|e^{(0)}\right\|)}$$

$$\Rightarrow \quad k \geq \log(1 + \frac{\log \rho}{\log L_0})/\log p\,, \quad L_0 := C^{1/p-1}\left\|e^{(0)}\right\| \,. \tag{2.4.2}$$

eq:IS

Beachte: $|\log \rho| \leftrightarrow$ Anzahl gültiger Stellen in $x^{(k)}$

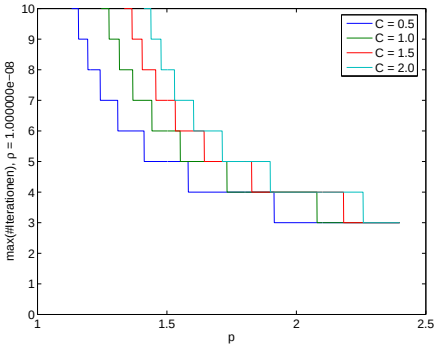
EFFICIENCY

Effizienzmass (**asymptotisch** für $\rho \to 0$ ➜ $|\log \rho| \to \infty$):

$$\text{Effizienz} = \begin{cases} -\dfrac{\log C}{W} & \text{, falls } p = 1\,, \\ \dfrac{\log p |\log \rho|}{W \log |\log \rho|} & \text{, falls } p > 1\,. \end{cases} \tag{2.4.3}$$

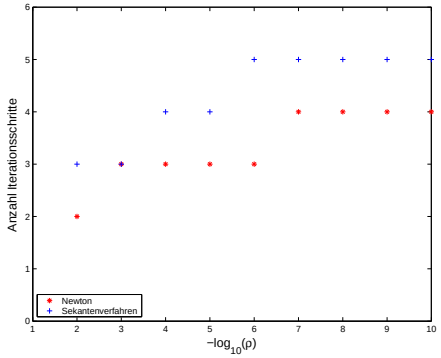
eq:efficie

Beispiel 45 (Effizienz von Iterationsverfahren).



Auswertung (2.4.2) für $\left\|e^{(0)}\right\| = 0.1$, $\rho = 10^{-8}$

eq:IS



Newton $\leftrightarrow$ Sekantenverfahren, $C = 1$, Anfangsfehler $\left\|e^{(0)}\right\| = 0.1$

$$W_{\text{Newton}} = 2W_{\text{Sekant}}\,, \quad p_{\text{Newton}} = 2,\ p_{\text{Sekant}} = 1.62 \quad \succ \quad \frac{\log p_{\text{Newton}}}{W_{\text{Newton}}} : \frac{\log p_{\text{Sekant}}}{W_{\text{Sekant}}} = 0.71\,.$$

◄ Sekantenverfahren effizienter als Newton-Verfahren

2.5 Newton-Verfahren für Gleichungssysteme

[File: section-newton-verfahren.tex, SVN: section-newton-verfahren.tex 1234 2006-12-19 12:04:51Z hiptmair]

Nichtlineares Gleichungssystem: zu $F : D \subset \mathbb{R}^n \mapsto \mathbb{R}^n$ finde $\mathbf{x}^* \in D$: $F(\mathbf{x}^*) = 0$

Annahme: $F : D \subset \mathbb{R}^n \mapsto \mathbb{R}^n$ stetig differenzierbar

2.5.1 Die Newton-Iteration

Idee, siehe Abschn. 2.3.2.1: sec:newton-verfahren-1d Lokale Linearisierung:

Gegeben $x^{(k)} \in D$ bestimme $x^{(k+1)}$ als Nullstelle der affin linearen Modellfunktion

$$F(\mathbf{x}) \approx \widetilde{F}(\mathbf{x}) := F(\mathbf{x}^{(k)}) + DF(\mathbf{x}^{(k)})(\mathbf{x} - \mathbf{x}^{(k)})\,,$$

$DF(\mathbf{x}) \in \mathbb{R}^{n,n}$ = Jacobi-Matrix, $DF(\mathbf{x}) = \left(\dfrac{\partial F_j}{\partial x_k}(\mathbf{x})\right)_{j,k=1}^{n}$.

◄ Newton-Iteration: (→ (2.3.1) für 1D-Fall) eq:Newton1D

$$\boxed{\mathbf{x}^{(k+1)} := \mathbf{x}^{(k)} - DF(\mathbf{x}^{(k)})^{-1} F(\mathbf{x}^{(k)})} \quad , \quad [\text{ falls } DF(\mathbf{x}^{(k)}) \text{ regulär }] \tag{2.5.1}$$

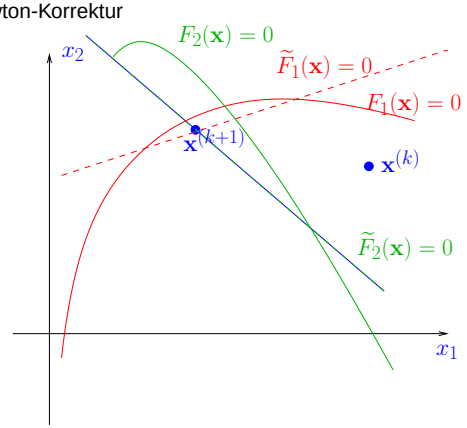
eq:Newto

NEWTONKORR

Terminologie: $-DF(\mathbf{x}^{(k)})^{-1} F(\mathbf{x}^{(k)})$ = Newton-Korrektur

Veranschaulichung in 2D: ▷

$\mathbf{x}^{(k+1)}$ als Schnitt von Geraden (= Nullstellen mengen der Komponenten der Modellfunktion)



Beispiel 46 (Newton-Verfahren in 2D, [32]). MEL03

$$F(\mathbf{x}) = \begin{pmatrix} x_1^2 - x_2^4 \\ x_1 - x_2^3 \end{pmatrix}\,, \quad \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \in \mathbb{R}^2 \quad \succ \quad F\begin{pmatrix} 1 \\ 1 \end{pmatrix} = 0\,.$$

$$\blacktriangleright \quad DF(\mathbf{x}) = \begin{pmatrix} \partial_{x_1} F_1(x) & \partial_{x_2} F_1(x) \\ \partial_{x_1} F_2(x) & \partial_{x_2} F_2(x) \end{pmatrix} = \begin{pmatrix} 2x_1 & -4x_2^3 \\ 1 & -3x_2^2 \end{pmatrix}$$

◄ Realisierung der Newton-Iteration (2.5.1): eq:Newton

1. Mit $\mathbf{x}^{(k)} = (x_1, x_2)^T$ löse das Gleichungssystem $\begin{pmatrix} 2x_1 & -4x_2^3 \\ 1 & -3x_2^2 \end{pmatrix} s = -\begin{pmatrix} x_1^2 - x_2^4 \\ x_1 - x_2^3 \end{pmatrix}$.

2. Setze $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + s$

k	$\mathbf{x}^{(k)}$	$\|\mathbf{x}^* - \mathbf{x}^{(k)}\|_2$
0	$(0.7,\ 0.7)^T$	4.24e-01
1	$(0.87850000000000,\ 1.064285714285714)^T$	1.37e-01
2	$(1.01815943274188,\ 1.00914882463936)^T$	2.03e-02
3	$(1.00023355916300,\ 1.00015913936075)^T$	2.83e-04
4	$(1.00000000583852,\ 1.00000002726552)^T$	2.79e-08
5	$(0.999999999999998,\ 1.000000000000000)^T$	2.11e-15
6	$(1,\ 1)^T$	

◇

Neuer Aspekt für $n \gg 1$ (im Vergleich zu $n = 1$,Abschn. sec:newton-verfahren-1d

Berechnung der Newton-Korrektur (evtl.) aufwändig, *cf.* eq:GEcost

MATLAB-Shell für Newton-Verfahren:

Löse lineares Gleichungssystem:

A\b = $\mathbf{A}^{-1}b$ → Abschn. sec:numer-losung-line

F, DF: Funktionshandles

MATLAB-CODE: Newtonverfahren

```
function x = newton(x,F,DF,tol)
% MATLAB template for Newton method
for i=1:MAXIT
  s = DF(x) \ F(x);
  x = x-s;
  if (norm(s) < tol), return; end;
end
```

Bemerkung 47. Wichtige Eigenschaft des Newton-Verfahrens: AFFININV Affin-Invarianz DEU04 [10, Sect .1.2.2]

$$G(\mathbf{x}) := \mathbf{A}F(\mathbf{x}) \quad \text{mit } \mathbf{A} \in \mathbb{R}^{n,n} \text{ regulär} \quad \succ \quad F(\mathbf{x}^*) = 0 \quad \Rightarrow \quad G(\mathbf{x}^*) = 0 \,.$$

Die Newton-Iteration für G ist unabhängig von $\mathbf{A}$!

SIMPLNEWT

Bemerkung 48. Vereinfachtes Newton-Verfahren: Verwende $DF(\mathbf{x}^{(k)})$ für mehrere Schritte (→ nur lokal lineare Konvergenz)

Bemerkung 49. Falls $DF(\mathbf{x})$ nicht verfügbar (z.B. wenn $F(\mathbf{x})$ nur als Prozedur gegeben):

Numerisches Differenzieren: $\dfrac{\partial F_i}{\partial x_j}(\mathbf{x}) \approx \dfrac{F_i(\mathbf{x} + h\vec{e}_j) - F_i(\mathbf{x})}{h}$.

Achtung: Auslöschung (→ Beispiel ex:numdiff 27) $\succ$ $h \approx \sqrt{\text{EPS}}$.

Übung 2.6. Given a matrix $\mathbf{A}$ implement a numerical method for computing one eigenvalue/eigenvector of $\mathbf{A}$ by applying Newton's method to

$$F\begin{pmatrix} \mathbf{x} \\ \lambda \end{pmatrix} := \begin{pmatrix} \mathbf{A}\mathbf{x} - \lambda\mathbf{x} \\ \|\mathbf{x}\|^2 - 1 \end{pmatrix} .$$

2.6.1 Konvergenzanalyse des Newton-Verfahrens

Newton-Iteration (eq:Newton 2.5.1) $=$ Fixpunktiteration (→ Abschnitt sec:fixpunktiteration 2.2) mit

$$\Phi(\mathbf{x}) = \mathbf{x} - DF(\mathbf{x})^{-1}F(\mathbf{x}) \,.$$

$$\Big[\text{„Produktregel“ : } \quad D\Phi(\mathbf{x}) = \mathbf{I} - D(DF(\mathbf{x})^{-1})F(\mathbf{x}) - DF(\mathbf{x})^{-1}DF(\mathbf{x}) \quad \Big]$$

$$F(\mathbf{x}^*) = 0 \quad \Rightarrow \quad D\phi(\mathbf{x}^*) = 0 \,.$$

Lemma lem:fpcvg 2.2.3 $\succ$ Vermutung (bestätigt durch Theorie, DEH02 [12, Thm. 4.10], DEU04 [10, Sect. 2.1]):

Newton-Verfahren konvergiert lokal quadratisch, falls $DF(\mathbf{x}^*)$ regulär

Beispiel für theoretische Aussage über Newton-Verfahren:

Theorem 2.6.1 (Thm. 4.10 in DEH02 [12]). *Sei $D \subset \mathbb{R}^n$ offen und konvex und $F : D \mapsto \mathbb{R}^n$ stetig differenzierbar mit invertierbarer Jacobi-Matrix $DF(x)$ $\forall x \in D$.*

$$\exists \omega \geq 0\colon \quad \left\| DF(x)^{-1}(DF(x + \sigma v) - DF(x))v \right\| \leq \sigma\omega \|v\|^2 \quad \forall \sigma \in [0,1], x \in D, v \in \mathbb{R}^n \,,$$

$$\exists x^*, x^{(0)} \in D\colon \quad F(x^*) = 0 \quad \wedge \quad \rho := \left\| x^* - x^{(0)} \right\| < \frac{2}{\omega} \quad \wedge \quad B_\rho(x^*) \subset D \,.$$

Dann erfüllt die Newton-Folge aus eq:Newton (2.5.1):

1. $x^{(k)} \in B_\rho(x^*) := \{y \in \mathbb{R}^n,\ \|y - x^*\| < \rho\}$ *für alle* $k \in \mathbb{N}$,
2. $\lim_{k\to\infty} x^{(k)} = x^*$,
3. $\left\| x^{(k+1)} - x^* \right\| \leq \frac{\omega}{2} \left\| x^{(k)} - x^* \right\|^2$.

Problem: Normalerweise sind weder ω noch x^* bekannt !

A priori Abschätzungen (wie in Thm. thm:newton 2.6.1) von geringem praktischen Nutzen

ABBRUCHKRITNEWTON
Abbruchkriterien für Newton-Verfahren:

Asymptotisch wegen quadratischer Konvergenz:

$$\left\|\mathbf{x}^{(k+1)}-\mathbf{x}^*\right\| \ll \left\|\mathbf{x}^{(k)}-\mathbf{x}^*\right\| \quad \Rightarrow \quad \left\|\mathbf{x}^{(k)}-\mathbf{x}^*\right\| \approx \left\|\mathbf{x}^{(k+1)}-\mathbf{x}^{(k)}\right\| .$$

Beende Iteration, wenn $\left\|\mathbf{x}^{(k+1)}-\mathbf{x}^{(k)}\right\| = \left\|DF(\mathbf{x}^{(k)})^{-1}F(\mathbf{x}^{(k)})\right\| < \tau \left\|\mathbf{x}^{(k)}\right\|$, τ = Toleranz

→ Überflüssiger (letzter) Schritt !

Beachte: für $n \gg 1$ kann jeder Newtonschritt mit grossem Rechenaufwand verbunden sein.

Alternativ: $DF(\mathbf{x}^{(k-1)}) \approx DF(\mathbf{x}^{(k)}) \succ$ Beende Iteration, wenn $\left\|DF(\mathbf{x}^{(k-1)})^{-1}F(\mathbf{x}^{(k)})\right\| < \tau$

Terminologie: $DF(\mathbf{x}^{(k-1)})^{-1}F(\mathbf{x}^{(k)})$ heisst vereinfachte Newton-Korrektur
(Billig verfügbar, falls LU-Zerlegung (→ Abschn. 3.2.3) von $DF(\mathbf{x}^{(k-1)})$ bereits berechnet)

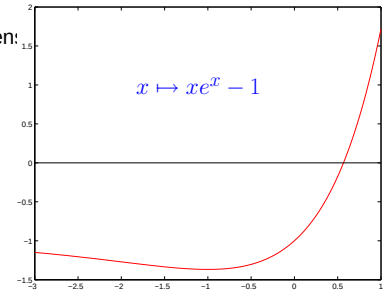
VEREINFNEWKORR
sec:die-lr-zerlegung

Beispiel 50 (Lokale Konvergenz des Newton-Verfahrens
Was heisst lokale Konvergenz **?** Beispiel:

$$F(x) = xe^x - 1 \quad \Rightarrow \quad F'(-1) = 0$$

$$x^{(0)} < -1 \succ x^{(k)} \to -\infty$$
$$x^{(0)} > -1 \succ x^{(k)} \to x^*$$

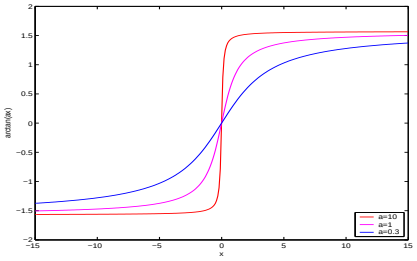
◇

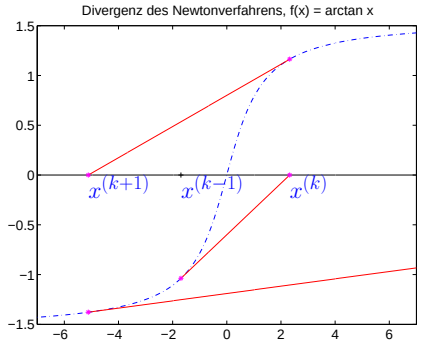


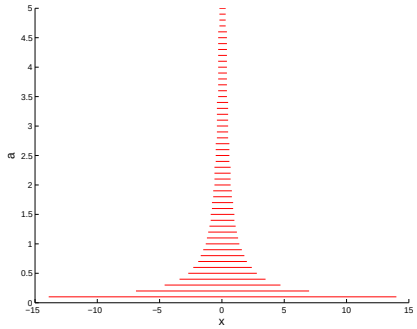
Beispiel 51 (Konvergenzbereich des Newton-Verfahrens).

$$F(x) = \arctan(ax) , \quad a > 0, x \in \mathbb{R}$$

$$\succ \quad x^* = 0 .$$







roter Bereich = $\{x^{(0)} \in \mathbb{R}, x^{(k)} \to 0\}$

Zusammenfassung: Das Newton-Verfahren

konvergiert *asymptotisch* sehr schnell: Verdopplung Zahl gültiger Stellen in jedem Schritt

hat u.U. nur sehr kleinen (lokalen) Konvergenzbereich

2.6.2 Gedämpftes Newton-Verfahren

ex:arctanNewt
Beispiel 51: Problem ist „Überschiessen" der Newton-Korrektur

GED
DAEMPFUNG
Idee: Dämpfung der Korrektur:

Mit $\lambda^{(k)} > 0$: $\mathbf{x}^{(k+1)} := \mathbf{x}^{(k)} - \lambda^{(k)} DF(\mathbf{x}^{(k)})^{-1}F(\mathbf{x}^{(k)})$. (2.6.1) eq:dam

Terminologie: $\lambda^{(k)}$ = Dämpfungsfaktor

MONOTIEST
DEU04
Wahl des Dämpfungsfaktors: Affininvarianter natürlicher Monotonietest [10, Ch. 3]

„grösstmögliches" $\lambda^{(k)} > 0$: $\left\|\overline{\mathbf{s}}(\lambda^{(k)})\right\| \leq (1 - \frac{\lambda^{(k)}}{2}) \|\mathbf{s}\|$ (2.6.2) eq:mnt

mit $\mathbf{s} := DF(\mathbf{x}^{(k)})^{-1}F(\mathbf{x}^{(k)})$ → aktuelle Newton-Korrektur ,
$\overline{\mathbf{s}}(\lambda^{(k)}) := DF(\mathbf{x}^{(k)})^{-1}F(\mathbf{x}^{(k)} + \lambda^{(k)}\mathbf{s})$ → versuchsweise vereinfachte Newton-Korrektur .

Heuristik: Konvergenx ⇔ Abnahme der „Länge" der Newton-Korrektur

MATLAB-CODE : Gedämpftes Newton-Verfahren

```
function [x,cvg] = dampnewton(x,F,DF,q,tol)
A = DF(x); s = A\F(x);
xn = x-s;
lambda = 1; cvg = 0;
while (norm(s) > tol)
```

```
    f = F(xn); st = A\f;
    while (norm(st) > (1-lambda/2)*norm(s))
      lambda = q*lambda;
      if (lambda < LMIN), cvg = -1; return; end
      xn = x-lambda*s;
      f = F(xn); st = A\f;
    end
  x = xn;
  A = DF(x); s = A\f;
  lambda = min(lambda/q,1);
  xn = x-lambda*s;
end
x = xn;
```

Strategie: Reduziere Dämpfungsparameter um Faktor $q \in]0,1[$ (üblich $q = \frac{1}{2}$) solange bis affininvarianter natürlicher Monotonietest (2.6.2) erfüllt

Beispiel 52 (Gedämpftes Newton-Verfahren). (→ Beispiel 51)

$F(x) = \arctan(x)$,

- $x^{(0)} = 20$
- $q = \frac{1}{2}$
- `LMIN = 0.001`

Beobachtung: asymptotische quadratische Konvergenz

k	$\lambda^{(k)}$	$x^{(k)}$	$F(x^{(k)})$
1	0.03125	0.94199967624205	0.75554074974604
2	0.06250	0.85287592931991	0.70616132170387
3	0.12500	0.70039827977515	0.61099321623952
4	0.25000	0.47271811131169	0.44158487422833
5	0.50000	0.20258686348037	0.19988168667351
6	1.00000	-0.00549825489514	-0.00549819949059
7	1.00000	0.00000011081045	0.00000011081045
8	1.00000	-0.00000000000001	-0.00000000000001

◇

Beispiel 53 (Versagen des gedämpften Newton-Verfahrens).

- Wie in Bsp. 50: $F(x) = xe^x - 1$,
- Startwert für gedämpftes Newton-Verfahren $x^{(0)} = -1.5$

k	$\lambda^{(k)}$	$x^{(k)}$	$F(x^{(k)})$
1	0.25000	-4.4908445351690	-1.0503476286303
2	0.06250	-6.1682249558799	-1.0129221310944
3	0.01562	-7.6300006580712	-1.0037055902301
4	0.00390	-8.8476436930246	-1.0012715832278
5	0.00195	-10.5815494437311	-1.0002685596314
		Abbruch wegen `lambda < LMIN` !	

Beobachtung: Newton-Korrektur hat „falsche Richtung” ➤ Keine Konvergenz

◇

2.6.3 Quasi-Newton-Verfahren

QUASINEWTON

Was tun, wenn $DF(\mathbf{x})$ nicht verfügbar und numerisches Differenzieren (→ Bem. 49) zu aufwändig?

Idee: in einer Dimension ($n = 1$, → Sekantenverfahren (2.3.4), Abschn. 2.3.2.3)

$$F'(x^{(k)}) \approx \frac{F(x^{(k)}) - F(x^{(k-1)})}{x^{(k)} - x^{(k-1)}} \quad \text{„Differenzenquotient”} \qquad (2.6.3)$$

Schon berechnet ! → billig

Verallgemeinerung für $n > 1$?

(2.6.3) ► SEKBED Sekantenbedingung für Approximation $\mathbf{J}_k \approx DF(\mathbf{x}^{(k)})$, $\mathbf{x}^{(k)} \hat{=}$ Iterierte:

$$\mathbf{J}_k(\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}) = F(\mathbf{x}^{(k)}) - F(\mathbf{x}^{(k-1)}) \; . \qquad (2.6.4)$$

Klar: Viele $\mathbf{J}_k$ erfüllen (2.6.4)

➢ Nötig: Weitere Bedingungen an $\mathbf{J}_k \in \mathbb{R}^{n,n}$

Idee: $\mathbf{J}_k$ durch Modifikation von $\mathbf{J}_{k-1}$

Nebenbedingung (Broyden): $\mathbf{J}_k\mathbf{z} = \mathbf{J}_{k-1}\mathbf{z} \quad \forall \mathbf{z}: \mathbf{z} \cdot (\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}) = 0$.

(2.6.5)

► $\mathbf{J}_k = \mathbf{J}_{k-1} + \frac{F(\mathbf{x}^{(k)})(\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)})^T}{\left\|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\right\|_2^2}$ (2.6.6)

(2.6.6) = RANGONEMOD Rang-1-Modifikation, Abschn. 3.2.8.1

BRPYQN

Broydens Quasi-Newton-Verfahren zur Lösung von $F(\mathbf{x}) = 0$:

$$\mathbf{x}^{(k+1)} := \mathbf{x}^{(k)} + \Delta\mathbf{x}^{(k)}, \; \Delta\mathbf{x}^{(k)} := -\mathbf{J}_k^{-1}F(\mathbf{x}^{(k)}) \; , \; \mathbf{J}_{k+1} := \mathbf{J}_k + \frac{F(\mathbf{x}^{(k+1)})(\Delta\mathbf{x}^{(k)})^T}{\left\|\Delta\mathbf{x}^{(k)}\right\|_2^2} \qquad (2.6.7)$$

Startwert $\mathbf{J}_0 =$? → Verwende etwa exakte Jacobimatrix $DF(\mathbf{x}^{(0)})$

Bemerkung 54 (Minimalität von Broydens Rang-1-Modifikation).

$\mathbf{J} \in \mathbb{R}^{n,n}$ erfüllt (2.6.4) eq:seccond
$\mathbf{J}_k, \mathbf{x}^{(k)}$ aus (2.6.7) eq:broydengn

$$\Rightarrow \quad (\mathbf{I} - \mathbf{J}_k^{-1}\mathbf{J})(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}) = -\mathbf{J}_k^{-1}F(\mathbf{x}^{(k+1)})$$

$$\left\|\mathbf{I} - \mathbf{J}_k^{-1}\mathbf{J}_{k+1}\right\|_2 = \left\|\frac{-\mathbf{J}_k^{-1}F(\mathbf{x}^{(k+1)})\Delta\mathbf{x}^{(k)}}{\left\|\Delta\mathbf{x}^{(k)}\right\|_2^2}\right\|_2 = \left\|(\mathbf{I} - \mathbf{J}_k^{-1}\mathbf{J})\frac{\Delta\mathbf{x}^{(k)}(\Delta\mathbf{x}^{(k)})^T}{\left\|\Delta\mathbf{x}^{(k)}\right\|_2^2}\right\|_2$$

$$\leq \left\|\mathbf{I} - \mathbf{J}_k^{-1}\mathbf{J}\right\|_2 .$$

$\Rightarrow$ (2.6.6) = $\|\cdot\|_2$-minimale relative Korrektur von $\mathbf{J}_{k-1}$, die Sekantenbedingung (2.6.4) respektiert. eq:brodupd eq:seccond △

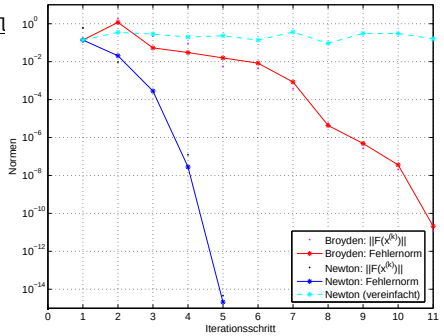
EXQUASINEWON

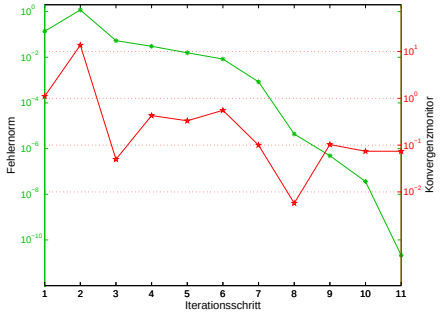
Beispiel 55 (Broydens Quasi-Newton-Verfahren: Konvergenz).

- Nullstellenbestimmung aus Bsp. 46, $n = 2$ exa:newton-mul
- $\mathbf{x}^{(0)} = (0.7.0.7)^T$
- $\mathbf{J}_0 = DF(\mathbf{x}^{(0)})$

Langsamer als Newton-Verfahren (2.5.1) eq:Newton

Besser als vereinfachtes Newton-Verfahren (→ Bem. 48) ▷ rem:simplNewt





Konvergenzmonitor

=

Grösse, die Probleme mit der Konvergenz einer Iteration anzeigt

Hier:

$$\mu := \frac{\left\|\mathbf{J}_{k-1}^{-1}F(\mathbf{x}^{(k)})\right\|}{\left\|\Delta\mathbf{x}^{(k-1)}\right\|}$$

Heuristik: $\mu > 1$ ➢ keine Konvergenz

◇

Bemerkung 56. Option: Gedämpftes Broyden-Verfahen (wie bei Newton-Verfahren, Abschn. 2.6.2) sec:gedampft-r △

Implementierung von (2.6.7): eq:broydengn Mit Sherman-Morrison-Woodbury Update-Formel, Lemma 96 rem:SMW

$$\mathbf{J}_{k+1}^{-1} = \left(\mathbf{I} - \frac{\mathbf{J}_k^{-1}F(\mathbf{x}^{(k+1)})(\Delta\mathbf{x}^{(k)})^T}{\left\|\Delta\mathbf{x}^{(k)}\right\|_2^2 + \Delta\mathbf{x}^{(k)} \cdot \mathbf{J}_k^{-1}F(\mathbf{x}^{(k+1)})}\right)\mathbf{J}_k^{-1} = \left(\mathbf{I} + \frac{\Delta\mathbf{x}^{(k+1)}(\Delta\mathbf{x}^{(k)})^T}{\left\|\Delta\mathbf{x}^{(k)}\right\|_2^2}\right)\mathbf{J}_k^{-1} \tag{2.6.8}$$ eq:Jupd

Sinnvoll, falls $\left\|\mathbf{J}_k^{-1}F(\mathbf{x}^{(k+1)})\right\|_2 < \left\|\Delta\mathbf{x}^{(k)}\right\|_2$ „vereinfachte Quasi-Newton-Korrektur”

MATLAB-CODE: Broyden-Verfahren (2.6.7) eq:broydengn

```
function x = broyden(F,x,J,tol)
k = 1;
[L,U] = lu(J);
s = U\(L\F(x)); sn = dot(s,s);
dx = [s]; dxn = [sn];
x = x - s; f = F(x);

while (sqrt(sn) > tol), k=k+1
  w = U\(L\f);
  for l=2:k-1
    w = w+dx(:,l)*(dx(:,l-1)'*w)...
        /dxn(l-1);
  end
  if (norm(w)>=sn)
    warning('Dubious step %d!',k);
  end
  z = s'*w; s = (1+z/(sn-z))*w;sn=s'*s;
  dx = [dx,s]; dxn = [dxn,sn];
  x = x - s; f = F(x);
end
```

- Einmalige LU-Zerlegung **!**
- Speicherung der $\Delta\mathbf{x}^{(k)}$, $\left\|\Delta\mathbf{x}^{(k)}\right\|_2^2$ (→ (2.6.8)) eq:Jupd
- Löse gestaffelte LGS
- Abbruch, vgl. Abschn. 2.6.1 sec:konv-des-newt
- Aufbau von $\mathbf{w} := \mathbf{J}_k^{-1}F(\mathbf{x}^{(k)})$ (→ Rekursion (2.6.8)) eq:Jupd
- Konvergenzmonitor $\frac{\left\|\mathbf{J}_{k-1}^{-1}F(\mathbf{x}^{(k)})\right\|}{\left\|\Delta\mathbf{x}^{(k-1)}\right\|} < 1$?
- Korrektur $\mathbf{s} = \mathbf{J}_k^{-1}F(\mathbf{x}^{(k)})$

Rechenaufwand : $O(N^2 \cdot n)$ Operationen Vektorarithmetik, (Level)
N Schritte
- 1 LU-Zerlegung von J, $N\times$ Lösung gestaffeltes LGS → Abschn. 3.2.3 sec:die-lr-zerl
- N F-Ausfwertungen !

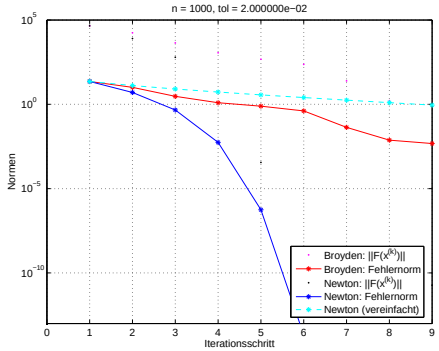
Speicheraufwand : LU-Faktoren von J + Hilfsvektoren $\in \mathbb{R}^n$
N Schritte
- N Vektoren $\mathbf{x}^{(k)} \in \mathbb{R}^n$

Beispiel 57 (Broyden-Verfahren für grosses nichtlineares Gleichungssystem).

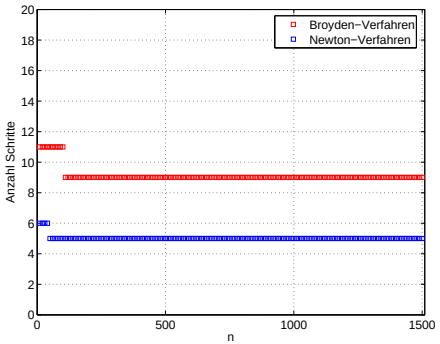
$$F(\mathbf{x}) = \begin{cases} \mathbb{R}^n \mapsto \mathbb{R}^n \\ \mathbf{x} \mapsto \operatorname{diag}(\mathbf{x})\mathbf{A}\mathbf{x} - \mathbf{b} \end{cases},$$
$$\mathbf{b} = (1, 2, \ldots, n) \in \mathbb{R}^n ,$$
$$\mathbf{A} = \mathbf{I} + \mathbf{a}\mathbf{a}^T \in \mathbb{R}^{n,n} ,$$
$$\mathbf{a} = \frac{1}{\sqrt{1 \cdot \mathbf{b} - 1}}(\mathbf{b} - \mathbf{1}) .$$

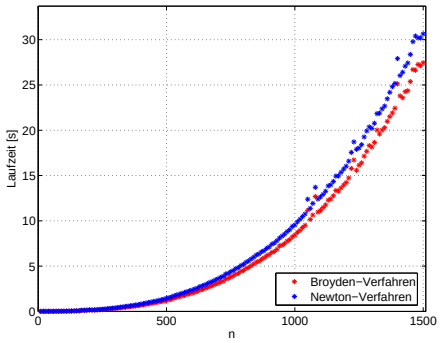
Auswertung wie in Bsp. 55 ex:broydencvg ▷

```
h = 2/n; x0 = (2:h:4-h)';
```



Vergleich der Effizienzen: Broyden-Verfahren ⟷ Newton-Verfahren:
(Problemgrösse n ➢ Toleranz $\texttt{tol} = 2n \cdot 10^{-5}$, `h = 2/n; x0 = (2:h:4-h)';`)





☞ Broyden-Verfahren konkurrenzfähig für $n \gg 1$ und geringe Genauigkeitsanforderungen ◇

2.7 Nichtlineare Ausgleichsrechnung

[File: section-nichtlineare-ausgleichsrechnung.tex, SVN: section-nichtlineare-ausgleichsrechnung.tex 1240 2006-12-30 13:32:59Z hiptmair]

Gegeben: $F : D \subset \mathbb{R}^n \mapsto \mathbb{R}^m$, $m, n \in \mathbb{N}$, $m > n$

➢ Erweiterte Aufgabenstellung: NICHTLINAUSGL
Nichtlineares Ausgleichsproblem

Suche $\mathbf{x}^* \in D$: $\mathbf{x}^* = \operatorname{argmin}_{\mathbf{x}\in D} \Phi(\mathbf{x})$, $\Phi(\mathbf{x}) := \frac{1}{2}\|F(\mathbf{x})\|_2^2$. (2.7.1) nlsq

(Existenz/Eindeutigkeit von $\mathbf{x}^*$ jeweils für jeden Fall konkret nachzuprüfen !)

Terminologie: $D \hat{=}$ Parameterbereich, $x_1, \ldots, x_n \hat{=}$ Parameter

Wir verlangen „Abhängigkeit von allen Parametern”

$\exists$ Umgebung $\mathcal{U}(\mathbf{x}^*)$: $DF(\mathbf{x})$ hat vollen Rang n $\forall \mathbf{x} \in \mathcal{U}(\mathbf{x}^*)$.

(bedeutet: Spalten der Jacobimatrix $DF(\mathbf{x})$ sind linear unabhängig.)

1. Möglichkeit: (gedämpftes) *Newton-Verfahren*

$$\Phi(\mathbf{x}^*) = \min \quad \Rightarrow \quad \mathbf{grad}\,\Phi(\mathbf{x}) = 0, \quad \mathbf{grad}\,\Phi(\mathbf{x}) := (\frac{\partial\Phi}{\partial x_1}(\mathbf{x}), \ldots, \frac{\partial\Phi}{\partial x_n}(\mathbf{x}))^T \in \mathbb{R}^n.$$

Idee: Newton-Verfahren zur Bestimmung einer Nullstelle von $\mathbf{grad}\,\Phi : D \mapsto \mathbb{R}^n$

Kettenregel ▶ $\mathbf{grad}\,\Phi(\mathbf{x}) = DF(\mathbf{x})^T F(\mathbf{x})$,

Produktregel ▶ $H\Phi(\mathbf{x}) := D(\mathbf{grad}\,\Phi)(\mathbf{x}) = DF(\mathbf{x})^T DF(\mathbf{x}) + \sum_{j=1}^{m} F_j(\mathbf{x}) D^2 F_j(\mathbf{x})$.

($H\Phi(\mathbf{x})$ = HESS Hesse-Matrix)

▶ Lineares Gleichungssystem zur Bestimmung der Newton-Korrektur $\mathbf{s} \in \mathbb{R}^n$ zu $\mathbf{x}^{(k)}$:

$$\left(DF(\mathbf{x}^{(k)})^T DF(\mathbf{x}^{(k)}) + \sum_{j=1}^{m} F_j(\mathbf{x}^{(k)}) D^2 F_j(\mathbf{x}^{(k)}) \right) \mathbf{s} = -DF(\mathbf{x}^{(k)})^T F(\mathbf{x}^{(k)}) . \quad (2.7.2)$$ eq:NCNL

Newton-Verfahren = Minimierung einer lokalen quadratischen Approximation von Φ:

$$\Phi(\mathbf{x}) \approx Q(\mathbf{s}) := \Phi(\mathbf{x}^{(k)}) + \mathbf{grad}\,\Phi(\mathbf{x}^{(k)})^T \mathbf{s} + \frac{1}{2}\mathbf{s}^T H\Phi(\mathbf{x}^{(k)})\mathbf{s} .$$

$$\mathbf{grad}\,Q(\mathbf{s}) = 0 \quad \Leftrightarrow \quad H\Phi(\mathbf{x}^{(k)})\mathbf{s} + \mathbf{grad}\,\Phi(\mathbf{x}^{(k)}) = 0 \quad \Leftrightarrow \quad (2.7.2) .$$ eq:NCNLLSQ

Bemerkung 58 (Differenzieren). → Wiederholung: Analysis Grundvorlesung

Seien V, W endlichdimensionale Vektorräume, $F : D \subset V \mapsto W$ hinreichend glatt. Ableitung $DF(\mathbf{x})$ von F in $\mathbf{x} \in V$

$$DF(\mathbf{x}) : V \mapsto W \text{ linear} \quad , \quad \|F(\mathbf{x}+\mathbf{h}) - F(\mathbf{x}) - DF(\mathbf{h})\mathbf{h}\| = o(\|\mathbf{h}\|) \quad \forall \mathbf{h}\,,\ \|\mathbf{h}\| < \delta\,.$$

- $F : V \mapsto W$ linear, d.h. $F(\mathbf{x}) = \mathbf{A}\mathbf{x}$, $\mathbf{A}$ Matrix ➤ $DF(\mathbf{x}) = \mathbf{A}$.

KETTREGEL
- Kettenregel: $F : V \mapsto W$, $G : W \mapsto U$ hinreichend glatt

$$D(G \circ F)(\mathbf{x})\mathbf{h} = DG(F(\mathbf{x}))(DF(\mathbf{x}))\mathbf{h}\,, \quad \mathbf{h} \in V\,, \mathbf{x} \in D\,.$$

PRODREGEL
- Produktregel: $F : D \subset V \mapsto W$, $G : D \subset V \mapsto U$ hinreichend glatt, $b : W \times U \mapsto Z$ bilinear

$$T(\mathbf{x}) = b(F(\mathbf{x}), G(\mathbf{x})) \quad \blacktriangleright \quad DT(\mathbf{x})\mathbf{h} = b(DF(\mathbf{x})\mathbf{h}, G(\mathbf{x})) + b(F(\mathbf{x}), DG(\mathbf{x})\mathbf{h})\,,$$
$$\mathbf{h} \in V, \mathbf{x} \in D\,.$$

Für $F : D \subset V \mapsto \mathbb{R}$ sind der Gradient $\mathbf{grad}\, F : D \mapsto \mathbb{R}^n$, $n := \dim V$, und die Hessematrix $HF(\mathbf{x}) : D \mapsto \mathbb{R}^{n,n}$ definiert durch

$$\mathbf{grad}\, F(\mathbf{x})^T\mathbf{h} := DF(\mathbf{x})\mathbf{h} \quad , \quad \mathbf{h}_1^T HF(\mathbf{x})\mathbf{h}_2 := D(DF(\mathbf{x})(\mathbf{h}_1))(\mathbf{h}_2)\,, \quad \mathbf{h}, \mathbf{h}_1, \mathbf{h}_2 \in V\,.$$

△

2. Möglichkeit: GAUSSNEWTON Gauss-Newton-Verfahren

Idee: Lokale Linearisierung von F (anstatt lokal quadratischer Approximation von Φ)

Korrektur $\mathbf{s}$ zur aktuellen Iterierten $\mathbf{x}^{(k)}$ so dass

$$\left\| F(\mathbf{x}^{(k)}) + DF(\mathbf{x}^{(k)})\mathbf{s} \right\|_2 \longrightarrow \min$$

(Lineares Ausgleichsproblem → Abschnitt 3.5) sec:numer-line-aus (2.7.3) GNs

Der Operator \: die Wunderwaffe in MATLAB:

Für $\mathbf{A} \in \mathbb{R}^{m,n}$

x = A\b

↕

$\mathbf{x}$ Minimierer von $\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2$ mit minimaler 2-Norm,

MATLAB-Template für Gauss-Newton-Verfahren

```
function x = gn(x,f,J,tol)
s = J(x)\f(x);
x = x-s;
while (norm(s) > tol)
  s = J(x)\f(x);
  x = x-s;
end
```

Vorteil des Gauss-Newton-Verfahrens : Keine zweiten Ableitungen von F erforderlich
Nachteil des Gauss-Newton-Verfahrens : Keine lokal quadratische Konvergenz

3. Möglichkeit: TRUSTREGION Trust-Region-Verfahren (Levenberg-Marquardt-Verfahren)

Idee: Dämpfung der Gauss-Newton-Korrektur aus (2.7.3) GNs durch Strafterm (engl. *penalty term*)

Statt $\left\| F(\mathbf{x}^{(k)}) + DF(\mathbf{x}^{(k)})\mathbf{s} \right\|^2$ minimiere $\left\| F(\mathbf{x}^{(k)}) + DF(\mathbf{x}^{(k)})\mathbf{s} \right\|^2 + \lambda \|\mathbf{s}\|_2^2$.

$\lambda > 0 \hat{=}$ Strafparameter (wie zu wählen ? → Heuristik)

$$\lambda = \gamma \left\| F(\mathbf{x}^{(k)}) \right\|_2 \quad , \quad \gamma := \begin{cases} 10 & \text{, falls } \left\| F(\mathbf{x}^{(k)}) \right\|_2 \geq 10\,, \\ 1 & \text{, falls } 1 < \left\| F(\mathbf{x}^{(k)}) \right\|_2 < 10\,, \\ 0.01 & \text{, falls } \left\| F(\mathbf{x}^{(k)}) \right\|_2 \leq 1\,. \end{cases}$$

► Modifizierte (regularisierte) Gleichung für Korrektur $\mathbf{s}$:

$$\left(DF(\mathbf{x}^{(k)})^T DF(\mathbf{x}^{(k)}) + \lambda \mathbf{I} \right) \mathbf{s} = -DF(\mathbf{x}^{(k)}) F(\mathbf{x}^{(k)})\,. \tag{2.7.4}$$ LMs

NICHTLINREGR

Beispiel 59 (Nichtlineare Regression, parametrische Statistik).

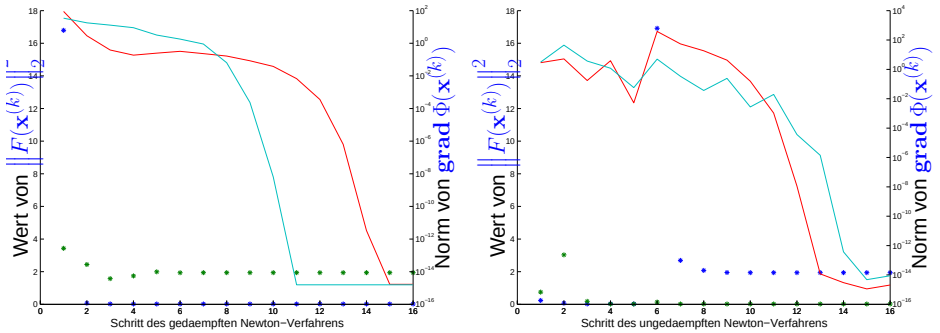
Parametrische Statistik: Fitten von Messwerten (t_i, y_i), $i = 1, \ldots, m$ durch nichtlinear parameterabhängige Funktion $f : \mathbb{R} \mapsto \mathbb{R}$, speziell $f(t) = x_1 + x_2 \exp(-x_3 t)$.

$$\blacktriangleright \quad F(\mathbf{x}) = \begin{pmatrix} x_1 + x_2 \exp(-x_3 t_1) - y_1 \\ \vdots \\ x_1 + x_2 \exp(-x_3 t_m) - y_m \end{pmatrix} : \mathbb{R}^3 \mapsto \mathbb{R}^m\,, \quad DF(\mathbf{x}) = \begin{pmatrix} 1 & e^{-x_3 t_1} & -x_2 t_1 e^{-x_3 t_1} \\ \vdots & \vdots & \vdots \\ 1 & e^{-x_3 t_m} & -x_2 t_m e^{-x_3 t_m} \end{pmatrix}$$

Numerisches Experiment:

Konvergenz des Newton-Verfahrens, gedämpften Newton-Verfahrens (→Abschnitt sec:gedampft-newt-verf 2.6.2) und Gauss-Newton-Verfahrens für verschiedene Startwerte

```
rand('seed',0);
t = (1:0.3:7)';
y = x(1) + x(2)*exp(-x(3)*t);
y = y+0.1*(rand(length(y),1)-0.5);
```



◇

Konvergenzverhalten des Newton-Verfahrens:

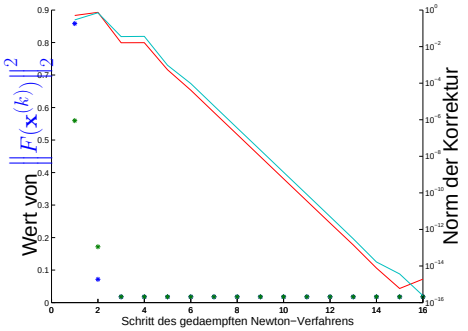
Anfangswert $(1.8, 1.8, 0.1)^T$ (rote Kurve) ➢ Newton-Verfahren gefangen in lokalem Minimum
Anfangswert $(1.5, 1.5, 0.1)^T$ (cyan) Kurve) ➢ Schnelle (lokal quadratische) Konvergenz

Gauss-Newton-Verfahren

Anfangswert $(1.8, 1.8, 0.1)^T$ (rote Kurve)
Anfangswert $(1.5, 1.5, 0.1)^T$ (cyan) Kurve)

Konvergenz in beiden Fällen

Beobachtung: Lineare Konvergenz



3 Numerische lineare Algebra

[File: chapter-numerische-lineare-algebra.tex, SVN: chapter-numerische-lineare-algebra.tex 1055 2006-10-18 15:44:34Z kalai]

☛ Klasse von Problemen, beschreibbar durch elementare Operationen mit Matrizen und/oder Vektoren im $\mathbb{R}^n$ oder $\mathbb{C}^n$, z.B.

- Lineare Gleichungssysteme (engl. *linear systems of equations*)
- Lineare Ausgleichsprobleme (engl. *linear least squares*)
- Eigenwertprobleme (engl. *eigenproblems*)

Algorithmen/Konzepte der numerischen linearen Algebra sind Teil/Grundlage fast aller numerischer Methoden

3.1 Grundbegriffe und -operationen

[File: section-grundbegriffe-und-operationen.tex, SVN: section-grundbegriffe-und-operationen.tex 1010 2006-09-11 15:44:43Z hiptmair]

Konventionen:

- Alle Vektoren sind Spaltenvektoren (engl. *column vectors*) [Zeilenvektor = $\mathbf{x}^T$],
  MATLAB Initialisierung: `x = [1;2];`, für Zeilenvektor `xt = [1,2];`
  Notation: **fette** Kleinbuchstaben
  Komponenten $\mathbf{x} = (x_1, \ldots, x_n)^T \in \mathbb{K}^n$, $\mathbb{K} = \mathbb{R}, \mathbb{C}$, bzw. $x_i = (\mathbf{x})_i$, $i = 1, \ldots, n$.
- **Fette** Grossbuchstaben für Matrizen, z.B. $\mathbf{A}$, Matrixeinträge Kleinbuchstaben mit subscript Indizes, z.B. a_{ij} oder $(\mathbf{A})_{ij}$ ($i \to$ Zeilenindex, $j \to$ Spaltenindex)
- Für Vektor $\mathbf{x} \in \mathbb{R}^n$/$\mathbf{x} \in \mathbb{C}^n$, $n \in \mathbb{N}$:

$$\mathbf{x}^T = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}^T := (x_1 \cdots x_n) \ , \quad \mathbf{x}^H = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}^H := (\bar{x}_1 \cdots \bar{x}_n) \ , \quad \bar{} = \text{komplexe Konjugation.}$$

- Für Matrix $\mathbf{A} \in \mathbb{K}^{n,m}$ → Transponierte Matrix (engl. *transposed matrix*)

$$\mathbf{A}^T = \begin{pmatrix} a_{11} & \dots & a_{1m} \\ \vdots & & \vdots \\ a_{n1} & \dots & a_{nm} \end{pmatrix}^T := \begin{pmatrix} a_{11} & \dots & a_{n1} \\ \vdots & & \vdots \\ a_{1m} & \dots & a_{mn} \end{pmatrix} \in \mathbb{K}^{m,n}, \quad \mathbf{A}^H := \begin{pmatrix} \bar{a}_{11} & \dots & \bar{a}_{n1} \\ \vdots & & \vdots \\ \bar{a}_{1m} & \dots & \bar{a}_{mn} \end{pmatrix} \in \mathbb{K}^{m,n}.$$

- Spalten (engl. *columns*) einer Matrix $\mathbf{A} \in \mathbb{K}^{m,n}$

$$\mathbf{A} = \left[\mathbf{a}_{\cdot,1} \cdots \mathbf{a}_{\cdot,n}\right]$$

MATLAB: `A(:,1)` ... `A(:,n)`

Zeilen (engl. *rows/lines*) einer Matrix $\mathbf{A} \in \mathbb{K}^{m,n}$

$$\mathbf{A} = \begin{bmatrix} \mathbf{a}_{1,\cdot}^T \\ \vdots \\ \mathbf{a}_{m,\cdot}^T \end{bmatrix}$$

MATLAB: `A(1,:)` ... `A(m,:)`

3.1.1 Operationen

KOMPLEXITY
Komplexität ↔ Aufwand für Algorithmus in Abhängigkeit von Problemgrössenparameter n

Notation („Landau-O"): $f(n) = O(g(n)) \Leftrightarrow \exists C > 0, N > 0 : |f(n)| \leq Cg(n)$ für alle $n > N$.

LEVEL I
- (Euklidisches) Skalarprodukt: $\mathbf{x} \cdot \mathbf{y} = \mathbf{x}^H \mathbf{y} \in \mathbb{K}$, $\mathbf{x}, \mathbf{y} \in \mathbb{K}^n$, $\mathbb{K} = \mathbb{R}, \mathbb{C}$.
  - ➢ MATLAB operator `dot(x,y)`, Rechenaufwand = $n - 1$ Additionen, n Multiplikationen.
  - ➢ Komplexität $O(n)$ (minimal)

LEVEL I
- SAXPY: $\mathbf{z} = \alpha\mathbf{x} + \mathbf{y}$, $\mathbf{x}, \mathbf{y} \in \mathbb{K}^n$, $\alpha \in \mathbb{K}$,
  - ➢ MATLAB `z=alpha*x+y`, Rechenaufwand = n Additionen, Multiplikationen.
  - ➢ Komplexität $O(n)$ (minimal)

LEVEL II
- Vektor×Matrix-Produkt $\mathbf{y} = \mathbf{A}\mathbf{x}$, $\mathbf{A} \in \mathbb{K}^{n,m}$
  - ➢ MATLAB `y=A*x`, Rechenaufwand = m Skalarprodukte.
  - ➢ Komplexität $O(mn)$ (minimal)

LEVEL III
- Matrix×Matrix-Produkt $\mathbf{C} = \mathbf{A}\mathbf{B}$, $\mathbf{A} \in \mathbb{K}^{m,n}$, $\mathbf{B} \in \mathbb{K}^{n,k}$
  - ➢ MATLAB `C=A*B`, Rechenaufwand = k-mal Matrix $\mathbf{A}\times$ Vektor.
  - ➢ Komplexität $O(kmn)$ (nicht minimal !)

Bemerkung 60. Effiziente maschinennahe Implementierung der Grundoperationen:
BLAS
BLAS-Bibliothek (`www.netlib.org/blas/`) → Vorlesung „Programmiertechniken"

△

Bemerkung 61. Blockweise Matrixmultiplikation: $\mathbf{A}_{11} \in \mathbb{K}^{n,n}$, $\mathbf{A}_{12} \in \mathbb{K}^{n,m}$, $\mathbf{A}_{21} \in \mathbb{K}^{m,n}$, $\mathbf{A}_{22} \in \mathbb{K}^{m,m}$, $\mathbf{B}_{11} \in \mathbb{K}^{n,n}$, $\mathbf{B}_{12} \in \mathbb{K}^{n,m}$, $\mathbf{B}_{21} \in \mathbb{K}^{m,n}$, $\mathbf{B}_{22} \in \mathbb{K}^{m,m}$, $m, n \in \mathbb{N}$:

$$\begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{pmatrix} \begin{pmatrix} \mathbf{B}_{11} & \mathbf{B}_{12} \\ \mathbf{B}_{21} & \mathbf{B}_{22} \end{pmatrix} = \begin{pmatrix} \mathbf{A}_{11}\mathbf{B}_{11} + \mathbf{A}_{12}\mathbf{B}_{21} & \mathbf{A}_{11}\mathbf{B}_{12} + \mathbf{A}_{12}\mathbf{B}_{22} \\ \mathbf{A}_{21}\mathbf{B}_{11} + \mathbf{A}_{22}\mathbf{B}_{21} & \mathbf{A}_{21}\mathbf{B}_{12} + \mathbf{A}_{22}\mathbf{B}_{22} \end{pmatrix}. \quad (3.1.1)$$

eq:Blocki

△

3.1.2 Matrix-Speicherformate

MATSPFORMAT
Zwei Hauptklassen von Matrizen:

VOLLBESMAT
Vollbesetze (engl. *dense*) Matrizen ◆ Dünnbesetzte (engl. *sparse*) Matrizen

DUENNBESMAT
Definition 3.1.1 (Dünnbesetzte Matrix). $\mathbf{A} \in \mathbb{K}^{m,n}$, $m, n \in \mathbb{N}$, *heisst dünnbesetzt, falls*

$$\operatorname{nnz}(\mathbf{A}) := \sharp\{(i,j) \in \{1,\dots,m\} \times \{1,\dots,n\} : a_{ij} \neq 0\} \ll mn.$$

- Speicherung vollbesetzter Matrizen („Standard"):

$\mathbf{A} \in \mathbb{K}^{m,n}$ ► Lineares Array (Grösse mn) + Indexarithmetik
(Beachte führende Dimension (engl. *row major, column major*))

- Speicherung dünnbesetzter Matrizen, Beispiel Compressed-Row-Storage (CRS)-Format:

CRS

Matrixinformation für $\mathbf{A} \in \mathbb{K}^{n,n}$ wird in drei Arrays gespeichert:

`real val`	Grösse $\operatorname{nnz}(\mathbf{A}) := \sharp\{(i,j) \in \{1,\dots,n\}^2, a_{ij} \neq 0\}$
`int col_ind`	Grösse $\operatorname{nnz}(\mathbf{A})$
`int row_ptr`	Grösse $n+1$ & `row_ptr`$[N+1] = \operatorname{nnz}(\mathbf{A}) + 1$

Zugriff auf Matrixeintrag $a_{ij} \neq 0$, $1 \leq i, j \leq n$:

$$\texttt{val}[k] = a_{ij} \quad \Leftrightarrow \quad \begin{cases} \texttt{col\_ind}[k] = j, \\ \texttt{row\_ptr}[i] \leq k < \texttt{row\_ptr}[i+1], \end{cases} \quad 1 \leq k \leq \operatorname{nnz}(\mathbf{A}).$$

Beispiel 62 (CRS-Matrixspeicherformat).

$$\mathbf{A} = \begin{pmatrix} 10 & 0 & 0 & 0 & -2 & 0 \\ 3 & 9 & 0 & 0 & 0 & 3 \\ 0 & 7 & 8 & 7 & 0 & 0 \\ 3 & 0 & 8 & 7 & 5 & 0 \\ 0 & 8 & 0 & 9 & 9 & 13 \\ 0 & 4 & 0 & 0 & 2 & -1 \end{pmatrix}$$

val	10	-2	3	9	3	7	8	7	3...9	13	4	2	-1
col_ind	1	5	1	2	6	2	3	4	1...5	6	2	5	6
row_ptr	1	3	6	9	13	17	20						

Variante: Diagonales CRS-Format (Matrixdiagonale wird separat gespeichert)

Dünnbesetzte Matrizen in MATLAB [18]:

Initialisierungsfunktionen:
```
A = sparse(m,n); A = spalloc(m,n,nnz)
A = sparse(i,j,s,m,n);
A = spdiags(B,d,m,n); A = speye(n); A = spones(S);
```

Beispiel 63 (Effiziente Initialisierung von Sparse-Matrizen).

① :

Initialisierung dünnbesetzter Matrizen I
```
A1 = sparse(n,n);
for i=1:n
   for j=1:n
   if (abs(i-j) == 1), A1(i,j) = A1(i,j) + 1; end;
   if (abs(i-j) == round(n/3)), A1(i,j) = A1(i,j) -1; end;
 end; end
```
Direkte Initialisierung

② :

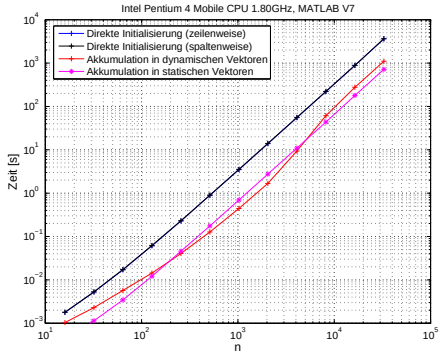
Initialisierung dünnbesetzter Matrizen II
```
dat = [];
for i=1:n
   for j=1:n
     if (abs(i-j) == 1), dat = [dat; i,j,1.0]; end;
     if (abs(i-j) == round(n/3)), dat = [dat; i,j,-1.0];
     end; end; end;
A2 = sparse(dat(:,1),dat(:,2),dat(:,3),n,n);
```
Akkumulation in dynamischem Vektor

③ :

Initialisierung dünnbesetzter Matrizen III
```
dat = zeros(6*n,3); k = 0;
for i=1:n
  for j=1:n
     if (abs(i-j) == 1), k=k+1; dat(k,:) = [i,j,1.0];
     end;
     if (abs(i-j) == round(n/3))
         k=k+1; dat(k,:) = [i,j,-1.0];
     end;
  end; end;
A3 = sparse(dat(1:k,1),dat(1:k,2),dat(1:k,3),n,n);
```
Akkumulation in statischem Vektor

Zeitmessung:

(MATLAB `tic,toc`-Funktionen)
(Mobile Intel Pentium 4 - M CPU 2.40GHz, Linux MATLAB V6.5)



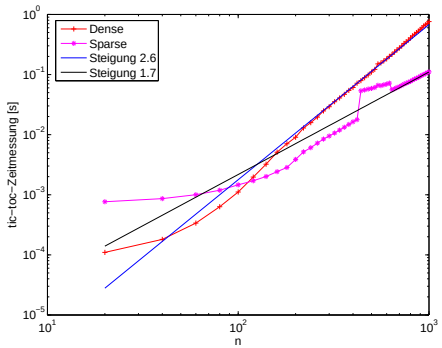
Beachte: Vollbesetzt $\star$ dünnbesetzt ➔ vollbesetzt
dünnbesetzt $+, -, *$ dünnbesetzt ➔ dünnbesetzt
`S = [A,B;C,D]` dünnbesetzt, wenn ein Block dünnbesetzt

Beispiel 64 (Effiziente MATLAB-Implementierungen mit `sparse`-Matrizen).

Initialisierung:
```
A = rand(n,n); v = rand(n,1);
```

☞ Aufgabe: Skalierung einer Matrix:

- Dense-Matrixarithmetik:
  ```
  S = A*diag(v)+eye(n)
  ```
- Sparse-Matrixarithmetik:
  ```
  S = A*spdiags(v,0,n,n)+speye(n)
  ```



3.1.3 Erinnerung: Normen

Begriffe Kondition (→ Abschn. sec:kondition 1.6), Stabilität (→ Abschn. sec:stabilitat 1.8), Fehler ➔ erfordern Metrik auf $\mathbb{K}^n$

Grundvorlesungen: Norm $\|\cdot\|$ auf $\mathbb{K}^n$ ⇒ Metrik auf $\mathbb{K}^n$

Definition. → Def. def:norm 2.1.4
X = $\mathbb{K}$-Vektorraum, $\mathbb{K} = \mathbb{C}, \mathbb{R}$. Eine Abbildung $\|\cdot\| : X \mapsto \mathbb{R}_0^+$ ist eine Norm auf X, falls

(i) $\forall x \in X: \quad x \neq 0 \quad \Leftrightarrow \quad \|x\| > 0$ (Definitheit),

(ii) $\|\lambda x\| = |\lambda| \|x\| \quad \forall x \in X, \lambda \in \mathbb{K}$ (Homogenität),

(iii) $\|x + y\| \leq \|x\| + \|y\| \quad \forall x, y \in X$ (Dreiecksungleichung).

Beispiele: (für Vektor $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T \in \mathbb{K}^n$)

Bezeichnung	:	Definition	MATLAB-Funktion
Euklidische Norm	:	$\|\mathbf{x}\|_2 := \sqrt{\lvert x_1\rvert^2 + \cdots + \lvert x_n\rvert^2}$	`norm(x)`
1-Norm	:	$\|\mathbf{x}\|_1 := \lvert x_1\rvert + \cdots + \lvert x_n\rvert$	`norm(x,1)`
Maximumnorm	:	$\|\mathbf{x}\|_\infty := \max\{\lvert x_1\rvert, \ldots, \lvert x_n\rvert\}$	`norm(x,inf)`

Definition. → Def. def:mnorm 2.2.6
Der Norm $\|\cdot\|$ auf $\mathbb{R}^n$ zugeordnete Matrixnorm

$$\mathbf{M} \in \mathbb{R}^{n,n}: \quad \|\mathbf{M}\| := \sup_{x \in \mathbb{R}^n \setminus \{0\}} \frac{\|\mathbf{M}x\|}{\|x\|} .$$

➤ SUBMULT Submultiplikativität: $\mathbf{A} \in \mathbb{K}^{n,m}, \mathbf{B} \in \mathbb{K}^{m,k}: \quad \|\mathbf{AB}\| \leq \|\mathbf{A}\| \, \|\mathbf{B}\|$

Notation: $\|\mathbf{x}\|_2 \to \|\mathbf{M}\|_2, \quad \|\mathbf{x}\|_1 \to \|\mathbf{M}\|_1, \quad \|\mathbf{x}\|_\infty \to \|\mathbf{M}\|_\infty$

Beispiel 65 (Matrixnorm zur Maximumnorm und 1-Norm).

z.B. für $\mathbf{M} \in \mathbb{K}^{2,2}$:

$$\begin{aligned} \|\mathbf{Mx}\|_\infty &= \max\{|m_{11}x_1 + m_{12}x_2|, |m_{21}x_1 + m_{22}x_2|\} \\ &\leq \max\{|m_{11}| + |m_{12}|, |m_{21}| + |m_{22}|\} \, \|\mathbf{x}\|_\infty , \\ \|\mathbf{Mx}\|_1 &= |m_{11}x_1 + m_{12}x_2| + |m_{21}x_1 + m_{22}x_2| \\ &\leq \max\{|m_{11}| + |m_{21}|, |m_{12}| + |m_{22}|\}(|x_1| + |x_2|) . \end{aligned}$$

➢ Matrixnorm zu $\|\cdot\|_\infty$ = Zeilensummennorm $\|\mathbf{M}\|_\infty := \max_{i=1,\ldots,n} \sum_{j=1}^{n} |m_{ij}|$, (3.1.2) eq:defrow

➢ Matrixnorm zu $\|\cdot\|_1$ = Spaltensummennorm $\|\mathbf{M}\|_1 := \max_{j=1,\ldots,m} \sum_{i=1}^{n} |m_{ij}|$. (3.1.3) eq:defcol

3.2 Numerische Lösung linearer Gleichungssysteme

[File: section-numerische-loesung-linearer-gleichungssysteme.tex, SVN: section-numerische-loesung-linearer-gleichungssysteme.tex 1137 2006-11-13 09:27

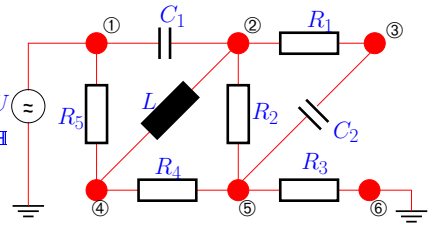
NOTATION Gegeben: Matrix $\mathbf{A} \in \mathbb{K}^{n,n}$, Vektor $\mathbf{b} \in \mathbb{K}^n$, $n \in \mathbb{K}$
Gesucht : Lösungsvektor $\mathbf{x} \in \mathbb{K}^n$: $\boxed{\mathbf{Ax} = \mathbf{b}}$ ← LSG Lineares Gleichungssystem (LGS)

KNOTENNANALYSE
Beispiel 66 (Knotenanalyse eines linearen elektrischen Netzwerks).

Numeriere Knoten $1, \ldots, n$
I_{kj}: Strom Knoten k → Knoten j, $I_{kj} = -I_{jk}$
KIRCHHOFF
Kirchhoffsche Regel: Zweigstromsumme $= 0$

$$\forall k \in \{1, \ldots, n\}: \quad \sum_{j=1}^{n} I_{kj} = 0 . \quad (3.2.1)$$ KH

Unbekannte: Knotenpotentiale u_k, $k = 1, \ldots, n$.
(davon einige fixiert: Erdung, Quellen)



BAUELGL
Bauelementgleichungen
(im Frequenzbereich mit Winkelfrequenz $\omega > 0$):
OHMWID
• Ohmscher Widerstand: $I = R^{-1}U$, $[R] = 1\mathrm{VA}^{-1}$
KONDENSATOR
• Kondensator: $I = i\omega CU$, Kapazität $[C] = 1\mathrm{AsV}^{-1}$
SPULE
• Spule : $I = -i\omega^{-1}L^{-1}U$, Induktivität $[L] = 1\mathrm{VsA}^{-1}$

➤ $I_{kj} = \begin{cases} R^{-1}(u_k - u_j) , \\ i\omega C(u_k - u_j) , \\ -i\omega^{-1}L^{-1}(u_k - u_j) . \end{cases}$

Bauelementgleichungen + (3.2.1) ➤ Lineares Gleichungssystem

$$\begin{aligned} ② &: \quad i\omega C_1(u_2 - u_1) + R_1^{-1}(u_2 - u_3) - i\omega^{-1}L^{-1}(u_2 - u_4) + R_2^{-1}(u_2 - u_5) = 0 , \\ ③ &: \quad R_1^{-1}(u_3 - u_2) + i\omega C_2(u_3 - u_5) = 0 , \\ ④ &: \quad R_5^{-1}(u_4 - u_1) - i\omega^{-1}L^{-1}(u_4 - u_2) + R_4^{-1}(u_4 - u_5) = 0 , \\ ⑤ &: \quad R_2^{-1}(u_5 - u_2) + i\omega C_2(u_5 - u_3) + R_4^{-1}(u_5 - u_4) + R_3(u_5 - u_6) = 0 , \\ & \quad u_1 = U \quad , \quad u_6 = 0 . \end{aligned}$$

$$\begin{pmatrix} i\omega C_1 + \frac{1}{R_1} - \frac{i}{\omega L} + \frac{1}{R_2} & -\frac{1}{R_1} & \frac{i}{\omega L} & -\frac{1}{R_2} \\ -\frac{1}{R_1} & \frac{1}{R_1} + i\omega C_2 & 0 & -i\omega C_2 \\ \frac{i}{\omega L} & 0 & \frac{1}{R_5} - \frac{i}{\omega L} + \frac{1}{R_4} & -\frac{1}{R_4} \\ -\frac{1}{R_2} & -i\omega C_2 & -\frac{1}{R_4} & \frac{1}{R_2} + i\omega C_2 + \frac{1}{R_4} \end{pmatrix} \begin{pmatrix} u_2 \\ u_3 \\ u_4 \\ u_5 \end{pmatrix} = \begin{pmatrix} i\omega C_1 U \\ 0 \\ \frac{1}{R_5} U \\ 0 \end{pmatrix}$$

3.2.1 Theorie und Kondition

Bekannt aus der linearen Algebra:

Definition 3.2.1. *Der Rang einer Matrix $\mathbf{M} \in \mathbb{K}^{m,n}$ (engl. rank) ist die maximale Anzahl linear unabhängiger Spalten/Zeilen von $\mathbf{M}$.*

Theorem 3.2.2 (Invertierbarkeit von Matrizen)**.** *Eine Matrix $\mathbf{A} \in \mathbb{R}^{n,n}$ heisst regulär oder invertierbar, falls eine der folgenden äquivalenten Bedingungen erfüllt ist:*

1. $\exists \mathbf{B} \in \mathbb{R}^{n,n}$: $\mathbf{BA} = \mathbf{AB} = \mathbf{I}$, *wobei* $\mathbf{I}$ *die* $n \times n$*-Einheitsmatrix ist.*
2. $\mathbf{x} \mapsto \mathbf{Ax}$ *definiert einen Isomorphismus des* $\mathbb{R}^n$ ($\mathbf{Ax} = \mathbf{b}$ *hat genau eine Lösung* $\forall \mathbf{b}$)
3. *Die Spalten von* $\mathbf{A}$ *sind linear unabhängig (voller Spaltenrang).*
4. *Die Zeilen von* $\mathbf{A}$ *sind linear unabhängig (voller Zeilenrang).*
5. $\det \mathbf{A} \neq 0$.

Abschn. 3.1.3: Fixiere Norm $\|\cdot\|$ auf $\mathbb{R}^n$ ➢ induziert Matrixnorm (→ Def. 2.2.6) auf $\mathbb{R}^{n,n}$

Vorüberlegung: Relative Konditionszahl von $\mathbf{x} \mapsto \mathbf{Ax}$, $\mathbf{A}$ regulär (bzgl. Störungen in $\mathbf{x}$):

$$\mathbf{A}(\mathbf{x} + \delta\mathbf{x}) = \mathbf{y} + \delta\mathbf{y} \quad \Rightarrow \quad \frac{\|\delta\mathbf{y}\|}{\|\mathbf{y}\|} \leq \boxed{\|\mathbf{A}\|\,\|\mathbf{A}^{-1}\|}\, \frac{\|\delta\mathbf{x}\|}{\|\mathbf{x}\|} \,. \tag{3.2.2}$$

Herleitung der Abschätzung durch

$$\|\delta\mathbf{y}\| \leq \|\mathbf{A}\|\,\|\delta\mathbf{x}\| \quad \text{and} \quad \|\mathbf{y}\| = \|\mathbf{Ax}\| \geq \left\|\mathbf{A}^{-1}\right\|^{-1} \|\mathbf{x}\| \,.$$

Bei LGS: Wie wirken sich (kleine) Störungen in Eingabedaten $\mathbf{A}, \mathbf{b}$ auf $\mathbf{x}$ aus **?**

$$(\mathbf{A} + \Delta\mathbf{A})(\mathbf{x} + \Delta\mathbf{x}) = \mathbf{b} + \Delta\mathbf{b} \quad , \quad \Delta\mathbf{x} = ?$$

Lemma 3.2.3 (Störungslemma)**.**

$$\mathbf{B} \in \mathbb{R}^{n,n},\ \|\mathbf{B}\| < 1 \quad \Rightarrow \quad \mathbf{I} + \mathbf{B} \text{ regulär} \quad \wedge \quad \left\|(\mathbf{I} + \mathbf{B})^{-1}\right\| \leq \frac{1}{1 - \|\mathbf{B}\|}.$$

Beweis. Dreiecksungleichung ➢ $\|(\mathbf{I} + \mathbf{B})\mathbf{x}\| \geq (1 - \|\mathbf{B}\|)\,\|\mathbf{x}\|, \forall \mathbf{x} \in \mathbb{R}^n$ ➢ $\mathbf{B}$ regulär.

$$\blacktriangleright \quad \left\|(\mathbf{I} + \mathbf{B})^{-1}\right\| = \sup_{\mathbf{x} \in \mathbb{R}^n \setminus \{0\}} \frac{\left\|(\mathbf{I} + \mathbf{B})^{-1}\mathbf{x}\right\|}{\|\mathbf{x}\|} = \sup_{\mathbf{y} \in \mathbb{R}^n \setminus \{0\}} \frac{\|\mathbf{y}\|}{\|(\mathbf{I} + \mathbf{B})\mathbf{y}\|} \leq \frac{1}{1 - \|\mathbf{B}\|}$$

Wegen Submultiplikativität der Matrixnorm: $\|\mathbf{AB}\| \leq \|\mathbf{A}\|\,\|\mathbf{B}\|$:

$\mathbf{A}$ regulär: $(\mathbf{A} + \Delta\mathbf{A}) = \mathbf{A}(\mathbf{I} + \mathbf{A}^{-1}\Delta\mathbf{A})$ regulär, falls $\|\Delta\mathbf{A}\| < \left\|\mathbf{A}^{-1}\right\|^{-1}$.

$$\mathbf{Ax} = \mathbf{b} \quad \leftrightarrow \quad (\mathbf{A} + \Delta\mathbf{A})(\mathbf{x} + \Delta\mathbf{x}) = \mathbf{b} + \Delta\mathbf{b} \quad \blacktriangleright \quad (\mathbf{A} + \Delta\mathbf{A})\Delta\mathbf{x} = \Delta\mathbf{b} - \Delta\mathbf{Ax} \,. \tag{3.2.3}$$

Theorem 3.2.4 (Kondition der Lösung linearer Gleichungssysteme)**.**
Falls $\mathbf{A}$ regulär, $\|\Delta\mathbf{A}\| < \left\|\mathbf{A}^{-1}\right\|^{-1}$ und (3.2.3), dann

$$\frac{\|\Delta\mathbf{x}\|}{\|\mathbf{x}\|} \leq \frac{\left\|\mathbf{A}^{-1}\right\|\,\|\mathbf{A}\|}{1 - \left\|\mathbf{A}^{-1}\right\|\,\|\mathbf{A}\|\,\|\Delta\mathbf{A}\| / \|\mathbf{A}\|} \left(\frac{\|\Delta\mathbf{b}\|}{\|\mathbf{b}\|} + \frac{\|\Delta\mathbf{A}\|}{\|\mathbf{A}\|} \right) .$$

Relativer Fehler — *Relative Störungen*

Lemma 3.2.3 ➢ $\left\|(\mathbf{A} + \Delta\mathbf{A})^{-1}\right\| \leq \dfrac{\left\|\mathbf{A}^{-1}\right\|}{1 - \left\|\mathbf{A}^{-1}\Delta\mathbf{A}\right\|}$ & (3.2.3)

$$\Rightarrow \|\Delta\mathbf{x}\| \leq \frac{\left\|\mathbf{A}^{-1}\right\|}{1 - \left\|\mathbf{A}^{-1}\Delta\mathbf{A}\right\|} (\|\Delta\mathbf{b}\| + \|\Delta\mathbf{Ax}\|) \leq \frac{\left\|\mathbf{A}^{-1}\right\|\,\|\mathbf{A}\|}{1 - \left\|\mathbf{A}^{-1}\right\|\,\|\Delta\mathbf{A}\|} \left(\frac{\|\Delta\mathbf{b}\|}{\|\mathbf{A}\|\,\|\mathbf{x}\|} + \frac{\|\Delta\mathbf{A}\|}{\|\mathbf{A}\|} \right) \|\mathbf{x}\| \,.$$

Definition 3.2.5 (Kondition)**.** *Kondition einer Matrix* $\mathbf{A} \in \mathbb{R}^{n,n}$: $\operatorname{cond}(\mathbf{A}) := \left\|\mathbf{A}^{-1}\right\|\,\|\mathbf{A}\|$

Beachte: $\operatorname{cond}(\mathbf{A})$ abhängig von $\|\cdot\|$!

Merkregel:
Wenn $\operatorname{cond}(\mathbf{A}) \approx 10^s$ und relative Störungen $\|\Delta\mathbf{A}\| : \|\mathbf{A}\|, \|\Delta\mathbf{b}\| : \|\mathbf{b}\| \approx 10^{-k}$, $k > s$, dann können von der Lösung des LGS $\mathbf{Ax} = \mathbf{b}$ nur $k - s$ korrekte Dezimalstellen erwartet werden.

Beispiel 67 („Schlecht" konditioniertes lineares Gleichungssystem).

$$\mathbf{Ax} = \mathbf{b} \quad \text{mit} \quad \mathbf{A} = \begin{pmatrix} 4.1 & 2.8 \\ 9.7 & 6.6 \end{pmatrix} , \mathbf{b} = \begin{pmatrix} 4.1 \\ 9.7 \end{pmatrix} , \mathbf{x} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} .$$

Normen: 1-Norm $\|\cdot\|_1$ ➢ Maxtrixnorm = Spaltensummennorm (3.1.2)

Störung: $\Delta\mathbf{b} = \begin{pmatrix} 0.01 \\ 0 \end{pmatrix}$, $\|\Delta\mathbf{b}\|_1 = 0.01$ $\blacktriangleright$ $\Delta\mathbf{x} = \begin{pmatrix} -0.66 \\ -0.97 \end{pmatrix}$, $\|\Delta\mathbf{x}\|_1 = 1.63$ („gross")

$$\mathbf{A}^{-1} = \begin{pmatrix} -66 & 28 \\ 97 & 41 \end{pmatrix} \quad \Rightarrow \quad \mathrm{cond}_1(\mathbf{A}) = \|\mathbf{A}\|_1 \cdot \left\|\mathbf{A}^{-1}\right\|_1 = 13.8 \cdot 163 = 2249.4 \ .$$

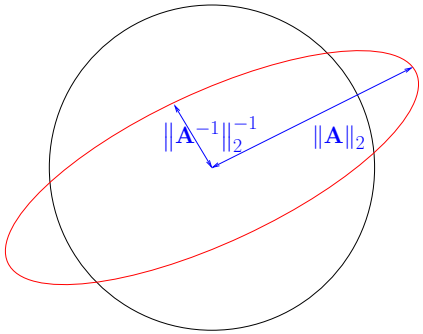
$$\Updownarrow$$

$$\frac{\|\Delta \mathbf{x}\|}{\|\mathbf{x}\|} : \frac{\|\Delta \mathbf{b}\|}{\|\Delta \mathbf{x}\|} = 1.63 : 0.0007246 = 2249.4 \ .$$

Kondition bzgl. Euklidischer Norm:

$$\mathrm{cond}(A) = \frac{\max\limits_{\|\mathbf{x}\|=1} \|\mathbf{A}\mathbf{x}\|}{\min\limits_{\|\mathbf{x}\|=1} \|\mathbf{A}\mathbf{x}\|}$$

Für $\|\cdot\| = \|\cdot\|_2$, $n = 2$: Verhältnis von maximalem zu minimalem Abstand des Bildes des Einheitskreises zum Ursprung. →



MATLAB-Funktionen zur Konditionsberechnung/-abschätzung:

- `cond(A)` → $\mathrm{cond}_2(A)$
- `cond(A,1)` und `cond(A,inf)` → $\mathrm{cond}_1(A)$ und $\mathrm{cond}_\infty(A)$
- `condest(A)` schätzt $\mathrm{cond}_1(A)$, geeignet für grosse Matrizen

3.2.2 Die Gausselimination

GAUSSELIM

! Ausnahmestellung linearer Gleichungssysteme (LGS):

☞ „exakt" lösbar mit endlich vielen arithmetischen Grundoperationen

Standardtechnik: Gausselimination (→ Sekundarstufe, Kurs „Lineare Algebra")

Idee: Transformation auf „einfachere" äquivalente Gleichungssysteme durch sukzessive Zeilenumformungen (evtl. auch Spaltenumformungen)

Beispiel 68 (Gausselimination).

$$\begin{pmatrix} 1 & 1 & 0 \\ 2 & 1 & -1 \\ 3 & -1 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 4 \\ 1 \\ -3 \end{pmatrix} \quad \longleftrightarrow \quad \begin{array}{rcl} x_1 + x_2 & = & 4 \\ 2x_1 + x_2 - x_3 & = & 1 \\ 3x_1 - x_2 - x_3 & = & -3 \end{array} .$$

$$\begin{pmatrix} 1 & & \\ & 1 & \\ & & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 0 \\ 2 & 1 & -1 \\ 3 & -1 & -1 \end{pmatrix} \begin{pmatrix} 4 \\ 1 \\ -3 \end{pmatrix} \rightarrowtail \begin{pmatrix} 1 & & \\ 2 & 1 & \\ & 0 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{1} & \mathbf{1} & \mathbf{0} \\ 0 & -1 & -1 \\ 3 & -1 & -1 \end{pmatrix} \begin{pmatrix} 4 \\ -7 \\ -3 \end{pmatrix} \rightarrowtail$$

$$\begin{pmatrix} 1 & & \\ 2 & 1 & \\ 3 & 0 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{1} & \mathbf{1} & \mathbf{0} \\ 0 & -1 & -1 \\ 0 & -4 & -1 \end{pmatrix} \begin{pmatrix} 4 \\ -7 \\ -15 \end{pmatrix} \rightarrowtail \underbrace{\begin{pmatrix} 1 & & \\ 2 & 1 & \\ 3 & 4 & 1 \end{pmatrix}}_{=L} \underbrace{\begin{pmatrix} 1 & 1 & 0 \\ 0 & \mathbf{-1} & -1 \\ 0 & 0 & 3 \end{pmatrix}}_{=U} \begin{pmatrix} 4 \\ -7 \\ 13 \end{pmatrix}$$

PIVOTELEM

■ = Pivotzeile, Pivotelement **fett**.

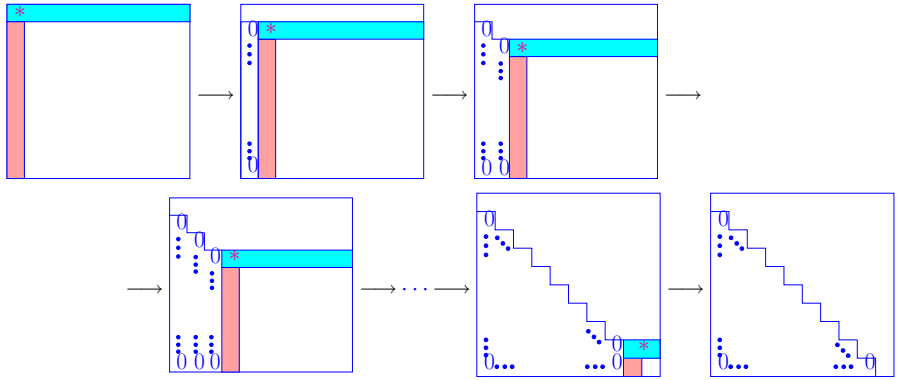
OBDREIECKSFORM

▻ Transformation auf obere Dreiecksform

RUECKSUBST

Anschliessend: Lösung durch Rücksubstitution:

$$\begin{array}{rcl} x_1 + x_2 & = & 4 \\ - x_2 - x_3 & = & -7 \\ 3x_3 & = & 13 \end{array} \quad \Rightarrow \quad \begin{array}{l} x_3 = \frac{13}{3} \\ x_2 = 7 - \frac{13}{3} = \frac{8}{3} \\ x_1 = 4 - \frac{8}{3} = \frac{4}{3} \ . \end{array}$$



$*$ ≙ Pivotelement (notwendig $\neq 0$), ■ = Pivotzeile

Im k. Schritt ($\mathbf{A} \in \mathbb{K}^{n,n}$, $1 \le k < n$, Pivotzeile $\mathbf{a}_k^T$): $\mathbf{A}\mathbf{x} = \mathbf{b}$ ➤ $\mathbf{A}'\mathbf{x} = \mathbf{b}'$ mit

$$a'_{ij} := \begin{cases} a_{ij} - \frac{a_{ik}}{a_{kk}} a_{kj} & \text{für } k < i, j \le n \ , \\ 0 & \text{für } k < i \le n,, j = k \ , \\ a_{ij} & \text{sonst,} \end{cases} \quad b'_i := \begin{cases} b_i - \frac{a_{ik}}{a_{kk}} b_k & \text{für } k < i \le n \ , \\ b_i & \text{sonst.} \end{cases} \quad (3.2.4)$$

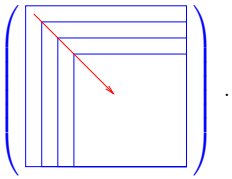
eq:GE

Block-Perspektive (erster Eliminationsschritt mit Pivotelement $\alpha \neq 0$):

$$\mathbf{A} := \left(\begin{array}{c|c} \alpha & \mathbf{c}^T \\ \hline \mathbf{d} & \mathbf{C} \end{array}\right) \rightarrow \mathbf{A}' := \left(\begin{array}{c|c} \alpha & \mathbf{c}^T \\ \hline \mathbf{0} & \mathbf{C}' := \mathbf{C} - \frac{\mathbf{d}\mathbf{c}^T}{\alpha} \end{array}\right) . \quad (3.2.5)$$

u:block

Sukzessive Durchführung:



Rechte Seite $\mathbf{b} \sim$ `A(:,end)` ➤

Demonstrations- MATLAB-CODE: Rekursive Gausselimination

```
function A = blockgs(A)
% Recursive Gaussian elimination, no pivoting
n = size(A,1);
if (n ~= size(A,2))
  error('Size mismatch!'); end
if (n > 1)
  C = blockgs(A(2:end,2:end)-...
      A(2:end,1)*A(1,2:end)/A(1,1));
  A = [A(1,:) ; zeros(n-1,1), C ];
end
```


Rechenaufwand (Anzahl elementarer Operationen) für die Gauss-Elimination:

$$\begin{aligned} \text{Elimination :} \quad & \sum_{i=1}^{n-1} (n-i-1)(n-i+1) = \tfrac{1}{3}n^3 - \tfrac{5}{6}n + \frac{1}{2}n^2 + 1 , \\ \text{Rücksubstitution :} \quad & \sum_{i=1}^{n} (n-i+1) = \tfrac{1}{2}n(n+1) . \end{aligned} \quad (3.2.6)$$

eq:GEco

Rechenaufwand für Gauss-Elimination (ohne Pivotsuche) für generisches LGS $\mathbf{A}\mathbf{x} = \mathbf{b}$, $\mathbf{A} \in \mathbb{R}^{n,n}$ = $\frac{1}{3}n^3 + O(n^2)$

Programmiere niemals eine Gausselimination selbst !

Benutze Programmbibliotheken oder MATLAB (MATLAB operator: \) !

3.2.3 Die LU-Zerlegung

LUZERL MATFAKTOR

Interpretation der Gausselimination als Matrixfaktorisierung → Bsp. 68 :

ex:GE

(Zeilenumformung = Multiplikation mit Eliminationsmatrix)

$$a_1 \neq 0 \quad \blacktriangleright \quad \begin{pmatrix} 1 & 0 & \cdots & \cdots & 0 \\ -\frac{a_2}{a_1} & 1 & & & 0 \\ -\frac{a_3}{a_1} & & \ddots & & \\ \vdots & & & & \\ -\frac{a_n}{a_1} & 0 & & & 1 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} a_1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} .$$

▶ Nach $n-1$ Schritten der Gausselimination: ➤ Matrixfaktorisierung (→ Bsp. 68)

ex:GE

$$\mathbf{A} = \mathbf{L}_1 \cdot \cdots \cdot \mathbf{L}_{n-1}\mathbf{U} \quad \text{mit} \quad \begin{array}{l} \text{Eliminationsmatrizen } \mathbf{L}_i,\ i = 1,\ldots,n-1 , \\ \text{oberer Dreiecksmatrix } \mathbf{U} \in \mathbb{R}^{n,n} . \end{array}$$

$$\begin{pmatrix} 1 & 0 & \cdots & \cdots & 0 \\ l_2 & 1 & & & 0 \\ l_3 & & \ddots & & \\ \vdots & & & & \\ l_n & 0 & & & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & \cdots & \cdots & 0 \\ 0 & 1 & & & 0 \\ 0 & h_3 & 1 & & \\ \vdots & \vdots & & \ddots & \\ 0 & h_n & 0 & & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & \cdots & \cdots & 0 \\ l_2 & 1 & & & 0 \\ l_3 & h_3 & 1 & & \\ \vdots & \vdots & & \ddots & \\ l_n & h_n & 0 & & 1 \end{pmatrix}$$

▶ $\mathbf{L}_1 \cdot \cdots \cdot \mathbf{L}_{n-1}$ sind normalisierte untere Dreiecksmatrizen

(Einträge = Faktoren $-\frac{a_{ik}}{a_{kk}}$ aus (3.2.4) → Bsp. 68)

eq:GE ex:GE

Definition 3.2.6 (Matrizentypen). *Matrix $\mathbf{A} = (a_{ij}) \in \mathbb{R}^{m,n}$ ist*

DIAGMAT

- *Diagonalmatrix, wenn $a_{ij} = 0$ für $i \neq j$,*

TRIU

- *obere Dreiecksmatrix (engl. upper triangular matrix), falls $a_{ij} = 0$ für $i > j$,*

TRIL

- *untere Dreiecksmatrix (engl. lower triangular matrix), falls $a_{ij} = 0$ für $i < j$.*

Eine Dreiecksmatrix heisst normalisiert, wenn $a_{ii} = 1$, $i = 1,\ldots,\min\{m,n\}$.

Die Vorwärtsrechnung in der Gausselimination für $\mathbf{A}\mathbf{x} = \mathbf{b}$, $\mathbf{A} \in \mathbb{R}^{n,n}$ ist algebraisch **und numerisch** (d.h. mit Berücksichtigung von Rundungsfehlern) äquivalent zur LU-Zerlegung (engl. *LU decomposition*) $\mathbf{A} = \mathbf{L}\mathbf{U}$ in eine normalisierte untere Dreiecksmatrix $\mathbf{L}$ und eine obere Dreiecksmatrix $\mathbf{U}$.

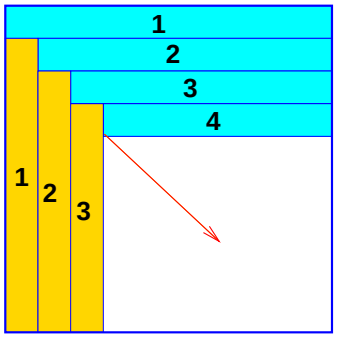
Eindeutigkeit der LU-Zerlegung:

Reguläre obere Dreiecksmatrizen/normalisierte untere Dreiecksmatrizen bilden eine Gruppe bzgl. Matrixmultiplikation. Der Schnitt beider Matrixmengen enthält nur die Einheitsmatrix.

$$\mathbf{L}_1\mathbf{U}_1 = \mathbf{L}_2\mathbf{U}_2 \quad \Rightarrow \quad \mathbf{L}_2^{-1}\mathbf{L}_1 = \mathbf{U}_2\mathbf{U}_1^{-1} = \mathbf{I} .$$

Direkte Herleitung der LU-Zerlegung

$$\mathbf{LU} = \mathbf{A} \quad \Rightarrow \quad a_{ik} = \sum_{j=1}^{\min\{i,k\}} l_{ij}u_{jk} = \begin{cases} \sum_{j=1}^{i-1} l_{ij}u_{jk} + 1 \cdot u_{ik} & \text{, falls } i \le k \text{ ,} \\ \sum_{j=1}^{k-1} l_{ij}u_{jk} + l_{ik}u_{kk} & \text{, falls } i > k \text{ .} \end{cases}$$



➤ • Zeilenweise Berechnung von $\mathbf{U}$

• Spaltenweise Berechnung von $\mathbf{L}$

Einträge von $\mathbf{A}$ können überschrieben werden.
(Bezeichnung: Rechnung in situ)

CROUT
(Algorithmus von Crout)

Algorithmus 69.
INSITULU
In situ LU-Zerlegung von $\mathbf{A} \in \mathbb{R}^{n,n}$:
(ohne Pivotsuche)

$\mathbf{A}$ wird mit $\mathbf{L}$,$\mathbf{U}$ überschrieben:

```
B = lurec(A);
L = tril(B,-1)+eye(size(A));
U = triu(B);
```

MATLAB-CODE: Rekursive LU-Zerlegung

```
function A = lurec(A)
%  Recursive LU factorization
if (size(A,1) > 1)
  fac = A(2:end,1)/A(1,1);
  C = lurec(A(2:end,2:end)-...
            fac*A(1,2:end));
  A = [A(1,:) ; fac , C ];
end
```

RECHAUFWLU
Rechenaufwand für LU-Zerlegung von $\mathbf{A} \in \mathbb{R}^{n,n}$ = $\frac{1}{3}n^3 + O(n^2)$ (3.2.7) tab:lucos

---

Lösen eines linearen Gleichungssystems via LU-Zerlegung:

$\mathbf{Ax} = \mathbf{b}$:
① LU-Zerlegung $\mathbf{A} = \mathbf{LU}$, Rechenaufwand $\frac{1}{3}n(n-1)(n+1)$
FORWARDSUBS
② Vorwärtssubstitution, löse $\mathbf{Lz} = \mathbf{b}$, Rechenaufwand $\frac{1}{2}n(n-1)$
BACKWARDSUBS
③ Rücksubstitution, löse $\mathbf{Ux} = \mathbf{z}$, Rechenaufwand $\frac{1}{2}n(n+1)$

➤ Äquivalent zur Gausselimination

➤ Aufwand zur Lösung von $\mathbf{Ax}_k = \mathbf{b}_k$, $\mathbf{b}_k \in \mathbb{R}^n$, $k = 1, \ldots, m$, $m \in \mathbb{N}$:

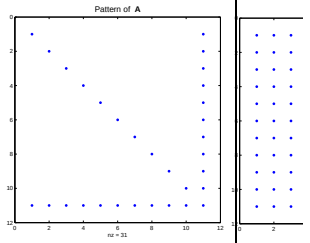
1× LU-Zerlegung + m× Vorwärts- & Rückwärtssubstitution = $\frac{1}{3}n(n-1)(n+1) + mn^2$

☞ Für $m \gg 1$ wesentlich „billiger” als m-fache Gausselimination !

BESETZMUSTER
Beispiel 70 (Besetzungsmuster der LU-Faktoren).

```
A = [diag(1:10),ones(10,1);ones(1,10),2];
```

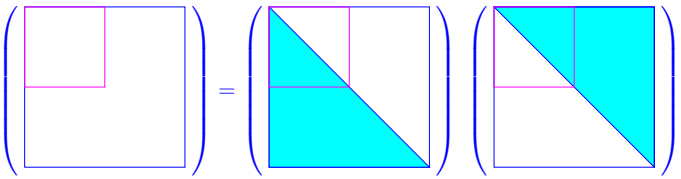
```
[L,U] = lu(A); spy(A); spy(L); spy(U); spy(inv(A));
```



Berechung $\mathbf{A}^{-1} = \mathbf{U}^{-1}\mathbf{L}^{-1}$ kann extrem unökonomisch sein !

◇

Bemerkung 71. „Teil-LU-Zerlegungen” in linken oberen Blöcken:



Bemerkung 72. Blockweise Matrixmultiplikation (3.1.1) ≅ Block-LU-Zerlegung:

eq:Blockmul

Mit $\mathbf{A}_{11} \in \mathbb{K}^{n,n}$ regulär, $\mathbf{A}_{12} \in \mathbb{K}^{n,m}$, $\mathbf{A}_{21} \in \mathbb{K}^{m,n}$, $\mathbf{A}_{22} \in \mathbb{K}^{m,m}$:

SCHURKOMP
Schur-Komplement

$$\begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{pmatrix} = \begin{pmatrix} \mathbf{I} & 0 \\ \mathbf{A}_{21}\mathbf{A}_{11}^{-1} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ 0 & \mathbf{S} \end{pmatrix} \quad , \quad \mathbf{S} := \mathbf{A}_{22} - \mathbf{A}_{21}\mathbf{A}_{11}^{-1}\mathbf{A}_{12} \qquad (3.2.8)$$

eq:Block

3.2.4 Pivotsuche

PIVOTSUCHE

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \qquad \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} b_2 \\ b_1 \end{pmatrix}$$

Zusammenbruch der Gauss-Elimination
Pivotelement $= 0$

Gauss-Elimination funktioniert

Idee: Vermeide sehr kleine Pivotelemente durch Zeilenvertauschungen

Terminologie: Pivotsuche/Pivotstrategie (engl.*pivoting*) = Wahl eines günstigen Pivotelements

Beispiel 73 (Pivotstrategie und Rundungsfehler).

$$\mathbf{A} = \begin{pmatrix} \epsilon & 1 \\ 1 & 1 \end{pmatrix} \quad 0 < \epsilon < \texttt{eps}\,, \quad \mathbf{b} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \quad \Rightarrow \quad \mathbf{x} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Einfache LU-Zerlegung:

$$\mathbf{L} = \begin{pmatrix} 1 & 0 \\ \epsilon^{-1} & 1 \end{pmatrix}\,, \quad \mathbf{U} = \begin{pmatrix} \epsilon & 1 \\ 0 & 1-\epsilon^{-1} \end{pmatrix} = \widetilde{\mathbf{U}} := \begin{pmatrix} \epsilon & 1 \\ 0 & -\epsilon^{-1} \end{pmatrix} \quad \text{in } \mathbb{M}\,! \qquad (3.2.9)$$ lu:1

Lösung von $\mathbf{L}\widetilde{\mathbf{U}}\mathbf{x} = \mathbf{b}$: $\mathbf{x} = \begin{pmatrix} 1-\epsilon \\ 1 \end{pmatrix}$ (unbrauchbares Ergebnis !)

LU-Zerlegung nach Zeilenvertauschung:

$$\mathbf{A} = \begin{pmatrix} 1 & 1 \\ \epsilon & 1 \end{pmatrix} \quad \Rightarrow \quad \mathbf{L} = \begin{pmatrix} 1 & 0 \\ \epsilon & 1 \end{pmatrix}\,, \quad \mathbf{U} = \begin{pmatrix} 1 & 1 \\ 0 & 1-\epsilon \end{pmatrix} = \widetilde{\mathbf{U}} := \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \qquad (3.2.10)$$ lu:2

Lösung von $\mathbf{L}\widetilde{\mathbf{U}}\mathbf{x} = \mathbf{b}$: $\mathbf{x} = \begin{pmatrix} \epsilon \\ 1-\epsilon \end{pmatrix}$ (brauchbares Ergebnis !)

Ohne Zeilenvertauschung, → (3.2.9): $\mathbf{L}\widetilde{\mathbf{U}} = \mathbf{A} + \mathbf{E}$ mit $\mathbf{E} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$ Instabil !

Nach Zeilenvertauschung, → (3.2.10): $\mathbf{L}\widetilde{\mathbf{U}} = \widetilde{\mathbf{A}} + \mathbf{E}$ mit $\mathbf{E} = \begin{pmatrix} 0 & 0 \\ 0 & \epsilon \end{pmatrix}$ Stabil !

Geeignete Pivotstrategie entscheidend für numerische Stabilität der Gausselimination

Heuristik: „Explosion" von Einträgen $\mathbf{L}, \mathbf{U}$ ➤ Auslöschung bei Vorwärts- & Rückwärtssubstitution.

Algorithmus 74.

Rekursive in situ LU-Zerlegung von $\mathbf{A} \in \mathbb{R}^{n,n}$ mit Spaltenpivotsuche SPALTENPIVOT ➤

Wahl des Pivotindex j:
$j \in \{i, \ldots, n\}$ so, dass

$$\frac{|a_{ki}|}{\sum_{l=i}^{n} |a_{kl}|} \to \max \qquad (3.2.11)$$ LUPS

für $k = j$, $k \in \{i, \ldots, n\}$.
(relativ grösstes Pivotelement)

Demonstrations- MATLAB-CODE: Rekursive Gausselimination mit Spaltenpivotsuche

```
function A = gsrecpiv(A)
n = size(A,1);
if (n > 1)
 [p,j]=max(abs(A(:,1))./sum(abs(A)')');
 if (p < eps*norm(A(:,1:n),1))
  error('Nearly Singular matrix'); end
 A([1,j],:) = A([j,1],:);
 C = gsrecpiv(A(2:end,2:end)-...
          A(2:end,1)*A(1,2:end)/A(1,1));
 A = [A(1,:) ; zeros(n-1,1), C ];
end
```

Warum relativ grösstes Pivotelement in (3.2.11)? Skalieren des Gleichungssystems aus Bsp. 73: lu:PS ex:pivot

$$\begin{pmatrix} 2/\epsilon & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \epsilon & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 2 & 2/\epsilon \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 2/\epsilon \\ 1 \end{pmatrix}$$

Keine Zeilenvertauschung, falls absolut grösstes Pivotelement genommen wird.

$$\begin{pmatrix} 2 & 2/\epsilon \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 2 & 2/\epsilon \\ 0 & 1-2/\epsilon \end{pmatrix} \doteq \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 2 & 2/\epsilon \\ 0 & -2/\epsilon \end{pmatrix} \quad \text{in } \mathbb{M}\,.$$

Konstruktiv bewiesen durch Algorithmus 74. alg:LUp

Lemma 3.2.7 (LU-Zerlegung mit Pivotsuche)**.** *Zu jeder regulären Matrix $\mathbf{A} \in \mathbb{R}^{n,n}$ existieren eine Zeilenpermutation P, eine normalisierte untere Dreiecksmatrix $\mathbf{L} \in \mathbb{R}^{n,n}$ und eine obere Dreiecksmatrix $\mathbf{U} \in \mathbb{R}^{n,n}$ (→ Def. 3.2.6) so dass* def:LU

$$P\mathbf{A} = \mathbf{LU}\,.$$

MATLAB-Funktion: `[L,U,P] = lu(A)` (P = Permutationsmatrix)

Gauss-Elimination mit Spaltenpivotsuche gemäss (3.2.11) ist „meistens" numerisch stabil. lu:PS

Beispiel 75 (Wilkinsons Gegenbeispiel).

n=10:

$$a_{ij} = \begin{cases} 1 & \text{, falls } i = j \vee j = n \text{ ,} \\ -1 & \text{, falls } i > j \text{ ,} \\ 0 & \text{sonst} \end{cases} \text{ , } \quad \mathbf{A} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ -1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ -1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ -1 & -1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & 1 \\ -1 & -1 & -1 & -1 & -1 & 1 & 0 & 0 & 0 & 1 \\ -1 & -1 & -1 & -1 & -1 & -1 & 1 & 0 & 0 & 1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & 0 & 1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 \end{pmatrix}$$

Spaltenpivotsuche führt zu keinen Zeilenvertauschungen !

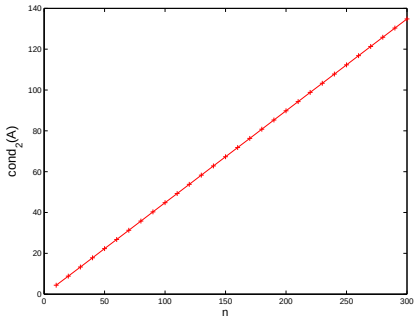
► $\mathbf{A} = \mathbf{LU}$, $l_{ij} = \begin{cases} 1 & \text{, falls } i = j \text{ ,} \\ -1 & \text{, falls } i > j \text{ ,} \\ 0 & \text{sonst} \end{cases}$ $u_{ij} = \begin{cases} 1 & \text{, falls } i = j \text{ ,} \\ 2^{i-1} & \text{, falls } j = n \text{ ,} \\ 0 & \text{sonst} \end{cases}$

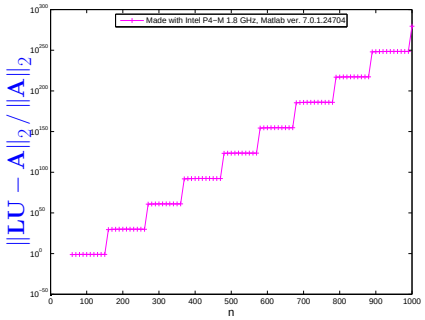
Bsp. Stabilität der LU-Zerlegung **?**

Ist $\mathbf{A} - \widetilde{\mathbf{L}}\widetilde{\mathbf{U}}$ eine kleine relative Störung von $\mathbf{A}$ **?**

MATLAB-CODE Stabilität der LU-Zerlegung

```
res = [];
for n=10:10:300
  A = [tril(-ones(n,n-1))+2*[eye(n-1);...
       zeros(1,n-1)],ones(n,1)];
  [L,U,P] = lu(A);
  E = L*U-P*A;
  res = [res; n,cond(A),norm(E)/norm(A)];
end
```





TOTALPIV

Bemerkung 76. Totalpivotsuche verwendet Zeilen- **und** Spaltenvertauschungen: verwende a_{lj} mit $|a_{lj}| \geq |a_{mk}|, i \leq m, k \leq n$ als Pivotelement im i. Schritt (bei in situ Rechnung !).

► Gauss-Elimination mit Totalpivotsuche numerisch stabil (aber meist zu teuer!)

PFEILMAT

Beispiel 77 („Pfeilmatrix").

$$\mathbf{A} = \left(\begin{array}{c|c} \alpha & \mathbf{b}^T \\ \hline \mathbf{c} & \mathbf{D} \end{array}\right) \text{ , } \quad \begin{array}{l} \alpha \in \mathbb{R} \text{ ,} \\ \mathbf{b}, \mathbf{c} \in \mathbb{R}^{n-1} \text{ ,} \\ \mathbf{D} \in \mathbb{R}^{n-1,n-1} \text{ reguläre Diagonalmatrix, } \rightarrow \text{ Def. 3.2.6} \end{array}$$

def:LU

(3.2.12) A arrow

► $\mathbf{A}$ ist Beispiel für dünnbesetzte Matrix (→ Def. 3.1.1), da

def:sparse

$$\sharp\{(i,j) \in \{1,\ldots,n\}^2 \colon a_{ij} \neq 0\} \ll n^2 \text{ .}$$

Anwendung von Algorithmus 74:

alg:LUp

ANWLUP

Faktormatrizen mit $O(n^2)$ Einträgen.

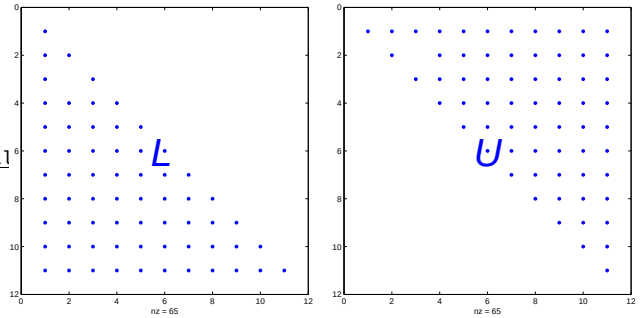
Rechenaufwand $O(n^3)$

MATLAB-CODE LU-Zerlegung von Pfeilmatrix

```
A = [ 1, ones(1,10); ...
    ones(10,1), eye(10,10)];
[L,U,P] = lu(A);
spy(L); spy(U);
```


Auffüllen der LU-Faktoren (engl. *Fill-in*) → Def. 3.2.14

def:fill



Rotiere Zeilen/Spalten:

- 1. Zeile/Spalte → n. Zeile/Spalte,
- i. Zeile/Spalte → $i-1$. Zeile/Spalte, $i = 2,\ldots,n$

➢ Alg. 69 kostet $O(n)$ Rechenoperationen:

alg:LU

$$\mathbf{A} = \left(\begin{array}{c|c} \mathbf{D} & \mathbf{c} \\ \hline \mathbf{b}^T & \alpha \end{array}\right)$$

(3.2.13) u:Amod

Für permutiertes $\mathbf{A}$ aus (3.2.13), vgl. (3.2.8):

$$\mathbf{L} = \left(\begin{array}{c|c} \mathbf{I} & 0 \\ \hline \mathbf{b}^T\mathbf{D}^{-1} & 1 \end{array}\right) , \quad \mathbf{U} = \left(\begin{array}{c|c} \mathbf{D} & \mathbf{c} \\ \hline 0 & \sigma \end{array}\right) , \quad \sigma := \alpha - \mathbf{b}^T\mathbf{D}^{-1}\mathbf{c} .$$

➢ Kein Auffüllen bei Berechnung der LU-Zerlegung, Rechenaufwand $O(n)$

Lösen des LGS $\mathbf{Ax} = \mathbf{y}$ mit $\mathbf{A}$ aus 3.2.12: Zwei MATLAB-Implementierungen

"Naive" Implementierung mittels „\":

MATLAB-CODE : LGS mit Pfeilmatrix , Implementierung I

```
function x = sa1(alpha,b,c,d,y)
A = [alpha, b'; c, diag(d)];
x = A\y;
```

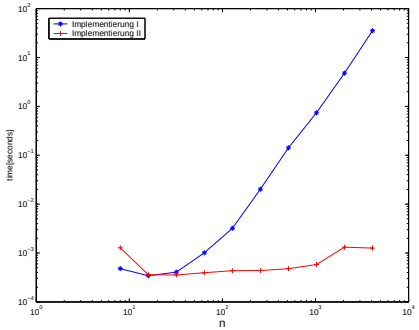
„Strukturbewusste" Implementierung:

MATLAB-CODE : LGS mit Pfeilmatrix, Implementierung II

```
function x = sa2(alpha,b,c,d,y)
z = b./d;
xi = (y(1) - dot(z,y(2:end)))...
  /(alpha-dot(z,c));
x = [xi; (y(2:end)-xi*c)./d];
```

Laufzeitmessung:

```
t = [];
for i=3:12
  n = 2^n; alpha = 2;
  b = ones(n,1); c = (1:n)';
  d = -ones(n,1); y = (-1).^(1:(n+1))';
  tic; x1 = sa1(alpha,b,c,d,y); t1 = toc;
  tic; x2 = sa2(alpha,b,c,d,y); t2 = toc;
  t = [t; n t1 t2];
end
loglog(t(:,1),t(:,2), ...
  ... 'b-*',t(:,1),t(:,3),'r-+');
```



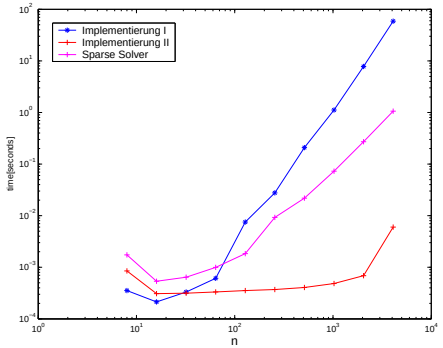
Resultat (Mobile Intel Pentium 4 - M CPU 2.40GHz, Linux MATLAB V6.5) ➤

MATLAB kann es besser **!**

Mit dünnbesetzten Matrizen:
(Abschnitt 3.1.2)

MATLAB-CODE : Sparse Löser für Pfeilmatrix

```
function x = sa3(alpha,b,c,d,y)
n = length(d);
A = [alpha, b';...
c,spdiags(d,0,n,n)];
x = A\y;
```



Nütze Struktur eines linearen Gleichungssystems aus!

Vorsicht: Stabilität der Implementierung

3.2.5 Symmetrisch positiv definite Matrizen

Definition 3.2.8 (Symmetrisch positiv definite Matrizen)**.** $\mathbf{M} \in \mathbb{K}^{n,n}$, $n \in \mathbb{N}$, *ist symmetrisch (Hermitesch) positiv definit (Abkürzung s.p.d.), falls*

$$\mathbf{M} = \mathbf{M}^H \quad \wedge \quad \mathbf{x}^H\mathbf{M}\mathbf{x} > 0 \quad \Leftrightarrow \quad \mathbf{x} \neq 0 .$$

Falls $\mathbf{x}^H\mathbf{M}\mathbf{x} \geq 0$ *für alle* $\mathbf{x} \in \mathbb{K}^n$*, so nennt man* $\mathbf{M}$ *positiv semidefinit.*

Bemerkung 78. (Lokale) Minimierung x^* von C^2-Funktion $f : \mathbb{R}^n \mapsto \mathbb{R}$ ➤ Hessematrix $D^2f(x^*)$ s.p.d.

Wie stelle ich fest, ob eine Matrix positiv definit ist **?**

Notwendige Bedingungen:

Lemma 3.2.9 (Notwendige Bedingungen für s.p.d. Matrix)**.** *Für eine symmetrisch positiv definite Matrix* $\mathbf{M} = \mathbf{M}^H \in \mathbb{K}^{n,n}$ *gilt:*

1. $m_{ii} > 0$, $i = 1, \ldots, n$,
2. $m_{ii}m_{jj} - |m_{ij}|^2 > 0 \quad \forall i, j = 1, \ldots, n$,
3. *alle Eigenwerte von* $\mathbf{M}$ *sind positiv.* *(← auch hinreichend für symmetrisches* $\mathbf{M}$*)*

Beispiel 79 (Eigenschaften von Leitwertmatrizen). Siehe Bsp. 66 ex:elnet

$\mathbf{A} \in \mathbb{R}^{n,n}$: Matrix, entstanden aus der Knotenanalyse eines linearen elektrischen Netzwerks mit ausschliesslich Ohmschen Widerständen

- $\mathbf{A} = \mathbf{A}^T \quad , \quad a_{kk} > 0 \quad , \quad a_{kj} \leq 0$ für $k \neq k$,
- $\sum_{j=1}^{n} a_{kj} \geq 0 \, , \quad k = 1, \ldots, n$.

◇

Definition 3.2.10 (Diagonaldominanz)**.** $\mathbf{A} \in \mathbb{K}^{n,n}$ *heisst diagonaldominant, falls* DIAGDOMIN

$$\forall k \in \{1, \ldots, n\}: \quad \sum_{j \neq k} |a_{kj}| \leq a_{kk} \, .$$

Lemma 3.2.11. *Eine diagonaldominante Hermitesche Matrix ist positiv* semi*definit.*

Beweis. $\mathbf{A} = \mathbf{A}^H$ diagonaldominant, mit AGM-Ungleichung

$$\begin{aligned}
\mathbf{x}^H\mathbf{A}\mathbf{x} &= \sum_{i=1}^{n} a_{ii}|x_i|^2 - \sum_{i \neq j} a_{ij}\bar{x}_i x_j \geq \sum_{i=1}^{n} a_{ii}|x_i|^2 - \sum_{i \neq j} |a_{ij}||x_i||x_j| \\
&\geq \sum_{i=1}^{n} a_{ii}|x_i|^2 - \tfrac{1}{2}\sum_{i \neq j} |a_{ij}|(|x_i|^2 + |x_j|^2) \\
&\geq \tfrac{1}{2}\Big(\sum_{i=1}^{n}\{a_{ii}|x_i|^2 - \sum_{j \neq i} |a_{ij}||x_i|^2\}\Big) + \tfrac{1}{2}\Big(\sum_{j=1}^{n}\{a_{ii}|x_j|^2 - \sum_{i \neq j} |a_{ij}||x_j|^2\}\Big) \\
&\geq \sum_{i=1}^{n} |x_i|^2\Big(a_{ii} - \sum_{j \neq i} |a_{ij}|\Big) \geq 0 \, .
\end{aligned}$$

Theorem 3.2.12 (Gauss-Elimination für s.p.d. Matrizen)**.**
Zu jeder symmetrisch positiv definite Matrix (→ Def. 3.2.8) def:spd existiert eine LU*-Zerlegung.*

Beweisskizze. Gauss-Elimination *ohne Pivotsuche* durchführbar.

Induktionsbeweis: betrachte ersten Eliminationsschritt:

$$\mathbf{A} = \begin{pmatrix} a_{11} & \mathbf{b}^T \\ \mathbf{b} & \widetilde{\mathbf{A}} \end{pmatrix} \xrightarrow[\text{Gauss-Elimination}]{\text{1. Schritt}} \begin{pmatrix} a_{11} & \mathbf{b}^T \\ 0 & \widetilde{\mathbf{A}} - \frac{\mathbf{b}\mathbf{b}^T}{a_{11}} \end{pmatrix} .$$

➢ Zu zeigen: $\widetilde{\mathbf{A}} - \frac{\mathbf{b}\mathbf{b}^T}{a_{11}}$ s.p.d. (→ Induktionsschluss)

- Klar ist Symmetrie von $\widetilde{\mathbf{A}} - \frac{\mathbf{b}\mathbf{b}^T}{a_{11}} \in \mathbb{R}^{n-1,n-1}$
- Da $\mathbf{A}$ s.p.d. (→ Def. 3.2.8) def:spd, für jedes $\boldsymbol{y} \in \mathbb{R}^{n-1} \setminus \{0\}$

$$0 < \begin{pmatrix} -\frac{\mathbf{b}^T\mathbf{y}}{a_{11}} \\ \mathbf{y} \end{pmatrix}^T \begin{pmatrix} a_{11} & \mathbf{b}^T \\ \mathbf{b} & \widetilde{\mathbf{A}} \end{pmatrix} \begin{pmatrix} -\frac{\mathbf{b}^T\mathbf{y}}{a_{11}} \\ \mathbf{y} \end{pmatrix} = \boldsymbol{y}^T(\widetilde{\mathbf{A}} - \frac{\mathbf{b}\mathbf{b}^T}{a_{11}})\boldsymbol{y} \, .$$

⯈ $\widetilde{\mathbf{A}} - \frac{\mathbf{b}\mathbf{b}^T}{a_{11}}$ positiv definit. □

Beachte: keine Pivotsuche (→ Sect. 3.2.4) sec:pivots-und-rund erforderlich
(Spaltenpivotsuche wählt immer aktuelle Zeile)

Lemma 3.2.13 (Cholesky-Zerlegung für s.p.d. Matrizen)**.** *Zu jedem s.p.d.* $\mathbf{A} \in \mathbb{R}^{n,n}$, $n \in \mathbb{N}$, *existiert eine eindeutige obere Dreiecksmatrix* $\mathbf{R} \in \mathbb{R}^{n,n}$ *mit* $r_{ii} > 0$, $i = 1, \ldots, n$, *so dass* $\mathbf{A} = \mathbf{R}^H\mathbf{R}$ *(Cholesky-Zerlegung).* CHOL

Thm. 3.2.12 thm:GEspd $\Rightarrow$ $\mathbf{A} = \mathbf{L}\mathbf{U}$ (LU-Zerlegung von $\mathbf{A}$, Sect. 3.2.3) sec:die-lr-zerlegung

$\mathbf{A} = \mathbf{L}\mathbf{D}\widetilde{\mathbf{U}}$,, $\mathbf{D} \mathrel{\hat{=}}$ Diagonale von $\mathbf{U}$,
$\widetilde{\mathbf{U}} \mathrel{\hat{=}}$ *normalisierte* obere Dreiecksmatrix → Def. 3.2.6 def:LU

Wegen Eindeutigkeit der LU-Zerlegung

$$\mathbf{A} = \mathbf{A}^T \;\Rightarrow\; \mathbf{U} = \mathbf{D}\mathbf{L}^T \;\Rightarrow\; \boxed{\mathbf{A} = \mathbf{L}\mathbf{D}\mathbf{L}^T} \, ,$$

mit eindeutigen $\mathbf{L}$, $\mathbf{D}$ (Diagonalmatrix).

$$\mathbf{x}^T\mathbf{A}\mathbf{x} > 0 \quad \forall \mathbf{x} \neq 0 \;\Rightarrow\; \mathbf{y}^T\mathbf{D}\mathbf{y} > 0 \quad \forall \mathbf{y} \neq 0 \, .$$

➤ D hat positive Diagonaleinträge ➡ $\mathbf{R} = \sqrt{\mathbf{D}}\mathbf{L}^T$. □

MATLAB-Kommando: `R = chol(A)`

RECHAUFWCHOL
Rechenaufwand für Cholesky-Zerlegung: $\frac{1}{6}n^3 + O(n^2)$ ➤ Halb so teuer wie Alg. 69 alg:LU

Lösen eines LGS mit s.p.d. Koeffizientenmatrix via Cholesky-Zerlegung, Vorwärts- und Rückwärtssubstitution ist numerisch stabil.

Bemerkung 80 (LU-Zerlegung diagonaldominanter Matrizen).

$$\mathbf{A} \text{ regulär, diagonaldominant} \Leftrightarrow \begin{cases} \mathbf{A} \text{ LU-zerlegbar} \\ \Updownarrow \\ \text{Gauss-Elimination ohne Pivotsuche durchführbar} \end{cases}$$

Beweis. (3.2.5) lu:block → Induktion nach n: Nach 1. Eliminationsschritt:

$$a_{ij}^{(1)} = a_{ij} - \frac{a_{i1}}{a_{11}} a_{1j}\,, \quad i,j = 2,\ldots,n \quad \Rightarrow \quad a_{ii}^{(1)} > 0\,.$$

➤
$$|a_{ii}^{(1)}| - \sum_{\substack{j=2\\ j\neq i}}^{n} |a_{ij}^{(1)}| = \left|a_{ii} - \frac{a_{i1}}{a_{11}} a_{1i}\right| - \sum_{\substack{j=2\\ j\neq i}}^{n} \left|a_{ij} - \frac{a_{i1}}{a_{11}} a_{1j}\right|$$

$$\geq a_{ii} - \frac{|a_{i1}||a_{1i}|}{a_{11}} - \sum_{\substack{j=2\\ j\neq i}}^{n} |a_{ij}| - \frac{|a_{i1}|}{a_{11}} \sum_{\substack{j=2\\ j\neq i}}^{n} |a_{1j}|$$

$$\geq a_{ii} - \frac{|a_{i1}||a_{1i}|}{a_{11}} - \sum_{\substack{j=2\\ j\neq i}}^{n} |a_{ij}| - |a_{i1}| \frac{a_{11} - |a_{1i}|}{a_{11}} \geq a_{ii} - \sum_{\substack{j=1\\ j\neq i}}^{n} |a_{ij}| \geq 0\,.$$

$\mathbf{A}$ lu:PS 3.2.11 i i

△

Bemerkung 81 (Übermitteln von Matrixeigenschaften an MATLAB).

MATLAB-$\backslash$ kann einer allgemeinen (vollbesetzten) Matrix spezielle Eigenschaften (symmetrisch, s.p.d., Dreiecksmatrix) nicht „ansehen”.

➤ Benutze `y = linsolve(A,b,opts)`

`opts` ∈ {
- `LT` ↔ A untere Dreiecksmatrix
- `UT` ↔ A obere Dreiecksmatrix
- `UHESS` ↔ A obere Hessenbergmatrix
- `SYM` ↔ A Hermitesche Matrix
- `POSDEF` ↔ A positiv definite Matrix }

△

3.2.6 Dünnbesetzte Gleichungssysteme

DUENNBESGLSYS
= LGS $\mathbf{Ax} = \mathbf{b}$ mit dünnbesetzter Koeffizientenmatrix $\mathbf{A}$, → Abschnitt sec:matr-def-sparse 3.1.2, Def. 3.1.1

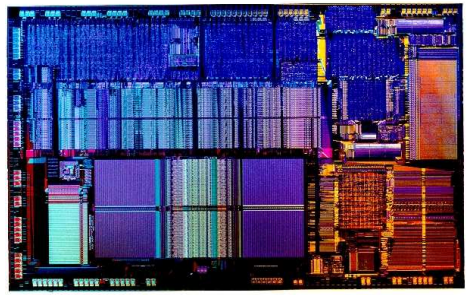
Beispiel 82 (Dünnbesetzte Gleichungssysteme in der Schaltkreissimulation).

Moderne Schaltkreise (Chips):
$10^5 - 10^7$ Elemente

- Jedes Element mit nur wenigen Knoten verbunden
- Jeder Knoten mit nur wenigen Elementen verbunden

[Im Fall eines linearen Netzwerks]

Knotenanalyse ➤ dünnbesetzte Schaltkreismatrix



◇

LUFAKTDUENNMAT
Beispiel 83 (LU-Faktorisierung dünnbesetzter Matrizen). → Bsp. ex:arrowmatrix 7.7

$$\mathbf{A} = \left(\begin{array}{cccc|cccc} 3 & -1 & & & -1 & & & \\ -1 & \ddots & \ddots & & & \ddots & & \\ & \ddots & \ddots & -1 & -1 & & \ddots & \\ & & -1 & 3 & & -1 & & -1 \\ \hline -1 & & & & 3 & -1 & & \\ & \ddots & & & -1 & \ddots & \ddots & \\ & & \ddots & & & \ddots & \ddots & -1 \\ & & & -1 & & & -1 & 3 \end{array}\right) \in \mathbb{R}^{n,n}, n \in 2\mathbb{N}$$

MATLAB `lu`-Funktion ergibt (für $n = 10$):

$$\mathbf{A} = \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0.33 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -0.38 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -0.38 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -0.38 & 1 & 0 & 0 & 0 & 0 & 0 \\ -0.33 & -0.13 & -0.05 & -0.02 & -0.01 & 1 & 0 & 0 & 0 & 0 \\ 0 & -0.38 & -0.14 & -0.05 & -0.02 & -0.44 & 1 & 0 & 0 & 0 \\ 0 & 0 & -0.38 & -0.15 & -0.06 & -0.02 & -0.58 & 1 & 0 & 0 \\ 0 & 0 & 0 & -0.38 & -0.15 & -0.01 & -0.03 & -0.65 & 1 & 0 \\ 0 & 0 & 0 & 0 & -0.38 & -0.003 & -0.01 & -0.04 & -0.67 & 1 \end{pmatrix}}_{\mathbf{L}} \underbrace{\begin{pmatrix} 3 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 2.67 & -1 & 0 & 0 & -0.33 & -1 & 0 & 0 & 0 \\ 0 & 0 & 2.63 & -1 & 0 & -0.13 & -0.38 & -1 & 0 & 0 \\ 0 & 0 & 0 & 2.62 & -1 & -0.05 & -0.14 & -0.38 & -1 & 0 \\ 0 & 0 & 0 & 0 & 2.62 & -0.02 & -0.05 & -0.15 & -0.38 & -1 \\ 0 & 0 & 0 & 0 & 0 & 2.62 & -1.15 & -0.06 & -0.02 & -0.01 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2.06 & -1.19 & -0.07 & -0.02 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.87 & -1.21 & -0.07 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.78 & -1.19 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.82 \end{pmatrix}}_{\mathbf{U}} \quad \diamond$$

$\mathbf{A}$ dünnbesetzt $\not\Rightarrow$ LU-Faktoren dünnbesetzt (siehe Bsp. 77) ex:arrowmatrix

Definition 3.2.14 (Fill-in). *Sei $\mathbf{A} = \mathbf{LU}$ eine LU-Faktorisierung von $\mathbf{A} \in \mathbb{K}^{n,n}$. Wenn $l_{ij} \neq 0$ oder $u_{ij} \neq 0$ obwohl $a_{ij} = 0$, dann handelt es sich um Fill-in an Piosition (i,j).* FILLIN FILLIN

Definition 3.2.15 (Bandbreite). *Für $\mathbf{A} = (a_{ij})_{i,j} \in \mathbb{K}^{m,n}$ heisst* BANDBREITE

$$\overline{m}(\mathbf{A}) := \min\{k \in \mathbb{N}: j - i > k \Rightarrow a_{ij} = 0\} \text{ Obere Bandbreite (engl. upper bandwidth)},$$
$$\underline{m}(\mathbf{A}) := \min\{k \in \mathbb{N}: i - j > k \Rightarrow a_{ij} = 0\} \text{ Untere Bandbreite (engl. lower bandwidth)}.$$
$$m(\mathbf{A}) := \overline{m}(\mathbf{A}) + \underline{m}(\mathbf{A}) + 1 = \text{Bandbreite von } \mathbf{A}. \quad \text{BANDWIDTH}$$

- $m(\mathbf{A}) = 0 \quad \triangleright \quad \mathbf{A}$ Diagonalmatrix, → Def. 3.2.6 def:LU
- $\overline{m}(\mathbf{A}) = \underline{m}(\mathbf{A}) = 1 \quad \triangleright \quad \mathbf{A}$ Tridiagonalmatrix TRIDIAG

MATLAB-Funktion zum Erzeugen von Bandmatrizen:

Vollbesetzte Matrizen	:	`X=diag(v);`
Dünnbesetzte Matrizen	:	`X=spdiags(B,d,m,n);`
Tridiagonalmatrizen	:	`X=gallery('tridiag',c,d,e);` (→ sparse)

Definition 3.2.16 (Strukturymmetrie). STRUKSYM
$\mathbf{A} \in \mathbb{K}^{n,n}$ *ist strukturymmetrisch (engl.* structurally symmetric*), falls*

$$\forall i,j \in \{1,\dots,n\}: \quad a_{ij} \neq 0 \quad \Leftrightarrow \quad a_{ji} \neq 0 \,.$$

Definition 3.2.17 (Hülle einer Matrix). *Für strukturymmetrisches, reguläres $\mathbf{A} \in \mathbb{K}^{n,n}$ definiere* ZEILENBANDBR

Zeilenbandreite $\quad m_i(\mathbf{A}) := \max\{0, j - i : a_{ij} \neq 0, 1 \leq j \leq n\}, i \in \{1,...,n\}$

ENV

Hülle (engl. envelope*)* $\quad \mathrm{env}(\mathbf{A}) := \{(i,j),(j,i) : i - m_i(A) \leq j \leq i; 1 \leq i \leq n\}$

Beispiel 84 (Hülle einer Matrix).

$$\mathbf{A} = \begin{pmatrix} * & 0 & * & 0 & 0 & 0 & 0 \\ 0 & * & 0 & 0 & * & 0 & 0 \\ * & 0 & * & 0 & 0 & 0 & * \\ 0 & 0 & 0 & * & * & 0 & * \\ 0 & * & 0 & * & * & * & 0 \\ 0 & 0 & 0 & 0 & * & * & 0 \\ 0 & 0 & * & * & 0 & 0 & * \end{pmatrix} \quad \begin{aligned} m_1(A) &= 0 \\ m_2(A) &= 0 \\ m_3(A) &= 2 \\ m_4(A) &= 0 \\ m_5(A) &= 3 \\ m_6(A) &= 1 \\ m_7(A) &= 4 \end{aligned}$$

$\mathrm{env}(A)$ = rote Einträge

$*$ ≙ nichtverschwindender Eintrag $a_{ij} \neq 0$

$\diamond$

Theorem 3.2.18 (Hülle & Fill-in). *Falls $\mathbf{A} \in \mathbb{K}^{n,n}$ regulär, strukturymmetrisch mit LU-Zerlegung $\mathbf{A} = \mathbf{LU}$, dann beschränkt sich der Fill-in auf $\mathrm{env}(\mathbf{A})$.*

Beweis. (induktiv) Im ersten Schritt der Gausselimination (ohne Pivotsuche):

$$\mathbf{A} = \begin{pmatrix} a_{11} & \mathbf{b}^T \\ \mathbf{c} & \tilde{\mathbf{A}} \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 \\ -\frac{\mathbf{c}}{a_{11}} & \mathbf{I} \end{pmatrix}}_{\tilde{\mathbf{L}}} \underbrace{\begin{pmatrix} a_{11} & \mathbf{b}^T \\ 0 & \tilde{\mathbf{A}} - \frac{\mathbf{c}\mathbf{b}^T}{a_{11}} \end{pmatrix}}_{\tilde{\mathbf{U}}}$$

Wenn $(i,j) \notin \mathrm{env}(\mathbf{A}), i \geq j \quad \Rightarrow \quad \begin{cases} c_i = b_i = 0 \\ a_{ij} = a_{ji} = 0 \end{cases} \quad \Rightarrow \quad \tilde{l}_{ij} = 0 \;\&\; \tilde{u}_{ji} = 0 \,.$

Insbesondere $(i-1, j-1) \notin \mathrm{env}(\tilde{\mathbf{U}})$ für $i,j > 1$ $\square$

Speichere nur a_{ij}, $(i,j) \in \mathrm{env}(\mathbf{A})$ bei in situ Berechnung der LU-Zerlegung von $\mathbf{A} \in \mathbb{K}^{n,n}$

- Speicherbedarf: $n + 2\sum_{i=1}^{n} m_i(\mathbf{A})$ Gleitpunktzahlen
- HUELLSPEICHERT hüllenorientierte Speichertechnik

Beispiel 85 (Hüllenorientierte Speicherung).

Lineare hüllenorientierte Speicherung von s.p.d. $\mathbf{A} = \mathbf{A}^T \in \mathbb{R}^{n,n}$:

Zwei Arrays:

real `val`(1:P),
integer `dptr`(0:n)

$$P := n + \sum_{i=1}^{n} m_i(A) \ .$$

(3.2.14) eq:Pdef

$$\mathbf{A} = \begin{pmatrix} * & 0 & * & 0 & 0 & 0 & 0 \\ 0 & * & 0 & 0 & * & 0 & 0 \\ * & 0 & * & 0 & 0 & 0 & * \\ 0 & 0 & 0 & * & * & 0 & * \\ 0 & * & 0 & * & * & * & 0 \\ 0 & 0 & 0 & 0 & * & * & 0 \\ 0 & 0 & * & * & 0 & 0 & * \end{pmatrix}$$

Regel:

$$\texttt{dptr}[j] = k \quad \Updownarrow \quad \texttt{val}[k] = a_{jj}$$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
val		a_{11}	a_{22}	a_{31}	a_{32}	a_{33}	a_{44}	a_{52}	a_{53}	a_{54}	a_{55}	a_{65}	a_{66}	a_{73}	a_{74}	a_{75}	a_{76}	a_{77}
dptr	0	1	2	5	6	10	12	17										

◇

Bemerkung 86. Hüllenorientierte Cholesky-Zerlegung von s.p.d. $\mathbf{A} \in \mathbb{R}^{n,n}$ überspringt $(i,j) \notin \mathrm{env}(\mathbf{A})$

➢ Rechenaufwand: $O(P \cdot \max_{1 \le i \le N} m_i(\mathbf{A}))$ elementare Operationen **!**

△

Algorithmus: Hüllenorientierte Cholesky-Zerlegung

for $(j = 1; j \le N; j++)$ {
  for $(r_{jj} = a_{jj}, k = j - m_j(A); k < j; k++)$
    $r_{jj} = r_{jj} - r_{jk}^2$;
  $r_{jj} = \sqrt{r_{jj}}$;
  for $(i = j+1; 1 \le N; i++)$ {
    if $((i,j) \in env(\mathbf{A}))$ {
      for $(r_{ij} = a_{ij}, k = j - m_i(\mathbf{A}); k < j; k++)$
        $r_{ij} -= r_{ik} * r_{jk}$;
      $r_{ij} = r_{ij}/r_{jj}$;
} } }

Rechenaufwand:

$O(P \cdot \max_{1 \le i \le N} m_i(\mathbf{A}))$ elementare Operationen

$$P := n + \sum_{i=1}^{n} m_i(\mathbf{A}) \ .$$

$\mathbf{A} \in \mathbb{R}^{n,n}$ s.p.d. Bandmatrix

Rechenaufwand für Cholesky-Zerlegung:

$$O(m(\mathbf{A}) \cdot n)$$

BANDBRRED
Bandbreitenreduktion:

Ziel: Reduziere $m_i(\mathbf{A})$,$\mathbf{A} = (a_{ij}) \in \mathbb{R}^{N,N}$, durch Zeilen- und Spaltenvertauschungen.

Beispiel 87 (Bandbreitenreduktion durch Indexpermutation).

Spiegelung an Nebendiagonalen ➤ Reduktion von P aus (3.2.14) eq:Pdef

$$\begin{pmatrix} * & 0 & 0 & * & * & * \\ 0 & * & 0 & 0 & 0 & 0 \\ 0 & 0 & * & 0 & 0 & 0 \\ * & 0 & 0 & * & * & * \\ * & 0 & 0 & * & * & * \\ * & 0 & 0 & * & * & * \end{pmatrix} \longrightarrow \begin{pmatrix} * & * & * & 0 & 0 & * \\ * & * & * & 0 & 0 & * \\ * & * & * & 0 & 0 & * \\ 0 & 0 & 0 & * & 0 & 0 \\ 0 & 0 & 0 & 0 & * & 0 \\ * & * & * & 0 & 0 & * \end{pmatrix}$$

$i \leftarrow N + 1 - i$

$P = 30$ $\qquad$ $P = 22$

Allgemeine Techniken benutzen MATRIXGRAPH Matrixgraph: (→ Graphentheorie)

Definition 3.2.19 (Matrixgraph). *Matrix-Graph* $G(\mathbf{A}) = (VG(\mathbf{A}), EG(\mathbf{A}))$ *zu* $\mathbf{A} \in \mathbb{R}^{n,n}$*:*

KNOTENMENGE
- *Knotenmenge* $VG(\mathbf{A})$ *von* $G(\mathbf{A}) = \{1, .., n\}$

KANTENMENGE
- *Kantenmenge* $EG(\mathbf{A})$ *von* $G(\mathbf{A})$*:* $EG(\mathbf{A}) = \{(i,j) \in \{1, .., n\}^2; a_{ij} \neq 0\}$

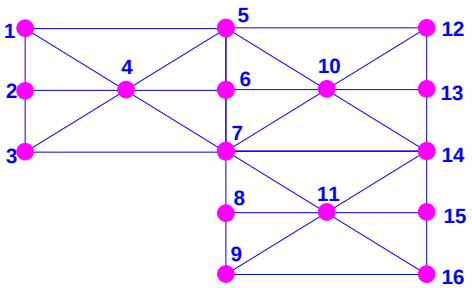
$\mathbf{A}$ *strukturSymmetrisch* $\Rightarrow$ $G(\mathbf{A})$ *ungerichtet.*

$\iota \in VG(\mathbf{A})$*:* *Grad* $\deg \iota := \sharp\{\pi \in EG(\mathbf{A}) : \exists \kappa \in VG(A), \pi = (\iota, \kappa)\}$

PLANARMATGRAPH

Beispiel 88 (Matrixgraph). Matrix mit planarem Matrixgraphen („Schaltkreis, Bsp. 66) ex:elnet

$$\mathbf{A} = \begin{pmatrix} * & * & 0 & * & * & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ * & * & * & * & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & * & * & * & 0 & 0 & * & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ * & * & * & * & * & * & * & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ * & 0 & 0 & * & * & * & 0 & 0 & * & 0 & * & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & * & * & * & * & 0 & 0 & * & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & * & * & 0 & * & * & * & 0 & * & * & 0 & 0 & * & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & * & * & * & 0 & * & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & * & * & 0 & * & 0 & 0 & 0 & * \\ 0 & 0 & 0 & 0 & 0 & * & * & 0 & 0 & * & 0 & * & * & * & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & * & * & * & 0 & 0 & * \\ 0 & 0 & 0 & 0 & 0 & * & 0 & 0 & 0 & * & 0 & * & * & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & * & * & * & * & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & * & 0 & 0 & * & 0 & * & 0 & * & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & * & * & * \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & * & 0 & * & 0 & 0 & * & * \end{pmatrix} \rightarrow$$



Umordnungsalgorithmus von Cuthill & McKee [41] (für strukturssymmetrische Matrix $\mathbf{A} \in \mathbb{R}^{n,n}$):

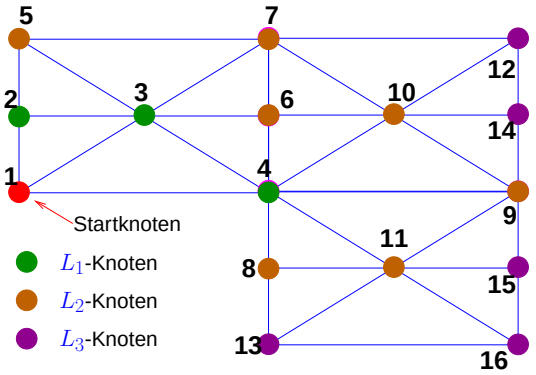
① Startknoten minimalen Grades $\iota_0 \in VG(\mathbf{A}) : \deg(\iota_0) = \min\{\deg(\nu) : \nu \in VG(\mathbf{A})\}$

② Partitionierung von $VG(\mathbf{A})$:

$$L_i := \{\nu \in VG(\mathbf{A}) : \operatorname{dist}(\nu\, \iota_0) = i\}\ , \quad i \in \mathbb{N}\ .$$

③ Sortierung $VG(\mathbf{A}) = \{\iota_0\} \cup L_1 \cup L_2 \cup \cdots \cup L_N$,
interne Sortierung der L_i nach aufsteigendem Grad.

Beispiel 89 (Algorithmus von Cuthill & McKee).
Matrix aus Bsp. 88: Partitionierung und Numerierung erzeugt durch CMK-Algorithmus



Erweiterung: Reverse Cuthill-McKee-Algorithmus
(CMK-Alg. nach Spiegelung an Nebendiagonalen → Bsp. 87)

☞ MATLAB-Funktion: `symrcm(M)`

Bemerkung 90. Alternative Umsortierungsstrategie: Optimieren der Anordnung unter Berücksichtigung der „Fill-in"-Historie der Cholesky-Zerlegung, minimum degree reordering [2].

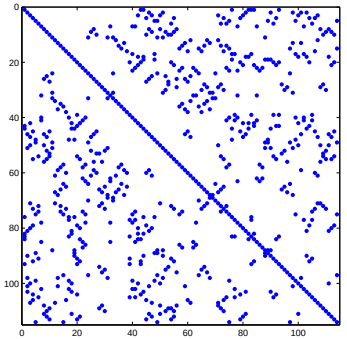
☞ MATLAB-Funktion: `symamd(M)`

Beispiel 91 (Fill-in-Minimierung durch Umordnung).

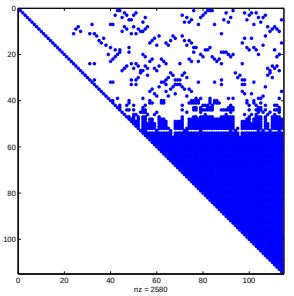
$\mathbf{M}$: 114×114 s.p.d. Matrix aus FE-Diskretisierung einer PDE.

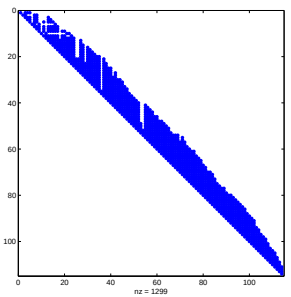
```
spy(M);
[R,P] = chol(M); spy(R);
r = symrcm(M);
[R,P] = chol(M(r,r)); spy(R);
m = symamd(M);
[R,p] = chol(M(m,m)); spy(R);
```

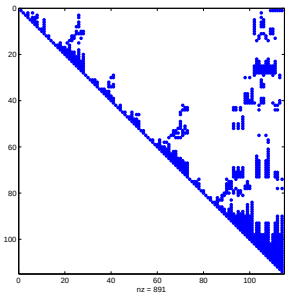
Besetzungsmuster von $\mathbf{M}$ ➥



Untersuche Besetzungmuster von Cholesky-Faktoren (→ Abschnitt 3.2.5) nach Umordnung:


keine Umordnung


Reverse Cuthill-McKee


Approximate Minimum Degree

Warnung: Immer Bibliotheksroutinen zur Lösung von allgemeinen LGS mit dünnbesetzten Koeffizientenmatrizen verwenden **!**
→ SuperLU (`http://www.cs.berkeley.edu/~demmel/SuperLU.html`)
→ UMFPACK (`http://www.cise.vfl.edu/research/sparse/umfpack.html`)
→ Pardiso (`http://www.computational.unibas.ch/cs/scicomp/software/pardiso`)

Matlab-\-Löser: verwendet Sparse-Techniken für `sparse`-Matrizen

3.2.7 Die QR-Zerlegung

QRZERLEGUNG
LU-Probleme (Abschn. 3.2.4): ☹ i.a. Pivotsuche notwendig (➞ Strukturzerstörung)
☹ Mögliche Instabilität (bei Spaltenpivotsuche)

Definition 3.2.20.
- $\mathbf{Q} \in \mathbb{K}^{n,n}$, $n \in \mathbb{N}$, *heisst unitär, falls* $\mathbf{Q}^{-1} = \mathbf{Q}^H$.
- $\mathbf{Q} \in \mathbb{R}^{n,n}$, $n \in \mathbb{N}$, *heisst orthogonal, falls* $\mathbf{Q}^{-1} = \mathbf{Q}^T$.

Theorem 3.2.21 (Kriterium für Unitarität)**.**

$$\mathbf{Q} \in \mathbb{C}^{n,n} \quad \textit{unitär} \quad \Leftrightarrow \quad \|\mathbf{Q}\mathbf{x}\|_2 = \|\mathbf{x}\|_2 \quad \forall \mathbf{x} \in \mathbb{K}^n\ .$$

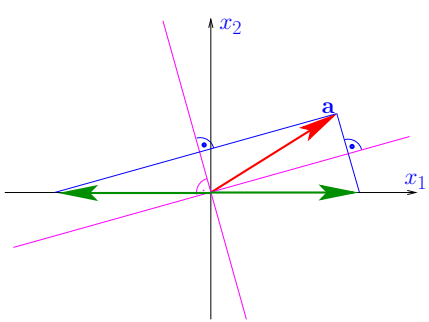
$\mathbf{Q}$ unitär ⇒ $\operatorname{cond}(\mathbf{Q}) = 1$ ➤ Unitäre Transformationen „stabilitätsfördernd"

Falls $\mathbf{Q} \in \mathbb{K}^{n,n}$ unitär, dann

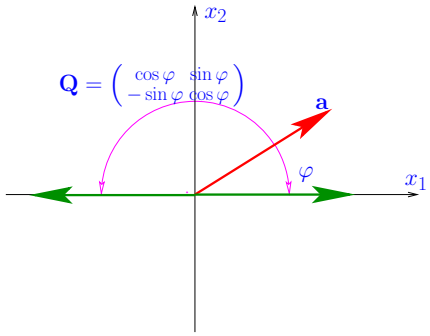
- haben alle Spalten- und Zeilenvektoren paarweise Euklidische Norm $= 1$,
- sind alle Spalten- und Zeilenvektoren paarweise orthogonal (bzgl. Euklidischem Skalarprodukt),
- ist $|\det \mathbf{Q}| = 1$, alle Eigenwerte haben Betrag $= 1$.

Ziel: Finde orthogonale/unitäre Zeilenumformungen zum Anullieren von Matrixkomponenten.

In 2D: Zwei orthogonale Transformationen anullieren zweite Komponente von $\mathbf{a} \in \mathbb{R}^2$:



Spiegelung an Winkelhalbierenden



Rotation auf x_1-Achse

➢ Es gibt jeweils zwei mögliche Reflektionen/Rotationen

In nD: Zu $\mathbf{a} \in \mathbb{K}^n$ suche unitäre Matrix $\mathbf{Q} \in \mathbb{K}^{n,n}$ mit $\mathbf{Q}\mathbf{a} = \|\mathbf{a}\|_2 \mathbf{e}_1$, $\mathbf{e}_1 \hat{=}$ 1. Einheitsvektor.

1. Möglichkeit: HOUSEREFL Householder-Reflektionen

$$\mathbf{Q} = \mathbf{H}(\mathbf{v}) := \mathbf{I} - 2\frac{\mathbf{v}\mathbf{v}^H}{\mathbf{v}^H\mathbf{v}} \quad \text{mit} \quad \mathbf{v} = \tfrac{1}{2}(\mathbf{a} \pm \|\mathbf{a}\|_2 \mathbf{e}_1) \,. \tag{3.2.15}$$ eq:HH

Zur *Vermeidung von Auslöschung* (→ Abschn. 1.7) wähle sec:ausloschung

$$\mathbf{v} = \begin{cases} \frac{1}{2}(\mathbf{a} + \|\mathbf{a}\|_2 \mathbf{e}_1) & \text{, falls } \operatorname{Re} a_1 > 0 \,, \\ \frac{1}{2}(\mathbf{a} - \|\mathbf{a}\|_2 \mathbf{e}_1) & \text{, falls } \operatorname{Re} a_1 \le 0 \,. \end{cases}$$

2. Möglichkeit: GIVENS Sukzessive Givens-Rotationen (→ 2D Fall)

$$\mathbf{G}_{1k}(a_1, a_k)\mathbf{A} := \begin{pmatrix} \bar\gamma & \cdots & \bar\sigma & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots \\ -\sigma & \cdots & \gamma & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 1 \end{pmatrix} \begin{pmatrix} a_1 \\ \vdots \\ a_k \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} a_1^{(1)} \\ \vdots \\ 0 \\ \vdots \\ a_n \end{pmatrix} \quad \text{, wenn} \quad \begin{aligned} \gamma &= \frac{a_1}{\sqrt{|a_1|^2+|a_k|^2}} \,, \\ \sigma &= \frac{a_k}{\sqrt{|a_1|^2+|a_k|^2}} \,. \end{aligned}$$

Zur *Vermeidung von Überlauf* (in $\mathbb{R}$) → Abschn. 1.5): sec:ueber-und-unterlauf

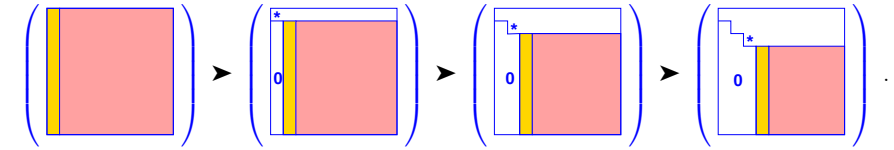
$$\begin{cases} \tau = \frac{a_1}{a_k} \,, & \sigma = \frac{1}{\sqrt{1+\tau^2}} \,, & \gamma = \sigma\tau & \text{, falls } |a_k| > |a_1| \,, \\ \tau = \frac{a_k}{a_1} \,, & \gamma = \frac{1}{\sqrt{1+\tau^2}} \,, & \sigma = \gamma\tau & \text{, falls } |a_k| \le |a_1| \,. \end{cases}$$

MATLAB-Funktion: `[G,x] = planerot(a);`

MATLAB-Code Givens-Rotation

```
function [G,x] = planerot(a)
%PLANEROT Givens plane rotation.
if (a(2) ~= 0), r = norm(a); G = [a'; -a(2) a(1)]/r; x = [r; 0];
else, G = eye(2); end
```

Transformation auf *obere Dreiecksgestalt* durch sukzessive unitäre Transformationen:



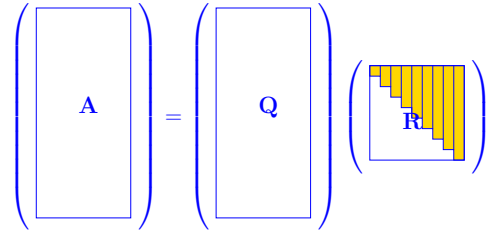
= „Zielspalte $\mathbf{a}$" (bestimmt unitäre Transformation), = Zu modifizierender Restblock.

$$\mathbf{Q}_{n-1}\mathbf{Q}_{n-2} \cdot \cdots \cdot \mathbf{Q}_1\mathbf{A} = \mathbf{R} \,,$$

QRDEC QR-Zerlegung von $\mathbf{A} \in \mathbb{C}^{n,n}$: $\mathbf{A} = \mathbf{Q}\mathbf{R}$, $\mathbf{Q} := \mathbf{Q}_1^H \cdot \cdots \cdot \mathbf{Q}_{n-1}^H$ unitäre Matrix , $\mathbf{R}$ obere Dreiecksmatrix .

Verallgemeinerung auf $\mathbf{A} \in \mathbb{K}^{m,n}$

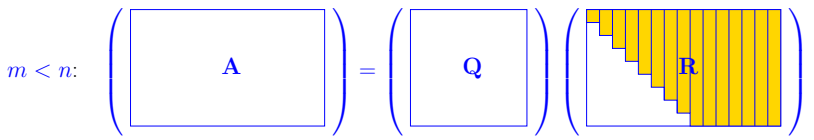
$m > n$:



, $\mathbf{A} = \mathbf{Q}\mathbf{R}$, $\begin{matrix} \mathbf{Q} \in \mathbb{K}^{m,n} \,, \\ \mathbf{R} \in \mathbb{K}^{n,n} \,, \end{matrix}$

wobei $\mathbf{Q}^H\mathbf{Q} = \mathbf{I}$ (orthonormierte Spalten), $\mathbf{R}$ obere Dreiecksmatrix.

$m < n$:



$\mathbf{A} = \mathbf{Q}\mathbf{R}$, $\mathbf{Q} \in \mathbb{K}^{m,m}$, $\mathbf{R} \in \mathbb{K}^{m,n}$,

wobei $\mathbf{Q}$ unitär, $\mathbf{R}$ obere Dreiecksmatrix.

Wann nimmt man welche Transformation **?**

- Householder-Reflektionen vorzuziehen, wenn Zielspalten vollbesetzt
- Givens-Rotationen effizienter (weil selektiver), wenn Zielspalte nur wenige Nicht-Null-Komponenten besitzt.

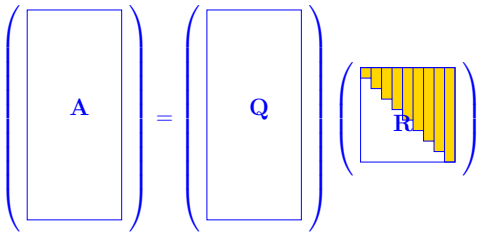
MATLAB-Funktionen:

`[Q,R] = qr(A)` $\mathbf{Q} \in \mathbb{K}^{m,m}$, $\mathbf{R} \in \mathbb{K}^{m,n}$ für $\mathbf{A} \in \mathbb{K}^{m,n}$

`[Q,R] = qr(A,0)` $\mathbf{Q} \in \mathbb{K}^{m,n}$, $\mathbf{R} \in \mathbb{K}^{n,n}$ für $\mathbf{A} \in \mathbb{K}^{m,n}$, $m > n$

(jede Funktion verfügbar für voll- und dünnbesetzte Matrizen)

Bemerkung 92 (QR-Orthogonalisierung).



, $\mathbf{A}, \mathbf{Q} \in \mathbb{K}^{m,n}$,$\mathbf{R} \in \mathbb{K}^{n,n}$.

Falls $m > n$, $\operatorname{rank}(\mathbf{R}) = \operatorname{rank}(\mathbf{A}) = n$ (voller Rang)

➤ $\{\mathbf{q}_{\cdot,1}, \ldots, \mathbf{q}_{\cdot,n}\}$ ist Orthonormalbasis von $\operatorname{Im}(\mathbf{A})$ mit $\operatorname{Span}\{\mathbf{q}_{\cdot,1}, \ldots, \mathbf{q}_{\cdot,k}\} = \operatorname{Span}\{\mathbf{a}_{\cdot,1}, \ldots, \mathbf{a}_{\cdot,k}\}$,$1 \le k \le n$.

△

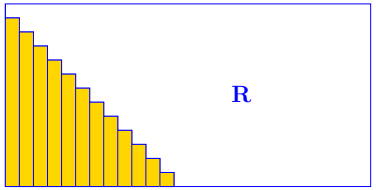
Bemerkung 93 (Akkumulation von unitären Transformationen).

Wie speichert man $\mathbf{G}_{i_1 j_1}(a_1, b_1) \cdot \cdots \cdot \mathbf{G}_{i_k j_k}(a_k, b_k)$, $\mathbf{H}(\mathbf{v}_1) \cdot \cdots \cdot \mathbf{H}(\mathbf{v}_k)$?

☛ Für Householder-Reflektionen

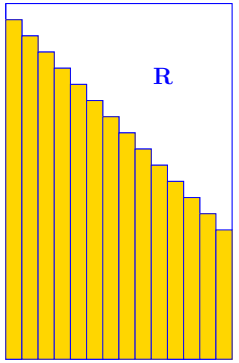
$\mathbf{H}(\mathbf{v}_1) \cdot \cdots \cdot \mathbf{H}(\mathbf{v}_k)$: Speichere $\mathbf{v}_1, \ldots, \mathbf{v}_k$

Für in situ QR-Zerlegung von $\mathbf{A} \in \mathbb{K}^{m,n}$: Speichere „Householder-Vektoren" $\mathbf{v}_j$ (abnehmende Längen !) im unteren Dreieck von $\mathbf{A}$



↑ Fall $m < n$

= Householder-Vektoren.



Fall $m > n \rightarrow$

☛ Konvention für Givens-Rotationen ($\mathbb{K} = \mathbb{R}$)

$$\mathbf{G} = \begin{pmatrix} \bar{\gamma} & \bar{\sigma} \\ -\sigma & \gamma \end{pmatrix} \quad \Rightarrow \quad \text{Speichere} \quad \rho := \begin{cases} 1 & \text{, falls } \gamma = 0 \text{ ,} \\ \frac{1}{2}\operatorname{sign}(\gamma)\sigma & \text{, falls } |\sigma| < |\gamma| \text{ ,} \\ 2\operatorname{sign}(\sigma)/\gamma & \text{, falls } |\sigma| \ge |\gamma| \text{ .} \end{cases}$$

➤
$$\begin{cases} \rho = 1 & \Rightarrow \quad \gamma = 0 \text{ ,} \quad \sigma = 1 \\ |\rho| < 1 & \Rightarrow \quad \sigma = 2\rho \text{ ,} \quad \gamma = \sqrt{1-\sigma^2} \\ |\rho| > 1 & \Rightarrow \quad \gamma = 2/\rho \text{ ,} \quad \sigma = \sqrt{1-\gamma^2} \text{ .} \end{cases}$$

Speichere $\mathbf{G}_{ij}(a, b)$ als Tripel (i, j, ρ)

Speicherung von Orthogonaltransformationen als Matrizen meist ineffizient **!**

△

LOESENLGSQR

Lösen eines linearen Gleichungssystems via QR-Zerlegung:

$\mathbf{A}\mathbf{x} = \mathbf{b}$:

① QR-Zerlegung $\mathbf{A} = \mathbf{Q}\mathbf{R}$, Rechenaufwand $\frac{2}{3}n^3 + O(n^2)$ (ca. doppelt so aufwendig wie LU-Zerlegung ohne Pivotsuche) RECHAUFWQR

② Orthogonaltransformation $\mathbf{z} = \mathbf{Q}^H\mathbf{b}$, Rechenaufwand $4n^2 + O(n)$ bei *kompakter Abspeicherung* der Reflektionen/Rotationen

③ Rücksubstitution, löse $\mathbf{R}\mathbf{x} = \mathbf{z}$, Rechenaufwand $\frac{1}{2}n(n+1)$

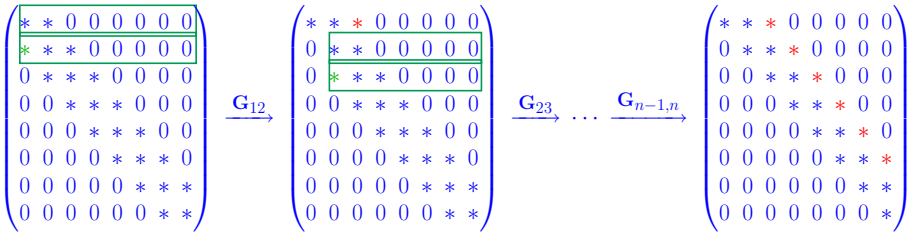
- Die Berechnung der (verallgemeinerten) QR-Zerlegung $\mathbf{A} = \mathbf{QR}$ mittels Householder-Reflektionen bzw. Givens-Rotationen ist für beliebiges $\mathbf{A} \in \mathbb{C}^{m,n}$ stabil.
- Für beliebige reguläre Koeffizientenmatrizen lässt sich ein LGS mittels QR-Zerlegung+ Orthogonaltransformation + Rücksubstitution stabil lösen.

Fill-in bei QR-Zerlegung ?

$\mathbf{A} \in \mathbb{C}^{n,n}$ mit QR-Zerlegung $\mathbf{A} = \mathbf{QR}$ $\Rightarrow$ $m(\mathbf{R}) \leq m(\mathbf{A})$ (→ Def. 3.2.15) def:bw

Beispiel 94 (QR-basiertes Lösen eines tridiagonalen Gleichungssystems).

Elimination der Subdiagonalen durch $n-1$ sukzessive Givens-Rotationen:



MATLAB-Code (c, d, e, b = Spaltenvektoren der Länge n, $n \in \mathbb{N}$, `e(n),c(n)` nicht benutzt):

MATLAB-CODE : Lösen eines tridiagonalen LGS mit QR-Zerlegung

```
function y = tridiagqr(c,d,e,b)
n = length(d); t = norm(d)+norm(e)+norm(c);
for k=1:n-1
  [R,z] = planerot([d(k);e(k)]);
  if (abs(z(1))/t < eps), error('Matrix singular'); end;
  d(k) = z(1); b(k:k+1) = R*b(k:k+1);
  Z = R*[c(k), 0;d(k+1), c(k+1)];
  c(k) = Z(1,1); d(k+1) = Z(2,1);
  e(k) = Z(1,2); c(k+1) = Z(2,2);
end
A = spdiags([d,[0;c(1:end-1)],[0;0;e(1:end-2)]],[0 1 2],n,n);
y = A\b;
```

➤ Rechenaufwand $O(n)$ ◇

Bemerkung 95 (Aufspüren fast singulärer Matrizen).

(Relativ) sehr kleine r_{ii} bei Berechnung der QR-Zerlegung $\leftrightarrow$ $\mathbf{A}$ „fast singulär" △

3.2.8 Modifikationstechniken

MODIFTECH

Problem: Effizientes Aktualisieren von Faktorisierungen bei „geringfügiger" Änderung der Matrix, GOL89 STO83 [19, Abschnitt 12.6], [43, Abschnitt 4.9].

3.2.8.1 Rang-1-Modifikationen

RANG1MOD

$$\mathbf{A} \in \mathbb{K}^{n,n} \mapsto \widetilde{\mathbf{A}} := \mathbf{A} + \mathbf{uv}^H, \quad \mathbf{u}, \mathbf{v} \in \mathbb{K}^n . \tag{3.2.16}$$ mod:1

Allgemeine Rang-1-Matrix

Bemerkung 96 (Lösen eines Rang-1-modifizierten LGS).

SHERMORRWOOD

Lemma 3.2.22 (Sherman-Morrison-Woodbury-Formel). *Für reguläres* $\mathbf{A} \in \mathbb{K}^{n,n}$ *und* $\mathbf{U}, \mathbf{V} \in \mathbb{K}^{n,k}$, $n, k \in \mathbb{N}$, $k \leq n$, *gilt*

$$(\mathbf{A} + \mathbf{UV}^H)^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}(\mathbf{I} + \mathbf{V}^H\mathbf{A}^{-1}\mathbf{U})^{-1}\mathbf{V}^H\mathbf{A}^{-1} ,$$

wenn $\mathbf{I} + \mathbf{V}^H\mathbf{A}^{-1}\mathbf{U}$ *invertierbar.*

Aufgabe: Zu lösen: $\widetilde{\mathbf{A}}\mathbf{x} = \mathbf{b}$, wobei LU-Zerlegung $\mathbf{A} = \mathbf{LU}$ bekannt

Anwendung von Lemma 3.2.22 lem:SMW für $k = 1$:

$$\mathbf{x} = \left(\mathbf{I} - \frac{\mathbf{A}^{-1}\mathbf{uv}^H}{1 + \mathbf{v}^H\mathbf{A}^{-1}\mathbf{u}}\right)\mathbf{A}^{-1}\mathbf{b} .$$

Effiziente Implementierung ! ➤

MATLAB-CODE : Lösen eines Rang-1-modifizierten LGS

```
function x = smw(L,U,u,v,b)
t = L\b; z = U\t;
t = L\u; w = U\t;
alpha = 1+dot(v,w);
if (abs(alpha) < eps*norm(U,1))
error('Nearly singular matrix');
end;
x = z - w*dot(v,z)/alpha;
```

Aufgabe: Effiziente Berechnung der QR-Zerlegung (→ Abschn. 3.2.7) sec:die-qr-zerlegung $\widetilde{\mathbf{A}} = \widetilde{\mathbf{Q}}\widetilde{\mathbf{R}}$ von $\widetilde{\mathbf{A}}$ aus (3.2.16), mod:1 wenn eine QR-Zerlegung $\mathbf{A} = \mathbf{QR}$ bekannt.

① Mit $\mathbf{w} := \mathbf{Q}^H\mathbf{u}$: $\mathbf{A} + \mathbf{uv}^H = \mathbf{Q}(\mathbf{R} + \mathbf{wv}^H)$

➡ Rechenaufwand $O(n^2)$ (abhängig von Abspeicherung von $\mathbf{Q}$)

② Ziel: $\mathbf{w} \to \|\mathbf{w}\|\,\mathbf{e}_1$ ➤ durch $n-1$ Givens-Rotationen

$$\mathbf{w} = \begin{pmatrix}*\\*\\\vdots\\*\\*\\*\end{pmatrix} \xrightarrow{\mathbf{G}_{n-1,n}} \begin{pmatrix}*\\*\\\vdots\\*\\*\\0\end{pmatrix} \xrightarrow{\mathbf{G}_{n-2,n-1}} \begin{pmatrix}*\\*\\\vdots\\*\\0\\0\end{pmatrix} \xrightarrow{\mathbf{G}_{n-3,n-2}} \cdots \xrightarrow{\mathbf{G}_{12}} \begin{pmatrix}*\\0\\\vdots\\0\\0\\0\end{pmatrix}$$

Beachte: Auswirkung auf $\mathbf{R}$!

$$\mathbf{R} = \begin{pmatrix} * & * & \cdots & * & * & * & * \\ 0 & * & \cdots & * & * & * & * \\ \vdots & & \ddots & & & & \vdots \\ 0 & \cdots & 0 & * & * & * & * \\ 0 & \cdots & 0 & 0 & * & * & * \\ 0 & \cdots & 0 & 0 & 0 & * & * \\ 0 & \cdots & 0 & 0 & 0 & * & * \end{pmatrix} \xrightarrow{\mathbf{G}_{n-1,n}} \begin{pmatrix} * & * & \cdots & * & * & * & * \\ 0 & * & \cdots & * & * & * & * \\ \vdots & & \ddots & & & & \vdots \\ 0 & \cdots & 0 & * & * & * & * \\ 0 & \cdots & 0 & 0 & * & * & * \\ 0 & \cdots & 0 & 0 & 0 & * & * \\ 0 & \cdots & 0 & 0 & 0 & * & * \end{pmatrix} \xrightarrow{\mathbf{G}_{n-2,n-1}}$$

$$\longrightarrow \begin{pmatrix} * & * & \cdots & * & * & * & * \\ 0 & * & \cdots & * & * & * & * \\ \vdots & & \ddots & & & & \vdots \\ 0 & \cdots & 0 & * & * & * & * \\ 0 & \cdots & 0 & 0 & * & * & * \\ 0 & \cdots & 0 & 0 & * & * & * \\ 0 & \cdots & 0 & 0 & 0 & * & * \end{pmatrix} \xrightarrow{\mathbf{G}_{n-3,n-2}} \cdots \xrightarrow{\mathbf{G}_{1,2}} \begin{pmatrix} * & * & \cdots & * & * & * & * \\ * & * & \cdots & * & * & * & * \\ & & \ddots & & & & \vdots \\ 0 & \cdots & * & * & * & * & * \\ 0 & \cdots & 0 & * & * & * & * \\ 0 & \cdots & 0 & 0 & * & * & * \\ 0 & \cdots & 0 & 0 & 0 & * & * \end{pmatrix} =: \mathbf{R}_1$$

OHESS
Obere Hessenbergmatrix: Eintrag $(i,j) = 0$, falls $i > j+1$.

$$\blacktriangleright \quad \mathbf{A} + \mathbf{u}\mathbf{v}^H = \mathbf{Q}\mathbf{Q}_1^H\big(\underbrace{\mathbf{R}_1 + \|\mathbf{w}\|_2\,\mathbf{e}_1\mathbf{v}^H}_{\text{obere Hessenbergmatrix}}\big) \quad \text{mit unitärem } \mathbf{Q}_1 := \mathbf{G}_{12}\cdot\cdots\cdot\mathbf{G}_{n-1,n}\,.$$

➡ Rechenaufwand $O(n^2)$

③ Sukzessive Givens-Rotationen: $\mathbf{R}_1 + \|\mathbf{w}\|_2\,\mathbf{e}_1\mathbf{v}^H \mapsto$ obere Dreiecksgestalt

$$\mathbf{R}_1 + \|\mathbf{w}\|_2\,\mathbf{e}_1\mathbf{v}^H = \begin{pmatrix} * & * & \cdots & * & * & * & * \\ * & * & \cdots & * & * & * & * \\ & & \ddots & & & & \\ 0 & \cdots & * & * & * & * & * \\ 0 & \cdots & 0 & * & * & * & * \\ 0 & \cdots & 0 & 0 & * & * & * \\ 0 & \cdots & 0 & 0 & 0 & * & * \end{pmatrix} \xrightarrow{\mathbf{G}_{12}} \begin{pmatrix} * & * & \cdots & * & * & * & * \\ 0 & * & \cdots & * & * & * & * \\ & & \ddots & & & & \\ 0 & \cdots & * & * & * & * & * \\ 0 & \cdots & 0 & * & * & * & * \\ 0 & \cdots & 0 & 0 & * & * & * \\ 0 & \cdots & 0 & 0 & 0 & * & * \end{pmatrix} \xrightarrow{\mathbf{G}_{23}} \cdots$$

$$\xrightarrow{\mathbf{G}_{n-2,n-1}} \begin{pmatrix} * & * & \cdots & * & * & * & * \\ 0 & * & \cdots & * & * & * & * \\ & & \ddots & & & & \\ 0 & \cdots & 0 & * & * & * & * \\ 0 & \cdots & 0 & 0 & * & * & * \\ 0 & \cdots & 0 & 0 & 0 & * & * \\ 0 & \cdots & 0 & 0 & 0 & * & * \end{pmatrix} \xrightarrow{\mathbf{G}_{n-1,n}} \begin{pmatrix} * & * & \cdots & * & * & * & * \\ 0 & * & \cdots & * & * & * & * \\ & & \ddots & & & & \\ 0 & \cdots & 0 & * & * & * & * \\ 0 & \cdots & 0 & 0 & * & * & * \\ 0 & \cdots & 0 & 0 & 0 & * & * \\ 0 & \cdots & 0 & 0 & 0 & 0 & * \end{pmatrix} =: \widetilde{\mathbf{R}}\,. \qquad (3.2.17)$$ mod:OHI

➡ Rechenaufwand $O(n^2)$

$$\mathbf{A} + \mathbf{u}\mathbf{v}^H = \widetilde{\mathbf{Q}}\widetilde{\mathbf{R}} \quad \text{mit } \widetilde{\mathbf{Q}} = \mathbf{Q}\mathbf{Q}_1^H\mathbf{G}_{n-1,n}^H\cdot\cdots\cdot\mathbf{G}_{12}^H\,.$$

MATLAB-Funktion: `[Q1,R1] = qrupdate(Q,R,u,v);`

Symmetrische und positivitätserhaltende Rang-1-Modifikation:

$$\mathbf{A} = \mathbf{A}^H \in \mathbb{K}^{n,n} \;\mapsto\; \widetilde{\mathbf{A}} := \mathbf{A} + \alpha\mathbf{v}\mathbf{v}^H\,, \quad \mathbf{v} \in \mathbb{K}^n\,,\ \alpha > 0\,. \qquad (3.2.18)$$ mod:2

Aufgabe: Effiziente Berechnung der Cholesky-Zerlegung $\widetilde{\mathbf{A}} = \widetilde{\mathbf{R}}^H\widetilde{\mathbf{R}}$ (→ Lemma 3.2.13) von $\widetilde{\mathbf{A}}$ aus (3.2.18), wenn eine solche $\mathbf{A} = \mathbf{R}^H\mathbf{R}$ für $\mathbf{A}$ bekannt.

① Mit $\mathbf{w} := \mathbf{R}^{-H}\mathbf{v}$: $\mathbf{A} + \alpha\mathbf{v}\mathbf{v}^H = \mathbf{R}^H(\mathbf{I} + \alpha\mathbf{w}\mathbf{w}^H)\mathbf{R}$.

➡ Rechenaufwand $O(n^2)$ (Rücksubstitution !)

② Idee: Formale Gauss-Elimination: mit $\widetilde{\mathbf{w}} = (w_2,\ldots,w_n)^T$ → (3.2.5)

$$\mathbf{I} + \alpha\mathbf{w}\mathbf{w}^H = \left(\begin{array}{c|c} 1+\alpha w_1^2 & \alpha w_1\widetilde{\mathbf{w}}^H \\ \hline \alpha w_1\widetilde{\mathbf{w}} & \mathbf{I} + \alpha\widetilde{\mathbf{w}}\widetilde{\mathbf{w}}^H \end{array}\right) \to \left(\begin{array}{c|c} 1+\alpha w_1^2 & \alpha w_1\widetilde{\mathbf{w}}^H \\ \hline 0 & \mathbf{I} + \alpha^{(1)}\widetilde{\mathbf{w}}\widetilde{\mathbf{w}}^H \end{array}\right) \qquad (3.2.19)$$ mod:rec

wobei $\alpha^{(1)} := 1 - \dfrac{\alpha^2 w_1^2}{1+\alpha w_1^2}$.

Analoge Struktur ➢ Rekursion

► Berechnung der Cholesky-Zerlegung

$$\mathbf{I} + \alpha \mathbf{w}\mathbf{w}^H = \mathbf{R}_1^H \mathbf{R}_1 \, .$$

Motiviert durch „Rekursion” mod:rec (3.2.19).

➜ Rechenaufwand $O(n^2)$ ($O(n)$ wenn nur Abspeicherung von `d, s` → eq:modr1 (3.2.20))

MATLAB-CODE Cholesky-Zerlegung bei Rang-1-Modifikation

```
function [d,s] = roid(alpha,w)
n = length(w);
d = []; s = [];
for i=1:n
  t = alpha*w(i);
  d = [d; sqrt(1+t*w(i))];
  s = [s; t/d(i)];
% R=[R;zeros(1,i-1),d(i), ...
%          s(i)*w(i+1:end)'];
  alpha = alpha - s(i)^2;
end
```

③ Spezielle Struktur von $\mathbf{R}_1$:

$$\mathbf{R}_1 = \begin{pmatrix} d_1 & & \\ & \ddots & \\ & & d_n \end{pmatrix} + \begin{pmatrix} s_1 & & \\ & \ddots & \\ & & s_n \end{pmatrix} \begin{pmatrix} 0 & w_2 & w_3 & \cdots & \cdots & w_n \\ 0 & 0 & w_3 & \cdots & \cdots & w_n \\ \vdots & & \ddots & & & \vdots \\ \vdots & & & 0 & w_{n-1} & w_n \\ 0 & \cdots & & \cdots & 0 & w_n \\ 0 & \cdots & & & \cdots & 0 \end{pmatrix} \tag{3.2.20}$$ eq:modr2

$$\mathbf{R}_1 = \begin{pmatrix} d_1 & & \\ & \ddots & \\ & & d_n \end{pmatrix} + \begin{pmatrix} s_1 & & \\ & \ddots & \\ & & s_n \end{pmatrix} \begin{pmatrix} 0 & 1 & 1 & \cdots & \cdots & 1 \\ 0 & 0 & 1 & \cdots & \cdots & 1 \\ \vdots & & \ddots & & & \vdots \\ \vdots & & & 0 & 1 & 1 \\ 0 & \cdots & & \cdots & 0 & 1 \\ 0 & \cdots & & & \cdots & 0 \end{pmatrix} \begin{pmatrix} w_1 & & \\ & \ddots & \\ & & w_n \end{pmatrix}$$

► Geschickte Multiplikation

$$\widetilde{\mathbf{R}} := \mathbf{R}_1 \mathbf{R}$$

➜ Rechenaufwand $O(n^2)$

$$\mathbf{A} + \alpha \mathbf{v}\mathbf{v}^H = \widetilde{\mathbf{R}}^H \widetilde{\mathbf{R}} \, .$$

MATLAB-CODE : roudchol

```
function Rt = roudchol(R,alpha,v)
w = R'\v;
[d,s] = roid(alpha,w);
T = zeros(1,n);
for j=n-1:-1:1
  T = [w(j+1)*R(j+1,:)+T(1,:);T];
end
Rt = spdiags(d,0,n,n)*R+spdiags(s,0,n,n)*T;
```

MATLAB-Funktion: `R = cholupdate(R,v);`

3.2.8.2 Hinzufügen einer Spalte

$$\mathbf{A} \in \mathbb{K}^{m,n} \;\mapsto\; \widetilde{\mathbf{A}} = \left[\mathbf{a}_{\cdot,1}, \ldots, \mathbf{a}_{\cdot,k-1}, \mathbf{v}, \mathbf{a}_{\cdot,k}, \ldots, \mathbf{a}_{\cdot,n}\right] \;, \quad \mathbf{v} \in \mathbb{K}^m \, . \tag{3.2.21}$$ mod:col

Gegeben: QR-Zerlegung $\mathbf{A} = \mathbf{Q}\mathbf{R}$, $\mathbf{Q} \in \mathbb{K}^{m,m}$ unitär, $\mathbf{R} \in \mathbb{K}^{m,n}$ obere Dreiecksmatrix.

Aufgabe: Effiziente Berechnung der QR-Zerlegung $\widetilde{\mathbf{A}} = \widetilde{\mathbf{Q}}\widetilde{\mathbf{R}}$ von $\widetilde{\mathbf{A}}$ aus mod:col (3.2.21), $\widetilde{\mathbf{Q}} \in \mathbb{K}^{m,m}$ unitär, $\widetilde{\mathbf{R}} \in \mathbb{K}^{m,n+1}$ obere Dreiecksmatrix.

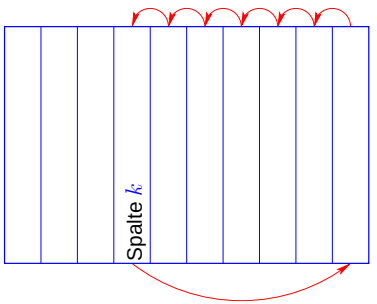
Idee: Aufgabe einfach, falls $k = n+1$ (Anfügen einer Spalte von rechts)

► $\exists$ Teilzyklische Spaltenpermutation

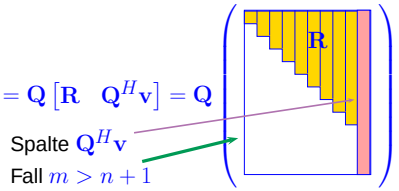
$$k \mapsto n+1 \, , \; i \mapsto i-1 \, , \; i = k+1, \ldots, n+1$$

$\sim$ Permutationsmatrix

$$\mathbf{P} = \begin{pmatrix} 1 & 0 & \cdots & & \cdots & 0 \\ 0 & \ddots & & & & \\ & & 1 & 0 & & \\ \vdots & & 0 & 1 & & \\ \vdots & & & & \ddots & \\ & & & & & 1 \\ 0 & \cdots & & 1 & 0 & \end{pmatrix} \in \mathbb{R}^{n+1,n+1}$$



$$\widetilde{\mathbf{A}} \;\longrightarrow\; \mathbf{A}_1 = \widetilde{\mathbf{A}}\mathbf{P} = [\mathbf{a}_{\cdot,1}, \ldots, \mathbf{a}_{\cdot,n}, \mathbf{v}] = \mathbf{Q}\left[\mathbf{R} \quad \mathbf{Q}^H\mathbf{v}\right] = \mathbf{Q}$$



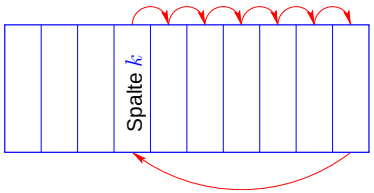
① Falls $m > n+1$: $\exists$ orthogonale Transformation $\mathbf{Q}_1 \in \mathbb{K}^{m,m}$ (Householder-Reflektion) mit

$$\mathbf{Q}_1\mathbf{Q}^H\mathbf{v} = \begin{pmatrix} * \\ \vdots \\ * \\ * \\ 0 \\ \vdots \\ 0 \end{pmatrix} \begin{matrix} \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} n+1 \\ \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} m-n-1 \end{matrix} \quad \blacktriangleright \quad \mathbf{Q}_1\mathbf{Q}^H\mathbf{A}_1 = \begin{pmatrix} * & \cdots & & & \cdots & * \\ 0 & * & & & & * \\ \vdots & & \ddots & & & \vdots \\ & & & & * & * \\ \vdots & & & & & * \\ 0 & \cdots & & & \cdots & 0 \\ \vdots & & & & & \vdots \\ 0 & \cdots & & & \cdots & 0 \end{pmatrix} \begin{matrix} \left.\begin{matrix} \\ \\ \\ \\ \\ \end{matrix}\right\} n+1 \\ \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} m-n-1 \end{matrix}$$

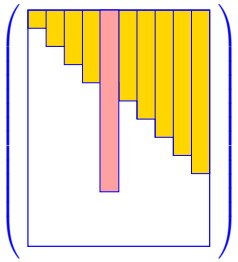
➨ Rechenaufwand $O(m-n)$ (Eine Reflektion)

Inverse Permutation:

$\sim$ Multiplikation mit $\mathbf{P}^H$ von rechts

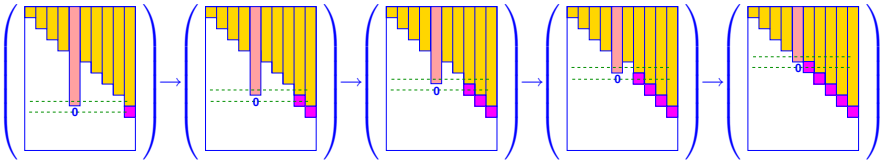


$$\mathbf{Q}_1\mathbf{Q}^H\widetilde{\mathbf{A}} = \mathbf{Q}_1\mathbf{Q}^H\mathbf{A}_1\mathbf{P}^H = \begin{pmatrix} * & \cdots & & * & & \cdots & * \\ 0 & * & & * & & & \vdots \\ \vdots & & \ddots & \vdots & & & \\ & & & \vdots & \ddots & & \vdots \\ & & & * & & * & * \\ \vdots & & & * & & & * \\ 0 & \cdots & & 0 & & \cdots & 0 \\ \vdots & & & \vdots & & & \vdots \\ 0 & \cdots & & 0 & & \cdots & 0 \end{pmatrix} =$$



② $n+1-k$ sukzessive Givens-Rotationen $\Rightarrow$ obere Dreiecksmatrix $\widetilde{\mathbf{R}}$

$$\mathbf{Q}_1\mathbf{Q}^H\mathbf{A}_1 = \begin{pmatrix} * & \cdots & & * & & \cdots & * \\ 0 & * & & * & & & \vdots \\ \vdots & & \ddots & \vdots & & & \\ & & & \vdots & \ddots & & \vdots \\ & & & * & & * & * \\ & & & * & & & * \\ \vdots & & & * & & & 0 \\ 0 & \cdots & & 0 & & \cdots & \vdots \\ \vdots & & & \vdots & & & \vdots \\ 0 & \cdots & & 0 & & \cdots & 0 \end{pmatrix} \xrightarrow{\mathbf{G}_{n,n+1}} \cdots \xrightarrow{\mathbf{G}_{k,k+1}} \begin{pmatrix} * & \cdots & & * & & \cdots & * \\ 0 & * & & * & & & \vdots \\ \vdots & & \ddots & \vdots & & & \\ & & & * & & & \\ & & & 0 & \ddots & & \\ & & & 0 & & * & \vdots \\ \vdots & & & 0 & & & * \\ 0 & \cdots & & 0 & & \cdots & 0 \\ \vdots & & & \vdots & & & \\ 0 & \cdots & & 0 & & \cdots & 0 \end{pmatrix}$$



- - - - $\hat{=}$ Zielzeilen der Givens-Rotationen, ■ $\hat{=}$ neue Einträge $\neq 0$

➡ Rechenaufwand $O\left((n-k)^2\right)$

3.2.8.3 Hinzufügen einer Zeile

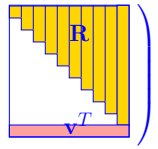
$$\mathbf{A} \in \mathbb{K}^{m,n} \quad \mapsto \quad \widetilde{\mathbf{A}} = \begin{bmatrix} \mathbf{a}_{1,\cdot} \\ \vdots \\ \mathbf{a}_{k-1,\cdot} \\ \mathbf{v}^T \\ \mathbf{a}_{k,\cdot} \\ \vdots \\ \mathbf{a}_{m,\cdot} \end{bmatrix}, \quad \mathbf{v} \in \mathbb{K}^n . \tag{3.2.22}$$ mod:row

Gegeben: QR-Zerlegung $\mathbf{A} = \mathbf{QR}$, $\mathbf{Q} \in \mathbb{K}^{m+1,m+1}$ unitär, $\mathbf{R} \in \mathbb{K}^{m,n}$ obere Dreiecksmatrix.

Aufgabe: Effiziente Berechnung der QR-Zerlegung $\widetilde{\mathbf{A}} = \widetilde{\mathbf{Q}}\widetilde{\mathbf{R}}$ von $\widetilde{\mathbf{A}}$ aus (3.2.22), $\widetilde{\mathbf{Q}} \in \mathbb{K}^{m+1,m+1}$ unitär, $\widetilde{\mathbf{R}} \in \mathbb{K}^{m+1,n+1}$ obere Dreiecksmatrix.

① $\exists$ Teilzyklische Zeilenpermutation $m+1 \leftarrow k$, $i \leftarrow i+1$, $i = k, \ldots, m$:
→ unitäre Permutationsmatrix $\mathbf{P} \in \{0,1\}^{m+1,m+1}$

$$\mathbf{P}\widetilde{\mathbf{A}} = \begin{pmatrix} \mathbf{A} \\ \mathbf{v}^T \end{pmatrix} \quad \blacktriangleright \quad \begin{pmatrix} \mathbf{Q}^H & 0 \\ 0 & 1 \end{pmatrix} \mathbf{P}\widetilde{\mathbf{A}} = \begin{pmatrix} \mathbf{R} \\ \mathbf{v}^T \end{pmatrix} =$$



.

Fall $m = n$

② Obere Dreiecksgestalt durch sukzessive Givens-Rotationen:

$$\begin{pmatrix} * & \cdots & & & \cdots & * \\ 0 & * & & & & \vdots \\ \vdots & 0 & \ddots & & & \\ \vdots & \vdots & & * & & \vdots \\ 0 & 0 & & 0 & * & * \\ * & \cdots & \cdots & * & * & * \end{pmatrix} \xrightarrow{\mathbf{G}_{1,m}} \begin{pmatrix} * & \cdots & & & \cdots & * \\ 0 & * & & & & \vdots \\ \vdots & 0 & \ddots & & & \\ \vdots & \vdots & & * & & \vdots \\ 0 & 0 & & 0 & * & * \\ 0 & * & \cdots & * & * & * \end{pmatrix} \xrightarrow{\mathbf{G}_{2,m}} \cdots$$

$$\cdots \xrightarrow{\mathbf{G}_{m-2,m}} \begin{pmatrix} * & \cdots & & & \cdots & * \\ 0 & * & & & & \vdots \\ \vdots & 0 & \ddots & & & \\ \vdots & \vdots & & * & & \vdots \\ 0 & 0 & & 0 & * & * \\ 0 & \cdots & & 0 & * & * \end{pmatrix} \xrightarrow{\mathbf{G}_{m-1,m}} \begin{pmatrix} * & \cdots & & & \cdots & * \\ 0 & * & & & & \vdots \\ \vdots & 0 & \ddots & & & \\ \vdots & \vdots & & * & & \vdots \\ 0 & 0 & & 0 & * & * \\ 0 & \cdots & & & 0 & * \end{pmatrix} := \widetilde{\mathbf{R}} \tag{3.2.23}$$ mod:row

③ Mit $\mathbf{Q}_1 = \mathbf{G}_{m-1,m} \cdot \cdots \cdot \mathbf{G}_{1,m}$

$$\widetilde{\mathbf{A}} = \mathbf{P}^T \begin{pmatrix} \mathbf{Q} & 0 \\ 0 & 1 \end{pmatrix} \mathbf{Q}_1^H \widetilde{\mathbf{R}} = \widetilde{\mathbf{Q}}\widetilde{\mathbf{R}} \quad \text{mit unitärem } \widetilde{\mathbf{Q}} \in \mathbb{K}^{m+1,m+1} .$$

☞ Ähnliche Update-Algorithmen für Streichen von Zeilen/Spalten

3.3 Numerische Berechnung von Eigenwerten und Eigenvektoren

[File: section-numerische-berechnung-von-eigenwerten-und-eigenvektoren.tex, SVN: section-numerische-berechnung-von-eigenwerten-und-eigenvektoren.te

NUMBERECHEIG

Beispiel 97 (Resonanzen elektrischer Netzwerke).

Gegeben: Grosses elektrisches Netzwerk mit linearen Bauelementen
Gesucht: Resonanzfrequenzen

Bsp. 66: ex:elnet Kirschhoffsche Knotenregel + Bauelementgleichungen ➡ Netzwerkmatrix

$$\mathbf{A}(\omega) := \mathbf{W} + i\omega\mathbf{C} - i\omega^{-1}\mathbf{S} \quad , \quad \mathbf{W},\mathbf{C},\mathbf{S} \in \mathbb{R}^{n,n} \text{ s.p.d. (→ Def. 3.2.8)} . \tag{3.3.1}$$ eq:NWM def:spd

Resonanzfrequenzen = $\omega \in \{\omega \in \mathbb{R}: \mathbf{A}(\omega) \text{ singulär}\}$.

Wie können diese ω numerisch berechnet werden (→ quadratisches Eigenwertproblem). ◇

3.3.1 Theorie und Kondition

Definition 3.3.1 (Eigenwerte und Eigenvektoren). def:ew

- $\lambda \in \mathbb{C}$ *Eigenwert (engl.* eigenvalue*) von* $\mathbf{A} \in \mathbb{K}^{n,n}$ $:\Leftrightarrow$ $\underbrace{\det(\lambda\mathbf{I} - \mathbf{A})}_{\text{Charakteristisches Polynom } \chi(\lambda)} = 0$ EIGWERT
- *Spektrum von* $\mathbf{A} \in \mathbb{K}^{n,n}$: $\sigma(\mathbf{A}) := \{\lambda \in \mathbb{C}: \lambda \text{ Eigenwert von } \mathbf{A}\}$ SPEKTRUM
- *Eigenraum (engl.* eigenspace*) zum Eigenwert* $\lambda \in \sigma(\mathbf{A})$: $\mathrm{Eig}_{\mathbf{A}}(\lambda) := \mathrm{Ker}(\lambda\mathbf{I} - \mathbf{A})$ EIGENRAUM
- $\mathbf{x} \in \mathrm{Eig}_{\mathbf{A}}(\lambda) \quad \Rightarrow \quad \mathbf{x}$ *ist Eigenvektor* EIGENVEKTOR
- *Geometrische Vielfachheit (engl.* multiplicity*) eines Eigenwerts* $\lambda \in \sigma(\mathbf{A})$: VIELFACHHEIT
$$m(\lambda) := \dim \mathrm{Eig}_{\mathbf{A}}(\lambda)$$

Lemma 3.3.2. *Das Spektrum einer Matrix ist invariant unter Ähnlichkeitstransformationen:* SIMILARTRAFO

$$\forall \mathbf{A} \in \mathbb{K}^{n,n}: \quad \sigma(\mathbf{S}^{-1}\mathbf{AS}) = \sigma(\mathbf{A}) \quad \forall \text{ regulären } \mathbf{S} \in \mathbb{K}^{n,n} .$$

SCHURLEMMA

Theorem 3.3.3 (Schursches Lemma).

$$\forall \mathbf{A} \in \mathbb{K}^{n,n}: \quad \exists \mathbf{U} \in \mathbb{C}^{n,n} \text{ unitär}: \quad \mathbf{U}^H\mathbf{AU} = \mathbf{T} \quad \text{mit } \mathbf{T} \in \mathbb{C}^{n,n} \text{ obere Dreiecksmatrix} .$$

Korollar 3.3.4 (Unitäre Diagonalisierbarkeit normaler Matrizen).

$$\mathbf{A} \in \mathbb{K}^{n,n},\ \mathbf{AA}^H = \mathbf{A}^H\mathbf{A}: \quad \exists \mathbf{U} \in \mathbb{C}^{n,n} \text{ unitär}: \quad \mathbf{U}^H\mathbf{AU} = \mathrm{diag}(\lambda_1,\ldots,\lambda_n), \quad \lambda_i \in \mathbb{C} .$$

Beachte:
- $\lambda_1,\ldots,\lambda_n$ = Eigenwerte von $\mathbf{A}$
- Spalten von $\mathbf{U}$ = Orthonormalbasis aus Eigenvektoren von $\mathbf{A}$

Matrizen $\mathbf{A} \in \mathbb{K}^{n,n}$ mit $\mathbf{AA}^H = \mathbf{A}^H\mathbf{A}$ heissen normal. NORMALMAT

Normal sind z.B.
- Hermitesche Matrizen: $\mathbf{A}^H = \mathbf{A}$ ➤ $\sigma(\mathbf{A}) \subset \mathbb{R}$ HERMMAT
- Unitäre Matrizen: $\mathbf{A}^H = \mathbf{A}^{-1}$ ➤ $|\sigma(\mathbf{A})| = 1$ UNITMAT
- Schiefhermitesche Matrizen: $\mathbf{A} = -\mathbf{A}^H$ ➤ $\sigma(\mathbf{A}) \subset i\mathbb{R}$ SCHIEFHERMMAT

Eigenwertprobleme:
- Gegeben $\mathbf{A} \in \mathbb{K}^{n,n}$ finde $\sigma(\mathbf{A})$.
- Gegeben $\mathbf{A} \in \mathbb{K}^{n,n}$ finde $\sigma(\mathbf{A})$ und alle Eigenvektoren.
- Gegeben $\mathbf{A} \in \mathbb{K}^{n,n}$ finde einige Eigenwerte und zugehörige Eigenvektoren.

Theorem 3.3.5 (Störungssatz für Eigenwerte). *Zu* $\mathbf{A} \in \mathbb{K}^{n,n}$ *gebe es* $\mathbf{S} \in \mathbb{C}^{n,n}$ *regulär, so dass* $\mathbf{S}^{-1}\mathbf{AS} = \mathbf{D}$, $\mathbf{D}$ *diagonal. Dann*

$$\forall \mathbf{E} \in \mathbb{K}^{n,n},\ \lambda \in \sigma(\mathbf{A}+\mathbf{E}): \quad \exists \lambda' \in \sigma(\mathbf{A}): \quad |\lambda - \lambda'| \le \mathrm{cond}_2(\mathbf{S})\,\|\mathbf{E}\|_2$$

Spalten von $\mathbf{S}$ = Eigenvektoren von $\mathbf{A}$

► Falls Eigenvektoren „fast" linear abhängig ($\Rightarrow \mathrm{cond}_2(\mathbf{S})$ gross), kann Eigenwertproblem sehr schlecht konditioniert sein

► Falls $\mathbf{A}$ normal ➤ EWP gut (absolut) konditioniert

Verfeinerte Analyse [19, Sect. 7.2.2], $\mathbf{A} \in \mathbb{K}^{n,n}$:

- $\lambda \in \mathbb{C}$ einfache Nullstelle von $\lambda \mapsto \det(\lambda \mathbf{I} - \mathbf{A})$ (einfacher Eigenwert)
- $\mathbf{x} \in \mathbb{C}^n$, $\|\mathbf{x}\|_2 = 1$ zugehöriger Eigenvektor (→ Def. 3.3.1),
- $\mathbf{y} \in \mathbb{C}^n$, $\|\mathbf{y}\|_2 = 1$, Linkseigenvektor zu λ, d.h. $\mathbf{y}^H \mathbf{A} = \lambda \mathbf{y}^H \Leftrightarrow \mathbf{A}^H \mathbf{y} = \lambda \mathbf{y}$.

► absolute Konditionszahl von λ bzgl. Euklidischer Matrixnorm & Störungen von $\mathbf{A}$ =

$$\kappa_{\text{abs}}(\lambda) := |\mathbf{y}^H \mathbf{x}|^{-1} . \tag{3.3.2}$$

Beispiel 98 (Störungsanfälligkeit von Eigenwerten).

$$\mathbf{A} = \begin{pmatrix} 1 & 2 & 3 \\ 0 & 4 & 5 \\ 0 & 0 & 4.001 \end{pmatrix} \quad \text{E.v. von } \mathbf{A}: \left\{ \begin{pmatrix} 1.0000 \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} 0.5547 \\ 0.8321 \\ 0 \end{pmatrix} \begin{pmatrix} 0.5547 \\ 0.8321 \\ 0.0002 \end{pmatrix} \right\} ,$$

$$\text{E.v. von } \mathbf{A}^T: \left\{ \begin{pmatrix} 0.8285 \\ -0.5523 \\ 0.0920 \end{pmatrix} \begin{pmatrix} 0 \\ 0.0002 \\ -1.0000 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1.0000 \end{pmatrix} \right\}$$

➤ $\kappa_{\text{abs}}(\lambda_1) = 1.2069722$,$\kappa_{\text{abs}}(\lambda_2) = 6009.25224$,$\kappa_{\text{abs}}(\lambda_3) = 6009.19059$

$$\mathbf{E} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 10^{-3} & 0 & 0 \end{pmatrix} \quad \sigma(\mathbf{A} + \mathbf{E}) = \{1.000111, 4.0581822, 3.9427066\} .$$

◇

Bemerkung 99. Die (absolute) Kondition eines Eigenvektors einer diagonalisierbaren Matrix $\mathbf{A} \in \mathbb{K}^{n,n}$ mit n verschiedenen Eigenwerten ist schlecht, wenn

- die Kondition des zugehörigen Eigenwertes λ^* schlecht ist,
- order wenn λ^* „schlecht separiert" $\min\{|\lambda^* - \lambda|, \lambda \in \sigma(\mathbf{A}) \setminus \{\lambda^*\}\} \ll |\lambda^*|$.

△

3.3.2 Transformationsmethoden

TRANSFOMETH

Bemerkung 100. Alle Verfahren zur numerischen Berechnung von Eigenwerten/Eigenvektoren sind Iterationsverfahren → Abschn. 2.1

△

Idee: Iteration via sukzessive **unitäre** Ähnlichkeitstransformationen

$$\mathbf{A} = \mathbf{A}^{(0)} \longrightarrow \mathbf{A}^{(1)} \longrightarrow \ldots \longrightarrow \begin{cases} \text{Diagonalmatrix} & \text{, falls } \mathbf{A} = \mathbf{A}^H , \\ \text{Obere Dreiecksmatrix} & \text{, sonst.} \end{cases}$$

(→ Thm. 3.3.3)

(Stabilitätsprobleme bei allgemeinen Ähnlichkeitstransformationen, Thm. 3.3.5)

► QR-Algorithmus (mit Shift): kubische Konvergenzordnung [19, Sect. 7.5,8.2]

Beispiel 101 (Unitäre Ähnlichkeitstransformation auf Tridiagonalgestalt).

$$\mathbf{A} = \begin{pmatrix} d_1 & b_1 & & & \alpha \\ b_1 & d_2 & \ddots & & \\ & \ddots & \ddots & b_{n-1} & \\ & & b_{n-1} & d_n & \\ \alpha & & & & \end{pmatrix} , \quad \mathbf{d} \in \mathbb{R}^n, \mathbf{b} \in \mathbb{R}^{n-1}, \alpha \in \mathbb{R} .$$

1. Schritt: Anulliere Matrixeinträge an Positionen $(n, 1)$ und $(1, n)$ durch Givens-Rotationen von rechts **und** links

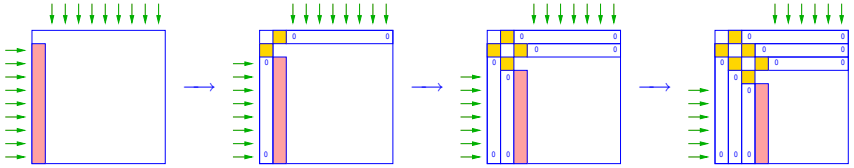
▷ MATLAB-Animation

$$\mathbf{G}_{2,n} \mathbf{A} \mathbf{G}_{2,n}^T = \begin{pmatrix} * & * & & & 0 \\ * & * & \ddots & & * & * \\ & \ddots & \ddots & * & & * \\ & & * & * & * & \\ 0 & & * & * & & \end{pmatrix} \rightarrow \begin{pmatrix} * & * & & & 0 \\ * & * & \ddots & & 0 & 0 \\ & \ddots & \ddots & * & * & * & * \\ & & * & * & * & * & \\ & 0 & * & * & & & \\ 0 & 0 & * & * & & & \end{pmatrix} \rightarrow \cdots$$

➤ Rechenaufwand, Speicheraufwand = $O(n^3)$! (da Dünnbesetzheit zerstört)

◇

Allgemeines Vorgehen: Sukzessive Householder-Ähnlichkeitstranstransformationen
(→ ≙ betroffene Zeilen/Spalten, ▭ ≙ Zielvektor)



Endlich viele **unitäre** Householder/Givens- **Ähnlichkeitstransformationen** können jede Hermitesche Matrix auf Tridiagonalform bringen (Rechenaufwand $O(n^3)$)

➜ Startwert $\mathbf{A}^{(0)}$ für iteratives Verfahren

MATLAB-Funktionen:
- Eigenwerte/normalisierte Eigenvektoren mit `[V,D] = eig(A);`
- Schur-Zerlegung mit `[U,T] = schur(A);`

RECHAUFWEIG
Rechenaufwand (elementare Operationen) für `eig()` (Ergebnis auf Maschinengenauigkeit):

Eigenwerte und Eigenvektoren von $\mathbf{A} \in \mathbb{K}^{n,n}$	$\sim 25n^3 + O(n^2)$
Nur Eigenwerte von $\mathbf{A} \in \mathbb{K}^{n,n}$	$\sim 10n^3 + O(n^2)$
Eigenwerte und Eigenvektoren von $\mathbf{A} = \mathbf{A}^H \in \mathbb{K}^{n,n}$	$\sim 9n^3 + O(n^2)$
Nur Eigenwerte von $\mathbf{A} = \mathbf{A}^H \in \mathbb{K}^{n,n}$	$\sim \frac{4}{3}n^3 + O(n^2)$
Nur Eigenwerte von tridiagonalem $\mathbf{A} = \mathbf{A}^H \in \mathbb{K}^{n,n}$	$\sim 30n^2 + O(n)$

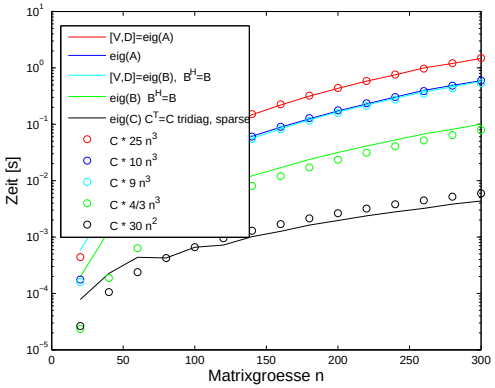
Beachte: Keine MATLAB-Implementierung von `eig` für dünnbesetzte Matrizen !

Ausnahme: `d=eig(A)` für *Hermitesche* dünnbesetzte Matrizen

Beispiel 102. (Komplexität von MATLAB-Eigenlösern)

Experiment beschreiben und Verwe
auf MATLAB-Code einfuegen. Leger
denbeschriftung kürzen und Erklärun
im Text.

◇



VERALLGEWP
Verallgemeinertes Eigenwertproblem (VEWP) (engl. *generalized eigenvalue problem*):

Gegeben: $\mathbf{A}, \mathbf{B} \in \mathbb{K}^{n,n}$

Gesucht: Verallgemeinerte Eigenwerte $\lambda \in \mathbb{C}$: $\mathbf{Ax} = \lambda\mathbf{Bx}$
Verallgemeinerte Eigenvektoren $\mathbf{x} \in \mathbb{K}^n \setminus \{0\}$

MATLAB-Funktion:
- `d = eig(A,B);` (→ nur Eigenwerte)
- `[V,D] = eig(A,B);` (→ Eigenwerte + -vektoren)

Insiderinformation: MATLAB benutzt QZ-Algorithmus [19, Sect. 7.7] (für vollbesetzte $\mathbf{A}, \mathbf{B}$) QZALGO GOL89

Falls $\mathbf{B}$ regulär ➢ VEWP formal äquivalent zu EWP für $\mathbf{B}^{-1}\mathbf{A}$
(kein Rezept zur effizienten numerischen Implementierung !)

Bemerkung 103. Falls $\mathbf{B} = \mathbf{B}^H$ s.p.d. mit Cholesky-Zerlegung $\mathbf{B} = \mathbf{R}^H\mathbf{R}$

$$\mathbf{Ax} = \lambda\mathbf{Bx} \quad \Leftrightarrow \quad \widetilde{\mathbf{A}}\mathbf{y} = \lambda\mathbf{y} \quad \text{mit } \widetilde{\mathbf{A}} := \mathbf{R}^{-H}\mathbf{A}\mathbf{R}^{-1},\ \mathbf{y} := \mathbf{Rx}\ .$$

➞ Algorithmisch relevant, da billiger als QZ. △

Beispiel 104 (Quadratisches Eigenwertproblem). (Fortsetzung von Bsp. 97) EX:NETWORKEWP

Finde $\mathbf{x} \neq 0$, $\omega \neq 0$, so dass

$$\mathbf{A}(\omega)\mathbf{x} = (\mathbf{W} + i\omega\mathbf{C} - i\omega^{-1}\mathbf{S})\mathbf{x} = 0\ . \tag{3.3.3}$$ eq:netwo

Substitution: $\mathbf{y} = -i\omega^{-1}\mathbf{x}$ TIM01 [44, Sect. 3.4]:

(3.3.3) eq:networkewp $\Leftrightarrow$ $\begin{pmatrix} \mathbf{W} & \mathbf{S} \\ \mathbf{I} & 0 \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} = -i\omega \begin{pmatrix} \mathbf{C} & 0 \\ 0 & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix}$

➢ Verallgemeinertes Eigenwertproblem ◇

Übung 3.4. (i) Welche Bedingung muss der Vektor $\mathbf{a} \in \mathbb{R}^{n-1}$ erfüllen, damit die Matrix

$$\mathbf{M} := \begin{pmatrix} 1 & \mathbf{a}^T \\ \mathbf{a} & \mathbf{I} \end{pmatrix} \in \mathbb{R}^{n,n}$$

positiv definit ist.

(ii) Schreibe eine effiziente MATLAB-Routine zur Berechnung der Eigenwerte für das verallgemeinerte Eigenwertproblem

$$\lambda \in \mathbb{R}, \mathbf{x} = (x_1, \ldots, x_n)^T \in \mathbb{R}^n \setminus \{0\}: \quad x_1\mathbf{e}_1 = \lambda\mathbf{Mx}\ ,$$

wobei $\mathbf{e}_1 := (1, 0, \ldots, 0)^T \in \mathbb{R}^n$.

3.4.1 Potenzmethoden

POTMETH
Spezielle Eigenwertprobleme für $\mathbf{A} \in \mathbb{K}^{n,n}$:

- Finde den betragsgrössten Eigenwert (+ Eigenvektor) von $\mathbf{A}$.
- Finde den betragskleinsten Eigenwert (+ Eigenvektor) von $\mathbf{A}$.
- Zu $\alpha \in \mathbb{C}$ finde $\lambda \in \sigma(\mathbf{A})$ mit $|\alpha - \lambda| = \min\{|\alpha - \mu|, \mu \in \sigma(\mathbf{A})\}$.

Idee für diagonalisierbares $\mathbf{A} \in \mathbb{K}^{n,n}$: $\mathbf{S}^{-1}\mathbf{A}\mathbf{S} = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$

$$\mathbf{z} = \sum_{j=1}^{n} \zeta_j \mathbf{s}_{\cdot,j} \quad \Rightarrow \quad \mathbf{A}^k\mathbf{z} = \sum_{j=1}^{n} \zeta_j \lambda_j^k \mathbf{s}_{\cdot,j} \,.$$

Falls $|\lambda_1| < |\lambda_2| < \cdots < |\lambda_n|$, $\left\|\mathbf{s}_{\cdot,j}\right\|_2 = 1$, $j = 1, \ldots, n$, $\zeta_n \neq 0$

$$\frac{\mathbf{A}^k\mathbf{z}}{\left\|\mathbf{A}^k\mathbf{z}\right\|} \to \pm\mathbf{s}_{\cdot,n} \quad = \text{Eigenvektor zu } \lambda_n \quad \text{für } k \to \infty \,. \tag{3.4.1}$$

ewp:pm

Motiviert direkte Potenzmethode (engl. *power method*): Iterationsverfahren (→ Abschn. 2.1)

sec:iterati

Startwert: $\mathbf{z}^{(0)}$ „beliebig" ,
Iterationsvorschrift: $\mathbf{w} := \mathbf{A}\mathbf{z}^{(k-1)}$, $\mathbf{z}^{(k)} := \frac{\mathbf{w}}{\|\mathbf{w}\|_2}$, $k = 1, 2, \ldots$.

Rechenaufwand: $1\times$ Matrix×Vektor pro Schritt ➢ Billig für dünnbesetzte Matrizen

Wie bekommt man den Eigenwert ?

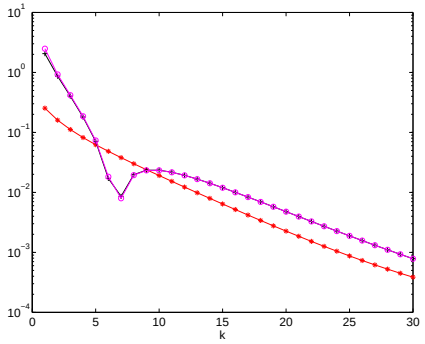
❶ Asymptotisch aus (3.4.1) ➢ $\mathbf{A}\mathbf{z}^{(k)} \approx \lambda_n \mathbf{z}^{(k)}$ ➢ $|\lambda_n| \approx \frac{\left\|\mathbf{A}\mathbf{z}^{(k)}\right\|}{\left\|\mathbf{z}^{(k)}\right\|}$

❷ Idee (für $\mathbf{A} = \mathbf{A}^H$): $\lambda_n \approx \operatorname*{argmin}_{\theta\in\mathbb{R}} \left\|\mathbf{A}\mathbf{z}^{(k)} - \theta\mathbf{z}^{(k)}\right\|_2^2$ ➢ $\lambda_n \approx \frac{(\mathbf{z}^{(k)})^H\mathbf{A}\mathbf{z}^{(k)}}{\left\|\mathbf{z}^{(k)}\right\|^2}$.

RAYLEIGHQUOT
Definition 3.4.1. *Für $\mathbf{A} \in \mathbb{K}^{n,n}$, $\mathbf{u} \in \mathbb{K}^n$ ist der Rayleigh-Quotient gegeben durch*

$$\rho_{\mathbf{A}}(\mathbf{u}) := \frac{\mathbf{u}^H\mathbf{A}\mathbf{u}}{\mathbf{u}^H\mathbf{u}} \,.$$

Beispiel 105 (Direkte Potenzmethode).




```
n = length(d);
S = triu(diag(n:-1:1,0)+...
    ones(n,n));
A = S*diag(d,0)*inv(S);
```

◁
o : Fehler $|\lambda_n - \rho_{\mathbf{A}}(\mathbf{z}^{(k)})|$
$*$: Fehlernorm $\left\|\mathbf{z}^{(k)} - \mathbf{s}_{\cdot,n}\right\|$
+ : $\left|\lambda_n - \frac{\left\|\mathbf{A}\mathbf{z}^{(k-1)}\right\|_2}{\left\|\mathbf{z}^{(k-1)}\right\|_2}\right|$

$\mathbf{z}^{(0)}$ = Zufallsvektor

Experimente:

① `d=(1:10)';` ➢ $|\lambda_{n-1}| : |\lambda_n| = 0.9$
② `d = [ones(9,1); 2];` ➢ $|\lambda_{n-1}| : |\lambda_n| = 0.5$
③ `d = = 1-2.^(-(1:0.5:5)');` ➢ $|\lambda_{n-1}| : |\lambda_n| = 0.9866$

$$\rho_{\mathrm{EV}}^{(k)} := \frac{\left\|\mathbf{z}^{(k)} - \mathbf{s}_{\cdot,n}\right\|}{\left\|\mathbf{z}^{(k-1)} - \mathbf{s}_{\cdot,n}\right\|} \,,$$

$$\rho_{\mathrm{EW}}^{(k)} := \frac{|\rho_{\mathbf{A}}(\mathbf{z}^{(k)}) - \lambda_n|}{|\rho_{\mathbf{A}}(\mathbf{z}^{(k-1)}) - \lambda_n|} \,.$$

	①		②		③	
k	$\rho_{\mathrm{EV}}^{(k)}$	$\rho_{\mathrm{EW}}^{(k)}$	$\rho_{\mathrm{EV}}^{(k)}$	$\rho_{\mathrm{EW}}^{(k)}$	$\rho_{\mathrm{EV}}^{(k)}$	$\rho_{\mathrm{EW}}^{(k)}$
22	0.9102	0.9007	0.5000	0.5000	0.9900	0.9781
23	0.9092	0.9004	0.5000	0.5000	0.9900	0.9791
24	0.9083	0.9001	0.5000	0.5000	0.9901	0.9800
25	0.9075	0.9000	0.5000	0.5000	0.9901	0.9809
26	0.9068	0.8998	0.5000	0.5000	0.9901	0.9817
27	0.9061	0.8997	0.5000	0.5000	0.9901	0.9825
28	0.9055	0.8997	0.5000	0.5000	0.9901	0.9832
29	0.9049	0.8996	0.5000	0.5000	0.9901	0.9839
30	0.9045	0.8996	0.5000	0.5000	0.9901	0.9844

Vermutung: Lineare Konvergenz (→ Def. 2.1.6)

def:cvglin

◇

Theorem 3.4.2 (Konvergenz der direkten Potenzmethode). *$\mathbf{A} \in \mathbb{K}^{n,n}$ habe einen betragsgrössten Eigenwert $\lambda_n > 0$ mit algebraischer Vielfachheit 1. Seien $\mathbf{v}, \mathbf{y}$ Rechts- bzw. Linkseigenvektoren von $\mathbf{A}$ zu λ_n mit $\|\mathbf{y}\|_2 = \|\mathbf{v}\|_2 = 1$. Dann konvergieren*

$$\left\|\mathbf{A}\mathbf{z}^{(k)}\right\|_2 \to \lambda_n \quad , \quad \mathbf{z}^{(k)} \to \pm\mathbf{v} \quad \textit{linear mit Rate} \quad \frac{|\lambda_{n-1}|}{|\lambda_n|} \,,$$

wobei $\mathbf{z}^{(k)}$ die Iterierten der direkten Potenzmethode sind und $\mathbf{y}^H\mathbf{z}^{(0)} \neq 0$ vorausgesetzt wird.

Bemerkung 106. Rundungsfehler ➤ $\mathbf{y}^H\mathbf{z}^{(0)} \neq 0$ „numerisch” immer erfüllt △

Bemerkung 107. Abbruchkriterium für Potenziteration ? (→ Abschn. 2.1) ABBRUCHKRITPOTIT sec:iterationsverfahren

„Relative Änderung” $\leq$ tol:

$$\begin{cases} \left\|\mathbf{z}^{(k)} - \mathbf{z}^{(k-1)}\right\| \leq \texttt{tol}\,, \\ \left| \frac{\left\|\mathbf{A}\mathbf{z}^{(k)}\right\|}{\left\|\mathbf{z}^{(k)}\right\|} - \frac{\left\|\mathbf{A}\mathbf{z}^{(k-1)}\right\|}{\left\|\mathbf{z}^{(k-1)}\right\|} \right| \leq (1/L - 1)\texttt{tol} \end{cases}$$ vgl. (2.1.2) . eq:apost

Geschätzte Konvergenzrate

△

Wie findet man den betragskleinsten Eigenwert (+ Eigenvektor) von $\mathbf{A}$?

Falls $\mathbf{A} \in \mathbb{K}^{n,n}$ regulär:

Betragskleinster EW von $\mathbf{A}$ = $\left(\text{Betragsgrösster EW von } \mathbf{A}^{-1}\right)^{-1}$

➤ Direkte Potenzmethode für $\mathbf{A}^{-1}$ = Inverse Iteration INVIT

Verallgemeinerung auf „Zu $\alpha \in \mathbb{C}$ finde $\lambda \in \sigma(\mathbf{A})$ mit $|\alpha - \lambda| = \min\{|\alpha - \mu|, \mu \in \sigma(\mathbf{A})\}$”:
INVITSHIFT
Inverse Iteration mit Shift:

$$\mathbf{z}^{(0)} \text{ beliebig}\,, \quad \mathbf{w} = (\mathbf{A} - \alpha\mathbf{I})^{-1}\mathbf{z}^{(k-1)} \quad , \quad \mathbf{z}^{(k)} := \frac{\mathbf{w}}{\|\mathbf{w}\|_2}\,, \quad k = 1, 2, \ldots\,. \tag{3.4.2}$$ ewp:invits

Dabei: $(\mathbf{A} - \alpha\mathbf{I})^{-1}\mathbf{z}^{(k-1)} \hat{=}$ Löse $(\mathbf{A} - \alpha\mathbf{I})\mathbf{w} = \mathbf{z}^{(k-1)}$ basierend auf Gausselimination:
einmalige LU-Zerlegung von $\mathbf{A} - \alpha\mathbf{I}$!

Und was passiert, wenn „versehentlich” $\alpha \in \sigma(\mathbf{A})$ (→ $\mathbf{A} - \alpha\mathbf{I}$ singulär) ?

„Magie der Gausselimination”:

Die Gauss-Elimination mit Spaltenpivotsuche (bzw. LU-Zerlegung/Cholesky-Zerlegung + Vorwärtsrechnung + Rücksubstitution) angewandt auf das LGS $\mathbf{A}\mathbf{x} = \mathbf{b}$ berechnet (fast immer → Bsp. 75) ein $\widetilde{\mathbf{x}}$ für das das Residuum $\mathbf{r} := \mathbf{b} - \mathbf{A}\widetilde{\mathbf{x}}$ „klein” ist: $\|\mathbf{r}\| \approx$ eps! ex:Wilkinson

Sei $\mathbf{A} = \mathbf{A}^H \in \mathbb{K}^{n,n}$ mit $\mathbf{S}^H\mathbf{A}\mathbf{S} = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$, $\mathbf{S}$ unitär, $0 < |\lambda_1| \ll |\lambda_j|$, $j = 2, \ldots, n$

Eigenvektorentwicklung $\mathbf{b} = \sum_{j=1}^n \beta_j \mathbf{s}_{\cdot,j}$, $\widetilde{\mathbf{x}} = \sum_{j=1}^n \xi_j \mathbf{s}_{\cdot,j}$ ➤ $\mathbf{r} := \mathbf{b} - \mathbf{A}\widetilde{\mathbf{x}} = \sum_{j=1}^n (\beta_j - \lambda_j\xi_j)\mathbf{s}_{\cdot,j}$

$\|\mathbf{r}\|_2$ „klein” ➤ $\begin{cases} \xi_j \approx \frac{\beta_j}{\lambda_j}\,, & j = 2, \ldots, n\,, \\ \xi_1 \text{ „gross”}. \end{cases}$

➤ Für $\mathbf{A} = \mathbf{A}^H \in \mathbb{K}^{n,n}$: Rundungsfehler bei der Gauss-Elimination können nur grosse Fehler in Richtung des Eigenvektors zum betragskleinsten Eigenwert zur Folge haben.

Bemerkung 108. Falls tatsächlich einmal Pivotelement $= 0$ ➤ ersetze es durch eps ! △

Thm. 3.4.2 thm:pm ➤ Für inverse Iteration (mit Shift):
Asymptotisch lineare Konvergenz, Rayleigh-Quotient → λ_j mit Rate

$$\frac{|\lambda_j - \alpha|}{\min\{|\lambda_i - \alpha|, i \neq j\}} \quad \text{mit} \quad \lambda_j \in \sigma(\mathbf{A})\,, \quad |\alpha - \lambda_j| \leq |\alpha - \lambda| \quad \forall \lambda \in \sigma(\mathbf{A})\,.$$

➤ Sehr schnell für $\alpha \approx \lambda_j$!

Idee: A posteriori Anpassung des Shifts

Benutze $\alpha := \rho_{\mathbf{A}}(\mathbf{z}^{(k-1)})$ im k. Schritt

RAYLEIGHQUOTIT
Rayleigh-Quotienten-Iteration

(für normale $\mathbf{A} \in \mathbb{K}^{n,n}$)

erhält Dünnbesetztheit, siehe Abschnitt 3.1.2 sec:matr-speich

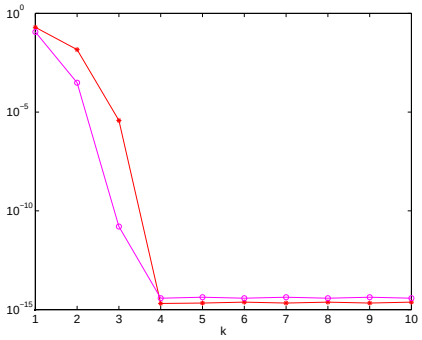
MATLAB-CODE : Rayleigh-Quotienten-Iteration

```
function [z,lmin] = rqui(A,tol,maxit)
alpha = 0; n = size(A,1);
z = rand(size(A,1),1); z = z/norm(z);
for i=1:maxit
  z = (A-alpha*speye(n))\z;
  z = z/norm(z); lmin=dot(A*z,z);
  if (abs(alpha-lmin) < tol), break; end;
  alpha = lmin;
end
```

(3.4.3) rqui

Nachteil: Kein Recycling von LU-Zerlegungen möglich (im Vergleich zu (3.4.2)) ewp:invitshift

Beispiel 109 (Rayleigh-Quotienten-Iteration).




```
d = (1:10)';
n = length(d);
Z = diag(sqrt(1:n),0)+ones(n,n);
[Q,R] = qr(Z);
A = Q*diag(d,0)*Q';
```

o : $|\lambda_{\min} - \rho_{\mathbf{A}}(\mathbf{z}^{(k)})|$

$*$: $\left\|z^{(k)} - \mathbf{x}_{\cdot,j}\right\|$, $\lambda_{\min} = \lambda_j$

k	$\lvert\lambda_{\min} - \rho_{\mathbf{A}}(\mathbf{z}^{(k)})\rvert$	$\left\|z^{(k)} - \mathbf{x}_{\cdot,j}\right\|$
1	0.09381702342056	0.20748822490698
2	0.00029035607981	0.01530829569530
3	0.00000000001783	0.00000411928759
4	0.00000000000000	0.00000000000000
5	0.00000000000000	0.00000000000000

thm:rqui

Theorem 3.4.3. *Für* $\mathbf{A} = \mathbf{A}^H$ *konvergiert* $\rho_{\mathbf{A}}(\mathbf{z}^{(k)})$ *für die Iterierten* $\mathbf{z}^{(k)}$ *aus der Rayleigh-Quotienten-Iteration* (3.4.3) rqui *lokal kubisch gegen den betragskleinsten Eigenwert.*

◇

Was tun, wenn man mehrere Eigenwerte $\lambda_{n-k}, \lambda_{n-k+1}, \ldots, \lambda_n$, $k \ll n$, und zugehörige Eigenvektoren von $\mathbf{A} \in \mathbb{K}^{n,n}$ benötigt **?**

DIRPOTITUR

Direkte Potenziteration im Unterraum (für Hermitesche Matrizen $\mathbf{A} \in \mathbb{K}^{n,n}$)

MATLAB-CODE : Potenzmethode im Unterraum

```
function [V,ev] = spowit(A,k,m)
n = size(A,1); V = rand(n,m); d = zeros(k,1);
for i=1:maxit
  V=A*V;
[Q,R] = qr(V,0);
  T=Q'*A*Q;
  [S,D] = eig(T); [l,perm] = sort(-abs(diag(D)));
  V = Q*S(:,perm);
  if (norm(d+l(1:k)) < tol), break; end;
  d = -l(1:k);
end
V = V(:,1:k); ev = diag(D(perm(1:k),perm(1:k)));
```

(3.4.4) Spowit

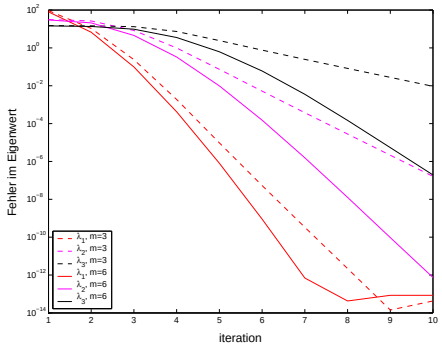
RITZPROJ

Ritz-Projektion

Verallgemeinerung der Normierung auf $\|\mathbf{z}\|_2 = 1$

➔ Analog: Unterraumvarianten der inversen Iteration/Rayleigh-Quotienten-Iteration

Beispiel 110 (Direkte Potenziteration im Unterraum).



S.p.d. Testmatrix: $a_{ij} := \min\{\frac{i}{j}, \frac{j}{i}\}$

```
n=200; A = gallery('lehmer',n);
```

„Anfangsmatrix":

```
V = eye(n,m);
```

- Beobachtung:
  Lineare Konvergenz der Eigenwerte
- Wahl $m > k$ beschleunigt Konvergenz der Eigenwerte

◇

3.4.2 Vorkonditionierte inverse Iteration

VORKONDINVIT

Aufgabe: Für *symmetrisch positiv definites* $\mathbf{A} \in \mathbb{K}^{n,n}$ (→ Def. 3.2.8) def:spd finde betragskleinsten Eigenwert (+ zugehörigen Eigenvektor).

Optionen: Inverse Iteration, Rayleigh-Quotienten-Iteration (3.4.3) rqui

Problem: Trotz sparse-Eliminationstechniken (→ 3.2.6): sec:bandmatrizen Direkte LU-Zerlegung (→ Abschn. 3.2.3) sec:die-lr-z oft unmöglich für $n \gg 1$.

Beachte: Inverse Iteration ➢ *Näherungslösung* des LGS in (3.4.2) ausreichend **!** ewp:invitshift

Idee: (für) Inverse Iteration ohne Shift

Statt Lösung von $\mathbf{A}\mathbf{w} = \mathbf{z}^{(k-1)}$: $\mathbf{w} = \mathbf{B}^{-1}\mathbf{z}^{(k-1)}$ mit "billiger" s.p.d. approximativer Inversen $\mathbf{B}^{-1} \approx \mathbf{A}^{-1}$

Kann man in der inversen Iteration $\mathbf{A}^{-1}$ einfach durch $\mathbf{B}^{-1}$ ersetzen **?**

NEIN, denn kleinster Eigenwert von $\mathbf{B}$ interessiert nicht!

Ersetzung $\mathbf{A}^{-1} \to \mathbf{B}^{-1}$ nur möglich bei Anwendung auf ein Residuum

RESIDUUM

Residuum = Grösse, die $\to 0$, wenn Konvergenz gegen gesuchte Lösung

Kandidat: Residuum für Eigenwertproblem $\mathbf{Ax} = \lambda\mathbf{x}$:

$$\mathbf{r} := \mathbf{Az} - \rho_{\mathbf{A}}(\mathbf{z})\mathbf{z} \quad , \quad \rho_{\mathbf{A}}(\mathbf{z}) = \text{Rayleigh-Quotient} \rightarrow \text{Def. 3.4.1} .$$

def:rq

Beachte: Nur Richtung von $\mathbf{A}^{-1}\mathbf{z}$ relevant in inverser Iteration (3.4.2)

ewp:invitshift

$$(\mathbf{A}^{-1}\mathbf{z}) \parallel (\mathbf{z} - \mathbf{A}^{-1}(\mathbf{Az} - \rho_{\mathbf{A}}(\mathbf{z})\mathbf{z})) \quad \Rightarrow \quad \text{definiert gleiche nächste Iterierte !}$$

[Vorkonditionierte inverse Iteration für s.p.d. $\mathbf{A}$]

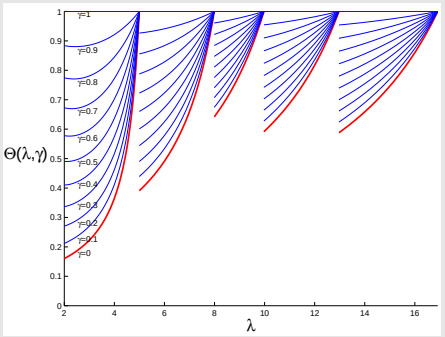
$\mathbf{z}^{(0)}$ beliebig,

$$\begin{aligned} \mathbf{w} &= \mathbf{z}^{(k-1)} - \mathbf{B}^{-1}(\mathbf{Az}^{(k-1)} - \rho_{\mathbf{A}}(\mathbf{z}^{(k-1)})\mathbf{z}^{(k-1)}) , \\ \mathbf{z}^{(k)} &= \frac{\mathbf{w}}{\|\mathbf{w}\|_2} , \end{aligned} \qquad k = 1, 2, \ldots . \quad (3.4.5)$$

eq:pcinvi

Theorie:
- Lineare Konvergenz von (3.4.5)
- Konvergenzgeschwindigkeit gross, wenn $\lambda_{\max}(\mathbf{B}^{-1}\mathbf{A}) : \lambda_{\min}(\mathbf{B}^{-1}\mathbf{A})$ klein

eq:pcinvit



Beispiel: ($\mathbf{A}$ aus Bsp. 112)

ex:gspoisson

`gallery('poisson',100)` (skaliert)

$\lambda := \rho_{\mathbf{A}}(\mathbf{z}^{(k-1)})$

$\lambda^* = \max\{\mu \in \sigma(\mathbf{A}), \mu \leq \lambda\}$

Plot zeigt theoretische Schranke [34, 33] für

NEY99c,NEY99

$$\Theta(\lambda, \rho) = \frac{|\rho_{\mathbf{A}}(\mathbf{z}^{(k)}) - \lambda^*|}{|\lambda - \lambda^*|} ,$$

$$\rho := \left\| \mathbf{I} - \mathbf{B}^{-1}\mathbf{A} \right\|_A . \quad (3.4.6)$$

binvit:rate

Notation: Die von einer s.p.d. Matrix $\mathbf{A} \in \mathbb{K}^{n,n}$ induzierte Energienorm ist

$$\|\mathbf{x}\|_A := (\mathbf{x}^H\mathbf{Ax})^{1/2} , \quad \mathbf{x} \in \mathbb{K}^n .$$

Beachte: Matrix-Energienorm: $\|\mathbf{M}\|_A := \sup_{\mathbf{x}\in\mathbb{K}^n\setminus\{0\}} \frac{\|\mathbf{Mx}\|_A}{\|\mathbf{x}\|_A} = \left\|\mathbf{A}^{1/2}\mathbf{MA}^{-1/2}\right\|_2$

Aus Lemma 3.4.6 folgt:

lem:pc

$$0 < \gamma < 1: \quad \gamma\,(\mathbf{x}^H\mathbf{Bx}) \leq \mathbf{x}^H\mathbf{Ax} \leq (\mathbf{x}^H\mathbf{Bx}) \quad \forall \mathbf{x} \in \mathbb{K}^n \quad \Rightarrow \quad \left\| \mathbf{I} - \mathbf{B}^{-1}\mathbf{A} \right\|_A \leq 1 - \gamma .$$

VORKOND

Konzept 3.4.4 (Vorkonditionierer). *Eine s.p.d. Matrix $\mathbf{B} \in \mathbb{K}^{n,n}$ heisst Vorkonditionierer (engl. preconditioner) für die s.p.d. Matrix $\mathbf{A} \in \mathbb{K}^{n,n}$, falls*

1. *$\exists 0 < \gamma < \Gamma$ „klein”: $\gamma\,(\mathbf{x}^H\mathbf{Bx}) \leq \mathbf{x}^H\mathbf{Ax} \leq \Gamma\,(\mathbf{x}^H\mathbf{Bx}) \quad \forall \mathbf{x} \in \mathbb{K}^n$ und*
2. *die Auswertung von $\mathbf{B}^{-1}\mathbf{x}$ erfordert ungefähr den gleichen Rechenaufwand wie die Matrix×Vektor-Multiplikation $\mathbf{Ax}$, $\mathbf{x} \in \mathbb{K}^n$.*

Ein Vorkonditionierer liefert eine „billige” approximative Inverse von $\mathbf{A}$

SPEKCOND

Definition 3.4.5 (Spektrale Kondition). *Spektrale Kondition für reguläres $\mathbf{A} \in \mathbb{K}^{n,n}$*

$$\kappa(\mathbf{A}) = \frac{\max\{|\sigma(\mathbf{A})|\}}{\min\{|\sigma(\mathbf{A})|\}} .$$

Lemma 3.4.6. *Für s.p.d. $\mathbf{A}, \mathbf{B} \in \mathbb{K}^{n,n}$*

$$0 < \gamma < \Gamma: \quad \gamma\,(\mathbf{x}^H\mathbf{Bx}) \leq \mathbf{x}^H\mathbf{Ax} \leq \Gamma\,(\mathbf{x}^H\mathbf{Bx}) \quad \forall \mathbf{x} \in \mathbb{K}^n \quad \Rightarrow \quad \kappa(\mathbf{B}^{-1}\mathbf{A}) \leq \frac{\Gamma}{\gamma} .$$

Beispiele für Vorkonditionierer (**Black-Box-Methoden**):

- Diagonalvorkonditionier (Jacobi-Vorkonditionierer): $\mathbf{B} = \mathrm{diag}(a_{11}, \ldots, a_{nn})$

SYMGSVORK

- Symmetrischer Gauss-Seidel-Vorkonditionierer Idee: Approximatives Lösen von $\mathbf{Ax} = \mathbf{b}$ in zwei Stufen

① Approximiere $\mathbf{A}^{-1} \approx \texttt{tril}(\mathbf{A})$ (unterer Dreiecksanteil): $\widetilde{\mathbf{x}} = \texttt{tril}(\mathbf{A})^{-1}\mathbf{b}$

② Approximiere $\mathbf{A}^{-1} \approx \texttt{triu}(\mathbf{A})$ (oberer Dreiecksanteil) und „löse” damit die Fehlergleichung $\mathbf{A}(\mathbf{x} - \widetilde{\mathbf{x}}) = \mathbf{r}$, mit Residuum $\mathbf{r} := \mathbf{b} - \mathbf{A}\widetilde{\mathbf{x}}$:

$$\mathbf{x} = \widetilde{\mathbf{x}} + \texttt{triu}(\mathbf{A})^{-1}(\mathbf{b} - \mathbf{A}\widetilde{\mathbf{x}}) .$$

Mit $\mathbf{L}_A := \texttt{tril}(\mathbf{A})$, $\mathbf{U}_A := \texttt{triu}(\mathbf{A})$ findet man

$$\mathbf{x} = (\mathbf{L}_A^{-1} + \mathbf{U}_A^{-1} - \mathbf{U}_A^{-1}\mathbf{AL}_A^{-1})\mathbf{b} \quad \blacktriangleright \quad \mathbf{B} = \mathbf{L}_A^{-1} + \mathbf{U}_A^{-1} - \mathbf{U}_A^{-1}\mathbf{AL}_A^{-1} .$$

Beispiel 111 (Jacobi-Vorkonditionierung).

Matrix mit „schwerer” Diagonalen

$$\mathbf{A} = \mathbf{A}^T \in \mathbb{R}^{n,n}: \quad \exists \delta \in]0,1[: \quad \delta a_{ii} \geq \sum_{j \neq i}^{n} |a_{ij}|. \tag{3.4.7}$$

J:adef

$$\blacktriangleright \quad \sigma(\mathbf{A}) \subset [(1-\delta)\min_i\{a_{ii}\}, (1+\delta)\max_i\{a_{ii}\}] \, .$$

Lemma 3.4.7 (Gerschgorin-Kreise). *Für* $\mathbf{A} \in \mathbb{K}^{n,n}$ *gilt*

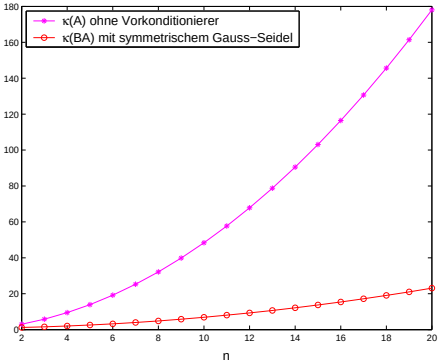
$$\sigma(\mathbf{A}) \subset \bigcup_{j=1}^{n} \{z \in \mathbb{C}: |z - a_{jj}|\} \leq \sum_{i \neq j} |a_{ji}|\} \, .$$

Für $\mathbf{A}$ aus (3.4.7) (Jacobi-Vorkonditionierer = „Equilibrierung”)

$$\kappa(\mathrm{diag}(a_{11}, \ldots, a_{nn})^{-1}\mathbf{A}) \subset [1-\delta, 1+\delta] \, .$$

◇

Beispiel 112 (Gauss-Seidel-Vorkonditionierung).



$$\mathbf{A} = \begin{pmatrix} \mathbf{T} & -\mathbf{I} & 0 & \cdots & \cdots & 0 \\ -\mathbf{I} & \mathbf{T} & -\mathbf{I} & & & \vdots \\ 0 & \ddots & \ddots & \ddots & & \\ \vdots & & & -\mathbf{I} & \mathbf{T} & -\mathbf{I} \\ 0 & \cdots & \cdots & 0 & -\mathbf{I} & \mathbf{T} \end{pmatrix} \in \mathbb{R}^{n^2,n^2} \, ,$$

$$\mathbf{T} = \begin{pmatrix} 4 & -1 & 0 & \cdots & \cdots & 0 \\ -1 & 4 & -1 & & & \vdots \\ 0 & \ddots & \ddots & \ddots & & \\ \vdots & & & -1 & 4 & -1 \\ 0 & \cdots & \cdots & 0 & -1 & 4 \end{pmatrix} \in \mathbb{R}^{n,n} \, .$$

◇

UNVOLLSTCHOL

- Unvollständige Cholesky-Zerlegung (engl. *incomplete Cholesky decomposition*)

„Cholesky-Zerlegung” mit Unterdrückung von Fill-In (→ def:fillin, 3.2.14, sec:bandmatrizen, Abschn. 3.2.6)

MATLAB-Funktion: `R=cholinc(A,'0');` (nur sinnvoll für dünnbesetzte Matrizen !)

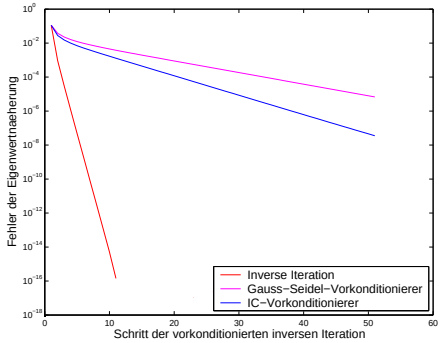
Beispiel 113 (Konvergenz der vorkonditionierten inversen Iteration).

Konvergenz der Eigenwertnäherung aus der vorkonditionierten inversen Iteration für

- `A = gallery('poisson',30);` (s.p.d.)

und

- $\mathbf{B} = \mathbf{A}$
- $\mathbf{B} \leftarrow$ Gauss-Seidel-Vorkonditionierer
- $\mathbf{B} \leftarrow$ `cholinc(A,'0');`



◇

3.4.3 Krylov-Unterraumverfahren

Gegeben: (Grosse, diagonalisierbare) Matrix $\mathbf{A} \in \mathbb{K}^{n,n}$: $\mathbf{S}^{-1}\mathbf{A}\mathbf{S} = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$
(Eigenwerte sortiert: $|\lambda_1| \leq |\lambda_2| \leq \cdots \leq |\lambda_n|$)

Eigenwertproblem: Finde $\lambda_1, \ldots, \lambda_l$ (und Eigenvektoren $\mathbf{s}_{:,1}, \ldots, \mathbf{s}_{:,l}$) bzw.
Finde $\lambda_{n-l+1}, \ldots, \lambda_n$ (und $\mathbf{s}_{:,n-l+1}, \ldots, \mathbf{s}_{:,n}$) , $l \ll n$

➤ Option: Potenzmethoden (direkt, invers), Abschnitte 3.4.1, 3.4.2 sec:direkte-iter sec:inverse-iter

RITZ

Idee der Ritz-Projektion, vgl. (3.4.4): spowit

❑ $\mathbf{s}_{:,m} \in V_l := \mathrm{Im}(\mathbf{V}) \subset \mathbb{K}^n$, $1 \leq m \leq n$, $\mathbf{V} \in \mathbb{K}^{n,l}$, $\Rightarrow$ $\lambda_m \in \sigma(\mathbf{V}^H\mathbf{A}\mathbf{V})$
$1 \leq l \leq n$, $\{\mathbf{v}_{:,1}, \ldots, \mathbf{v}_{:,l}\}$ *Orthonormalbasis* von V_l

❒ Falls V_l „signifikante Komponenten in Richtung von $\mathbf{s}_{:,m}$” $\Rightarrow$ $\exists \mu \in \sigma(\mathbf{V}^H\mathbf{A}\mathbf{V})$, $\mu \approx \lambda_m$ enthält

Aus Stabilitätsgründen !

Wie bekommt man Unterraum mit „signifikanten Komponenten in Richtung von $\mathbf{s}_{:,n-l+1}, \ldots, \mathbf{s}_{:,n}$” ?

Tipp: Unterraumversion der direkten Potenzmethode (3.4.4) spowit

$$V_l^{(0)} \text{ beliebig} \quad , \quad V_l^{(k)} := \mathbf{A}V_l^{(k-1)} \quad , \quad k = 1, 2, \ldots \, .$$

Wie bekommt man Unterraum mit „signifikanten Komponenten in Richtung von $\mathbf{s}_{:,1}, \ldots, \mathbf{s}_{:,l}$” ?

Tipp: $\sigma(\|\mathbf{A}\|_2\mathbf{I}-\mathbf{A}) = \|\mathbf{A}\|_2 - \sigma(\mathbf{A})$

($\mathbf{A}$ s.p.d.: Betragskleinste Eigenwerte von $\mathbf{A}$ = betragsgrösste Eigenwerte von $\|\mathbf{A}\|_2\mathbf{I}-\mathbf{A}$)

Definition 3.4.8. *Für $\mathbf{A}\in\mathbb{K}^{n,n}$, $\mathbf{z}\in\mathbb{K}^n$, $\mathbf{z}\neq 0$, ist der l. Krylov-Unterraum*

$$\mathcal{K}_l(\mathbf{A},\mathbf{z}) := \operatorname{Span}\left\{\mathbf{z},\mathbf{A}\mathbf{z},\ldots,\mathbf{A}^{l-1}\mathbf{z}\right\} .$$

Äquivalent: $\mathcal{K}_l(\mathbf{A},\mathbf{z}) = \{p(\mathbf{A})\mathbf{z}: p \text{ Polynom vom Grad } \leq l\}$

Sukzessive Konstruktion einer Orthonormalbasis (ONB) von $\mathcal{K}_l(\mathbf{A},\mathbf{z})$: ARNOLDI Arnoldi-Prozess

Induktives Vorgehen: $\{\mathbf{v}_1,\ldots,\mathbf{v}_m\}$ ONB von $\mathcal{K}_m(\mathbf{A},\mathbf{z})$ $\forall 1\leq m\leq l$

➤ $\mathbf{A}\mathbf{v}_l \in \mathcal{K}_{l+1}(\mathbf{A},\mathbf{z})$ (warum **?**)

$$\Rightarrow\quad \widetilde{\mathbf{v}}_{l+1} := \mathbf{A}\mathbf{v}_l - \sum_{j=1}^{l}(\mathbf{v}_j^H\mathbf{A}\mathbf{v}_l)\,\mathbf{v}_j\,,\quad \mathbf{v}_{l+1} := \frac{\widetilde{\mathbf{v}}_{l+1}}{\|\widetilde{\mathbf{v}}_{l+1}\|_2}\quad\Rightarrow\quad \mathbf{v}_{l+1}\perp\mathcal{K}_l(\mathbf{A},\mathbf{z})\,. \qquad (3.4.8)$$

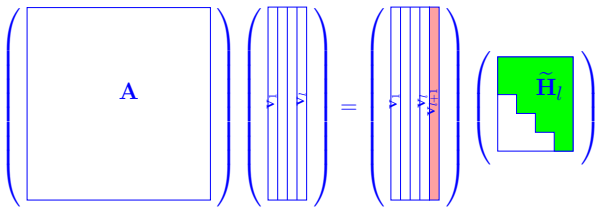
GRAMSCHMIDT

(vgl. Gram-Schmidt-Orthonormalisierung → lineare Algebra)

orthogonal

$$\mathbf{V}_l = [\mathbf{v}_1,\ldots,\mathbf{v}_l]\ :\ \mathbf{A}\mathbf{V}_l = \mathbf{V}_{l+1}\widetilde{\mathbf{H}}_l\quad,\quad \widetilde{\mathbf{H}}_l\in\mathbb{K}^{l+1,l} \text{ mit } \widetilde{h}_{ij} = \begin{cases}\mathbf{v}_i^H\mathbf{A}\mathbf{v}_j & \text{, falls } i\leq j\,,\\ \|\widetilde{\mathbf{v}}_j\|_2 & \text{, falls } i=j+1\,,\\ 0 & \text{sonst.}\end{cases}$$

➡ $\widetilde{\mathbf{H}}_l$ = obere Hessenbergmatrizen



Stabile Implementierung des Arnoldi-Prozesses:

Modifizierter Gram-Schmidt

Im l. Schritt:

$1\times$	$\mathbf{A}\times$Vektor
$l+1$	Skalarprodukte
l	SAXPYs
n	Divisionen

➢ Rechenaufwand für l Schritte, wenn $\mathbf{A}$ maximal k Einträge $\neq 0$ pro Zeile: $O(nkl^2)$

`H(l+1,l) = 0` ➞ Abbruch !

MATLAB-CODE: Arnoldi-Prozess

```
function [V,Ht] = arnoldi(A,k,v0)
V = [v0/norm(v0)];
Ht = zeros(k+1,k);
for l=1:k
  vt = A*V(:,l);
  for j=1:l
    Ht(j,l) = dot(V(:,j),vt);
    vt = vt - Ht(j,l)*V(:,j);
  end
  Ht(l+1,l) = norm(vt);
  V = [V, vt/Ht(l+1,l)];
end
```

(3.4.9) arnoldi

Lemma 3.4.9. *Für die Matrizen $\mathbf{V}_l\in\mathbb{K}^{n,l}$, $\widetilde{\mathbf{H}}_l\in\mathbb{K}^{l+1,l}$ aus dem l. Schritt , $l\leq n$, des Arnoldi-Verfahrens gilt*

(i) $\mathbf{V}_l^H\mathbf{V}_l = \mathbf{I}$,

(ii) $\mathbf{A}\mathbf{V}_l = \mathbf{V}_{l+1}\widetilde{\mathbf{H}}_l$, $\widetilde{\mathbf{H}}_l$ *obere Hessenbergmatrix,*

(iii) $\mathbf{V}_l^H\mathbf{A}\mathbf{V}_l = \mathbf{H}_l\in\mathbb{K}^{l,l}$, $h_{ij}=\widetilde{h}_{ij}$ *für* $1\leq i,j\leq l$,

(iv) *Falls* $\mathbf{A}=\mathbf{A}^H$ ➤ $\mathbf{H}_l$ *tridiagonal.*

ewp:Gra

Abbruch des Arnoldi-Prozesses (3.4.9) arnoldi im l. Schritt ➢ $\operatorname{Span}\{v_1,\ldots,v_l\}$ invarianter Teilraum von $\mathbf{A}$.

(Praxis: rechne in diesem Fall mit „beliebigem” `vt` weiter)

Beachte: (iii) $\Leftrightarrow$ $\mathbf{H}_l$ ist Ritz-Projektion von $\mathbf{A}$ auf $\mathcal{K}_l(\mathbf{A},\mathbf{z})$.

Arnoldi-Prozess zur Eigenwertapproximation:

Im l. Schritt: $\lambda_n\approx\mu_l^{(l)}$, $\lambda_{n-1}\approx\mu_{l-1}^{(l)},\ldots,\lambda_1\approx\mu_1^{(l)}$,

$\sigma(\mathbf{H}_l) = \{\mu_1^{(l)},\ldots,\mu_l^{(l)}\}$, $|\mu_1^{(l)}|\leq|\mu_2^{(l)}|\leq\cdots\leq|\mu_l^{(l)}|$.

MATLAB-CODE Arnoldi-Verfahren

```
function [dn,V,Ht] = arnolidev(A,v0,k,tol)
n = size(A,1); V = [v0/norm(v0)];
H = zeros(1,0); dn = zeros(k,1);
for l=1:n
  d = dn;
  Ht = [Ht, zeros(l,1); zeros(1,l)];
  vt = A*V(:,l);
  for j=1:l
    Ht(j,l) = dot(V(:,j),vt);
    vt = vt - Ht(j,l)*V(:,j);
  end
  ev = sort(eig(Ht(1:l,1:l)));
  dn(1:min(l,k)) = ev(end:-1:end-min(l,k)+1);
  if (norm(d-dn) < tol*norm(dn)), break; end;
  Ht(l+1,l) = norm(vt);
  V = [V, vt/Ht(l+1,l)];
end
```

Arnoldi-Verfahren
zur Berechnung der k, $k \leq n$
betragsgrössten Eigenwerte von
$\mathbf{A} \in \mathbb{K}^{n,n}$

Pro Schritt nur eine Multiplikation mit Matrix $\mathbf{A}$! (Effizient für dünnbesetzte Matrizen)

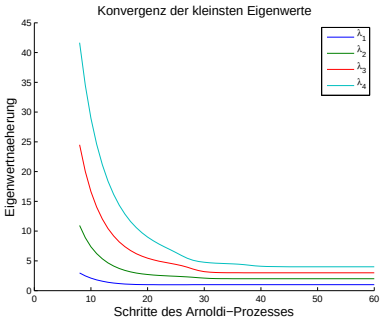
Monoton wachsender Speicheraufwand !

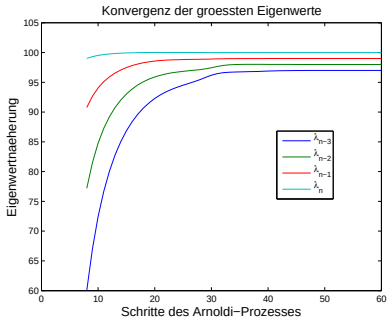
Beispiel 114 (Arnoldi-Verfahren zur Eigenwertberechnung).

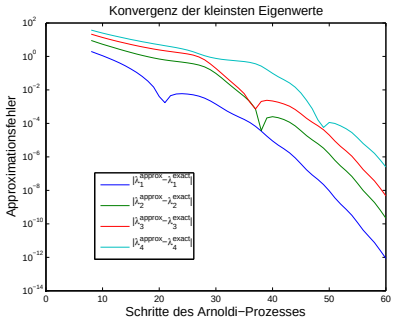
```
n=100;
M=gallery('tridiag',-0.5*ones(n-1,1),2*ones(n,1),-1.5*ones(n-1,1));
[Q,R]=qr(M);
v=ones(n,1);  v(2:2:end)=2*v(2:2:end);
S=diag(v)*Q; A=S*diag(1:n)*inv(S);
```

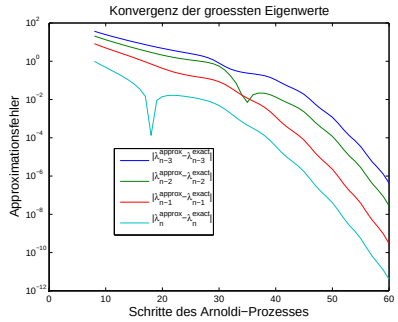
Startvektor: `ones(100,1)`:

$$\sigma(\mathbf{A}) = \{1, 2, 3, 4, \ldots, 97, 98, 99, 100\}$$









◇

Im Fall $\mathbf{A} = \mathbf{A}^H \in \mathbb{K}^{n,n}$ ➤ $\mathbf{H}_l = \mathbf{H}_l^H$ ➤ kurze Rekursionen: LANCZOS Lanczos-Prozess

Lemma 3.4.9 (lem:arnoldi) ➤ l. Schritt, $1 \leq l \leq n$ des Lanczos-Prozesses erzeugt $\mathbf{V}_l \in \mathbb{K}^{n,l}$ mit orthonormierten Spalten, so dass

$$\mathbf{V}_l^H \mathbf{A} \mathbf{V}_l = \begin{pmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \beta_2 & \alpha_3 & \ddots & \\ & & \ddots & \ddots & \\ & & & & \ddots \\ & & & \ddots & \ddots & \beta_{k-1} \\ & & & & \beta_{k-1} & \alpha_k \end{pmatrix} =: \mathbf{T}_l \in \mathbb{K}^{k,k} \quad \text{[Tridiagonalmatrix]}$$

Lanczos-Prozess

Rechenaufwand pro Schritt:

$1\times$	$\mathbf{A}\times$Vektor
2	Skalarprodukte
2	SAXPYs
1	Division

➢ Rechenaufwand für l Schritte, wenn $\mathbf{A}$ maximal k Einträge $\neq 0$ pro Zeile: $O(nkl)$

`beta(l+1) = 0` ➞ Abbruch !

Speicherbedarf: 2 Vektoren Länge n, wenn V nicht berechnet wird.

MATLAB-CODE Lanczos-Prozess

```
function [V,alpha,beta] = lanczos(A,k,w)
V = [w/norm(w)];
alpha = zeros(k,1);
beta = zeros(k,1);
for l=1:k
  vt = A*V(:,l);
  if (l>1)
    vt = vt - beta(l)*V(:,l-1);
  end
  alpha(l) = dot(V(:,l),vt);
  vt = vt - alpha(l)*V(:,l);
  beta(l+1) = norm(vt);
  V = [V, vt/beta(l+1)];
end
beta = beta(2:end-1);
```

Lanczos-Prozess zur Eigenwertapproximation: (Theorie [12, Abschn. 8.5])

Im l. Schritt: $\lambda_n \approx \mu_l^{(l)}$, $\lambda_{n-1} \approx \mu_{l-1}^{(l)}, \ldots, \lambda_1 \approx \mu_1^{(l)}$,

$\sigma(\mathbf{T}_l) = \{\mu_1^{(l)}, \ldots, \mu_l^{(l)}\}$, $|\mu_1^{(l)}| \leq |\mu_2^{(l)}| \leq \cdots \leq |\mu_l^{(l)}|$.

Beispiel 115 (Rundungsfehlereinfluss beim Lanczos-Prozess).

$\mathbf{A} \in \mathbb{R}^{10,10}$, $a_{ij} = \min\{i,j\}$. `A = gallery('minij',10);`

Berechnet durch `[V,alpha,beta] = lanczos(A,n,ones(n,1));`:

$$\mathbf{T} = \begin{pmatrix}
38.500000 & 14.813845 & & & & & & & & \\
14.813845 & 9.642857 & 2.062955 & & & & & & & \\
 & 2.062955 & 2.720779 & 0.776284 & & & & & & \\
 & & 0.776284 & 1.336364 & 0.385013 & & & & & \\
 & & & 0.385013 & 0.826316 & 0.215431 & & & & \\
 & & & & 0.215431 & 0.582380 & 0.126781 & & & \\
 & & & & & 0.126781 & 0.446860 & 0.074650 & & \\
 & & & & & & 0.074650 & 0.363803 & 0.043121 & \\
 & & & & & & & 0.043121 & 3.820888 & 11.991094 \\
 & & & & & & & & 11.991094 & 41.254286
\end{pmatrix}$$

$\sigma(\mathbf{A}) = \{0.255680, 0.273787, 0.307979, 0.366209, 0.465233, 0.643104, 1.000000, 1.873023, 5.048917, 44.766069\}$

$\sigma(\mathbf{T}) = \{0.263867, 0.303001, 0.365376, 0.465199, 0.643104, 1.000000, 1.873023, 5.048917, 44.765976, 44.766069\}$

➢ Mysteriöser Cluster von Eigenwerten von $\mathbf{T}$ („ghost eigenvalues” [19, Sect. 9.2.5])

$$\mathbf{V}^H\mathbf{V} = \begin{pmatrix}
1.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000251 & 0.258801 & 0.883711 \\
0.000000 & 1.000000 & -0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000106 & 0.109470 & 0.373799 \\
0.000000 & -0.000000 & 1.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000005 & 0.005373 & 0.018347 \\
0.000000 & 0.000000 & 0.000000 & 1.000000 & -0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000096 & 0.000328 \\
0.000000 & 0.000000 & 0.000000 & -0.000000 & 1.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000001 & 0.000003 \\
0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 1.000000 & -0.000000 & 0.000000 & 0.000000 & 0.000000 \\
0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & -0.000000 & 1.000000 & -0.000000 & 0.000000 & 0.000000 \\
0.000251 & 0.000106 & 0.000005 & 0.000000 & 0.000000 & 0.000000 & -0.000000 & 1.000000 & -0.000000 & 0.000000 \\
0.258801 & 0.109470 & 0.005373 & 0.000096 & 0.000001 & 0.000000 & 0.000000 & -0.000000 & 1.000000 & 0.000000 \\
0.883711 & 0.373799 & 0.018347 & 0.000328 & 0.000003 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 1.000000
\end{pmatrix}$$

➢ Orthogonalitätsverlust der Basisvektoren durch Rundungsfehlereinfluss (Lemma 3.4.9 nur von „theoretischem Wert”)

l	$\sigma(\mathbf{T}_l)$									
1										38.500000
2									3.392123	44.750734
3								1.117692	4.979881	44.766064
4							0.597664	1.788008	5.048259	44.766069
5						0.415715	0.925441	1.870175	5.048916	44.766069
6					0.336507	0.588906	0.995299	1.872997	5.048917	44.766069
7				0.297303	0.431779	0.638542	0.999922	1.873023	5.048917	44.766069
8			0.276160	0.349724	0.462449	0.643016	1.000000	1.873023	5.048917	44.766069
9		0.276035	0.349451	0.462320	0.643006	1.000000	1.873023	3.821426	5.048917	44.766069
10	0.263867	0.303001	0.365376	0.465199	0.643104	1.000000	1.873023	5.048917	44.765976	44.766069

➢ Gute Approximation der betragsgrössten Eigenwerte für $l < n$

Analytisch äquivalent: Arnoldi-Prozess angewandt auf $\mathbf{A}$, versuchen wir es mal:

l	$\sigma(\mathbf{T}_l)$								
1									38.500000
2								3.392123	44.750734
3							1.117692	4.979881	44.766064
4						0.597664	1.788008	5.048259	44.766069
5					0.415715	0.925441	1.870175	5.048916	44.766069
6				0.336507	0.588906	0.995299	1.872997	5.048917	44.766069
7			0.297303	0.431779	0.638542	0.999922	1.873023	5.048917	44.766069
8		0.276159	0.349722	0.462449	0.643016	1.000000	1.873023	5.048917	44.766069
9	0.263872	0.303009	0.365379	0.465199	0.643104	1.000000	1.873023	5.048917	44.766069
10	0.255680	0.273787	0.307979	0.366209	0.465233	0.643104	1.000000	1.873023	5.048917	44.766069

Perfekt !!

Beispiel 116 (Eigenwertberechnung mit Lanczos/Arnoldi-Prozess).

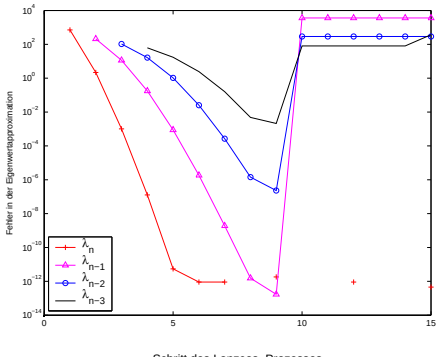
Lanczos-Verfahren:

```
A = gallery('minij',100);
```

Startvektor: `ones(1,100)`

$\lambda_l(\mathbf{T}_l), \lambda_{l-1}(\mathbf{T}_l), \ldots$

Gute Konvergenz
bevor „ghost eigenvalues" auftreten.

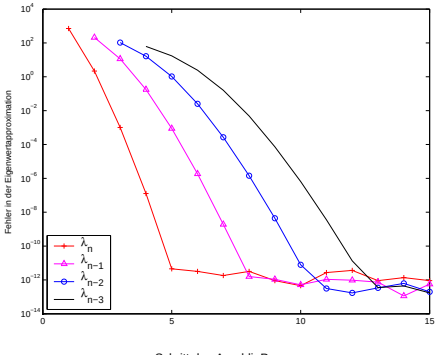


Arnoldi-Verfahren:

(Matrix/Startvektor wie oben)

$\lambda_l(\mathbf{T}_l), \lambda_{l-1}(\mathbf{T}_l), \ldots$

Lanczos-Prozess rundungsfehleranfällig
Arnoldi-Prozess wesentlich stabiler
(Keine „ghost eigenvalues")



Krylov-Unterraum-Verfahren basierend auf Arnoldi-/Lanczos-Prozess sind die Methoden der Wahl zur Berechnung von wenigen Eigenwerten/Eigenvektoren für *grosse dünnbesetzte Matrizen*

Bemerkung 117. Erweiterung der Krylov-Unterraum-Verfahren auf verallgemeinertes Eigenwertproblem $\mathbf{Ax} = \lambda\mathbf{Bx}$, $\mathbf{B}$ s.p.d., durch Ersetzung des Euklidischen Skalarprodukts mit „$\mathbf{B}$-Skalarprodukt" $(\mathbf{x}, \mathbf{y}) \mapsto \mathbf{x}^H\mathbf{By}$.

MATLAB-Funktion:

`d = eigs(A,k,sigma)` : k grösste/kleinste Eigenwerte von $\mathbf{A}$

`d = eigs(A,B,k,sigma)`: k grösste/kleinste Eigenwerte für verallgemeinertes Eigenwertproblem $\mathbf{Ax} = \lambda\mathbf{Bx}$,$\mathbf{B}$ s.p.d.

`d = eigs(Afun,n,k)` : `Afun` = Funktionshandle für Anwendung von $\mathbf{A}$/$\mathbf{A}^{-1}$/$\mathbf{A} - \alpha\mathbf{I}$/$(\mathbf{A} - \alpha\mathbf{B})^{-1}$ auf Vektor. Eigenschaften der durch `Afun` repräsentierten Matrix müssen durch Flaggen spezifiziert werden.

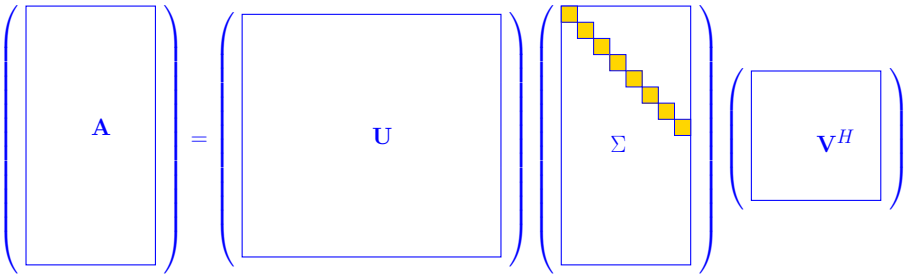
3.4.4 Singulärwertzerlegungen

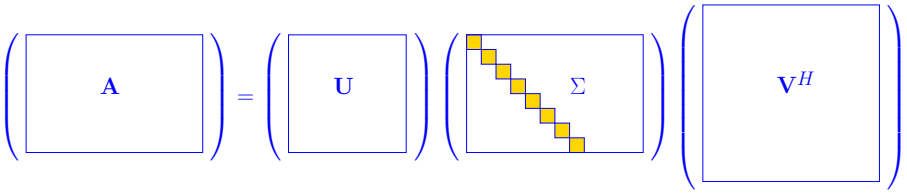
SINGWERTZERL

= Diagonalisierung durch separate unitäre Basistransformationen in Bild- und Urbildbereich

Theorem 3.4.10. *Zu jedem $\mathbf{A} \in \mathbb{K}^{m,n}$ gibt es unitäre Matrizen $\mathbf{U} \in \mathbb{K}^{m,m}$, $\mathbf{V} \in \mathbb{K}^{n,n}$ und eine Diagonalmatrix $\Sigma = \operatorname{diag}(\sigma_1, \ldots, \sigma_p) \in \mathbb{R}^{m,n}$, $p := \min\{m, n\}$, $\sigma_1 \geq \sigma_2 \geq \sigma_p \geq 0$ mit*

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^H .$$





Definition 3.4.11. *Die Zerlegung* $\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^H$ *aus Satz 3.4.10 heisst Singulärwertzerlegung (SVD) mit Singulärwerten* σ_i.

Singulärwertzerlegung = Zerlegung in Rang-1-Komponenten:

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^H = \sum_{j=1}^{p} \sigma_j \mathbf{u}_{\cdot,j}\mathbf{v}_{\cdot,j}^H \,. \tag{3.4.10}$$

Lemma 3.4.12. *Die Quadrate* σ_i^2 *der nichtverschwindenden Singulärwerte von* $\mathbf{A}$ *sind Eigenwerte von* $\mathbf{A}^H\mathbf{A}$, $\mathbf{A}\mathbf{A}^H$ *und* $\begin{pmatrix} 0 & \mathbf{A}^H \\ \mathbf{A} & 0 \end{pmatrix}$ *zu den Eigenvektoren* $\mathbf{v}_{\cdot,1}, \ldots, \mathbf{v}_{\cdot,p}$, $\mathbf{u}_{\cdot,1}, \ldots, \mathbf{u}_{\cdot,p}$, *bzw.* $\begin{pmatrix} \mathbf{u}_{\cdot,1} \\ \mathbf{v}_{\cdot,1} \end{pmatrix}, \ldots, \begin{pmatrix} \mathbf{u}_{\cdot,p} \\ \mathbf{v}_{\cdot,p} \end{pmatrix}$.

Thm. 3.3.5 ➢ $|\sigma_k(\mathbf{A}) - \sigma_k(\mathbf{A}+\mathbf{E})| \le \|\mathbf{E}\|_2$

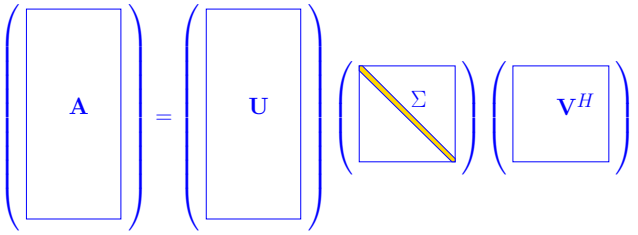
Das Problem der Berechnung der Singulärwerte ist immer hervorragend konditioniert

Singulärwerte von $\mathbf{A}$ sind eindeutig bestimmt

MATLAB-Funktionen:

`s = svd(A)`	: Berechnet Singulärwerte der Matrix $\mathbf{A}$
`[U,S,V] = svd(A)`	: Berechnet Singulärwertzerlegung gemäss Thm. 3.4.10
`[U,S,V] = svd(A,0)`	: Sparsame Singulärwertzerlegung für $m > n$: : $\mathbf{U} \in \mathbb{K}^{m,n}$, $\Sigma \in \mathbb{R}^{n,n}$, $\mathbf{V} \in \mathbb{K}^{n,n}$
`s = svds(A,k)`	: k grösste Singulärwerte (sinnvoll für dünnbesetztes $\mathbf{A}$)
`[U,S,V] = svds(A,k)`	: Partielle Singulärwertzerlegung: $\mathbf{U} \in \mathbb{K}^{m,k}$, $\mathbf{V} \in \mathbb{K}^{n,k}$, $\Sigma \in \mathbb{R}^{k,k}$ diagonal, enthält k grösste Singulärwerte von $\mathbf{A}$.

Sparsame Singulärwertzerlegung:



SVD-Algorithmus ist in jedem Fall numerisch stabil

RECHAUFWSVD
Rechenaufwand:

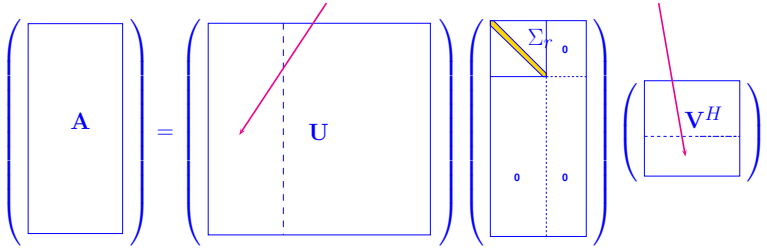
$2mn^2 + 2n^3 + O(n^2) + O(mn)$ für `s = svd(A)`,
$4m^2n + 22n^3 + O(mn) + O(n^2)$ für `[U,S,V] = svd(A)`,
$O(mn^2) + O(n^3)$ für `[U,S,V]=svd(A,0)`, $m \gg n$.

- Anwendung: NUMRANGBEST
  Numerische Rangbestimmung

Lemma 3.4.13 (SVD und Rang einer Matrix)**.** *Erfüllen die Singulärwerte von* $\mathbf{A}$

$$\begin{aligned} &\sigma_1 \ge \cdots \ge \sigma_r > \sigma_{r+1} = \cdots \sigma_p = 0 \,, \\ &\operatorname{rank}(\mathbf{A}) = r \quad , \\ &\operatorname{Ker}(\mathbf{A}) = \operatorname{Span}\left\{\mathbf{v}_{\cdot,r+1}, \ldots, \mathbf{v}_{\cdot,n}\right\} \quad , \\ &\operatorname{Im}(\mathbf{A}) = \operatorname{Span}\left\{\mathbf{u}_{\cdot,1}, \ldots, \mathbf{u}_{\cdot,r}\right\} \,. \end{aligned}$$

Graphische Darstellung: Spalten = ONB von $\mathrm{Im}(\mathbf{A})$ Zeilen = ONB von $\mathrm{Ker}(\mathbf{A})$



Anmerkung: MATLAB-Funktion r=rank(A) benutzt SVD zur Rangberechnung.

• Berechnung von Matrixnormen

Lemma 3.4.14 (SVD und Euklidische Matrixnorm).

$$\forall \mathbf{A} \in \mathbb{K}^{m,n}\colon \ \|\mathbf{A}\|_2 = \sigma_1(\mathbf{A}) \ , \quad \forall \mathbf{A} \in \mathbb{K}^{n,n} \textit{ regulär}\colon \ \mathrm{cond}_2(\mathbf{A}) = \sigma_1/\sigma_n \ .$$

Anmerkung: MATLAB-Funktion `norm(A)` und `cond(A)` benutzen SVD.

• Anwendung: *Bestapproximation durch Niedrigrangmatrizen* (engl. *low rank approximation*)

FROBNORM

Definition 3.4.15 (Frobeniusnorm). *Die Frobeniusnorm von $\mathbf{A} \in \mathbb{K}^{m,n}$ ist definiert durch*

$$\|\mathbf{A}\|_F^2 := \sum_{i=1}^{m}\sum_{j=1}^{n} |a_{ij}|^2 \ .$$

Frobeniusnorm und SVD: $\|\mathbf{A}\|_F^2 = \sum_{j=1}^{p} \sigma_j^2$

Beachte: $\|\mathbf{A}\|_F$ invariant bei unitären Transformationen.

Notation: $\mathcal{R}_k(m,n) := \{\mathbf{A} \in \mathbb{K}^{m,n}\colon \ \mathrm{rank}(\mathbf{A}) \le k\}$, $m,n,k \in \mathbb{N}$

Theorem 3.4.16 (Niedrigrang-Bestapproximation). *Sei $\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^H$ die SVD von $\mathbf{A} \in \mathbb{K}^{m.n}$. Setze für $1 \le k \le \mathrm{rank}(\mathbf{A})$ $\mathbf{U}_k := [\mathbf{u}_{.,1}, \ldots, \mathbf{u}_{.,k}]$, $\mathbf{V}_k := [\mathbf{v}_{.,1}, \ldots, \mathbf{v}_{.,k}]$, $\Sigma_k := \mathrm{diag}(\sigma_1, \ldots, \sigma_k)$. Dann gilt für $\|\cdot\| = \|\cdot\|_F$ und $\|\cdot\| = \|\cdot\|_2$*

$$\left\|\mathbf{A} - \mathbf{U}_k\Sigma_k\mathbf{V}_k^H\right\| \le \|\mathbf{A} - \mathbf{F}\| \quad \forall \mathbf{F} \in \mathcal{R}_k(m,n) \ .$$

Beispiel 118 (Hauptkomponentenanalyse).

$\mathbf{A} \in \mathbb{R}^{m,n}$, $m \gg n$:

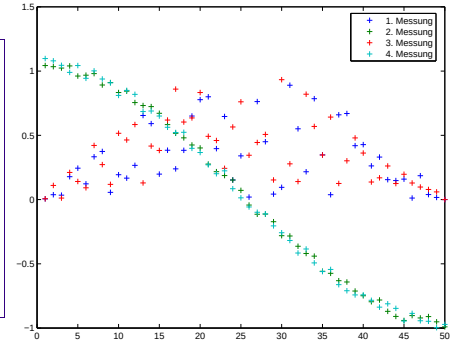
Zeilen von $\mathbf{A}$ → Messung mehrerer Grössen
Spalten von $\mathbf{A}$ → Messgrössen

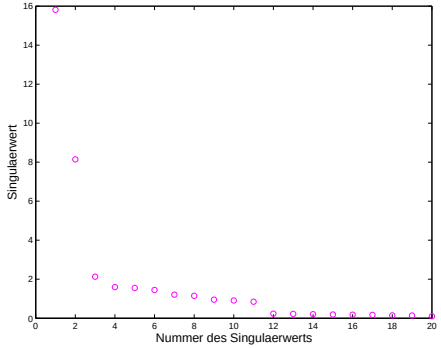
Ziel: Identifikation linearer Korrelationen

Im Jahresverlauf werden zwei verschiedene Grössen an 10 verschiedenen Orten gemessen:

(Die Messungen sind fehlerbehaftet)

```
n = 10;
m = 50;
x = sin(pi*(1:m)'/m);
y = cos(pi*(1:m)'/m);
A = [];
for i = 1:n
  A = [A, x.*rand(m,1),...
       y+0.1*rand(m,1)];
end
```



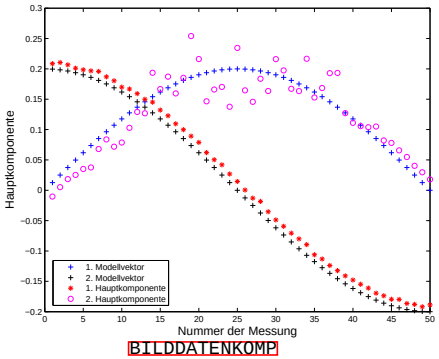


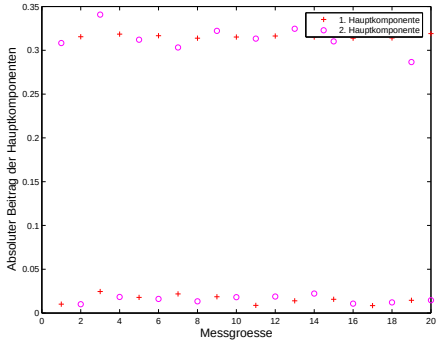
← Verteilung der Singulärwerte der Messmatrix

Zwei dominante Singulärwerte

Messungen linear korreliert mit **zwei** Hauptkomponenten.

Hauptkomponenten = $\mathbf{u}_{.,1}, \mathbf{u}_{.,2}$ (führende Spalten von $\mathbf{U}$ in SVD)
Beiträge = $\mathbf{v}_{.,1}, \mathbf{v}_{.,2}$ (führende Spalten von $\mathbf{V}$ in SVD)





BILDDATENKOMP

Beispiel 119 (Bilddatenkompression). $m \times n$-Pixelbild (Graustufen, BMP-Format) $\leftrightarrow$ Matrix $\mathbf{A} \in \mathbb{R}^{m,n}$, $a_{ij} \in \{0, \ldots, 255\}$

Niedrigrang-Bestapproximation des Bildes: $\widetilde{\mathbf{A}} = \mathbf{U}_k \Sigma_k \mathbf{V}^T$, Thm. 3.4.16 (→ Animation)

thm:rankapprox


```
[img_data] = imread(<file>.jpg');
img_data = double(img_data);
image(img_data);
[U,S,V] = svd(img_data(:,:,1));
s = diag(S);
k=20;
img_data_comp = U(:,1:k)*diag(s(1:k))*V(:,1:k)';
image(img_data_comp);
col = [0:1/215:1]'*[1,1,1];
colormap(col);
```

Zugegeben: Es gibt bessere Verfahren (JPEG, Wavelets, etc.) als SVD.

3.5 Numerik linearer Ausgleichsprobleme

[File: section-numerik-linearer-ausgleichsprobleme.tex, SVN: section-numerik-linearer-ausgleichsprobleme.tex 1199 2006-12-05 06:48:09Z hiptmair]

LSQ

Lineares Ausgleichsproblem, „Methode der kleinsten Quadrate" (engl. *linear least squares*):

Gegeben: $\mathbf{A} \in \mathbb{K}^{m,n}$, $m, n \in \mathbb{N}$, $\mathbf{b} \in \mathbb{K}^m$
Gesucht: $\mathbf{x} \in \mathbb{K}^n$ mit

(i) $\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2 = \inf\{\|\mathbf{A}\mathbf{y} - \mathbf{b}\|_2 : \mathbf{y} \in \mathbb{K}^n\}$, (3.5.1) eq:LSQ

(ii) $\|\mathbf{x}\|_2$ minimal unter Bedingung (i)

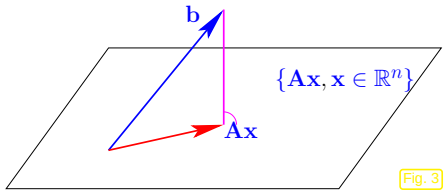


Fig. 3

Geometrische Interpretation:

$\mathbf{x} \hat{=}$ Orthogonalprojektion von $\mathbf{b}$ auf Unterraum $\mathrm{Span}\{\mathbf{a}_{\cdot,1}, \ldots, \mathbf{a}_{\cdot,n}\}$

Lemma 3.5.1. *Existenz & Eindeutigkeit von Lösungen des Linearen Ausgleichsproblems*

PSEUDOINV

Definition 3.5.2 (Pseudoinverse). *Die Pseudoinverse $\mathbf{A}^+ \in \mathbb{K}^{n,m}$ von $\mathbf{A} \in \mathbb{K}^{m,n}$ ist die Matrixdarstellung des (linearen) Lösungsoperators $\mathbb{R}^m \mapsto \mathbb{R}^n$, $\mathbf{b} \mapsto \mathbf{x}$ zum linearen Ausgleichsproblem* (eq:LSQ 3.5.1) $\|\mathbf{Ax} - \mathbf{b}\| \to \min$, $\|\mathbf{x}\| \to \min$.

MATLAB:
`x = A\b` („backslash”) löst (eq:LSQ 3.5.1) für $\mathbf{A} \in \mathbb{K}^{m,n}$, $m \neq n$
`P = pinv(A)` berechnet Pseudoinverse

Beispiel 120 (Lineares Fitten von Daten, lineare Regression). → Bsp. 59 (ex:NLLSQ)

Gegeben: „Messpunkte” $(t_i, y_i) \in \mathbb{K}^2$, $i = 1, \ldots, m$, $t_i \in I \subset \mathbb{K}$
Basisfunktionen $b_j : I \mapsto \mathbb{K}$, $j = 1, \ldots, n$.
Gesucht: Koeffizienten $x_j \in \mathbb{K}$, $j = 1, \ldots, n$, so dass

$$\sum_{i=1}^{m} |f(t_i) - y_i|^2 \to \min \quad , \quad f(t) := \sum_{j=1}^{n} x_j b_j(t) \, .$$

► Lineares Ausgleichsproblem mit $\mathbf{A} = (b_i(t_j))$, $1 \leq i \leq n$, $1 \leq j \leq m$.

Hier: $\|\mathbf{b} - \mathbf{Ax}\|_2$ misst Qualität des Fit-Modells.

Spezialfall: Polynomiales Fitten: $b_j(t) = t^{j-1}$
MATLAB-Funktion: `p = polyfit(t,y,n);` , n = Polynomgrad ◇

Kondition des linearen Ausgleichsproblems:

VERALLGKOND

Definition 3.5.3 (Verallgemeinerte Kondition einer Matrix, → Def. 3.2.5 (def:cond)). *Seien $\sigma_1 \geq \sigma_2 \geq \sigma_r > \sigma_{r+1} = \ldots = \sigma_p = 0$, $p := \min\{m, n\}$, die singulären Werte (→ Def. 3.4.11 (def:svd)) von $\mathbf{A} \in \mathbb{K}^{m,n}$. Dann ist*

$$\mathrm{cond}_2(\mathbf{A}) := \frac{\sigma_1}{\sigma_r}$$

die verallgemeinerte Kondition (bzgl. der 2-Norm) von $\mathbf{A}$.

Theorem 3.5.4. *Für $m \geq n$, $\mathbf{A} \in \mathbb{K}^{m,n}$, $\mathrm{rank}(\mathbf{A}) = n$, löse $\mathbf{x} \in \mathbb{K}^n$ das lineare Ausgleichsproblem $\|\mathbf{Ax} - \mathbf{b}\| \to \min$ und $\widehat{\mathbf{x}}$ das lineare Ausgleichsproblem $\|(\mathbf{A} + \Delta\mathbf{A})\widehat{\mathbf{x}} - \mathbf{b}\| \to \min$. Dann gilt*

$$\frac{\|\mathbf{x} - \widehat{\mathbf{x}}\|_2}{\|\mathbf{x}\|_2} \lesssim \left(2\,\mathrm{cond}_2(\mathbf{A}) + \mathrm{cond}_2^2(\mathbf{A}) \frac{\|\mathbf{r}\|_2}{\|\mathbf{A}\|_2 \, \|\mathbf{x}\|_2} \right) \frac{\|\Delta\mathbf{A}\|_2}{\|\mathbf{A}\|_2} \, ,$$

mit Residuum $\mathbf{r} = \mathbf{Ax} - \mathbf{b}$.

Falls $\|\mathbf{r}\|_2 \ll 1$ ➢ Kondition(Lineares Ausgleichsproblem) $\approx \mathrm{cond}_2(\mathbf{A})$
Falls $\|\mathbf{r}\|_2$ „gross” ➢ Kondition(Lineares Ausgleichsproblem) $\approx \mathrm{cond}_2^2(\mathbf{A})$

3.5.1 Orthogonaltransformationsmethode

Annahme: $\mathbf{A} \in \mathbb{K}^{m,n}$, $m \geq n$, hat vollen Rang: $\mathrm{rank}(\mathbf{A}) = n$

QR-Zerlegung (→ Abschnitt 3.2.7 (sec:die-qr-zerlegung)): $\mathbf{A} = \mathbf{QR}$, $\mathbf{Q} \in \mathbb{K}^{m,m}$ unitär, $\mathbf{R} \in \mathbb{K}^{m,n}$ (reguläre) obere Dreiecksmatrix.

$$\|\mathbf{Ax} - \mathbf{b}\|_2 = \left\| \mathbf{Q}(\mathbf{Rx} - \mathbf{Q}^H\mathbf{b}) \right\|_2 = \left\| \mathbf{Rx} - \widetilde{\mathbf{b}} \right\|_2 \quad , \quad \widetilde{\mathbf{b}} := \mathbf{Q}^H\mathbf{b} \, .$$

$$\|\mathbf{Ax} - \mathbf{b}\|_2 \to \min \quad \Leftrightarrow \quad \left\| \begin{pmatrix} \mathbf{R} \\ \mathbf{0} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} - \begin{pmatrix} \widetilde{b}_1 \\ \vdots \\ \\ \\ \widetilde{b}_m \end{pmatrix} \right\|_2 \to \min \, .$$

$$\mathbf{x} = \begin{pmatrix} \mathbf{R} \end{pmatrix}^{-1} \begin{pmatrix} \widetilde{b}_1 \\ \vdots \\ \widetilde{b}_n \end{pmatrix} \quad , \quad \text{Residuum} \quad \mathbf{r} = \mathbf{Q} \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \widetilde{b}_{n+1} \\ \vdots \\ \widetilde{b}_m \end{pmatrix} \, .$$

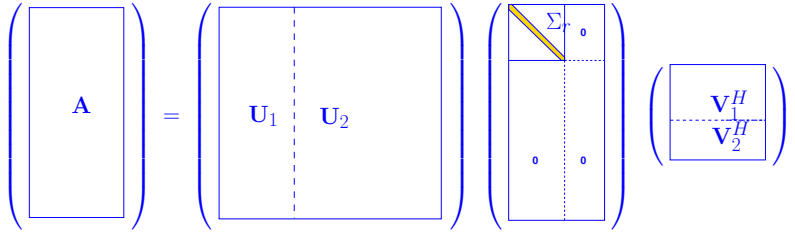
Beachte: $\|\mathbf{r}\|_2 = \sqrt{\widetilde{b}_{n+1}^2 + \cdots + \widetilde{b}_m^2}$

Implementierung: Sukzessive Orthogonaltransformation (von links) von $\mathbf{A}$ ($\rightarrow \mathbf{R}$), $\mathbf{b}$ ($\rightarrow \widetilde{\mathbf{b}}$)

☞ QR-basierter Algorithmus implementiert im Least-Squares-Löser des MATLAB-Operators \ (für vollbesetzte Matrizen)

Ausgleichsrechnung und SVD (für allgemeines $\mathbf{A} \in \mathbb{K}^{m,n}$, $\operatorname{rank}(\mathbf{A}) = r \leq \min\{m,n\}$):

$$\mathbf{A} = [\,\mathbf{U}_1 \quad \mathbf{U}_2\,] \begin{pmatrix} \Sigma_r & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{V}_1^H \\ \mathbf{V}_2^H \end{pmatrix}$$



$\mathbf{U}_1 \in \mathbb{K}^{m,r}$, $\mathbf{U}_2 = \mathbb{K}^{m,m-r}$, $\Sigma_r = \operatorname{diag}(\sigma_1, \ldots, \sigma_r) \in \mathbb{R}^{r,r}$, $\mathbf{V}_1 \in \mathbb{K}^{n,r}$, $\mathbf{V}_2 \in \mathbb{K}^{n,n-r}$, Spalten von $\mathbf{U}_1, \mathbf{U}_2, \mathbf{V}_1, \mathbf{V}_2$ orthonormiert.

$$\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2 = \left\| [\mathbf{U}_1\ \mathbf{U}_2] \begin{pmatrix} \Sigma_r & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{V}_1^H \\ \mathbf{V}_2^H \end{pmatrix} \mathbf{x} - \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{pmatrix} \right\|_2 = \left\| \begin{pmatrix} \Sigma_r \mathbf{V}_1^H \mathbf{x} \\ 0 \end{pmatrix} - \begin{pmatrix} \mathbf{U}_1^H \mathbf{b}_1 \\ \mathbf{U}_2^H \mathbf{b}_2 \end{pmatrix} \right\|_2$$

► $\mathbf{x} = (\mathbf{V}_1 \Sigma_r^{-1} \mathbf{U}_1^H)\mathbf{b}_1$.

Praxis:

NUMRANK
Numerischer Rang":

$r = \max\{i\colon \sigma_i/\sigma_1 > \texttt{tol}\}$

MATLAB-Code Lösung eines linearen Ausgleichsproblems mit SVD

```
function y = lsqsvd(A,b)
[U,S,V] = svd(A,0);
sv = diag(S);
r = max(find(sv/sv(1) > eps));
y = V(:,1:r)*(diag(1./sv(1:r))*...
          (U(:,1:r)'*b));
```

Theorem 3.5.5. *Hat $\mathbf{A} \in \mathbb{K}^{m,n}$ die SVD (3.5.1), dann gilt* $\mathbf{A}^+ = \mathbf{V}_1 \Sigma_r^{-1} \mathbf{U}_1^H$.

lsq:svd

Anwendung der SVD: Minimierung auf der Einheitssphäre:

Gegeben: $\mathbf{A} \in \mathbb{K}^{m,n}$, $m > n$: Suche $\mathbf{x} \in \mathbb{K}^n$, $\|\mathbf{x}\|_2 = 1$: $\|\mathbf{A}\mathbf{x}\|_2 \rightarrow \min$. (3.5.2)

Mit SVD $\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^H$:

$$\min_{\|\mathbf{x}\|_2=1} \|\mathbf{A}\mathbf{x}\|_2^2 = \min_{\|\mathbf{x}\|_2=1} \left\|\mathbf{U}\Sigma\mathbf{V}^H\mathbf{x}\right\|_2^2 = \min_{\|\mathbf{V}^H\mathbf{x}\|_2=1} \left\|\mathbf{U}\Sigma(\mathbf{V}^H\mathbf{x})\right\|_2^2$$
$$= \min_{\|\mathbf{y}\|_2=1} \|\Sigma\mathbf{y}\|_2^2 = \min_{\|\mathbf{y}\|_2=1} (\sigma_1^2 y_1^2 + \cdots + \sigma_n^2 y_n^2) \geq \sigma_n^2 .$$

Das Minimum wird für $\mathbf{y} = \mathbf{e}_n$ angenommen ➢ $\mathbf{x} = \mathbf{V}\mathbf{e}_n = \mathbf{v}_{\cdot,n}$.

► Minimaler Wert $= \sigma_n$

Beispiel 121 (Fitten von Hyperebenen).

Hessesche Normalform einer Hyperebene im $\mathbb{R}^d$: $g = \{\mathbf{x} \in \mathbb{R}^d\colon c + \mathbf{n}^T\mathbf{x} = 0\}$, $\|\mathbf{n}\|_2 = 1$

► Euklidischer Abstand $\operatorname{dist}(g, \mathbf{y}) = |c + \mathbf{n}^T\mathbf{y}|$.

Ziel: Zu gegebenen Punkten $\mathbf{y}_1, \ldots, \mathbf{y}_m$, $m > d$, finde $g \leftrightarrow c \in \mathbb{R}, \mathbf{n} \in \mathbb{R}^d$, $\|\mathbf{n}\|_2 = 1$, so dass

$$\sum_{j=1}^m \operatorname{dist}(g, \mathbf{y}_j)^2 \rightarrow \min .$$

$$\|\mathbf{A}\mathbf{x}\|_2 \rightarrow \min \quad \leftrightarrow \quad \left\| \begin{pmatrix} 1 & y_{1,1} & \cdots & y_{1,n} \\ 1 & y_{2,1} & \cdots & y_{2,n} \\ \vdots & \vdots & & \vdots \\ 1 & y_{m,1} & \cdots & y_{m,n} \end{pmatrix} \begin{pmatrix} c \\ n_1 \\ \vdots \\ n_n \end{pmatrix} \right\|_2 \rightarrow \min \quad \text{wobei} \quad \|\mathbf{n}\|_2 = 1 .$$

sec:die-qr-zerlegung

Mit QR-Zerlegung (→ Abschnitt 3.2.7)

$$\mathbf{A} := \begin{pmatrix} 1 & y_{1,1} & \cdots & y_{1,n} \\ 1 & y_{2,1} & \cdots & y_{2,n} \\ \vdots & \vdots & & \vdots \\ 1 & y_{m,1} & \cdots & y_{m,n} \end{pmatrix} = \mathbf{Q}\mathbf{R} \quad , \quad \mathbf{R} := \begin{pmatrix} r_{11} & r_{12} & \cdots & \cdots & r_{1,d+1} \\ 0 & r_{22} & \cdots & \cdots & r_{2,d+1} \\ \vdots & & \ddots & & \vdots \\ 0 & & & & r_{d+1,d+1} \\ 0 & \cdots & & \cdots & 0 \\ \vdots & & & & \vdots \\ 0 & \cdots & & \cdots & 0 \end{pmatrix} \in \mathbb{R}^{m,d+1} .$$

$$\|\mathbf{A}\mathbf{x}\|_2 \rightarrow \min \quad \Leftrightarrow \quad \|\mathbf{R}\mathbf{x}\|_2 = \left\| \begin{pmatrix} r_{11} & r_{12} & \cdots & \cdots & r_{1,d+1} \\ 0 & r_{22} & \cdots & \cdots & r_{2,d+1} \\ \vdots & & \ddots & & \vdots \\ 0 & & & & r_{d+1,d+1} \\ 0 & \cdots & & \cdots & 0 \\ \vdots & & & & \vdots \\ 0 & \cdots & & \cdots & 0 \end{pmatrix} \begin{pmatrix} c \\ n_1 \\ \vdots \\ \vdots \\ n_d \end{pmatrix} \right\|_2 \rightarrow \min .$$

① Löse mittels SVD (Problem vom Typ (3.5.2)): |lsq:linconst

$$\left\| \begin{pmatrix} r_{21} & r_{22} & \cdots & \cdots & r_{2,d+1} \\ 0 & r_{33} & \cdots & \cdots & r_{3,d+1} \\ \vdots & & \ddots & & \vdots \\ 0 & & & & r_{d+1,d+1} \end{pmatrix} \begin{pmatrix} n_1 \\ \vdots \\ \vdots \\ n_d \end{pmatrix} \right\|_2 \to \min \quad , \quad \|\mathbf{n}\|_2 = 1 \, .$$

② Da $r_{11} = \sqrt{d+1} \neq 0$ ► $c = -r_{11}^{-1} \sum_{j=1}^{d} r_{1,j+1} n_j$.

MATLAB-Funktion:

Zu $\mathbf{A} \in \mathbb{K}^{m,n}$ suche $\mathbf{n} \in \mathbb{K}^{\mathtt{dim}}$, $\mathbf{c} \in \mathbb{K}^{n-\mathtt{dim}}$ mit

$$\left\| \mathbf{A} \begin{pmatrix} \mathbf{c} \\ \mathbf{n} \end{pmatrix} \right\|_2 \to \min$$

mit Nebenbedingung:

$$\|\mathbf{n}\|_2 = 1 \, .$$

MATLAB-CODE: Lineares Ausgleichsproblem mit Nebenbedingung

```
function [c,n] = clsq(A,dim);
[m,p] = size(A);
if p < dim+1, error ('not enough unknowns'); end;
if m < dim, error ('not enough equations'); end;
m = min (m, p);
R = triu (qr (A));
[U,S,V] = svd(R(p-dim+1:m,p-dim+1:p));
n = V(:,dim);
c = -R(1:p-dim,1:p-dim)\R(1:p-dim,p-dim+1:p)*n;
```


3.5.2 Normalengleichungen

NORMGLEICH
$\mathbf{A} \in \mathbb{K}^{m,n}$, $m \geq n$, mit vollem Rang $\mathrm{rank}(\mathbf{A}) = n$. Zu lösen ist (3.5.1) für $\mathbf{b} \in \mathbb{K}^m$ |eq:LSQ

$$\mathbf{x} \in \mathbb{K}^n: \quad \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2 \to \min \, . \tag{3.5.3}$$ sq:prb

$$f(\mathbf{x}) := \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 = \mathbf{x}^H (\mathbf{A}^H \mathbf{A}) \mathbf{x} - 2\mathbf{b}^H \mathbf{A}\mathbf{x} + \mathbf{b}^H \mathbf{b} \, .$$

► $\mathbf{grad}\, f(\mathbf{x}) = 2(\mathbf{A}^H \mathbf{A})\mathbf{x} - 2\mathbf{A}^H \mathbf{b}$.

$\mathbf{grad}\, f(\mathbf{x}) \overset{!}{=} 0$: $\mathbf{A}^H \mathbf{A}\mathbf{x} = \mathbf{A}^H \mathbf{b}$ = Normalengleichung zu (3.5.3) . |lsq:prb

Beachte: $\mathrm{rank}(\mathbf{A}) = n$ ➢ $\mathbf{A}^H \mathbf{A} \in \mathbb{K}^{n,n}$ regulär

Vorsicht: Gefahr der Instabilität: Mit SVD $\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^H$

$$\mathrm{cond}_2(\mathbf{A}^H \mathbf{A}) = \mathrm{cond}_2(\mathbf{V}\Sigma^H \mathbf{U}^H \mathbf{U}\Sigma\mathbf{V}^H) = \mathrm{cond}_2(\Sigma^H \Sigma) = \frac{\sigma_1^2}{\sigma_n^2} = \mathrm{cond}_2(\mathbf{A})^2 \, .$$

Vorsicht: Informationsverlust bei Berechnung von $\mathbf{A}^H \mathbf{A}$, z.B.

$$\mathbf{A} = \begin{pmatrix} 1 & 1 \\ \delta & 0 \\ 0 & \delta \end{pmatrix} \quad ► \quad \mathbf{A}^H \mathbf{A} = \begin{pmatrix} 1+\delta^2 & 1 \\ 1 & 1+\delta^2 \end{pmatrix}$$

Wenn $\delta < \sqrt{\mathtt{eps}}$ ➢ $1 + \delta^2 = 1$ in $\mathbb{M}$, d.h. $\mathbf{A}^H \mathbf{A}$ „numerisch singulär", obwohl $\mathrm{rank}(\mathbf{A}) = 2$

```
>> A = [1         1;...
        sqrt(eps) 0;...
        0     sqrt(eps)];
>> rank(A)
     ans =  2
>> rank(A'*A)
     ans =  1
```

Vermeidung der Berechnung von $\mathbf{A}^H \mathbf{A}$: Residuum $\mathbf{r} := \mathbf{A}\mathbf{x} - \mathbf{b}$ als neue Unbekannte:

$$\mathbf{A}^H \mathbf{A}\mathbf{x} = \mathbf{A}^H \mathbf{b} \quad \Leftrightarrow \quad \mathbf{B} \begin{pmatrix} \mathbf{r} \\ \mathbf{x} \end{pmatrix} := \begin{pmatrix} -\mathbf{I} & \mathbf{A} \\ \mathbf{A}^H & 0 \end{pmatrix} \begin{pmatrix} \mathbf{r} \\ \mathbf{x} \end{pmatrix} = \begin{pmatrix} \mathbf{b} \\ 0 \end{pmatrix} \, . \tag{3.5.4}$$ sq:trafo

Allgemeinere Substitution $\mathbf{z} := \alpha\mathbf{x}$, $\mathbf{r} := \alpha^{-1}\mathbf{A}\mathbf{x} - \mathbf{b}$, $\alpha > 0$ (zur Verbesserung der Kondition):

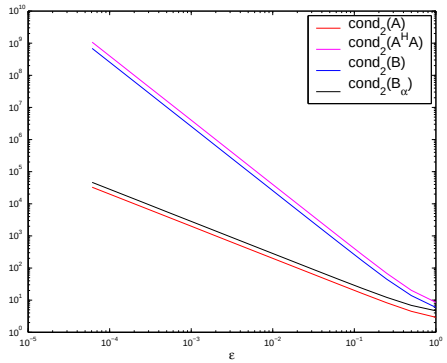
$$\mathbf{A}^H \mathbf{A}\mathbf{x} = \mathbf{A}^H \mathbf{b} \quad \Leftrightarrow \quad \mathbf{B}_\alpha \begin{pmatrix} \mathbf{r} \\ \mathbf{x} \end{pmatrix} := \begin{pmatrix} -\alpha\mathbf{I} & \mathbf{A} \\ \mathbf{A}^H & 0 \end{pmatrix} \begin{pmatrix} \mathbf{r} \\ \mathbf{z} \end{pmatrix} = \begin{pmatrix} \alpha\mathbf{b} \\ 0 \end{pmatrix} \tag{3.5.5}$$ sq:trafo2

Beispiel 122 (Kondition des erweiterten Systems).

$$\mathbf{A} = \begin{pmatrix} 1+\epsilon & 1 \\ 1-\epsilon & 1 \\ \epsilon & \epsilon \end{pmatrix}$$

Plot verschiedener Konditionszahlen in Abhängigkeit von ϵ ($\alpha = \|\mathbf{A}\|_2 / \sqrt{2}$)

➢



◇

Bemerkung 123 (Normalengleichunge vs. Orthogonaltransformationsmethode).

Vorteil der Normalengleichungen in der Form (3.5.4)/(3.5.5): |lsq:trafo |lsq:trafo2

$\mathbf{A}$ sparse (→ Def. 3.1.1) ⇒ $\mathbf{B}/\mathbf{B}_\alpha$ sparse |def:sparse

△

3.5.3 Totales Ausgleichsproblem

TOTAUSGLPROB

Gegeben: $\mathbf{A} \in \mathbb{K}^{m,n}$, $m \geq n$, $\operatorname{rank}(\mathbf{A}) = n$, $\mathbf{b} \in \mathbb{K}^m$

Gesucht: $\widehat{\mathbf{A}} \in \mathbb{K}^{m,n}$, $\widehat{\mathbf{b}} \in \mathbb{K}^m$ mit

$$\left\| \underbrace{\begin{bmatrix} \mathbf{A} & \mathbf{b} \end{bmatrix}}_{=:\mathbf{C}} - \underbrace{\begin{bmatrix} \widehat{\mathbf{A}} & \widehat{\mathbf{b}} \end{bmatrix}}_{=:\widehat{\mathbf{C}}} \right\|_F \to \min \quad , \quad \widehat{\mathbf{b}} \in \operatorname{Im}(\widehat{\mathbf{A}}) \, .$$

(3.5.6) sq:total

$$\widehat{\mathbf{b}} \in \operatorname{Im}(\widehat{\mathbf{A}}) \quad \Rightarrow \quad \operatorname{rank}(\widehat{\mathbf{C}}) = n \quad \blacktriangleright \quad \text{(3.5.6)} \overset{\text{lsq:total}}{\Rightarrow} \quad \min_{\operatorname{rank}(\widehat{\mathbf{C}})=n} \left\| \mathbf{C} - \widehat{\mathbf{C}} \right\|_F \, .$$

thm:rankapprox

Thm. 3.4.16 ➤ Verwende SVD $\mathbf{C} = \mathbf{U}\Sigma\mathbf{V}^T$

$$\widehat{\mathbf{C}} = \sum_{j=1}^{n} \sigma_j \mathbf{u}_{\cdot,j} \mathbf{v}_{\cdot,j}^H \quad \Rightarrow \quad \widehat{\mathbf{C}}\mathbf{v}_{\cdot,n+1} = 0 \, .$$

Wenn $\mathbf{v}_{n+1,n+1} \neq 0$, dann

$$\widehat{\mathbf{A}}\mathbf{x} = \widehat{\mathbf{b}} \quad \text{mit} \quad \mathbf{x} = \mathbf{v}_{n+1,n+1}^{-1} \mathbf{v}_{\cdot,n+1} \, .$$

MATLAB-Code Totales Ausgleichsproblem

```
function x = lsqtotal(A,b);
[m,n]=size(A);
[U, Sigma, V] = svd([A,b]);
s = V(n+1,n+1);
if s == 0 ,
 error('No solution')
end
x = -V(1:n,n+1)/s;
```

3.5.4 Ausgleichsrechnung mit linearen Nebenbedingungen

AUSGLPROBLINNB

Gegeben: $\mathbf{A} \in \mathbb{K}^{m,n}$, $m \geq n$, $\operatorname{rank}(\mathbf{A}) = n$, $\mathbf{b} \in \mathbb{K}^m$,
$\mathbf{C} \in \mathbb{K}^{p,n}$, $p < n$, $\operatorname{rank}(\mathbf{C}) = p$, $\mathbf{d} \in \mathbb{K}^p$

Gesucht: $\mathbf{x} \in \mathbb{K}^n$ mit $\quad \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2 \to \min \quad , \quad \boxed{\mathbf{C}\mathbf{x} = \mathbf{d}} \, .$

(3.5.7) sq:linc

Lineare Nebenbedingung

LagrMult

Idee: Ankopplung der Nebenbedingung durch Lagrange-Multiplikator $\mathbf{m} \in \mathbb{K}^p$

$$\mathbf{x} = \operatorname*{argmin}_{\mathbf{x}\in\mathbb{K}^n} \max_{\mathbf{m}\in\mathbb{K}^p} L(\mathbf{x},\mathbf{m}) \, , \quad L(\mathbf{x},\mathbf{m}) := \frac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2 + \mathbf{m}^H(\mathbf{C}\mathbf{x} - \mathbf{d}) \, .$$

Notwendige (und hinreichende) Bedingungen für Lösung (→ Abschnitt 3.5.2) sec:normalengleichungen

$$\frac{\partial L}{\partial \mathbf{x}}(\mathbf{x},\mathbf{m}) = \mathbf{A}^H(\mathbf{A}\mathbf{x} - \mathbf{b}) + \mathbf{C}^H\mathbf{m} \overset{!}{=} 0 \quad , \quad \frac{\partial L}{\partial \mathbf{m}}(\mathbf{x},\mathbf{m}) = \mathbf{C}\mathbf{x} - \mathbf{d} \overset{!}{=} 0; \, .$$

$$\begin{pmatrix} \mathbf{A}^H\mathbf{A} & \mathbf{C}^H \\ \mathbf{C} & 0 \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \mathbf{m} \end{pmatrix} = \begin{pmatrix} \mathbf{A}^H\mathbf{b} \\ \mathbf{d} \end{pmatrix}$$

ERWNORMGL

Erweiterte Normalengleichungen (Sattelpunktproblem)

Algorithmus (basierend auf Block-LU-Zerlegung):

$$\begin{pmatrix} \mathbf{A}^H\mathbf{A} & \mathbf{C}^H \\ \mathbf{C} & 0 \end{pmatrix} = \begin{pmatrix} \mathbf{R}^H & 0 \\ \mathbf{G} & -\mathbf{S}^H \end{pmatrix} \begin{pmatrix} \mathbf{R} & \mathbf{G}^H \\ 0 & \mathbf{S} \end{pmatrix} \quad , \quad \begin{array}{l} \mathbf{R},\mathbf{S} \in \mathbb{K}^{n,n} \text{ obere Dreiecksmatrizen} \\ \mathbf{G} \in \mathbb{K}^{p,n} \, . \end{array}$$

$\mathbf{R}^H\mathbf{R} = \mathbf{A}^H\mathbf{A}$ → Cholesky-Zerlegung → Sect. 3.2.5 sec:symm-posit-defin
$\mathbf{R}^H\mathbf{G}^H = \mathbf{C}^H$ → n Vorwärtssubstitutionen → Sect. 3.2.3 sec:die-lr-zerlegung
$\mathbf{G}\mathbf{G}^H = \mathbf{S}^H\mathbf{S}$ → Cholesky-Zerlegung → Sect. 3.2.5 sec:symm-posit-defin

Vorsicht: Abschn. 3.5.2: Berechnung von $\mathbf{A}^H\mathbf{A}$ kann teuer und problematisch sein ! sec:normalengleichungen

(Abhilfe durch Einführung einer neuen Unbekannten $\mathbf{r} = \mathbf{A}\mathbf{x} - \mathbf{b}$, vgl. (3.5.4)) lsg:trafo

$$\blacktriangleright \quad \begin{pmatrix} -\mathbf{I} & \mathbf{A} & 0 \\ \mathbf{A}^H & 0 & \mathbf{C}^H \\ 0 & \mathbf{C} & 0 \end{pmatrix} \begin{pmatrix} \mathbf{r} \\ \mathbf{x} \\ \mathbf{m} \end{pmatrix} = \begin{pmatrix} \mathbf{b} \\ 0 \\ \mathbf{d} \end{pmatrix} \, . \qquad (3.5.8) \text{ sq:ctrf}$$

SVD-basierter Algorithmus:

① Berechne Orthonormalbasis von $\operatorname{Ker}(\mathbf{C})$ mit SVD (→ Abschnitt 3.5.1): sec:orth

$$\mathbf{C} = \mathbf{U}\,[\Sigma\ 0] \begin{bmatrix} \mathbf{V}_1^H \\ \mathbf{V}_2^H \end{bmatrix} \quad , \quad \mathbf{U} \in \mathbb{K}^{p,p}, \Sigma \in \mathbb{R}^{p,p}, \mathbf{V}_1 \in \mathbb{K}^{n,p}, \mathbf{V}_2 \in \mathbb{K}^{n,n-p}$$

$$\blacktriangleright \quad \operatorname{Ker}(\mathbf{C}) = \operatorname{Im}(\mathbf{V}_2) \, .$$

und partikuläre Lösung

$$\mathbf{x}_0 := \mathbf{V}_1\Sigma^{-1}\mathbf{U}^H\mathbf{d} \, .$$

Darstellung für Lösung $\mathbf{x}$ von (3.5.7): lsg:linc $\qquad \mathbf{x} = \mathbf{x}_0 + \mathbf{V}_2\mathbf{y}, \quad \mathbf{y} \in \mathbb{K}^{n-p}$

② Einsetzen der Darstellung ➤ Standard lineares Ausgleichsproblem

$$\|\mathbf{A}(\mathbf{x}_0 + \mathbf{V}_2\mathbf{y}) - \mathbf{b}\|_2 \to \min \quad \Leftrightarrow \quad \|\mathbf{A}\mathbf{V}_2\mathbf{y} - (\mathbf{b} - \mathbf{A}\mathbf{x}_0)\| \to \min \, .$$

3.6 Krylov-Verfahren für lineare Gleichungssysteme

[File: section-krylov-verfahren-fuer-lineare-gleichungssysteme.tex, SVN: section-krylov-verfahren-fuer-lineare-gleichungssysteme.tex 1062 2006-10-21 09:33:

KRYLURVERF

Klasse von Iterationsverfahren (→ Abschnitt sec:iterationsverfahren 2.1) zur approximativen Lösung linearer Gleichungssysteme $\mathbf{A}\mathbf{x} = \mathbf{b}$.

Charakteristika:
- Nur Auswertung $\mathbf{A}\times$Vektor erforderlich
- (In vielen Fällen) Aufwand für einen Schritt $\approx$ Kosten($\mathbf{A}\times$Vektor)

Iterationsverfahren zur Lösung linearer Gleichungssysteme **!?** Warum **?**
- Eliminationsverfahren zu aufwändig (z.B. für $\mathbf{A}$ sehr gross, dünnbesetzt), → (lab:lucosts 3.2.7)
- Direktes Lösen „overkill", da nur geringe Genauigkeit erforderlich (z.B. im Newton-Verfahren → Abschnitt sec:newtonverfahren 2.5)
- Gute Näherung als Startwert verfügbar
- Koeffizientenmatrix nur als Prozedur `y=evalA(x)` $\leftrightarrow \mathbf{y} = \mathbf{A}\mathbf{x}$ gegeben

Beobachtung: Lösungen vieler LGS aus Wissenschaft/Technik mit s.p.d. Koeffizientenmatrix weisen dominierende Anteile von Eigenvektoren zu betragsgrössten/betragskleinsten Eigenwerten auf.

[siehe Motivation für Krylovräume, Abschnitt sec:kryl-unterr 3.4.3]

Idee: Gegeben Näherungslösung $\mathbf{x}^{(0)}$, suche Lösung der FEHLERGL Fehlergleichung

$$\mathbf{A}\mathbf{c} = \mathbf{r} \quad , \quad \text{Residuum } \mathbf{r} := \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)} \tag{3.6.1}$$ eq:cfix

im Krylovraum $\mathcal{K}_l(\mathbf{A}, \mathbf{z})$ (→ Def. def:Kryl 3.4.8), $\mathbf{z}, l$ „geeignet"

! Beachte: $\dim \mathcal{K}_l(\mathbf{A}, \mathbf{z}) \le n$ ➢ eq:cfix (3.6.1) = überbestimmtes Problem

3.6.1 Das Verfahren der konjugierten Gradienten (CG)

Annahme: $\mathbf{A} = \mathbf{A}^H \in \mathbb{K}^{n,n}$, $n \in \mathbb{N}$, Hermitesch, positiv definit (→ Def. def:spd 3.2.8)

➡ ASKALARPROD $\mathbf{A}$-Skalarprodukt $(\mathbf{x}, \mathbf{y}) \mapsto \mathbf{x}^H\mathbf{A}\mathbf{y}$ ⇒ „$\mathbf{A}$-Geometrie"

Definition 3.6.1 (Energienorm). *Die von einer s.p.d. Matrix $\mathbf{A} \in \mathbb{K}^{n,n}$ induzierte Energienorm ist*

$$\|\mathbf{x}\|_A := (\mathbf{x}^H\mathbf{A}\mathbf{x})^{1/2} , \quad \mathbf{x} \in \mathbb{K}^n .$$

3.6.1.1 Prinzip des CG-Verfahrens

Idee:
- Abbruch für exakte Lösung: $\mathbf{z} = \mathbf{r} := \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)}$ (Residuum)
- Fixiere $\mathbf{c}$ aus (eq:cfix 3.6.1) durch Galerkin-Bedingung GALCOND

$$\mathbf{r} - \mathbf{A}\mathbf{c} \perp \mathcal{K}_l(\mathbf{A}, \mathbf{r}) \quad \Leftrightarrow \quad \mathbf{r}^{(\text{neu})} := \mathbf{b} - \mathbf{A}\mathbf{x}^{(\text{neu})} \perp \mathcal{K}_l(\mathbf{A}, \mathbf{r}) . \tag{3.6.2}$$ eq:galco

Euklidisch orthogonal

CGVERF

(Abstraktes) Verfahren der konjugierten Gradienten (CG) berechnet Näherungslösung $\mathbf{x}^{(l)}$ des LGS $\mathbf{A}\mathbf{x} = \mathbf{b}$ durch Korrektur der Startnäherung $\mathbf{x}^{(0)}$ in $\mathcal{K}_l(\mathbf{A}, \mathbf{r})$, $\mathbf{r} := \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)}$, festgelegt durch Galerkin-Bedingung (eq:galcond 3.6.2)

➢ falls $\{\mathbf{p}_1, \ldots, \mathbf{p}_l\}$ = Basis von $\mathcal{K}_l(\mathbf{A}, \mathbf{r})$

(eq:galcond 3.6.2) eq:gc2

$$\mathbf{p}_j^H(\mathbf{r} - \mathbf{A}\mathbf{c}) = \mathbf{p}_j^H(\mathbf{b} - \mathbf{A}\mathbf{x}^{(\text{neu})}) = 0 \quad \forall j = 1, \ldots, l . \tag{3.6.3}$$ eq:gc2

➢ Berechne Koeffizienten der Basisdarstellung $\mathbf{c} = \gamma_1\mathbf{p}_1 + \cdots + \gamma_l\mathbf{p}_l$ aus

$$\begin{pmatrix} \mathbf{p}_1^H\mathbf{A}\mathbf{p}_1 & \cdots & \mathbf{p}_1^H\mathbf{A}\mathbf{p}_l \\ \vdots & & \vdots \\ \mathbf{p}_l^H\mathbf{A}\mathbf{p}_1 & \cdots & \mathbf{p}_l^H\mathbf{A}\mathbf{p}_l \end{pmatrix} \begin{pmatrix} \gamma_1 \\ \vdots \\ \gamma_l \end{pmatrix} = \begin{pmatrix} \mathbf{p}_1^H\mathbf{r} \\ \vdots \\ \mathbf{p}_l^H\mathbf{r} \end{pmatrix} . \tag{3.6.4}$$ eq:gclsg

Besonders einfach, wenn $\{\mathbf{p}_1, \ldots, \mathbf{p}_l\}$ $\mathbf{A}$-orthogonale Basis von $\mathcal{K}_l(\mathbf{A}, \mathbf{r})$: $\mathbf{p}_j^H\mathbf{A}\mathbf{p}_i = 0$ für $i \ne j$.

Bemerkung 124 (CG als direkter Löser).

Falls $\mathbf{A}^l\mathbf{r} \in \mathcal{K}_l(\mathbf{A}, \mathbf{r})$, dann $\mathbf{x}^{(l)} = \mathbf{A}^{-1}\mathbf{b}$ für die Näherungslösung aus dem abstrakten CG-Algorithmus.
➢ CG terminiert nach spätestens n Schritten mit $\mathbf{x}^{(l)} = \mathbf{A}^{-1}\mathbf{b}$.

Sei l minimal, so dass $\mathbf{A}^l\mathbf{r} \in \mathcal{K}_l(\mathbf{A}, \mathbf{r})$. Nach dem Austauschaxiom $\mathbf{r} \in \operatorname{Span}\left\{\mathbf{A}\mathbf{r}, \ldots, \mathbf{A}^l\mathbf{r}\right\} = \mathbf{A}\mathcal{K}_l(\mathbf{A}, \mathbf{r})$, also $\mathbf{A}(\mathbf{x}^* - \mathbf{x}^{(0)}) \in \mathbf{A}\mathcal{K}_l(\mathbf{A}, \mathbf{r}) \Leftrightarrow \mathbf{x}^* - \mathbf{x}^{(0)} \in \mathcal{K}_l(\mathbf{A}, \mathbf{r})$, so dass die Korrektur die exakte Lösung liefert.

Bemerkung 125 (CG-Verfahren und quadratische Minimierung).

Lemma 3.6.2 (S.p.d. Gleichungssysteme und Minimierungsproblem). *Für* $\mathbf{A} \in \mathbb{K}^{n,n}$ *s.p.d.,* $\mathbf{b} \in \mathbb{K}^n$ *gilt*

$$\mathbf{Ax} = \mathbf{b} \quad \Leftrightarrow \quad \mathbf{x} = \arg\min_{\mathbf{y}\in\mathbb{K}^n} J(\mathbf{y}) \,, \quad J(\mathbf{y}) := \tfrac{1}{2}\mathbf{y}^H\mathbf{Ay} - \mathrm{Re}\{\mathbf{b}^H\mathbf{y}\} \,.$$

Unterraumkorrektur: Sei $V \subset \mathbb{K}^n$ Unterraum mit Basis $\{\mathbf{p}_1, \ldots, \mathbf{p}_l\}$, $l \ll n$.

► Approximatives Lösen der Fehlergleichung (3.6.1)durch „Minimierung in Richtung von V":

(i) $\mathbf{c} = \mathrm{argmin}_{\mathbf{y}\in V} J(\mathbf{y})$, $J(\mathbf{y}) := \tfrac{1}{2}\mathbf{y}^H\mathbf{Ay} - \mathrm{Re}\{(\mathbf{b}-\mathbf{Ax}^{(0)})^H\mathbf{y}\}$, (3.6.5) cg:1

(ii) Neue Näherung $\mathbf{x}^{(\mathrm{neu})} := \mathbf{x}^{(0)} + \mathbf{c}$. (3.6.6) cg:2

Lemma 3.6.3. $\mathbf{c}$, $\mathbf{x}^{(\mathrm{neu})}$ *aus* (3.6.5), (3.6.6) *eindeutig bestimmt durch die Galerkin-Bedingung* (3.6.3).

Bemerkung 126. Für $\mathbf{b} \in \mathbb{R}^n$, $J(\mathbf{y}) := \frac{1}{2}\mathbf{y}^H\mathbf{Ay} - \mathrm{Re}\{\mathbf{b}^H\mathbf{y}\}$ ist $-\mathbf{grad}\, J(\mathbf{x}) = \mathbf{b} - \mathbf{Ax}$

(≻ Residuum = Abstiegsrichtung):

► Wähle Residuum als Grundlage des Krylov-Raumes.

3.6.1.2 Implementierung des CG-Verfahrens

IMPLEMCG

Annahme: $\mathbf{A}$-orthogonale Basis $\{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$ von $\mathbb{R}^n$ verfügbar, so dass

$$\mathrm{Span}\{\mathbf{p}_1, \ldots, \mathbf{p}_l\} = \mathcal{K}_l(\mathbf{A}, \mathbf{r}) \,.$$

► (Effiziente) sukzessive Berechnung der $\mathbf{x}^{(l)}$

Input : Anfangsnäherung $\mathbf{x}^{(0)} \in \mathbb{K}^n$
Output : Bessere Näherungslösung $\mathbf{x}^{(l)} \in \mathbb{K}^n$

$$\mathbf{r}^{(0)} := \mathbf{b} - \mathbf{Ax}^{(0)};$$

for $j = 1$ **to** l **do** $\{ \quad \mathbf{x}^{(j)} := \mathbf{x}^{(j-1)} + \dfrac{\mathbf{p}_j^H\mathbf{r}^{(0)}}{\mathbf{p}_j^H\mathbf{Ap}_j}\mathbf{p}_j \,; \quad \}$ (3.6.7) ALG:Krylt

► Für $1 \le j \le m \le l$: Orthogonalität $\mathbf{p}_j \perp \mathbf{r}^{(m)} := \mathbf{b} - \mathbf{Ax}^{(m)}$

$$\mathbf{p}_j \cdot (\mathbf{b} - \mathbf{Ax}^{(m)}) = \mathbf{p}_j^H(\underbrace{\mathbf{b} - \mathbf{Ax}^{(0)}}_{=\mathbf{r}^{(0)}} - \sum_{k=1}^{m} \frac{\mathbf{p}_k^H\mathbf{r}^{(0)}}{\mathbf{p}_k^H\mathbf{Ap}_k}\mathbf{Ap}_k) = 0 \,. \qquad (3.6.8)\ \text{KrylOrth}$$

Aufgabe: Konstruktion einer $\mathbf{A}$-orthogonalen Basis $\{\mathbf{p}_1, \ldots, \mathbf{p}_l\}$ des Krylov-Raums $\mathcal{K}_l(\mathbf{A}, \mathbf{r}^{(0)})$
(→ Def. 3.4.8, Arnoldi-Prozess Abschnitt 3.4.3).

def:Kryl sec:kryl-unterr

(3.6.8) ⇒ Idee: Gram-Schmidt-Orthogonalisierung, → (3.4.8), der intermediären Residuen $\mathbf{r}^{(j)} := \mathbf{b} - \mathbf{Ax}^{(j)}$:

$$\mathbf{p}_1 := \mathbf{r}^{(0)} \,, \; \mathbf{p}_{j+1} := \underbrace{(\mathbf{b} - \mathbf{Ax}^{(j)})}_{=:\mathbf{r}^{(j)}} - \sum_{k=1}^{j} \frac{\mathbf{p}_k^H\mathbf{Ar}^{(j)}}{\mathbf{p}_k^H\mathbf{Ap}_k}\mathbf{p}_k \,, \quad j = 1, \ldots, l-1 \quad . \qquad (3.6.9)\ \text{GRS}$$

Lemma 3.6.4. *Nichtverschwindende Vektoren* $\mathbf{p}_j$, $1 \le j \le l$, *und* $\mathbf{r}^{(j)} := \mathbf{b} - \mathbf{Ax}^{(j)}$, $0 \le j \le l$, *aus* (3.6.7),(3.6.9) *erfüllen*

(i) $\{\mathbf{p}_1, \ldots, \mathbf{p}_j\}$ *ist* $\mathbf{A}$*-orthogonale Basis von* $\mathcal{K}_j(\mathbf{A}, \mathbf{r}^{(0)})$,

(ii) $\{\mathbf{r}^{(0)}, \ldots, \mathbf{r}^{(j-1)}\}$ *ist Orthogonalbasis von* $\mathcal{K}_j(\mathbf{A}, \mathbf{r}^{(0)})$.

Beweis. $\mathbf{A}$-Orthogonalität der $\mathbf{p}_j$ nach Konstruktion → (3.6.9).

$$(3.6.7)\ \&\ (3.6.9) \quad \Rightarrow \quad \mathbf{p}_{j+1} = \mathbf{r}^{(0)} - \sum_{k=1}^{j} \frac{\mathbf{p}_k^H\mathbf{r}^{(0)}}{\mathbf{p}_k^H\mathbf{Ap}_k}\mathbf{Ap}_k - \sum_{k=1}^{j} \frac{\mathbf{p}_k^H\mathbf{Ar}^{(j)}}{\mathbf{p}_k^H\mathbf{Ap}_k}\mathbf{p}_k \,.$$

$$\Rightarrow \quad \mathbf{p}_{j+1} \in \mathrm{Span}\left\{\mathbf{r}^{(0)}, \mathbf{p}_1, \ldots, \mathbf{p}_j, \mathbf{Ap}_1, \ldots, \mathbf{Ap}_j\right\} \,.$$

$$\mathbf{p}_1 := \mathbf{r}^{(0)} \,,\; \mathbf{p}_{j+1} := \underbrace{(\mathbf{b}-\mathbf{A}\mathbf{x}^{(j)})}_{=:\mathbf{r}^{(j)}} - \sum_{k=1}^{j} \frac{\mathbf{p}_k^H \mathbf{r}^{(j)}}{\mathbf{p}_k^H \mathbf{A}\mathbf{p}_k}\mathbf{p}_k \,,\quad j=1,\ldots,l-1 \quad , \tag{3.6.9}$$

GRS

$$\mathbf{p}_1 := \mathbf{r}^{(0)} \,,\; \mathbf{p}_{j+1} := \underbrace{(\mathbf{b}-\mathbf{A}\mathbf{x}^{(j)})}_{=:\mathbf{r}^{(j)}} - \sum_{k=1}^{j} \frac{\mathbf{p}_k^H \mathbf{r}^{(j)}}{\mathbf{p}_k^H \mathbf{A}\mathbf{p}_k}\mathbf{p}_k \,,\quad j=1,\ldots,l-1 \quad . \tag{3.6.8}$$

KrylOrth

GRS (3.6.9) cglanz $\mathbf{r}^{(j)} \in \operatorname{Span}\{\mathbf{p}_1,\ldots,\mathbf{p}_{j+1}\}$ & $\mathbf{p}_j \in \operatorname{Span}\left\{\mathbf{r}^{(0)},\ldots,\mathbf{r}^{(j-1)}\right\}$. cglanz

$$\Rightarrow \quad \operatorname{Span}\{\mathbf{p}_1,\ldots,\mathbf{p}_j\} = \operatorname{Span}\left\{\mathbf{r}^{(0)},\ldots,\mathbf{r}^{(j-1)}\right\} = \mathcal{K}_l(\mathbf{A},\mathbf{r}^{(0)}) \quad .$$

KrylOrth (3.6.8) resorth $\mathbf{r}^{(j)} \perp \operatorname{Span}\{\mathbf{p}_1,\ldots,\mathbf{p}_j\} = \operatorname{Span}\left\{\mathbf{r}^{(0)},\ldots,\mathbf{r}^{(j-1)}\right\}$. (3.6.12) resorth

Bemerkung 127 (Lanczos-Prozess und CG).

$\{\mathbf{r}^{(0)},\ldots,\mathbf{r}^{(j-1)}\} \leftrightarrow$ Orthonormalbasis von $\mathcal{K}_j(\mathbf{A},\mathbf{r})$ erzeugt im Lanczos-Prozess $\rightarrow$ Abschn. 3.4.3 sec:kryl:un

△

Analog zum Lanczos-Prozess: Kurze Rekursion für Basisvektoren $\mathbf{p}_j$:

KrylOrth (3.6.8) $\Rightarrow$ GRS (3.6.9) vereinfacht sich zu $\mathbf{p}_{j+1} := \mathbf{r}^{(j)} - \dfrac{\mathbf{p}_j^H \mathbf{A}\mathbf{r}^{(j)}}{\mathbf{p}_l^H \mathbf{A}\mathbf{r}^{(j)}}\mathbf{p}_j \,,\quad j=1,\ldots,l$.

Rekursion für Residuen:

ALG:Krylit (3.6.7) $\Rightarrow$ $\mathbf{r}^{(j)} = \mathbf{r}^{(j-1)} + \dfrac{\mathbf{p}_j^H \mathbf{r}^{(0)}}{\mathbf{p}_j^H \mathbf{A}\mathbf{p}_j}\mathbf{A}\mathbf{p}_j$. (3.6.13)

Lemma lem:CG 3.6.4, (i) $\Rightarrow$ $\mathbf{r}^{(j-1)} \cdot \mathbf{p}_j = \left(\mathbf{r}^{(0)} + \sum_{k=1}^{m-1} \dfrac{\mathbf{r}^{(0)} \cdot \mathbf{p}_k}{\mathbf{p}_k \cdot \mathbf{A}\mathbf{p}_k}\mathbf{A}\mathbf{p}_k\right) \cdot \mathbf{p}_j = \mathbf{r}^{(0)} \cdot \mathbf{p}_j$. (3.6.14)

Algorithmus: CG-Verfahren zur Lösung von $\mathbf{A}\mathbf{x}=\mathbf{b}$, $\mathbf{A}$ s.p.d.

Input : Anfangsnäherung $\mathbf{x}^{(0)} \in \mathbb{R}^n$
Output : Bessere Näherungslösung $\mathbf{x}^{(l)} \in \mathbb{R}^n$

$\mathbf{p}_1 = \mathbf{r}^{(0)} := \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)}$;
for $j = 1$ **to** l **do** {

$\mathbf{x}^{(j)} := \mathbf{x}^{(j-1)} + \dfrac{\mathbf{p}_j \cdot \mathbf{r}^{(j-1)}}{\mathbf{p}_j \cdot \mathbf{A}\mathbf{p}_j}\mathbf{p}_j$;

$\mathbf{r}^{(j)} = \mathbf{r}^{(j-1)} - \dfrac{\mathbf{p}_j \cdot \mathbf{r}^{(j-1)}}{\mathbf{p}_j \cdot \mathbf{A}\mathbf{p}_j}\mathbf{A}\mathbf{p}_{j-1}$;

$\mathbf{p}_{j+1} = \mathbf{r}^{(j)} - \dfrac{\mathbf{A}\mathbf{p}_j \cdot \mathbf{r}^{(j)}}{\mathbf{p}_j \cdot \mathbf{A}\mathbf{p}_j}\mathbf{p}_j$;

}

Input: Anfangsäherung $\mathbf{x} \hat{=} \mathbf{x}^{(0)} \in \mathbb{R}^n$
Abbruchschranke $\tau > 0$
Output: Bessere Näherungslösung $\mathbf{x} \hat{=} \mathbf{x}^{(l)}$

$\mathbf{p} := \mathbf{r}^{(0)} := \mathbf{r} := \mathbf{b} - \mathbf{A}\mathbf{x}$;
for $j = 1$ **to** $l_{\max}$ **do** {
$\beta := \mathbf{r} \cdot \mathbf{r}$;
$\mathbf{h} := \mathbf{A}\mathbf{p}$;
$\alpha := \frac{\beta}{\mathbf{p} \cdot \mathbf{h}}$;
$\mathbf{x} := \mathbf{x} + \alpha\mathbf{p}$;
$\mathbf{r} := \mathbf{r} - \alpha\mathbf{h}$;
if $\|\mathbf{r}\| \leq \tau\|\mathbf{r}^{(0)}\|$ **then stop**;
$\beta := \frac{\beta}{\mathbf{r} \cdot \mathbf{r}}$;
$\mathbf{p} := \mathbf{r} + \beta\mathbf{p}$;
}

1 Matrix×Vektor-Produkt, 3 Skalarprodukte, 3 SAXPY-Operationen pro Schritt:
Wenn $\mathbf{A}$ dünnbesetzt, $\operatorname{nnz}(\mathbf{A}) \sim n$ ➤ Rechenaufwand $O(n)$ für einen Schritt

MATLAB-Funktion:

`x=pcg(A,b,tol,maxit,[],[],x0)` : Löse $\mathbf{A}\mathbf{x}=\mathbf{b}$ mit maximal `maxit` CG Schritten: terminiere wenn $\left\|\mathbf{r}^{(l)}\right\| : \left\|\mathbf{r}^{(0)}\right\| < \mathrm{tol}$.

`x=pcg(Afun,b,tol,maxit,[],[],x0)`: `Afun` = Handle auf Funktion, die Auswertung $\mathbf{A}\times$Vektor realisiert.

`[x,flag,relr,it,resv] = pcg(...)` : Diagnostische Informationen über Verlauf der Iteration

Grundgerüst der Implementierung in MATLAB (`type pcg`):

MATLAB-Code CG-Algorithmus

```
function x = cg(A,b,tol,maxit)
r = b - A * x; rho = 1;
for i = 1 : maxit
   rho1 = rho; rho = r' * r;
   if (i == 1), p = r;
   else  beta = rho/rho1; p = r + beta * p; end
   q = A * p; alpha = rho/(p' * q);
   x = x + alpha * p;
   if (norm(b - A * x) <= tol*norm(b)) return; end
   r = r - alpha * q;
end
```

Bemerkung 128 (Abbruchkriterium für nicht-vorkonditioniertes CG-Verfahren).

Mit $\mathbf{r}^{(l)} := \mathbf{b} - \mathbf{A}\mathbf{x}^{(l)}$ gilt für beliebige Vektornormen und die zugehörige Matrixkondition (→ Def. 3.2.5) def:cond

$$\frac{1}{\operatorname{cond}(\mathbf{A})}\frac{\left\|\mathbf{r}^{(l)}\right\|}{\left\|\mathbf{r}^{(0)}\right\|} \leq \frac{\left\|\mathbf{x}^{(l)}-\mathbf{x}\right\|}{\left\|\mathbf{x}^{(0)}-\mathbf{x}\right\|} \leq \operatorname{cond}(\mathbf{A})\frac{\left\|\mathbf{r}^{(l)}\right\|}{\left\|\mathbf{r}^{(0)}\right\|} . \tag{3.6.15}$$ cg:term

Relative Abnahme des Iterationsfehlers

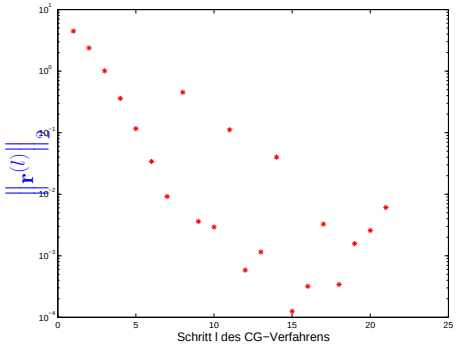
△

Beispiel 129 (Orthogonalitätsverlust der Residuen).

Numerisches Experiment: `A=hilb(20)`, $\mathbf{x}^{(0)} = 0$, $\mathbf{b} = (1,\ldots,1)^T$

Hilbert-Matrix: extrem schlecht konditioniert

Residuen während der CG-Iteration: $\mathbf{R} = \left[\mathbf{r}^{(0)},\ldots,\mathbf{r}^{(10)}\right]$



$$\mathbf{R}^H\mathbf{R} =$$

$$\begin{pmatrix}
1.000000 & -0.000000 & 0.000000 & -0.000000 & 0.000000 & -0.000000 & 0.016019 & -0.795816 & -0.430569 & 0.348133 \\
-0.000000 & 1.000000 & -0.000000 & 0.000000 & -0.000000 & 0.000000 & -0.012075 & 0.600068 & -0.520610 & 0.420903 \\
0.000000 & -0.000000 & 1.000000 & -0.000000 & 0.000000 & -0.000000 & 0.001582 & -0.078664 & 0.384453 & -0.310577 \\
-0.000000 & 0.000000 & -0.000000 & 1.000000 & -0.000000 & 0.000000 & -0.000024 & 0.001218 & -0.024115 & 0.019394 \\
0.000000 & -0.000000 & 0.000000 & -0.000000 & 1.000000 & -0.000000 & 0.000000 & -0.000002 & 0.000151 & -0.000118 \\
-0.000000 & 0.000000 & -0.000000 & 0.000000 & -0.000000 & 1.000000 & -0.000000 & 0.000000 & -0.000000 & 0.000000 \\
0.016019 & -0.012075 & 0.001582 & -0.000024 & 0.000000 & -0.000000 & 1.000000 & -0.000000 & -0.000000 & 0.000000 \\
-0.795816 & 0.600068 & -0.078664 & 0.001218 & -0.000002 & 0.000000 & -0.000000 & 1.000000 & 0.000000 & -0.000000 \\
-0.430569 & -0.520610 & 0.384453 & -0.024115 & 0.000151 & -0.000000 & -0.000000 & 0.000000 & 1.000000 & 0.000000 \\
0.348133 & 0.420903 & -0.310577 & 0.019394 & -0.000118 & 0.000000 & 0.000000 & -0.000000 & 0.000000 & 1.000000
\end{pmatrix}$$

➢ Orthogonalitätsverlust der Residuen als Folge von Rundungsfehlern, vgl. Lanczos-Prozess, Bsp. 115. ex:rflanczos

◇

3.6.1.3 Konvergenzgeschwindigkeit

KONVGESCHWCG

Bsp. 129 ex:cgorth ➢ n CG-Schritte untauglich als „direkter Löser".

Hoffnung: Substantielle Fehlerreduktion bereits in wenigen $l \ll n$ Schritten

Frage (für grosse n): Wie gut ist die Näherungslösung $\mathbf{x}^{(l)}$, $l \ll n$?

Lemma 3.6.5 („Optimalität" der CG-Iterierten). *Mit der exakten Lösung $\mathbf{x} \in \mathbb{K}^n$ von $\mathbf{A}\mathbf{x} = \mathbf{b}$ gilt für die CG-Iterierten $\mathbf{x}^{(l)}$*

$$\left\|\mathbf{x}-\mathbf{x}^{(l)}\right\|_A = \min\{\|\mathbf{y}-\mathbf{x}\|_A : \mathbf{y} \in \mathbf{x}^{(0)} + \mathcal{K}_l(\mathbf{A},\mathbf{r})\} \quad , \quad \mathbf{r} := \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)} .$$

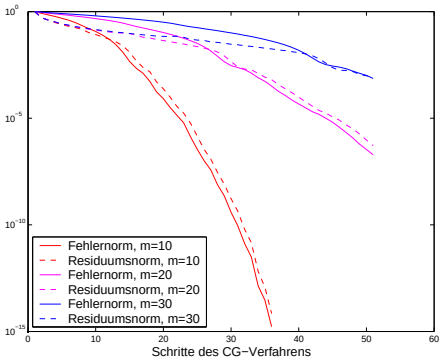
➢ Monotone Abnahme der Energienorm des Fehlers während der CG-Iteration („Gedächtnis" des CG-Verfahrens)

Beispiel 130 (Konvergenz des CG-Verfahrens).

S.p.d. Matrix $\mathbf{A} \in \mathbb{R}^{m^2,m^2}$, → Bsp. 112 ex:gspoisson

```
A = gallery('poisson',m);
x0 = (1:n)';
b = zeros(n,1);
```

Relative Abnahme von $\left\|\mathbf{r}^{(l)}\right\|_2$, $\left\|\mathbf{x}^{(l)}\right\|_2$



◇

Quantitative Konvergenzabschätzung aus Lemma 3.6.5: lem:cgpt

$$\mathbf{y} \in \mathbf{x}^{(0)} + \mathcal{K}_l(\mathbf{A},\mathbf{r}) \quad \Leftrightarrow \quad \mathbf{y} = \mathbf{x}^{(0)} + \mathbf{A}\,p(\mathbf{A})(\mathbf{x}-\mathbf{x}^{(0)})\,, \quad p = \text{Polynom vom Grad } \le l-1\,.$$

$$\mathbf{x}-\mathbf{y} = q(\mathbf{A})(\mathbf{x}-\mathbf{x}^{(0)}) \quad \text{mit } q = \text{Polynom vom Grad } \le l\,, \quad q(0)=1\,.$$

$$\left\|\mathbf{x}-\mathbf{x}^{(l)}\right\|_A \le \boxed{\min\{\max_{\lambda\in\sigma(\mathbf{A})}|q(\lambda)|\colon q \text{ Polynom vom Grad } \le l\,,\ q(0)=1\}} \cdot \left\|\mathbf{x}-\mathbf{x}^{(0)}\right\|_A\,.$$

Abzuschätzen für $\lambda \in [\lambda_{\min}(\mathbf{A}), \lambda_{\max}(\mathbf{A})]$ durch „Kandidatenpolynome"

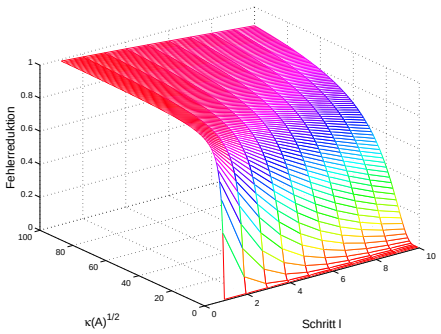
Mit Tschebyscheff-Polynomen (→ Abschnitt 4.1.3) zeigt man [21, Satz 9.4.2]: sec:tschbysch HA194

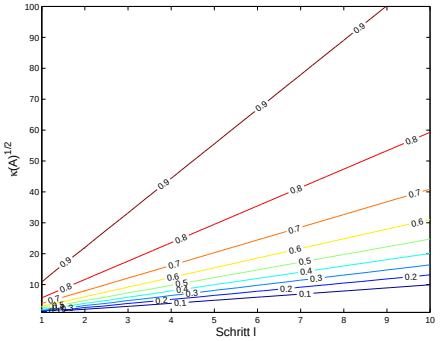
Theorem 3.6.6 (Konvergenz des CG-Verfahrens). *Für die Iterierten des CG-Verfahrens zur Lösung von $\mathbf{Ax} = \mathbf{b}$, $\mathbf{A} = \mathbf{A}^H$ positiv definit, gilt*

$$\left\|\mathbf{x}-\mathbf{x}^{(l)}\right\|_A \le \frac{2\left(1-\frac{1}{\sqrt{\kappa(\mathbf{A})}}\right)^l}{\left(1+\frac{1}{\sqrt{\kappa(\mathbf{A})}}\right)^{2l}+\left(1-\frac{1}{\sqrt{\kappa(\mathbf{A})}}\right)^{2l}} \left\|\mathbf{x}-\mathbf{x}^{(0)}\right\|_A\,.$$

($\kappa(\mathbf{A})$ = spektrale Kondition (→ Def. 3.4.5) von $\mathbf{A}$, $\kappa(\mathbf{A}) = \mathrm{cond}_2(\mathbf{A})$) def:speccond

Theoretische Schranken aus Thm. 3.6.6 für Fehlerreduktion (in Energienorm) nach l CG-Schritten: thm:CGconv

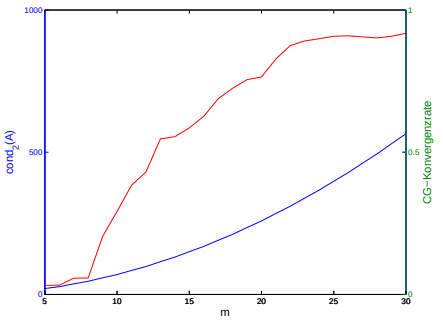


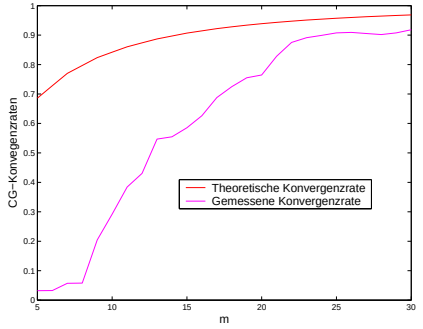


Beispiel 131 (Konvergenzraten für CG-Verfahren).

```
A = gallery('poisson',m); n = size(A,1);
x0 = (1:n)'; b = ones(n,1); maxit = 30; tol =0;
[x,flag,relres,iter,resvec] = pcg(A,b,tol,maxit,[],[],x0);
```

Messen der Konvergenzrate: $\text{rate} = \sqrt[10]{\dfrac{\left\|\mathbf{r}^{(30)}\right\|_2}{\left\|\mathbf{r}^{(20)}\right\|_2}}$.



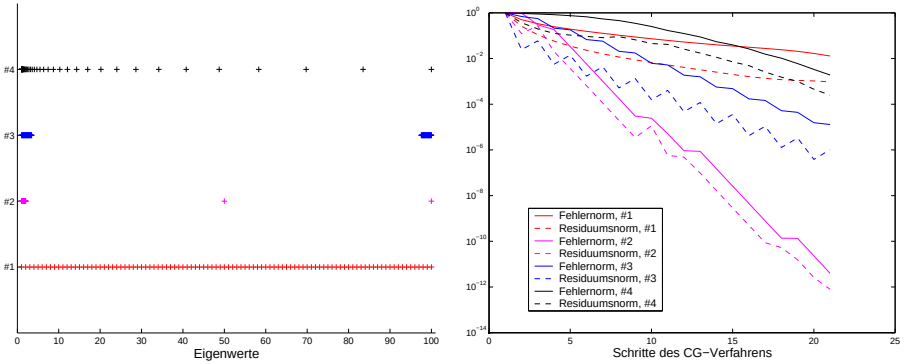


◇

Beispiel 132 (CG-Konvergenz und Spektrum).

Testmatrix #1: `A=diag(d); d = (1:100);`
Testmatrix #2: `A=diag(d); d = [1+(0:97)/97 , 50 , 100];`
Testmatrix #3: `A=diag(d); d = [1+(0:49)*0.05, 100-(0:49)*0.05];`
Testmatrix #4: Spektrum exponentiell gehäuft um 1

```
x0 = cos((1:n)'); b = zeros(n,1);
```



➢ Verteilung der Eigenwerte hat entscheidenden Einfluss auf CG-Konvergenzgeschwindigkeit:

Wichtig: Schnellere Konvergenz bei gehäuften Eigenwerten ◇

3.6.2 Vorkonditionierung

VORKOND

Thm. 3.6.6 ➢ CG konvergiert (unter Umständen) sehr langsam für $\kappa(\mathbf{A}) \gg 1$.

Idee: Vorkonditionierung (→ Abschnitt 3.4.2)

Wende CG an auf das transformierte Gleichungssystem

$$\widetilde{\mathbf{A}}\widetilde{\mathbf{x}} = \widetilde{\mathbf{b}} \quad , \quad \widetilde{\mathbf{A}} := \mathbf{B}^{-1/2}\mathbf{A}\mathbf{B}^{-1/2} \, , \, \widetilde{\mathbf{x}} := \mathbf{B}^{1/2}\mathbf{x} \, , \, \widetilde{\mathbf{b}} := \mathbf{B}^{-1/2}\mathbf{b} \, , \qquad (3.6.16)$$

mit $\kappa(\widetilde{\mathbf{A}})$ = "klein", $\mathbf{B} = \mathbf{B}^H \in \mathbb{K}^{N,N}$ positiv definiter Vorkonditionierer → Def. 3.4.4.

„Black-box"-Vorkonditionierer aus Abschnitt 3.4.2): Jacobi-, Gauss-Seidel-, IC-Vorkonditionierer.

Bemerkung 133 (Vorkonditionierung und Kontraktionszahl).

Für s.p.d. Vorkonditionierer $\mathbf{B}$ von s.p.d.$\mathbf{A}$ heisst $\rho := \left\|\mathbf{I} - \mathbf{B}^{-1}\mathbf{A}\right\|_A$ die Kontraktionszahl von $\mathbf{B}$ bzgl. $\mathbf{A}$, siehe (3.4.6), Lemma ??.

$$\rho := \lambda_{\max}(\mathbf{I} - \mathbf{B}^{-1}\mathbf{A}) = \lambda_{\max}(\mathbf{I} - \mathbf{B}^{-1/2}\mathbf{A}\mathbf{B}^{-1/2}) < 1 \quad \Rightarrow \quad \begin{cases} \lambda_{\min}(\mathbf{B}^{-1/2}\mathbf{A}\mathbf{B}^{-1/2}) \geq 1 - \rho \, , \\ \lambda_{\max}(\mathbf{B}^{-1/2}\mathbf{A}\mathbf{B}^{-1/2}) \leq 1 + \rho \, , \end{cases}$$

$$\Rightarrow \quad \kappa(\mathbf{B}^{1/2}\mathbf{A}\mathbf{B}^{1/2}) \leq \frac{1+\rho}{1-\rho} \, .$$

△

Implementierung: CG-Iteration in untransformierten Variablen $\mathbf{x}$ (intrinsische Transformation)

Algorithmus: VORKCG Vorkonditioniertes CG-Verfahren (PCG)

Input: Anfangsnäherung $\mathbf{x} \in \mathbb{K}^n \hat{=} \mathbf{x}^{(0)} \in \mathbb{K}^n$, Abbruchschranke $\tau > 0$

Output: Bessere Näherung $\mathbf{x} \hat{=} \mathbf{x}^{(l)}$

$\mathbf{p} := \mathbf{r} := \mathbf{b} - \mathbf{A}\mathbf{x}$; $\mathbf{p} := \mathbf{B}^{-1}\mathbf{r}$; $\mathbf{q} := \mathbf{p}$; $\tau_0 := \mathbf{p}^H\mathbf{r}$;
for $l = 1$ **to** $l_{\max}$ **do** {
$\beta := \mathbf{r}^H\mathbf{q}$; $\mathbf{h} := \mathbf{A}\mathbf{p}$; $\alpha := \frac{\beta}{\mathbf{p}^H\mathbf{h}}$;
$\mathbf{x} := \mathbf{x} + \alpha\mathbf{p}$;
$\mathbf{r} := \mathbf{r} - \alpha\mathbf{h}$; (3.6.17) alg:PCG
$\mathbf{q} := \mathbf{B}^{-1}\mathbf{r}$; $\beta := \frac{\mathbf{r}^H\mathbf{q}}{\beta}$;
if $|\mathbf{q}^H\mathbf{r}| \leq \tau \cdot \tau_0$ **then stop**;
$\mathbf{p} := \mathbf{q} + \beta\mathbf{p}$;
}

➢ Rechenaufwand pro Schritt: 1 Auswertung $\mathbf{A}\times$Vektor, 1 Auswertung $\mathbf{B}^{-1}\times$Vektor, 3 Skalarprodukte, 3 SAXPY-Operationen

Bemerkung 134 (Konvergenztheorie für PCG). Aussagen von Thm. 3.6.6 bleiben gültig mit $\widetilde{\mathbf{A}}$-Energienorm anstelle $\mathbf{A}$-Energienorm.

Bemerkung 135 (Abbruch des vorkonditionierten CG-Verfahrens).

Bem. 128, (3.6.15) ➢ Überwache transformiertes Residuum

$$\widetilde{\mathbf{r}} = \widetilde{\mathbf{b}} - \widetilde{\mathbf{A}}\widetilde{\mathbf{x}} = \mathbf{B}^{-1/2}\mathbf{r} \quad \Rightarrow \quad \|\widetilde{\mathbf{r}}\|_2^2 = \mathbf{r}^H\mathbf{B}^{-1}\mathbf{r} \, .$$

➢ Abschätzungen für Energienorm des Fehlers $\mathbf{e}^{(l)} := \mathbf{x} - \mathbf{x}^{(l)}$:

Benutze die Fehlergleichung (3.6.1) $\mathbf{A}\mathbf{e} = \mathbf{r}$:

$$\mathbf{r}^{(l)} \cdot \mathbf{B}^{-1}\mathbf{r}^{(l)} = \mathbf{B}^{-1}\mathbf{A}\mathbf{e}^{(l)} \cdot \mathbf{A}\mathbf{e}^{(l)} \leq \lambda_{\max}(\mathbf{B}^{-1}\mathbf{A}) \left\|\mathbf{e}^{(l)}\right\|_A^2 \, ,$$

$$\left\|\mathbf{e}^{(l)}\right\|_A^2 = \mathbf{A}\mathbf{e}^{(l)} \cdot \mathbf{e}^{(l)} = \mathbf{r}^{(l)} \cdot \mathbf{A}^{-1}\mathbf{r}^{(l)} = \mathbf{B}^{-1}\mathbf{r} \cdot \mathbf{B}\mathbf{A}^{-1}\mathbf{r}^{(l)} \leq \lambda_{\max}(\mathbf{B}\mathbf{A}^{-1})\, \mathbf{B}^{-1}\mathbf{r}^{(l)} \cdot \mathbf{r}^{(l)} \, .$$

verfügbar in Algorithmus (3.6.17)

$$\frac{1}{\kappa(\mathbf{B}^{-1}\mathbf{A})} \frac{\left\|\mathbf{e}^{(l)}\right\|_A^2}{\left\|\mathbf{e}^{(0)}\right\|_A^2} \leq \frac{\mathbf{B}^{-1}\mathbf{r}^{(l)} \cdot \mathbf{r}^{(l)}}{\mathbf{B}^{-1}\mathbf{r}^{(0)} \cdot \mathbf{r}^{(0)}} \leq \kappa(\mathbf{B}^{-1}\mathbf{A}) \frac{\left\|\mathbf{e}^{(l)}\right\|_A^2}{\left\|\mathbf{e}^{(0)}\right\|_A^2} \qquad (3.6.18)$$

$\kappa(\mathbf{B}^{-1}\mathbf{A})$ „klein" ➢ $\mathbf{B}^{-1}$-Energienorm des Residuums ≈ $\mathbf{A}$-Energienorm des Fehlers !
($\mathbf{r}^{(l)} \cdot \mathbf{B}^{-1}\mathbf{r}^{(l)} = \mathbf{q}^H\mathbf{r}$ in Algorithmus (3.6.17))

MATLAB-Funktion: `[x,flag,relr,it,rv] = pcg(A,b,tol,maxit,B,[],x0);`
(A, B können Handles auf Funktionen sein, die $\mathbf{Ax}$ bzw. $\mathbf{B}^{-1}\mathbf{x}$ realisieren)

Interne MATLAB-Implementierung:

MATLAB-CODE : PCG-Algorithmus

```
function x = pcg(Afun,b,tol,maxit,Binvfun,x0)
x = x0; r = b - feval(Afun,x); rho = 1;
for i = 1 : maxit
  y = feval(Binvfun,r);
  rho1 = rho; rho = r' * y;
  if (i == 1)
    p = y;
  else
    beta = rho / rho1;
    p = y + beta * p;
  end
  q = feval(Afun,p);
  alpha = rho /(p' * q);
  x = x + alpha * p;
  if (norm(b - evalf(Afun,x)) <= tolb*norm(b)), return; end
  r = r - alpha * q;
end
```

Zweifelhaftes Abbruchkriterium !

3.6.3 Weitere Krylov-Unterraumverfahren

3.6.3.1 Residuenminimierende Verfahren

Idee: Ersetze Euklidisches Skalarprodukt $\longrightarrow$ $\mathbf{A}$-Skalarprodukt im CG-Algorithmus

$$\left\|\mathbf{x}^{(l)}-\mathbf{x}\right\|_A \quad\rightarrow\quad \left\|\mathbf{A}(\mathbf{x}^{(l)}-\mathbf{x})\right\|_2 = \left\|\mathbf{r}^{(l)}\right\|_2$$

► MINRES-Verfahren (für beliebige Hermitesche Matrizen !)

Theorem 3.6.7. *Für $\mathbf{A}=\mathbf{A}^H$ erfüllen die Residuen $\mathbf{r}^{(l)}$ in der MINRES-Iteration*

$$\left\|\mathbf{r}^{(l)}\right\|_2 = \min\{\|\mathbf{Ay}-\mathbf{b}\|_2 : \mathbf{y}\in\mathbf{x}^{(0)}+\mathcal{K}_l(\mathbf{A},\mathbf{r}^{(0)})\}$$

$$\left\|\mathbf{r}^{(l)}\right\|_2 \le \frac{2\left(1-\frac{1}{\kappa(\mathbf{A})}\right)^l}{\left(1+\frac{1}{\kappa(\mathbf{A})}\right)^{2l}+\left(1-\frac{1}{\kappa(\mathbf{A})}\right)^{2l}}\left\|\mathbf{x}-\mathbf{x}^{(0)}\right\|_A .$$

Beachte: Konvergenzrate bestimmt durch $\kappa(\mathbf{A})$, nicht durch $\sqrt{\kappa(\mathbf{A})}$ wie bei CG !

MATLAB-Funktion:
- `[x,flg,res,it,resv] = minres(A,b,tol,maxit,B,[],x0);`
- `[...] = minres(Afun,b,tol,maxit,Binvfun,[],x0);`

Rechenaufwand : 1 $\mathbf{A}\times$Vektor, 1 $\mathbf{B}^{-1}\times$Vektor pro Schritt, wenige Skalarprodukte & SAXPYs
Speicheraufwand: Wenige Vektoren $\in\mathbb{K}^n$

Idee: Löse überbestimmtes Gleichungssystem

$$\mathbf{x}^{(l)}\in\mathbf{x}^{(0)}+\mathcal{K}_l(\mathbf{A},\mathbf{r}^{(0)}):\quad \mathbf{Ax}^{(l)}=\mathbf{b}$$

im Sinn der Methode der kleinsten Quadrate, $\longrightarrow$ Abschnitt 3.5 `sec:numer-line-ausgl`

$$\mathbf{x}^{(l)} = \operatorname{argmin}\{\|\mathbf{Ay}-\mathbf{b}\|_2 : \mathbf{y}\in\mathbf{x}^{(0)}+\mathcal{K}_l(\mathbf{A},\mathbf{r}^{(0)})\} .$$

➤ GMRES-Verfahren für allgemeine Matrizen $\mathbf{A}\in\mathbb{K}^{n,n}$

MATLAB-Funktion:
- `[x,flag,relr,it,rv] = gmres(A,b,rs,tol,maxit,B,[],x0);`
- `[...] = gmres(Afun,b,rs,tol,maxit,Binvfun,[],x0);`

Rechenaufwand : 1 $\mathbf{A}\times$Vektor, 1 $\mathbf{B}^{-1}\times$Vektor pro Schritt,
: $O(l)$ Skalarprodukte & SAXPYs im l. Schritt
Speicheraufwand: $O(l)$ Vektoren $\in\mathbb{K}^n$ im l. Schritt

Bemerkung 136 (GMRES und Arnoldi-Prozess). GMRES ↔ Arnoldi-Prozess, → Abschnitt 3.4.3 (vgl. Bem. 127) ➢ Lange Rekursionen ▷ wachsender Speicher- und Rechenaufwand

Bemerkung 137 (Neustart von GMRES). Wenn Speicher- und Rechenaufwand zu gross wird: Starte GMRES-Iteration neu mit bisher erhaltener Näherung als Startwert → `rs`-Parameter = Neustart nach jeweils `rs` Schritten (Vorsicht: Gefahr der Stagnation **!**)

3.6.3.2 Verfahren mit kurzen Rekursionen

Idee: Gegeben $\mathbf{x}^{(0)} \in \mathbb{K}^n$ bestimme verbesserte Näherung $\mathbf{x}^{(l)}$ aus Petrov-Galerkin-Bedingung

$$\mathbf{x}^{(l)} \in \mathbf{x}^{(0)} + \mathcal{K}_l(\mathbf{A}, \mathbf{r}^{(0)}): \quad \mathbf{p}^H(\mathbf{b} - \mathbf{A}\mathbf{x}^{(l)}) = 0 \quad \forall \mathbf{p} \in W_l \,,$$

mit geeignetem Testraum W_l, $\dim W_l = l$, z.B. $W_l := \mathcal{K}_l(\mathbf{A}^H, \mathbf{r}^{(0)})$ (→ bi-conjugate gradients, BiCG)

➢ „Zoo" von Verfahren mit kurzen Rekursionen (→ Lanczos-Prozess) *für allgemeine Matrizen* $\mathbf{A} \in \mathbb{K}^{n,n}$, siehe [3].

MATLAB-Funktion:
- `[x,flag,r,it,rv] = bicgstab(A,b,tol,maxit,B,[],x0)`
- `[...] = bicgstab(Afun,b,tol,maxit,Binvfun,[],x0);`

Rechenaufwand : 2 $\mathbf{A}\times$Vektor, 2 $\mathbf{B}^{-1}\times$Vektor, 4 Skalarprodukte, 6 SAXPYs pro Schritt
Speicheraufwand: 8 Vektoren $\in \mathbb{K}^n$

MATLAB-Funktion:
- `[x,flag,r,it,rv] = qmr(A,b,tol,maxit,B,[],x0)`
- `[...] = qmr(Afun,b,tol,maxit,Binvfun,[],x0);`

Rechenaufwand : 2 $\mathbf{A}\times$Vektor, 2 $\mathbf{B}^{-1}\times$Vektor, 2 Skalarprodukte, 12 SAXPYs pro Schritt
Speicheraufwand: 10 Vektoren $\in \mathbb{K}^n$

Keine Garantie für Konvergenz bei diesen Verfahren:
Stagnation & Breakdowns „an der Tagesordnung"

Beispiel 138 (Versagen von Krylov-Raum basierten iterativen Lösern).

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & \cdots & & \cdots & 0 \\ 0 & 0 & 1 & 0 & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & & & \\ & & & & \ddots & \ddots & \vdots \\ \vdots & & & & \ddots & \ddots & 0 \\ 0 & & & & & 0 & 1 \\ 1 & 0 & \cdots & & & \cdots & 0 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 0 \\ \vdots \\ \\ \\ \vdots \\ 0 \\ 1 \end{pmatrix} \quad ➢ \quad \mathbf{x} = \mathbf{e}_1 \,.$$

$$\mathbf{x}^{(0)} = 0 \quad ➢ \quad \mathbf{r}^{(0)} = \mathbf{e}_n \quad ➢ \quad \mathcal{K}_l(\mathbf{A}, \mathbf{r}^{(0)}) = \mathrm{Span}\{\mathbf{e}_n, \mathbf{e}_{n-1}, \ldots, \mathbf{e}_{n-l+1}\}$$

$$➢ \quad \min\{\|\mathbf{y} - \mathbf{x}\|_2 : \mathbf{y} \in \mathcal{K}_l(\mathbf{A}, \mathbf{r}^{(0)})\} = \begin{cases} 1 & \text{, falls } l \le n \,, \\ 0 & \text{, für } l = n \,. \end{cases}$$

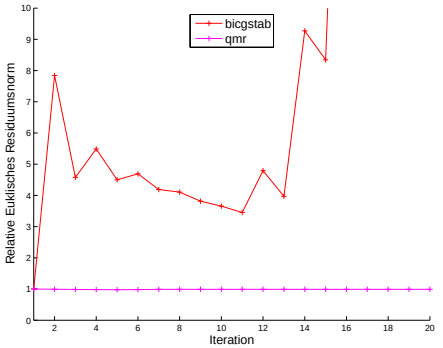
◇

Was soll der Anwender tun**?** Ausprobieren („try and pray")

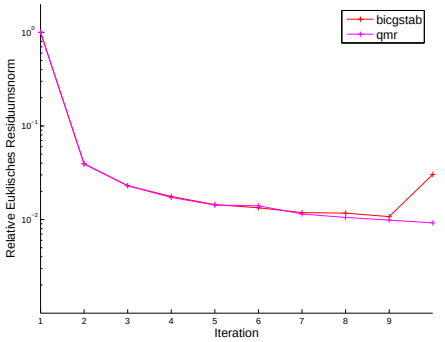
Beispiel 139 (Konvergenz von Krylovverfahren für nichtsymmetrische Matrix).

```
A = gallery('tridiag',-0.5*ones(n-1,1),2*ones(n,1),-1.5*ones(n-1,1));
B = gallery('tridiag',0.5*ones(n-1,1),2*ones(n,1),1.5*ones(n-1,1));
```

Plots zeigen $\left\|\mathbf{r}^{(l)}\right\|_2 : \left\|\mathbf{r}^{(0)}\right\|_2$:



Tridiagonalmatrix $\mathbf{A}$



Tridiagonalmatrix $\mathbf{B}$

◇

3.7 Spezielle Matrizen

[File: section-specialmat.tex, SVN: section-specialmat.tex 1056 2006-10-18 15:52:41Z kalai]

SPEZMAT

Spezielle Algorithmen für $n \times n$-Matrizen mit $O(n)$ Information (Bsp. → Bandmatrizen)

3.7.1 Diskrete Fouriertransformationen

DISKRFOURIERTRAFO

$n \in \mathbb{N}$ fest ➢ n. Einheitswurzel $\omega_n := \exp(-2\pi i/n) = \cos(2\pi i/n) - i\sin(2\pi i/n)$

$$\Rightarrow \quad \omega_n^k = \omega_n^{k+n} \; \forall k \in \mathbb{Z} \quad , \quad \omega_n^n = 1 \quad , \quad \omega_n^{n/2} = -1 \; , \tag{3.7.1}$$ eq:compl

$$\sum_{k=0}^{n-1} \omega_n^{kj} = \begin{cases} n & \text{, falls } j = 0 \; , \\ 0 & \text{, falls } j \neq 0 \; . \end{cases} \tag{3.7.2}$$ eq:compl

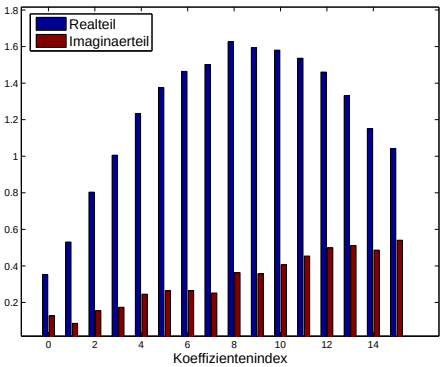
Eine Basistransformation in $\mathbb{C}^n$:

Standardbasis des $\mathbb{K}^n$ — TRIGBASIS — Trigonometrische Basis

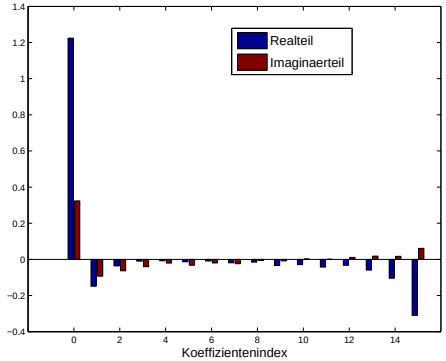
$$\left\{ \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \cdots \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ \vdots \\ \\ \vdots \\ 0 \\ 1 \end{pmatrix} \right\} \longleftarrow \left\{ \begin{pmatrix} \omega_n^0 \\ \vdots \\ \\ \vdots \\ \omega_n^0 \end{pmatrix} \begin{pmatrix} \omega_n^0 \\ \omega_n^1 \\ \vdots \\ \\ \vdots \\ \omega_n^{n-1} \end{pmatrix} \cdots \begin{pmatrix} \omega_n^0 \\ \omega_n^{n-2} \\ \omega_n^{2(n-2)} \\ \vdots \\ \\ \vdots \\ \omega_n^{(n-1)(n-2)} \end{pmatrix} \begin{pmatrix} \omega_n^0 \\ \omega_n^{n-1} \\ \omega_n^{2(n-1)} \\ \vdots \\ \\ \vdots \\ \omega_n^{(n-1)^2} \end{pmatrix} \right\}$$

Matrix der Basistransformation Trigonometrische Basis ⟶ Standardbasis: Fourier-Matrix

$$\mathbf{F}_n = \begin{pmatrix} \omega_n^0 & \omega_n^0 & \cdots & \omega_n^0 \\ \omega_n^0 & \omega_n^1 & \cdots & \omega_n^{n-1} \\ \omega_n^0 & \omega_n^2 & \cdots & \omega_n^{2n-2} \\ \vdots & \vdots & & \vdots \\ \omega_n^0 & \omega_n^{n-1} & \cdots & \omega_n^{(n-1)^2} \end{pmatrix} = \left(\omega_n^{ij}\right)_{i,j=0}^{n-1} \in \mathbb{C}^{n,n} \; . \tag{3.7.3}$$ dft:FM



Vektor in Standardbasis



Vektor in trigonometrischer Basis

Definition 3.7.1 (Diskrete Fouriertransformation). *Die lineare Abbildung* $\mathcal{F}_n : \mathbb{C}^n \mapsto \mathbb{C}^n$, $\mathcal{F}_n(\mathbf{y}) := \mathbf{F}_n\mathbf{y}$, $\mathbf{y} \in \mathbb{C}^n$, *heisst diskrete Fouriertransformation (DFT), d.h. für* $\mathbf{c} := \mathcal{F}_n(\mathbf{y})$

$$c_k = \sum_{j=0}^{n-1} y_j \, \omega_n^{kj} \quad , \quad k = 0, \ldots, n-1 \; . \tag{3.7.4}$$ eq:DFT

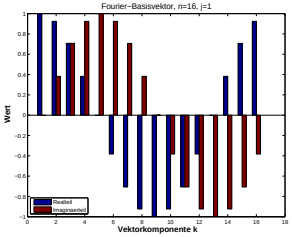
Konvention: Bei Diskussion der DFT: Vektorindizes laufen von 0 bis $n-1$.

Lemma 3.7.2 (Abbildungseigenschaft der diskreten Fourier-Transformation).
Die skalierte Fourier-Matrix $\frac{1}{\sqrt{n}}\mathbf{F}_n$ *ist unitär:* $\mathbf{F}_n^{-1} = \frac{1}{n}\mathbf{F}^H = \frac{1}{n}\overline{\mathbf{F}}_n$

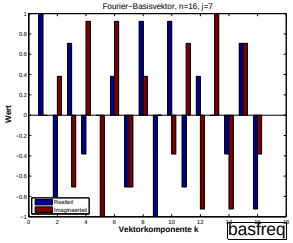
MATLAB-Funktionen: `c=fft(y)` ↔ `y=ifft(c);`

Beispiel 140 (Frequenzanalyse mit DFT).

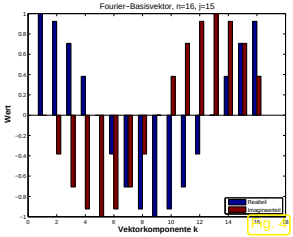
Einige Vektoren der Fourierbasis ($n = 16$): ⟶ Fig. 7 speccircle



„niederfrequent”



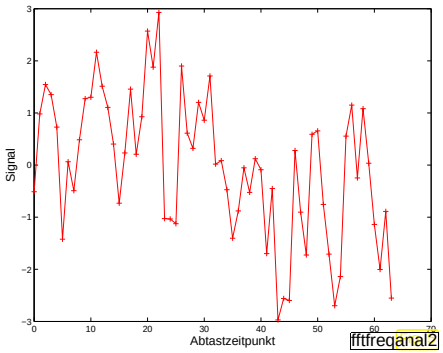
„hochfrequent” basfreq



„niederfrequent”

Extraktion charakteristischer Frequenzen aus gestörtem, zeitdiskretem und periodischem Signal:

```
t = 0:63; x = sin(2*pi*t/64)+sin(7*2*pi*t/64);
y = x + randn(size(t));
```



fftfreqanal2

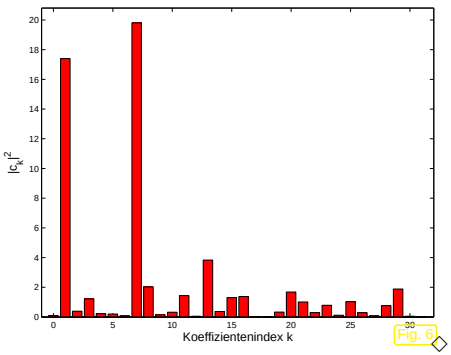


Fig. 6

„Naive” Auswertung von $\mathbf{Fy}$ → $O(n^2)$ Rechenoperationen

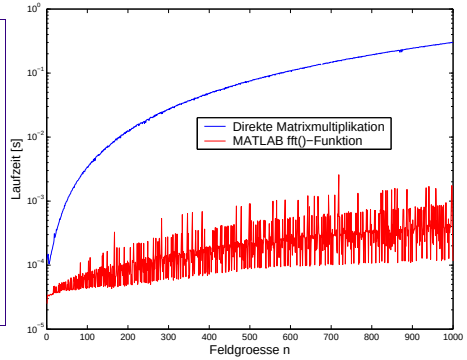
Beispiel 141 (Effizienz von `fft`).

`tic-toc`-Zeitmessung in MATLAB: Vergleich von `fft` und direkter Matrixmultiplikation
(MATLAB V6.5, Linux, Mobile Intel Pentium 4 - M CPU 2.40GHz, Minimum über 100 Runs)

◇

MATLAB-Code Naive DFT-Implementierung

```
c = zeros(n,1);
omega = exp(-2*pi*i/n);
c(1) = sum(y); s = omega;
for j=2:n
  c(j) = y(n);
  for k=n-1:-1:1
    c(j) = c(j)*s+y(k);
  end
  s = s*omega;
end
```



3.7.1.1 Schnelle Fouriertransformation

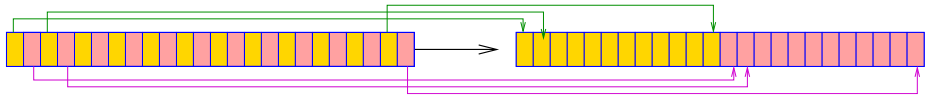
SCHNELLFOUTRAFO

„Milleniums-Algorithmus” *Schnelle Fouriertransformation (FFT)* (C.F. Gauss, siehe DUV90 [13])

`http://orion.math.iastate.edu/burkardt/misc/algorithms_dongarra.html`

Für $n = 2m$, $m \in \mathbb{N}$,

Permutation $P_m^{\mathrm{OE}}(1,\ldots,n) = (1,3,\ldots,n-1,2,4,\ldots,n)$.



Wegen $\omega_n^{2j} = \omega_m^j$:

Zeilenpermutation $P_m^{\mathrm{OE}}\mathbf{F}_n =$

$$\left(\begin{array}{c|c} \mathbf{F}_m & \mathbf{F}_m \\ \hline \mathbf{F}_m \begin{pmatrix} \omega_n^0 & & & \\ & \omega_n^1 & & \\ & & \ddots & \\ & & & \omega_n^{n/2-1} \end{pmatrix} & \mathbf{F}_m \begin{pmatrix} \omega_n^{n/2} & & & \\ & \omega_n^{n/2+1} & & \\ & & \ddots & \\ & & & \omega_n^{n-1} \end{pmatrix} \end{array}\right) =$$

$$\left(\begin{array}{c|c} \mathbf{F}_m & \\ \hline & \mathbf{F}_m \end{array}\right) \left(\begin{array}{c|c} \mathbf{I} & \mathbf{I} \\ \hline \begin{pmatrix} \omega_n^0 & & & \\ & \omega_n^1 & & \\ & & \ddots & \\ & & & \omega_n^{n/2-1} \end{pmatrix} & \begin{pmatrix} -\omega_n^0 & & & \\ & -\omega_n^1 & & \\ & & \ddots & \\ & & & -\omega_n^{n/2-1} \end{pmatrix} \end{array}\right)$$

Beispiel 142 (Zerlegung der Fourier-Matrix).

$$P_5^{\mathrm{OE}}\mathbf{F}_{10} = \left(\begin{array}{ccccc|ccccc} \omega^0 & \omega^0 & \omega^0 & \omega^0 & \omega^0 & \omega^0 & \omega^0 & \omega^0 & \omega^0 & \omega^0 \\ \omega^0 & \omega^2 & \omega^4 & \omega^6 & \omega^8 & \omega^0 & \omega^2 & \omega^4 & \omega^6 & \omega^8 \\ \omega^0 & \omega^4 & \omega^8 & \omega^2 & \omega^6 & \omega^0 & \omega^4 & \omega^8 & \omega^2 & \omega^6 \\ \omega^0 & \omega^6 & \omega^2 & \omega^8 & \omega^4 & \omega^0 & \omega^6 & \omega^2 & \omega^8 & \omega^4 \\ \omega^0 & \omega^8 & \omega^6 & \omega^4 & \omega^2 & \omega^0 & \omega^8 & \omega^6 & \omega^4 & \omega^2 \\ \hline \omega^0 & \omega^1 & \omega^2 & \omega^3 & \omega^4 & \omega^5 & \omega^6 & \omega^7 & \omega^8 & \omega^9 \\ \omega^0 & \omega^3 & \omega^6 & \omega^9 & \omega^2 & \omega^5 & \omega^8 & \omega^1 & \omega^4 & \omega^7 \\ \omega^0 & \omega^5 & \omega^0 & \omega^5 & \omega^0 & \omega^5 & \omega^0 & \omega^5 & \omega^0 & \omega^5 \\ \omega^0 & \omega^7 & \omega^4 & \omega^1 & \omega^8 & \omega^5 & \omega^2 & \omega^9 & \omega^6 & \omega^3 \\ \omega^0 & \omega^9 & \omega^8 & \omega^7 & \omega^6 & \omega^5 & \omega^4 & \omega^3 & \omega^2 & \omega^1 \end{array}\right) , \quad \omega := \omega_{10} .$$

◇

Rekursive Implementierung für $n = 2^p$, $p \in \mathbb{N}$

„Teile-und-herrsche" (engl. *divide and conquer*)

Rechenaufwand $O(n \log_2 n)$
MATLAB-`fft` $\approx 5n \log_2 n$

(Auf Rekursionstiefe $0 \le q < p$:
2^q Aufrufe, jeder 2^{p-q} Additionen,
2^{p-q-1} Multiplikationen)

MATLAB-Code Rekursiv Implementierung von FFT

```
function c = myfft(y)
n = length(y);
if (n == 1), c = y; return; end
omega = exp(-2*pi*i/n);
z = (omega.^((0:n-1)')).*y;
c = [myfft(y(1:n/2)+y(n/2+1:n));...
     myfft(z(1:n/2)+z(n/2+1:n))];
c = reshape(reshape(c,n/2,2)',n,1);
```

FFT basierend auf allgemeiner Faktorisierung: $n = pq$, $p, q \in \mathbb{N}$ (Cooley-Tuckey-Algorithmus)

$$c_k = \sum_{j=0}^{n-1} y_j \omega_n^{jk} \overset{[j=:lp+m]}{=} \sum_{m=0}^{p-1} \sum_{l=0}^{q-1} y_{lp+m} e^{-\frac{2\pi i}{pq}(lp+m)k} = \sum_{m=0}^{p-1} \omega_n^{mk} \sum_{l=0}^{q-1} y_{lp+m} \omega_q^{l(k \mod q)}$$

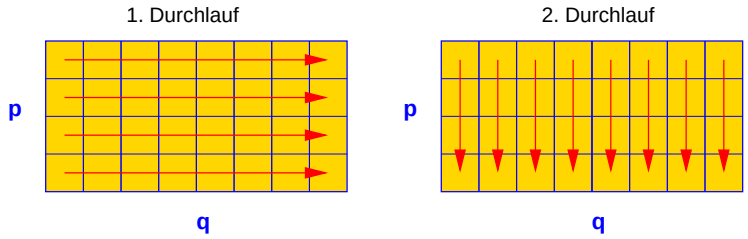
(3.7.5) fft:factor

► p DFTs der Länge q berechnen $z_{m,k} := \sum_{l=0}^{q-1} y_{lp+m} \omega_q^{lk}$, $0 \le m < p, 0 \le k < q$.

► für $k =: rq + s$, $0 \le r < p, 0 \le s < q$

$$c_{rq+s} = \sum_{m=0}^{p-1} e^{-\frac{2\pi i}{pq}(rq+s)m} z_{m,s} = \sum_{m=0}^{p-1} \left(\omega_n^{ms} z_{m,s}\right) \omega_p^{mr}$$

➢ q DFTs der Länge p berechnen alle c_k.



Aus dem MATLAB-Manual:

To compute an n-point DFT when n is composite (that is, when $n = pq$), the FFTW library decomposes the problem using the Cooley-Tukey algorithm, which first computes p transforms of size q, and then computes q transforms of size p. The decomposition is applied recursively to both the p- and q-point DFTs until the problem can be solved using one of several machine-generated fixed-size codelets.The codelets in turn use several algorithms in combination, including a variation of Cooley-Tukey, a prime factor algorithm, and a split-radix algorithm. The particular factorization of is chosen heuristically.

The execution time for fft depends on the length of the transform. It is fastest for powers of two. It is almost as fast for lengths that have only small prime factors. It is typically several times slower for lengths that are prime or which have large prime factors → Bsp. 141. ex:ffttime

Bemerkung 143 (Zweidimensionale diskrete Fouriertransformation).

Zu $y_{j_1,j_2} \in \mathbb{C}, 0 \le j_1 < n_1, 0 \le j_2 < n_2$ berechne (geschachtelte DFTs !)

$$c_{k_1,k_2} = \sum_{j_1=0}^{n_1-1} \sum_{j_2=0}^{n_2-1} y_{j_1,j_2} \omega_{n_1}^{j_1 k_1} \omega_{n_2}^{j_2 k_2} \quad , \quad 0 \le k_1 < n_1 \, , \, 0 \le k_2 < n_2 \, .$$

MATLAB-Kommando: fft2(X)

△

REELLEDFT

Bemerkung 144 (Reelle DFT).

Aufgabe: Effiziente Berechnung der DFT (Def. 3.7.1 def:DFT) $(c_0, \ldots, c_{n-1})$ für reelle Koeffizienten $(y_0, \ldots, y_{n-1})^T \in \mathbb{R}^n$, $n = 2m$, $m \in \mathbb{N}$

Redundanz:

$$\omega_n^{(n-k)j} = \overline{\omega_n^{kj}} \, , \quad k = 0, \ldots, n-1 \quad \blacktriangleright \quad c_{n-k} = \overline{c}_k \, , \quad k = 1, \ldots, n-1 \, .$$

➤ Idee: Komplexifizierung & Rückführung auf DFT der Länge m.

$$h_k = \sum_{j=0}^{m-1} (y_{2j} + i y_{2j+1}) \omega_m^{jk} = \boxed{\textstyle\sum_{j=0}^{m-1} y_{2j} \omega_m^{jk}} + i \cdot \boxed{\textstyle\sum_{j=0}^{m-1} y_{2j+1} \omega_m^{jk}} \, , \qquad (3.7.6) \text{ fft:1}$$

$$\overline{h}_{m-k} = \sum_{j=0}^{m-1} \overline{y_{2j} + i y_{2j+1}} \, \overline{\omega}_m^{j(m-k)} = \boxed{\textstyle\sum_{j=0}^{m-1} y_{2j} \omega_m^{jk}} - i \cdot \boxed{\textstyle\sum_{j=0}^{m-1} y_{2j+1} \omega_m^{jk}} \, . \qquad (3.7.7) \text{ fft:2}$$

fft:factor

Speziallfall von (3.7.5) ($p = 2$):

$$c_k = \sum_{j=0}^{n-1} y_j \omega_n^{jk} = \boxed{\textstyle\sum_{j=0}^{m-1} y_{2j} \omega_m^{jk}} + \omega_n^k \cdot \boxed{\textstyle\sum_{j=0}^{m-1} y_{2j+1} \omega_m^{jk}} \, . \qquad (3.7.8) \text{ fft:3}$$

$$c_k = \tfrac{1}{2}(h_k + \overline{h}_{m-k}) - \tfrac{1}{2} i \omega_n^k (h_k - \overline{h}_{m-k}) \, , \quad k = 0, \ldots, m-1 \, ,$$

$$\blacktriangleright \quad c_m = \operatorname{Re}\{h_0\} - \operatorname{Im} h_0 \, , \qquad (3.7.9) \text{ fft:4}$$

$$c_k = \overline{c}_{n-k} \, , \quad k = m+1, \ldots, n-1 \, .$$

MATLAB-Implementierung (auf der Grundlage einer `fft` der Länge $n/2$):

Ineffizient !

```
function c = fftreal(y)
n = length(y); m = n/2;
if (mod(n,2) ~= 0), error('n odd!'); end
y = y(1:2:n)+i*y(2:2:n);
h = fft(y); h = [h;h(1)];
c = 0.5*(h+conj(h(m+1:-1:1))) - ...
    (0.5*i*exp(-2*pi*i/n).^((0:m)')).*...
    (h-conj(h(m+1:-1:1)));
c = [c;conj(c(m:-1:2))];
```

△

FREQFILTDISKRSIG

Beispiel 145 (Frequenzfilterung diskreter Signale).

Frequenzfilterung diskreter periodischer Signale durch Ausblenden von „Fourierkoeffizienten"

```
m = length(y)/2;
c = fft(y);
clow = c;
clow(m+1-k:m+1+k) = 0;
chigh = c-clow;
low = real(ifft(clow));
high = real(ifft(chigh));
```

(Optimierungspotential wegen $y_j \in \mathbb{R}$ und $c_{n/2-k} = \overline{c}_{n/2+k}$)

speccircle

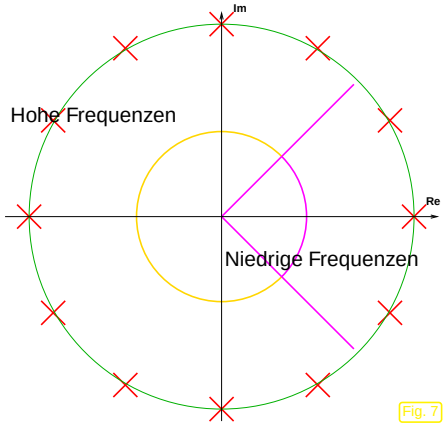
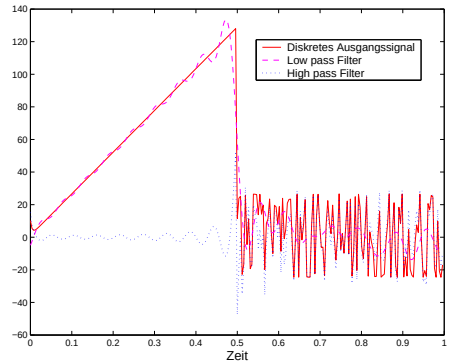
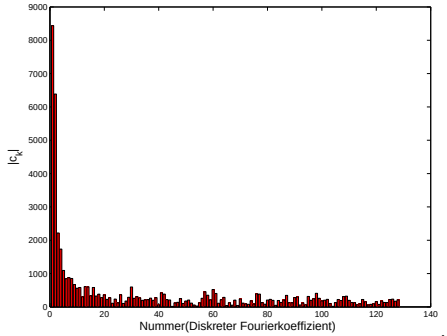


Fig. 7

Konkretes Signal:

```
k = 110; n = 256; m = n/2;
y = [(1:m)';ones(m,1)] + n/10*sin(exp([-(1:m)';(1:m)']));
```



◇

Übung 3.8. Implement an efficient routine `x=solvetrig(m,y)` for the solution of a linear system of equations $\mathbf{M}\mathbf{x} = \mathbf{y}$ for $\mathbf{y} \in \mathbb{C}^p$, $p = 2m+1$, $m \in \mathbb{N}$, and

$$\mathbf{M} := \begin{pmatrix} 1 & 1 & 1 & \cdots & \cdots & 1 \\ \cos(\varphi_1) & \cos(\varphi_2) & \cos(\varphi_3) & \cdots & \cdots & \cos(\varphi_p) \\ \sin(\varphi_1) & \sin(\varphi_2) & \sin(\varphi_3) & \cdots & \cdots & \sin(\varphi_p) \\ \cos(2\varphi_1) & \cos(2\varphi_2) & \cos(2\varphi_3) & \cdots & \cdots & \cos(2\varphi_p) \\ \sin(2\varphi_1) & \sin(2\varphi_2) & \sin(2\varphi_3) & \cdots & \cdots & \sin(2\varphi_p) \\ \vdots & \vdots & \vdots & & & \vdots \\ \cos(m\varphi_1) & \cos(m\varphi_2) & \cos(m\varphi_3) & \cdots & \cdots & \cos(m\varphi_p) \\ \sin(m\varphi_1) & \sin(m\varphi_2) & \sin(m\varphi_3) & \cdots & \cdots & \sin(m\varphi_p) \end{pmatrix},$$

where $\varphi_j = \frac{2\pi}{p}(j-1)$, $j = 1, \ldots, p$.

Hint: First compute $\mathbf{M}\mathbf{M}^T$ and establish that this yields a diagonal matrix.

3.8.0.2 Sinustransformation

SINTRAFO

Eine weitere trigonometrische Basistransformation in $\mathbb{R}^{n-1}$, $n \in \mathbb{N}$:

Standardbasis des $\mathbb{R}^{n-1}$ „Sinusbasis”

$$\left\{\begin{pmatrix}1\\0\\\vdots\\0\end{pmatrix}\begin{pmatrix}0\\1\\0\\\vdots\\0\end{pmatrix}\cdots\begin{pmatrix}0\\\vdots\\0\\1\\0\end{pmatrix}\begin{pmatrix}0\\\vdots\\0\\1\end{pmatrix}\right\} \longleftarrow \left\{\begin{pmatrix}\sin(\frac{\pi}{n})\\\sin(\frac{2\pi}{n})\\\vdots\\\sin(\frac{(n-1)\pi}{n})\end{pmatrix}\begin{pmatrix}\sin(\frac{2\pi}{n})\\\sin(\frac{4\pi}{n})\\\vdots\\\sin(\frac{2(n-1)\pi}{n})\end{pmatrix}\cdots\begin{pmatrix}\sin(\frac{(n-1)\pi}{n})\\\sin(\frac{2(n-1)\pi}{n})\\\vdots\\\sin(\frac{(n-1)^2\pi}{n})\end{pmatrix}\right\}$$

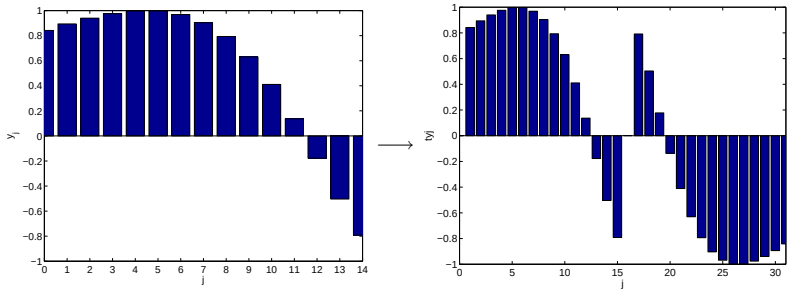
Matrix(Basistransformation Sinusbasis $\longrightarrow$ Standardbasis): $\mathbf{S}_n := (\sin(jk\pi/n))_{j,k=1}^{n-1} \in \mathbb{R}^{n-1,n-1}$

Lemma 3.8.1 (Eigenschaften der Sinusmatrix). $\sqrt{2/n}\,\mathbf{S}_n$ *ist reell, symmetrisch und unitär.*

Sinustransformation: $$s_k = \sum_{j=1}^{n-1} y_j \sin(\pi jk/n) \quad , \quad k = 1,\ldots,n-1 \; . \tag{3.8.1}$$ fft:sinft

DFT-basierter Algorithmus zur Auswertung der Sinustransformation ($\hat{=}$ $\mathbf{S}_n\times$Vektor):

„Wrap around”: $\widetilde{\mathbf{y}} \in \mathbb{R}^{2n}$: $\widetilde{y}_j = \begin{cases} y_j & \text{, falls } j = 1,\ldots,n-1 \; , \\ 0 & \text{, falls } j = 0, n \; , \\ -y_{2n-j} & \text{, falls } j = n+1,\ldots,2n-1 \; . \end{cases}$ ($\widetilde{\mathbf{y}}$ „punktsymmetrisch”)



$$(\mathbf{F}_{2n}\widetilde{\mathbf{y}})_k \overset{\text{eq:DFT (3.7.4)}}{=} \sum_{j=1}^{2n-1} \widetilde{y}_j e^{-\frac{2\pi}{2n}kj}$$
$$= \sum_{j=1}^{n-1} y_j e^{-\frac{\pi}{n}kj} - \sum_{j=n+1}^{2n-1} y_{2n-j} e^{-\frac{\pi}{n}kj}$$
$$= \sum_{j=1}^{n-1} y_j (e^{-\frac{\pi}{n}kj} - e^{\frac{\pi}{n}kj})$$
$$= -2i\,(\mathbf{S}_n\mathbf{y})_k \quad , k = 1,\ldots,n-1 \; .$$

Wrap-around Implementierung

```
function c = sinetrans(y)
n = length(y)+1;
yt = [0,y,0,-y(end:-1:1)];
ct = fft(yt);
c = -ct(2:n)/(2*i);
```

MATLAB-Code Sinustransformation

Bemerkung 146 (Sinustransformation per DFT halber Länge).

Schritt ①: Transformation der Koeffizienten

$$\widetilde{y}_j = \sin(j\pi/n)(y_j + y_{n-j}) + \tfrac{1}{2}(y_j - y_{n-j}) \; , \quad j = 1,\ldots,n-1 \quad , \quad \widetilde{y}_0 = 0 \; .$$

Schritt ②: Reelle DFT (→ Def. 3.7.1) von $(\widetilde{y}_0,\ldots,\widetilde{y}_{n-1}) \in \mathbb{R}^n$: $c_k := \sum_{j=0}^{n-1} \widetilde{y}_j\, e^{-\frac{2\pi i}{n}jk}$ def:DFT

$$\mathrm{Re}\{c_k\} = \sum_{j=0}^{n-1} \widetilde{y}_j \cos(-\tfrac{2\pi i}{n}jk) = \sum_{j=1}^{n-1} (y_j + y_{n-j}) \sin(\tfrac{\pi j}{n}) \cos(\tfrac{2\pi i}{n}jk)$$
$$= \sum_{j=0}^{n-1} 2y_j \sin(\tfrac{\pi j}{n}) \cos(\tfrac{2\pi i}{n}jk) = \sum_{j=0}^{n-1} y_j \left(\sin(\frac{2k+1}{n}\pi j) - \sin(\frac{2k-1}{n}\pi j)\right)$$
$$= s_{2k+1} - s_{2k-1} \; .$$
$$\mathrm{Im}\{c_k\} = \sum_{j=0}^{n-1} \widetilde{y}_j \sin(-\tfrac{2\pi i}{n}jk) = -\sum_{j=1}^{n-1} \tfrac{1}{2}(y_j - y_{n-j}) \sin(\tfrac{2\pi i}{n}jk) = -\sum_{j=1}^{n-1} y_j \sin(\tfrac{2\pi i}{n}jk)$$
$$= -s_{2k} \; .$$

Schritt ③: Extraktion der s_k

s_{2k+1} , $k = 0,\ldots,\frac{n}{2}-1$ ➤ Aus Rekursion $s_{2k+1} - s_{2k-1} = \mathrm{Re}\{c_k\}$, $s_1 = \sum_{j=1}^{n-1} y_j \sin(\pi j/n)$,

s_{2k}, $k = 1,\ldots,\frac{n}{2}-2$ ➤ $s_{2k} = -\mathrm{Im}\{c_k\}$.

MATLAB-Implementierung (basierend auf einer `fft` der Länge $n/2$):

MATLAB-CODE Sinustransformation

```
function s = sinetrans(y)
n = length(y)+1;
sinevals = imag(exp(i*pi/n).^(1:n-1));
yt = [0 (sinevals.*(y+y(end:-1:1)) + 0.5*(y-y(end:-1:1)))];
c = fftreal(yt);
s(1) = dot(sinevals,y);
for k=2:N-1
if (mod(k,2) == 0), s(k) = -imag(c(k/2+1));
else, s(k) = s(k-2) + real(c((k-1)/2+1)); end
end
```

△

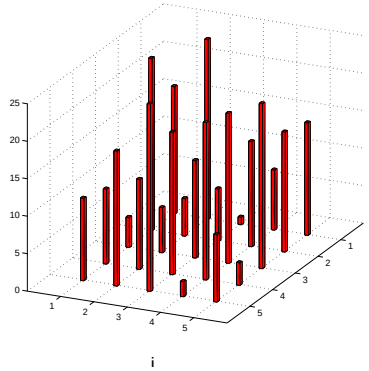
ANWSINUSTRAFO
Anwendung: Diagonalisierung lokaler translationsinvarianter linearer Operatoren

FUENFPKTSTERN
5-Punkt-Stern-Operator auf $\mathbb{R}^{n,n}$, $n \in \mathbb{N}$, in Gitterdarstellung:

$$T : \mathbb{R}^{n,n} \mapsto \mathbb{R}^{n,n}, \quad \begin{array}{c} \mathbf{X} \mapsto T(\mathbf{X}) \\ (T(\mathbf{X}))_{ij} := cx_{ij} + c_y x_{i,j+1} + c_y x_{i,j-1} + c_x x_{i+1,j} + c_x x_{i-1,j} \end{array}$$

mit $c, c_y, c_x \in \mathbb{R}$, Konvention: $x_{ij} := 0$ für $(i,j) \notin \{1,\ldots,n\}^2$.

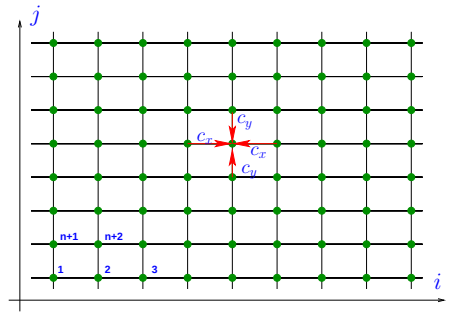
$$\mathbf{X} \in \mathbb{R}^{n,n}$$
$$\Updownarrow$$
GITTERFKT
Gitterfunktion $\in \{1,\ldots,n\}^2 \mapsto \mathbb{R}$



Identifikation $\mathbb{R}^{n,n} \cong \mathbb{R}^{n^2}$, $x_{ij} \sim \widetilde{x}_{(j-1)n+i}$ ➤ Matrixdarstellung $\mathbf{T} \in \mathbb{R}^{n^2,n^2}$ von T:

$$\mathbf{T} = \begin{pmatrix} \mathbf{C} & c_y\mathbf{I} & 0 & \cdots & \cdots & 0 \\ c_y\mathbf{I} & \mathbf{C} & c_y\mathbf{I} & & & \vdots \\ 0 & \ddots & \ddots & \ddots & & \\ \vdots & & & c_y\mathbf{I} & \mathbf{C} & c_y\mathbf{I} \\ 0 & \cdots & \cdots & 0 & c_y\mathbf{I} & \mathbf{C} \end{pmatrix} \in \mathbb{R}^{n^2,n^2},$$

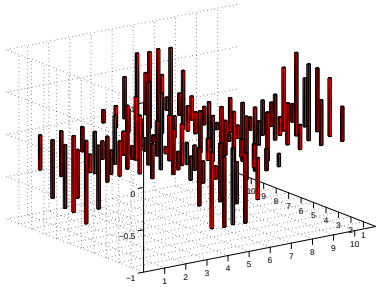
$$\mathbf{C} = \begin{pmatrix} c & c_x & 0 & \cdots & \cdots & 0 \\ c_x & c & c_x & & & \vdots \\ 0 & \ddots & \ddots & \ddots & & \\ \vdots & & & c_x & c & c_x \\ 0 & \cdots & \cdots & 0 & c_x & c \end{pmatrix} \in \mathbb{R}^{n,n}.$$



Sinusbasis von $\mathbb{R}^{n,n}$:

$$\mathbf{B}^{kl} = \left(\sin(\tfrac{\pi}{n+1}ki)\sin(\tfrac{\pi}{n+1}lj)\right)_{i,j=1}^{n}. \quad (3.8.2)$$

$n = 10$: Gitterfunktion $\mathbf{B}^{2,3}$ ➤



$$\begin{aligned} (T(\mathbf{B}^{kl}))_{ij} &= c\sin(\tfrac{\pi}{n}ki)\sin(\tfrac{\pi}{n}lj) + c_y\sin(\tfrac{\pi}{n}ki)\left(\sin(\tfrac{\pi}{n+1}l(j-1)) + \sin(\tfrac{\pi}{n+1}l(j+1))\right) + \\ &\quad c_x\sin(\tfrac{\pi}{n}lj)\left(\sin(\tfrac{\pi}{n+1}k(i-1)) + \sin(\tfrac{\pi}{n+1}k(i+1))\right) \\ &= \sin(\tfrac{\pi}{n}ki)\sin(\tfrac{\pi}{n}lj)(c + 2c_y\cos(\tfrac{\pi}{n+1}l) + 2c_x\cos(\tfrac{\pi}{n+1}k)) \end{aligned}$$

➤ $\mathbf{B}^{kl}$ ist Eigenvektor von $T \leftrightarrow \mathbf{T}$ zum Eigenwert $c + 2c_y\cos(\tfrac{\pi}{n+1}l) + 2c_x\cos(\tfrac{\pi}{n+1}k)$

Algorithmus für Basistransformation:

$$\mathbf{X} = \sum_{k=1}^{n}\sum_{l=1}^{n} y_{kl}\mathbf{B}^{kl} \quad \Rightarrow \quad x_{ij} = \sum_{k=1}^{n}\sin(\tfrac{\pi}{n+1}ki)\sum_{l=1}^{n} y_{kl}\sin(\tfrac{\pi}{n+1}lj).$$

➤ Geschachtelte Sinustransformationen (→ Bem. dft:2d 143) angewandt auf Zeilen/Spalten von $\mathbf{Y} = (y_{kl})_{k,l=1}^{n}$

Hier: Implementierung der Sinustransformation (3.8.1) mit fft:sinft „wrapping"-Technik

MATLAB-CODE Zweidimensionaler Sinustr.

```
function C = sinft2d(Y)
[m,n] = size(Y);
C = fft([zeros(1,n); Y;...
         zeros(1,n);...
         -Y(end:-1:1,:)]);
C = i*C(2:m+1,:)'/2;
C = fft([zeros(1,m); C;...
         zeros(1,m);...
         -C(end:-1:1,:)]);
C= i*C(2:n+1,:)'/2;
```

MATLAB-Code FFT-basiertes Lösen aus lokalen translationsinvarianten linearen Operatoren

```
function X = fftsolve(B,c,cx,cy)
[m,n] = size(B);
[I,J] = meshgrid(1:m,1:n);
X = 4*sinft2d(sinft2d(B)...
    ./(c+2*cx*cos(pi/(n+1)*I)+...
         2*cy*cos(pi/(m+1)*J)))...
     /((m+1)*(n+1));
```

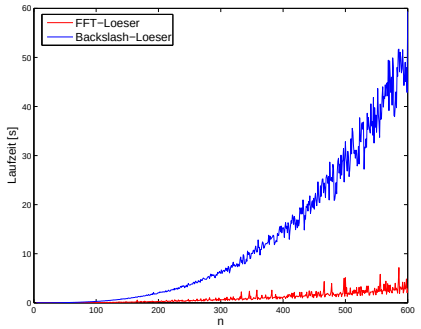
Diagonalisierung von $\mathbf{T}$
durch 2D Sinustransformation

Effizienter Algorithmus
zum Lösen des LGS $T(\mathbf{X}) = \mathbf{B}$
Rechenaufwand $O(n^2 \log n)$

Beispiel 147 (Effizienz FFT-basierter Gleichungslöser).

`tic-toc`-Zeitmessung (MATLAB V7, Linux, Intel Pentium 4 Mobile CPU 1.80GHz)

```
A = gallery('poisson',n);
B = magic(n);
b = reshape(B,n*n,1);
tic;
C = fftsolve(B,4,-1,-1);
t1 = toc;
tic; x = A\b; t2 = toc;
```



◇

3.8.0.3 Kosinustransformation

COSTRANS
Noch eine trigonometrische Basistransformation im $\mathbb{R}^n$, $n \in \mathbb{N}$:

Standardbasis des $\mathbb{R}^n$ „Kosinusbasis"

$$\left\{ \begin{pmatrix}1\\0\\\vdots\\0\end{pmatrix} \begin{pmatrix}0\\1\\0\\\vdots\\0\end{pmatrix} \cdots \begin{pmatrix}0\\\vdots\\0\\1\\0\end{pmatrix} \begin{pmatrix}0\\\vdots\\0\\1\end{pmatrix} \right\} \leftarrow \left\{ \begin{pmatrix} 2^{-1/2} \\ \cos(\frac{\pi}{2n}) \\ \cos(\frac{2\pi}{2n}) \\ \vdots \\ \cos(\frac{(n-1)\pi}{2n}) \end{pmatrix} \begin{pmatrix} 2^{-1/2} \\ \cos(\frac{3\pi}{2n}) \\ \cos(\frac{6\pi}{2n}) \\ \vdots \\ \cos(\frac{3(n-1)\pi}{2n}) \end{pmatrix} \cdots \begin{pmatrix} 2^{-1/2} \\ \cos(\frac{(2n-1)\pi}{2n}) \\ \cos(\frac{2(2n-1)\pi}{2n}) \\ \vdots \\ \cos(\frac{(n-1)(2n-1)\pi}{2n}) \end{pmatrix} \right\}$$

Matrix(Basistransformation Kosinusbasis ⟶ Standardbasis):

$$\mathbf{C}_n = (c_{ij}) \in \mathbb{R}^{n,n} \quad \text{mit} \quad c_{ij} = \begin{cases} 2^{-1/2} & \text{, falls } i = 1 \text{ ,} \\ \cos((i-1)\frac{2j-1}{2n}\pi) & \text{, falls } i > 1 \text{ .} \end{cases}$$

Lemma 3.8.2 (Eigenschaften der Kosinusmatrix). $\sqrt{2/n}\,\mathbf{C}_n$ *ist reell und unitär (aber nicht symmetrisch).*

Kosinustransformation: $c_k = \sum_{j=0}^{n-1} y_j \cos(k\frac{2j+1}{2n}\pi)$, $k = 1, \ldots, n-1$, (3.8.3) fft:cosfl

$$c_0 = \frac{1}{\sqrt{2}} \sum_{j=0}^{n-1} y_j \text{ .}$$

MATLAB-Implementierung von $\mathbf{Cy}$ („Wrapping"-Technik)

MATLAB-Code Kosinustansformation

```
function c = costrans(y)
n = length(y);
z = fft([y,y(end:-1:1)]);
c = real([z(1)/(2*sqrt(2)), ...
     0.5*(exp(-i*pi/(2*n)).^(1:n-1)).*z(2:n)]);
```

MATLAB-Implementierung von $\mathbf{C}^{-1}\mathbf{y}$ („Wrapping"-Technik):

MATLAB-Code : Inverse Kosinustransformation

```
function y=icostrans(c)
n = length(c);
y = [sqrt(2)*c(1),(exp(i*pi/(2*n)).^(1:n-1)).*c(2:end)];
y = ifft([y,0,conj(y(end:-1:2))]);
y = 2*y(1:n);
```


3.8.1 Zirkulante Matrizen

ZIRKMAT
Folge $(x_k)_{k\in\mathbb{Z}} = (\ldots, x_{-2}, x_{-1}, x_0, x_1, x_2, \ldots)$ ist n-periodisch, wenn $x_k = x_{n+k}$ $\forall k \in \mathbb{Z}$.
(⟶ festgelegt durch $x_0, \ldots, x_{n-1}$)

DISCVONV
Definition 3.8.3. *Die diskrete Faltung (engl.* discrete convolution*) zweier n-periodischer Folgen $(x_k)_{k\in\mathbb{Z}}$, $(y_k)_{k\in\mathbb{Z}}$* ➡ *n-periodische Folge*

$$z_k := (x_k) * (y_k) := \sum_{j=0}^{n-1} x_{k-j} y_j = (y_k) * (x_k) := \sum_{j=0}^{n-1} y_{k-j} x_j \text{ , } \quad k \in \mathbb{Z} \text{ .}$$

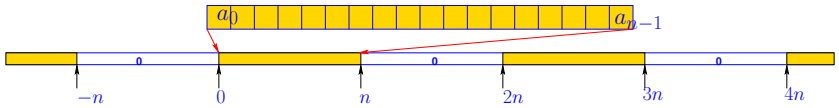
$$\begin{pmatrix} z_0 \\ \vdots \\ \\ \\ \\ \vdots \\ z_{n-1} \end{pmatrix} = \begin{pmatrix} x_0 & x_{n-1} & x_{n-2} & \cdots & & \cdots & x_1 \\ x_1 & x_0 & x_{n-1} & & & & \vdots \\ x_2 & x_1 & x_0 & \ddots & & & \\ \vdots & & \ddots & \ddots & \ddots & & \\ & & & \ddots & \ddots & \ddots & \\ \vdots & & & & \ddots & \ddots & x_{n-1} \\ x_{n-1} & \cdots & & & & x_1 & x_0 \end{pmatrix} \begin{pmatrix} y_0 \\ \vdots \\ \\ \\ \\ \vdots \\ y_{n-1} \end{pmatrix}$$

Beispiel 148 (Multiplikation von Polynomen).

$$p(z)=\sum_{j=0}^{n-1} a_j z^j\ ,\quad q(z)=\sum_{j=0}^{n-1} b_j z^j \quad\blacktriangleright\quad (pq)(z)=\sum_{j=0}^{2n-2}\Big(\sum_{k=0}^{j} a_k b_{j-k}\Big) z^j\ .$$

Konstruiere $2n-1$-periodische Koeffizientenfolgen

$$x_k := \begin{cases} a_k & \text{, falls } 0\le k<n\ ,\\ 0 & \text{, falls } n\le k<2n-1\end{cases}\ ,\quad y_k := \begin{cases} b_k & \text{, falls } 0\le k<n\ ,\\ 0 & \text{, falls } n\le k<2n-1\ .\end{cases}$$



$\blacktriangleright$ k. Koeffizient von pq: $c_j=((x_k)*(y_k))_j\ ,\quad j=0,\dots,2n-2\ .$

$\diamond$

ZIRKULMAT

Definition 3.8.4 (Zirkulante Matrix)**.**

Matrix $\mathbf{C}=(c_{ij})_{i,j=1}^{n}\in\mathbb{K}^{n,n}$ = *zirkulant* $:\Leftrightarrow \exists\,(u_k)_{k\in\mathbb{Z}}$ *n-periodisch:* $c_{ij}=u_{i-j}$, $1\le i,j\le n$.

- ☞ Zirkulante Matrix hat konstante (Neben-)Diagonalen $i-j=\text{const}$.
- ☞ Zeilen zyklisch verschoben.

$$\begin{pmatrix} u_0 & u_1 & u_2 & \cdots & & \cdots & u_{n-1}\\ u_{n-1} & u_0 & & & & & u_{n-2}\\ u_{n-2} & & & & & & \vdots\\ \vdots & & & & & & u_1\\ & & & & & & \\ u_1 & \cdots & & & \cdots & u_{n-1} & u_0\end{pmatrix}$$

STRAHLTRANSP

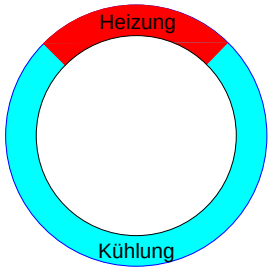
Beispiel 149 (Strahlungstransport).

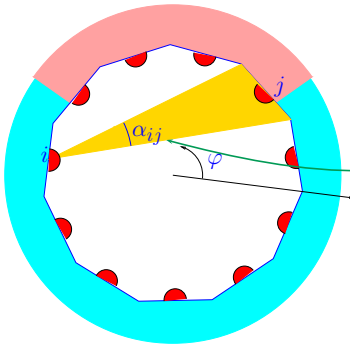
Zylindrisches Rohr:

Heizung auf Teilen Γ_H des Umfangs (→ Wärmefluss vorgegeben)

Kühlung auf restlichem Umfang Γ_K (→ konstanter Wärmeabfluss)

Zu berechnen: Lokaler Wärmefluss





Modellierung:

- Approximation durch reguläres n-Eck, Seiten Γ_j
- Isotroper Halbkreisstrahler (Leistung I_j) auf Γ_j

Strahlungsfluss $\Gamma_j\to\Gamma_i$: $P_{ji}:=\dfrac{\alpha_{ij}}{\pi}I_j$

Aperturwinkel: $\alpha_{ij}=\gamma_{|i-j|}$, $1\le i,j\le n$

Gleichgewicht: $\underbrace{\sum_{i=1,i\neq j}^{n} P_{ji}}_{=I_j} - \sum_{i=1,i\neq j}^{n} P_{ij} = Q_j\ .$ (3.8.4) eq:piperа

$Q_j \mathrel{\hat{=}}$ Wärmezufluss/-abfluss durch Γ_j

$$Q_j:=\int_{\frac{2\pi}{n}(j-1)}^{\frac{2\pi}{n}j} q(\varphi)\,\mathrm{d}\varphi\ ,\quad q(\varphi):=\begin{cases}\text{lokale Heizleistung} & \text{, falls } \varphi\in\Gamma_H\ ,\\ \frac{1}{|\Gamma_K|}\int_{\Gamma_H} q(\varphi)\,\mathrm{d}\varphi & \text{, falls } \varphi\in\Gamma_K\ .\end{cases}$$

(3.8.4) eq:piperadbal ⇒ LGS: $I_j-\sum_{i=1,i\neq j}^{n}\frac{\alpha_{ij}}{\pi}I_i=Q_j\ ,\quad j=1,\dots,n\ .$

$n=8$:

$$\begin{pmatrix} 1 & -\gamma_1 & -\gamma_2 & -\gamma_3 & -\gamma_4 & -\gamma_3 & -\gamma_2 & -\gamma_1\\ -\gamma_1 & 1 & -\gamma_1 & -\gamma_2 & -\gamma_3 & -\gamma_4 & -\gamma_3 & -\gamma_2\\ -\gamma_2 & -\gamma_1 & 1 & -\gamma_1 & -\gamma_2 & -\gamma_3 & -\gamma_4 & -\gamma_3\\ -\gamma_3 & -\gamma_2 & -\gamma_1 & 1 & -\gamma_1 & -\gamma_2 & -\gamma_2 & -\gamma_4\\ -\gamma_4 & -\gamma_3 & -\gamma_2 & -\gamma_1 & 1 & -\gamma_1 & -\gamma_2 & -\gamma_3\\ -\gamma_3 & -\gamma_4 & -\gamma_3 & -\gamma_2 & -\gamma_1 & 1 & -\gamma_1 & -\gamma_2\\ -\gamma_2 & -\gamma_3 & -\gamma_4 & -\gamma_3 & -\gamma_2 & -\gamma_1 & 1 & -\gamma_1\\ -\gamma_1 & -\gamma_2 & -\gamma_3 & -\gamma_4 & -\gamma_3 & -\gamma_2 & -\gamma_1 & 1\end{pmatrix}\begin{pmatrix} I_1\\ I_2\\ I_3\\ I_4\\ I_5\\ I_6\\ I_7\\ I_8\end{pmatrix}=\begin{pmatrix} Q_1\\ Q_2\\ Q_3\\ Q_4\\ Q_5\\ Q_6\\ Q_7\\ Q_8\end{pmatrix}$$

$\blacktriangleright$ Symmetrische, zirkulante, singuläre, positiv semidefinite (→ Lemma 3.4.7) Koeffizientenmatrix lem:Gerschgorin

$\diamond$

Effiziente Algorithmen für zirkulante Matrizen:

Sei $\mathbf{C}\in\mathbb{C}^{n,n}$ zirkulante Matrix (→ Def. 3.8.4) def:zirmat $c_{ij}=u_{i-j}$, $(u_k)_{k\in\mathbb{Z}}$ = n-periodische Folge

$\mathbf{v}_k:=(\omega_n^{-jk})_{j=0}^{n-1}\in\mathbb{C}^n$, $k=0,\dots,n-1$ (= k. Spalte der Fourier-Matrix $\overline{\mathbf{F}}_n=n\mathbf{F}_n^{-1}$)

$$(\mathbf{C}\mathbf{v}_k)_j = \sum_{l=0}^{n-1} u_{j-l}\omega_n^{-lk} = \sum_{l=0}^{n-1} u_l \omega_n^{(l-j)k} = \omega_n^{-jk} \boxed{\sum_{l=0}^{n-1} u_l \omega_n^{lk}} = \lambda_k \omega_n^{-jk} .$$

unabhängig von j !

$\mathbf{v}_k$ ist Eigenvektor von $\mathbf{C}$ zum Eigenwert $\lambda_k = \sum_{l=0}^{n-1} u_l \omega_n^{lk} = (\mathbf{F}_n \mathbf{u})_k$.

Lemma 3.8.5 (Diagonalisierung zirkulanter Matrizen (→ Def. 3.8.4)). *Für jede zirkulante Matrix* $\mathbf{C} \in \mathbb{K}^{n,n}$, $c_{ij} = u_{i-j}$, $(u_k)_{k\in\mathbb{Z}}$ *n-periodische Folge, gilt*

$$\mathbf{C}\overline{\mathbf{F}}_n = \overline{\mathbf{F}}_n \operatorname{diag}(d_1, \ldots, d_n) \quad , \quad \mathbf{d} = \mathbf{F}_n (u_0, \ldots, u_{n-1})^T .$$

def:zirmat

Beachte: $\overline{\mathbf{F}}_n = n\mathbf{F}_n^{-1}$ ➤ $\mathbf{C} = \mathbf{F}_n^{-1} \operatorname{diag}(d_1, \ldots, d_n)\mathbf{F}_n$

Effizienter Algorithmus zur Berechnung der diskreten Faltung (→ Def. 3.8.3) zweier n-periodischer Folgen $(x_k)_{k\in\mathbb{Z}}$, $(y_k)_{k\in\mathbb{Z}}$:

def:conv

```
function z=conv(x,y)
n = length(y); z = zeros(n,1);
for i=1:n
  z(i)=dot(conj(x),...
      y([i:-1:1,n:-1:i+1]));
end
```

MATLAB-Code Naive Diskrete Faltung

Gedankenlos: Aufwand $O(n^2)$

```
function z=conv(x,y)
z = ifft(fft(x).*fft(y));
```

MATLAB-Code FFT-basierte diskrete Faltung

Professionell: Aufwand $O(n \log n)$

Effizienter FFT-basierter Algorithmus zur Invertierung einer zirkulanten Matrix

Bemerkung 150 (Primzahl-FFT).

Aus dem MATLAB-Manual:

When n is a prime number, the FFTW library first decomposes an n-point problem into three $(n-1)$-point problems using Rader's algorithm [37]. It then uses the Cooley-Tukey decomposition described above to compute the $(n-1)$-point DFTs.

RAD68

Ein Satz aus der Zahlentheorie:

$$\forall p \in \mathbb{N} \text{ prim} \quad \exists g \in \{1, \ldots, p-1\}: \quad \{g^k \mod p : k = 1, \ldots, p-1\} = \{1, \ldots, p-1\} .$$

Permutation $P_{p,g} : \{1, \ldots, p-1\} \mapsto \{1, \ldots, p-1\}$, $P_{p,g}(k) = g^k \mod p$.

Spiegelpermutation $P_k : \{1, \ldots, k\} \mapsto \{1, \ldots, k\}$, $P_k(i) = k - i + 1$.

Für Fouriermatrix $\mathbf{F} = (f_{ij})_{i,j=1}^p$: $P_{p-1}P_{p,g}(f_{ij})_{i,j=2}^p P_{p,g}^T$ ist zirkulant

Beispiel für $p = 13$: $g = 2$, Permutation: $(2\ 4\ 8\ 3\ 6\ 12\ 11\ 9\ 5\ 10\ 7\ 1)$.

$\mathbf{F}_{13} \longrightarrow$

ω^0	ω^0	ω^0	ω^0	ω^0	ω^0	ω^0	ω^0	ω^0	ω^0	ω^0	ω^0
ω^0	ω^2	ω^4	ω^8	ω^3	ω^6	ω^{12}	ω^{11}	ω^9	ω^5	ω^{10}	ω^7	ω^1
ω^0	ω^1	ω^2	ω^4	ω^8	ω^3	ω^6	ω^{12}	ω^{11}	ω^9	ω^5	ω^{10}	ω^7
ω^0	ω^7	ω^1	ω^2	ω^4	ω^8	ω^3	ω^6	ω^{12}	ω^{11}	ω^9	ω^5	ω^{10}
ω^0	ω^{10}	ω^7	ω^1	ω^2	ω^4	ω^8	ω^3	ω^6	ω^{12}	ω^{11}	ω^9	ω^5
ω^0	ω^5	ω^{10}	ω^7	ω^1	ω^2	ω^4	ω^8	ω^3	ω^6	ω^{12}	ω^{11}	ω^9
ω^0	ω^9	ω^5	ω^{10}	ω^7	ω^1	ω^2	ω^4	ω^8	ω^3	ω^6	ω^{12}	ω^{11}
ω^0	ω^{11}	ω^9	ω^5	ω^{10}	ω^7	ω^1	ω^2	ω^4	ω^8	ω^3	ω^6	ω^{12}
ω^0	ω^{12}	ω^{11}	ω^9	ω^5	ω^{10}	ω^7	ω^1	ω^2	ω^4	ω^8	ω^3	ω^6
ω^0	ω^6	ω^{12}	ω^{11}	ω^9	ω^5	ω^{10}	ω^7	ω^1	ω^2	ω^4	ω^8	ω^3
ω^0	ω^3	ω^6	ω^{12}	ω^{11}	ω^9	ω^5	ω^{10}	ω^7	ω^1	ω^2	ω^4	ω^8
ω^0	ω^8	ω^3	ω^6	ω^{12}	ω^{11}	ω^9	ω^5	ω^{10}	ω^7	ω^1	ω^2	ω^4
ω^0	ω^4	ω^8	ω^3	ω^6	ω^{12}	ω^{11}	ω^9	ω^5	ω^{10}	ω^7	ω^1	ω^2

△

Übung 3.9. Gegeben sei die Matrix $\mathbf{A} = (a_{lm})_{l,m=0}^{n-1} \in \mathbb{R}^{n,n}$, $n \in \mathbb{N}$. Zu berechnen sind die Werte

$$f_k := \sum_{l=0}^{k} a_{l,k-l}\, x_l y_{k-l} , \quad k = 0, \ldots, n-1 , \tag{3.9.1}$$

eq:fk

für beliebige Vektoren $\mathbf{x} := (x_0, \ldots, x_{n-1})^T \in \mathbb{R}^n$, $\mathbf{y} := (y_0, \ldots, y_{n-1})^T \in \mathbb{R}^n$.

(i) Schreibe eine MATLAB-Funktion `function f = compf(A,x,y)`, die die f_k gemäss (3.9.1) berechnet. eq:fk

(ii) Was ist die asymptotische Komplexität von `compf` bzgl. n?

(iii) Entwickle eine *effiziente* Implementierung der Funktion `function f = compflr(p,q,x,y)` die `compf` für $\mathbf{A} = \mathbf{p}\mathbf{q}^T$, $\mathbf{p},\mathbf{q} \in \mathbb{R}^n$, realisiert (Hinweis: Beispiel 148) ex:polmul

(iv) Was ist die asymptotische Komplexität von `compflr` bzgl. n?

3.9.1 Toeplitz-Matrizen

TOEPLITZMAT

Matrizen mit konstanten Diagonalen (Spezialfall: Zirkulante Matrizen → Def. 3.8.4) def:zirmat

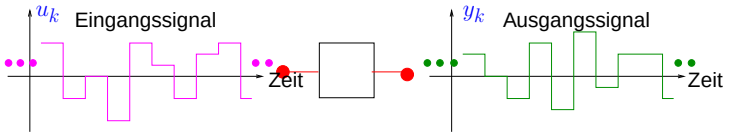
Definition 3.9.1 (Toeplitz-Matrix). $\mathbf{T} = (t_{ij})_{i,j=1}^{n} \in \mathbb{K}^{m,n}$ *heisst Toeplitz-Matrix, wenn ein Vektor* $\mathbf{u} = (u_{-m+1},\dots,u_{n-1}) \in \mathbb{K}^{m+n-1}$ *existiert, so dass* $t_{ij} = u_{j-i}$, $1 \le i \le m$, $1 \le j \le n$.

$$\mathbf{T} = \begin{pmatrix} u_0 & u_1 & \cdots & & \cdots & u_{n-1} \\ u_{-1} & u_0 & u_1 & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & \ddots & \ddots & u_1 \\ u_{1-m} & \cdots & & \cdots & u_{-1} & u_0 \end{pmatrix}$$

Beispiel 151 (Lineare zeitinvariante Systeme).

- $(u_k)_{k\in\mathbb{Z}}$ m-periodisches zeitdiskretes Signal = input
- $(y_k)_{k\in\mathbb{Z}}$ m-periodisches Ausgabesignal eines zeitinvarianten linearen Übertragungssystems

$$\exists \mathbf{x} = (x_1,\dots,x_n)^T \in \mathbb{R}^n,\quad n \le m: \quad y_k = \sum_{j=1}^{n} x_j u_{k-j+1}\,.$$



Identifikationsproblem = Gesucht: $\mathbf{x} \in \mathbb{R}^n$ so, dass

$$\|\mathbf{A}\mathbf{x}-\mathbf{y}\|_2 = \left\| \begin{pmatrix} u_0 & u_{-1} & \cdots & & \cdots & u_{1-n} \\ u_1 & u_0 & u_{-1} & & & \vdots \\ \vdots & u_1 & u_0 & \ddots & & \\ & & \ddots & \ddots & & \vdots \\ \vdots & & & \ddots & \ddots & u_{-1} \\ u_{n-1} & & & & u_1 & u_0 \\ u_n & u_{n-1} & & & & u_1 \\ \vdots & & & & & \vdots \\ u_{m-1} & \cdots & & & \cdots & u_{m-n} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ \\ \vdots \\ x_n \end{pmatrix} - \begin{pmatrix} y_0 \\ \vdots \\ \\ \\ \vdots \\ y_{m-1} \end{pmatrix} \right\|_2 \to \min\,.$$

➢ Lineares Ausgleichsproblem, → Abschnitt 3.5, mit Toeplitz-Matrix $\mathbf{A}$: $a_{ij} = u_{i-j}$. sec:numer-line-ausgl

Matrix der Normalengleichungen (→ Abschnitt 3.5.2) sec:normalengleichungen

$$\mathbf{M} := \mathbf{A}^H\mathbf{A} \quad , \quad m_{ij} = \sum_{k=1}^{m} u_{k-i}u_{k-j} = z_{i-j} \text{ wegen Periodizität der Folge } (u_k)_{k\in\mathbb{Z}}\,.$$

➢ $\mathbf{M} \in \mathbb{R}^{n,n}$ ist ebenfalls Toeplitz-Matrix & symmetrisch positiv semi-definit.

Beispiel 152 (Lineare Regression bei stationären Markov-Ketten).

Folge von Zufallsvariablen: $(y_k)_{k\in\mathbb{Z}}$ = Markov-Kette mit zeitunabhängiger Korrelation

Erwartungswert $E(y_{i-j}y_{i-k}) = u_{k-j} \quad \forall i,j,k \in \mathbb{Z}\,,\quad u_i = u_{-i}\,.$

Modell: Linearer Zusammenhang

$$\exists \mathbf{x} = (x_1,\dots,x_n)^T \in \mathbb{R}^n: \quad y_k = \sum_{j=1}^{n} x_j y_{k-j} \quad \forall k \in \mathbb{Z}\,.$$

mit zu schätzenden Koeffizienten x_j, $j = 1,\dots,n$: Für festes $i \in \mathbb{Z}$

$$\mathbf{x} = \operatorname{argmax}_{\mathbf{x}\in\mathbb{R}^n} E\left|y_i - \sum_{j=1}^{n} x_j y_{i-j}\right|^2 \quad \blacktriangleright \quad E|y_i|^2 - 2\sum_{j=1}^{n} x_j u_k + \sum_{k,j=1}^{n} x_k x_j u_{k-j} \to \max\,.$$

$$\blacktriangleright \quad \mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{b}^T\mathbf{x} \to \max \quad \text{mit} \quad \mathbf{b} = (u_k)_{k=1}^{n}\,,\quad \mathbf{A} = (u_{i-j})_{i,j=1}^{n}\,.$$

Kovarianzmatrix $\mathbf{A}$ ist s.p.d. Toeplitz-Matrix, $\mathbf{x}$ Lösung der Yule-Walker-Gleichung $\mathbf{A}\mathbf{x} = \mathbf{b}$. ◇

3.9.1.1 Toeplitz-Matrix-Arithmetik

TOEPMATHARITH

Gegeben: Toeplitz-Matrix $\mathbf{T} = (u_{j-i}) \in \mathbb{K}^{m,n}$ mit erzeugendem Vektor $\mathbf{u} = (u_{-m+1},\dots,u_{n-1}) \in \mathbb{C}^{2n-1}$, Vektor $\mathbf{x} \in \mathbb{K}^n$

Aufgabe: Effiziente Realisierung des Matrix×Vektor-Produkts $\mathbf{Tx}$

Beobachtung: Folgende erweiterte Matrix ist zirkulant ! (Beispiel für $m = n$)

$$\mathbf{C} = \begin{pmatrix} \mathbf{T} & \mathbf{S} \\ \mathbf{S} & \mathbf{T} \end{pmatrix} = \left(\begin{array}{ccccc|ccccc} u_0 & u_1 & \cdots & \cdots & u_{n-1} & 0 & u_{1-n} & \cdots & \cdots & u_{-1} \\ u_{-1} & u_0 & u_1 & & \vdots & u_{n-1} & 0 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \ddots & \ddots & & \\ \vdots & & \ddots & \ddots & u_1 & \vdots & & & \ddots & u_{1-n} \\ u_{1-n} & \cdots & \cdots & u_{-1} & u_0 & u_1 & & & u_{n-1} & 0 \\ \hline 0 & u_{1-n} & \cdots & \cdots & u_{-1} & u_0 & u_1 & \cdots & \cdots & u_{n-1} \\ u_{n-1} & 0 & \ddots & & \vdots & u_{-1} & u_0 & u_1 & & \vdots \\ \vdots & \ddots & \ddots & & & \vdots & \ddots & \ddots & \ddots & \vdots \\ & & & \ddots & & \vdots & & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & u_{1-n} & \vdots & & & \ddots & u_1 \\ u_1 & & & u_{n-1} & 0 & u_{1-n} & \cdots & \cdots & u_{-1} & u_0 \end{array}\right)$$

Allgemein:

```
T = toeplitz(u(0:-1:1-m),u(0:n-1));
S = toeplitz([0,u(n-1:-1:n-m+1)],[0,u(1-m:1:-1)]);
```

$$\mathbf{C}\begin{pmatrix} \mathbf{x} \\ 0 \end{pmatrix} = \begin{pmatrix} \mathbf{Tx} \\ \mathbf{Sx} \end{pmatrix}$$

Rechenaufwand $O(n \log n)$ für Auswertung $\mathbf{Tx}$ (mit FFT, Abschn 3.8.1)

sec:zirkulante-matrizen

3.9.1.2 Der Levinson-Algorithmus

Gegeben: S.p.d. Toeplitz-Matrix $\mathbf{T} = (u_{j-i})_{i,j=1}^{n}$
mit erzeugendem Vektor $\mathbf{u} = (u_{-n+1}, \ldots, u_{n-1}) \in \mathbb{C}^{2n-1}$, $u_{-k} = u_k$, o.B.d.A. $u_0 = 1$,
Vektor $\mathbf{b} \in \mathbb{K}^n$

Gesucht: Effizientes Verfahren zur Lösung des LGS $\mathbf{Tx} = \mathbf{b}$

Induktive Konstruktion des Algorithmus:

Definiere: $\mathbf{T}_k := (u_{j-i})_{i,j=1}^{k} \in \mathbb{K}^{k,k}$ (linkes oberes Dreieck, ebenfalls s.p.d. & Toeplitz),
$\mathbf{x}^k \in \mathbb{K}^k$: $\mathbf{T}_k\mathbf{x}^k = (b_1, \ldots, b_k)^T$,
$\mathbf{u}^k := (u_1, \ldots, u_k)^T$

$$\mathbf{T}_{k+1}\mathbf{x}^{k+1} = \left(\begin{array}{ccc|c} & \mathbf{T}_k & & \begin{matrix} u_k \\ \vdots \\ u_1 \end{matrix} \\ \hline u_k & \cdots & u_1 & 1 \end{array}\right)\begin{pmatrix} \widetilde{\mathbf{x}}^{k+1} \\ \hline x_{k+1}^{k+1} \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ \hline b_k \\ \hline b_{k+1} \end{pmatrix}$$

Spiegelungspermutation: $P_k : \{1, \ldots, k\} \mapsto \{1, \ldots, k\}$, $P_k(i) := k - i + 1$

$$\widetilde{\mathbf{x}}_{k+1} = \mathbf{T}_k^{-1}(\widetilde{\mathbf{b}}_{k+1} - x_{k+1}^{k+1}P_k\mathbf{u}^k) = \mathbf{x}^k - x_{k+1}^{k+1}\mathbf{T}_k^{-1}P_k\mathbf{u}^k ,$$
$$x_{k+1}^{k+1} = b_{k+1} - P_k\mathbf{u}^k \cdot \widetilde{\mathbf{x}}^{k+1} = b_{k+1} - P_k \cdot \mathbf{x}^k + x_{k+1}^{k+1}P_k \cdot \mathbf{T}_k^{-1}P_k\mathbf{u}^k .$$

Mit Hilfsvektoren $\mathbf{y}^k := \mathbf{T}_k^{-1}P_k\mathbf{u}^k$

$$\mathbf{x}^{k+1} = \begin{pmatrix} \widetilde{\mathbf{x}}^{k+1} \\ x_{k+1}^{k+1} \end{pmatrix} \quad \text{mit} \quad \begin{matrix} x_{k+1}^{k+1} = (b_{k+1} - P_k\mathbf{u}^k)/\sigma_k \\ \widetilde{\mathbf{x}}^{k+1} = \mathbf{x}^k - x_{k+1}^{k+1}\mathbf{y}^k \end{matrix} \quad , \quad \sigma_k := 1 - P_k\mathbf{u}^k \cdot \mathbf{y}^k .$$

MATLAB-Implementierung
(rekursiv, u_{n+1} nicht verwendet !)

Lineare Rekursion:
Rechenaufwand $O(n-k)$ auf k. Rekursionsstufe, $k = 0, \ldots, n-1$

➢ Gesamtaufwand $O(n^2)$

MATLAB-Code Levinson-Algorithmus

```
function [x,y] = levinson(u,b)
k = length(u)-1;
if (k == 0)
  x=b(1); y = u(1); return;
end
[xk,yk] = levinson(u(1:k),b(1:k));
sigma = 1-dot(u(1:k),yk);
t= (b(k+1)-dot(u(k:-1:1),xk))/sigma;
x= [ xk-t*yk(k:-1:1);t];
s= (u(k+1)-dot(u(k:-1:1),yk))/sigma;
y= [yk-s*yk(k:-1:1); s];
```

Bemerkung 153 („Superschnelle" Toeplitz-Löser). FFT-basierte Algorithmen zur Lösung von $\mathbf{Tx} = \mathbf{b}$ benötigen Rechenaufwand $O(n \log^3 n)$ [42] !

STE03a

△

Übung 3.10. Gegeben eine grosse ganze Zahl $N \gg 1$ und zwei Polynome

$$p(z) = \sum_{j=0}^{2N} \alpha_j z^j , \quad q(z) = \sum_{j=0}^{N} \beta_j z^j , \quad \alpha_j, \beta_j \in \mathbb{C} ,$$

die Polynomdivision mit fragt nach einem Polynom s vom Grad N und einem Polynom r vom Grad $N-1$, so dass

$$p(z) = q(z)s(z) + r(z) .$$

Implementiere eine effiziente MATLAB-Funktion `[s,r] = polydiv(a,b)`, der die Koeffizienten α_j und β_j übergeben werden und die die Koeffizienten von s und r berechnet. Dabei ist zu berücksichtigen, dass in MATLAB ein Polynom durch den Vektor seiner Koeffizienten zu *absteigenden* Potenzen kodiert wird, so etwa das Polynom q durch $(\alpha_{2N}, \alpha_{2N-1}, \ldots, \alpha_0) \in \mathbb{C}^{2N+1}$.

Gib die Asymptotik des Rechenaufwandes in N an.

Übung 3.11. Für stetige Funktionen $f, u, h \in C^0([0,1])$ betrachte die Integralleichung

$$\int_0^t f(t-\tau)u(\tau)\,\mathrm{d}\tau = h(t) , \quad 0 \le t \le \tau . \qquad (3.11.1)$$

eq:integ

Die Gleichung wird auf folgende Art *diskretisiert*: Man wählt ein äquidistantes Gitter $\{t_j := j/N\}_{j=0,\ldots,N}$, $N \in \mathbb{N}$, betrachtet die Gleichung nur an den Gitterpunkten t_j, $j = 1, \ldots, N$

(Kollokation), und approximiert die Integrale durch Mittelpunktsquadratur (die einfachste Gauss-Quadraturformel) auf den Intervallen $[t_{j-1}, t_j]$, $j = 1, \ldots, N$.

So gelangt man zu der diskretisierten Integralgleichung

$$\frac{1}{N}\sum_{j=1}^{n} f(t_n - m_j)u(m_j) = h(t_n)\ , \quad n = 1, \ldots, N\ , \tag{3.11.2}$$

wobei $m_j := \frac{1}{2}(t_j + t_{j-1})$. Implementiere eine effiziente MATLAB-Funktion `u = inteqsolve(N,h,f)`, die für Funktionshandles `h`, `f` die Werte $u(m_j)$, $j = 1, \ldots, N$, aus (3.11.2) berechnet.

4 Interpolation und Approximation

[File: chapter-interpolation-und-approximation.tex, SVN: chapter-interpolation-und-approximation.tex 1299 2007-02-01 09:09:04Z kalai]

Allgemeine „Funktion" $f : D \subset \mathbb{R} \mapsto \mathbb{K}$ = Konzept der Analysis (unendlich viel Information)

Herausforderung: „Computergerechte" Repräsentation von Funktionen

Idee: Parametrisierung

➔ endlich viele Parameter $\alpha_1, \ldots, \alpha_n$, $n \in \mathbb{N}$, charakterisieren Funktion.

(→ Bsp. 59)

Spezialfall: Darstellung durch (endliche) Linearkombination von Basisfunktionen $b_j : D \subset \mathbb{R} \mapsto \mathbb{K}$, $j = 1, \ldots, n$:

$$f = \sum_{j=1}^{n} \alpha_j b_j \quad , \quad \alpha_j \in \mathbb{K}\ .$$

► $f \in$ endlichdimensionaler Funktionenraum $V_n := \mathrm{Span}\,\{b_1, \ldots, b_n\}$

4.1 Polynomiale Techniken

[File: section-polynomiale-techniken.tex, SVN: section-polynomiale-techniken.tex 1299 2007-02-01 09:09:04Z kalai]

Notation: Vektorraum der Polynome vom Grad $\leq k$, $k \in \mathbb{N}$:

$$\mathcal{P}_k := \{t \mapsto \alpha_k t^k + \alpha_{k-1}t^{k-1} + \cdots + \alpha_1 t + \alpha_0\,,\ \alpha_j \in \mathbb{K}\}\ . \tag{4.1.1}$$

► $\dim \mathcal{P}_k = k+1$ und $\mathcal{P}_k \subset C^\infty(\mathbb{R})$

Warum Polynome ? → Einfach auszuwerten, zu integrieren, zu differenzieren

→ Vektorraumstruktur & Algebra

↔ Analysis: Taylorpolynome & Potenzreihen

Bemerkung 154. $t \mapsto \alpha_k t^k + \alpha_{k-1}t^{k-1} + \cdots + \alpha_0$ = monomiale Darstellung eines Polynoms

MATLAB: $\alpha_k t^k + \alpha_{k-1}t^{k-1} + \cdots + \alpha_0$ ➨ Vektor $(\alpha_k, \alpha_{k-1}, \ldots, \alpha_0)$ (Anordnung !)

• Auswertung eines Polynoms in monomialer Darstellung (→ Bem. 154): Hornerschema

$$p(t) = (x \cdots x(x(\alpha_n x + \alpha_{n-1}) + \alpha_{n-2}) + \cdots + \alpha_1) + \alpha_0\ .$$

```
function y = polyval(p,x)
y = p(1); for i=2:length(p), y = x*y+p(i); end
```

➤ Rechenaufwand $O(n)$

Benutze: MATLAB „built-in"-Funktion `polyval(x,p);`

4.1.1 Polynominterpolation

Ziel: Rekonstruktion von Funktionen aus Wertepaaren, vgl. „Fitten" Bsp. 120

LAGRANGEINTP

Allgemeine *Lagrange-Polynominterpolationsaufgabe*

Zu (einfachen) Knoten (engl. *nodes*) $t_0, \ldots, t_n$, $n \in \mathbb{N}$, $-\infty < t_0 < t_1 < \cdots < t_n < \infty$ und Werten $y_0, \ldots, y_n \in \mathbb{K}$ bestimme $p \in \mathcal{P}_n$ so, dass

$$p(t_j) = y_j \quad \text{für} \quad j = 0, \ldots, n \,. \tag{4.1.2}$$

intp:lagr

Allgemeine *Polynominterpolationsaufgabe*

Zu (evtl. mehrfachen) Knoten $t_0, \ldots, t_n$, $n \in \mathbb{N}$, $-\infty < t_0 \le t_1 \le \cdots \le t_n < \infty$ und Werten $y_0, \ldots, y_n \in \mathbb{K}$ bestimme $p \in \mathcal{P}_n$ so, dass

$$\frac{\mathrm{d}}{\mathrm{d}t^k} p(t_j) = y_j \quad \text{für} \quad k = 0, \ldots, l_j \quad \text{und} \quad j = 0, \ldots, n \,, \tag{4.1.3}$$

mit $l_j := \max\{j - i : t_j = t_i, i = 0, \ldots, n\}$ (Vielfachheit des Knotens t_j)

intp:state

Doppelte Knoten $t_0 = t_1 < t_2 = t_3 < \cdots < t_{n-1} = t_n$ ▷ $p(t_{2j}) = y_{2j}, p'(t_{2j}) = y_{2j+1}$

HERMITINTP

Hermite-Interpolation

4.1

p. 369

4.1.1.1 Theorie und Kondition

Für $t_0 < t_1 < \cdots < t_n$ (→ Lagrange-Interpolation) betrachte

LAGRANGEPOL

Lagrange-Polynome $L_i(t) := \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{t - t_j}{t_i - t_j}$, $i = 0, \ldots, n$. (4.1.4)

eq:lagrar

► $L_i \in \mathcal{P}_n$ und $\boxed{L_i(t_j) = \delta_{ij}}$

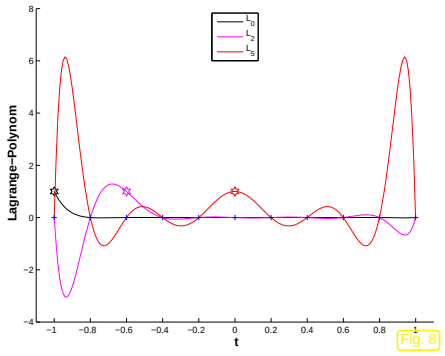
Beispiel 155 (Lagrange-Polynome).

Lagrange-Polynome für äquidistante Knoten

$$\mathcal{T} := \left\{ -5 + \tfrac{10}{n} j \right\}_{j=0}^{n} \,,$$

$$y_j = \frac{1}{1 + t_j^2} \,, \quad j = 0, \ldots \,.$$

Plot $n = 10$ ➢ ◇



lagrplot

Fig. 8

4.1

p. 370

► Für Lagrange-Interpolation: Darstellung $p(t) = \sum_{i=0}^{n} y_i L_i(t)$. (4.1.5)

eq:lagrre

Theorem 4.1.1 (Existenz & Eindeutigkeit des Interpolationspolynoms). *Die Polynominterpolationsaufgabe* (4.1.3) *hat eine eindeutige Lösung* $p \in \mathcal{P}_n$.

ntp

intp:state

Beweis. für Lagrange-Interpolation:

$$p(t_j) = y_j \iff \sum_{i=0}^{n} a_i t_j^i = y_j, \; j = 0, \ldots, n$$
$$\iff (n+1) \times (n+1) \text{ lineares Gleichungssystem}$$

((4.1.5) ⇒ Existenz) ⇒ Eindeutigkeit des Interpolationspolynoms □

eq:lagrrep

➤ Polynominterpolation zu Knoten $\mathcal{T} := \{t_j\}_{j=0}^{n}$ = lineare Abbildung

$$\mathsf{I}_{\mathcal{T}} : \mathbb{K}^{n+1} \mapsto \mathcal{P}_n \quad , \quad (y_0, \ldots, y_n)^T \mapsto \text{Interpolationspolynom } p \,. \tag{4.1.6}$$

intp:polint

4.1

p. 371

Definition 4.1.2 (Verallgemeinerte Lagrange-Polynome).

Die verallgemeinerten Lagrange-Polynome zu einer Knotenmenge $\mathcal{T} = \{t_j\}_{j=0}^{n} \subset \mathbb{R}$ *sind* $L_i := \mathsf{I}_{\mathcal{T}}^{-1}(\mathbf{e}_{i+1})$, $i = 0, \ldots, n$, *wobei* $\mathbf{e}_i = (0, \ldots, 0, 1, 0, \ldots, 0)^T \in \mathbb{R}^{n+1} \hat{=} i$. *Einheitsvektor.*

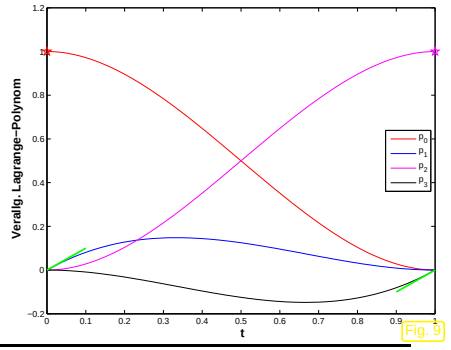
Beispiel 156. (Verallgemeinerte Lagrange-Polynome für Hermite-Interpolation)

Doppelte Knoten $t_0 = 0, t_1 = 0, t_2 = 0, t_3 = 0$

➤ $n = 3$

(Kubische Hermite-Interpolation)

◇



hermlagr

Fig. 9

4.1

p. 372

Erforderlich für Konditionsuntersuchung: Normen auf $C^\infty(I)$, $I \subset \mathbb{R}$

NINF
Supremumsnorm $\|f\|_{L^\infty(I)} := \sup\{|f(t)|: t \in I\}$, (4.1.7) Ninf

NLTWO
L^2-Norm $\|f\|^2_{L^2(I)} := \int_I |f(t)|^2 \,\mathrm{d}t$, (4.1.8) NLtwo

NLONE
L^1-Norm $\|f\|_{L^1(I)} := \int_I |f(t)| \,\mathrm{d}t$. (4.1.9) NLone

Lemma 4.1.3 (Absolute Kondition der Polynominterpolation). *Für eine Knotenmenge $\mathcal{T} \subset \mathbb{R}$ mit zugehörigen verallgemeinerten Lagrange-Polynomen L_i, $i = 0, \ldots, n$, und festes $I \subset \mathbb{R}$ gilt*

$$\|\mathsf{I}_\mathcal{T}\|_{\infty\to\infty} := \sup_{\mathbf{y}\in\mathbb{K}^{n+1}\setminus\{0\}} \frac{\|\mathsf{I}_\mathcal{T}(\mathbf{y})\|_{L^\infty(I)}}{\|\mathbf{y}\|_\infty} = \left\|\sum_{i=0}^n |L_i|\right\|_{L^\infty(I)} , \quad (4.1.10)$$ eq:IPN1

$$\|\mathsf{I}_\mathcal{T}\|_{2\to 2} := \sup_{\mathbf{y}\in\mathbb{K}^{n+1}\setminus\{0\}} \frac{\|\mathsf{I}_\mathcal{T}(\mathbf{y})\|_{L^2(I)}}{\|\mathbf{y}\|_2} \le \left(\sum_{i=0}^n \|L_i\|^2_{L^2(I)}\right)^{\frac{1}{2}} . \quad (4.1.11)$$ eq:IPN2

Beweis. (für L^∞-Norm) Mit $\triangle$-Ungleichung

$$\|\mathsf{I}_\mathcal{T}(\mathbf{y})\|_{L^\infty(I)} = \left\|\sum_{j=0}^n y_j L_j\right\|_{L^\infty(I)} \le \sup_{t\in I} \sum_{j=0}^n |y_j||L_j(t)| ,$$

Gleichheit in (4.1.10) für $\mathbf{y} := (\operatorname{sgn}(L_j(t^*)))_{j=0}^n$, $t^* := \operatorname{argmax}_{t\in I} \sum_{i=0}^n |L_i(t)|$. □

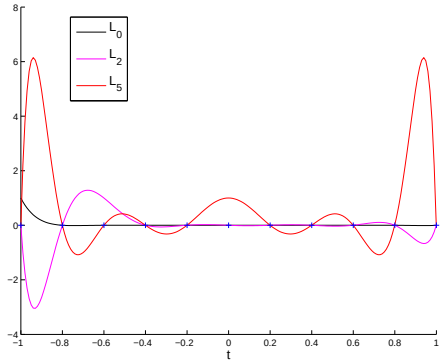
Beweis. (für L^2-Norm) Mit $\triangle$-Ungleichung und Cauchy-Schwarz-Ungleichung

$$\|\mathsf{I}_\mathcal{T}(\mathbf{y})\|_{L^2(I)} \le \sum_{j=0}^n |y_j| \, \|L_j\|_{L^2(I)} \le \left(\sum_{j=0}^n |y_j|^2\right)^{\frac{1}{2}} \left(\sum_{j=0}^n \|L_j\|^2_{L^2(I)}\right)^{\frac{1}{2}} . \quad \square$$

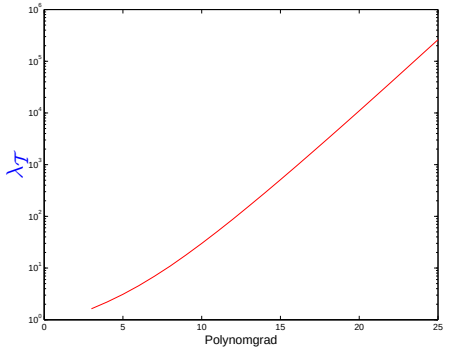
Terminologie: LEBESGUE
Lebesgue-Konstante zu $\mathcal{T}$: $\lambda_\mathcal{T} := \left\|\sum_{i=0}^n |L_i|\right\|_{L^\infty(I)}$

Beispiel 157 (Berechnung von Lebesgue-Konstanten).

$I = [-1, 1]$, $\mathcal{T} = \{-1 + \frac{k}{2n}\}_{k=0}^n$ (äquidistante Knoten)



Lagrange-Polynome für $n = 10$



Lebesgue-Konstante für äquidistante Knoten

Asymptotische Abschätzung (Stirlingsche Formel) : für $n = 2m$, $I = [-1, 1]$, $\mathcal{T} = \{-1 + \frac{2k}{n}\}_{k=0}^n$

$$|L_m(1 - \tfrac{1}{n})| = \frac{\frac{1}{n} \cdot \frac{1}{n} \cdot \frac{3}{n} \cdot \ldots \cdot \frac{n-3}{n} \cdot \frac{n+1}{n} \cdot \ldots \cdot \frac{2n-1}{n}}{\left(\frac{2}{n} \cdot \frac{4}{n} \cdot \ldots \cdot \frac{n-2}{n} \cdot 1\right)^2} = \frac{(2n)!}{(n-1)2^{2n}((n/2)!)^2 n!} \sim \frac{2^{n+1/2}}{\pi(n-1)n^{3/2}}$$

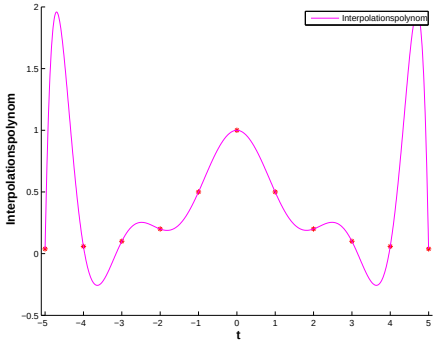
Theorie [8]: COR92 *für äquidistante Knoten* $\lambda_\mathcal{T} \ge Ce^{n/2}$ für $C > 0$ unabhängig von n. ◇

Beispiel 158 (Oszillatorische Interpolationspolynome).

Polynominterpolation mit äquidistanten Knoten:

$$\mathcal{T} := \left\{-5 + \tfrac{10}{n} j\right\}_{j=0}^n ,$$
$$y_j = \frac{1}{1 + t_j^2} , \quad j = 0, \ldots .$$

Plot $n = 10$ ➢



Gefahr: Starke Oszillationen von Interpolationspolynomen hohen Grades auf äquidistanten Knoten

◇

4.1.1.2 Algorithmen

• Vielfache Auswertung eines Interpolationspolynoms zu Knotenmenge $\mathcal{T} = \{t_j\}_{j=0}^n$, Stützwerten $y_0, \ldots, y_n$:

$$p(t) = \sum_{i=0}^{n} \prod_{\substack{j=0 \\ j\neq i}}^{n} \frac{t-t_j}{t_i-t_j}\, y_i = \sum_{i=0}^{n} \lambda_i \prod_{\substack{j=0 \\ j\neq i}}^{n} (t-t_j)\, y_i = \prod_{j=0}^{n}(t-t_j) \cdot \sum_{i=0}^{n} \frac{\lambda_i}{t-t_i}\, y_i \,.$$

mit $\lambda_i = \dfrac{1}{(t_i-t_0)\cdots(t_i-t_{i-1})(t_i-t_{i+1})\cdots(t_i-t_n)}$, $i = 0, \ldots, n$.

Aus $1 = \prod_{j=0}^{n}(t-t_j) \sum_{i=0}^{n} \frac{\lambda_i}{t-t_i} \quad \Rightarrow \quad \prod_{j=0}^{n}(t-t_j) = \frac{1}{\sum_{i=0}^{n} \frac{\lambda_i}{t-t_i}}$

Baryzentrische Interpolationsformel $p(t) = \dfrac{\sum_{i=0}^{n} \frac{\lambda_i}{t-t_i}\, y_i}{\sum_{i=0}^{n} \frac{\lambda_i}{t-t_i}}$.

Falls λ_i verfügbar: Rechenaufwand(Auswertung von $p(t)$) = $O(n)$:

MATLAB-CODE : Auswertung eines Interpolationspolynomes

```
function p = intpolyval(t,y,x)
n = length(t); N = length(x);
for k = 1:n, lambda(k) = 1 / prod(t(k) - t([1:k-1,k+1:n])); end;
for i = 1:N
  z = (x(i)-t); j = find(z == 0);
  if (~isempty(j)), p(i) = y(j);
  else
    mu = lambda./z;
    p(i) = dot(mu,y)/sum(mu);
  end
end
```

Vermeide Division durch 0 !

• Auswertung eines Interpolationspolynoms an wenigen Stellen: Aitken-Neville-Schema

Gegeben: Knotenmenge $\mathcal{T} := \{t_j\}_{j=0}^n \subset \mathbb{R}$ paarweise verschieden, $t_i \neq t_j$ für $i \neq j$, Stützwerte $y_0, \ldots, y_n$, Auswertungsstelle $x \in \mathbb{R}$

Für $\{i_0, \ldots, i_m\} \subset \{0, \ldots, n\}$, $0 \leq m \leq n$:

$p_{i_0,\ldots,i_m}$ = Interpolationspolynom durch $(t_{i_0}, y_{i_0}), \ldots, (t_{i_m}, y_{i_m})$:

$$\blacktriangleright \quad p_i \equiv y_i \,, \quad i = 0, \ldots, n \quad , \quad p_{i_0,\ldots,i_m}(t) = \frac{(t-t_{i_0})p_{i_1,\ldots,i_m} - (t-t_{i_m})p_{i_0,\ldots,i_{m-1}}}{t_{i_m} - t_{i_0}} \,. \quad (4.1.12)$$ eq:ipolrec

Schema:

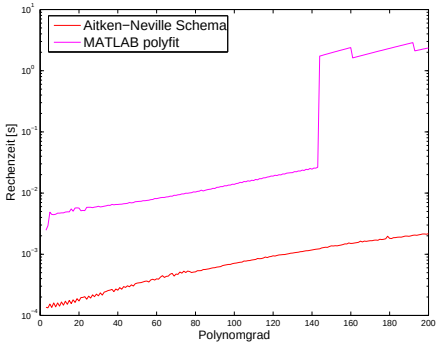
	n =0	1	2	3
t_0	$y_0 =: p_0(x)$	$p_{01}(x)$	$p_{012}(x)$	$p_{0123}(x)$
t_1	$y_1 =: p_1(x)$	$p_{12}(x)$	$p_{123}(x)$	
t_2	$y_2 =: p_2(x)$	$p_{23}(x)$		
t_3	$y_3 =: p_3(x)$			

```
function v = ANipoleval(t,y,x)
for i=1:length(y)
  for k=i-1:-1:1
    y(k) = y(k+1)+(y(k+1)-y(k))*...
           (x-t(i))/(t(i)-t(k));
  end
end
v = y(1);
```


Äquivalente MATLAB-Funktion:
(mit deutlich höherem Rechenaufwand !)

```
function v=ipoleval(t,y,x)
p = polyfit(t,y,length(y)-1);
v=polyval(p,x);
```

`tic-toc`-Rechenzeitmessung, 100 runs ➢



Wichtige Anwendung: *Extrapolation nach Null*

Problem: Gesucht ist $f(0)$ mit vorgeschriebener Genauigkeit, aber implementierte Funktion `y=f(h)` ist für $|t| \ll 1$ nicht stabil auszuwerten.

Bekannt: Existenz einer asymptotischen Entwicklung in h^2

$$f(h) = f(0) + A_1 h^2 + A_2 h^4 + \cdots + A_n h^{2n} + R(h) \quad , \quad A_k \in \mathbb{K} \,,$$

mit Restgliedabschätzung $|R(h)| = O(h^{2n+2})$ für $h \to 0$.

Idee:

① Auswertung $f(t_i)$ für verschiedene t_i, $i = 0, \ldots, n$, $|t_i| > 0$.

② $f(0) \approx p(0)$ mit Interpolationspolynom $p \in \mathcal{P}_n$, $p(t_i) = f(t_i)$.

Beispiel 159 (Numerische Differentiation durch Extrapolation, → Bsp. ex:numdiff 27).

Für $2(n+1)$-mal stetig differenzierbare Funktion $f : D \subset \mathbb{R} \mapsto \mathbb{R}$, $x \in D$ (Taylorsumme mit Lagrange-Restglied)

$$T(h) := \frac{f(x+h) - f(x-h)}{2h} \sim f'(x) + \sum_{k=1}^{n} \frac{1}{(2k)!} \frac{\mathrm{d}^{2k} f}{\mathrm{d}x^{2k}}(x)\, h^{2k} + \frac{1}{(2n+2)!} f^{(2n+2)}(\xi(x)) \,.$$

MATLAB-Code (f = handle auf Funktion):

MATLAB-CODE : Numerische Differentiation durch Interpolation

```
function d = diffex(f,x,h0,tol)
h = h0;
y(1) = (f(x+h0)-f(x-h0))/(2*h0);
for i=2:10
  h(i) = h(i-1)/2;
  y(i) = (f(x+h(i))-f(x-h(i)))/h(i-1);
  for k=i-1:-1:1
    y(k) = y(k+1)-(y(k+1)-y(k))*h(i)/(h(i)-h(k));
  end
  if (abs(y(2)-y(1)) < tol*abs(y(1))), break; end
end
d = y(1);
```

A posteriori Fehlerschätzung

`diffex(@atan,1.1,0.5)`

Grad	Relativer Fehler
0	0.04262829970946
1	0.02044767428982
2	0.00051308519253
3	0.00004087236665
4	0.00000048930018
5	0.00000000746031
6	0.00000000001224

`diffex(@sqrt,1.1,0.5)`

Grad	Relativer Fehler
0	0.02849215135713
1	0.01527790811946
2	0.00061205284652
3	0.00004936258481
4	0.00000067201034
5	0.00000001253250
6	0.00000000004816
7	0.00000000000021

`diffex(@exp,1.1,0.5)`

h	Relativer Fehler
0	0.04219061098749
1	0.02129207652215
2	0.00011487434095
3	0.00000825582406
4	0.00000000589624
5	0.00000000009546
6	0.00000000000002

Vorteile: Garantierte Genauigkeit ➢ Effizienz

Zum Vergleich: Numerische Differentiation mit finiten (Vorwärts-) Differenzen ➢ Auslöschung
(→ Bsp. ex:numdiff 27)

```
x=1.1; h=2.^[-1:-5:-36];
atan_err =  atan_err=abs(dirnumdiff(@atan,x,h)-1/(1+x^2))*(1+x^2);
sqrt_err=abs(dirnumdiff(@sqrt,x,h)-1/(2*sqrt(x)))*(2*sqrt(x));
exp_err=abs(dirnumdiff(@exp,x,h)-exp(x))/exp(x);

function[df]=dirnumdiff(f,x,h)
df=(f(x+h)-f(x))./h;
end
```

$f(x) = \arctan(x)$

h	Relativer Fehler
2^{-1}	0.20786640808609
2^{-6}	0.00773341103991
2^{-11}	0.00024299312415
2^{-16}	0.00000759482296
2^{-21}	0.00000023712637
2^{-26}	0.00000001020730
2^{-31}	0.00000005960464
2^{-36}	0.00000679016113

$f(x) = \sqrt{x}$

h	Relativer Fehler
2^{-1}	0.09340033543136
2^{-6}	0.00352613693103
2^{-11}	0.00011094838842
2^{-16}	0.00000346787667
2^{-21}	0.00000010812198
2^{-26}	0.00000001923506
2^{-31}	0.00000001202188
2^{-36}	0.00000198842224

$f(x) = \exp(x)$

h	Relativer Fehler
2^{-1}	0.29744254140026
2^{-6}	0.00785334954789
2^{-11}	0.00024418036620
2^{-16}	0.00000762943394
2^{-21}	0.00000023835113
2^{-26}	0.00000000429331
2^{-31}	0.00000012467100
2^{-36}	0.00000495453865

◇

4.1.2 Interpolationsfehlerabschätzungen

INTERPABSCH

Wir betrachten Lagrangesche Polynominterpolation zu paarweise verschiedenen Knoten $\mathcal{T} := \{t_0, \ldots, t_n\} \subset I$, $I \subset \mathbb{R}$ Intervall mit Länge $|I|$.

✎ Notation: Für eine stetige Funktion $f : I \mapsto \mathbb{K}$ definiere Polynominterpolationsoperator
$\mathsf{I}_{\mathcal{T}}(f) := \mathsf{I}_{\mathcal{T}}(\mathbf{y}) \in \mathcal{P}_n$ mit $\mathbf{y} := (f(t_0), \ldots, f(t_n))^T \in \mathbb{K}^{n+1}$.

Fragestellung: Genauigkeit der Approximatation einer Funktion (= Interpoland) durch ihr Lagrangesches Interpolationspolynom zur Knotenmenge $\mathcal{T}$.

Spezielle Fragestellung: Asymptotisches Fehlerverhalten

$$\exists C \neq C(n): \quad \|f - \mathsf{I}_{\mathcal{T}} f\| \leq C\, T(n) \quad \text{für } n \to \infty \ .$$

Klassifikation (beste Schranken für $T(n)$):

$\exists p > 0$: $T(n) \leq n^{-p}$: Algebraische Konvergenz, Rate p , ALGCVG

$\exists 0 < q < 1$: $T(n) \leq q^n$: Exponentielle Konvergenz . EXPCVG

Beachte: Gleicher Begriff $\leftrightarrow$ verschiedene Bedeutungen:

- Konvergenz einer Folge (von Iterierten $\boldsymbol{x}^{(k)} \to$ Abschn. 2.1) sec:iterationsverfahren
- Konvergenz einer Approximation (abhängig von einem Approximationsparameter)

Theorem 4.1.4 (Fehler der Polynominterpolation). *Für $f \in C^{n+1}(I)$: $\forall t \in I$:*

$$f(t) - \mathsf{I}_{\mathcal{T}}(f)(t) = \int_0^1 \int_0^{\tau_1} \cdots \int_0^{\tau_{n-1}} \int_0^{\tau_n} f^{(n+1)}(t_0 + \tau_1(t_1 - t_0) + \cdots + \tau_n(t_n - t_{n-1}) + \tau(t - t_n))\, \mathrm{d}\tau \mathrm{d}\tau_n \cdots \mathrm{d}\tau_1 \cdot \prod_{j=0}^{n}(t - t_j) \ .$$

Beweis. Durch Induktion nach n, verwendet (4.1.12) eq:ipolrec und Hauptsatz der Integral- und Differentialrechnung [38, Sect. 3.1]: RAN00

Bemerkung 160. Thm. 4.1.4 thm:polintperr gilt auch für Hermite-Interpolation.

△

Korollar 4.1.5. $f \in C^{n+1}(I)$: $\forall t \in I$: $\exists \tau_t \in]\min\{t, t_0, \ldots, t_n\}, \max\{t, t_0, \ldots, t_n\}[$:

$$f(t) - \mathsf{I}_{\mathcal{T}}(f)(t) = \frac{f^{(n+1)}(\tau_t)}{(n+1)!} \cdot \prod_{j=0}^{n}(t - t_j) \ .$$

Interpolationsfehlerabschätzungen erfordern Glattheit !

Beispiel 161 (Interpolationsfehler).

Interpolation von $f(t) = \sin t$, äquidistante Knoten in $I := [0, \pi]$: $\mathcal{T} := \{j\pi/n\}_{j=0}^n$: Interpolationspolynom $p := \mathsf{I}_{\mathcal{T}} f \in \mathcal{P}_n$

$$\left\| f^{(k)} \right\|_{L^\infty(I)} \leq 1 \ , \quad \forall k \in \mathbb{N}_0 \quad \Rightarrow \quad \|f - p\|_{L^\infty(I)} \leq \frac{1}{(1+n)!} \left\| (. - 0)(. - \tfrac{\pi}{n})(. - \tfrac{2\pi}{n}) \cdot \cdots \cdot (. - \pi) \right\|_{L^\infty(I)} \leq \frac{1}{n+1} \left(\frac{\pi}{n}\right)^{n+1} \ .$$

➢ Gleichmässige exponentielle Konvergenz der Interpolationspolynome
(Gilt für beliebige Knotenmengen $\mathcal{T}$)

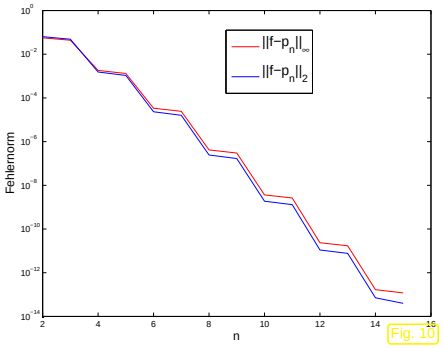


Fig. 10

MATLAB-Experiment:

Berechnung der Normen:

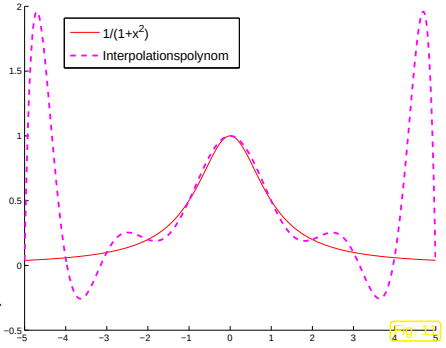
- L^∞-Norm: Abtastung auf Gitter mit Maschenweite $\pi/1000$.
- L^2-Norm: Numerische Quadratur ($\rightarrow$ Abschnitt 4.4) cha:numerische-quadratur mit Trapezregel auf Gitter mit Maschenweite $\pi/1000$.

◇

Beispiel 162 (Runges Beispiel).

Polynominterpolation von $f(t) = \frac{1}{1+t^2}$ mit äquidistanten Knoten:

$$\mathcal{T} := \left\{ -5 + \frac{10}{n} j \right\}_{j=0}^{n} ,$$
$$y_j = \frac{1}{1+t_j^2} , \quad j = 0, \ldots .$$



Plot $n = 10$ ➢

Oszillationen von $\mathsf{I}_{\mathcal{T}} f$ am Intervallende

► $\|f - \mathsf{I}_{\mathcal{T}} f\|_{L^\infty(]-5,5[)} \to \infty$ für $n \to \infty$.

runge

Hier: $f(t) = \frac{1}{1+t^2}$ ► $|f^{(n)}(t)| = 2^n n! \cdot O(|t|^{-2-n})$.

➢ Schranke für Fehlerglied aus Cor. 4.1.5 $\to \infty$ für $n \to \infty$.

cor:polintperr

◇

Bemerkung 163 (L^2-Fehlerabschätzungen).

Cor. 4.1.5 ➢ Fehlerabschätzung für L^∞-Norm andere Normen **?**

Thm. 4.1.4

Aus Thm. 4.1.4 mit Cauchy-Schwarz-Ungleichung für Integrale:

$$\begin{aligned} \|f - \mathsf{I}_{\mathcal{T}}(f)\|_{L^2(I)}^2 &= \int_I \left| \int_0^1 \int_0^{\tau_1} \cdots \int_0^{\tau_{n-1}} \int_0^{\tau_n} f^{(n+1)}(\ldots) \,\mathrm{d}\tau \mathrm{d}\tau_n \cdots \mathrm{d}\tau_1 \cdot \prod_{j=0}^{n} (t - t_j) \right|^2 \mathrm{d}t \\ &\le \int_I |I|^{2n+2} \mathrm{vol}(S_{n+1}) \int_{S_{n+1}} |f^{(n+1)}(\ldots)|^2 \,\mathrm{d}\boldsymbol{\tau} \,\mathrm{d}t \\ &\le \int_I \frac{|I|^{2n+2}}{(n+1)!} \int_I \underbrace{\mathrm{vol}(C_{t,\tau})}_{\le 2^{(n-1)/2}/n!} |f^{(n+1)}(\tau)|^2 \,\mathrm{d}\tau \mathrm{d}t , \end{aligned}$$

wobei

$$S_{n+1} := \{\mathbf{x} \in \mathbb{R}^{n+1} : 0 \le x_n \le x_{n-1} \le \cdots \le x_1 \le 1\} \quad \text{(Einheitssimplex)} ,$$
$$C_{t,\tau} := \{\mathbf{x} \in S_{n+1} : t_0 + x_1(t_1 - t_0) + \cdots + x_n(t_n - t_{n-1}) + x_{n+1}(t - t_n) = \tau\} .$$

► $\|f - \mathsf{I}_{\mathcal{T}}(f)\|_{L^2(I)} \le \frac{2^{(n-1)/4} |I|^{n+1}}{\sqrt{(n+1)! n!}} \left(\int_I |f^{(n+1)}(\tau)|^2 \,\mathrm{d}\tau \right)^{1/2}$. (4.1.13) eq:intpl2

Beachte: $f \mapsto \left\| f^{(n)} \right\|_{L^2(I)}$ definiert eine Seminorm auf $C^{n+1}(I)$

(Sobolev-Seminorm, Mass für Glattheit einer Funktion)

△

4.1.3 Tschebyscheff-Interpolation

TSCHEBYSCHEFF

Knotenmenge: $\mathcal{T} := \{t_0 < t_1 < \cdots < t_{n-1} < t_n\}$, $n \in \mathbb{N}$, $I := [t_0, t_n]$

Cor. 4.1.5 cor:polintperr ►

$$\|f - p\|_{L^\infty(I)} \le \frac{1}{(n+1)!} \left\| f^{(n+1)} \right\|_{L^\infty(I)} \|w\|_{L^\infty(I)} ,$$
$$w(t) := (t - t_0) \cdot \cdots \cdot (t - t_n) .$$

Idee: Knoten $t_0, \ldots, t_n$ so, dass $\|w\|_{L^\infty(I)} \to \min$!

$\leftrightarrow$ Finde $q \in \mathcal{P}_{n+1}$, führender Koeffizient $= 1$, $\|q\|_{L^\infty(I)} \to \min$.

► Wähle $t_0, \ldots, t_n$ **:=** Nullstellen von q (Vorsicht: $t_j \in I$ **?**)

Heuristik:
- q hat $n+1$ Nullstellen in I
- $|q(t_0)| = |q(t_n)| = \|q\|_{L^\infty(I)}$
- t^* Extremalstelle von q ➢ $|q(t^*)| = \|q\|_{L^\infty(I)}$

Definition 4.1.6 (Tschebyscheff-Polynome)**.**

n. *Tschebyscheff-Polynom* **=** $T_n(t) := \cos(n \arccos t)$, $-1 \le t \le 1$

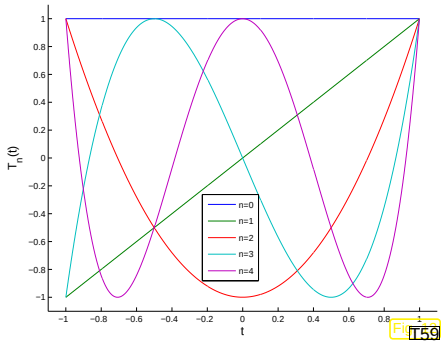


Fig. 13

Tschebyscheff-Polynome $T_0, \ldots, T_4$

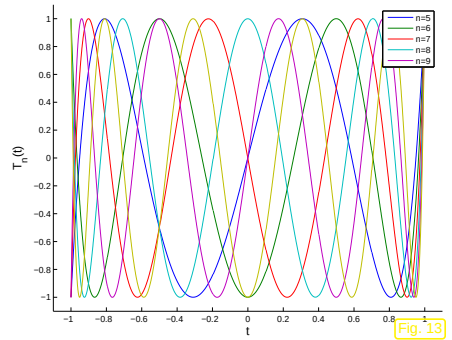


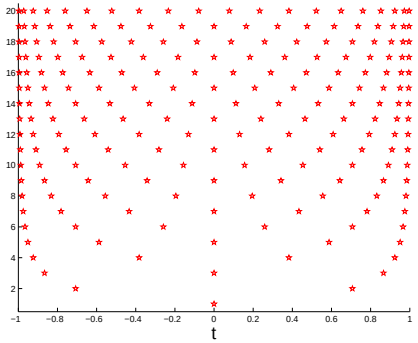
Fig. 13

Tschebyscheff-Polynome $T_5, \ldots, T_9$

Nullstellen von T_n:

$$t_k = \cos\left(\frac{2k-1}{2n}\pi\right) \ , \quad k = 1, \ldots, n \ .$$

(4.1.14)



Extrema (Alternanten) von T_n an

$$|T_n(\bar{t}_k)| = 1 \Leftrightarrow \exists k = 0, \ldots, n\colon \bar{t}_k = \cos\frac{k\pi}{n} \ .$$

3-Term-Rekursion (→ Abschnitt 4.2) aus $\cos(n+1)x = 2\cos nx \cos x - \cos(n-1)x$ mit $\cos x = t$:

$$T_{n+1}(t) = 2t\,T_n(t) - T_{n-1}(t) \quad , \quad T_0 \equiv 1 \ , \quad T_1(t) = t \ , \quad n \in \mathbb{N} \ . \tag{4.1.15}$$

$T_n \in \mathcal{P}_n$, führender Koeffizient 2^{n-1}

$\mathcal{P}_n = \operatorname{Span}\{T_0, \ldots, T_n\}$, $n \in \mathbb{N}_0$, T_n linear unabhängig (→ Basis)

Theorem 4.1.7 (Minimax Eigenschaft der Tschebyscheff-Polynome).

$$\|T_n\|_{L^\infty([-1,1])} = \inf\{\|p\|_{L^\infty([-1,1])} : p \in \mathcal{P}_n, p(t) = 2^{n-1}t^n + \cdots\} \ , \quad \forall n \in \mathbb{N} \ .$$

Zum Beweis siehe [12], Abschnitt 7.1.4.

Für $I = [-1,1]$ → „Optimale" Interpolationsknoten $\mathcal{T} = \left\{\cos\left(\frac{2k+1}{2(n+1)}\pi\right) \ , k = 0, \ldots, n\right\}$,

→ $w = 2^{-n}T_{n+1}$, $\|w\|_{L^\infty(I)} = 2^{-n}$,

→ $$\|f - \mathsf{I}_{\mathcal{T}}(f)\|_{L^\infty([-1,1])} = \frac{2^{-n}}{(n+1)!}\left\|f^{(n+1)}\right\|_{L^\infty([-1,1])} \ .$$

Ein Skalierungsargument: Intervalltransformation ↔ Transport von Funktionen

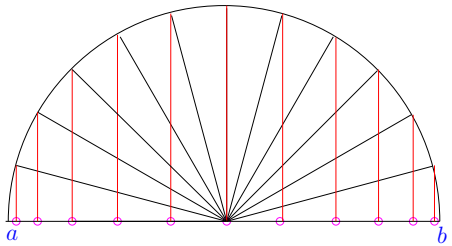
$$[-1,1] \xrightarrow{\widehat{t} \mapsto t := a + \frac{1}{2}(\widehat{t}+1)(b-a)} [a,b] \quad \leftrightarrow \quad \widehat{f}(\widehat{t}) := f(t) \ .$$

$$p \in \mathcal{P}_n \ \wedge \ p(t_j) = f(t_j) \ \Leftrightarrow \ \widehat{p} \in \mathcal{P}_n \ \wedge \ \widehat{p}(\widehat{t}_j) = \widehat{f}(\widehat{t}_j) \ .$$

Mit Transformationsformel für Integrale & $\frac{\mathrm{d}^n\widehat{f}}{\mathrm{d}\widehat{t}^n}(\widehat{t}) = (\frac{1}{2}|I|)^n \frac{\mathrm{d}^n f}{\mathrm{d}t^n}(t)$:

$$\|f - \mathsf{I}_{\mathcal{T}}(f)\|_{L^\infty(I)} = \left\|\widehat{f} - \mathsf{I}_{\widehat{\mathcal{T}}}(\widehat{f})\right\|_{L^\infty([-1,1])} \le \frac{2^{-n}}{(n+1)!}\left\|\frac{\mathrm{d}^{n+1}\widehat{f}}{\mathrm{d}\widehat{t}^{n+1}}\right\|_{L^\infty([-1,1])}$$

$$\le \frac{2^{-2n-1}}{(n+1)!}|I|^{n+1}\left\|f^{(n+1)}\right\|_{L^\infty(I)} \ . \tag{4.1.16}$$



CHEBNODES
Für $I = [a,b]$: Tschebyscheff-Knoten

$$t_k := a + \tfrac{1}{2}(b-a)\left(\cos\left(\frac{2k+1}{2(n+1)}\pi\right) + 1\right) \ , \tag{4.1.17}$$

$$k = 0, \ldots, n \ .$$

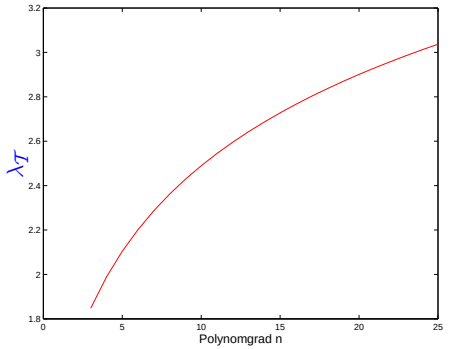
Lebegue-Konstanten für Tschebyscheff-Knoten

Theorie [6]:

$$\lambda_{\mathcal{T}} \sim \frac{2}{\pi}\log(1+n) + o(1) \ ,$$

$$\lambda_{\mathcal{T}} \le \tfrac{2}{\pi}\log(1+n) + 1 \ . \tag{4.1.18}$$

→



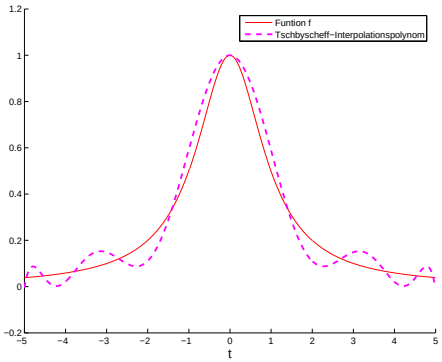
Beispiel 164 (Tschebyscheff-Interpolation).

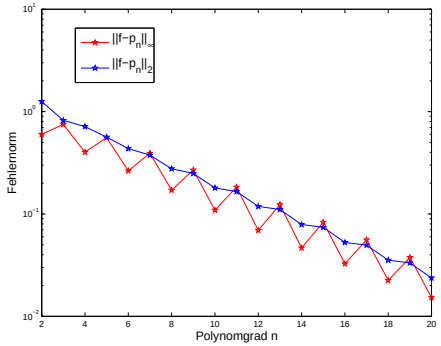
Für $I = [a,b]$ seien $x_l := a + \frac{b-a}{N}l$, $l = 0, ..., N$, $N = 1000$ und man approximiert

$$||f-p||_\infty \approx \max_{0 \le l \le N} |f(x_l) - p(x_l)|$$

$$||f-p||_2^2 \approx \frac{b-a}{2N}\sum_{0 \le l < N}\left(|f(x_l) - p(x_l)|^2 + |f(x_{l+1}) - p(x_{l+1})|^2\right)$$

① $f(t) = (1+t^2)^{-1}$ → Bsp. 162, $I = [-5,5]$, $n = 10$ im linken Plot.

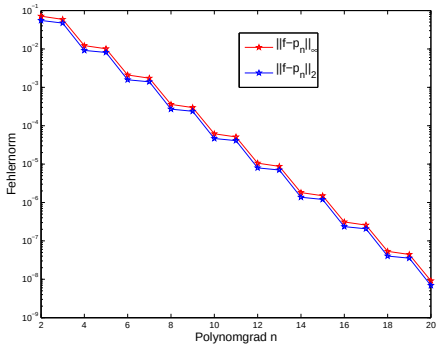




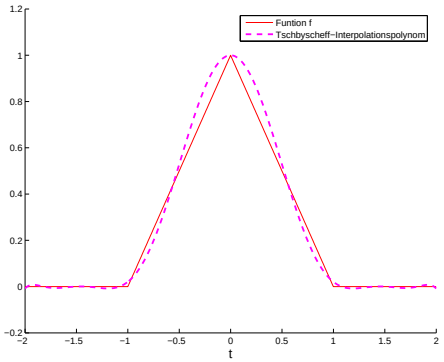
➢ Beobachtung: Exponentielle Konvergenz der Tschebyscheff-Interpolation
$p_n \to f$, $\|f - \mathsf{I}_n f\|_{L^2([-5,5])} \approx 0.8^n$.

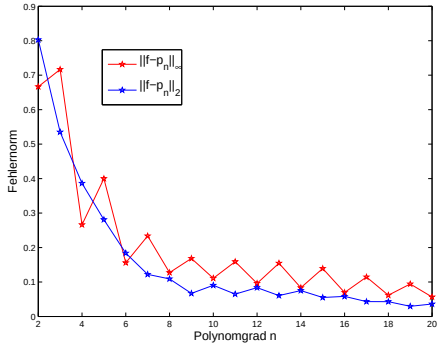
Nun: $I = [-1, 1]$
Weiterhin $f(t) = (1 + t^2)^{-1}$
➢ Beobachtung:
Exponentielle Konvergenz:

$$\|f - \mathsf{I}_n f\|_{L^2([-5,5])} \approx 0.42^n.$$

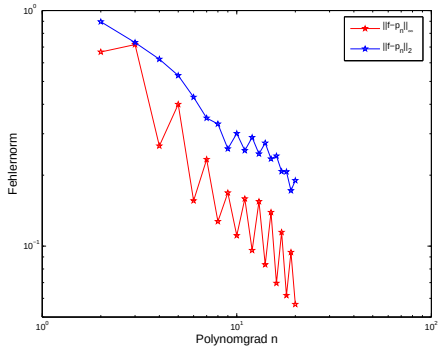


② $f(t) = \max\{1 - |t|, 0\}$, $I = [-2, 2]$, $n = 10$ im linken Plot.

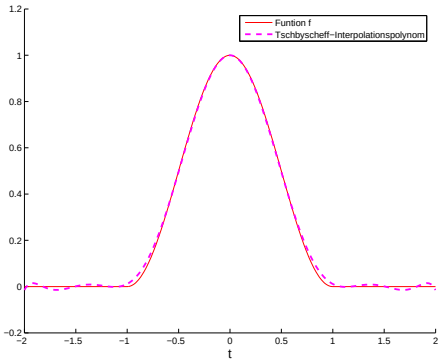


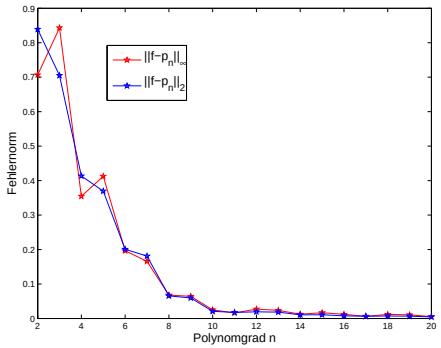


Doppeltlogarithmischer Plot
➢ Beobachtung:
- keine exponentielle Konvergenz
- algebraische Konvergenz (**?**)



② $f(t) = 1 - \frac{1}{2}(1 + \cos \pi t)$ für $|t| < 1$, $f \equiv 0$ sonst, $I = [-2, 2]$, $n = 10$ im linken Plot.





➢ Beobachtung: Nur algebraische Konvergenz ◇

4.1.4 Trigonometrische Interpolation

4.1.4.1 Trigonometrische Polynome

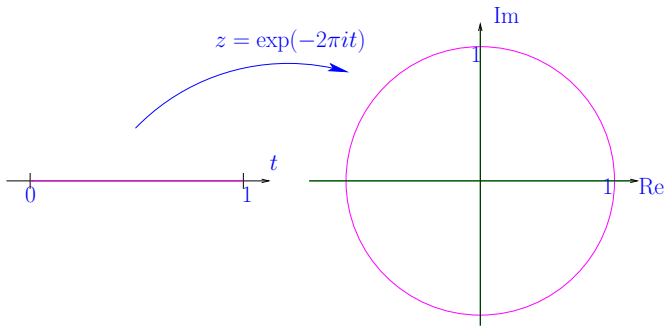
Definition 4.1.8 (Trigonometrisches Polynom)**.**
TRIGPOL
Der Raum $\mathcal{P}_n^T$ der trigonometrischen Polynome vom Grad $\leq n$ ist gegeben durch

$$\mathcal{P}_n^T := \begin{cases} \{t \mapsto \sum_{j=-m}^{m} \gamma_j e^{-2\pi i j t},\ \gamma_j \in \mathbb{C}\} & \text{, falls } n = 2m,\ m \in \mathbb{N}\ , \\ \{t \mapsto \sum_{j=-m+1}^{m} \gamma_j e^{-2\pi i j t},\ \gamma_j \in \mathbb{C}\} & \text{, falls } n = 2m-1,\ m \in \mathbb{N}\ . \end{cases}$$

Warum trigonometrische Polynome ?

$$p(t) := \sum_{j=-m[+1]}^{m} \gamma_j e^{-2\pi i j t} = \sum_{j=-m[+1]}^{m} \gamma_j z^j \ \hat{=}\ \text{(Laurent)-Polynom auf } \mathbb{S}^1$$

Parametrisierung: $[0,1[\xrightarrow[t \mapsto z]{z = \exp(-2\pi i t)} \mathbb{S}^1 := \{z \in \mathbb{C}\colon |z| = 1\}$.



Definition 4.1.9 (Laurent-Polynome)**.**
Der Raum $\mathcal{P}_n^L$ der Laurent-Polynome vom Grad $\leq n$ über $\mathbb{K}$ ist gegeben durch

$$\mathcal{P}_n^L := \begin{cases} \{z \mapsto \sum_{j=-m}^{m} \gamma_j z^j,\ \gamma_j \in \mathbb{K}\} & \text{, falls } n = 2m,\ m \in \mathbb{N}\ , \\ \{z \mapsto \sum_{j=-m+1}^{m} \gamma_j z^j,\ \gamma_j \in \mathbb{K}\} & \text{, falls } n = 2m-1,\ m \in \mathbb{N}\ . \end{cases}$$

4.1 p. 401 Beachte: trigonometrische Polynome $\subset$ 1-periodische Funktionen auf $\mathbb{R}$ ($\leftrightarrow f(t) = f(t+1)$) 4.1 p. 403

TIP
Trigonometrische Polynominterpolationsaufgabe

Zur *äquidistanten* Knoten $\mathcal{T} := \{j/n, j = 0, \ldots, n-1\}$ und Stützwerten $y_0, \ldots, y_{n-1} \in \mathbb{C}$ finde $p \in \mathcal{P}_{n-1}^T$, so dass

$$p(j/n) = y_j\ , \quad j = 0, \ldots, n-1\ .$$

Theorem 4.1.10. *Die trigonometrische Polynominterpolationsaufgabe ist eindeutig lösbar $\forall y_k$.*

Beweis. siehe unten. □

◣ trigonometrische Polynominterpolationsaufgabe $\Leftrightarrow$ lineare Abbildung $\mathsf{T}_n : \mathbb{C}^n \mapsto \mathcal{P}_{n-1}^T$.

Beachte: 1-periodische Funktionen auf $\mathbb{R}$ $\leftrightarrow$ Funktionen $\mathbb{S}^1 \mapsto \mathbb{K}$

$$f(t) = f(t+1)\ \ \forall t \in \mathbb{R} \ \leftrightarrow\ g(z) = g(e^{2\pi i t}) := f(t)\ \ \forall z \in \mathbb{S}^1\ .$$

4.1 p. 402 4.1 p. 404

► Trigonometrische Polynominterpolationsaufgabe ⇔ Äquidistante Polynominterpolation auf $\mathbb{S}^1$

► Zu Knotenmenge $\mathcal{T}_n = \{\omega_n^j\}_{j=0}^{n-1}$ (komplexe Einheitswurzeln, Abschnitt 3.7.1) und Stützwerten $y_0, \ldots, y_{n-1}$ finde $p \in \mathcal{P}_{n-1}^L$ (komplexe Koeffizienten), so dass

$$p(\omega_n^k) = y_k\,, \quad k = 0, \ldots, n-1\,.$$

► $\mathsf{T}_n \leftrightarrow$ Lineare Abbildung $\mathbb{C}^{n+1} \mapsto \mathcal{P}_n^L$.

Umformulierung der trigonometrischen Polynominterpolationsaufgabe:

❶ n gerade, $n = 2m$, $m \in \mathbb{N}$, ➢ $p(t) = \sum_{j=-m+1}^{m} \gamma_j e^{-2\pi i j t} \in \mathcal{P}_{n-1}^L$

$$p(\tfrac{k}{n}) = y_k\,, \quad k = 0, \ldots, n-1\,.$$

$$\Updownarrow$$

$$\sum_{j=-m+1}^{m} \gamma_j \omega_n^{jk} = \sum_{j=0}^{m} \gamma_j \omega_n^{jk} + \sum_{j=1}^{m-1} \gamma_{-j} \omega_n^{(n-j)k} = \boxed{\sum_{j=0}^{m} \gamma_j \omega_n^{jk} + \sum_{j=m+1}^{n-1} \gamma_{j-n} \omega_n^{jk} = y_k}\,,$$

$\omega_n \mathrel{\hat{=}}$ komplexe Einheitswurzeln, Abschnitt 3.7.1.

❶ n ungerade, $n = 2m+1$, $m \in \mathbb{N}$, ➢ $p(t) = \sum_{j=-m}^{m} \gamma_j e^{-2\pi i j t} \in \mathcal{P}_{n-1}^L$

$$p(\tfrac{k}{n}) = y_k\,, \quad k = 0, \ldots, n-1\,.$$

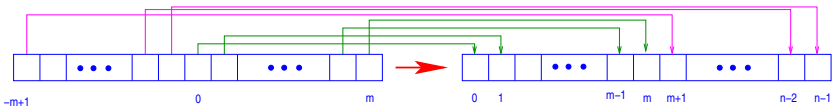
$$\Updownarrow$$

$$\sum_{j=-m}^{m} \gamma_j \omega_n^{jk} = \sum_{j=0}^{m} \gamma_j \omega_n^{jk} + \sum_{j=1}^{m} \gamma_{-j} \omega_n^{(n-j)k} = \boxed{\sum_{j=0}^{m} \gamma_j \omega_n^{jk} + \sum_{j=m+1}^{n-1} \gamma_{j-n} \omega_n^{jk} = y_k}\,.$$

$\boxed{} \mathrel{\hat{=}} \mathbf{F}_n \mathbf{g} = \mathbf{y}\,, \quad \mathbf{g} = \begin{pmatrix} \gamma_0 & \cdots & \gamma_m & \gamma_{m+1-n} & \cdots & \gamma_{n-1} \end{pmatrix}\,, \quad \mathbf{y} = \begin{pmatrix} y_0 & \cdots & \cdots & y_{n-1} \end{pmatrix}\,.$

$\mathbf{F}_n$ Fourier-Matrix (3.7.3)

➤ γ_j durch inverse diskrete Fouriertransformation (→ Def. 3.7.1) + Umsortieren der Koeffizienten:



Aufgabe: *Auswertung* eines trigonometrischen Polynoms $p \in \mathcal{P}_{n-1}^T$ (→ Def. 4.1.8) an äquidistanten Stellen $\frac{l}{N}$, $l = 0, \ldots, N-1$, $N > n$, in $[0, 1[$:

Für $n = 2m$:

$$p(\tfrac{l}{N}) = \sum_{j=-m+1}^{m} \gamma_j \omega_N^{lj} = \sum_{j=0}^{m} \gamma_j \omega_N^{lj} + \sum_{j=1}^{m-1} \gamma_{-j} \omega_N^{(N-j)l} = \sum_{j=0}^{N-1} \widetilde{\gamma}_j \omega_N^{lj}\,,$$

$$\text{mit} \quad \widetilde{\gamma}_j := \begin{cases} \gamma_j & \text{, falls } 0 \le j \le m-1\,, \\ \gamma_{j-N} & \text{, falls } N-m+1 \le j \le N-1\,, \\ 0 & \text{sonst.} \end{cases}$$

➤ diskrete Fouriertransformation → Def. 3.7.1 der Länge N

Koeffizienten in c
umsortiert gespeichert !

MATLAB-CODE Auswertung eines trigonometrischen Interpolationspolynom

```
function v= evaliptrig(y,N)
n = length(y);
if (mod(n,2) ~= 0), error; end
c = ifft(y);
c = [c(1:n/2),zeros(1,N-n),...
     c(n/2+1:n)];
v = fft(c);
```

4.1.4.2 Trigonometrische Interpolation 1-periodischer Funktionen:

☞ Linearer trigonometrischer Interpolationsoperator: für 1-periodisches $f : \mathbb{R} \mapsto \mathbb{C}$

$$\mathsf{T}_n(f) := p \in \mathcal{P}_{n-1}^T \quad \text{mit} \quad p(\tfrac{j}{n}) = y_j\,, \quad j = 0, \ldots, n-1\,.$$

Beispiel 165 (Trigonometrische Interpolation).

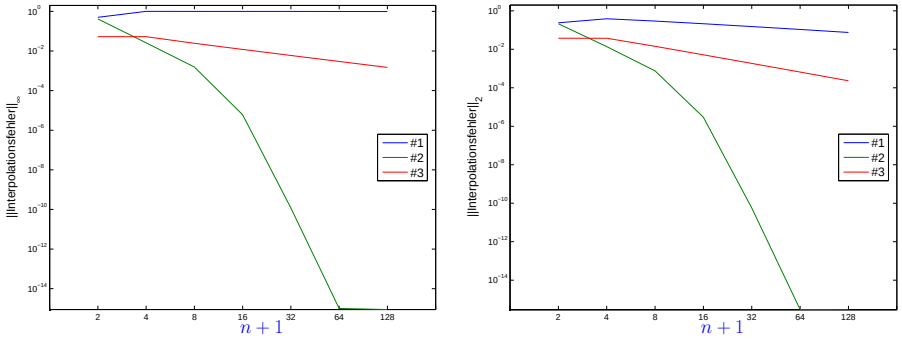
#1 Stufenfunktion: $f(t) = 0$ für $|t - \frac{1}{2}| > \frac{1}{4}$, $f(t) = 1$ für $|t - \frac{1}{2}| \le \frac{1}{4}$

#2 Glatte periodische Funktion: $f(t) = \dfrac{1}{\sqrt{1 + \frac{1}{2}\sin(2\pi t)}}$.

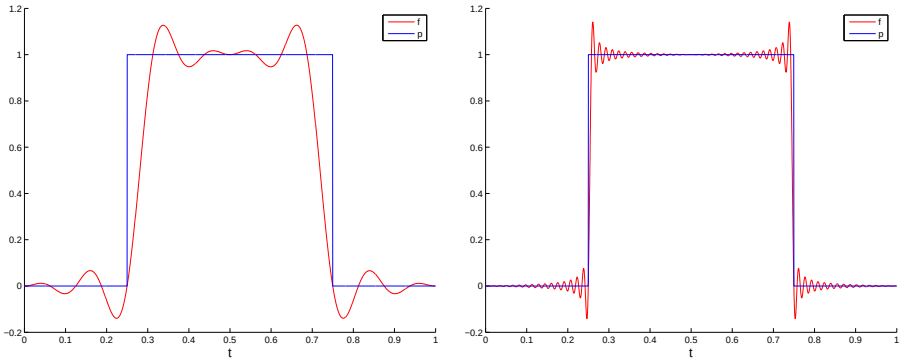
#3 Keilfunktion: $f(t) = |t - \frac{1}{2}|$

Berechnung der Normen der Interpolationsfehler:

```
t = 0:1/n:1; y = feval(f,t(1:end-1));
v = real(evaliptrig(y,4096));
v = [v,v(1)]; fv = feval(f,0:1/4096:1);
d = abs(fv-v); linf = max(d); l2 = 1/64*norm(d(1:end-1));
```



Beobachtung: Fall #1, #3: Algebraische Konvergenz
Fall #2: Exponentielle Konvergenz

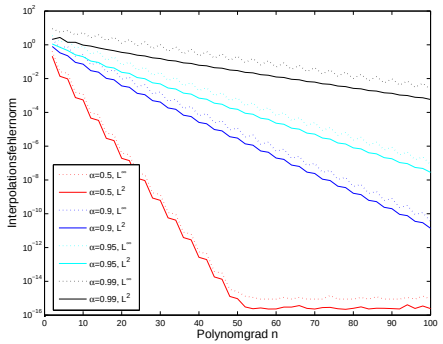


➢ Gibbsches Phänomen: Überschwinger an Unstetigkeitsstellen ◇

Beispiel 166 (Trigonometrische Interpolation analytischer Funktionen).

$$f(t) = \frac{1}{\sqrt{1-\alpha\sin(2\pi t)}} \quad \text{auf } I = [0,1] \ .$$

Approximative Messung der Normen: “oversampling” in 4096 Punkten



➢ Beobachtet: Exponentielle Konvergenz im Grad n, je schneller, je kleiner α ◇

Analyse benutze: Trigonometrische Polynome = Partialsummen von Fourierreihen

Bekannt aus → Analysis, → Mathematische Methoden der Physik:

Theorem 4.1.11 (L^2-Konvergenz der Fourierreihe)**.** *Jede quadratintegrierbare Funktion* $f \in L^2(]0,1[) := \{f :]0,1[\mapsto \mathbb{C}: \|f\|_{L^2(]0,1[)} < \infty\}$ *ist der* $L^2(]0,1[)$*-Grenzwert ihrer Fourierreihe*

$$f(t) = \sum_{k=-\infty}^{\infty} \widehat{f}(k)\, e^{2\pi i k t} \quad \text{in } L^2(]0,1[) \ ,$$

mit Fourierkoeffizienten

$$\widehat{f}(k) = \int_0^1 f(t)\, e^{-2\pi i k t}\, \mathrm{d}t \ , \quad k \in \mathbb{Z} \ .$$

Isometrieeigenschaft: $\sum_{k=-\infty}^{\infty} |\widehat{f}(k)|^2 = \|f\|^2_{L^2(]0,1[)}$ (4.1.19) eq:Fouriso

Lemma 4.1.12 (Ableitung und Fourierkoeffizienten).

$$f \in L^1(]0,1[) \;\&\; f' \in L^1(]0,1[) \quad \Rightarrow \quad \widehat{f'}(k) = 2\pi i k \widehat{f}(k), \quad k \in \mathbb{Z}.$$

$$\blacktriangleright \quad \left\| f^{(n)} \right\|^2_{L^2([0,1])} = (2\pi)^{2n} \sum_{k=-\infty}^{\infty} k^{2n} |\widehat{f}(k)|^2 \,. \tag{4.1.20}$$ eq:FN

Falls $|\widehat{f^{(n)}}(k)| \le \int_0^1 |f^{(n)}(t)|\,\mathrm{d}t < \infty \quad \Rightarrow \quad \widehat{f}(k) = O(k^{-n})$ für $|k| \to \infty$.

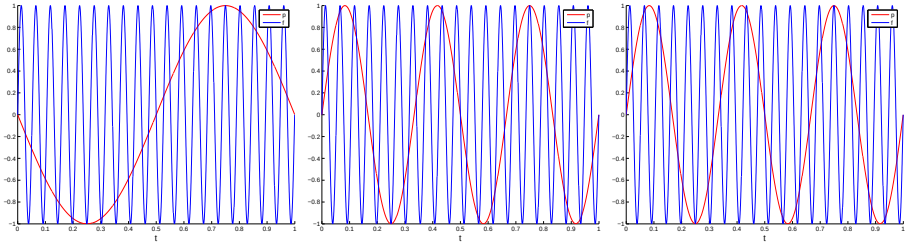
Je glatter eine 1-periodische Funktion, desto schneller der Abfall ihrer Fourierkoeffizienten

Bemerkung 167 (Aliasing).

$$\mathcal{T} = \{\tfrac{j}{n}\}_{j=0}^{n-1} \quad \blacktriangleright \quad e^{2\pi i N t}_{|\mathcal{T}} = e^{2\pi i (N-n) t}_{|\mathcal{T}} \quad \blacktriangleright \quad \mathsf{T}_n(e^{2\pi i N \cdot}) = \mathsf{T}_n(e^{2\pi i (N \bmod n)\cdot}) \,.$$

➢ Die trigonometrische Interpolation von $t \to e^{2\pi i N t}$ und $t \to e^{2\pi i (N \bmod n) t}$ mit Polynomgrad n liefert das gleiche trigonometrische Interpolationspolynom **!** → ALIASING Aliasing

Beispiel für $f(t) = \sin(2\pi \cdot 19t)$ ➢ $N = 38$:



$n = 4$ $\qquad$ $n = 8$ $\qquad$ $n = 16$ △

[Linearität der trigonometrischen Interpolation]

Für $f \in C([0,1])$ 1-periodisch, $n = 2m$:

$$f(t) = \sum_{j=-\infty}^{\infty} \widehat{f}(j) e^{2\pi i j t} \quad \blacktriangleright \quad \mathsf{T}_n(f)(t) = \sum_{j=-m+1}^{m} \gamma_j e^{2\pi i j t} \,, \quad \gamma_j = \sum_{l=-\infty}^{\infty} \widehat{f}(j+ln) \,.$$

$\blacktriangleright$ Fourier-Koeffizienten der Interpolationsfehlerfunktion $e = f - \mathsf{T}_n(f)$:

$$\widehat{e}(j) = \begin{cases} -\sum\limits_{|l|=1}^{\infty} \widehat{f}(j+ln) & \text{, falls } -m+1 \le j \le m \,, \\ \widehat{f}(j) & \text{, falls } j \le -m \vee j > m \,. \end{cases}$$

(4.1.19) eq:Fouriso eq:fourierest

$$\blacktriangleright \quad \|f - \mathsf{T}_n(f)\|^2_{L^2(]0,1[)} \le \left| \sum_{|l|=1}^{\infty} \widehat{f}(j+ln) \right|^2 + \sum_{|j|\ge m} |\widehat{f}(j)|^2 \,. \tag{4.1.21}$$ eq:fourie

$\blacktriangleright$ Abschätzbar bei bekanntem Abfall der $\widehat{f}(j)$ $\leftrightarrow$ Glattheit von f, vgl. (4.1.20) eq:FN

Theorem 4.1.13 (Fehlerabschätzung für trigonometrische Interpolation). *Für $k \in \mathbb{N}$:*

$$f^{(k)} \in L^2(]0,1[) \quad \Rightarrow \quad \|f - \mathsf{T}_n f\|_{L^2(]0,1[)} \le \sqrt{1+c_k}\, n^{-k} \left\| f^{(k)} \right\|_{L^2(]0,1[)} \,,$$

mit $c_k = 2\sum_{l=1}^{\infty} (2l-1)^{-2k}$.

4.1.4.3 Trigonomentrische Interpolation analytischer Funktionen

Was passiert wenn $f : \mathbb{R} \mapsto \mathbb{C}$ 1-periodisch und $f \in C^\infty(\mathbb{R})$ **?** (Aussage von Thm. 4.1.13) thm:triperrest

Definition 4.1.14 (Analytische Funktion).
Eine Funktion $f : D \subset \mathbb{C} \mapsto \mathbb{C}$ heisst analytisch ($\leftrightarrow$ holomorph) (in D), falls (im Sinne lokal gleichmässiger Konvergenz)

$$\forall z \in D\colon \;\; \exists r > 0,\, (\alpha_k)_{k=0}^{\infty} \in \mathbb{C}^{\mathbb{N}_0}\colon \;\; |w - z| < r \quad \Rightarrow \quad f(w) = \sum_{k=0}^{\infty} \alpha_k (w-z)^k \,.$$

Analytizität $\leftrightarrow$ lokale Darstellbarkeit durch Potenzreihe

Noch glatter: Ganze Funktionen $\leftrightarrow$ global konvergente Potenzreihe

➢ analytisch in $\mathbb{C}$, z.B. Polynome, $\exp z, \sin z, \cos z, \ldots$ + Kompositionen.

„Eine Funktion sucht sich ihr Analytizitätsgebiet selbst **!**"

Beispiel 168 (Analytizitätsgebiete von Funktionen).

- $f(t) = (1+t^2)^{-1}$ (→ Bsp. 162): Interpretation als Funktion in $z \in \mathbb{C}$: ex:runge

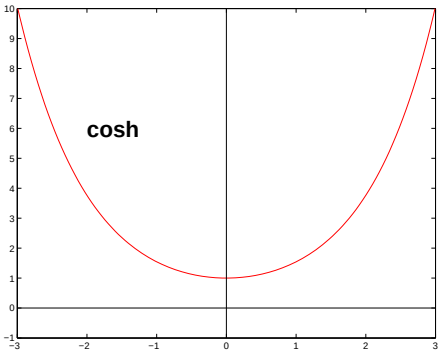
$$f(z) = \frac{1}{1+z^2} = \frac{1}{(z-i)(z+i)} \quad \Rightarrow \quad f \text{ analytisch in } \quad \mathbb{C} \setminus \{\pm i\} \,.$$

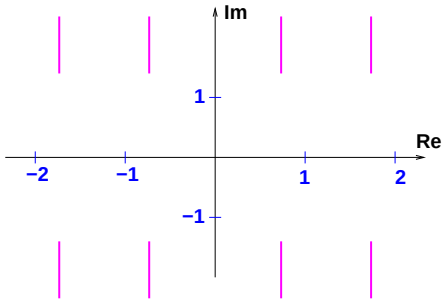
- $f(t) = \dfrac{1}{\sqrt{1+\alpha\sin(2\pi t)}}$, $\alpha > 0$ (→ Bsp. 165): Fortsetzung in $\mathbb{C}$, $\sqrt{}$ = Hauptzweig, analytisch auf $\mathbb{C} \setminus \mathbb{R}_0^-$.

ex:trigip

$1+\alpha\sin(2\pi z) \notin \mathbb{R}_0^- \Leftrightarrow \sin(2\pi z) = \sin(2\pi x)\cosh(2\pi y) + i\cos(2\pi x)\sinh(2\pi y) \notin]-\infty, -1-\frac{1}{\alpha}]$.

► Analytizitätsgebiet: $\mathbb{C} \setminus \bigcup_{k\in\mathbb{Z}} (\frac{k}{2} + \frac{1}{4} + i(\mathbb{R}\setminus]-\zeta,\zeta[))$, $\zeta \in \mathbb{R}^+$, $\cosh(2\pi\zeta) = 1 + \frac{1}{\alpha}$.





➢ $f(z)$ analytisch in Streifen $S := \{z \in \mathbb{C}: : -\zeta < \mathrm{Im}(z) < \zeta\}$.

◇

g **Theorem 4.1.15** (Exponentielle Konvergenz der trigonometrischen Interpolation)**.**
Falls $f : \mathbb{R} \mapsto \mathbb{C}$ 1-periodisch und analytisch (→ Def. 4.1.14) im Streifen

def:analytic

$$\bar{S} := \{z \in \mathbb{C}: \ -\eta \le \mathrm{Im}\, z \le \eta\} , \quad \eta > 0 ,$$

dann existiert ein nur von η abhängiges $C_\eta > 0$, so dass

$$\|f - \mathsf{T}_n f\|_X \le C_\eta e^{-\pi\eta n} \|f\|_{L^\infty(\bar{S})} , \quad n \in \mathbb{N} , \quad X = L^2(]0,1[), L^\infty(]0,1[) .$$

Beweis auf der Grundlage des folgenden Satzes, der aus (4.1.21) folgt.

eq:fourierest

1 **Theorem 4.1.16.** *Falls $f \in L^2(]0,1[)$ mit*

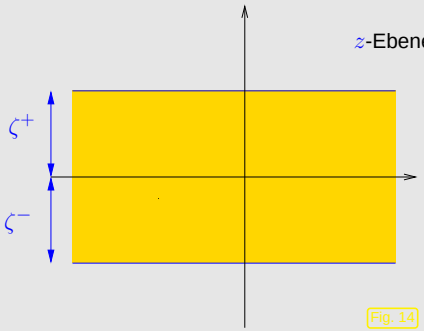
$$\exists M > 0, \rho \in]0,1[: \ |\widehat{f}(k)| \le M\rho^{|k|} \ \Rightarrow \ \begin{aligned} \|f - \mathsf{T}_n(f)\|_{L^2(]0,1[)} &\le M\frac{\rho^{n/2}}{1-\rho^n}\frac{2\sqrt{2}}{\sqrt{1-\rho^2}} , \\ \|f - \mathsf{T}_n(f)\|_{L^\infty(]0,1[)} &\le 4M\frac{\rho^{n/2}}{1-\rho} . \end{aligned}$$

Proof. (Beweis von Theorem 4.1.15)

thm:expcvg

Sei $f : \mathbb{R} \mapsto \mathbb{C}$ 1-periodisch, analytisch fortsetzbar in abgeschlossenen Streifen $S := \{z \in \mathbb{C}: -\eta^- \le \mathrm{Im}\{z\} \le \eta^+\}$

➢ $g_r(t) := f(t+ir)$, 1-periodisch und glatt für $-\eta^- \le r \le \eta^+$



analstrip

Fig. 14

$$\widehat{g_r}(k) = \int_0^1 f(t+ir)e^{-2\pi ikt}\,\mathrm{d}t = \int_0^1 f(t)e^{2\pi ik(t-ir)}\,\mathrm{d}t = e^{-2\pi rk}\widehat{f}(k) .$$

$$|\widehat{f}(k)| \le e^{-2\pi rk} \underbrace{\max\{|f(t+ir)|, 0 \le t \le 1\}}_{\|g_r\|_{L^\infty(]0,1[)}} \quad \forall k \in \mathbb{Z} , \quad \forall -\eta^- \le r \le \eta^+ .$$

$$|\widehat{f}(-k)| \le \exp(-2\pi\eta^-)^k \|f\|_{L^\infty(\bar{S})} \quad \forall k \in \mathbb{N} ,$$
$$|\widehat{f}(k)| \le \exp(-2\pi\eta^+)^k \|f\|_{L^\infty(\bar{S})} \quad \forall k \in \mathbb{N} .$$

(4.1.22) eq:TIPstr

► Exponentieller Abfall $\widehat{f}(k)$

thm:expcvg1
4.1.16

Beispiel 169 (Exponentielle Konvergenz der trigonometrischen Interpolation).

$f(t) = \dfrac{1}{\sqrt{1+\alpha\sin(2\pi t)}}$, $\alpha > 0$ (→ Bsp. 165 & Bsp. 168)

ex:trigip ex:anal

f analytisch in Streifen $S := \{x \in \mathbb{C}:| \mathrm{Im}\, z| < \zeta, \cosh(2\pi\zeta) = 1 + \frac{1}{a}\}$

Vorsicht: f unbeschränkt auf S **!** ➢ betrachte schmäleren Streifen.

◇

4.1.4.4 Trigonometrische Interpolation und Tschebyscheff-Interpolation

$p \in \mathcal{P}_n \hat{=}$ Tschebyscheff-Interpolationspolynom auf $[-1,1]$ zu $f : [-1,1] \mapsto \mathbb{C}$ (→ Abschnitt 4.1.3) sec:tschbyscl

$p(t_k) = f(t_k)$ für Tschebyscheff-Knoten, siehe (4.1.17), eq:CHEBNODES $t_k := \cos\left(\frac{2k+1}{2(n+1)}\pi\right)$, $k = 0, \ldots, n$.

$$\left.\begin{array}{rcl} f : [-1,1] \mapsto \mathbb{C} & \leftrightarrow & g(s) := f(\cos 2\pi s) , \\ p : [-1,1] \mapsto \mathbb{C} & \leftrightarrow & q(s) := p(\cos 2\pi s) , \end{array}\right\}$$ 1-periodisch, achsensymmetrisch.

$$p(t_k) = f(t_k) \quad \Leftrightarrow \quad q\left(\frac{2k+1}{4(n+1)}\right) = g\left(\frac{2k+1}{4(n+1)}\right) . \tag{4.1.23}$$ eq:TIPCS

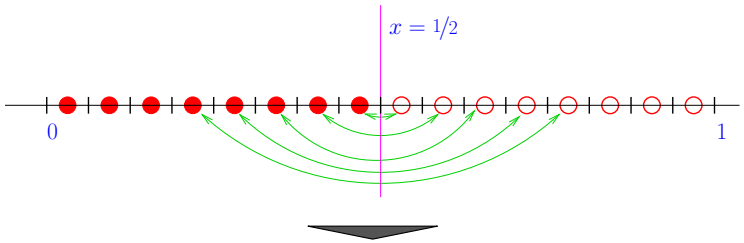
! Wir zeigen: $q(s)$ ist trigonometrisches Polynom: $\boxed{q \in \mathcal{P}_{2n}^T}$

$\mathcal{P}_n = \mathrm{Span}\{T_0, \ldots, T_n\}$, T_j = Tschebyscheff-Polynome → Def. 4.1.6 def:Tpol

$$\blacktriangleright \quad p(t) = \gamma_0 T_0(t) + \gamma_1 T_1(t) + \cdots + \gamma_n T_n(t) , \quad \gamma_j \in \mathbb{C} .$$

$$\blacktriangleright \quad q(s) = p(\cos 2\pi s) = \sum_{j=0}^{n} \gamma_j \cos(2\pi j s) = \sum_{j=-n}^{n} \tfrac{1}{2}\gamma_{|j|} e^{-2\pi i j s} , \quad s \in \mathbb{R} . \tag{4.1.24}$$ eq:TTP

$$p(t_k) = f(t_k) \quad \Leftrightarrow \quad \sum_{j=0}^{n} \gamma_j \cos\left(2\pi j \cdot \frac{2k+1}{4(n+1)}\right) = g\left(\frac{2k+1}{4(n+1)}\right) , \quad k = 0, \ldots, n .$$



$$\widetilde{g}(s) := g(s + \frac{1}{4(n+1)}) , \quad \widetilde{q}(s) := q(s + \frac{1}{4(n+1)}) \quad \Rightarrow \quad \widetilde{q}\left(\frac{k}{2(n+1)}\right) = \widetilde{g}\left(\frac{k}{2n+2}\right) , \quad k = 0, \ldots, 2n+1 .$$

$$\underbrace{\sum_{j=-n}^{n} \widetilde{\gamma}_j \exp\left(2\pi i j \cdot \frac{k}{2(n+1)}\right)}_{\in \mathcal{P}_{2n}^T} = \widetilde{g}\left(\frac{k}{2n+2}\right) , \quad k = 0, \ldots, 2n+1 ,$$

mit $\widetilde{\gamma}_j := \begin{cases} \frac{1}{2}\exp\left(2\pi i \frac{j}{4(n+1)}\right)\gamma_{|j|} & \text{, falls } j \neq 0 , \\ \gamma_0 & \text{, für } j = 0 . \end{cases}$

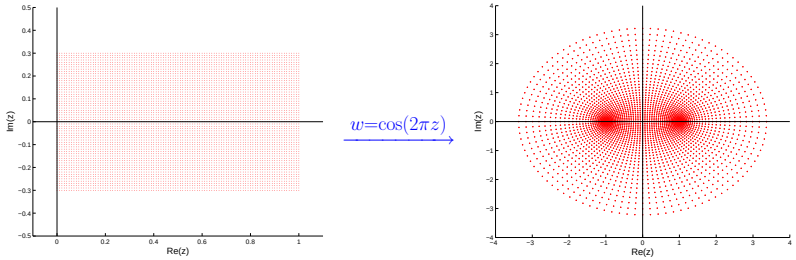
➤ Transformiertes Tschebyscheff-Interpolationspolynom $p(\cos 2\pi t)$ = trigonometrisches Interpolationspolynom zu $\widetilde{g}$! ⇒ $\|f - p\|_{L^\infty([-1,1])} = \|\widetilde{g} - \widetilde{q}\|_{L^\infty([0,1])}$

➤ Übertragung der L^∞ Fehlerabschätzungen von Thm. 4.1.15 thm:expcvg

Wo muss f analytisch sein, damit $g(w) = f(\cos 2\pi z)$ analytisch in Streifen $S := \{w \in \mathbb{C}: -\zeta^- < \mathrm{Im}\{z\} < \zeta^+\}, \zeta^-, \zeta^+ > 0$?

⇕

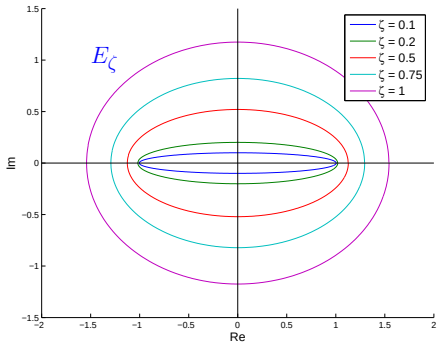
Gestalt von $\cos(S)$?



Mit $\cos(x+iy) = \cos x \cosh y - i \sin x \sinh y$: $\cos(S)$ = Inneres der Ellipse

$$E_\zeta := \{z = \cos\varphi\cosh(2\pi\zeta) - i\sin\varphi\sinh(2\pi\zeta): 0 \le \varphi \le 2\pi\}, \quad \zeta := \max\{\zeta^-, \zeta^+\} .$$

E_ζ: horizontale Halbachse $= \cosh(2\pi\zeta)$, vertikale Halbachse $= \sinh(2\pi\zeta)$.



Analyse: Tschebyscheff-Interpolation von analytischen Funktionen

1. Transfomation auf $[-1,1]$
2. Analytizitätsgebiet $A \subset \mathbb{C}$ der transfomierten Funktion
3. Maximale Ellipse $E_\zeta \subset A$ + Maximum auf Ellipse
4. Konvergenzabschätzung aus Thm. 4.1.15 thm:expcvg

4.1.5 Approximation durch Polynome

4.1.5.1 Bestapproximation

☞ Entscheidendes theoretisches Hilfsmittel:

$\mathcal{T}$ = Knotenmenge in $I \subset \mathbb{R}$, $\mathsf{I}_\mathcal{T} : C(I) \mapsto \mathcal{P}_n$ Polynominterpolationsoperator → (4.1.6) thm:polintpop

$$\mathsf{I}_\mathcal{T}(p) = p \quad \forall p \in \mathcal{P}_n \quad \Rightarrow \quad \|f - \mathsf{I}_\mathcal{T}(f)\| = \|(f-p) - \mathsf{I}_\mathcal{T}(f-p)\| \le (1 + \|\mathsf{I}_\mathcal{T}\|)\,\|f-p\| \ .$$

$$\blacktriangleright \quad \|f - \mathsf{I}_\mathcal{T}(f)\| \le (1 + \|\mathsf{I}_\mathcal{T}\|) \underbrace{\inf_{p\in\mathcal{P}_n} \|f-p\|}_{\text{Bestapproximationsfehler}} \ .$$

Speziell für $\|\cdot\| = \|\cdot\|_{L^\infty(I)}$:

$$\|f - \mathsf{I}_\mathcal{T}(f)\|_{L^\infty(I)} \le (1 + \lambda_\mathcal{T}) \inf_{p\in\mathcal{P}_n} \|f-p\|_{L^\infty(I)} \quad \text{mit Lebesgue-Konstante } \lambda_\mathcal{T} \ . \quad (4.1.25)$$ eq:lcrole

Resultat von Jackson [9, Thm. 13.3.7]: DAV75

Theorem 4.1.17 (L^∞-Bestapproximation durch Polynome)**.**

$$f \in C^r([-1,1]),\ r \ge 1: \quad \inf_{p\in\mathcal{P}_n} \|f-p\|_{L^\infty([-1,1])} \le (1+\pi^2/2)^r \frac{(n-r)!}{n!} \left\|f^{(r)}\right\|_{L^\infty([-1,1])} \ .$$

Thm. 4.1.17 thm:polbestapprox + (4.1.25) eq:lcrole + Abschätzung $\lambda_\mathcal{T} \le \frac{2}{\pi}\log(1+n) + 1$ [39, Thm. 1.2] RIV84b

➤ L^∞-Fehlerabschätzung für Tschebyscheff-Interpolation ($\mathcal{T}$ = Tschebyscheff-Knoten)

$$f \in C^r([-1,1]),\ r \ge 1: \quad \|f - \mathsf{I}_\mathcal{T}(f)\|_{L^\infty([-1,1])} \le C(r)(1+\tfrac{1}{\pi}\log n)\frac{(n-r)!}{n!}\left\|f^{(r)}\right\|_{L^\infty([-1,1])} \ ,$$

mit $C(r) := 2(1+\pi^2/2)^r$.

☛ Übertragung auf beliebige Intervalle mit Skalierungsargument.

4.1.5.2 Polynomiale Least-Squares Approximation

Gegeben: Knotenmenge $\mathcal{M} := \{\nu_0 < \nu_1 < \cdots < \nu_m\}$ ($I := [\nu_0, \nu_m]$), $y_j \in \mathbb{C}$, $j = 0,\ldots,m$

Gesucht: $p_n \in \mathcal{P}_n$ ($n < m$) mit $\displaystyle\sum_{j=0}^m |p(\nu_j) - y_j|^2 \to \min$

➡ Lineares Ausgleichsproblem → Abschnitt 3.5 sec:numer-line-ausgl

Abstrakt: Inneres Produkt auf $\mathcal{P}_m$: $\displaystyle\langle p, q\rangle := \sum_{j=0}^m p(\nu_j)\overline{q}(\nu_j)$

➢ Norm auf $\mathcal{P}_m$: $\|p\|_\mathcal{T} := \sqrt{\langle p,p\rangle}$

Ausgleichsproblem: $p \in \mathcal{P}_n$: $\|p-y\|_\mathcal{T}^2 \to \min$,

wobei etwa $y := \mathsf{I}_m(\mathbf{y})$.

Theorem 4.1.18. *Sei $(V, (\cdot,\cdot))$ ein Innenproduktraum (mit Norm $\|\cdot\|^2 := (\cdot,\cdot)$) mit Orthonormalbasis (ONB) $\{b_1,\ldots,b_m\}$, $m := \dim V \in \mathbb{N}$, und $W_n := \operatorname{Span}\{b_1,\ldots,b_n\}$, $n \le m$:*

$$\forall y \in V: \quad x^{(n)} := \arg\min_{v\in W_n} \|y - v\| \quad \Leftrightarrow \quad x^{(n)} = \sum_{j=1}^n (y, b_j)\, b_j \ .$$

Beweis: Benutze $v = \sum_{j=1}^m \xi_j b_j \quad \Rightarrow \quad \|v\|^2 = \sum_{j=1}^n |\xi|^2$

oder Normalengleichungen → Abschnitt 3.5.2. sec:normalengleichungen

Ziel: Finde $\langle\cdot,\cdot\rangle$-ONB $\{r_0, r_1, \ldots, r_m\}$ von $\mathcal{P}_m$ mit

$$\operatorname{Span}\{r_0,\ldots,r_n\} = \mathcal{P}_n\ , \quad 0 \le n \le m\ . \tag{4.1.26}$$

op:1

Idee: Gram-Schmidt-Orthonormalisierung wie im Lanczos-Prozess:
Gegeben ONB $\{r_0,\ldots,r_n\}$, $r_j \in \mathcal{P}_j$, orthogonalisiere $tr_n(t)$ gegen $\{r_0,\ldots,r_n\}$.

Krylov-Connection: $C(I) \mapsto C(I)$, $f(t) \mapsto tf(t)$ ist $\langle\cdot,\cdot\rangle$-selbstadjungierter Operator

Theorem 4.1.19 (Diskrete Orthogonalpolynome). *Definiere $r_1 := 0$, $r_0 \equiv 1$ und*

$$r_{k+1}(t) = (t-\alpha_{k+1})r_k(t) - \beta_k r_{k-1}(t)\ , \quad k = 0,1,\ldots,m-1\ ,$$
$$\text{mit} \quad \alpha_{k+1} := \frac{\langle tr_k, r_k\rangle}{\|r_k\|_{\mathcal{I}}^2}\ , \quad \beta_k := \frac{\|r_k\|_{\mathcal{I}}^2}{\|r_{k-1}\|_{\mathcal{I}}^2}\ .$$

Dann ist $\{r_0,\ldots,r_m\}$ eine $\langle\cdot,\cdot\rangle$-orthogonale Basis von $\mathcal{P}_m$ und erfüllt (4.1.26). Die r_j heissen (diskrete) Orthogonalpolynome

ORTHPOLY

op:1

Beweis: (durch Induktion)

$$\langle r_{k+1}, r_j\rangle = \langle (t-\alpha_{k+1})r_k - \beta_k r_{k-1}, r_j\rangle$$
$$= \begin{cases} \langle r_k, (t-\alpha_{k+1})r_j\rangle - \beta_k \langle r_{k-1}, r_j\rangle = 0 & \text{, falls } j < k-1\ , \\ \langle tr_k, r_{k-1}\rangle - \|r_k\|_{\mathcal{I}}^2 = 0 & \text{, falls } j = k-1\ , \\ \langle tr_k, r_k\rangle - \alpha_{k+1}\langle r_k, r_k\rangle = 0 & \text{, falls } j = k\ . \end{cases}$$

MATLAB-Code aus GGG05 [16, Sect. 4.2.5]:

Input:
ν: Vektor der m Knotenpunkte
y: Werte y_j, $j = 0,\ldots,m$
n: Maximaler Wert für k

Output:
`alpha`: Rekursionskoeffizienten α_k
`beta` : Rekursionskoeffizienten β_k
b : Koeffizienten $\langle y, r_j\rangle$, $j = 0,\ldots,n$
P : $p_{kj} = r_k(\nu_j)$

MATLAB-Code : Berechnung diskreter Orthogonalpolynome

```
function [alpha,beta,b,P]=dorthp(ν,y,n)
m = length(ν);
alpha(1) = sum(ν)/m;
p1 = ones(size(ν)); p2 = ν-alpha(1);
P = [p1;p2];
b(1) = dot(p1,y)/dot(p1,p1);
b(2) = dot(p2,y)/dot(p2,p2);
for k=1:min(n-1,m-2)
  p0 = p1; p1 = p2;
  alpha(k+1) =dot(p1,(ν.*p1))/norm(p1)^2;
  beta(k) = (norm(p1)/norm(p0))^2;
  p2 = (ν-alpha(k+1)).*p1-beta(k)*p0;
  P = [P; p2];
  b(k+2) = dot(p2,y)/norm(p2)^2;
end
```

➢ Aufwand zur Berechnung von $\alpha_k, \beta_k, \langle y, r_j\rangle$, $j = 0,\ldots,n$ = $O(nm)$

MATLAB-Code Auswertung diskreter Orthogonalpolynome

```
function[Y,Z]= evlortho(alpha,beta,b,z)
m = max(size(b));
p1 = ones(size(z));
p2 = z-alpha(1);
Z = [ p1; p2];
Y = [p1*b(1); p1*b(1)+p2*b(2)];
for i = 2:m-1
  p0 = p1;  p1 = p2;
  p2 = (z-alpha(i)).*p1...
       -beta(i-1)*p0;
  Z = [Z;p2];
  Y = [Y; Y(i,:)+p2*b(i+1)];
end
```

Auswertung von diskreten Orthogonalpolynomen: (MATLAB-Code aus GGG05 [16, Sect. 4.2.5])

Input:
☞ Ausgabe von `dorthp`
z: Auswertungsstellen

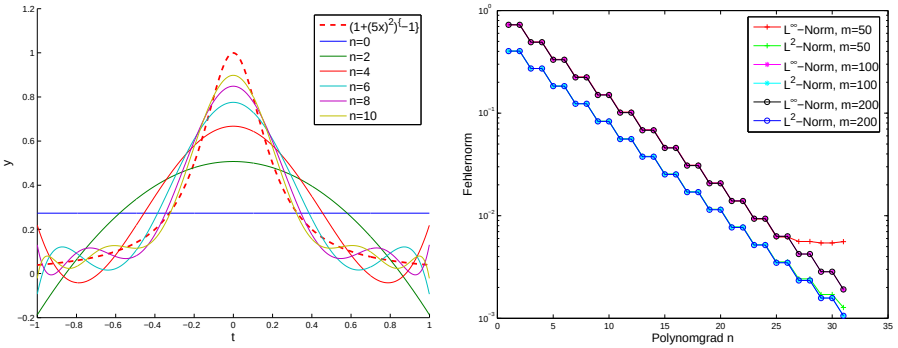
Output:
Y: Fitpolynomwerte an Stellen z
Z: Othogonalpolynomwerte an Stellen z

Beispiel 170 (Approximation durch Fitpolynome).

① $f(t) = (1+(5t)^2)^{-1}$, $I = [-1,1]$ → Bsp. ex:runge 162, analytische Funktion

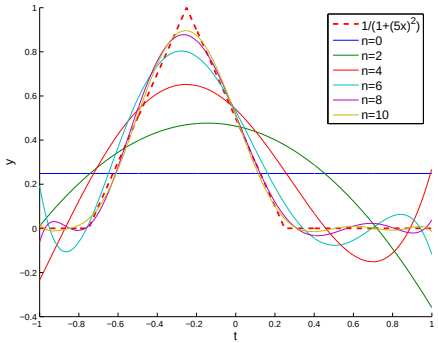
Fitten in $t_k = -1 + k\frac{2}{m}$, $k = 0,\ldots,m$, $m \in \mathbb{N}$

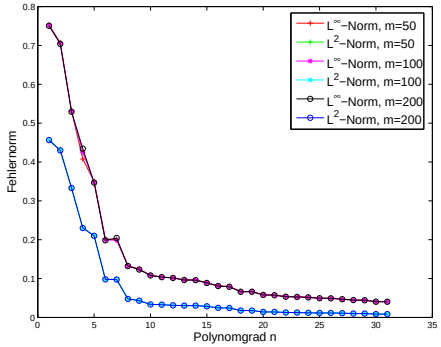
Approximative Berechnung der Fehlernormen (Sampling in $\xi_j = -1 + \frac{j}{500}$, $j = 0,\ldots,1000$)



➢ Beobachtet: Exponentielle Konvergenz (im Polynomgrad n) für $n \ll m$

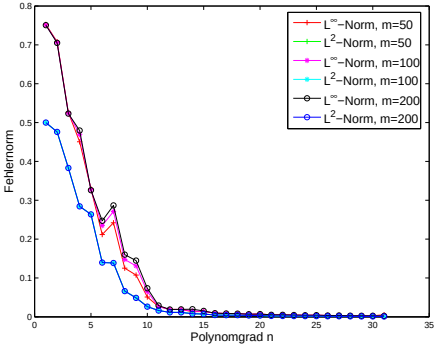
② $f(t) = \max\{0, 1-2*|x+\frac{1}{4}|\}$, f nur $C^0([-1,1])$, sonst wie oben

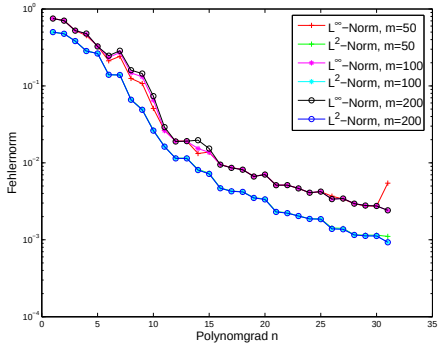




➢ Beobachtet: Nur algebraische Konvergenz (im Polynomgrad n) für $n \ll m$

③ $f(t) = \max\{\cos(4\pi|t+\frac{1}{4}|), 0\}$, nur $f \in C^1([-1,1])$, sonst wie oben





4.1.5.3 Tschebyscheff-Approximation

Gegeben: $f : I \subset \mathbb{R} \mapsto \mathbb{C}$ beschränkt, I Intervall

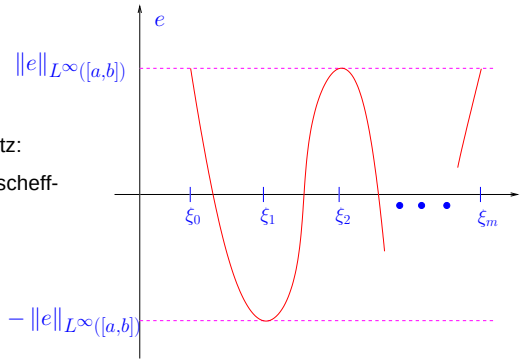
Gesucht: L^∞-bestapproximimierendes Polynom: $q_n = \operatorname{argmin}_{p\in\mathcal{P}_n} \|f-p\|_{L^\infty(I)}$

Theorem 4.1.20 (Tschebyscheffscher Alternantensatz). *Für $f \in C[a,b]$, $a<b$, $n \in \mathbb{N}$:*

$$q = \arg\min_{p\in\mathcal{P}_n} \|f-p\|_{L^\infty(I)} \quad \Leftrightarrow \quad \begin{array}{l} e(t) := f(t) - q(t) \text{ erfüllt:} \\ \exists a \le \xi_0 < \xi_1 < \cdots < \xi_m \le b\,,\ m \ge n+1\,, \\ |e(\xi_j)| = \|e\|_{L^\infty([a,b])}\,, \quad j = 0,\ldots,m\,, \\ e(\xi_{j-1}) = -e(\xi_{j+1})\,, \quad j = 1,\ldots,m\,. \end{array}$$

Veranschaulichung Alternantensatz:

Verhalten von $e \leftrightarrow$ Tschebyscheff-Polynome → Def. 4.1.6



Terminologie: Extremalstellen ξ_j der Fehlerfunktion = Alternanten

Der Remes-Algorithmus [45, Abschn. II.5]:

Darstellung von Polynomen bzgl. Basis $\{u_0,\ldots,u_n\}$ von $\mathcal{P}_n$: u_j = Monome, Tschebyscheff-Polynome,

Idee: Iterative Aktualisierung einer Alternantenmenge: $\mathcal{A}^{(0)} \to \mathcal{A}^{(1)} \to \ldots$, $\sharp\mathcal{A}^{(l)} = n+2$

① $\mathcal{A}^{(0)} := \{\xi_0^{(0)} < \xi_1^{(0)} < \cdots < \xi_n^{(0)} < \xi_{n+1}^{(0)}\} \subset [a,b]$ „willkürlich"
(z.B. Extremalstellen des Tschebyscheff-Polynoms T_{n+1}, → 4.1.6: Tschebyscheff-Alternanten)

② Falls $\mathcal{A}^{(l)} := \{\xi_0^{(l)} < \xi_1^{(l)} < \cdots < \xi_n^{(l)} < \xi_{n+1}^{(l)}\} \subset [a,b]$ exakte Alternantenmenge (→ Thm. 4.1.20):

$$\sum_{j=0}^{n} \alpha_j u_j(\xi_k^{(l)}) + (-1)^k \delta = f(\xi_k^{(l)})\,, \quad k = 0,\ldots n+1\,, \qquad (4.1.27)$$

liefert L^∞-bestapproximierendes Polynom $q = \sum_{j=0}^{n} \alpha_j u_j$, $\delta = \pm\|f-q\|_{L^\infty([a,b])}$.

➢ Löse (4.1.27) mit $\mathcal{A}^{(l)}$. Wenn $||\delta| - \|f-q\|_{L^\infty([a,b])}| \le$ TOL STOP

③ $\mathcal{A}^{(l+1)}$ = Extremalpunkte von $f - q$ (zu finden durch Newton/Sekanten-Verfahren, Sampling). $l \leftarrow l+1$ und GOTO ②.

Alternativ: $\mathcal{A}^{(l+1)} := \mathcal{A}^{(l)} \cup \{\text{Maximalstelle von } |f-q|\} \setminus \{\text{Minimalstelle von } |f-q|\}$
(Austausch → robusterer Algorithmus)

MATLAB-Code: Remes-Algorithmus zur Tschebyscheff-Approximation auf `[a,b]`:

`f`: Handle auf $f : [a,b] \mapsto \mathbb{R}$
`n`: Polynomgrad
`N`: Sampling-Auflösung

Tschebyscheff-Alternanten

Vandermonde-Matrix:

Die ersten $n+1$ Spalten des LGS (eq:ALTSYS 4.1.27)

Suche: Maximum der Fehlerfunktion (Sampling-Technik)

MATLAB-CODE : Remes-Algorithmus

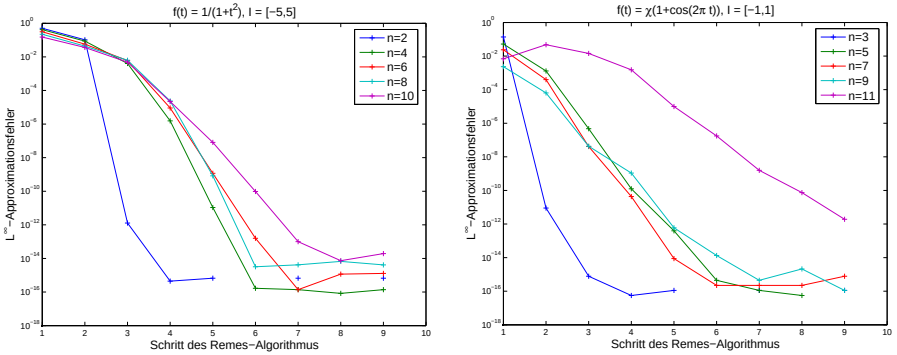
```
function dev = remes(f,a,b,n,N,tol)
t = [a:(b-a)/N:b]'; y = feval(f,t);
xe=(a+b)/2+(a-b)/2*cos(pi/(n+1)*[0:d+1]');
fxe=feval(f,xe);
for k=1:maxit
  V=vander(xe);
  A=[V(:,2:d+2),(-1).^[0:d+1]'];
  c=A\fxe;
  delta = c(d+2);

  err = polyval(c(1:d+1),t) - y;
  delta = diff(err);
  s=[delta(1:n-1)].*[delta(2:n)];
  ind=find(s<0)+1;
  xe= [a;ind;b];
  fxe=feval(f,xe);
  del = polyval(c(1:d+1),xe) - fxe;
  dev = max(abs(del));
  if (abs(dev-abs(e)) < tol), break; end
end;
```

```
nx = length(ind); xe = x(ind);
if (nx == d), xe=[a;xe;b];
elseif (nx == d+1)
  xmin = min(xe); xmax =
max(xe);
  if ((xmin-a) > (b-xmax))
    xe = [a;xe];
  else xe = [xe;b];
end
```

Beispiel 171 (Konvergenz des Remes-Algorithmus).

- $f(t) = (1+t^2)^{-1}$, $I = [-5,5]$ → Bsp. ex:runge 162
- $f(t) = \begin{cases} \frac{1}{2}(1+\cos(2\pi t)) & \text{, falls } |t| < \frac{1}{2} \\ 0 & \text{sonst.} \end{cases}$, $I = [-1,1]$

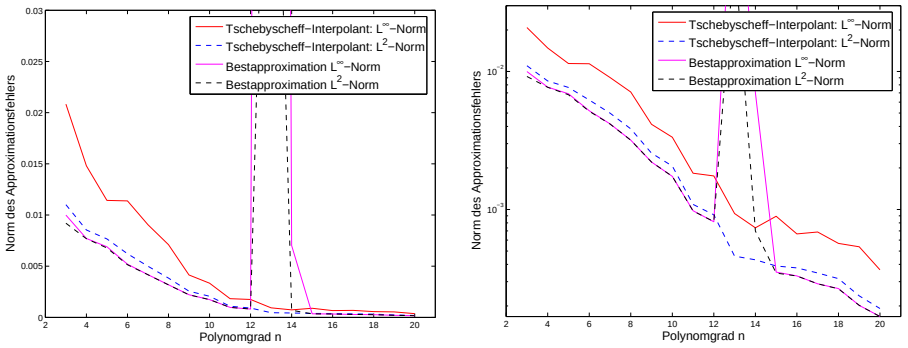


Beispiel 172 (Tschebyscheff-Interpolation vs. Bestapproximation).

$$f(t) = \begin{cases} \exp\left(\frac{1}{(t-0.1)^2 - \frac{1}{4}}\right) & \text{, falls } -0.4 < t < 0.6 \text{ ,} \\ 0 & \text{sonst. , } \end{cases} \quad I = [-1,1] \text{ .}$$

$f \in C^\infty([-1,1])$, aber f **nicht** analytisch !

Approximative Berechnung der Fehlernormen für verschiedene Polynomgrade:



Beobachtungen: Indizien für Exponentielle Konvergenz
$n = 13$: Divergenz des Remes-Algorithmus **!?**

4.1.6 Clusteringapproximation

CLUSTAPPROX

Aufgabenstellung:

KERNFN

Gegeben: Kernfunktion $G : I \times J \mapsto \mathbb{C}$, $I, J \subset \mathbb{R}$ Intervalle: $G(x,y)$ glatt für $x \neq y$

Kollokationspunkte $x_1 < x_2 < \cdots < x_n$, $x_j \in I$, $y_1 < y_2 < \cdots < y_m$, $y_j \in J$

KOLLMAT

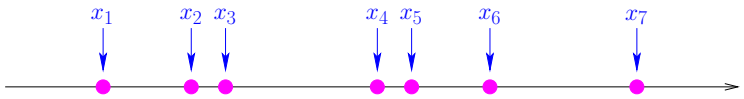
Kollokationsmatrix: $\mathbf{M} \in \mathbb{C}^{n,m} \quad :\Leftrightarrow \quad (\mathbf{M})_{ij} := G(x_i, y_j)$, $1 \leq i \leq n$, $1 \leq j \leq m$.

(4.1.28) eq:Mdef

Gesucht: *Effizienter* Algorithmus für approximative Auswertung $\mathbf{M}\times$Vektor

(Naiv: Rechenaufwand $O(mn)$!)

Beispiel 173 (Interaktionsberechnung für Vielteilchensysteme).



n parallele stromdurchflossene Drähte, Draht $j \rightarrow$ Strom c_j, Position $x_j \in \mathbb{R}$, $c_j \in \mathbb{R}$

Zu berechnen: Magnetische Kraft auf *alle* Drähte

Kraft auf Draht j: $f_j = \sum_{\substack{k=1 \\ k\neq j}}^{n} \frac{1}{|x_j - x_k|} c_k c_j$, $j = 1, \ldots, n$.

$$\mathbf{f} = \operatorname{diag}(c_1, \ldots, c_n)\mathbf{M} \begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix} \quad , \quad \mathbf{M} = (m_{ij})_{i,j=1}^{n} , \quad m_{ij} = \begin{cases} \frac{1}{|x_j - x_i|} & \text{. falls } i \neq j , \\ 0 & \text{für } i = j . \end{cases}$$

➢ Kollokationsmatrix $\mathbf{M}$ wird erzeugt durch Kernfunktion $G(x,y) = \frac{1}{|x-y|}$ ◇

Beispiel 174 (Gravitationskräfte in Galaxien).

Galaxie: n ($\approx 10^9$) Sterne mit Positionen $\boldsymbol{x}_i \in \mathbb{R}^3$ und Massen m_i, $i = 1, \ldots, n$.

Erforderlich für Simulation der Galaxiendynamik:

☞ Gravitationskraft auf *alle* Sterne !

$$f_j = \frac{G}{4\pi} \sum_{i \neq j} \frac{1}{\|\mathbf{x}_i - \mathbf{y}_j\|} m_i m_j , \quad j \in \{1, \ldots, n\} .$$

$$\mathbf{f} = \operatorname{diag}(m_1, \ldots, m_n)\mathbf{M} \begin{pmatrix} m_1 \\ \vdots \\ m_n \end{pmatrix} \quad , \quad m_{ij} := \begin{cases} \frac{G}{4\pi} \frac{1}{\|\mathbf{x}_i - \mathbf{y}_j\|} & \text{for } i \neq j , \\ 0 & \text{for } i = j , \end{cases} \quad 1 \leq i, j \leq n .$$

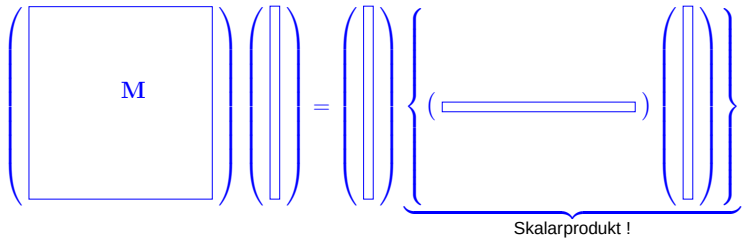
▷ dreidimensionale Verallgemeinerung der Aufgabenstellung ◇

4.1.6.1 Separierte Kernapproximation

Beobachtung: Falls Kernfunktion separiert: $G(x,y) = g(x)h(y)$, $g : I \mapsto \mathbb{C}$, $h : J \mapsto \mathbb{C}$

$$\mathbf{M} = \left(g(x_j)\right)_{j=1}^{n} \cdot \left(\left(h(y_j)\right)_{j=1}^{m}\right)^T \quad \blacktriangleright \quad \operatorname{rank}(\mathbf{M}) = 1 .$$

$\blacktriangleright$ Rechenaufwand($\mathbf{M}\times$Vektor) = $m + n$

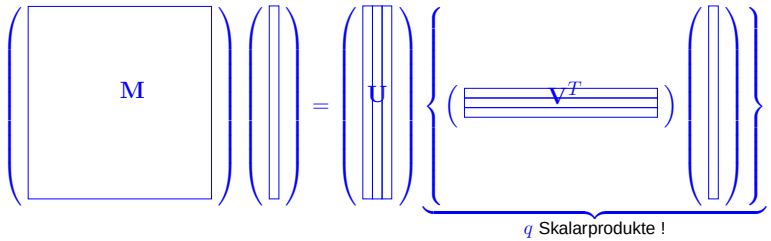


Verallgemeinerung: $G(x,y) = \sum_{j=1}^{q} g_j(x) h_j(y)$, $g_j : I \mapsto \mathbb{C}$, $h_j : J \mapsto \mathbb{C}$, $q \in \mathbb{N}$

$$\mathbf{M} = \mathbf{U}\mathbf{V}^T , \quad \begin{array}{l} \mathbf{U} \in \mathbb{R}^{n,q} , \quad u_{ij} = g_j(x_i) , \quad j \in \{1, \ldots, q\}, i \in \{1, \ldots, n\} , \\ \mathbf{V} \in \mathbb{R}^{q,m} , \quad v_{ij} = h_i(y_j) , \quad i \in \{1, \ldots, q\}, j \in \{1, \ldots, m\} . \end{array}$$

$\operatorname{rank}(\mathbf{M}) = q$

Rechenaufwand($\mathbf{M}\times$ Vektor) $= q(m+n)$



Idee: Globale Approximation von G durch (Summe) separierter Kernfunktionen:

$$G(x,y) \approx \widetilde{G}(x,y) := \sum_{l=0}^{d}\sum_{k=0}^{d} \kappa_{l,k} g_l(x) h_k(y) \;, \quad \kappa_{l,j} \in \mathbb{C} \;, \quad (x,y) \in I \times J \;. \tag{4.1.29}$$

CA:1

$\longleftrightarrow$ Ersetze $\mathbf{M} \to \widetilde{\mathbf{M}} = (\widetilde{m}_{ij})$, $\widetilde{m}_{ij} = \widetilde{G}(x_i, y_j)$

Bemerkung 175 (Qualitätsmass für Kernapproximation).

$$\left\|\mathbf{M}-\widetilde{\mathbf{M}}\right\|_2^2 \le \left\|\mathbf{M}-\widetilde{\mathbf{M}}\right\|_F^2 = \sum_{i,j}(m_{ij}-\widetilde{m}_{ij})^2 = \boxed{\sum_{i,j}(G(x_i,y_j)-\widetilde{G}(x_i,y_j))^2} \;.$$

QUALMEAS

Normalisiertes Qualitätsmass: $\dfrac{1}{mn}\sum_{i,j}(G(x_i,y_j)-\widetilde{G}(x_i,y_j))^2$. (4.1.30)

CA:qualn

△

Wie bekommt man eine separierte Kernapproximation **?**

Definition 4.1.21 (Tensorprodukt-Polynominterpolation). $L_j^x \in \mathcal{P}_n$ ($L_k^y \in \mathcal{P}_m$), $j = 0,\dots,n$ ($k = 0,\dots,m$), Lagrange-Polynome zu Knotenmenge $\mathcal{X} := \{x_j\}_{j=0}^n \subset I$ ($\mathcal{Y} := \{y_k\}_{k=0}^m \subset J$), $I, J \subset \mathbb{R}$ Intervalle. Zu stetigem $f : I \times J \mapsto \mathbb{C}$ definiert

$$(\mathrm{I}_{\mathcal{X}\times\mathcal{Y}} f)(x,y) := \sum_{j=0}^{n}\sum_{k=0}^{m} f(x_j,y_k) L_j^x(x) L_k^y(y) \;, \quad x,y \in \mathbb{R} \;,$$

das Tensorprodukt-Interpolationspolynom.

Abschnitt 4.1.3: Benutze Tensorprodukt-Tschebyscheff-Polynominterpolation

$$\widetilde{G}(x,y) := \sum_{j=0}^{d}\sum_{k=0}^{d} G(t_j^x, t_k^y) L_j^x(x) L_k^y(y) \;, \tag{4.1.31}$$

CA:2

$t_0^x,\dots,t_d^x / t_0^y,\dots,t_d^y$ Tschebyscheff-Knoten in I/J,
L_j^x, L_k^y zugeordnete Lagrange-Polynome.

Beispiel 176 (Globale separierbare Approximation bei glatter Kernfunktion).

- Glatte (analytische !) Kernfunktion $G(x,y) = \dfrac{1}{1+|x-y|^2}$, Kollokationspunkte $x_i = y_i = \frac{i-1}{n}$, $i = 1,\dots,n$.
- $\widetilde{G}$ durch Tensorprodukt-Tschebyscheff-Polynominterpolation (178) vom Grad $d \in \mathbb{N}$. ▷

CA:2

$$\frac{1}{n^2}\sum_{i,j}(G(x_i,y_j)-\widetilde{G}(x_i,y_j))^2$$

als Funktion von d für $n \in \{100, 200, 400\}$

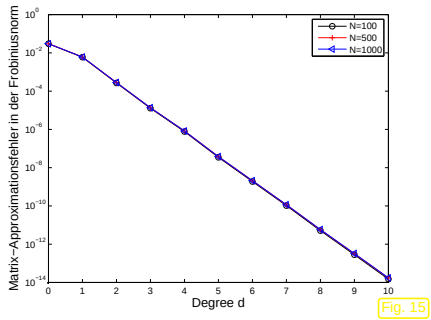


Fig. 15

smoothkernel

Beobachtet: Exponentielle Konvergenz $\left\|\mathbf{M}-\widetilde{\mathbf{M}}\right\|_F \to 0$ in Abhängigkeit von d ◇

Beispiel 177 (Globale separierbare Approximation bei nichtglatter Kernfunktion).

Auswertungen wie in Bsp. 176, nun für Kernfunktion

$$G(x,y) = \begin{cases} \frac{1}{|x-y|} & \text{für } x \neq y \;, \\ 0 & \text{sonst,} \end{cases} \tag{4.1.32}$$

ex:globsepappr

vgl. Bsp. 173.

ex:wires

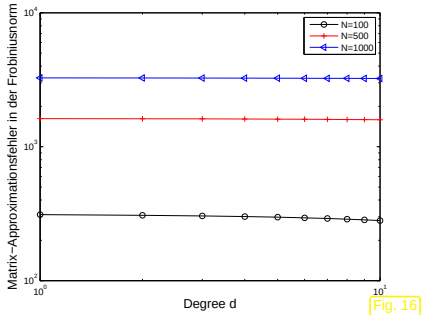


Fig. 16

nonsmoothkernel

Beobachtet: (Praktisch) keine Konvergenz $\left\|\mathbf{M}-\widetilde{\mathbf{M}}\right\|_F \to 0$ für $d \to \infty$ ◇

Schuldiger: Fehlende *globale* Glattheit

➢ Schlechte Approximierbarkeit von $G(x,y) := |x-y|^{-1}$ in Umgebung von $\{(x,y) \in I \times J : x = y\}$.

JEDOCH $G(x,y)$ aus (4.1.32) ist glatt (sogar analytisch → Def. 4.1.14) in "grösserer Entfernung" von $\{(x,y) \in I \times J : x = y\}$.

Idee: Lokale Approximation von G durch (Summe) separierter Kernfunktionen:

$$G(x,y) \approx \sum_{l=0}^{d}\sum_{k=0}^{d} \kappa_{l,k} g_l(x) h_k(y) \ , \ \kappa_{l,j} \in \mathbb{C} \ , \quad (x,y) \in \widetilde{I} \times \widetilde{Y}, \widetilde{I} \subset I, \widetilde{J} \subset J \ , \tag{4.1.33}$$

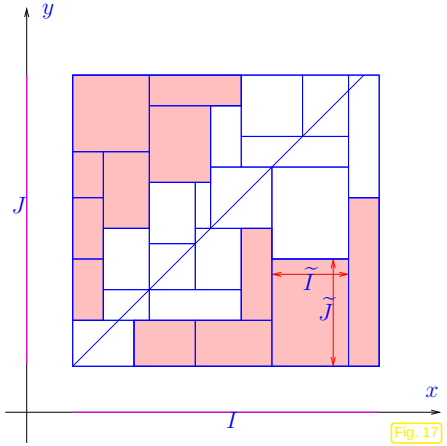
mit $\widetilde{I} \times \widetilde{J} \cap \{(x,y) : x = y\} = \emptyset$.

Konkret:

Lokale Approxmation von G auf Rechtecken $\widetilde{I} \times \widetilde{J}$ einer *Partitionierung* von $I \times J$

Möglichkeit der separierten Kernapproximation durch lokale Tensorprodukt-Tschebyscheff-Polynominterpolation (4.1.33)

(zulässige Rechtecke)



capart

Fig. 17

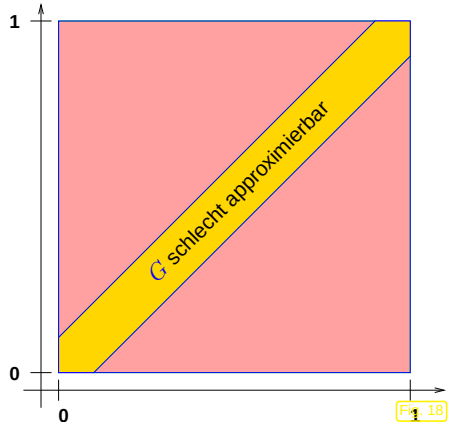


Fig. 18

Terminologie:

NEARF
Nahfeld: G nicht approximierbar

FARF
Fernfeld: G gut approximierbar

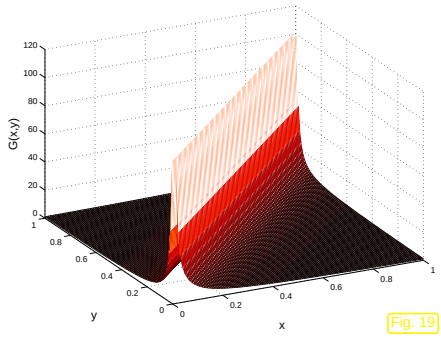


Fig. 19

Kernfunktion (4.1.32) auf $[0,1]^2$

$$G(x,y) = \frac{1}{|x-y|}$$

➢ singulär für $x = y$

➢ Analytisch fern von $\{(x,y) : x = y\}$: Analytizitätsellipsen (→ Abschn. 4.1.4) wachsen mit $|x-y|$

Ähnliche Kernfunktionen: $G(x,y) = \log|x-y|$, $G(x,y) = \frac{\partial}{\partial x}\frac{1}{|x-y|}$, etc.

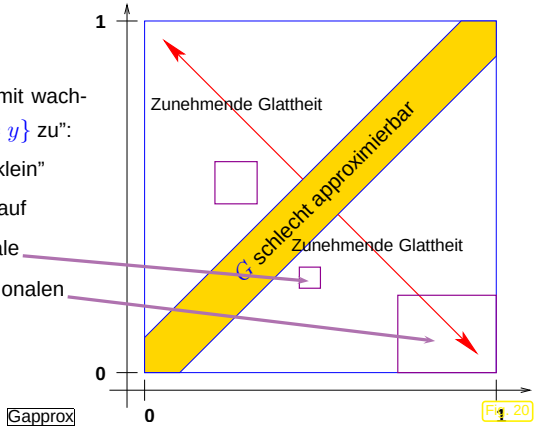
„‘Glattheit von $G(x,y) = \frac{1}{|x-y|}$ nimmt mit wachsender Entfernung von Diagonalen $\{x=y\}$ zu”:

➤ Keine Approximation, wenn $|x-y|$ „klein”

Tensorprodukt-Polynominterpolation auf

- kleinen Rechtecken nahe Diagonale
- grossen Rechtecken fern der Diagonalen

Wahl geeigneter Rechtecke **?**



Gapprox Fig. 20

Beispiel 178 (Tensorprodukt-Tschebyscheff-Interpolation auf Rechtecken).

$I = J = [0,1]^2$, $G(x,y) = |x-y|^{-1}$ from (4.1.32), (Genäherte) Supremumsnorm des Fehlers der lokalen Tensorprodukt-Tschebyscheff-Polynominterpolation von G auf

Rechtecken konstanter Grösse, aber mit wachsendem Abstand von $\{(x,y): x=y\}$.

$$\widetilde{I}_k = [0.55 + k\cdot 0.05, 0.75 + k\cdot 0.05] \quad , \quad \widetilde{J}_k = [0.25 - k\cdot 0.05, 0.45 - k\cdot 0.05] \, , \quad k \in \{0,\ldots,5\} \, .$$

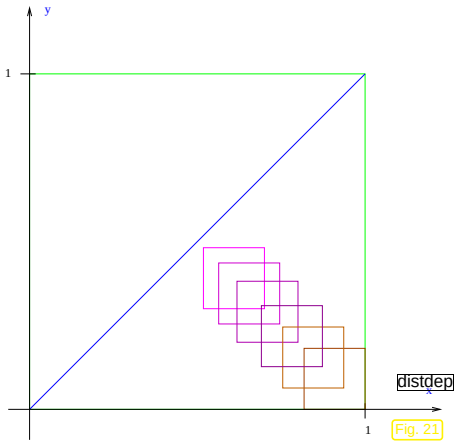


Fig. 21

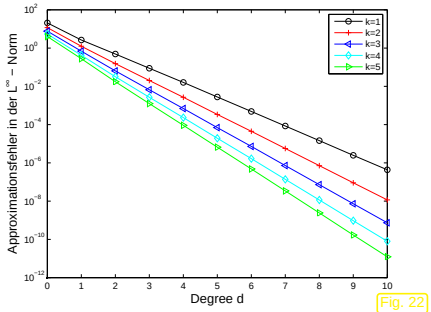


Fig. 22

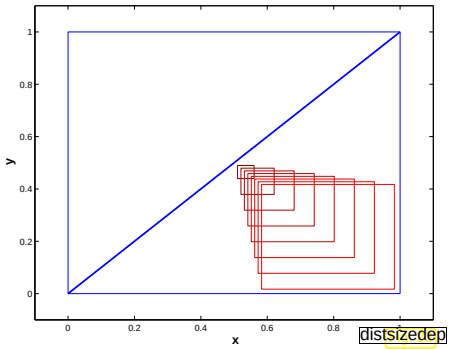
☞ Abnahme des Interpolationsfehlers mit zunehmendem Abstand von $\{(x,y): x=y\}$ ◇

Beispiel 179 (Tensorprodukt-Tschebyscheff-Interpolation auf variablen Rechtecken).

siehe ➔ Bsp. 178: nun Supremumsnorm des Interpolationsfehlers auf Rechtecken

$$[\tfrac{1}{2}(\sqrt{2}-1)\xi + \tfrac{1}{2}, \tfrac{1}{2}(\sqrt{2}+1)\xi + \tfrac{1}{2}] \times [-\tfrac{1}{2}(\sqrt{2}-1)\xi + \tfrac{1}{2}, -\tfrac{1}{2}(\sqrt{2}+1)\xi + \tfrac{1}{2}] \, , \quad 0.05 \le \xi \le \frac{1}{1+\sqrt{2}} \, .$$

➢ Grösse der Rechecke $\sim$ Abstand von Diagonalen



distsizedep

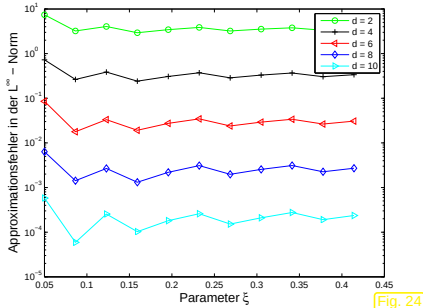


Fig. 24

☞ Abnahme des Interpolationsfehlers trotz Zunahme der Rechtecksgrösse ◇

ZULB

Zulässigkeitsbedingung:

$[a,b]\times[c,d]$ heisst η-zulässig (engl. *admissible*), $\eta > 0$, falls

Abstand von Diagonalen

Grösse des Rechtecks

$$\eta \operatorname{dist}([a,b],[c,d]) \ge \max\{b-a, d-c\} \, . \qquad \text{(4.1.34)}$$ CA:adm

$(\operatorname{dist}([a,b],[c,d]) := \min\{|x-y|: x\in[a,b], y\in[c,d]\})$

4.1.6.2 Clustertechnik

Aufgabe: Gegeben $\eta > 0$, finde (➔ schneller Algorithmus) Partitionierung von $I\times J$ „fern der Diagonalen” in η-zulässige Rechtecke.

Bemerkung 180. Perspektiven:

Partitionierung von $I\times J$ $\Longrightarrow$ Partitionierung von $\{x_i\}_{i=1}^n \times \{y_j\}_{j=1}^m$ $\Leftrightarrow$ Partitionierung der Indexmenge $\{1,\ldots,n\}\times\{1,\ldots,m\}$

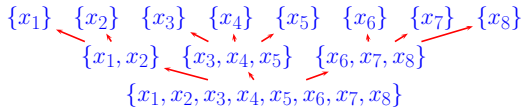
△

➤ *Clustertechnik*:

Definition 4.1.22 (Clusterbaum). *Ein Baum (→ Informatik,Graphentheorie) T heisst Clusterbaum (engl. cluster tree) zu $\mathbb{P} := \{x_1,\ldots,x_n\} \subset \mathbb{R}$:⇔*

- *Knoten von T (= Cluster) sind Teilmengen von $\mathbb{P}$.*
- $\operatorname{root}(T) = \mathbb{P}$.
- *Für alle Knoten σ von T: $\{\sigma': \sigma' \in \operatorname{sons}(\sigma)\}$ ist Partitionierung von σ.*

Bounding Box von Cluster $\sigma \in T$: $\Gamma(\sigma) := [\min\{x\}_{x\in\sigma}, \max\{x\}_{x\in\sigma}]$



← Bätter (leaves)

← sons(Wurzel)

← Wurzel (root)

Terminologie (→ Graphentheorie): Clusterbaum T: $\sigma \in T$: $\mathrm{sons}(\sigma) = \emptyset$:⇔ σ Blatt

Clusterbaum T:
Rekursive Konstruktion

MATLAB-Datenstruktur:

$N \times 6$-Matrix, $N = \sharp T$,
Zeilen $\hat{=}$ Cluster

`T = [T;i,j,xl,xr,s1,s2]`

`i, j` : $\sigma = \{x_i, \ldots, x_j\}$
`xl` : $\mathrm{xl} = \min_{i \le k \le j} x_k$
`xr` : $\mathrm{xr} = \max_{i \le k \le j} x_k$
`s1, s2`: Zeilenindices der Söhne

MATLAB-Code : Konstruktion eines Clusterbaumes

```
function [T,idx] = ct_rec(x,ofs,m,T)
N = length(x);
if (N <= m)
  T = [T;ofs,ofs+N-1,x(1),x(end),0,0];
else
  n = round(N/2);
  [T,s1] = ct_rec(x(1:n),ofs,m,T);
  [T,s2] = ct_rec(x(n+1:N),ofs+n,m,T);
  T = [T;ofs,ofs+N-1,x(1),x(end),s1,s2];
end
idx = size(T,1);
```

$\sharp T \le 2n$

Beispiel 181 (Clusterbaum).

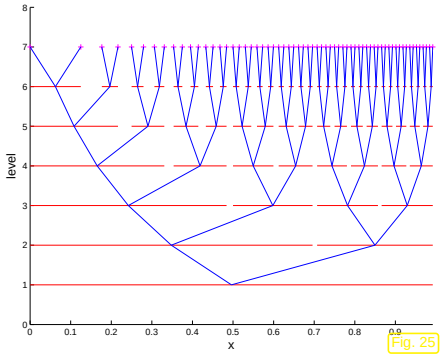


Fig. 25

← Binärer Clusterbaum:

```
x = sqrt(0:1/64:(1-1/64));
```

➢ 2^{l-1} Cluster auf Level l mit je 2^{7-l} Punkten.
(ausgewogener Clusterbaum)

◇

Idee: Wähle Approximationsrechtecke = Tensorprodukte von Bounding Boxen von Clustern $\in T$

Respektiere Zulässigkeitsbedingung (4.1.34) CA:adm

Cluster σ, μ zulässig :⇔ $\Gamma(\sigma), \Gamma(\mu)$ zulässig.

Rekursiver Aufbau einer zulässigen Partitionierung:

`Pnear` = Nahfeld:

Zone nahe der Diagonalen (→ keine Approximation)

`Pfar` = Fernfeld:

Partitioniert mit η-zulässigen Rechtecken

MATLAB-Code : Rekursiver Aufbau einer zulässigen Partitionierung

```
function [Pnear,Pfar] = ...
         divide(Tx,Ty,σ,μ,Pnear,Pfar,η)
cls = Tx(σ,:); clm = Ty(μ,:);
if (σ = Blatt | μ = Blatt)
  Pnear = [Pnear; cls(2),cls(3),clm(2),clm(3)];
elseif ((σ,μ) zulässig)
  Pfar = [Pfar; cls(2),cls(3), clm(2),clm(3)];
else
  for s1 = cls(6:7), for s2 = clm(6:7)
    [Pnear,Pfar] = divide(Tx,Ty,s1,s2,Pnear,Pfar,η);
  end; end
end
```

MATLAB-Code : Aufruf

```
function [Pnear,Pfar] = partition(Tx,Ty,η)
Pnear = []; Pfar = [];
σ = find(Tx(:,1) == min(Tx(:,1)));
μ = find(Ty(:,1) == min(Ty(:,1)));
[Pnear,Pfar] = divide(Tx,Ty,σ,μ,Pnear,Pfar,η);
```


MATLAB Animation zum Aufbau einer Partitionierung in File
./MATLAB/Kapitel_4_ApproximationInterpolation/Clustering/clusteringanimation.m
Aufruf: clusteringapproximation(N,waittime)
N: Anzahl Kollokationspunkte in einer Dimension, waittime: Zeit (sec) zwischen 2 Animationsschritten

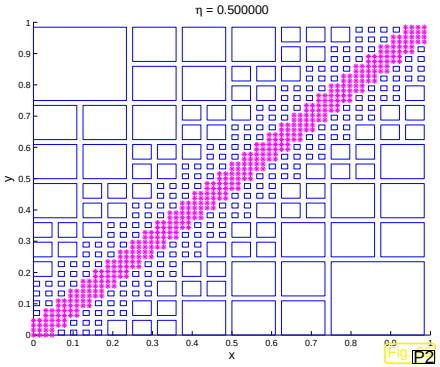


Fig. 26

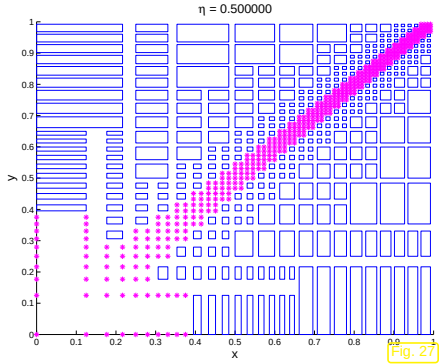


Fig. 27

```
x = (0:1/64:(1-1/64));
```

```
x = sqrt(0:1/64:(1-1/64));
```

Punktmenge(n): $*$ = Nahfeldpunktpaare , $\square$ = Partitionsrechtecke (Fernfeld)

! Achsenabschnitte der Partitionsrechtecke $\in \{\Gamma(\sigma): \sigma \in T\}$, $T \mathrel{\hat{=}}$ Clusterbaum (→ Def. def:CT 4.1.22)

4.1.6.3 Multipol-Matrixmultiplikation

Lokale Kernapproximation auf Fernfeld-Rechtecken durch Tensorprodukt-Tschebyscheff-Polynominterpolation	◁▷	LOWRANKAPPROX Niedrigrang-Approximation von *Blöcken* von $\mathbf{M}$ aus (eq:Mdef 4.1.28) siehe Figs. P1P2 26,27

✎ Notation:

$$\sigma := \{i_1, \ldots, i_2\} \subset \{1, \ldots, n\} \\ \mu := \{j_1, \ldots, j_2\} \subset \{1, \ldots, m\} \quad \Rightarrow \quad \mathbf{M}_{|\sigma\times\mu} := (m_{ij})_{\substack{i=i_1,\ldots,i_2 \\ j=j_1,\ldots,j_2}} .$$

Betrachte Kernapproximation auf $[x_{i_1}, x_{i_2}] \times [y_{j_1}, y_{j_2}]$ durch Tensorprodukt-Tschebyscheff-Polynominterpolation (→ siehe (CA:2 178)):

$$G(x,y) \approx \widetilde{G}(x,y) := \sum_{j=0}^{d}\sum_{k=0}^{d} G(t_j^x, t_k^y) L_j^x(x) L_k^y(y) ,$$

- $t_0^x, \ldots, t_d^x / t_0^y, \ldots, t_d^y \mathrel{\hat{=}}$ Tschebyscheff-Knoten auf $[x_{i_1}, x_{i_2}] / [y_{j_1}, y_{j_2}]$,
- L_j^x, $L_k^y \mathrel{\hat{=}}$ zugeordnete Lagrange-Polynome.

$$\widetilde{G}(x,y) = \sum_{j=0}^{d}\sum_{k=0}^{d} G(t_j^x, t_k^y) L_j^x(x) L_k^y(y) , \quad x \in [x_{i_1}, x_{i_2}],\ y \in [y_{j_1}, y_{j_2}] .$$

► $$\widetilde{\mathbf{M}}_{|\sigma\times\mu} = \underbrace{\underbrace{\left(L_j^x(x_i)\right)_{\substack{i=i_1,\ldots,i_2 \\ j=0,\ldots,d}}}_{\in\mathbb{R}^{\sharp\sigma,d+1}} \underbrace{\left(G(t_j^x, t_k^y)\right)_{j,k=0,\ldots,d}}_{\in\mathbb{R}^{d+1,d+1}} \underbrace{\left(L_k^y(y_j)\right)_{\substack{k=0,\ldots,d \\ j=j_1,\ldots,j_2}}}_{\in\mathbb{R}^{d+1,\sharp\mu}}}_{\text{rank}\le d+1} .$$

$$\widetilde{\mathbf{M}}_{|\sigma\times\mu} = \left(\begin{array}{|c|}\hline\ \ \\ \hline\end{array}\right) \left(\begin{array}{|c|}\hline\ \ \\ \hline\end{array}\right) \left(\begin{array}{|c|}\hline\ \ \\ \hline\end{array}\right) .$$

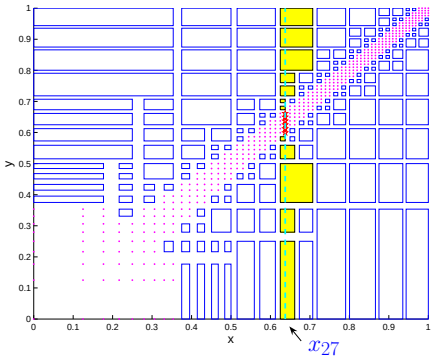
Anwendung auf Partitionierung

$$\{x_1, \ldots, x_n\} \times \{y_1, \ldots, y_m\} = \left(\bigcup_{(\sigma,\mu)\in P^{\text{near}}} \sigma\times\mu\right) \cup \left(\bigcup_{(\sigma,\mu)\in P^{\text{far}}} \sigma\times\mu\right) .$$

Algorithmus: Auswertung $\mathbf{f} = \mathbf{Mc}$, $\mathbf{M}$ aus (eq:Mdef 4.1.28)

$$f_i = \sum_{j=1}^{m} G(x_i, y_j) c_j = \underbrace{\sum_{j\in P^{\text{near}}(i)} G(x_i, y_j) c_j}_{\text{Nahfeldbeiträge}} + \underbrace{\sum_{\substack{\sigma\in T_x \\ x_i\in\sigma}} \sum_{\substack{\mu\in T_y \\ (\sigma,\mu)\in P^{\text{far}}}} \sum_{\substack{1\le j\le m \\ y_j\in\mu}} G(x_i, y_j) c_j}_{\text{Fernfeldbeiträge}} .$$

Nahfeldkopplungsindices $P^{\text{near}}(i) := \{j \in \{1, \ldots, m\}: (\{x_i\} \times \{y_j\}) \in P^{\text{near}}\}$



Veranschaulichung:

`x = sqrt(0:1/64:1);`, $i = 27$, $\eta = 1$:

⟶ Fernfeldclusterpaare mit Beiträgen zu f_i

☆⟶ Nahfeldpunktpaare mit Beiträgen zu f_i

$$\approx \sum_{j\in P^{\text{near}}(i)} G(x_i, y_j) c_j + \sum_{\substack{\sigma\in T_x \\ x_i\in\sigma}} \sum_{\substack{\mu\in T_y \\ (\sigma,\mu)\in P^{\text{far}}}} \sum_{\substack{1\le j\le m \\ y_j\in\mu}} \sum_{l=0}^{d}\sum_{k=0}^{d} G(t_l^\sigma, t_k^\mu) L_l^\sigma(x_i) L_k^\mu(y_j) c_j$$

$$\approx \sum_{j\in P^{\text{near}}(i)} G(x_i, y_j) c_j + \sum_{\substack{\sigma\in T_x \\ x_i\in\sigma}} \sum_{\substack{\mu\in T_y \\ (\sigma,\mu)\in P^{\text{far}}}} (\mathbf{V}_\sigma \mathbf{X}_{\sigma,\mu} \mathbf{V}_\mu^T \mathbf{c}_{|\mu})_i ,$$

mit

$$t_l^\sigma, t_k^\mu := \text{Tschebyscheff-Knoten (eq:CHEBNODES 4.1.17) in } \Gamma(\sigma), \Gamma(\mu) , \quad l = 0, \ldots, d ,$$

$$\mathbf{X}_{\sigma,\mu} := \left(G(t_l^\sigma, t_k^\mu)\right)_{l,k=0}^{d} \in \mathbb{R}^{d+1,d+1} , \tag{4.1.35}$$

$$\mathbf{V}_\sigma := \left(L_l^\sigma(x_i)\right)_{\substack{i:\, x_i\in\sigma \\ l=0,\ldots,d}} \in \mathbb{R}^{\sharp\sigma,d+1} , \tag{4.1.36}$$

$$\mathbf{V}_\mu := \left(L_k^\mu(y_j)\right)_{\substack{j:\, y_j\in\mu \\ k=0,\ldots,d}} \in \mathbb{R}^{\sharp\mu,d+1} , \tag{4.1.37}$$

Komplexitätsanalyse (unter der Annahme $n \approx m$):

①: Trick: Berechnung „auf Vorrat"

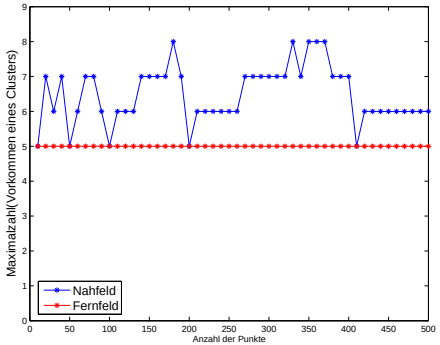
Berechne $\mathbf{w}_\mu := \mathbf{V}_\mu^T \mathbf{c}_{|\mu} \in \mathbb{R}^{d+1}$, $\mu \in T_y$ → $O(n)$ Operationen auf jedem Level von T_y

► Rechenaufwand $O((d+1)n\log_2 n)$, Speicheraufwand $O((d+1)n\log_2 n)$

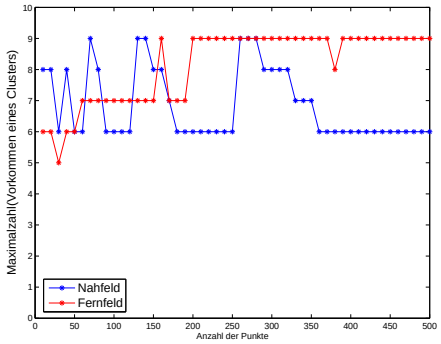
②: Berechne $\mathbf{X}_{\sigma,\mu}$ aus (4.1.35), $\sigma \in T_x$, $\mu \in T_y$, $(\sigma,\mu) \in P^{\mathrm{far}}$.

Regelmässige Punktverteilung ➤ $\max\{\sharp\{\mu \in T_y\colon (\sigma,\mu) \in P^{\mathrm{far}}\}, \sigma \in T_x\} = O(1)$

Beispiel 182 (Vorkommen von Clustern in Partitionsrechecken).



`x = (0:1/n:(1-1/n))`, $\eta = 1$

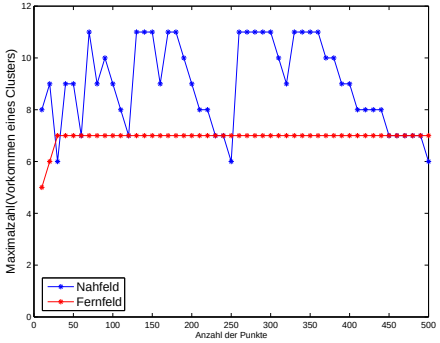


`x = sqrt(0:1/n:(1-1/n))`, $\eta = 1$

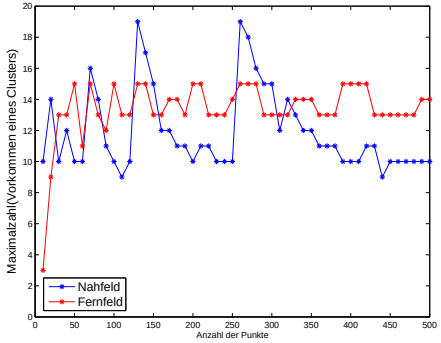
Bei „gleichmässiger" Verteilung der x_i, y_j: Für alle $\sigma \in T_x$

$$\sharp\{\mu \in T_y\colon (\sigma,\mu) \in \texttt{Pfar}\} = O(1)\,,$$

➤ jeder Cluster trägt nur zu einer kleinen Anzahl von Partitionsrechtecken bei.



`x = (0:1/n:(1-1/n));`, $\eta = 0.5$



`x = sqrt(0:1/n:(1-1/n));`, $\eta = 0.5$

◇

Rechenaufwand für Schritt ②: $O(n(d+1)^2)$, Speicheraufwand $O(n(d+1)^2)$

③: Fernfeldauswertung $\hat{=}$ Summe $\displaystyle\sum_{\substack{\sigma\in T_x\\ x_i\in\sigma}} \sum_{\substack{\mu\in T_y\\ (\sigma,\mu)\in P^{\mathrm{far}}}} \mathbf{V}_\sigma \mathbf{X}_{\sigma,\mu}\mathbf{w}_\mu$

foreach $\sigma \in T_x$ { Berechne $\mathbf{V}_\sigma$; $\mathbf{s} := 0$
foreach $\mu \in T_y$, $(\sigma,\mu) \in P^{\mathrm{far}}$ { $\mathbf{s} \leftarrow \mathbf{s} + \mathbf{X}_{\sigma,\mu}\mathbf{w}_\mu$ }
$\mathbf{f}_{|\sigma} \leftarrow \mathbf{f}_{|\sigma} + \mathbf{V}_\sigma \mathbf{s}$ }

Rechenaufwand $O(n \log_2 n (d+1)^2)$, Speicheraufwand $O(n(d+1))$

④: Nahfeldberechnung

foreach $(\sigma,\mu) \in P^{\mathrm{near}}$ { **foreach** i: $x_i \in \sigma$ { $\mathbf{f}_i \leftarrow \mathbf{f}_i + \displaystyle\sum_{j\colon y_j\in\mu} G(x_i,y_j)c_j$ } }

Rechenaufwand $O(n)$

Gesamtrechenaufwand/Gesamtspeicherbedarf $O((d+1)^2\, n \log_2 n)$

Bemerkung 183 (Konvergenz der Clusteringapproximation).

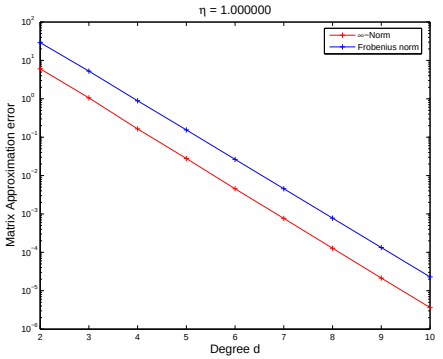
Falls $G(x,y)$ analytisch (→ Def. 4.1.14) in $\{(x,y)\colon x \neq y\}$

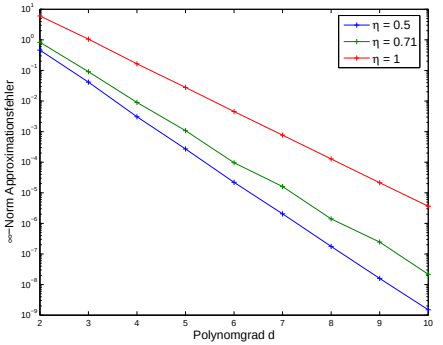
➤ Clusteringapproximation erbt exponentielle Konvergenz von Tschebyscheff-Interpolation, Bem. 4.1.4.4

△

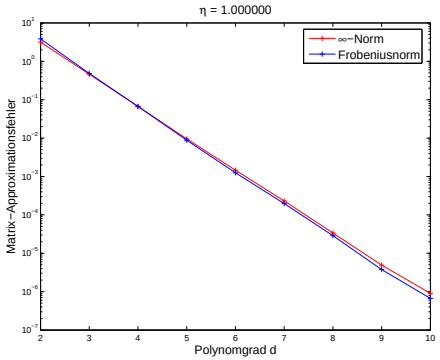
Beispiel 184 (Konvergenz der Clusteringapproximation einer Kollokationsmatrix).

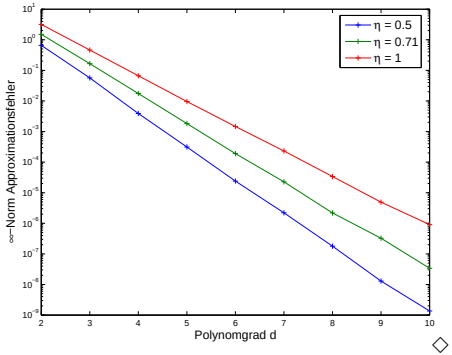
- `x = 0:1/128:1;`, $G(x,y) = |x-y|^{-1}$ für $x \neq y$, $G(x,x) = 0$.





- `x = sqrt(0:1/128:1);`, $G(x,y) = |x-y|^{-1}$ für $x \neq y$, $G(x,x) = 0$.





◇

Beispiel 185 (Auswertung eines trigonometrischen Polynoms). [DUR95] [14]

Gegeben: $\{t_0, \ldots, t_{n-1}\} \subset [0,1[$, $\alpha_{-m+1}, \ldots, \alpha_m \in \mathbb{C}$, $n = 2m$, $m \in \mathbb{N}$, berechne

$$c_k := p(t_k)\,, \quad j = 0, \ldots, n-1 \quad \text{für} \quad \underbrace{p(t) := \sum_{j=-m+1}^{m} \alpha_j e^{-2\pi i j t}}_{\text{Trigonometrisches Polynom}}\,.$$

Diskrete Fouriertransformation (DFT) → Abschnitt 3.7.1:

$$f_l := \sum_{j=-m+1}^{m} \alpha_j e^{-\frac{2\pi i}{n} j l} \quad \text{Lemma 3.7.2} \quad \Leftrightarrow \quad \alpha_j = \frac{1}{n} \sum_{l=-m+1}^{m} f_l e^{\frac{2\pi i}{n} l j}$$

FFT: $f_l, l = -m+1, \ldots, m$, zu berechnen mit Rechenaufwand $O(n \log_2 n)$

$$\begin{aligned} c_k &= \sum_{j=-m+1}^{m} \alpha_j e^{2\pi i j t_k} = \sum_{j=-m+1}^{m} \frac{1}{n} \Big(\sum_{l=-m+1}^{m} f_l e^{\frac{2\pi i}{n} l j} \Big) e^{-2\pi i j t_k} \\ &= \frac{1}{n} \sum_{l=-m+1}^{m} f_l \sum_{j=-m+1}^{m} e^{-2\pi i j (t_k - l/n)} \\ &= \frac{1}{n} \sum_{l=-m+1}^{m} f_l e^{-2\pi i (t_k - l/n)(-m+1)} \frac{1 - e^{-2\pi i n (t_k - l/n)}}{1 - e^{-2\pi i (t_k - l/n)}} \\ &= \frac{1}{n} \sum_{l=-m+1}^{m} f_l e^{-\pi i t_k} \sin(\pi n t_k) \frac{1}{\sin(\pi (t_k - l/n))} (-1)^l e^{-\pi i l/n}\,. \end{aligned}$$

$$\blacktriangleright \quad \mathbf{c} = \operatorname{diag}\left(\frac{\sin(\pi n t_k)}{e^{\pi i t_k}} \right)_{k=-m+1}^{m} \mathbf{M} \operatorname{diag}\left((-1)^l e^{-\pi i l/n} \right)_{l=-m+1,\ldots,m} \mathbf{f}\,.$$

mit Kollokationsmatrix gemäss (4.1.28)

$$\mathbf{M} := \left(\frac{1}{\sin(\pi(t_k - l/n))} \right)_{\substack{k=-m+1,\ldots,m \\ l=-m+1,\ldots,m}} \in \mathbb{R}^{2n,2n}\,.$$

➢ Approximative Auswertung mit Clustering-Algorithmus **!** → USFFT

◇

Bemerkung 186. Clusteringapproximation Beispiel für schnelle approximative Realisierung eines Algorithmus der numerischen linearen Algebra (→ Kap. 3) ➔ Trend in der numerischen linearen Algebra **?**

Problem	Exakte/direkte Methoden	Approximative/iterative Verfahren
Lineare Gleichungssysteme	Gauss-Elimination	CG-artige iterative Löser → Sect. 3.6
Eigenwertproblem	Transformationsmethoden	Krylov-Unterraumverfahren → Sect. 3.4.3
Kollokationsmatrix×Vektor	BLAS (SAXPY)	Clusteringtechniken

△

Mit Clusteringapproximation verwandt, bzw. Verallgemeinerungen davon:

- $\mathcal{H}$-Matrix-Techniken [BGH03] [4]
- $\mathcal{H}^2$-Matrix-Techniken [HAB02] [22]
- Multipol-Methoden [ROK85a, GRR97] [40, 20] → weiterer „Milleniums-Algorithmus”
  `http://orion.math.iastate.edu/burkardt/misc/algorithms_dongarra.html`

4.2 Dreitermrekursionen

[File: section-dreitermrekursion.tex, SVN: section-dreitermrekursion.tex 1011 2006-09-11 18:44:51Z hiptmair]

Technik zur *Darstellung von Funktionen* $I \subset \mathbb{R} \mapsto \mathbb{R}$ (→ Einleitung Kap. 4):

Gegeben: FUNCSYS Funktionensystem $b_k : I \mapsto \mathbb{R}$, $k \in \{1, \ldots, n\}$, $n \in \mathbb{N}$
COEFFS Entwicklungskoeffizienten α_j, $j \in \{1, \ldots, n\}$

Funktion $f(t) = \sum_{j=0}^{\infty} \alpha_j b_j(t)$, $t \in I$.

Aufgabe: Effiziente Auswertung von $f(x)$ für ein/mehrere $x \in I$

DREITERMREK

Dreitermrekursionen für viele relevante Funktionensysteme

$$b_0, b_1 \in \mathbb{R} \quad , \quad b_{k+1}(x) = \kappa_k(x) b_k(x) + \sigma_k(x) b_{k-1}(x) \ , \quad \kappa_k, \sigma_k \in \mathbb{R}, k \in \mathbb{N} \ . \qquad (4.2.1)$$

($\sigma_k \neq 0$)

eq:dreiterm

Beispiel 187 (Orthogonalpolynome). (→ Abschn. 4.1.5.2)

sec:least-squar-best

- Legendre-Polynome auf $I := [-1, 1]$ → Def. 4.4.1, Fig. 38

def:Legendrepol legendre

$$b_0(t) \equiv 1, b_1(t) = t \quad , \quad b_{k+1}(t) = \frac{2k+1}{k+1} t b_k(t) - \frac{k}{k+1} b_{k-1}(t) \ . \qquad (4.4.6)$$

eq:Legpol

- Tschebyscheff-Polynome auf $I := [-1, 1]$ → Def. 4.1.6, Figs. 12, 13

def:Tpol T04 T59

$$b_0(t) \equiv 1, b_1(t) = t \quad , \quad b_{k+1}(t) = 2t b_k(t) - b_{k-1}(t) \ . \qquad (4.1.15)$$

eq:TPrek

Immer gilt: Polynome vom Grad $\leq n$: $\mathcal{P}_n = \mathrm{Span}\{b_0, \ldots, b_n\}$ ◇

Beispiel 188 (Spezielle Funktionen).

- Trigonometrische Funktionen $b_k(t) = \cos(kt)$ bzw. $b_k(t) = \sin(kt)$

$$b_{k+1}(t) = 2\cos t \cdot b_k(t) - b_{k-1}(t) \ , \quad \begin{array}{ll} b_0(t) \equiv 1, b_1(t) = \cos t & \text{für Cosinus,} \\ b_0(t) \equiv 0, b_1(t) = \sin t & \text{für Sinus .} \end{array}$$

Herleitung benutzt Additionstheorem

$$\cos x + \cos y = 2 \cos \frac{x+y}{2} \cos \frac{x-y}{2} \ , \quad x, y \in \mathbb{R} \ ,$$

für $x = (k+1)t$, $y = (k-1)t$.

- Bessel-Funktionen (erster Art) $b_k := J_k : \mathbb{R} \mapsto \mathbb{R}$

Reihenentwicklung:

$$J_k(t) := \sum_{l=0}^{\infty} \frac{(-1)^l}{2^{2l+k} l!(k+l)!} t^{2l+k} \ .$$

Dreitermrekursion:

$$b_{k+1}(t) = \frac{2k}{t} b_k(t) - b_{k-1} \ , k \in \mathbb{N} \ , \qquad (4.2.2)$$
$$b_0 := J_0, b_1 := J_1 \ .$$

besselfunctions

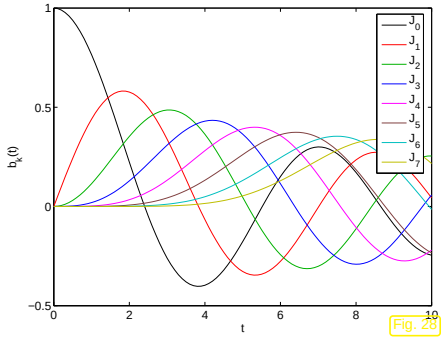


Fig. 20

☛ Wichtig z.B. in Fluidmechanik, Elektromagnetik: Lösungsdarstellung für Feldgleichungen bei Rotationssymmetrie ◇

Plotten von Funktionensystemen aus (4.2.1)

eq:dreiterm

```
function plotbk(kappa,sigma,t,b0,b1,N)
% Plotten von Funktionensystemen
V = [b0;b1];
for k=1:N-1
  V = [V;kappa(k,t).*V(end,:) +...
       sigma(k,t).*V(end-1,:)];
end
figure; hold on;
for k=0:N, plot(t,V(k+1,:),'-'); end
```

◁ MATLAB-Code zur graphischen Ausgabe der b_k

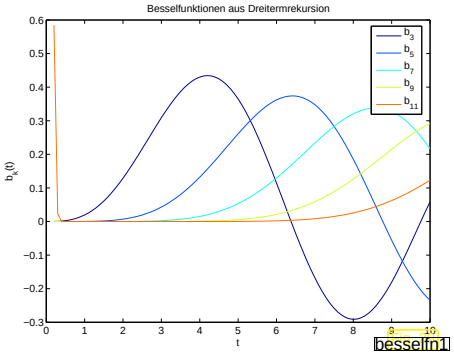
Funktionen `sigma,kappa` ↔ $\sigma_k(x)$, $\kappa_k(x)$

Für Besselfunktionen J_k:

- `kappa = @(k,t) 2*k./t;`
- `sigma = @(k,t) -1;`

Beispiel 189 (Besselfunktionen aus Dreitermrekursion).

Beobachtung: “Instabilität” für kleine t, grosse k **!?**



besselfn1

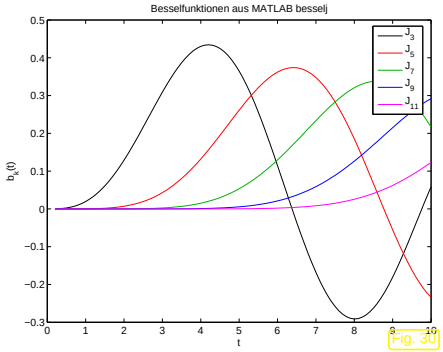
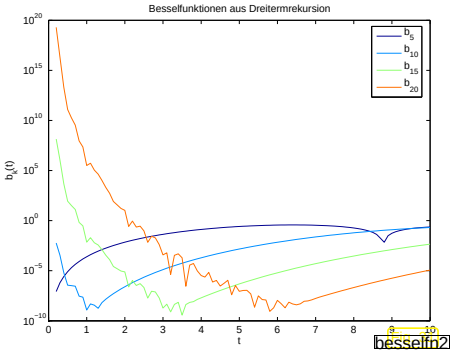


Fig. 30



besselfn2

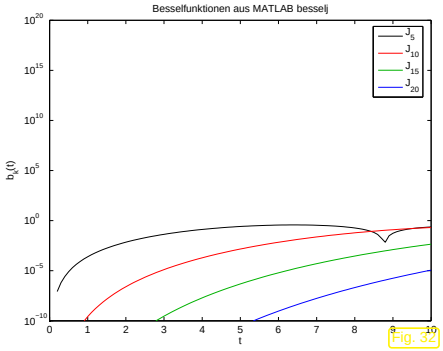


Fig. 32 ◇

Modellproblem: Skalare Dreitermrekursion mit konstanten Koeffizienten:

$$x^{(k+1)} = \kappa \cdot x^{(k)} - x^{(k-1)} \quad , \quad x^{(0)}, x^{(1)} \in \mathbb{R} \, . \tag{4.2.3}$$

eq:3tsimj

Ansatz: $\boxed{x^{(k)} = \zeta^k}$, $\zeta \in \mathbb{R}$

◣ notwendig: $\zeta^2 - \kappa\zeta + 1 = 0$.

◣ $\zeta_{1,2} = \kappa/2 \pm \sqrt{(\kappa/2)^2 - 1}$: $\begin{array}{lll} |\kappa| \leq 2 & \Rightarrow & |\zeta_{1,2}| = 1 \, , \\ |\kappa| > 2 & \Rightarrow & ; |\zeta_1| > 1 \, , \, |\zeta_2| < 1 \, . \end{array}$

Allgmeine Lösung: $x^{(k)} = c_1\zeta_1^k + c_2\zeta_2^k \, , \quad c_1, c_2 \in \mathbb{R} \, .$ (4.2.4)

eq:3tgen

Beispiel 190 (Instabile Dreitermrekursion).

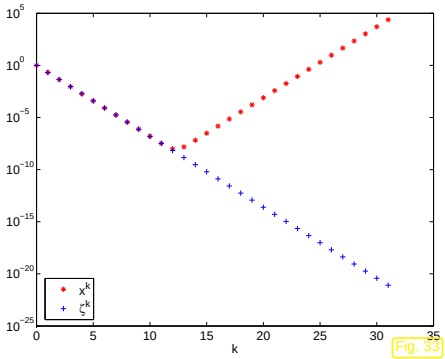
eq:3tsimple
(4.2.3) für $\kappa = 5$, $x^{(0)} = 1$, $x^{(1)} = 5/2 - \frac{1}{2}\sqrt{21}$

◣ $x^{(k)} = (5/2 - \frac{1}{2}\sqrt{21})^k$.

Beobachtung:

“Umspringen” zu $x^{(k)} = (5/2 + \frac{1}{2}\sqrt{21})^k$

? Warum denn das **?**



3tinstable

Fig. 33 ◇

Im Computerprogramm: $x^{(1)} = 0.20871215252208 \neq 5/2 - \frac{1}{2}\sqrt{21}$ Rundung !

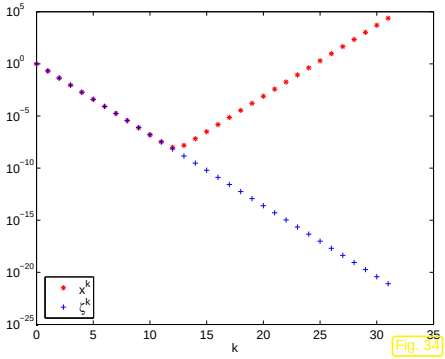
Gestörte $x^{(0)}, x^{(1)}$ $\Rightarrow$ Dominanz exponentiell wachsender Lösung für $k \gg 1$

Beispiel 191 (Instabile Dreitermrekursion für Besselfunktionen).

eq:3tbess
Dreitermrekursion (4.2.2):

$t = 1/2$, $b_0 = J_0(1/2)$ $b_1 = J_1(1/2)$

Beobachtung: ▷ “Wuchern” einer exponentiell wachsenden Lösung



3tbessinstable

Fig. 34 ◇

Idee: Rückwärtsrekursion

Fixiere $N \gg n$ und berechne $b_k(t)$, $k = 0, \ldots, n$, aus (für $\sigma_k(t) \neq 0$)

$$b_{k-1}(t) = -\frac{\kappa_k(t)}{\sigma_k(t)} b_k(t) + \frac{1}{\sigma_k(t)} b_{k+1}(t) \quad , \quad b_N(t), b_{N-1}(t) \text{ “beliebig”}$$

(4.2.5) eq:inv3t

Modellproblemanalyse: Rückwärtsrekursion für (4.2.3) eq:3tsimple

$$x^{(k-1)} = \kappa \cdot x^{(k)} - x^{(k+1)} \quad = \quad \text{(4.2.3)} \quad \text{eq:3tsimple}$$

▶ Weiterhin: allgemeine Lösung (4.2.4) eq:3tgensol: $x^{(k)} = c_1 \zeta_1^k + c_2 \zeta_2^k$

Annahme: $0 < |\zeta_1| < 1 < |\zeta_2|$, Lösung $x^{(k)} = \zeta_1^k$ gesucht

$$x^{(N)}, x^{(N-1)} \text{ “beliebig”} \quad \Rightarrow \quad c_1 \zeta_1^N \approx c_2 \zeta_2^N \quad \Rightarrow \quad |c_1| \gg |c_2|$$

▶ für $x^{(k)}$ aus (4.2.3) eq:3tsimple: $x^{(k)} \approx c_1 \zeta_1^k$.

Nun gut, aber $c_1 = ?$

Idee: Nutze zusätzliche Normierungsbedingung

$$\sum_{k=0}^{\infty} \omega_k b_k(t) = \text{bekannt} \,, \quad \omega_k \in \mathbb{R} \,. \qquad (4.2.6) \quad \text{eq:normt}$$

Normierungsbedingungen:

- Für Modellproblem (4.2.3) eq:3tsimple: $x^{(0)} = 1$
- Für Besselfunktionen ABS70 [1]: $J_0(t) + 2\sum_{k=1}^{\infty} J_{2k}(t) = 1$

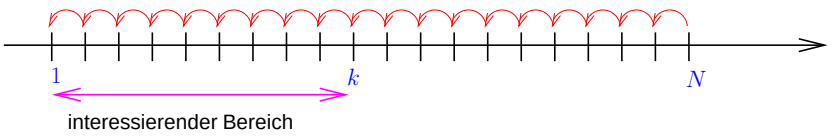
Rückwärtsrekursion für Besselfunktionen

```
function x = revrecbess(t,n,N)
x = [1,0];
for k=N:-1:1
  x = [2*k/t*x(1)-x(2),x];
end
G = x(1) + 2*sum(x(3:2:length(x)));
x = x(1:n+1)/G;
```

◁ MATLAB-Funktion: Auswertung von Besselfunktionen J_k, $k = 0, \ldots, n$ für Argument $t > 0$

Technik:

Rückwärtsrekursion
+
Normierungsbedingung



interessierender Bereich

Beispiel 192 (Approximation von Besselfunktionen mittels Rückwärtsrekursion).

Für $t = 10, 20, 30, 40$ Näherungen $\widetilde{J}_k(t)$ durch Rückwärtsrekursion für $k = 0, \ldots, 20$

Fehlermass:

$$\max_{0 \leq k \leq 20} |J_k(t) - \widetilde{J}_k(t)|$$

Beobachtung ▷

Asymptotisch exponentielle Konvergenz

revrecbess

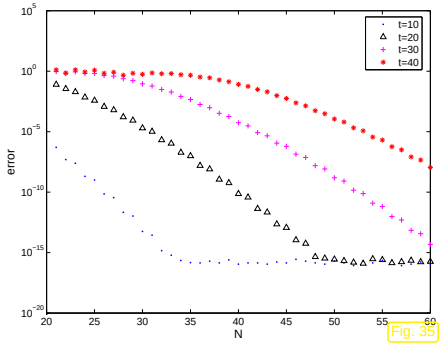


Fig. 35

◇

Bemerkung 193.

Adaptive Rückwärtsrekursion für Besselfunktionen

```
function y = besscomp(t,n,tol)
d = realmax; N = n + 10;
x = revrecbess(t,n,N);
while(d > tol)
  N = N + 10;
  y = revrecbess(t,n,N);
  d = max(abs(x-y));
end
```

Adaptive Wahl von N:

Erhöhe N solange bis sich “Approximation stabilisiert”
(d.h. Fehler $<$ Toleranz `tol`)

△

4.3 Stückweise Polynome

[File: section-stueckweise-polynome.tex, SVN: section-stueckweise-polynome.tex 1270 2007-01-15 10:44:02Z hiptmair]

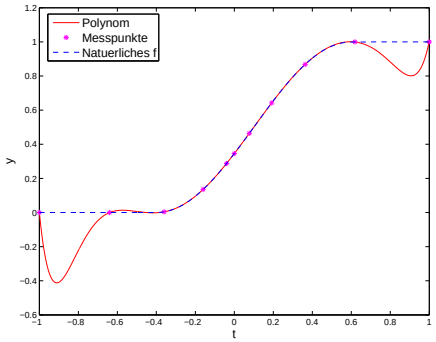
STKWPOLY

Aufgabenstellung: Modellieren eines funktionalen Zusammenhangs $f : I \subset \mathbb{R} \mapsto \mathbb{R}$ basierend auf (genauen) Messungen (t_i, y_i), $i = 0, \ldots, n$:

➢ Interpolationsbedingung $f(t_i) = y_i$, $i = 0, \ldots, n$

Beispiel 194 (Versagen der Polynominterpolation).

t_i	-1.0000	-0.6400	-0.3600	-0.1600	-0.0400	0.0000	0.0770	0.1918	0.3631	0.6187	1.0000
y_i	0.0000	0.0000	0.0039	0.1355	0.2871	0.3455	0.4639	0.6422	0.8678	1.0000	1.0000



← Interpolationspolynom, Grad = 10

Oszillationen an Intervallenden (→ Bsp. 162) ex:runge

- Keine Lokalität
- Keine Monotonie
- Keine Krümmungserhaltung

$$f(t) = \begin{cases} 0 & \text{, falls } t < -\frac{2}{5}\,, \\ \frac{1}{2}(1+\cos(\pi(t-\frac{3}{5}))) & \text{, falls } -\frac{2}{5} < t < \frac{3}{5}\,, \\ 1 & \text{sonst.} \end{cases}$$

Idee: Benutze stückweise Polynome bzgl. einer Partition (Gitter, engl. *mesh*) $\mathcal{M} := \{a = x_0 < x_1 < \ldots < x_m = b\}$ des Intervalls $I := [a,b]$, $a < b$.

Beachte: $x_j \neq t_j$ erlaubt !

4.3.1 Stückweise Polynominterpolation

Für Polynomgrad $d \in \mathbb{N}$ fasse jeweils $d+1$ Stützstellen zusammen ➤ Gitter $\mathcal{M}$

:	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}	t_{14}	t_{15}	t_{16}	t_{17}	t_{18}	t_{19}	t_{20}
$d=2$:	x_0		x_1		x_2		x_3		x_4		x_5		x_6		x_7		x_8		x_9		x_{10}
$d=4$:	x_0				x_1				x_2				x_3				x_4				x_5

Sückweises Interpolationspolynom $s : [x_0, x_n] \to \mathbb{K}$:

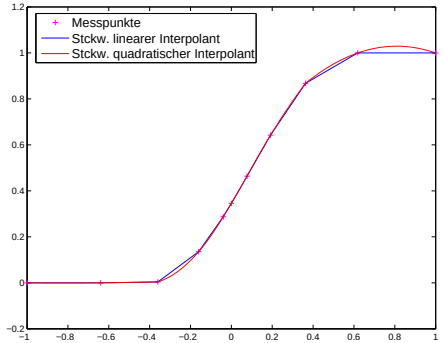
$$s_j := s_{|[x_{j-1},x_j]} \in \mathcal{P}_d \quad \text{und} \quad s_j(t_i) = y_i \quad i = 0,\ldots,n\,, \quad j = 1,\ldots,m\,.$$

Beispiel 195 (Stückweise Polynominterpolation von Messpunkten).

Messpunkte aus Bsp. 194 ex:logpolyintpol

Stückweise lineare/quadratische Interpolation

Lokalität



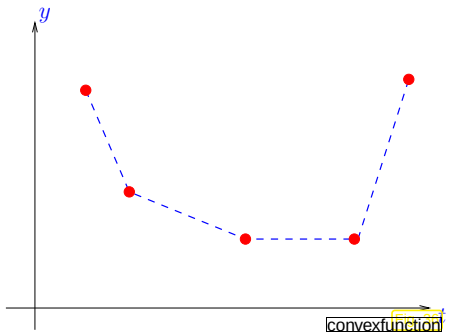
4.3.1.1 Stückweise lineare Interpolation

Daten: $(t_i, y_i) \in \mathbb{R}^2$, $i = 0,\ldots,n$, $n \in \mathbb{N}$, $t_0 < t_1 < \cdots < t_n$

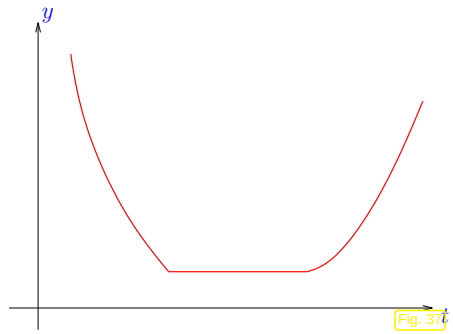
Definition 4.3.1 (Monotone Daten). *Daten monoton, wenn* $y_i \geq y_{i-1}$ *oder* $y_i \leq y_{i-1}$, $i = 1,\ldots,n$.

Definition 4.3.2 (Konvexe/konkave Daten). *Daten* $\{(t_i,y_i)\}_{i=0}^n$ *konvex (konkav), falls*

$$\Delta_j \overset{(\geq)}{\leq} \Delta_{j+1}\,, \quad j = 1,\ldots,n-1 \quad , \quad \Delta_j := \frac{y_j - y_{j-1}}{t_j - t_{j-1}}\,, j = 1,\ldots,n\,.$$



Konvexe Daten



Konvexe Funktion

Definition 4.3.3 (Konvexe/konkave Funktion).

$$f : I \subset \mathbb{R} \mapsto \mathbb{R} \quad \begin{matrix}\text{konvex}\\ \text{konkav}\end{matrix} \quad :\Leftrightarrow \quad \begin{matrix} f(\lambda x + (1-\lambda)y) \le \lambda f(x) + (1-\lambda) f(y) \\ f(\lambda x + (1-\lambda)y) \ge \lambda f(x) + (1-\lambda) f(y) \end{matrix} \quad \forall 0 \le \lambda \le 1 \,, \quad \forall x, y \in I \,.$$

SHAPEPRESERV

Theorem 4.3.4 (Formerhaltung bei stückweise linearer Interpolation).

Für stückweise linearen Interpolanten $s \in C([t_0, t_n])$ von $(t_i, y_i) \in \mathbb{R}^2$, $i = 0, \ldots, n$:

(t_i, y_i) *monoton (steigend/fallend)* $\Rightarrow$ s *monoton (steigend/fallend)*
(t_i, y_i) *konvex/konkav* $\Rightarrow$ s *konvex/konkav*

Lineare Interpolation ist linear (als Abbildung $\mathbf{y} \mapsto s$) und *lokal*:

$$y_j = \delta_{ij} \,, \quad i, j = 0, \ldots, n \quad \Rightarrow \quad \operatorname{supp}(s) \subset [t_{i-1}, t_{i+1}] \,.$$

4.3.1.2 Stückweise polynomiale Interpolation von Funktionen

Lokale Lagrange-Interpolation von $f \in C(I)$:

Zu Gitter $\mathcal{M}$ wähle Knotenmenge $\mathcal{T}^j := \{t_0^j, \ldots, t_{n_j}^j\} \subset I_j$ für jede Gitterzelle $I_j := [x_{j-1}, x_j]$.

Sückweises Interpolationspolynom $s : [x_0, x_m] \to \mathbb{K}$:

$$s_j := s_{|I_j} \in \mathcal{P}_{n_j} \quad \text{und} \quad s_j(t_i^j) = f(t_i^j) \quad i = 0, \ldots, n_j \,, \quad j = 1, \ldots, m \,.$$

Beachte: $$t_{n_{j-1}}^{j-1} = x_{j-1} = t_0^j, \, j = 2, \ldots, m \quad \Leftrightarrow \quad s \in C^0(I)$$

Fragestellung: Asymptotik des Interpolationsfehlers für n fest, $m \to \infty$
(Interpolationsfehler $\le T(h)$, MESHWIDTH Maschenweite $h := \max\{|x_j - x_{j-1}| : j = 1, \ldots, m\}$)

Beispiel 196 (Stückweise Polynominterpolation).

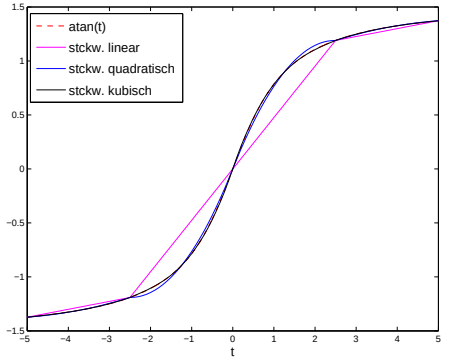
vgl. Bsp. 194: ex:logpolyintpol

$f(t) = \arctan t$, $I = [-5, 5]$

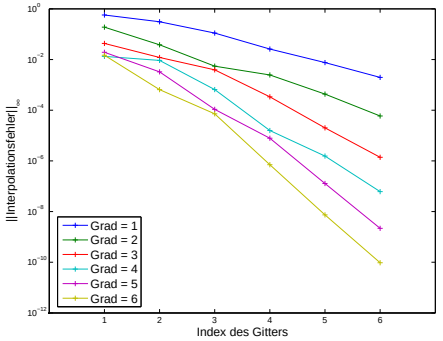
Gitter $\mathcal{M} := \{-5, -\frac{5}{2}, 0, \frac{5}{2}, 5\}$

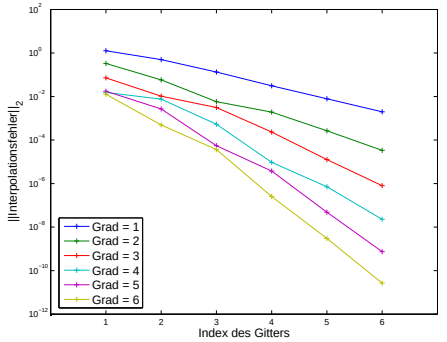
$\mathcal{T}^j$ äquidistant in I_j

Plots der stückweise linearen, quadratischen und kubischen Polynominterpolanten



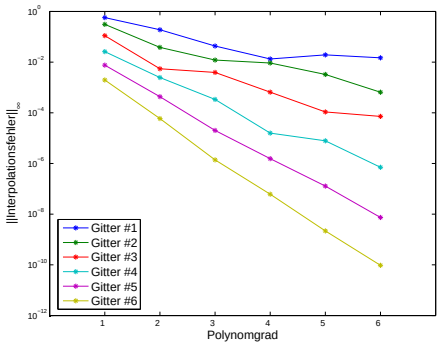
Interpolationsfehlernormen auf Gittern $\mathcal{M}_i := \{-5 + j\, 2^{-i} 10\}_{j=0}^{2^i}$, $i = 1, \ldots, 6$, äquidistante Knotenmengen:

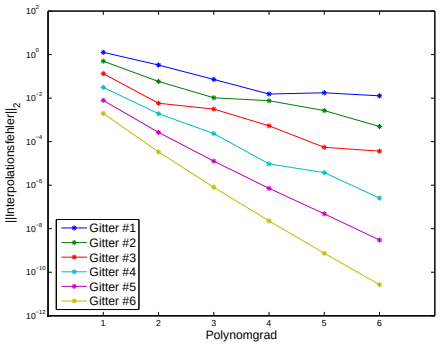




➤ Algebraische Konvergenz in Maschenweite

➤ Exponentielle Konvergenz im Polynomgrad





◇

Interpolationsfehlerabschätzungen:

- Für festen Polynomgrad $n = n_j$, $j = 1, \ldots, m$:
  Anwendung von Cor 4.1.5 auf jedes Teilintervall: falls $f \in C^{n+1}([x_0, x_m])$ cor:polintperr

$$\|f - s\|_{L^\infty([x_0,x_m])} \leq \frac{h^{n+1}}{(n+1)!} \left\| f^{(n+1)} \right\|_{L^\infty([x_0,x_m])} \quad ,$$

  mit Maschenweite $h := \max\{|x_j - x_{j-1}| : j = 1, \ldots, m\}$.
- Für festes Gitter: Situation wie bei Standard-Polynominterpolation → Abschnitt 4.1.2. sec:interp

4.3.1.3 Kubische Hermite-Interpolation

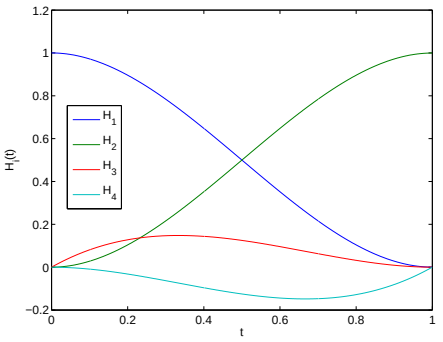
Gegeben: Messpunkte $(t_i, y_i) \in \mathbb{R}^2$, $i = 0, \ldots, n$, $t_0 < t_1 < \cdots < t_n$

Gesucht: Modellfunktion $f \in C^1([x_0, x_n])$, $f(t_i) = y_i$, $i = 0, \ldots, n$

➤ Stückweise kubisches Hermite-Interpolationspolynom $s \in C^1([t_0, t_n])$:
zu gegebenen Steigungen $c_i \in \mathbb{R}$, $i = 0, \ldots, n$:

$$s_{|[t_{i-1},t_i]} \in \mathcal{P}_3 \,, \quad i = 1, \ldots, n \quad , \quad s(t_i) = y_i \,, \quad i = 0, \ldots, n \quad , \quad s'(t_i) = c_i \,, \quad i = 0, \ldots, n \,.$$

$$s(t) = y_{i-1} H_1(t) + y_i H_2(t) + c_{i-1} H_3(t) + c_i H_4(t) \quad , \quad t \in [t_{i-1}, t_i] \,, \qquad (4.3.1)$$ eq:kub



$$H_1(t) := \phi(\tfrac{t_i - t}{h_i}) \,, \quad H_2(t) := \phi(\tfrac{t - t_{i-1}}{h_i}) \,,$$
$$H_3(t) := -h_i \psi(\tfrac{t_i - t}{h_i}) \,, \quad H_4(t) := h_i \psi(\tfrac{t - t_{i-1}}{h_i}) \,,$$
$$h_i := t_i - t_{i-1} \quad ,$$
$$\phi(\tau) := 3\tau^2 - 2\tau^3 \quad ,$$
$$\psi(\tau) := \tau^3 - \tau^2 \,.$$

Stückweise kubisches Polynom s auf t_1, t_2 mit $s(t_1) = y_1$, $s(t_2) = y_2$, $s'(t_1) = c_1$, $s'(t_2) = c_2$:

Effiziente lokale Auswertung ➤

```
function s = kubeval(t,t1,t2,y1,y2,c1,c2)
h = t2-t1; t = (t-t1)/h;
a1 = y2-y1; a2 = a1-h*c1;
a3 = h*c2-a1-a2;
s = y1+(a1+(a2+a3*t).*(t-1)).*t;
```

Wahl der Steigungen c_i **?**

☞ Mittelwert von Abschnittssteigungen:

$$c_i = \begin{cases} \Delta_1 & \text{, für } i = 0 \,, \\ \Delta_n & \text{, für } i = n \,, \\ \frac{t_{i+1} - t_i}{t_{i+1} - t_{i-1}} \Delta_i + \frac{t_i - t_{i-1}}{t_{i+1} - t_{i-1}} \Delta_{i+1} & \text{, falls } 1 \leq i < n \,. \end{cases} \quad , \quad \Delta_j := \frac{y_j - y_{j-1}}{t_j - t_{j-1}} \,, j = 1, \ldots, n \,.$$

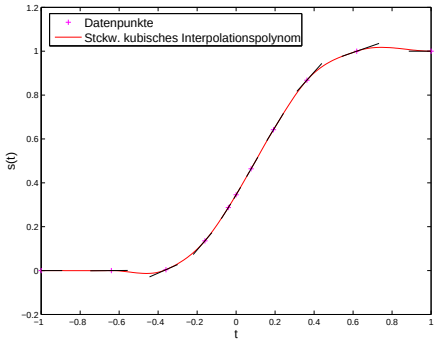
➤ Linearer lokaler Interpolationsoperator

Beispiel 197 (Stückweise kubische Hermite-Interpolation).

Daten aus Bsp. 194 ex:logpolyintpol

Verwendung gewichtet gemittelter Steigungen ➢

Keine Erhaltung von Monotonie und Krümmung



◇

☞ Wahl der c_i mit „Limiter” → Monotonieerhaltung [FRC80 [15]]

$$c_i = \begin{cases} 0 & \text{, falls } \operatorname{sgn}(\Delta_i) \neq \operatorname{sgn}(\Delta_{i+1}) \text{ ,} \\ \text{gewichtetes Mittel von } \Delta_i, \Delta_{i+1} & \text{sonst} \end{cases} \quad , \quad i = 1, \ldots, n-1 \text{ .}$$

Beispiel 198 (Monotoniererhaltende stückweise kubische Polynominterpolation).

Daten aus Bsp. 194 [ex:logpolyintpol]

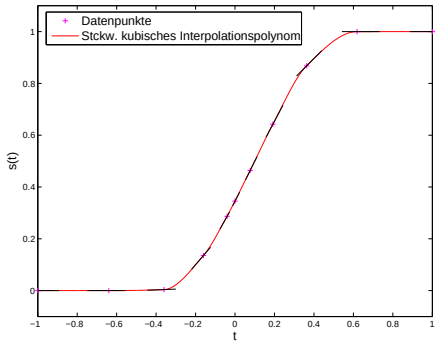
MATLAB-Funktion:

```
v = pchip(t,y,x);
```

t: Stützstellen
y: Stützwerte
x: Auswertungsstellen
v: Vektor $s(x_i)$

Lokaler Interpolationsoperator

! Nichtlinearer Interpolationsoperator



◇

Berechnung der c_i in `pchip` (Details in [FRC80 [15]]):

```
function c = crfslopes(x,y)
n = length(x); h = diff(x); del = diff(y)./h;
if n==2, c = [del(1),del(1)];
    return;
end
k = find(sign(del(1:n-2)).*sign(del(2:n-1)) > 0);
c = zeros(size(y)); hs = h(k)+h(k+1);
w1 = (h(k)+hs)./(3*hs); w2 = (hs+h(k+1))./(3*hs);
dmax = max(abs(del(k)), abs(del(k+1))); dmin = min(abs(del(k)), abs(del(k+1)));
c(k+1) = dmin./conj(w1.*(del(k)./dmax) + w2.*(del(k+1)./dmax));
c(1) = ((2*h(1)+h(2))*del(1) - h(1)*del(2))/(h(1)+h(2));
if (sign(d(1)) ~= sign(del(1)))
    d(1) = 0;
elseif (sign(del(1)) ~= sign(del(2))) && (abs(d(1)) > abs(3*del(1)))
  c(1) = 3*del(1);
end
c(n) = ((2*h(n-1)+h(n-2))*del(n-1) - h(n-1)*del(n-2))/(h(n-1)+h(n-2));
if (sign(d(n)) ~= sign(del(n-1)))
    d(n) = 0;
elseif (sign(del(n-1)) ~= sign(del(n-2))) && (abs(d(n)) > abs(3*del(n-1)))
  c(n) = 3*del(n-1);
end
```


4.3.2 Splines

SPLINES

line **Definition 4.3.5** (Splineraum). *für ein Intervall $I := [a,b] \subset \mathbb{R}$, eine Knotenmenge (Gitter) $\mathcal{M} := \{a = t_0 < t_1 < \ldots < t_{n-1} < t_n = b\}$ ist der Vektorraum $\mathcal{S}_{d,\mathcal{M}}$ der Splinefunktionen vom Grad d und der Ordnung $d+1$ definiert durch*

$$\mathcal{S}_{d,\mathcal{M}} := \{s \in C^{d-1}(I) \colon s_j := s_{|[t_{j-1},t_j]} \in \mathcal{P}_d \; \forall j = 1, \ldots, d\} \text{ .}$$

► $s \in \mathcal{S}_{d,\mathcal{M}} \;\Rightarrow\; s' \in \mathcal{S}_{d-1,\mathcal{M}} \;\wedge\; \int_a^b s(\tau)\,\mathrm{d}\tau \in \mathcal{S}_{d+1,\mathcal{M}}$.

- $d = 0$: $\mathcal{M}$-stückweise konstante *unstetige* Funktionen
- $d = 1$: $\mathcal{M}$-stückweise lineare *stetige* Funktionen
- $d = 2$: *Stetig differenzierbare* $\mathcal{M}$-stückweise quadratische Funktionen

Abzählargument (Heuristik):

$$\dim \mathcal{S}_{d,\mathcal{M}} = n \cdot \dim \mathcal{P}_d - (n-1) \cdot d = n + d \text{ .}$$

4.3.2.1 Splineinterpolation

Spezialfall: Interpolation in $\mathcal{S}_{1,\mathcal{M}}$ = Stückweise lineare Interpolation

Weiterer Spezialfall: **Kubische Splineinterpolation** $d = 3$ (→ Abschnitt 4.3.1.3) [sec:kubische-herm-interp]

[eg:kub (4.3.1)]

► $$s_{|[t_{j-1},t_j]}(t) = s(t_{j-1})(1 - 3\tau^2 + 2\tau^3) + s(t_j)(3\tau^2 - 2\tau^3) + h_j s'(t_{j-1})(\tau - 2\tau^2 + \tau^3) + h_h s'(t_j)(-\tau^2 + \tau^3) \text{ ,}$$

mit $h_j := t_j - t_{j-1}$, $\tau := (t - t_{j-1})/h_j$.

➤ Falls $s(t_j), s'(t_j)$, $j = 0, \ldots, l$ bekannt → einfache Auswertung von s (→ Abschnitt 4.3.1.3) [sec:kubische-herr

Interpolationsbedingung $s(t_j) = y_j$, $j = 0, \ldots, n$: Steigungen $c_j := s'(t_j)$ = **?**

! $s \in C^2([t_0, t_n])$ ➤ $n-1$ Stetigkeitsbedingung für $s''(t)$ ➤ $n-1$ Gleichungen

$$\frac{1}{h_j} c_{j-1} + \left(\frac{2}{h_j} + \frac{2}{h_{j+1}}\right) c_j + \frac{1}{h_{j+1}} c_{j+1} = 3\left(\frac{y_j - y_{j-1}}{h_j} + \frac{y_{j+1} - y_j}{h_{j+1}}\right) \quad , \quad j = 1, \ldots, n-1 \text{ .}$$

➤ Zwei zusätzliche Bedingungen erforderlich:

① Vollständige kubische Splineinterpolation: $s'(t_0) = c_0$, $s'(t_n) = c_n$ vorgeschrieben

② Natürliche kubische Splineinterpolation: $s''(t_0) = s''(t_n) = 0$

$$\frac{2}{h_1}c_0 + \frac{1}{h_1}c_1 = 3\frac{y_1 - y_0}{h_1^2} \quad , \quad \frac{1}{h_n}c_{n-1} + \frac{2}{h_n}c_n = 3\frac{y_n - y_{n-1}}{h_n^2} \, .$$

➤ Jeweils LGS mit tridiagonaler s.p.d. (→ Def. 3.2.8, Lemma 3.2.11) Koeffizientenmatrix → $c_0, \ldots, c_n$

Thm. 3.2.18 ▷ Rechenaufwand zur Lösung $= O(n)$

③ Periodische kubische Splineinterpolation: $s'(t_0) = s'(t_n)$, $s''(t_0) = s''(t_n)$

► $n \times n$-LGS mit s.p.d. Koeffizientenmatrix

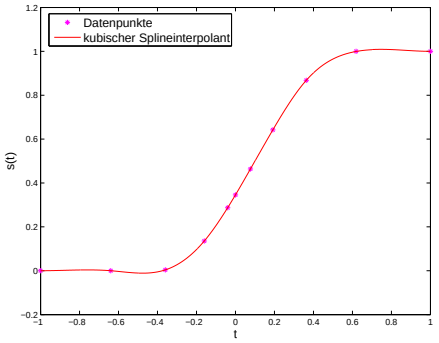
$$\mathbf{A} := \begin{pmatrix} a_1 & b_2 & 0 & \cdots & 0 & b_n \\ b_2 & a_2 & b_3 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & 0 \\ 0 & & & \ddots & \ddots & b_{n-1} \\ b_n & 0 & \cdots & 0 & b_{n-1} & a_{n-1} \end{pmatrix} \quad , \quad \begin{aligned} b_i &:= \tfrac{1}{h_{i-1}}, \quad i = 2, 3, \ldots, n \, , \\ a_i &:= \tfrac{2}{h_i} + \tfrac{2}{h_{i+1}}, \quad i = 1, 2, \ldots, n-1 \, . \end{aligned}$$

➤ Lösung durch Rang-1-Modifikationstechniken (→ Abschnitt 3.2.8.1, Lemma 3.2.22) + tridiagonale Elimination, Rechenaufwand $O(n)$

Beispiel 199 (Vollständige kubische Splineinterpolation).

Daten aus Bsp. 194

$$c_0 := \frac{y_1 - y_0}{t_1 - t_0} \, ,$$
$$c_n := \frac{y_n - y_{n-1}}{t_n - t_{n-1}} \, .$$



Kubischer Splineinterpolant nicht monotonie- oder krümmungserhaltend
(kubische Splineinterpolation linear !)

◇

MATLAB-Funktion: `v = spline(t,y,x)`: Natürliche/vollständige Splineinterpolation
(→ Spline-Toolbox in MATLAB)

Beispiel 200 (Lokalität der natürlichen kubischen Splineinterpolation).

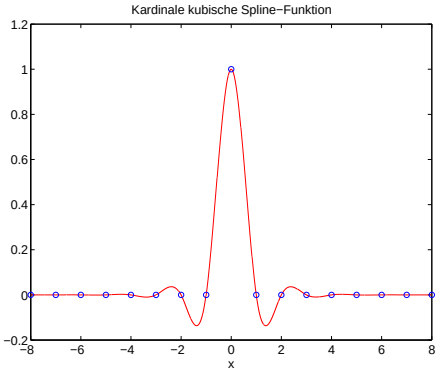
Gegeben: Gitter $\mathcal{M} := \{t_0 < t_1 < \cdots < t_n\}$

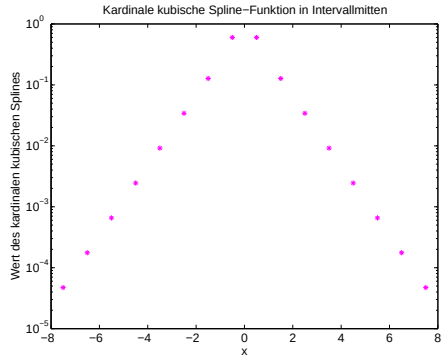
$L_i \in \mathcal{S}_{3,\mathcal{M}}$ = i. natürlicher Kardinalspline $:\Leftrightarrow$ $L_i(t_j) = \delta_{ij}$, $L_i''(t_0) = L_i''(t_n) = 0$.

► Natürlicher Spline-Interpolant: $s(t) = \sum_{j=0}^{n} y_j L_j(t)$.

Abfall der L_i ↔ Lokalität der kubischen Splineinterpolation

Hier: $\mathcal{M} = \{-30, -29, \ldots, 29, 30\}$





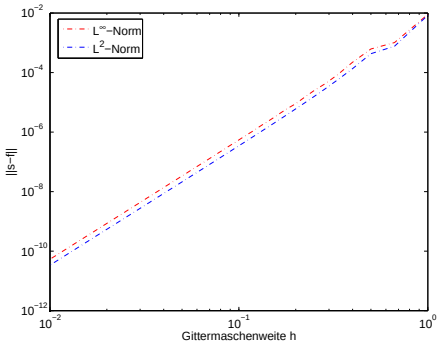
➢ Exponentieller Abfall der Kardinalsplines ➞ kubische Splineinterpolation „fast lokal”

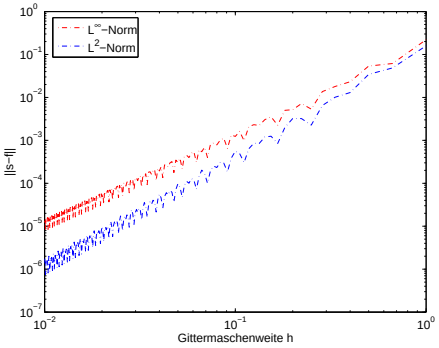
◇

Beispiel 201 (Approximation durch vollständigen kubischen Splineinterpolanten).

Gitter $\mathcal{M} := \{-1 + \frac{2}{n}j\}_{j=0}^{n}$, $n \in \mathbb{N}$ ➢ Maschenweite $h = 2/n$, $I = [-1, 1]$

$$f_1(t) = \frac{1}{1 + e^{-2t}} \in C^\infty(I) \quad , \quad f_2(t) = \begin{cases} 0 & \text{, falls } t < -\frac{2}{5} \, , \\ \frac{1}{2}(1 + \cos(\pi(t - \frac{3}{5}))) & \text{, falls } -\frac{2}{5} < t < \frac{3}{5} \, , \\ 1 & \text{sonst} \end{cases} \in C^1(I) \, .$$





$\|f_1 - s\|_{L^\infty([-1,1])} = O(h^4)$ $\|f_2 - s\|_{L^\infty([-1,1])} = O(h^2)$

Theorie [27]: $f \in C^4([t_0,t_n])$ ➤ $\|f - s\|_{L^\infty([t_0,t_n])} \le \frac{5}{384} h^4 \left\| f^{(4)} \right\|_{L^\infty([t_0,t_n])}$ ◇

Glattheit des kubischen Splineinterpolanten

Für $f : [a,b] \mapsto \mathbb{R}$, $f \in C^2([a,b])$: $\frac{1}{2}\int_a^b |f''(t)|^2 \,\mathrm{d}t$ = elastische Krümmungsenergie

Zu Gitter $\mathcal{M} := \{t_0 < t_1 < \cdots < t_n\}$: $s \in \mathcal{S}_{3,\mathcal{M}}$ = natürlicher kubischer Splineinterpolant von $(t_i, y_i) \in \mathbb{R}^2$, $i = 0,\ldots,n$.

$k \in C^2([t_0,t_n])$ erfülle $k(t_i) = 0$, $i = 0,\ldots,n$:

$$h(\lambda) := \tfrac{1}{2}\int_a^b |s'' + \lambda k|^2 \,\mathrm{d}t \quad \Rightarrow \quad \frac{\mathrm{d}h}{\mathrm{d}\lambda|_{\lambda=0}} = \int_a^b s''(t)k''(t)\,\mathrm{d}t = \sum_{j=1}^{n}\int_{t_{j-1}}^{t_j} s''(t)k''(t)\,\mathrm{d}t$$

Zweimalige partielle Integration:

$$\blacktriangleright \quad \int_a^b s''(t)k''(t)\,\mathrm{d}t = s''(t_n)k'(t_n) - s''(t_0)k'(t_0) = 0\ .$$

Theorem 4.3.6 (Optimalität des natürlichen kubischen Splineinterpolanten)**.** *Unter allen interpolierenden zweimal stetig differenzierbaren Funktionen minimiert der natürliche kubische Splineinterpolant die elastische Krümmungsenergie.*

4.3.2.2 Formerhaltende Splineinterpolation

Gegeben: Messpunkte $(t_i, y_i) \in \mathbb{R}^2$, $i = 0,\ldots,n$, $t_0 < t_1 < \cdots < t_n$

Gesucht: Gitter $\mathcal{M} \subset [t_0,t_n]$ & formerhaltende interpolierende quadratische Splinefunktion $s \in \mathcal{S}_{2,\mathcal{M}}$, $s(t_i) = y_i$, $i = 0,\ldots,n$ ($\mathcal{M} \neq \{t_j\}_{j=0}^n$!)

① „Formtreue” Wahl von Steigungen c_i, $i = 0,\ldots,n$ MAR81, DQQ01 [31, 36] ⟶ Abschnitt 4.3.1.3 sec:kubische-herm-inter

Limiter $c_i := \begin{cases} \frac{2}{\Delta_i^{-1}+\Delta_{i+1}^{-1}} & \text{, falls } \operatorname{sign}(\Delta_i) = \operatorname{sign}(\Delta_{i+1}) \\ 0 & \text{sonst,} \end{cases}$, $i = 1,\ldots,n-1$.

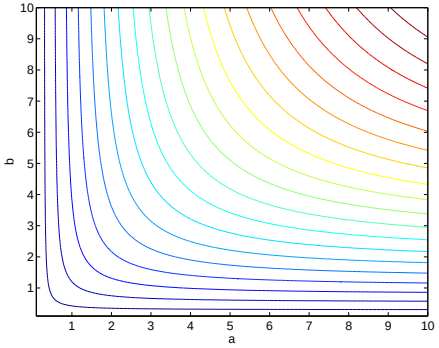
$c_0 := 2\Delta_1 - c_1$, $c_n := 2\Delta_n - c_{n-1}$.

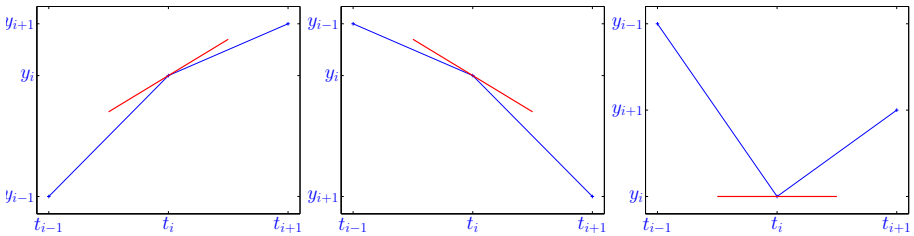
$$c_i := \frac{2}{\Delta_i^{-1} + \Delta_{i+1}^{-1}}$$

= Harmonisches Mittel

Harmonisches Mittel von a und b ➢

„Geglättete $\min(\cdot,\cdot)$-Funktion”

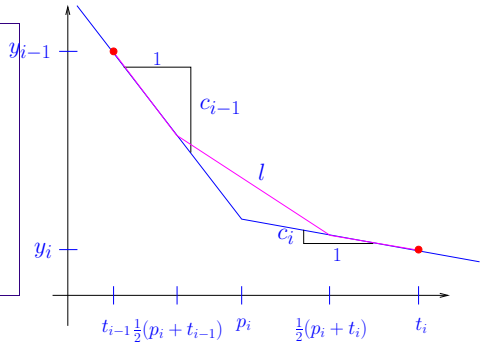




② Wahl von „Zwischenpunkten” $p_i \in]t_{i-1}, t_i]$, $i = 1, \ldots, n$:

$$p_i = \begin{cases} \text{Schnittpunkt der Geraden durch } (t_{i-1}, y_{i-1}), (t_i, y_i) \\ \text{mit Steigungen } c_{i-1}, c_i \\ \frac{1}{2}(t_{i-1} + t_i) \end{cases} \quad \begin{array}{l} \text{, falls Schnittpunkt } \in]t_{i-1}, t_i] \text{ ,} \\ \\ \text{sonst.} \end{array}$$

```
p = (t(1)-1)*ones(1,length(t)-1);
for j=1:n-1
  if (c(j) ~= c(j+1))
    p(j)=(y(j+1)-y(j)+...
        t(j)*c(j)-t(j+1)*c(j+1))/...
          (c(j)-c(j+1));
  end
  if ((p(j)<t(j))|(p(j)>t(j+1)))
    p(j) = 0.5*(t(j)+t(j+1));
  end;
end
```



➢ Gitter für quadratischen Spline: $\mathcal{M} = \{t_0 < p_1 < t_1 < p_2 < \cdots < p_n < t_n\}$

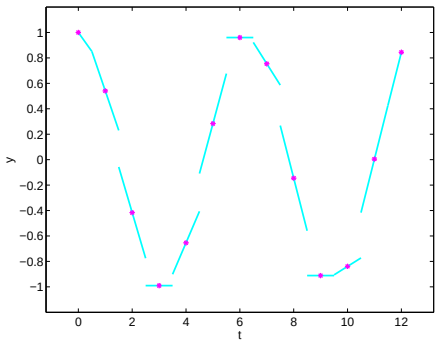
③ Setze l = Linearer Spline auf Gitter

$$\{t_0 < \tfrac{1}{2}(t_0 + p_1) < \tfrac{1}{2}(p_1 + t_1) < \tfrac{1}{2}(t_1 + p_2) < \cdots < \tfrac{1}{2}(t_{n-1} + p_n) < \tfrac{1}{2}(p_n + t_n) < t_n\} \text{ ,}$$

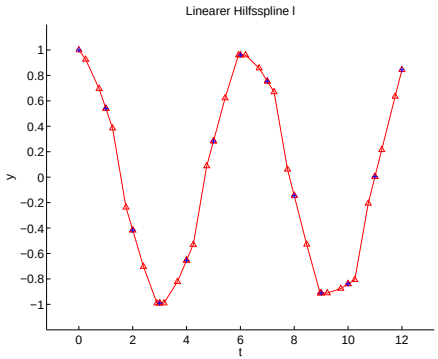
mit $l(t_i) = y_i$, $l'(t_i) = c_i$.

Beispiel 202 (Hilfskonstruktion für formerhaltende quadratische Splineinterpolation).

Datenpunkte: `t=(0:12); y = cos(t);`



Lokale Steigungen c_i, $i = 0, \ldots, n$



Linearer Hilfsspline l

◇

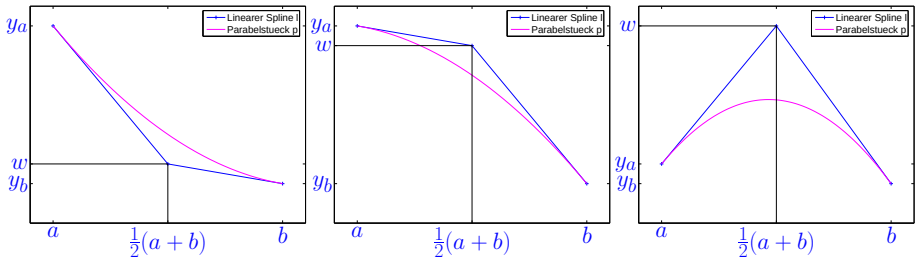
l „erbt” lokale Monotonie/Krümmung der Daten

③ Lokale quadratische Approximation/Interpolation von l:

g: Streckenzug (linearer Spline) durch (a, y_a), $(\frac{1}{2}(a+b), w)$, (b, y_b), $a < b$, $y_a, y_b, w \in \mathbb{R}$.

Parabel: $p(t) := (y_a(b-t)^2 + 2w(t-a)(b-t) + y_b(t-a)^2)/(b-a)^2$, $a \le t \le b$.

➢ $p(a) = y_a$, $p(b) = y_b$, $p'(a) = g'(a)$, $p'(b) = g'(b)$.

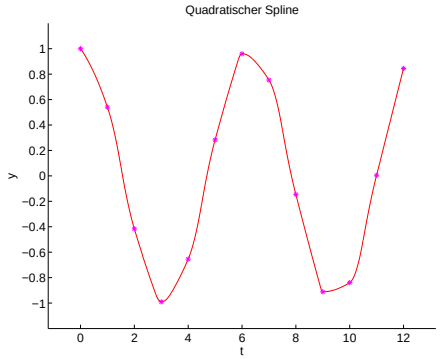


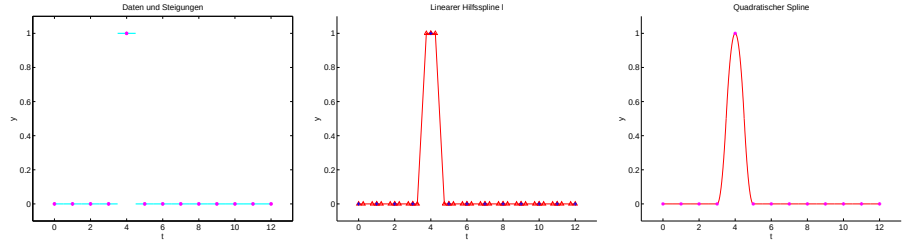
g monoton wachsend/fallend $\Rightarrow$ p monoton wachsend/fallend

g konvex/konkav $\Rightarrow$ p konvex/konkav

Fortsetzung von Bsp. 202: `ex:formspline`

Interpolierender quadratischer Spline ➢

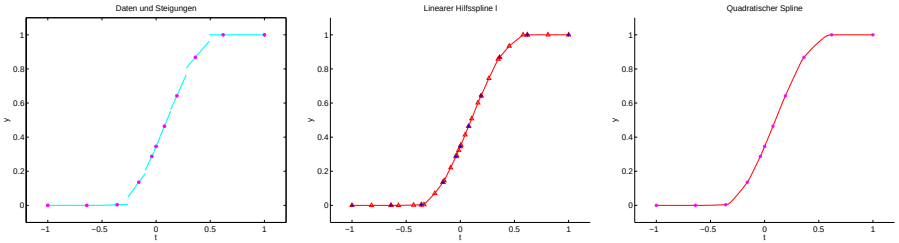




Formerhaltende quadratische Splineinterpolation = lokal + nichtlinear

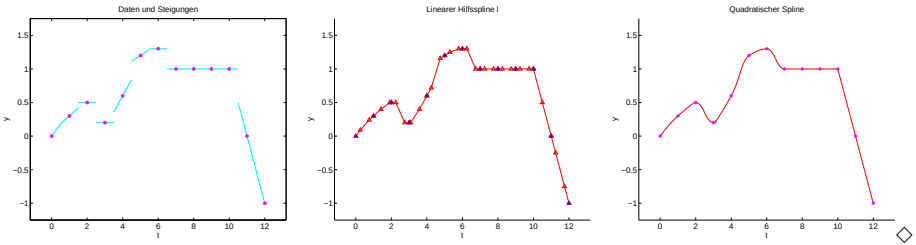
Beispiel 203 (Formerhaltende quadratische Splineinterpolation).

Daten aus Bsp. 194: ex:logpolyintpol



Daten aus [31]: MAR81

t_i	0	1	2	3	4	5	6	7	8	9	10	11	12
y_i	0	0.3	0.5	0.2	0.6	1.2	1.3	1	1	1	1	0	-1



◇

4.3.3 Bezier-Techniken

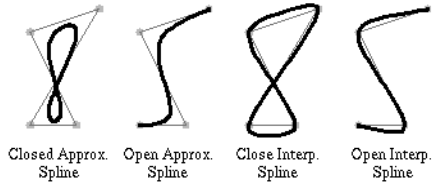
Ziel: Kurvenapproximation (**nicht** Interpolation) durch stückweise Polynome

Aus dem XFIG-Manual (`http://www.xfig.org/`):

About Spline Curves

A *Spline curve* is a smooth curve controlled by specified points.

- CLOSED APPROXIMATING SPLINE: Smooth closed curve which approximates specified points.
- OPEN APPROXIMATING SPLINE: Smooth curve which approximates specified points.
- CLOSED INTERPOLATING SPLINE: Smooth closed curve which passes through specified points.
- OPEN INTERPOLATING SPLINE: Smooth curve which passes through specified points.



Using splines, curves such as the following may be easily drawn.

Ein Beweis des Weierstrassschen Approximationssatzes (→ Analysis, [9, Sect. 6.2]): BAY73

Theorem 4.3.7 (Approximation durch Bernstein-Polynome). *Falls* $f \in C([0,1])$ *und*

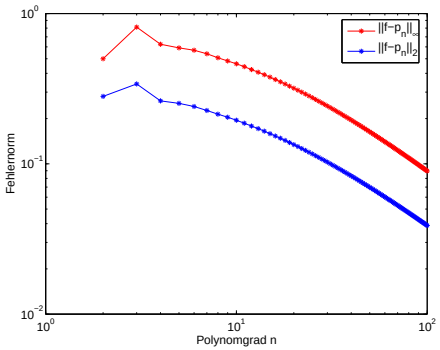
$$p_n(t) := \sum_{j=0}^{n} f(j/n) \binom{n}{j} t^j (1-t)^{n-j} , \quad 0 \le t \le 1 ,$$

dann $p_n \to f$ *gleichmässig für* $n \to \infty$. *Wenn* $f \in C^m([0,1])$, *dann* $p_n^{(k)} \to f^{(k)}$ *gleichmässig für* $n \to \infty$.

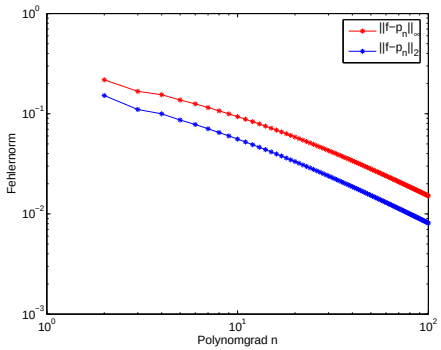
Beispiel 204 (Bernstein-Approximation).

$$f_1(t) := \begin{cases} 0 & \text{, falls } |2t-1| > \frac{1}{2} , \\ \frac{1}{2}(1+\cos(2\pi(2t-1))) & \text{sonst.} \end{cases} , \quad f_2(t) := \frac{1}{1+e^{-12(x-1/2)}} .$$

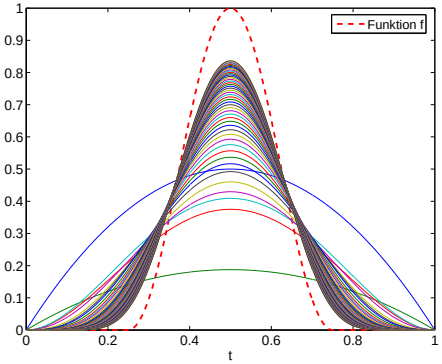
Normen des Approximationsfehlers $f - p_n$, p_n aus Thm. 4.3.7 thm:bs

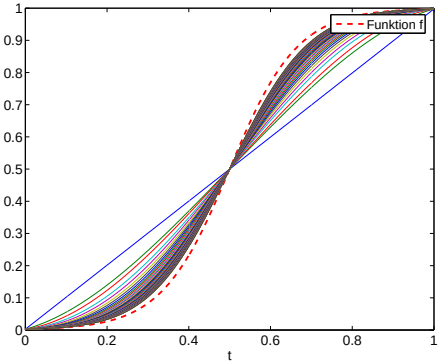


Bernstein-Approximation von f_1



Bernstein-Approximation von f_2





➢ Schlechte Approximation, aber gute „Formwidergabe” durch p_n

|DAV75

Aus [9, Sect. 6.3]: Monotonic and convex functions yield monotonic and convex approximants, respectively. In a work, the Bernstein approximants mimic the the behavior of the function to a remarkable degree. There is a price that must be paid for these beautiful approximation properties: the convergence of Bernstein polynomials is very slow.

It is far slower than what can be achieved by other means. If f is bounded, then at a point t where $f'(t)$ exists and does not vanish, $p_n(t)$ converges to $f(t)$ precisely like C/n. This fact seems to have precluded any numerical application of Bernstein polynomials from having been made (1965!). Perhaps they will find application when the properties of the approximant in the large are of more importance than closeness of the approximation.

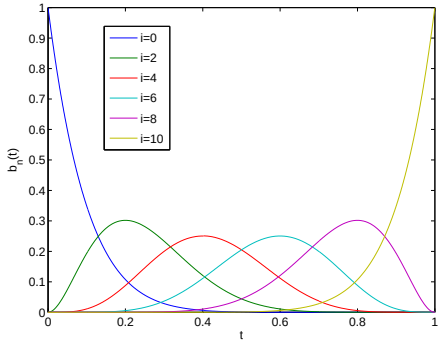
Definition 4.3.8 (Bernstein-Polynome).

Bernstein-Polynome vom Grad n:

$$b_{i,n}(t) := \binom{n}{i} t^i (1-t)^{n-i}, \quad i = 0, \ldots, n.$$

($b_{i,n} := 0$ für $i < 0$, $i > n$)

Bernstein-Polynome für $n = 10$ ➢



Lemma 4.3.9 (Eigenschaften von Bernstein-Polynomen).

(i) $t = 0$ = *i-fache Nullstelle von* $b_{i,n}$, $t = 1$ = $n - i$-*fache Nullstelle von* $b_{i,n}$,

(ii) $\sum_{i=0}^{n} b_{i,n} \equiv 1$ (*Zerlegung der Eins*),

(iii) $b_{i,n}(t) = (1-t)b_{i,n-1} + tb_{i-1,n-1}(t)$,

(iv) $b_{i,n}(t) \geq 0 \quad \forall 0 \leq t \leq 1$.

|lem:bs

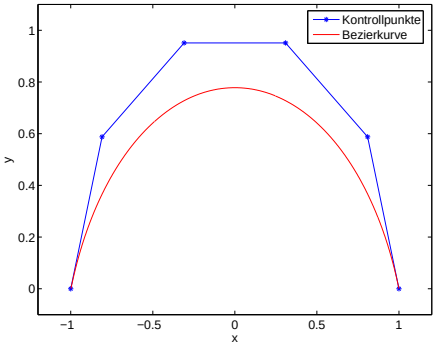
Lemma 4.3.9 (Iii)

➢ Effiziente Auswertung von $b_{i,n}(t)$ (Rekursion)

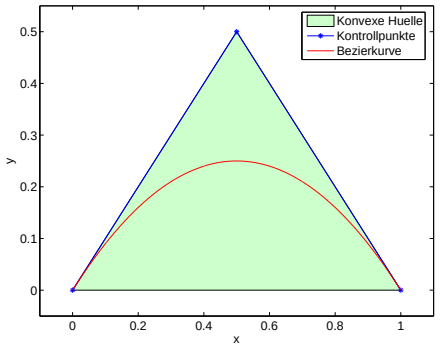
```
function V = bernstein(n,x)
V = [ones(1,length(x));...
     zeros(n,length(x))];
for j=1:n
  for k=j+1:-1:2
    V(k,:) = x.*V(k-1,:) + (1-x).*V(k,:);
  end
  V(1,:) = (1-x).*V(1,:);
end
```

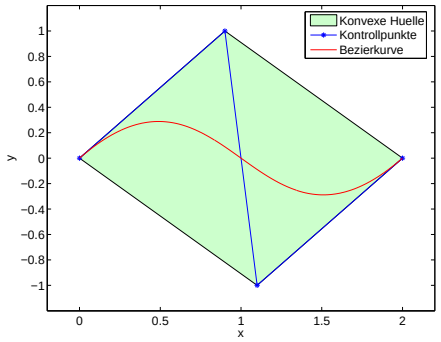
Definition 4.3.10 (Bézier-Kurven). *Bézier-Kurve zu Kontrollpunkten* $\mathbf{d}_i \in \mathbb{R}^d$, $i = 0, \ldots, n$, $d \in \mathbb{N}$:

$$\gamma : \begin{cases} [0,1] \mapsto \mathbb{R}^d \\ t \mapsto \sum_{i=0}^{n} \mathbf{d}_i \, b_{i,n}(t) \, . \end{cases}$$



- $\gamma(0) = \mathbf{d}_0$, $\gamma(1) = \mathbf{d}_n$ (Interpolationseigenschaft)
- $\gamma'(0) = n(\mathbf{d}_1 - \mathbf{d}_0)$, $\gamma'(1) = n(\mathbf{d}_n - \mathbf{d}_{n-1})$ (Tangenten)





Lemma 4.3.9 (ii) $\gamma \subset \operatorname{convex}\{\mathbf{d}_0, \ldots, \mathbf{d}_n\}$

Konvexe Hülle: $\operatorname{convex}\{d_0, \ldots, d_n\} := \{\mathbf{x} = \sum_{i=0}^{n} \lambda_i \mathbf{d}_i \, , \, 0 \le \lambda_i \le 1 \, , \, \sum_{i=0}^{n} \lambda_i = 1\}$.

Praxis:
- Stückweise Bézier-Kurven ➤ Spline-Kurven
- X-Splines & NURBS (→ CAGD, Computergraphik)

4.4 Numerische Quadratur

[File: section-numerische-quadratur.tex, SVN: section-numerische-quadratur.tex 1331 2007-03-07 09:16:14Z hiptmair]

NUMQUAD
= Näherungsweise Auswertung von $\int_\Omega f(\mathbf{x}) \, \mathrm{d}\mathbf{x}$ ($\neq$ Quadratur des Kreises)

$f : \Omega \subset \mathbb{R}^n \mapsto \mathbb{R}$ $\hat{=}$ `function y = f(x)` (Nur Punktauswertungen verfügbar)

Speziallfall $n = 1$: $\Omega = [a, b]$ (Intervall)

QUADFORM
Quadraturformel: $\int_a^b f(t) \, \mathrm{d}t \approx Q_n(f) := \sum_{j=0}^{n} \omega_j \, f(\xi_j)$.

ω_j : Quadraturgewichte $\in \mathbb{R}$ (engl. *weights*)
ξ_j : Quadraturknoten $\in [a, b]$ (engl. *nodes*)

Fragestellung: (analog zur Interpolationsfehlerabschätzung, Abschnitt 4.1.2) sec:interp

Asymptotisches Verhalten $\int_a^b f(t) \, \mathrm{d}t - Q_n(f)$ für $n \to \infty$
- ▷ Algebraische Konvergenz
- ▷ Exponentielle Konvergenz

4.4.1 Polynomiale Quadraturformeln

Idee: Ersetze $f \to p_n$, $p_n \in \mathcal{P}_n$ = Interpolationspolynom von f zu gegebener Knotenmenge $\{\xi_0, \ldots, \xi_n\} \subset [a, b]$

$$\int_a^b f(t) \, \mathrm{d}t \approx \int_a^b p_n(t) \, \mathrm{d}t \, . \qquad (4.4.1)$$ eq:polqf

Lagrange-Polynome: $L_i(t) := \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{t - \xi_j}{\xi_i - \xi_j}$, $i = 0, \ldots, n$ ➤ $p_n(t) = \sum_{j=0}^{n} f(\xi_j) L_j(t)$.

$$\int_a^b p_n(t) \, \mathrm{d}t = \sum_{j=0}^{n} f(\xi_j) \int_a^b L_j(t) \, \mathrm{d}t \quad ➤ \text{ Gewichte } \quad \omega_j := \int_a^b L_j(t) \, \mathrm{d}t \, . \qquad (4.4.2)$$ eq:quadv

Beispiel 205 (Newton-Cotes-Formeln).

Äquidistante Knoten $\nu_j := a + hj$, $h := \frac{b-a}{n}$, $j = 0, \ldots, n$:

Symbolische Berechnung der Quadraturformeln mit MAPLE:

```
> newtoncotes := n -> factor(int(interp([seq(i*h, i=0..n)],
                [seq(f(i*h), i=0..n)], z),z=0..n*h)):
```

- $n = 1$: Trapezregel

```
> trapez := newtoncotes(1);
```

$$\frac{h}{2}\,(f(0)+f(h)) \quad \left(= \frac{b-a}{2}(f(a)+f(b)) \right) \tag{4.4.3}$$

eq:trapez

- $n = 2$: Simpson-Regel

```
> simpson := newtoncotes(2);
```

$$\frac{h}{3}\,(f(0)+4\,f(h)+f(2\,h)) \quad \left(= \frac{b-a}{6}\,(f(0)+4\,f(h)+f(2\,h)) \right) \tag{4.4.4}$$

eq:simps

- $n = 4$: Milne-Regel

```
> milne := newtoncotes(4);
```

$$\frac{2}{45}\,h\,(7\,f(0)+32\,f(h)+12\,f(2\,h)+32\,f(3\,h)+7\,f(4\,h))$$

- $n = 6$: Weddle-Regel

```
> weddle := newtoncotes(6);
```

$$\frac{1}{140}\,h\,(41\,f(0)+216\,f(h)+27\,f(2\,h)+272\,f(3\,h) \\ +27\,f(4\,h)+216\,f(5\,h)+41\,f(6\,h))$$

- $n = 8$: Quadraturformel mit *negativen* Gewichten

```
> newtoncotes(8);
```

$$\frac{4}{14175}\,h\,(989\,f(0)+5888\,f(h)-928\,f(2\,h)+10496\,f(3\,h) \\ -4540\,f(4\,h)+10496\,f(5\,h)-928\,f(6\,h)+5888\,f(7\,h)+989\,f(8\,h))$$

Negative Quadraturgewichte beeinträchtigen numerische Stabilität **!**

Alternative: Falls ν_j = Tschebyscheff-Knoten ➤ Tschebyscheff-Quadratur

---

Quadraturfehlerabschätzungen ← Direkt aus L^∞-Interpolationsfehlerabschätzungen, Cor. 4.1.5:

cor:polintpe

$$f \in C^{n+1}([a,b]) \;\Rightarrow\; \left| \int_a^b f(t)\,\mathrm{d}t - Q_n(f) \right| \le \frac{1}{(n+1)!}(b-a)^{n+1} \left\| f^{(n+1)} \right\|_{L^\infty([a,b])} . \tag{4.4.5}$$

eq:polqua

(Schärfere Abschätzungen für Tschebyscheff-Quadratur)

4.4.2 Gauss-Quadratur

Nach Konstruktion: Polynomiale Quadraturformeln (4.4.1) exakt für $f \in \mathcal{P}_n$

eq:polqf

Mass für Qualität einer Quadraturformel Q_n:

$$\text{Ordnung}(q_n) := \max\{n \in \mathbb{N}_0 : \; Q_n(p) = \int_a^b p(t)\,\mathrm{d}t \quad \forall p \in \mathcal{P}_n\} + 1$$

➢ Jede polynomiale Quadraturformel mit n Knoten ▷ Ordnung $\ge n$

Geht es auch besser **?**

Beispiel 206 (Gauss-Legendre-Quadratur der Ordnung 4).

Linearität $\Rightarrow$ $Q_n(p) = \int_a^b p(t)\,\mathrm{d}t \;\forall p \in \mathcal{P}_3 \;\Leftrightarrow\; Q_n(t^q) = \frac{1}{q+1}(b^{q+1}-a^{q+1})$, $q = 0,1,2,3$.

4 Gleichungen für Gewichte ω_j, Knoten ξ_j, $j = 1,2$ $(a = -1, b = 1)$

$$\int_{-1}^{1} 1\,\mathrm{d}t = 2 = 1\omega_1 + 1\omega_2 \quad , \quad \int_{-1}^{1} t\,\mathrm{d}t = 0 = \xi_1\omega_1 + \xi_2\omega_2$$

$$\int_{-1}^{1} t^2\,\mathrm{d}t = \frac{2}{3} = \xi_1^2\omega_1 + \xi_2^2\omega_2 \quad , \quad \int_{-1}^{1} t^3\,\mathrm{d}t = 0 = \xi_1^3\omega_1 + \xi_2^3\omega_2 .$$

```
> eqns := seq(int(x^k, x=-1..1) = w[1]*xi[1]^k+w[2]*xi[2]^k,k=0..3);
> sols := solve(eqns, indets(eqns, name)):
> convert(sols, radical);
```

$$\left\{\omega_2 = 1, \omega_1 = 1, \xi_1 = 1/3\sqrt{3}, \xi_2 = -1/3\sqrt{3}\right\}$$

➤ Quadraturformel: $\int_{-1}^{1} f(x)\,dx \approx f\left(\frac{1}{\sqrt{3}}\right) + f\left(-\frac{1}{\sqrt{3}}\right)$

Heuristik: n Knoten + n Gewichte ➤ 2n Freiheitsgrade = $\dim \mathcal{P}_{2n-1}$

➤ Hoffnung: Quadraturformel der Ordnung $2n$ mit n Knoten

Herleitung: Polynomdivision mit Rest: für gegebenes $q \in \mathcal{P}_n$

$$p \in \mathcal{P}_{2n-1}: \quad p(t) = h(t)q(t) + r(t)\,, \quad h \in \mathcal{P}_{n-1}, r \in \mathcal{P}_{n-1}\,.$$

$$\int_a^b p(t)\,\mathrm{d}t - \sum_{j=1}^n \omega_j p(\xi_j) = \int_a^b h(t)q(t)\,\mathrm{d}t - \sum_{j=1}^n \omega_j h(\xi_j)q(\xi_j) + \int_a^b r(t)\,\mathrm{d}t - \sum_{j=1}^n \omega_j r(\xi_j) \stackrel{!}{=} 0\,.$$

① Wähle $q \in \mathcal{P}_n$: $\int_a^b h(t)q(t)\,\mathrm{d}t = 0 \quad \forall h \in \mathcal{P}_{n-1}$ ➤ $\int_a^b h(t)q(t)\,\mathrm{d}t = 0$

➤ q = Orthogonalpolynome zum Skalarprodukt $(f,g) \mapsto \int_a^b f(t)g(t)\,\mathrm{d}t$ → Abschnitt 4.1.5.2 (sec:least-squar)

Definition 4.4.1 (Legendre-Polynome). *Das n. Legendre-Polynom P_n ist charakterisiert durch $P_n \in \mathcal{P}_n$, $\int_{-1}^1 P_n(t)q(t)\,\mathrm{d}t = 0\ \forall q \in \mathcal{P}_{n-1}$, $P_n(1) = 1$.*

Legendre-Polynome $P_0, \ldots, P_5$ ➤

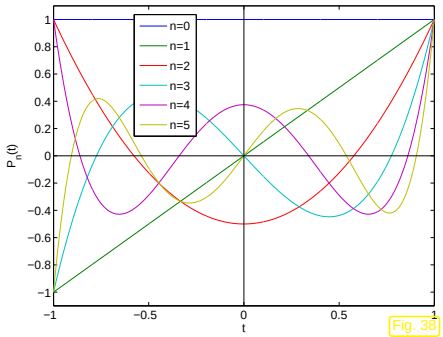


Fig. 38

legendre

Dreiterm-Rekursion analog zu Thm. 4.1.19. (thm:orthpoly)

$$P_{n+1}(t) = \frac{2n+1}{n+1}tP_n(t) - \frac{n}{n+1}P_{n-1}(t)\,. \tag{4.4.6}$$

(eq:Legpol)

```
function V = legendre(N,x)
V = ones(size(x)); V = [V; x];
for n=1:N-1
  V = [V;...
       (2*n+1)/(n+1).*x.*V(end,:)...
       - n/(n+1)*V(end-1,:)];
end
```

② Wähle Knoten $\xi_1, \ldots, \xi_n$ = auf $[a,b]$ transformierte (vgl. (4.1.17)) (eq:CHEBNODES) Nullstellen von Legendre-Polynom P_n, $n \geq 1$ (→ Gauss-Punkte) (GAUSSPOINTS).

➤ $\sum_{j=1}^n \omega_j h(\xi_j)q(\xi_j) = 0$, da $q = P_n$!

Theorem 4.4.2 (Nullstellen von Orthogonalpolynomen).
P_n hat genau n einfache Nullstellen in $[-1,1]$.

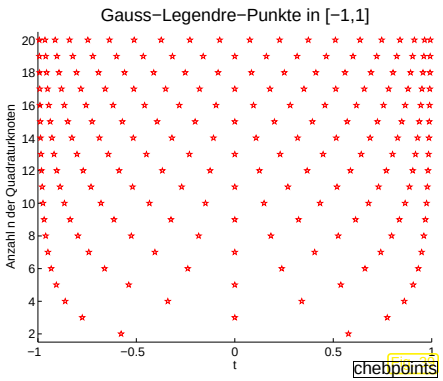


Fig. 39

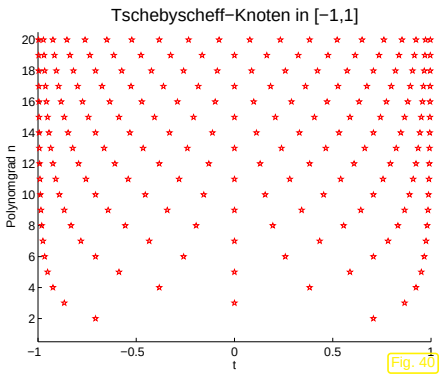


Fig. 40

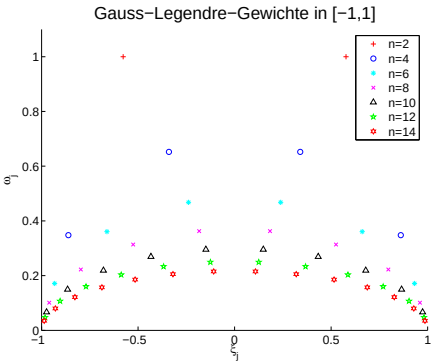
chebpoints

③ Für $[a,b] = [-1,1]$ wähle $\omega_j \in \mathbb{R}$: $\sum_{j=1}^n \omega_j \xi_j^k = \frac{1}{k+1}(1-(-1)^{k+1})$, $k = 0, \ldots, n-1$

➤ $\int_a^b r(t)\,\mathrm{d}t - \sum_{j=1}^n \omega_j r(\xi_j) = 0$, da $r \in \mathcal{P}_{n-1}$!

➤ Gauss-Legendre-Quadraturformeln: n Knoten & Ordnung $2n$

Theorem 4.4.3 (Gewichte der Gauss-Legendre-Quadraturformeln)**.** *Die Gewichte der Gauss-Legendre-Quadraturformeln sind eindeutig bestimmt und positiv.*



Bemerkung 207. Berechnung der Knoten/Gewichte der Gauss-Legendre-Quadraturformeln aus Eigenwertproblem **!**

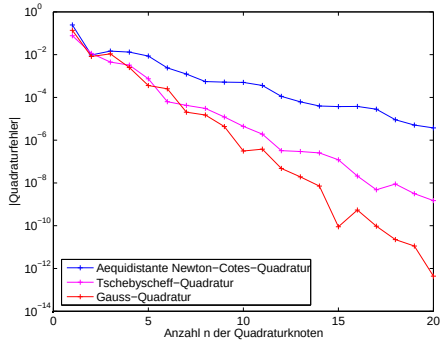
(Golub-Welsch-Algorithmus [16, Sect. 3.5.4])

GGG05

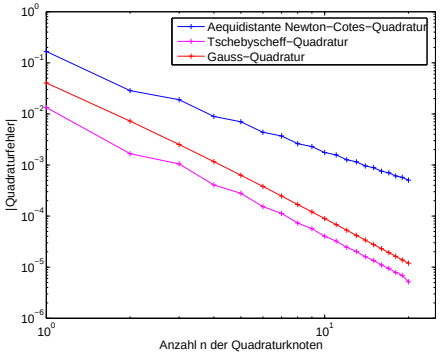
In Codes: ξ_j, ω_j tabelliert

```
function [x,w]=gaussQuad(p)
b = 1:p-1;
b = b./sqrt(4*b.*b-1);
J=diag(b,-1)+diag(b,1);
[ev,ew]=eig(J);
for i=1:p
  ev(:,i)./norm(ev(:,i));
end
x=diag(ew);
w=(2*(ev(1,:).*ev(1,:)))';
```

Beispiel 208 (Asymptotisches Verhalten des Quadraturfehlers).



Quadraturfehler, $f_1(t) := \frac{1}{1+(5t)^2}$ auf $[0,1]$



Quadraturfehler, $f_2(t) := \sqrt{t}$ auf $[0,1]$

Asymptotik des Quadraturfehlers $\epsilon_n := \left| \int_0^1 f(t)\,\mathrm{d}t - Q_n(f) \right|$ für „$n \to \infty$":

➢ Exponentielle Konvergenz $\epsilon_n \approx O(q^n)$, $0 < q < 1$, für analytischen Integranden (→ Def. 4.1.14) f_1 : Newton-Cotes-Quadratur: $q \approx 0.61$, Tschebyscheff-Quadratur: $q \approx 0.40$, Gauss-Legendre-Quadratur: $q \approx 0.27$

def:analytic

➢ Algebraische Konvergenz $\epsilon_n \approx O(n^{-\alpha})$, $\alpha > 0$, für Integranden f_2 mit Singularität in $t = 0$: Newton-Cotes-Quadratur: $\alpha \approx 1.8$, Tschebyscheff-Quadratur: $\alpha \approx 2.5$, Gauss-Legendre-Quadratur: $\alpha \approx 2.7$

Geht es noch besser als Gauss-Quadratur **?** (etwa Ordnung $2n+1$ mit n Knoten)

Zu Knoten $a \le \xi_1 < \xi_2 < \ldots < \xi_n \le b$ ➤ $p(t) := \prod_{j=1}^{n} (t - \xi_j)^2 \in \mathcal{P}_{2n}$

◥◤ $Q_n(p) = 0$ aber $\int_a^b p(t)\,\mathrm{d}t > 0$.

➢ Ordnung $> 2n$ kann mit n Quadraturpunkten nicht erreicht werden

4.4.3 Zusammengesetzte Quadraturformeln

Analogon: Globale Polynominterpolation ⟷ stückweise Polynominterpolation

(→ Abschnitt 4.3.1)

sec:stuckw-polyn

Idee:
- Partitionierung des Integrationsintervalls $[a,b]$ durch Gitter (→ Abschnitt 4.3) $\mathcal{M} := \{a = x_0 < x_1 < \ldots < x_m = b\}$
- Anwendung der Quadraturformeln aus Abschnitt 4.4.1 auf Teilintervalle $I_j := [x_{j-1}, x_j]$, $j = 1, \ldots, m$, + Summation.

(Wir betrachten nur zusammengesetzte Quadraturformeln basierend auf einer einzigen lokalen Quadraturformel)

Konvergenzuntersuchung (Skalierungsargument, vgl. (4.1.16)) und (4.4.5)):

Asymptotik des Quadraturfehlers bzgl. Maschenweite h von $\mathcal{M}$:

Theorem 4.4.4 (Konvergenz zusammengesetzter Quadraturformeln). *Für eine zusammengesetzter Quadraturformel Q basierend auf einer lokalen Quadraturformel der Ordnung m gilt*

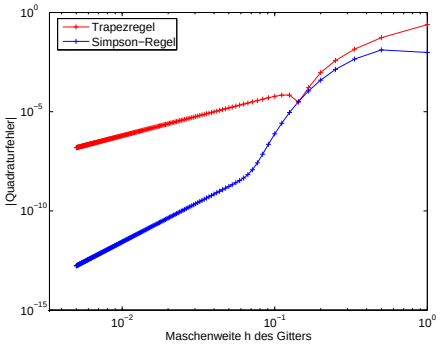
$$\exists C > 0: \quad \left| \int_I f(t)\,\mathrm{d}t - Q(f) \right| \leq C h^m \left\| f^{(m)} \right\|_{L^\infty(I)} \quad \forall f \in C^m(I), \forall \mathcal{M} .$$

Beispiel 209 (Quadraturfehler bei zusammengesetzten Regeln).

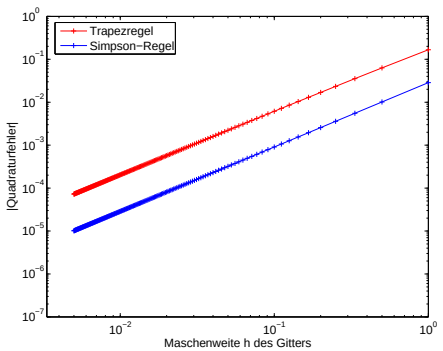
Zusammengesetzte Quadraturformeln basierend auf

- Trapezregel (4.4.3) ➢ Lokale Ordnung 2 (exakt für lineare Funktionen),
- Simpson-Regel (4.4.4) ➢ Lokale Ordnung 3 (exakt für Parabeln)

auf äquidistantem Gitter $\mathcal{M} := \{jh\}_{j=0}^n$, $h = 1/n$, $n \in \mathbb{N}$.



Quadraturfehler, $f_1(t) := \frac{1}{1+(5t)^2}$ auf $[0,1]$



Quadraturfehler, $f_2(t) := \sqrt{t}$ auf $[0,1]$

Asymptotik des Quadraturfehlers $\epsilon_n := \left| \int_0^1 f(t)\,\mathrm{d}t - Q_n(f) \right|$ für Maschenweite „$h \to 0$"

☛ Algebraische Konvergenz $\epsilon = O(h^\alpha)$, $\alpha > 0$:

➢ Hinreichend glatter Integrand f_1: α gemäss Thm. 4.4.4, Trapezegel → $\alpha = 2$, Simpson-Regel → $\alpha = 4$!?

➢ Singulärer Integrand f_2: $\alpha = 3/2$ für Trapezegel **&** Simpson-Regel **!** (Glattheit des Integranden begrenzt Konvergenz)

Simpson-Regel: Ordnung = 4 **?** MAPLE schafft Klarheit

```
> rule := 1/3*h*(f(2*h)+4*f(h)+f(0))
> err := taylor(rule - int(f(x),x=0..2*h),h=0,6);
```

$$err := \left(\frac{1}{90} \left(D^{(4)} \right)(f)(0)\, h^5 + O\left(h^6\right), h, 6 \right)$$

➢ Simpson-Regel hat sogar Ordnung 4 **!**

◇

Bemerkung 210 (Auflösung einer Singularität durch Transformation).

Für $f \in C^\infty([0,b])$ approximiere $\int_0^b \sqrt{t} f(t)\,\mathrm{d}t$ durch numerische Quadratur (→ Bsp. 209)

Substitution $s = \sqrt{t}$: $\int_0^b \sqrt{t} f(t)\,\mathrm{d}t = \int_0^{\sqrt{b}} 2s^2 f(s^2)\,\mathrm{d}s$.

Dann: Anwendung einer Quadraturformel auf glatten Integranden

Die äquidistante Trapezregel:

$\hat{=}$ stückweise lineare Approximation des Integranden auf äquidistantem Gitter:

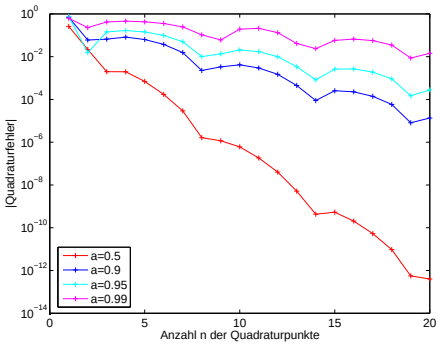
$$\int_a^b f(t)\,\mathrm{d}t \approx T_n(f) := h\left(\tfrac{1}{2} f(a) + \sum_{k=1}^{n-1} f(kh) + \tfrac{1}{2} f(b) \right), \quad h := \frac{b-a}{n} .$$

Beispiel 211 (Konvergenz für äquidistante Trapezregel).

1-periodischer analytischer Integrand (→ Bsp. 166):

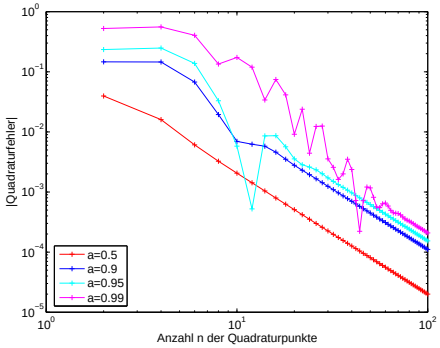
$$f(t) = \frac{1}{\sqrt{1 - a\sin(2\pi t - \gamma)}}, \quad 0 < a < 1, \gamma \in \mathbb{R} .$$

(„Exakte Integralwerte" aus T_{500})



Quadraturfehler für $T_n(f)$, $\gamma = 1$, auf $[0,1]$

Exponentielle Konvergenz



Quadraturfehler für $T_n(f)$, $\gamma = 1$, auf $[0, \frac{1}{2}]$

Nur algebraische Konvergenz

◇

Erklärung:

$$f(t) = e^{2\pi i k t} \quad \blacktriangleright \quad \begin{cases} \int_0^1 f(t)\,\mathrm{d}t = \begin{cases} 0 & \text{, falls } k \neq 0\,, \\ 1 & \text{für } k = 0\,. \end{cases} \\ T_n(f) = \frac{1}{n}\sum_{l=0}^{n-1} e^{\frac{2\pi i}{n} l k} = \begin{cases} 0 & \text{, falls } k \notin n\mathbb{Z}\,, \\ 1 & \text{, falls } k \in n\mathbb{Z}\,. \end{cases} \end{cases} \quad \text{eq:comproot2} \quad (3.7.2)$$

$\blacktriangleright$

Äquidistante Trapezregel T_n ist exakt für trigonometrische Polynome (→ Def. 4.1.8) vom Grad $< 2n$ **!**

def:tp

$\blacktriangleright$

$T_n \mathrel{\hat{=}}$ Approximation des Integranden durch trigonometrisches Interpolationspolynom → Abschn. 4.1.4

sec:trig-interp

$\blacktriangleright$ Glatte periodische Integranden ➢ Benutze äquidistante Trapezregel **!**

Übung 4.5. Was versteht man unter Tschebyscheff-Quadratur? Gib die Quadraturknoten an und bestimmte numerisch eine Naeherung für die Quadraturgewichte unter Verwendung der MATLAB-Function `intpolyval(t,y,x)` aus Abschnitt 4.1.1.2 und der standard MATLAB-Funktion `quad`.

sec:algorithmen

Übung 4.6. Die Integralwerte

$$\int_0^1 \sqrt{1-x^2} f(x)\,\mathrm{d}x \qquad (4.6.1)$$

eq:logqua

sollen effizient, d.h. mit möglichst wenig Auswertungen von f, numerisch approximiert werden. Dabei darf $f \in C^\infty([0,1])$ angenommen werden.

1. Wie ist das Integral vor Anwendung einer Quadraturformel geeignet zu transformieren? (Hinweis: Verwende $1 - x^2 = (1-x)(1+x)$)
2. Schreibe eine auf Gauss-Quadratur basierende effiziente MATLAB-Funktion `logquad(f,p)` (`f` $\hat{=}$ handle auf Funktion `f`, $2p \hat{=}$ Ordnung der zugrundeliegenden Gauss-Quadratur) zur Approximation von (4.6.1). Die Routine `gaussQuad(p)` aus Bemerkung der Vorlesunsunterlagen darf verwendet werden.

eq:logquad

3. Für $f(x) = \sqrt{1+x}$ plotte den Quadraturfehler in Abhängigkeit von p für $p = 1, \ldots, 20$. Welche qualitative Konvergenz kann abgelesen werden?

4.6.1 Adaptive Quadratur

Betrachte zusammengesetzte Trapezregel auf Gitter $\mathcal{M} := \{a = x_0 < x_1 < \cdots < x_m = b\}$

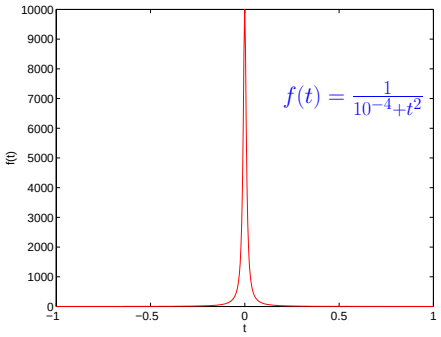
Lokaler Quadraturfehler (für $f \in C^2([a,b])$):

$$\int_{x_{k-1}}^{x_k} f(t)\,\mathrm{d}t - \frac{1}{2}(f(x_{k-1}) + f(x_k)) \leq (x_k - x_{k-1})^3 \left\| f'' \right\|_{L^\infty([x_{k-1},x_k])}\,.$$

➢ Benutze nichtäquidistante Gitter **!**

(fein, wo $|f''|$ gross)

Sinnvoll etwa für „Nadelfunktion" ➢



Ziel:	Gleiche Beiträge aller Gitterintervalle zum Gesamtfehler
Werkzeug:	APEST Lokale a posteriori Fehlerschätzung (Schätzen der Fehlerbeiträge von Gitterintervallen aus Zwischenergebnissen)
Technik:	Lokale Gitterverfeinerung

Adaptive Mehrgitterquadratur (vereinfacht nach [12, Abschnitt 9.7])

Idee: Vergleich der Resultate zweier lokaler Quadraturformeln Q_1, Q_2 unterschiedlicher Ordnung → lokale Fehlerschätzung

Heuristik: $\text{Fehler}(Q_2) \ll \text{Fehler}(Q_1) \quad \Rightarrow \quad \text{Fehler}(Q_1) \approx Q_2(f) - Q_1(f)$.

Hier: Trapezregel ↔ Simpson-Regel

Gegeben: Gitter $\mathcal{M} := \{a = x_0 < x_1 < \cdots < x_m = b\}$

❶ (Fehlerschätzung) Für $I_k = [x_{k-1}, x_k]$, $k = 1, \ldots, m$ ($m_k := \frac{1}{2}(x_{k-1} + x_k)$)

$$\texttt{EST}_k := \underbrace{\frac{h_k}{6}(f(x_{k-1}) + 4f(m_k) + f(x_k))}_{\text{Simpson-Regel}} - \underbrace{\frac{h_k}{4}(f(x_{k-1}) + 2f(m_k) + f(x_k))}_{\text{Trapezregel (auf unterteilter Gitterzelle)}}$$

❷ (Abbruch) Simpson-Regel auf $\mathcal{M}$ ⇒ Vorläufiger Integralwert I

Wenn $\sum_{k=1}^{m} \texttt{EST}_k \leq \texttt{TOL} \cdot I$ (TOL := vorgegebene Toleranz) ⇒ **STOP**

❸ (Lokale Gitterverfeinerung)

$$\mathcal{S} := \{k \in \{1, \ldots, m\} : \texttt{EST}_k \geq \eta \cdot \frac{1}{m} \sum_{j=1}^{m} \texttt{EST}_j\} , \quad \eta \approx 0.9 .$$

Neues Gitter $\mathcal{M}^* := \mathcal{M} \cup \{m_k : k \in \mathcal{S}\}$.

Dann weiter mit ❶ und $\mathcal{M} \leftarrow \mathcal{M}^*$.

Adaptive Quadratur in MATLAB (nach [17]):

`q = quad(fun,a,b,tol)`: Adaptive Mehrgitterquadratur (lokale Quadraturformeln niedriger Ordnung)

`q = quadl(fun,a,b,tol)`: Adaptive Gauss-Quadratur (lokale Gauss-Quadraturformeln & Ordnungserhöhung)

4.6.2 Numerische Berechnung oszillatorischer Integrale

4.7 Multiskalenbasen

[File: section-multiskalenbasen.tex, SVN: section-multiskalenbasen.tex 1010 2006-09-11 15:44:43Z hiptmair]

Ziel: Analyse von Daten (↔ Funktionen) durch Darstellung in geeigneter Basis

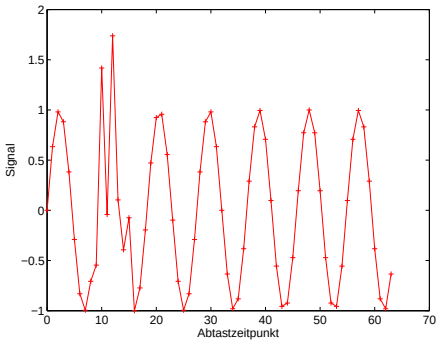
Beispiel 212 (Nichtlokalität der DFT).

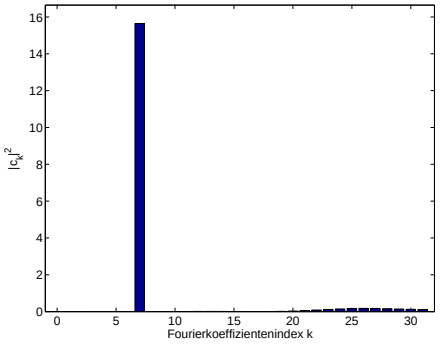
Diskrete Fouriertransformation (→ Abschn. 3.7.1), Bsp. 145)

- Gute Identifikation von Frequenzanteilen
- Keine Auflösung zeitlokaler Effekte

Daten:

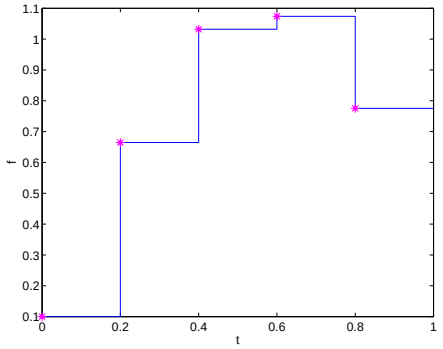
```
t = 0:63; y = sin(7*2*pi*t/64);
y(10:16) = y(10:16) + cos(10*t(10:16));
```





Gegeben: Abtastzeitpunkte $t_j := j/n$, $j = 0, \ldots, n-1$, $n \in \mathbb{N}$ (äquidistant), Abtastwerte y_j, $j = 0, \ldots, n-1$ ⇔ Vektor $\mathbf{y} \in \mathbb{R}^n$

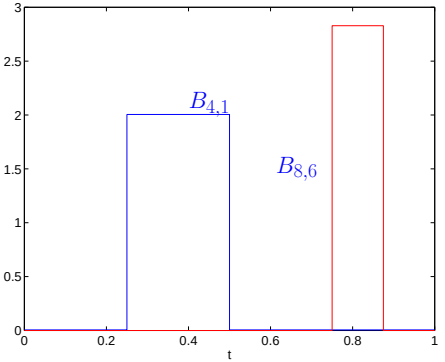
Vektor $\mathbf{y} \in \mathbb{R}^n$



Assoziierte Treppenfunktion

$$f(t) = y_j \quad \text{für } t_j \le t < t_{j+1} .$$

(mit $t_n := 1$)



$$f(t) = \sum_{j=0}^{n-1} \left(\frac{1}{\sqrt{n}} y_j\right) B_{n,j}(t) ,$$

$$B_{n,j}(t) := \sqrt{n} B(nt - j) ,$$

$$B(t) := \begin{cases} 1 & \text{, falls } 0 \le t < 1 , \\ 0 & \text{sonst.} \end{cases}$$

Beachte:
- $\operatorname{supp}(B(n \cdot -j)) = [j/n, j+1/n]$
- $\|B_{n,j}\|_{L^2([0,1])} = 1$

$B \mathrel{\hat{=}}$ „Grundbasisfunktion" (Skalierungsfunktion)

Terminologie:

$$B(nt - j) = B(n(t - \frac{j}{n}))$$

Dilatation — Translation um j/n

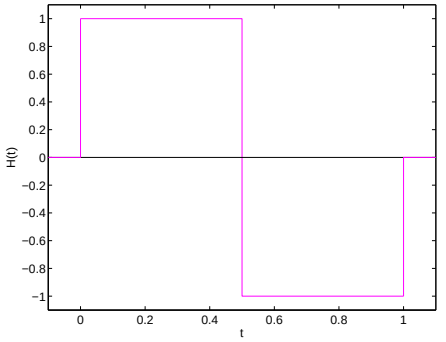
$\rightarrow$ $\{B_{n,j}\}_{j=0}^{n-1}$ = L^2-orthonormale Basis des Treppenfunktionsraums $\mathcal{C}_n$ auf äquidistantem Gitter $\mathcal{T} := \{j/n,\ j = 0, \dots, n\}$ in $[0, 1]$.

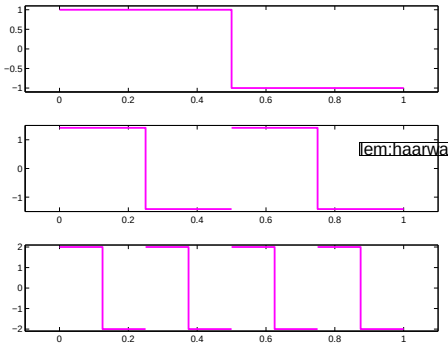
Beachte: $\mathcal{C}_n = \mathcal{S}_{0,\mathcal{T}}$ (Splineraum niedrigster Ordnung, → Def. 4.3.5)

Eine weitere L^2-Orthonormalbasis (für $n = 2^L$, $L \in \mathbb{N}$):

Definition 4.7.1 (Haar-Wavelet). *Haar-Wavelet auf* $[0, 1]$:

$$H(t) := \begin{cases} 1 & \text{für } 0 \le t < \frac{1}{2} , \\ -1 & \text{für } \frac{1}{2} \le t < 1 , \\ 0 & \text{sonst.} \end{cases}$$
$$= B(2t) - B(2t - 1) .$$





Translierte/dilatierte Haar-Wavelets:

$$H_{l,j}(t) := 2^{l/2} H(2^l t - j) , \quad j = 0, \dots, l - 1 .$$

lem:haarwav

Lemma 4.7.2.

$$\bigcup_{l=0}^{L-1} \bigcup_{j=0}^{2^l-1} \{H_{l,j}\} \cup \{B\}$$

ist eine L^2*-Orthonormalbasis von* $\mathcal{C}_{2^L}$*, die Haar-Waveletbasis.*

HWFBAS

$\rightarrow$ Waveletdarstellung von $f \in \mathcal{C}_{2^L}$: $f = c_0 B + \sum_{l=0}^{L-1} \sum_{j=0}^{2^l-1} c_{l,j} H_{l,j}$.

Terminologie: $c_{l,j}$ = Waveletkoeffizienten

$\sum_{j=0}^{2^l-1} c_{l,j} H_{l,j}$ = Level l der Waveletzerlegung

Folgerung aus Orthonormalbasiseigenschaft:

$$\sum_{j=0}^{n-1} y_j B_{n,j} = c_0 B + \sum_{l=0}^{L-1} \sum_{j=0}^{2^l-1} c_{l,j} H_{l,j} \quad \Rightarrow \quad \sum_{j=0}^{n-1} y_j^2 = c_0^2 + \sum_{l=0}^{L-1} \sum_{j=0}^{2^l-1} c_{l,j}^2 \,.$$

$\longleftrightarrow$ Unitarität der (skalierten) diskreten Fouriertransformation ($\rightarrow$ Lemma 3.7.2) lem:dft

---

Algorithmen zur Basistransformation Boxfunktionsbasis $\leftrightarrow$ Haar-Waveletbasis:

Verfeinerungsrelationen:

$$\begin{aligned} B_{2^l,j} &= \tfrac{1}{2}\sqrt{2}\left(B_{2^{l+1},2j} + B_{2^{l+1},2j+1}\right), \\ H_{l,j} &= \tfrac{1}{2}\sqrt{2}\left(B_{2^{l+1},2j} - B_{2^{l+1},2j+1}\right), \end{aligned} \quad \Leftrightarrow \quad \begin{aligned} B_{2^{l+1},2j} &= \tfrac{1}{2}\sqrt{2}\left(B_{2^l,j} + H_{l,j}\right), \\ B_{2^{l+1},2j+1} &= \tfrac{1}{2}\sqrt{2}\left(B_{2^l,j} - H_{l,j}\right). \end{aligned} \quad (4.7.1)$$ eq:refrel

$$g = y_1 B_{2^{l+1},2j} + y_2 B_{2^{l+1},2j+1} \quad \Rightarrow \quad g = \tfrac{1}{2}\sqrt{2}\left((y_1+y_2) B_{2^l,j} + (y_1-y_2) H_{l,j}\right) \quad (4.7.2)$$ eq:filter

Tiefpassfilter Hochpassfilter

Lineare Anordnung der Waveletkoeffizienten $c_{l,j}$ für $f = c_0 B + \sum_{l=0}^{L-1}\sum_{j=0}^{2^l-1} c_{l,j} H_{l,j}$:

$$c_0, c_{0,0}, c_{1,0}, c_{1,1}, c_{2,}, \ldots, c_{2,3}, \ldots, c_{l,0}, \ldots, c_{l,2^l-1}, \ldots, c_{L-1,0}, \ldots, c_{L-1,2^{L-1}-1}\,. \quad (4.7.3)$$ eq:cord

Boxfunktionsbasis $\rightarrow$ Haar-Waveletbasis:

MATLAB-CODE : Basistransformation

```
function c = boxbastohwf(y,L)
c = zeros(size(y)); c(1) = y(1);
for l=2.^(L-1:-1:0)
  y = y/sqrt(2); m = 2*l;
  c(l+1:m)=y(1:2:m)-y(2:2:m);
  y(1:l) = y(1:2:m)+y(2:2:m);
end
```

Rekursiver *bottom-up* Algorithmus:
(für `length(y)` $= 2^L$)
(Sukzessive Berechnung der Waveletkoeffizienten auf Level l und Boxfunktionsdarstellung auf Level $l-1$)

Rechenaufwand $O(2^L)$

Haar-Waveletbasis $\rightarrow$ Boxfunktionsbasis:

MATLAB-CODE : Basistransformation

```
function y = hwftoboxbas(c,L)
y = zeros(size(c));
y(1) = c(1); ofs = 1;
for l=2.^(0:L-1)
  z = y(1:l); w=c(ofs+(1:l));
  y(1:2:2*l) = (z+w)/sqrt(2);
  y(2:2:2*l) = (z-w)/sqrt(2);
  ofs = ofs+l;
end
```

Rekursiver *top-down* Algorithmus:
(für `length(y)` $= 2^L$)
(Sukzessive Berechnung der Boxfunktionsdarstellung auf Level $0, 1, 2, \ldots$)

Rechenaufwand $O(2^L)$

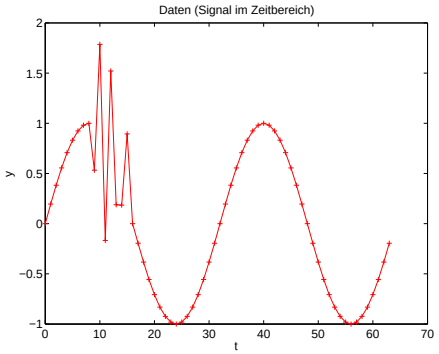
Beispiel 213 (Waveletzerlegung eines Signals). Wavelet!Zerlegung eines Signals

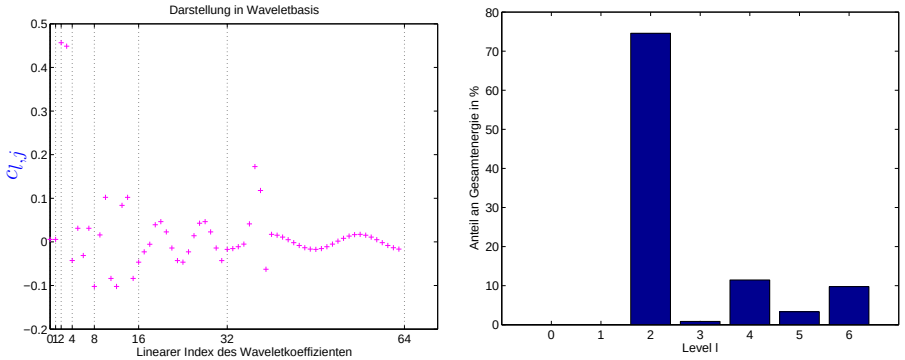
Signal ($\rightarrow$ Bsp. 2.1.2): ex:dftnonloc

```
t = 0:63; y = sin(4*pi*t/64);
y(10:16) = y(10:16) +
cos(10*t(10:16));
```

Signal mit

- dominierendem Grundfrequenzanteil
- zeitlokaler Störung

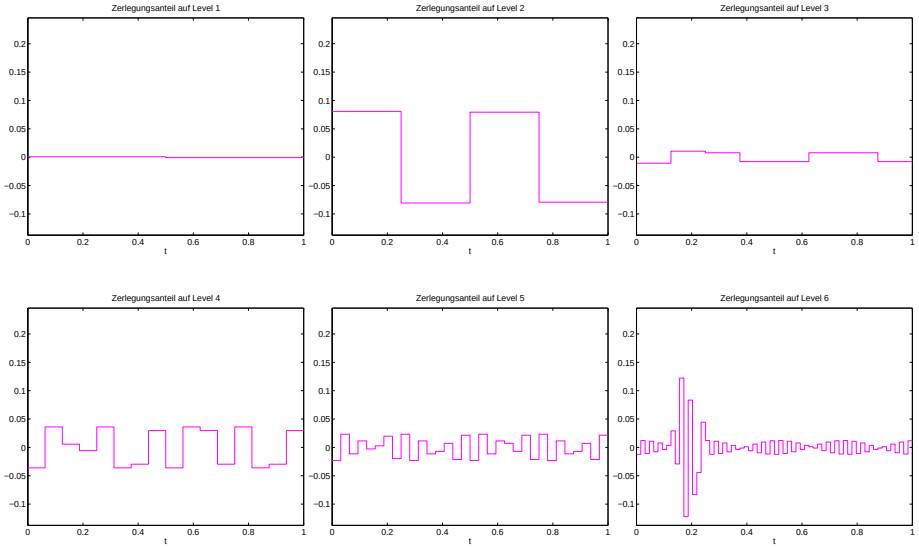




Leistung auf Level l: $P_l := \sum_{j=0}^{2^l-1} c_{l,j}^2$ ➤ $\sum_{l=0}^{L} P_l = \|\mathbf{y}\|_2^2$

➢ Auflösung der „Grössenordnung” der Grundfrequenz im Leistungsspektrum
(→ Koeffizientenindex l ↔ Frequenzzerlegung)

Wavelet-Zerlegungsanteile:



➢ Zeitliche Auflösung der Störung (→ Koeffizientenindex k)

Raum der Treppenfunktionen = Splineraum vom Grad 0/Ordnung 1 Verallgemeinerung ?

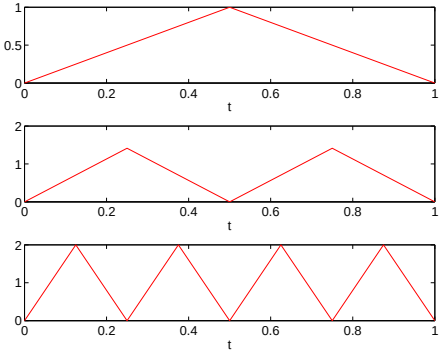
➤ Spline-Wavelets [7] (Komplizierte Konstruktionen)
(Herausforderung: Kleine Träger, Orthogonalität ↔ Stabilität)

Hierarchische Basis für $\mathcal{S}_{1,\mathcal{M}}$:

Äquidistante Knoten in $[0,1]$: $\mathcal{M} := \{t_j := j/n, j = 0, \ldots, n\}$, $n = 2^L$, $L \in \mathbb{N}$

Skalierungsfunktion („Hutfunktion”) $\varphi(t) = \begin{cases} 1-t & \text{, falls } 0 \le t \le 1 \text{ ,} \\ 1+t & \text{, falls } -1 \le t \le 0 \text{ ,} \\ 0 & \text{sonst.} \end{cases}$ $\in C^0(\mathbb{R})$, stückweise linear.

$\varphi_{l,n}(t) := 2^{l/2}\varphi(2^l t - j)$ ⇒ $\{\varphi_{L,j}\}_{j=0}^{2^L}$ = Basis von $\mathcal{S}_{1,\mathcal{M}}$ auf $[0,1]$ (Knotenbasis)



Hierarchische Basis von $\mathcal{S}_{1,\mathcal{M}}$:

$$\{\varphi_{0,0}, \varphi_{0,1}\} \cup \bigcup_{l=1}^{L} \bigcup_{j=0}^{2^{l-1}-1} \{\varphi_{l,2j+1}\} \,. \quad (4.7.4)$$ eq:hierba

Hierarchische-Basis-Zerlegung:

$$s \in \mathcal{S}_{1,\mathcal{M}} \;\Rightarrow\; s = c_0\,\varphi_{0,0} + c_1\,\varphi_{0,1} + \sum_{l=1}^{L} \sum_{j=0}^{2^{l-1}-1} c_{l,j}\,\varphi_{l,2j+1} \,.$$

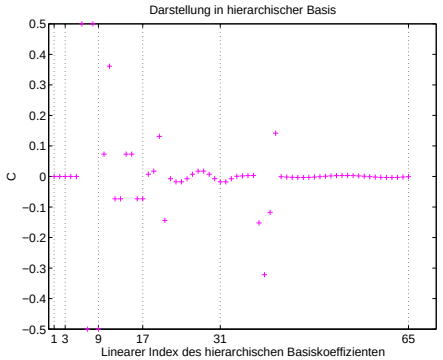
Beispiel 214 (Hierarchische-Basis-Zerlegung).

$s \in \mathcal{S}_{1,\mathcal{M}}$, Knotenwerte $s(t_j)$ aus

```
t = 0:63; y = sin(4*pi*t/64);
y(10:16) = y(10:16) +
cos(10*t(10:16));
```

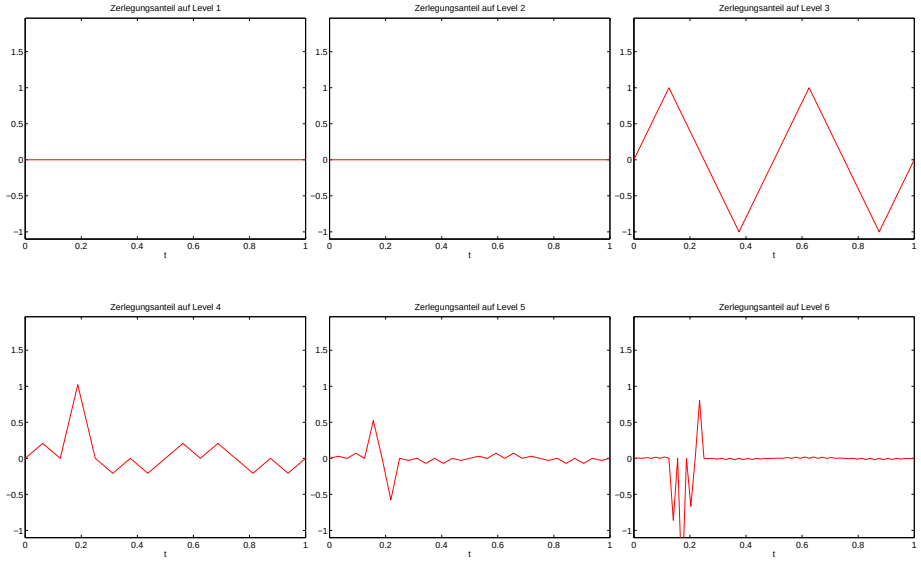
Signal aus → Bsp. 213 ex:wavsignal

Hierarchische-Basis-Koeffizienten $c_{l,j}$ (Anordnung wie in (4.7.3)) eq:cord



➢ Auflösung von Grundschwingungen und zeitlokalen Merkmalen

Hierarchische-Basis-Zerlegungsanteile:

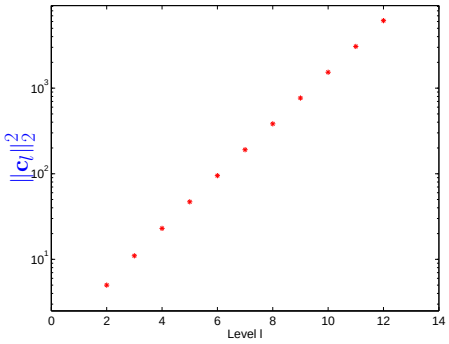


Beispiel 215 (L^2-Instabilität der hierarchischen Basis).

Auf Gitter $\mathcal{M}_l := \{j/n, j = 0, \ldots, n\}$, $n := 2^l$ wähle

$$s_l := \varphi_{l,2^{l-1}} .$$

Euklidische Norm der hierarchischen-Basis-Koeffizienten (4.7.4) von s_l eq:hierbasdec



➢ $\frac{\|\mathbf{c}_l\|_2^2}{\|\mathbf{c}_l\|_2^2}$

$$\|\mathbf{c}\|_2^2 := c_0^2 + c_1^2 + \sum_{l=1}^{L} \sum_{j=0}^{2^{l-1}-1} c_{l,j}^2 .$$

Exponentielles Wachstum von $\|\mathbf{c}_l\|_2^2$!

➢ Hierarchische Basis *nicht* l*-gleichmässig stabil* (im L^2-Sinn)

5 Numerik Gewöhnlicher Differentialgleichungen

Gegeben:
- Phasenraum (engl. *phase space*) $\Omega \subset \mathbb{R}^d$, $d \in \mathbb{N}$ ($\mathbf{y} \in \Omega$ = „Zustand”, engl. *state*)
  ➢ Erweiterter Phasenraum $\hat{\Omega} \subset \mathbb{R} \times \Omega$ „Zeitvariable”
  (Annahme: Tensorproduktstruktur $\hat{\Omega} = I \times \Omega$)
- *Stetige* Funktion $f : \hat{\Omega} \mapsto \mathbb{R}^d$ („rechte Seite”)
- Anfangsdaten $(t_0, \mathbf{y}_0) \in \hat{\Omega}$, ($\mathbf{y}_0$ = Anfangswert engl. *initial value*)

Gesucht: C^1-Lösungsfunktion $\mathbf{y} : J \mapsto \Omega$, $J \subset \mathbb{R}$ Intervall, $t_0 \in J$, für das Anfangswertproblem (AWP) für die gewöhnliche Differentialgleichung (engl. *ordinary differential equation*, ODE) [1. Ordnung]:

$$\mathbf{y}(t_0) = \mathbf{y}_0 \quad , \quad \dot{\mathbf{y}} = f(t, \mathbf{y}(t)) \quad \forall t \in J .$$

Notation: Punkt $\dot{} \mathrel{\hat{=}}$ „Zeitableitung” $\frac{\mathrm{d}}{\mathrm{d}t}$

Mechanik
Reaktionskinetik
Populationsdynamik
Schaltkreismodelle
...
} ⇒ Modellierung durch gewöhnliche Differentialgleichungen

Beispiel 216 (Mechanisches System Pendel).

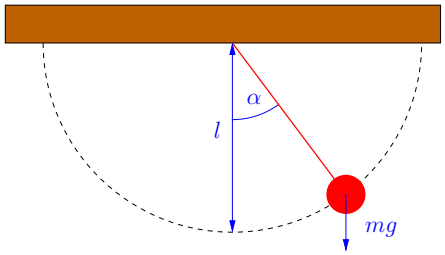
Phasenraum Ω = Konfigurationsraum für Minimal-koordinaten (= Auslenkungswinkel)

➢ $d = 1, \Omega =]-\pi/2, \pi/2[$
(→ zulässige Auslenkungswinkel α)

Newtonsche Bewegungsgleichungen:

$$ml\,\ddot{\alpha}(t) = -mg\sin\alpha(t)\ .$$

► ODE 2. Ordnung



Formale Umwandlung in gewöhnliche Differentialgleichungen 1. Ordnung (Hamiltonsche Form der Bewegungsgleichungen → Physik):

$$p := \dot{\alpha} \quad \Rightarrow \quad \begin{pmatrix} \dot{\alpha} \\ \dot{p} \end{pmatrix} = \begin{pmatrix} p \\ -\frac{g}{l}\sin\alpha \end{pmatrix}\ .$$

5.1 Theorie gewöhnlicher Differentialgleichungen

[File: section-theorie-gewoehnlicher-differentialgleichungen.tex, SVN: section-theorie-gewoehnlicher-differentialgleichungen.tex 1010 2006-09-11 15:44:43Z h

Gegeben: $f : \hat{\Omega} \subset \mathbb{R} \times \Omega \mapsto \mathbb{R}^d$ stetig, Anfangsdaten $(t_0, \mathbf{y}_0) \in \hat{\Omega}$

► AWP: $\mathbf{y}(t_0) = \mathbf{y}_0 \quad , \quad \dot{\mathbf{y}}(t) = f(t, \mathbf{y}(t))\ .$ (5.1.1) eq:awp

Terminologie: AWP AUTONOM autonom ⇔ $f(t, \mathbf{y}) = f(\mathbf{y})$

Definition 5.1.1. *$J \subset \mathbb{R}$ offenes Intervall, $t_0 \in J$*

$y \in C^1(J, \Omega)$ *Lösung des AWP* (5.1.1) eq:awp :⇔ $\mathbf{y}(t_0) = \mathbf{y}_0 \quad , \quad \dot{\mathbf{y}}(t) = f(t, \mathbf{y}(t)) \quad \forall t \in J\ .$

Annahme: (Lokale Lipschitz-Stetigkeit)
$\forall (t, \mathbf{y}) \in \hat{\Omega}$: $\exists L > 0, \delta > 0$: $|t - \widetilde{t}| + \|\mathbf{y} - \widetilde{\mathbf{y}}\| < \delta \quad \Rightarrow \quad \left\|f(t, \mathbf{y}) - f(\widetilde{t}, \widetilde{\mathbf{y}})\right\| \le L \|\mathbf{y} - \widetilde{\mathbf{y}}\|$

Kriterium: $\frac{\mathrm{d}}{\mathrm{d}\mathbf{y}} f$ stetig auf $\hat{\Omega}$ (→ Kompaktheitsargument)

„Die Lösung eines Anfangswertproblems sucht sich Ihren Definitionsbereich selbst"

Lösung $y \in C^1([t_0, t_1[, \Omega)$ von AWP (5.1.1) eq:awp maximal (in die Zukunft) fortsetzbar :⇔

Es gibt Lösung $\widetilde{\mathbf{y}} \in C^1([t_0, t_+[, \Omega)$ des AWP (5.1.1) eq:awp mit $t_+ \ge t_1$, $\mathbf{y}(t) = \widetilde{\mathbf{y}}(t)\ \forall t_0 \le t < t_1$, wobei

$$t_+ = \infty \quad \wedge \quad \begin{matrix} t_+ < \infty\ , \\ \lim\limits_{t \to t_+} \|\widetilde{\mathbf{y}}(t)\| = \infty \end{matrix} \quad \wedge \quad \begin{matrix} t_+ < \infty\ , \\ \lim\limits_{t \to t_+} \operatorname{dist}(\widetilde{\mathbf{y}}(t), \partial\hat{\Omega}) = 0\ . \end{matrix}$$

BLOWUP Blow-up

(Analog: Maximale Fortsetzbarkeit in die Vergangenheit auf $]t_-, t_0]$)

Notation: $J(t_0, \mathbf{y}_0) =]t_-, t_+[$ = maximales Existenzintervall für Lösung von AWP (5.1.1) eq:awp.

Theorem 5.1.2 (Satz von Peano & Picard-Lindelöf)**.** *Falls $f : \hat{\Omega} \mapsto \mathbb{R}^d$ lokale Lipschitz-stetig, so hat das AWP* (5.1.1) eq:awp *$\forall (t_0, \mathbf{y}_0) \in \hat{\Omega}$ eine eindeutige maximal fortsetzbare Lösung $\mathbf{y} : J(t_0, \mathbf{y}_0) \mapsto \Omega$.*

EVOLUTION Evolutionsoperator $\mathbf{\Phi}^{s,t} : \Omega \mapsto \Omega$, $\mathbf{\Phi}^{s,t}\mathbf{y}_0 := \mathbf{y}(s)$, $\mathbf{y}$ Lsg. von $\dot{\mathbf{y}} = f(t, \mathbf{y})$, $\mathbf{y}(t) = \mathbf{y}_0$, $s \in J(t, \mathbf{y}_0)$

► $\mathbf{\Phi}^{t,t} = \mathrm{Id} \quad , \quad \mathbf{\Phi}^{s,t} = \mathbf{\Phi}^{s,r} \circ \mathbf{\Phi}^{r,t}\ .$ (5.1.2) eq:evrule

Bemerkung 217. Falls AWP (5.1.1) eq:awp autonom:

$$\mathbf{\Phi}^{s,t} = \mathbf{\Phi}^{s-t,0} \quad \text{falls } s - t \in I\ .$$

$\mathbf{y}$ Lösung mit $\mathbf{y}(t_0) = \mathbf{y}_0$ ➢ $\widetilde{\mathbf{y}}(t) := \mathbf{y}(t + \tau)$ Lsg. mit $\widetilde{\mathbf{y}}(t_0 - \tau) = \mathbf{y}_0$

► Für autonome AWP: „Kanonischer" Anfangszeitpunkt $t_0 = 0$

Beispiel 218 (Skalare Differentialgleichungen). ▷ $d = 1$

- $f(t, y) = -\lambda y$, $\lambda \in \mathbb{R}$ ► Lösung des AWP $y(t) = y_0 e^{-\lambda t}$, $t \in \mathbb{R}$
  (existiert für alle Zeiten, d.h. $]t_-, t_+[= \mathbb{R}$ für jedes y_0)
- $f(t, y) = \lambda y^2$, $\lambda \in \mathbb{R}$: $\dot{y} = \lambda y^2$, $y(0) = y_0 \in \mathbb{R}$

  Lösung $y(t) = \begin{cases} -\frac{1}{y_0^{-1} - \lambda t} & \text{, falls } y_0 \neq 0\ , \\ 0 & \text{, falls } y_0 = 0\ . \end{cases}$ (Blow-up) ,

  $\lambda, y_0 > 0 \quad \Rightarrow \quad J(0, y_0) =]-\infty, 1/\lambda y_0[\ .$

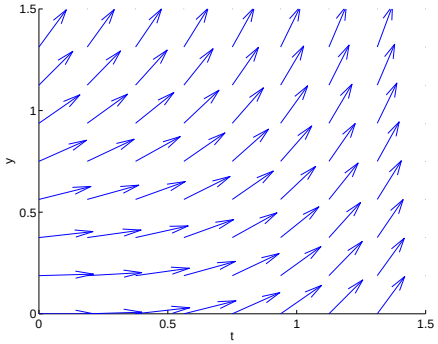
- $f(t,y) = -\frac{1}{\sqrt{y}}$, $\Omega = \mathbb{R}^+$, Anfangswert $y(0) = 1$

  ➢ $y(t) = (1 - 3t/2)^{2/3}$, $t_- = -\infty$, $t_+ = 2/3$

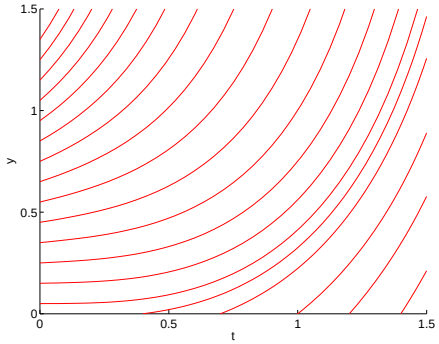
  (Lösung läuft zum Rand $y = 0$ des Phasenraums)

Beispiel 219 (Richtungsfeld und Orbits).

Ricatti Differentialgleichung $\dot{y} = y^2 + t^2$. (5.1.3) eq:ric



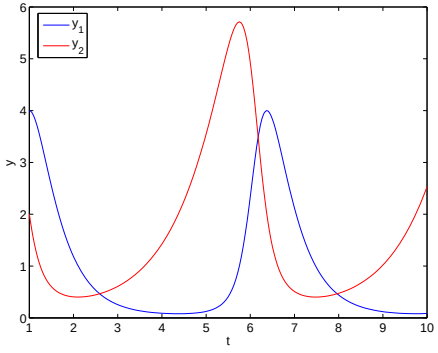
Richtungsfeld



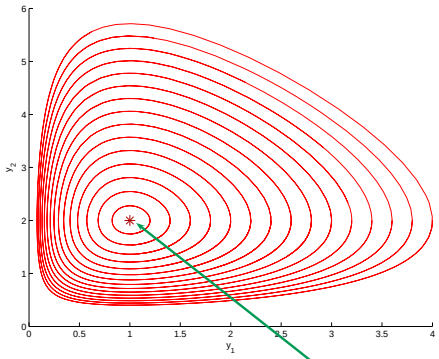
Lösungskurven

Lotka-Volterra Dgl. $\begin{pmatrix} \dot{y}_1 \\ \dot{y}_2 \end{pmatrix} = \begin{pmatrix} y_1(y_2 - 2) \\ y_2(1 - y_1) \end{pmatrix}$ (5.1.4) eq:lv

5.1 p. 569



Zeitabhängige Lösung, $\mathbf{y}(0) = \binom{4}{2}$



Orbits des von (5.1.4) erzeugten Flusses

Stationärer Punkt

eq:lv

Basiswechsel im Phasenraum (kovariante Transformation): $\widehat{\mathbf{y}} = \mathbf{S}^{-1}\mathbf{y}$, $\mathbf{S} \in \mathbb{R}^{d,d}$ regulär

$\mathbf{y}$ löst $\begin{cases} \dot{\mathbf{y}} = f(t,\mathbf{y}) , \\ \mathbf{y}(t_0) = \mathbf{y}_0 \end{cases}$ $\Leftrightarrow$ $\widehat{\mathbf{y}} := \mathbf{S}^{-1}\mathbf{y}$ löst $\begin{cases} \dot{\widehat{\mathbf{y}}} = \widehat{f}(t,\widehat{\mathbf{y}}) , \\ \widehat{\mathbf{y}}(t_0) = \mathbf{S}^{-1}\mathbf{y}_0 \end{cases}$ mit $\widehat{f}(t,\mathbf{y}) = \mathbf{S}^{-1} f(t,\mathbf{S}\mathbf{y})$.

(5.1.5) eq:kov

5.1 p. 570

Beispiel 220 (Lineare Differentialgleichungen mit konstanten Koeffizienten).

$\Omega = \mathbb{R}^d$, $\hat{\Omega} = \mathbb{R}^{d+1}$, $f(t,\mathbf{y}) = \mathbf{A}\mathbf{y}(t) + g(t)$ mit $\mathbf{A} \in \mathbb{R}^{d,d}$, $g : \mathbb{R} \mapsto \mathbb{R}^d$

Annahme: $\mathbf{A}$ diagonalisierbar, $\exists \mathbf{S} \in \mathbb{R}^{d,d}$ regulär: $\mathbf{S}^{-1}\mathbf{A}\mathbf{S} = \operatorname{diag}(\lambda_1, \ldots, \lambda_d)$, $\lambda_i \in \mathbb{C}$

- $g \equiv 0$: $\dot{\mathbf{y}} = \mathbf{A}\mathbf{y}$ (autonome homogene lineare Dgl.)

  $\widehat{\mathbf{y}} := \mathbf{S}^{-1}\mathbf{y}$ löst $\begin{aligned} \dot{\widehat{y}}_1 &= \lambda_1 \widehat{y}_1 , \\ &\vdots \\ \dot{\widehat{y}}_d &= \lambda_d \widehat{y}_1 \end{aligned}$ $\Rightarrow$ $\widehat{y}_i(t) = (\mathbf{S}^{-1}\mathbf{y}_0)_i \, e^{\lambda_i t}$, $t \in \mathbb{R}$.

  ◣ $\mathbf{y}(t) = \mathbf{S} \underbrace{\begin{pmatrix} e^{\lambda_1 t} & & \\ & \ddots & \\ & & e^{\lambda_d t} \end{pmatrix}}_{\text{Matrixexponentialfunktion } \exp(\mathbf{A}t)} \mathbf{S}^{-1}\mathbf{y}_0$.

5.1 p. 571

- Inhomogener Fall $\dot{\mathbf{y}}(t) = \mathbf{A}\mathbf{y}(t) + g(t)$ partikuläre Lösung durch „Variation der Konstanten”:

  Ansatz: $\mathbf{y}(t) = \exp(\mathbf{A}t)\mathbf{G}(t)\mathbf{y}_0$ mit $\mathbf{G} \in C^1(\mathbb{R}, \mathbb{R}^{d,d})$, $\mathbf{G}(t_0) = \mathbf{I}$

  $$\dot{\mathbf{y}}(t) = \Big(\mathbf{A}\exp(\mathbf{A}t)\mathbf{G}(t) + \exp(\mathbf{A}t)\dot{\mathbf{G}}(t)\Big)\mathbf{y}_0 = \mathbf{A}\mathbf{y}(t) = \mathbf{A}\exp(\mathbf{A}t)\mathbf{G}(t)\mathbf{y}_0 + g(t)$$

  ◣ $(\dot{\mathbf{G}}(t)\mathbf{y}_0) = g(t)$ $\Rightarrow$ $\mathbf{G}(t)\mathbf{y}_0 = \mathbf{y}_0 + \int_{t_0}^{t} \exp(-\mathbf{A}\tau) g(\tau)\, \mathrm{d}\tau$

  ◣ $\mathbf{y}(t) = \boxed{\exp(\mathbf{A}t)\mathbf{y}_0} + \boxed{\int_{t_0}^{t} \exp(\mathbf{A}(t-\tau)) g(\tau)\, \mathrm{d}\tau}$

  Lsg. des homogenen Problems — Faltung mit Inhomogenität

5.2 Kondition von Anfangswertproblemen

[File: section-kondition-von-anfangswertproblemen.tex, SVN: section-kondition-von-anfangswertproblemen.tex 1010 2006-09-11 15:44:43Z hiptmair]

Wie wirken sich *Störungen im Anfangswert* $\mathbf{y}_0$ in (5.1.1) eq:awp auf die Lösung $\mathbf{y}(t)$ aus **?**

Absolute Konditionszahl (→ Def. 1.6.1) def:wellconditioned durch differentielle Konditionsanalyse (→ Abschnitt 1.6) sec:kondition

5.2 p. 572

$$\frac{d}{ds}\mathbf{\Phi}^{s,t}\mathbf{y} = f(s,\mathbf{\Phi}^{s,t}\mathbf{y}) \quad \blacktriangleright \quad \frac{d}{ds}\left(\frac{d}{d\mathbf{y}}\mathbf{\Phi}^{s,t}\mathbf{y}\right) = \frac{d}{d\mathbf{y}}f(s,\mathbf{\Phi}^{s,t}\mathbf{y}) = \frac{\partial f}{\partial \mathbf{y}}(s,\mathbf{\Phi}^{s,t}\mathbf{y})\frac{d}{d\mathbf{y}}\mathbf{\Phi}^{s,t}\mathbf{y}$$

Propagationsmatrix (Wronski-Matrix) $\mathbf{W}(s,t_0) := \frac{d}{d\mathbf{y}}\mathbf{\Phi}^{s,t_0}\mathbf{y} \in \mathbb{R}^{d,d}$ zum AWP (5.1.1) erfüllt

VARG
Variationsgleichung

$$\frac{d}{ds}\mathbf{W}(s,t_0) = \frac{\partial f}{\partial \mathbf{y}}(s,\mathbf{\Phi}^{s,t_0}\mathbf{y}_0)\mathbf{W}(s,t_0) \quad , \quad \mathbf{W}(t_0,t_0) = \mathbf{I} \, . \qquad (5.2.1)$$

Beachte: Variationsgleichung = lineare Differentialgleichung mit $\Omega = \mathbb{R}^{d,d}$

$$\mathbf{y}_0 \leftarrow \mathbf{y}_0 + \delta\mathbf{y} \quad \succ \quad \delta\mathbf{y}(t) \approx W(t,t_0)\delta\mathbf{y} \quad \text{für „kleine } \delta\mathbf{y}\text{"}$$

Definition 5.2.1 (Kondition des AWP). *Intervallweise Kondition des AWP (5.1.1) bzgl. Norm $\|\cdot\|$ auf $\mathbb{R}^d$:*

$$\kappa(t_0,t) := \max\{\|\mathbf{W}(s,t_0)\| : t_0 \le s \le t\} \, .$$

Lemma 5.2.2 (Konditionsabschätzung für AWP).

$$\left\|\frac{\partial f}{\partial \mathbf{y}}(t,\mathbf{\Phi}^{t,t_0}\mathbf{y}_0)\right\| \le \chi(t) \quad \Rightarrow \quad \kappa(t_0,t) \le \exp\left(\int_{t_0}^{t}\chi(\tau)\,\mathrm{d}\tau\right) \, .$$

Beweis mit *Lemma von Gronwall* (→ Analysis)

Beispiel 221 (Kondition linearer AWPs mit konstanten Koeffizienten).

$$\dot{y} = \lambda y \quad \succ \quad y(t) = e^{\lambda t}y_0 \quad \Rightarrow \quad \begin{cases} \kappa(0,\infty) = \infty \, , \quad \kappa(-\infty,0) = 1 & \text{, falls } \operatorname{Re}\lambda > 0 \, , \\ \kappa(0,\infty) = \kappa(-\infty,0) = 1 & \text{, falls } \operatorname{Re}\lambda = 0 \, , \\ \kappa(0,\infty) = 1 \, , \quad \kappa(-\infty,0) = \infty & \text{, falls } \operatorname{Re}\lambda < 0 \, . \end{cases}$$

Beachte: Hier Variationsgleichung: $\dot{y} = \lambda y \quad \triangleright$ Propagation von Störungen durch Dgl. selbst

$\blacktriangleright$ Lemma 5.2.2 liefert zu pessimistische Aussagen.

Für $\dot{\mathbf{y}} = \mathbf{A}\mathbf{y}$, $\mathbf{A} \in \mathbb{R}^{d,d}$ diagonalisierbar → Bsp. 220: (Spektrum $\sigma(\mathbf{A})$ → Def. 3.3.1)

$$\begin{aligned} \operatorname{Re}\{\sigma(\mathbf{A})\} \subset]-\infty,0[\quad &\Rightarrow \quad \kappa(0,\infty) \sim 1 \, , \quad \kappa(-\infty,0) = \infty \, , \\ \operatorname{Re}\{\sigma(\mathbf{A})\} = \{0\} \quad &\Rightarrow \quad \kappa(0,\infty) \sim 1 \, , \quad \kappa(-\infty,0) \sim 1 \, , \\ \operatorname{Re}\{\sigma(\mathbf{A})\} \subset]0,\infty[\quad &\Rightarrow \quad \kappa(0,\infty) = \infty \, , \quad \kappa(-\infty,0) \sim 1 \, . \end{aligned}$$

Allgemeiner: Kondition bzgl. Störungen in f ▷ [11, Kap. 3]

5.3 Einschrittverfahren

[File: section-einschrittverfahren.tex, SVN: section-einschrittverfahren.tex 1010 2006-09-11 15:44:43Z hiptmair]

Gegeben: $f : \hat{\Omega} \mapsto \mathbb{R}^d$ lokale Lipschitz-stetig auf erweitertem Phasenraum $\hat{\Omega} \subset \mathbb{R} \times \Omega$

$\succ$ Definiert Familie von AWP (5.1.1)

$\blacktriangleright$ Zugehörige Evolution: $\mathbf{\Phi}^{s,t} : \Omega \mapsto \Omega$

Gegeben: Anfangsdaten $(t_0,\mathbf{y}_0) \in \hat{\Omega}$ $\succ$ Konkretes AWP

Ziel: ☞ Approximation von $\mathbf{y}(T)$ für ein $T \in J(t_0,y_0)$.
☞ Approximation der Funktion $t \mapsto \mathbf{y}(t)$, $t \in [t_0,T]$, $T \in J(t_0,y_0)$ $\triangleright$ $\mathbf{y}_h(t)$.

5.3.1 Kollokation

Idee: ❶ Approximiere $\mathbf{y}(t)$, $t \in [t_0,T]$, in $s+1$-dimensionalem Ansatzraum V von Funktionen $[t_0,T] \mapsto \mathbb{R}^d$ $\triangleright$ $\mathbf{y}_h$. .
❷ Festlegung von $\mathbf{y}_h \in V$ durch Kollokationsbedingungen

$$\mathbf{y}_h(t_0) = \mathbf{y}_0 \quad , \quad \dot{\mathbf{y}}_h(\tau_j) = f(\tau_j,\mathbf{y}_h(\tau_j)) \, , \quad j = 1,\ldots,s \, , \qquad (5.3.1)$$

für Kollokationspunkte $t_0 \le \tau_1 < \ldots < \tau_s \le T$.

„Standardoption": Polynomialer Ansatzraum $V = \mathcal{P}_s$

Herleitung: Formel für $\mathbf{y}_h(T)$ $(h := T - t_0, \tau_j := t_0 + c_j h, 0 \le c_1 < c_2 < \ldots < c_s \le 1)$

$\{L_j\}_{j=1}^s \subset \mathcal{P}_{s-1} \hat{=}$ Lagrange-Polynome (4.1.4): $L_j(c_i) = \delta_{ij}$, $i,j = 1,\ldots,s$.

(5.3.1)
$\blacktriangleright$ $$\dot{\mathbf{y}}_h(t_0+\tau h) = \sum_{j=1}^{s}\mathbf{k}_j L_j(\tau) \quad , \quad \mathbf{k}_j := f(t_0+c_j h,\mathbf{y}_h(t_0+c_j h)) \, .$$

$\blacktriangleright$ $$\mathbf{y}_h(t_0+\tau h) = \mathbf{y}_0 + h\sum_{j=1}^{s}\mathbf{k}_j\int_0^\tau L_j(\zeta)\,\mathrm{d}\zeta \, .$$

$$\mathbf{k}_i = f(t_0 + c_i h, \mathbf{y}_0 + h\sum_{j=1}^{s} a_{ij}\mathbf{k}_j)\,, \quad a_{ij} = \int_0^{c_i} L_j(\tau)\,\mathrm{d}\tau\,,$$
$$\text{mit} \qquad \mathbf{y}_h(T) = \mathbf{y}_0 + h\sum_{i=1}^{s} b_i\mathbf{k}_i\,. \quad b_i = \int_0^1 L_i(\tau)\,\mathrm{d}\tau\,. \qquad (5.3.2)$$

eq:rkimpl

DEVOLUTION

Diskrete Evolution $\Psi_h^{T,t_0} : \Omega \mapsto \Omega$, $\Psi_h^{T,t_0}\mathbf{y}_0 := \mathbf{y}_h(T)$

➤ (5.3.2) = (Nichtlineares) Gleichungssystem für Inkremente $\mathbf{k}_i$

Bemerkung 222. $f(t,\mathbf{y}) = f(t)$ & $\mathbf{y}_0 = 0$ ➤ Numerische Quadratur

$$\mathbf{y}(T) = \int_{t_0}^{T} f(t)\,\mathrm{d}t \approx h\sum_{i=1}^{s} b_j f(t_0 + c_j h)$$ = Quadraturformel, vgl. (4.4.2)

eq:guadweights

➞ $c_1, \ldots, c_s \leftrightarrow$ Knoten einer Quadraturformel (z.B. Gauss-Punkte auf $[0,1]$ → Abschnitt 4.4.2)

sec:gauss-quad

5.3 p. 577

• Fall $s = 1$ ➤ $c_1 = 1/2$ (↔ einfachste Gauss-Legendre-Quadraturformel)

$$L_1 \equiv 1 \;\Rightarrow\; a_{11} = 1/2\,,\quad b_1 = 1\,.$$
$$\mathbf{k}_1 = f(t_0 + 1/2h, \mathbf{y}_0 + 1/2h\mathbf{k}_1)\quad,\quad \mathbf{y}_h(T) = \mathbf{y}_0 + h\mathbf{k}_1\,. \qquad (5.3.3)$$

eq:implm

eq:implmidprule

(5.3.3) = Implizite Mittelpunktsregel

• Fall $s = 1$ & $c_1 = 0$ (↔ linksseitige Ein-Punkt-Quadraturformel)

$$L_1 \equiv 1 \;\Rightarrow\; a_{11} = 0\,,\quad b_1 = 1\,.$$
$$\mathbf{k}_1 = f(t_0, \mathbf{y}_0)\quad,\quad \mathbf{y}_h(T) = \mathbf{y}_0 + h\mathbf{k}_1 = \mathbf{y}_0 + hf(t_0,\mathbf{y}_0)\,.$$

eq:exEul

eq:exEul

(5.3.1) = Explizites Eulerverfahren (kein Lösen einer Gleichung erforderlich !)

• Fall $s = 1$ & $c_1 = 1$ (↔ rechtsseitige Ein-Punkt-Quadraturformel)

$$L_1 \equiv 1 \;\Rightarrow\; a_{11} = 1\,,\quad b_1 = 1\,.$$
$$\mathbf{k}_1 = f(t_0, \mathbf{y}_0 + h\mathbf{k}_1)\quad,\quad \mathbf{y}_h(T) = \mathbf{y}_0 + h\mathbf{k}_1 = \mathbf{y}_0 + hf(t_0,\mathbf{y}_h(T))\,.$$

eq:implEul

eq:implEul

(5.3.1) = Implizites Eulerverfahren

Beispiel 223 (Konvergenz von Kollokationsverfahren).

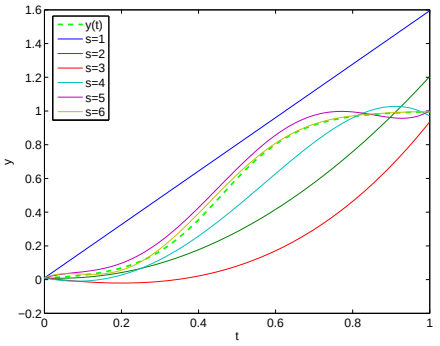
5.3 p. 578

Logistische Differentialgleichung (→ durch Resourcenknappheit beschränktes Wachstum)

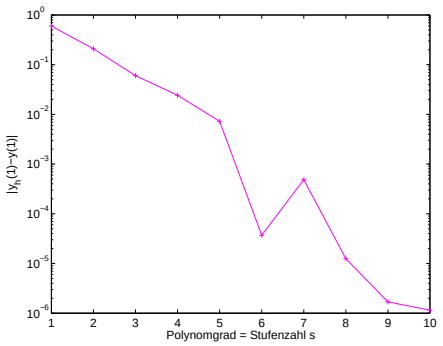
$$\dot{y} = \lambda y(1-y)\,,\quad y_0 \in]0,1[\;\Rightarrow\; y(t) = \frac{1}{1 + (y_0^{-1} - 1)e^{-\lambda t}}\,,\quad t \in \mathbb{R}\,. \qquad (5.3.4)$$

eq:logdgl

Numerische Experimente mit Gauss-Kollokationsverfahren auf $[0,1]$, $y_0 = 0.01$, $\lambda = 10$:
(Lösung der Gleichungen für Inkremente $\mathbf{k}_i$: MATLAB `fsolve`, Toleranz 10^{-9})



Näherungslösungen $y_h(t)$



▷ Exponentielle Konvergenz in s

! Lösbarkeit von (5.3.2) nur gesichert für „kleines h"

eq:rkimpl

Abschnitt 4.3 → Problematik globaler Polynominterpolation

sec:stuckweise-polynome

➤ Stückweise Polynominterpolation

Idee: ① Wähle Gitter $\mathcal{M} := \{t_0 < t_1 < t_2 < \cdots < t_n = T\}$

② $$\Psi_{\mathcal{M}}^{T,t_0} := \Psi_h^{t_n,t_{n-1}} \circ \cdots \circ \Psi_h^{t_2,t_1} \circ \Psi_h^{t_1,t_0}$$

$\Psi_h^{t_k,t_{k-1}}$:= diskrete Evolution durch Kollokation des AWP auf $[t_{k-1},t_k]$, → Formel (5.3.2).

eq:rkimpl

▷ Stückweise polynomiale Lösung $\mathbf{y}_h \in C^0([t_0,T])$

5.3 p. 579

Definition 5.3.1 (Einschrittverfahren). *Ein Einschrittverfahren (ESV, engl. single step method) für ein AWP (5.1.1) auf einem Gitter $\mathcal{M} := \{t_0 < t_1 < t_2 < \cdots < t_n = T\}$ und auf der Grundlage einer diskreten Evolution $\Psi_h^{s,t}$ erzeugt Folge von Näherungswerten*

eq:awp

$$\mathbf{y}_k \approx \mathbf{y}(t_k) \quad \textit{gemäss} \quad \mathbf{y}_k := \Psi_{h_k}^{t_k,t_{k-1}}\mathbf{y}_{k-1}\,,\quad k = 1,\ldots,n\,.$$

Definition 5.3.2 (Explizite und implizite Verfahren). *Ein Einschrittverfahren zur approximativen Lösung eines AWP heisst explizit, falls die zugrundeliegende diskrete Evolution durch endlich viele f-Auswertungen zu realisieren ist.*

Die diskrete Evolution eines impliziten Einschrittverfahrens erfodert die Lösung eines Gleichungssystems.

5.3 p. 580

Alle Kollokationsverfahren$\setminus\{$Explizites Eulerverfahren$\}$ sind implizit

5.3.2 Runge-Kutta-Verfahren

AWP: $\dot{\mathbf{y}}(t) = f(t,\mathbf{y}(t))\,,\ \mathbf{y}(t_0)=\mathbf{y}_0 \quad\Rightarrow\quad \mathbf{y}(t) = \mathbf{y}_0 + \int_{t_0}^{T} f(\tau,\mathbf{y}(t_0+\tau))\,\mathrm{d}\tau$

Approximation durch Quadraturformel (auf $[0,1]$) mit s Knoten $c_1,\ldots,c_s$:

$$\mathbf{y}(T) \approx \mathbf{y}_h(T) = \mathbf{y}_0 + h\sum_{i=1}^{s} b_i f(t_0+c_i h, \boxed{\mathbf{y}(t_0+c_i h)})\,.$$

Wie bekommt man diese Werte **?** ▷ Bootstrapping

Beispiel 224 (Konstruktion einfacher Runge-Kutta-Verfahren).

Quadraturformel → Trapezregel (4.4.3) **&** $\mathbf{y}_h(T)$ aus explizitem Eulerschritt (5.3.1)

$$\mathbf{k}_1 = f(t_0,\mathbf{y}_0)\,,\quad \mathbf{k}_2 = f(t_0+h,\mathbf{y}_0+h\mathbf{k}_1)\,,\quad \mathbf{y}_h(T) = \mathbf{y}_0 + \tfrac{h}{2}(\mathbf{k}_1+\mathbf{k}_2)\,. \tag{5.3.5}$$

(5.3.5) = explizite Trapezregel Quadraturformel → Einfachste Gauss-Quadraturformel (Mittelpunktsregel) **&** $\mathbf{y}_h(\frac{1}{2}(T+t_0))$ aus explizitem Eulerschritt (5.3.1)

$$\mathbf{k}_1 = f(t_0,\mathbf{y}_0)\,,\quad \mathbf{k}_2 = f(t_0+\tfrac{h}{2},\mathbf{y}_0+\tfrac{h}{2}\mathbf{k}_1)\,,\quad \mathbf{y}_h(T) = \mathbf{y}_0 + h\mathbf{k}_2\,. \tag{5.3.6}$$

(5.3.6) = explizite Mittelpunktsregel

Diskrete Evolutionen der Form (5.3.2)

Definition 5.3.3 (Runge-Kutta-Verfahren). *Für* $b_i, a_{ij} \in \mathbb{R}$, $i,j = 1,\ldots,s$, $s\in\mathbb{N}$, $c_i = \sum_{j=1}^{s} a_{ij}$ ➡ *s-stufiges Runge-Kutta-Einschrittverfahren (RK-ESV) für AWP* (5.1.1)

$$\mathbf{k}_i = f(t_0+c_i h, \mathbf{y}_0 + h\sum_{j=1}^{s} a_{ij}\mathbf{k}_j)\,,\quad i=1,\ldots,s \quad,\quad \mathbf{y}_h(t_0+h) = \mathbf{y}_0 + h\sum_{i=1}^{s} b_i\mathbf{k}_i\,.$$

$\mathbf{k}_i$ = *Inkremente/Stufen*

Falls $a_{ij}=0$ für $i\le j$ ◣ Explizites Runge-Kutta-Verfahren → Def. 5.3.2

Kurznotation für Runge-Kutta-Verfahren:

BS Butcher-Schema ▷

$$\begin{array}{c|c} \mathbf{c} & \mathfrak{A} \\ \hline & \mathbf{b}^T \end{array} := \begin{array}{c|ccc} c_1 & a_{11} & \cdots & a_{1s} \\ \vdots & \vdots & & \vdots \\ c_s & a_{s1} & \cdots & a_{ss} \\ \hline & b_1 & \cdots & b_s \end{array}\,. \tag{5.3.7}$$

Interpretation: Runge-Kutta-Verfahren ↔ Polygonzugapproximation der Lösungskurve

Beispiel 225 (Explizite Runge-Kutta-Schritte für Ricatti-Differentialgleichung). → Bsp 219

Anfangswertproblem: $\dot{y} = t^2 + y^2$, $y(0) = 0.2$.

Explizite Mittelpunktsregel:

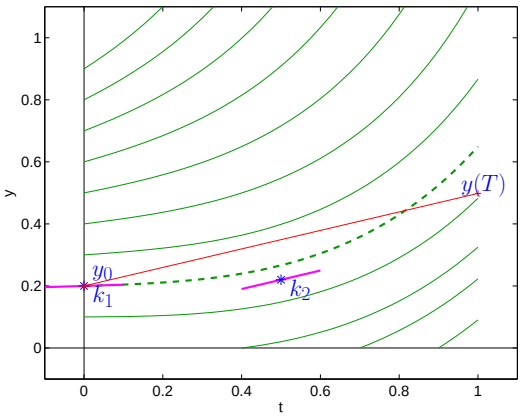
$$\begin{array}{c|cc} 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \\ \hline & 0 & 1 \end{array}$$

grün Lösungskurven

magenta Abschnittsteigungen $\mathbf{k}_i$

$*$ Punkte f-Auswertung

rot: Polygonzug



Explizite Trapezregel

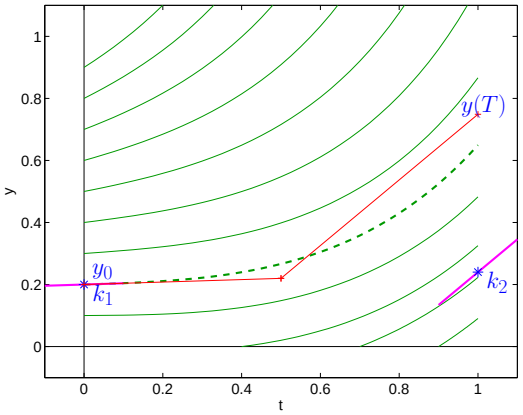
$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1 & 0 \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

grün: Lösungskurven

magenta: Abschnittsteigungen $\mathbf{k}_i$

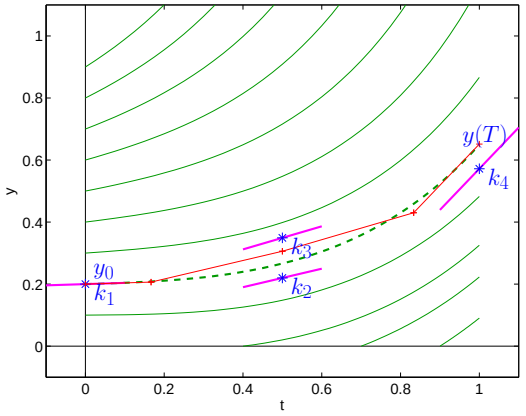
$*$: Punkte f-Auswertung

rot: Polygonzug



Klassisches Runge-Kutta-Verfahren (RK4)

$$\begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ \hline & \frac{1}{6} & \frac{2}{6} & \frac{2}{6} & \frac{1}{6} \end{array} \quad (5.3.8)$$



grün Lösungskurven

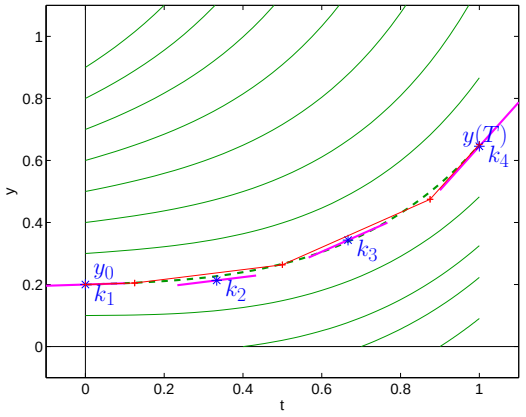
magenta Abschnittsteigungen $\mathbf{k}_i$

$*$ Punkte f-Auswertung

rot: Polygonzug

Kuttas 3/8-Regel

$$\begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & 0 & 0 & 0 \\ \frac{2}{3} & -\frac{1}{3} & 1 & 0 & 0 \\ 1 & 1 & -1 & 1 & 0 \\ \hline & \frac{1}{8} & \frac{3}{8} & \frac{3}{8} & \frac{1}{8} \end{array} \quad (5.3.9)$$



grün Lösungskurven

magenta Abschnittsteigungen $\mathbf{k}_i$

$*$ Punkte f-Auswertung

rot: Polygonzug

RKKOV

Affin-Kovarianz der Runge-Kutta-Verfahren:

Für $\mathbf{S} \in \mathbb{R}^{d,d}$ regulär, $\widehat{\mathbf{y}} := \mathbf{S}^{-1}\mathbf{y}$:

$\Psi_h^{s,t}$ = Diskrete Evolution zu $\dot{\mathbf{y}} = f(t,\mathbf{y})$,

$\widehat{\Psi}_h^{s,t}$ = Diskrete Evolution zu $\dot{\hat{\mathbf{y}}} = \widehat{f}(t,\widehat{\mathbf{y}})$ $\rightarrow$ eq:kov

$$\mathbf{S}\widehat{\Psi}_h^{s,t}\mathbf{S}^{-1}\mathbf{y} = \Psi_h^{s,t}\mathbf{y} \quad . \quad (5.3.10)$$

eq:rkkov

Warum $c_i = \sum_{j=1}^s a_{ij}$ in Def. def:rk 5.3.3 ?

Autonomisierung:

$$\begin{array}{l} \dot{\mathbf{y}} = f(t,\mathbf{y}) \ , \\ \mathbf{y}(t_0) = \mathbf{y}_0 \end{array} \quad \Rightarrow \quad \begin{pmatrix} \dot{\mathbf{y}} \\ \dot{s} \end{pmatrix} = \begin{pmatrix} f(s,\mathbf{y}) \\ 1 \end{pmatrix} =: \widehat{f}\left(\begin{pmatrix} \mathbf{y} \\ s \end{pmatrix}\right) \ , \quad \begin{pmatrix} \mathbf{y}(0) \\ s(0) \end{pmatrix} = \begin{pmatrix} \mathbf{y}_0 \\ t_0 \end{pmatrix} .$$

Evolutionen: $\Phi^{t+h,t} \leftrightarrow \widehat{\Phi}^{t+h,t}$,

Diskrete Evl: $\Psi_h^{t+h,t} \leftrightarrow \widehat{\Psi}_h^{t+h,t}$.

Wunsch: $\begin{pmatrix} \Phi^{t+h,t}\mathbf{y} \\ t+h \end{pmatrix} = \widehat{\Phi}^{t+h,t}\begin{pmatrix} \mathbf{y} \\ t \end{pmatrix}$ $\blacktriangleright$ $\begin{pmatrix} \Psi_h^{t+h,t}\mathbf{y} \\ t+h \end{pmatrix} = \widehat{\Psi}_h^{t+h,t}\begin{pmatrix} \mathbf{y} \\ t \end{pmatrix}$. (5.3.11) eq:rkatinv

$$\widehat{\Psi}_h^{t+h,t}\begin{pmatrix} \mathbf{y} \\ t \end{pmatrix} = \begin{pmatrix} \mathbf{y} + h\sum_{i=1}^s b_i\widehat{\mathbf{k}}_i \\ t + h\sum_{i=1}^s b_i\widehat{\kappa}_i \end{pmatrix} \ , \quad \begin{pmatrix} \widehat{\mathbf{k}}_i \\ \widehat{\kappa}_i \end{pmatrix} = \begin{pmatrix} f(t + h\sum_{j=1}^s a_{ij}\widehat{\kappa}_j, \mathbf{y} + h\sum_{j=1}^s a_{ij}\widehat{\mathbf{k}}_j) \\ 1 \end{pmatrix} .$$

$\blacktriangleright$ $\boxed{c_i = \sum_{j=1}^s a_{ij}}$ $\succ$ $\widehat{\mathbf{k}}_i = \mathbf{k}_i$ & $\boxed{\sum_{i=1}^s b_i = 1}$. (5.3.12) ex:rkatinv

= Hinreichende + notwendige Bedingungen für Autonomisierungsinvarianz eines RK-Verfahrens

5.3 p. 585

▷ Analyse von autonomisierungsinvarianten RK-Verfahren kann sich auf autonome Probleme beschränken.

5.4 p. 587

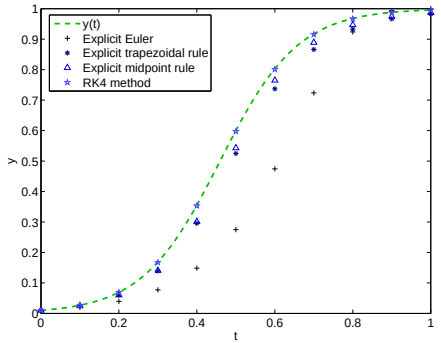
5.4 Konvergenz

[File: section-konvergenz.tex, SVN: section-konvergenz.tex 1010 2006-09-11 15:44:43Z hiptmair]

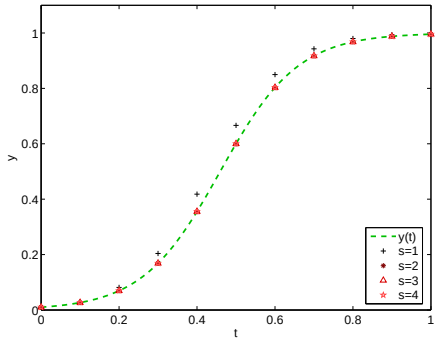
Beispiel 226 (Konvergenz von Einschrittverfahren).

Skalare logistische Differentialgleichung (eq:logdgl 5.3.4), $\lambda = 10$, $y(0) = 0.01$, $T = 1$:

Test: Explizite/implizite Runge-Kutta-Einschrittverfahren, verschiedene Schrittweiten h



$y_h(j/10)$, $j = 1,\ldots,10$: Explizite RK-Verfahren

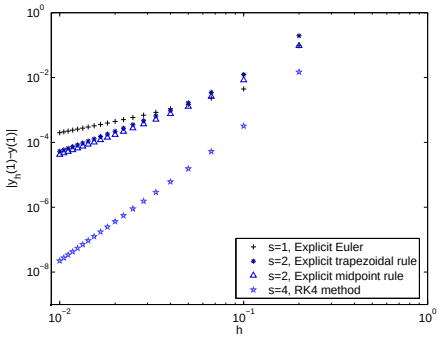


$y_h(j/10)$: Gauss-Kollokationsverfahren

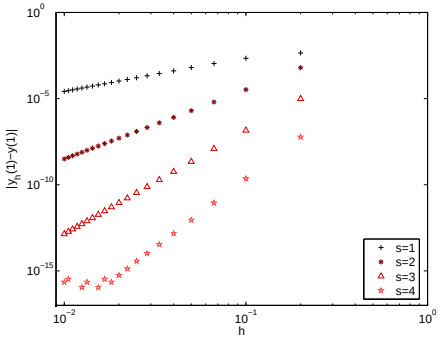
5.3 eq:rkkov p. 588

5.4 p. 588

Fehler $y_n - y(1)$ zum Endzeitpunkt, äquidistantes Gitter, `n=5:5:100`



Explizite RK-Verfahren



Gauss-Kollokationsverfahren

➢ Beobachtung: Algebraische Konvergenz $|y(1) - y_n| = O(h^\alpha)$

Definition 5.4.1 (Konvergenzordnung von Einschrittverfahren). *Ein ESV (→ Def. 5.3.1) für das AWP (5.1.1) auf $[t_0, T]$ hat Konvergenzordnung p (vgl. Thm. 4.4.4), wenn*

$$\exists C > 0: \quad \max_{j=0,\ldots,n} |\mathbf{y}_j - \mathbf{y}(t_j)| \leq C h_{\mathcal{M}}^p \quad \forall \text{Gitter } \mathcal{M} := \{t_0 < t_1 < \cdots < t_n \leq T\} \ , .$$

mit Schrittweite $h_{\mathcal{M}} := \max_{k=1,\ldots,n} |t_k - t_{k-1}|$.

Diskretisierungsfehler

Für ESV: Lokale Konsistenz(ordnung) ⇒ (globale) Konvergenz(ordnung)

Definition 5.4.2 (Konsistenzordnung einer diskreten Evolution). *Diskrete Evolution $\boldsymbol{\Psi}_h$ besitzt Konsistenzordnung p bzgl. einer Evolution $\boldsymbol{\Phi}$ (zu einer Dgl.), falls*

$$\forall (t, \mathbf{y} \in \widehat{\Omega}: \quad \exists C > 0: \quad \underbrace{|\boldsymbol{\Psi}_h^{t+h,t}\mathbf{y} - \boldsymbol{\Phi}^{t+h,t}\mathbf{y}|}_{\texttt{CONSERR}} \leq C h^{p+1} \quad \text{für kleine } h \ .$$

Konsistenzfehler

Theorem 5.4.3 (Konvergenz von Einschrittverfahren). *Runge-Kutta ESVs (→ Def. 5.3.1) auf der Grundlage der diskreten Evolution $\boldsymbol{\Psi}_h^{t+h,t}$ für AWP (5.1.1) erfüllen:*

$$\boldsymbol{\Psi}_h \text{ konsistent mit Ordnung } p \quad \Leftrightarrow \quad \text{ESV konvergent mit Ordnung } p$$

Merke: Konvergenzordnung erlaubt i.a. **keine** Aussage über Grösse des Diskretisierungsfehlers

Annahme: $f = f(\mathbf{y})$ (→ autonomes AWP) **&** f „hinreichend" glatt

➢ Konsistenzanalyse von Runge-Kutta-Verfahren durch *Taylorentwicklung* (→ MAPLE)

Beispiel 227 (Konsistenzanalyse durch Computeralgebra).

Explizite Trapezregel (5.3.5)


```
D(y) := x -> f(y(x)); y0 := y(0);
```

$$\mathsf{D}(y) := x \mapsto f(y(x)); \quad y0 := y(0)$$

```
k1 := f(y0); k2 := f(y0+h*k1);
```

$$k1 := f(y0); \quad k2 := f(y0 + hf(y0))$$

```
ym := y0+h/2*(k1+k2);
```

$$ym := y0 + 1/2\,h\,(f(y0) + f(y0 + hf(y0)))$$

```
taylor(ym-y(h),h=0,4);
```

$$series\left(\left(1/12\left(D^{(2)}\right)(f)(y(0))(f(y(0)))^2 - 1/6\,(\mathsf{D}(f)(y(0)))^2 f(y(0))\right)h^3 + O\left(h^4\right), h, 4\right)$$

Implizite Trapezregel: $\mathbf{y}_1 = \mathbf{y}_0 + \frac{h}{2}(f(\mathbf{y}_0) + f(\mathbf{y}_1))$

```
D(y) := x -> f(y(x));
```

$$\mathsf{D}(y) := x \mapsto f(y(x))$$

```
y0 := y(0);
```

$$y0 := y(0)$$

```
solve(y0+h/2*(f(y0) + f(yt))=yt,{yt});
```

$$\{yt = RootOf(-2\,y(0) - hf(y(0)) - hf(\_Z) + 2\,\_Z)\}$$

```
assign(%);
taylor(yt-y(h),h=0,4);
```

$$series\Big(\Big(-1/6\,(D^{(2)})(f)(y(0))(f(y(0)))^2 - 1/6\,(\mathsf{D}(f)(y(0)))^2 f(y(0)) + 1/4\,f(y(0))((\mathsf{D}(f)(y(0)))^2 + f(y(0))(D^{(2)})(f)(y(0)))\Big)h^3 + O(h^4), h, 4)$$

Für allgemeines Runge-Kutta-Verfahren → Def. 5.3.3, autonome Dgl. $\dot{\mathbf{y}} = f(\mathbf{y})$

$$\mathbf{\Psi}_h^{h,0}\mathbf{y} = \mathbf{y} + h\sum_{i=1}^{s} b_i\mathbf{k}_i \quad \blacktriangleright \quad \begin{aligned} &\mathbf{\Psi}_h^{h,0}\mathbf{y} - \mathbf{\Phi}^{h,0}\mathbf{y}_{|h=0} = 0\,, \\ &\tfrac{\mathrm{d}}{\mathrm{d}h}(\mathbf{\Psi}_h^{h,0}\mathbf{y} - \mathbf{\Phi}^{h,0}\mathbf{y})_{|h=0} = \sum_{i=1}^{s} b_i\mathbf{k}_{i|h=0} - f(\mathbf{y}) \end{aligned}$$

Da $\mathbf{k}_{i|h=0} = f(\mathbf{y})$ ▶ RK-ESV konsistent $\Leftrightarrow$ $\sum_{i=1}^{s} b_i = 1$.

Analoge Herleitung von Bedingungsgleichungen für RK-Koeffizienten b_i, a_{ij} (→ Def. 5.3.3) für Konsistenzordnungen $p > 1$:

$$p \geq 2 \;\Leftrightarrow\; \text{zusätzlich} \quad \sum_{i=1}^{s} b_i c_i = \tfrac{1}{2}\,, \quad \sum_{i=1}^{s} b_i c_i^2 = \tfrac{1}{3}\,, \qquad (5.4.1)$$

$$p \geq 3 \;\Leftrightarrow\; \text{zusätzlich} \quad \sum_{i,j=1}^{s} b_i a_{ij} c_j = \tfrac{1}{6}\,. \qquad (5.4.2)$$

(Algorithmus zur Herleitung der Bedingungsgleichungen ▷ [11, Abschn. 4.2.3], [23, Sect. III.1])

➞ Konstruktion von RK-Verfahren vorgegebener Konvergenzordnung durch Lösen der (nichtlinearen) Bedingungsgleichungen (B.G.) (vom Typ (5.4.1), (5.4.2)):

p	1	2	3	4	5	6	7	8	9	10	20
♯B.G.	1	2	4	8	17	37	85	200	486	1205	20247374

Einige Konvergenzordnungen → Bsp. 226:

Explizite Verfahren		Implizite Verfahren	
Expliztes Eulerverfahren	$p = 1$	Implizites Eulerverfahren	$p = 1$
Explizite Trapezregel	$p = 2$	Implizite Trapezregel	$p = 2$
Explizite Mittelpunktsregel	$p = 2$		
Klassisches Runge-Kutta-V.	$p = 4$		
Kuttas $3/8$-Regel	$p = 4$		

Viele weitere Verfahren ▷ [25, 26]

Theorem 5.4.4 (Konvergenzordnung von Kollokationsverfahren).
Für ein Kollokations-Einschrittverfahren (5.3.2) gilt

Konsistenzordnung p $\Leftrightarrow$ *Quadraturformel auf $[0,1]$ mit Knoten c_i und Gewichten b_i, $i = 1,\ldots,s$ hat Ordnung p (→ Abschnitt 4.4.2).*

Ordnungsschranken:

Für explizite Runge-Kutta-Verfahren $p \leq s$
Für allgemeine Runge-Kutta-Verfahren $p \leq 2s$ (vgl. Abschnitt 4.4.2))

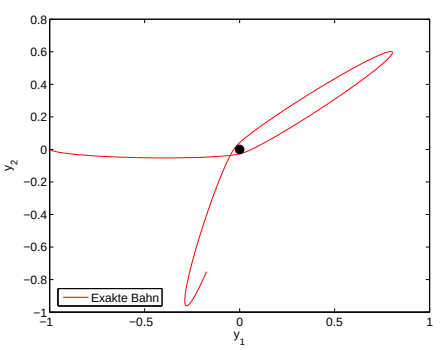
➢ Gauss-Kollokationsverfahren realisieren maximale Ordnung

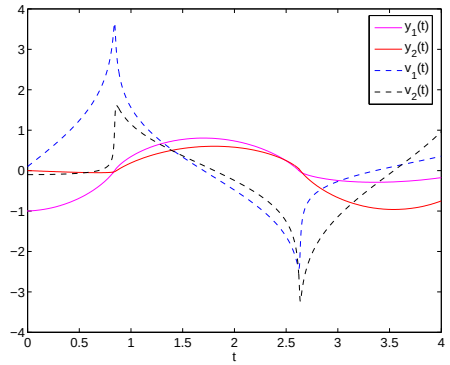
5.4.1 Schrittweitensteuerung für Einschrittverfahren

Beispiel 228 (Satellitenbahn).

Satellitenbahn im Gravitationsfeld eines Planenten (mit Koordinaten $\binom{0}{0}$): Keplerproblem

$$\ddot{\mathbf{y}} = -\frac{2\mathbf{y}}{\|\mathbf{y}\|^2} \;\Rightarrow\; \begin{pmatrix}\dot{\mathbf{y}}\\ \dot{\mathbf{v}}\end{pmatrix} = \begin{pmatrix}\mathbf{v}\\ -\frac{2\mathbf{y}}{\|\mathbf{y}\|^2}\end{pmatrix}\,, \quad \mathbf{y}(0) = \begin{pmatrix}-1\\0\end{pmatrix}, \mathbf{v}(0) = \begin{pmatrix}0.1\\-0.1\end{pmatrix}\,.$$





Beobachtung: „Peaks” in der Lösungskomponente $\mathbf{v}(t)$ (= zeitlokale Charakteristika) ◇

Adaptive Wahl des Rechengitters für ESV durch zeitlokale Fehlerschätzung

(→ Adaptive numerische Quadratur, Abschnitt 4.6.1) sec:adaptive-quadratur

Ziel: $\sharp\mathcal{M}$ möglichst klein & $\max_{t\in[t_0,T]} \|\mathbf{y}(t)-\mathbf{y}_h(t)\| < \mathtt{TOL}$ bzw. $\|\mathbf{y}(T)-\mathbf{y}_h(T)\| < \mathtt{TOL}$, $\mathtt{TOL}$ = Toleranz

Technik: Zeitlokaler Fehlerschätzer (basierend auf $\mathbf{y}(t_k)$) ➞ Wahl von t_{k+1}

Heuristik: Kontrolle des (lokalen !) Konsistenzfehlers → Def. 5.4.2 def:ssmloccons

Idee: *Schätzung des Konsistenzfehlers*

Vergleich zweier diskreter Evolutionen $\Psi_h^{t+h,t}$, $\widetilde{\Psi}_h^{t+h,t}$ (für eine *aktuelle Zeitschrittweite* h) verschiedener Ordnung (→ Abschnitt 4.6.1): sec:adaptive-quadratur

Falls Ordnung$(\widetilde{\Psi}_h) >$ Ordnung(Ψ_h)

$$\Rightarrow \underbrace{\Phi^{t+h,t}\mathbf{y}(t_k) - \Psi_h^{t+h,t}\mathbf{y}(t_k)}_{\text{Konsistenzfehler}} \approx \mathtt{EST}_k := \widetilde{\Psi}_h^{t+h,t}\mathbf{y}(t_k) - \Psi_h^{t+h,t}\mathbf{y}(t_k)\,. \tag{5.4.3}$$ eq:ssc1

Vergleich $\mathtt{EST}_k \leftrightarrow \mathtt{TOL}$ (Absolute Toleranz) bzw. $\mathtt{EST}_k \leftrightarrow \mathtt{TOL}\,\|\mathbf{y}(t_k)\|$ (Relative Toleranz) ➢ Verwerfen/Akzeptieren des aktuellen Schritts

Wir wollen mehr **!**

Wenn $\mathtt{EST}_k > \mathtt{TOL}$: Schrittweitenkorrektur $t_{k+1} = ?$

Wenn $\mathtt{EST}_k < \mathtt{TOL}$: Schrittweitenvorschlag $t_{k+2} = ?$

Falls Ordnung$(\Psi_h) = p$, Ordnung$(\widetilde{\Psi}_h) > p$, $p \in \mathbb{N}$, Ziel: Effizienz

$$\begin{aligned}\Psi_h^{t+h,t}\mathbf{y}(t_k) - \Phi^{t+h,t}\mathbf{y}(t_k) &= ch^{p+1} + O(h^{p+2})\,,\\ \widetilde{\Psi}_h^{t+h,t}\mathbf{y}(t_k) - \Phi^{t+h,t}\mathbf{y}(t_k) &= O(h^{p+2})\end{aligned} \quad \overset{h\ll 1}{\Rightarrow} \quad \mathtt{EST}_k \approx ch^{p+1} \overset{!}{=} \mathtt{TOL}\,.$$

„Optimale Schrittweite": $h^* = h\sqrt[p+1]{\frac{\mathtt{TOL}}{\mathtt{EST}_k}}$ (5.4.4) eq:ssc — Korrigierte Schrittweite / Schrittweitenvorschlag

Algorithmische Realisierung (ESV):

Eingebettete Runge-Kutta-Verfahren

Gleiche Inkremente $\mathbf{k}_i$, verschiedene Gewichte b_i (→ Def 5.3.3) def:rk realisieren RK-Evolutionen $\Phi_h, \widehat{\Phi}_h$ der Ordnungen p und $p+1$.

Gebräuchlich $p = 4$, $p = 7$

$$\begin{array}{c|c}\mathbf{c} & \mathfrak{A}\\ \hline & \mathbf{b}^T\\ \hline & \widehat{\mathbf{b}}^T\end{array} := \begin{array}{c|ccc} c_1 & a_{11} & \cdots & a_{1s}\\ \vdots & \vdots & & \vdots\\ c_s & a_{s1} & \cdots & a_{ss}\\ \hline & b_1 & \cdots & b_s\\ & \widehat{b}_1 & \cdots & \widehat{b}_s\end{array}$$

Eingebettetes RK-ESV: Butcher-Schema

Beispiel 229 (Eingebettete Runge-Kutta-Verfahren).

$$\begin{array}{c|ccccc} 0 & & & & & \\ \frac{1}{3} & \frac{1}{3} & & & & \\ \frac{1}{3} & \frac{1}{6} & \frac{1}{6} & & & \\ \frac{1}{2} & \frac{1}{8} & 0 & \frac{3}{8} & & \\ 1 & \frac{1}{2} & 0 & -\frac{3}{2} & 2 & \\ \hline y_1 & \frac{1}{6} & 0 & 0 & \frac{2}{3} & \frac{1}{6} \\ \hline \widehat{y}_1 & \frac{1}{10} & 0 & \frac{3}{10} & \frac{2}{5} & \frac{1}{5} \end{array}$$

Eingebettes RK-Verfahren der Ordnung 4(5) von Merson

$$\begin{array}{c|ccccc} 0 & & & & & \\ \frac{1}{2} & \frac{1}{2} & & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & & \\ 1 & 0 & 0 & 1 & & \\ \frac{3}{4} & \frac{5}{32} & \frac{7}{32} & \frac{13}{32} & -\frac{1}{32} & \\ \hline y_1 & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} & \\ \hline \widehat{y}_1 & -\frac{1}{2} & \frac{7}{3} & \frac{7}{3} & \frac{13}{6} & -\frac{16}{3} \end{array}$$

Eingebettes RK-Verfahren der Ordnung 4(5) von Fehlberg

◇

Adaptive Integratoren für Anfangswertprobleme in MATLAB:

```
options = odeset('abstol',atol,'reltol',rtol,'stats','on');
[t,y] = ode45/ode23(@(t,x) f(t,x),tspan,y0,options);
```

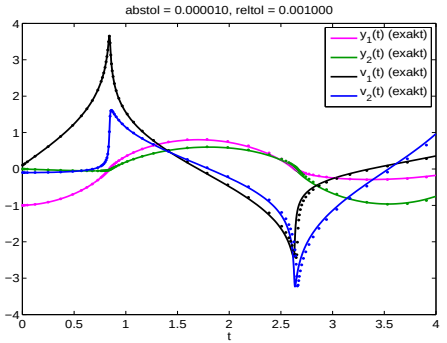
(`f` = function handle, `tspan` $\hat{=}\,[t_0,T]$, `y0` $\hat{=}\,\mathbf{y}_0$, `t` $\hat{=}\,t_k$, `y` $\hat{=}\,\mathbf{y}_k$)

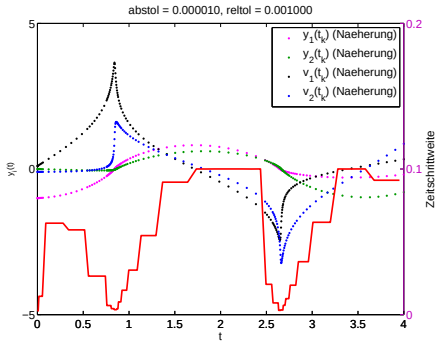
Beispiel 230 (Adaptive RK-ESV zur Satellitenbahnberechnung). → Bsp. 228 ex:satellite

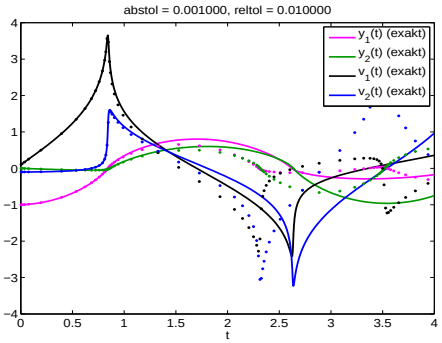
Adaptiver Integrator: `ode45(@(t,x) satf,[0 4],[-1;0;0.1;-0.1,],options])`:

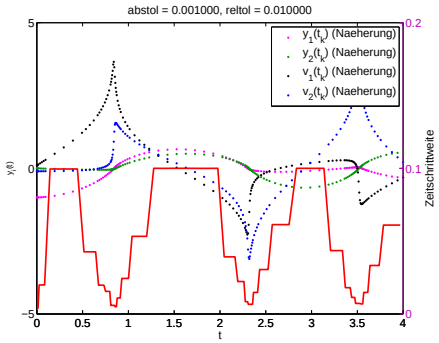
❶ `options = odeset('reltol',0.001,'abstol',1e-5);`

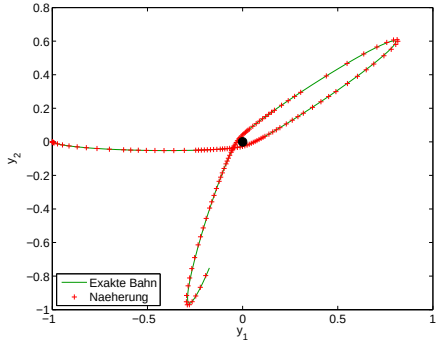
❷ `options = odeset('reltol',0.01,'abstol',1e-3);`



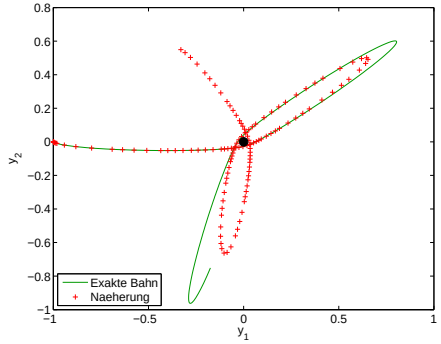








reltol=0.001, abstol=1e-5



reltol=0.01, abstol=1e-3

☞ Qualitativ falsche Lösung bei geringfügig erhöhter Toleranz **!**

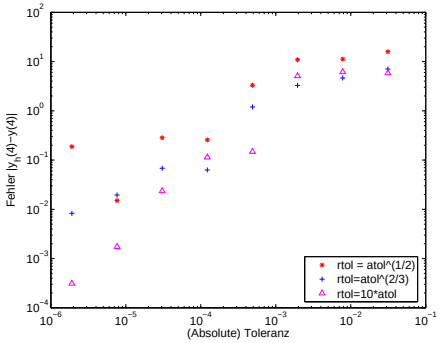
◇

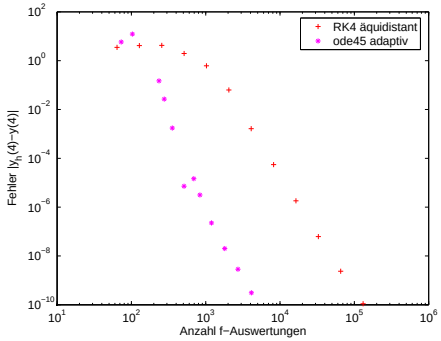
Beispiel 231 (Schrittweitensteuerung bei Satellitenbahnberechnung). → Bsp. 228 ex:satellite

AWP aus Bsp. 228 ex:satellite

`ode45` mit verschiedenen absoluten/relativen Toleranzen

! Toleranzen sagen nichts über tatsächliche Genauigkeit





Effizienz von Schrittweitensteuerung:

Vergleich:

- Klassisches Runge-Kutta-Verfahren (5.3.8) eq:rk4
- Eingebettetes Runge-Kutta-Verfahren mit Schrittweitensteuerung: `ode45`

Aufwandsmass: $\sharp f$-Auswertungen

Adaptivität zahlt sich aus !

◇

Übung 5.5. Kurve in der Ebene lässt sich als Niveaulinie einer stetigen Funktion $F : \mathbb{R}^2 \mapsto \mathbb{R}$ beschreiben:

$$\mathcal{C} := \{\boldsymbol{x} \in \mathbb{R}^2 \colon F(\boldsymbol{x}) = 0\} .$$

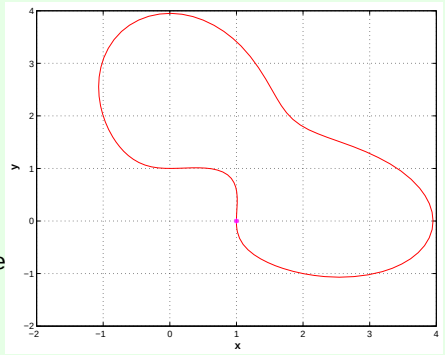
Falls F stetig differenzierbar ist, können zusammenhängende Teile der Kurve als Lösungskurve zu Anfangswertproblemen

$$\dot{\mathbf{y}}(t) = \frac{\mathbf{grad}\, F(\mathbf{y}(t))^{\perp}}{\|\mathbf{grad}\, F(\mathbf{y}(t))\|} , \quad F(\mathbf{y}(0)) = 0 ,$$

erhalten werden. Dabei bezeichnet $^\perp$ die Rotation eines Vektor $\in \mathbb{R}^2$ um $90°$ gegen den Uhrzeigersinn.

Entwerfe eine MATLAB-Funktion `draweggcurve()`, die ein solches Anfangswertproblem löst, um die Kurve

$$\{\boldsymbol{x} = (x,y) \in \mathbb{R}^2 \colon F(\boldsymbol{x}) := (x^2+y^2)^2 - x^3 - y^3 = 0\}$$



durch $(1,0)$ zu plotten. Löse dazu das Anfangswertproblem auf $[0, 15.8]$ und erwende dazu die MATLAB-Funktion `ode45` mit folgenden Parametern:

```
options = odeset( 'reltol',0.0001,
                  'abstol',1e-9);
http://en.wikipedia.org/wiki/Crooke
```

5.6
p. 605

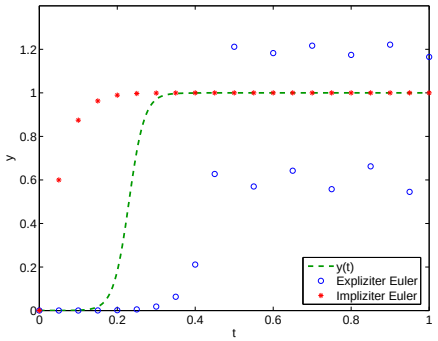
5.6 Stabilität

[File: section-ngdiffgleichungen-stabilitaet.tex, SVN: section-ngdiffgleichungen-stabilitaet.tex 1771 2007-08-26 02:14:34Z hiptmair]

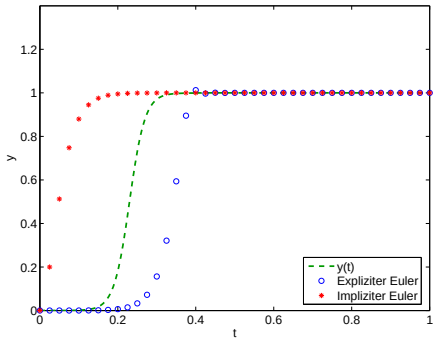
Beispiel 232 (Unterschiedliches Verhalten expliziter und impliziter Integratoren).

Logistische Differentialgleichung $\dot{y} = f(y)$, $f(y) = \lambda y(1-y)$ → (5.3.4), $\lambda = 50$, Anfangswert $y_0 = 10^{-4}$, Zeitintervall $[0,1]$:

eg:logdgl



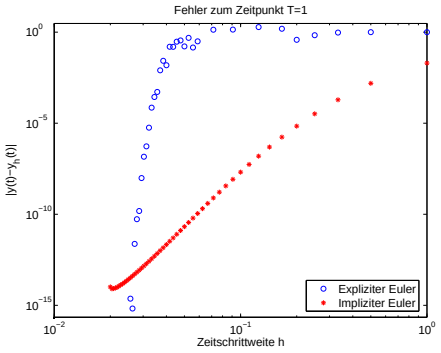
Äquidistantes Gitter, $h_{\mathcal{M}} = \frac{1}{20}$

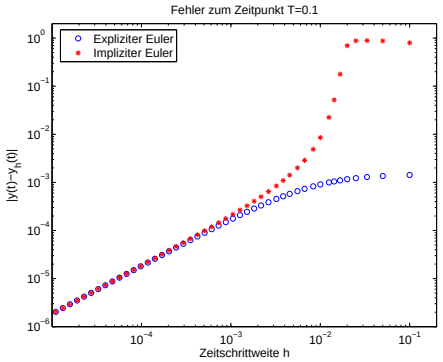


Äquidistantes Gitter, $h_{\mathcal{M}} = \frac{1}{40}$

5.6
p. 606

$h_{\mathcal{M}}$-Abhängigkeit der Fehler zum Endzeitpunkt für Integrationsintervalle $[0,T]$, $T = 0.1, 1$:





Beobachtung: $y(T) \approx 0$ ➢ Impliziter Euler *präasymptotisch* unbrauchbar
$y(T) \approx 1$ ➢ Expliziter Euler *präasymptotisch* unbrauchbar

$y \in \{0,1\}$ ⇒ $f(y) = 0$: $0, 1$ sind stationäre Punkte der Evolution zur Dgl.

$$\frac{df}{dy}(0) = \lambda \;\Rightarrow\; \text{für } y \approx 0: \text{ Dgl.} \approx \dot{y} = \lambda y\,,$$
$$\frac{df}{dy}(1) = -\lambda \;\Rightarrow\; \text{für } y \approx 1: \text{ Dgl.} \approx \dot{y} = -\lambda y\,.$$

5.6
p. 607

5.6.1 Modellproblemanalyse

Autonomes skalares lineares AWP: $\dot{y} = \lambda y\,,\ y(0) = 1\,,\quad \operatorname{Re}\lambda < 0$ auf $[0,\infty[$ (5.6.1) eq:stiffmj

$$y(t) = e^{\lambda t} \to 0 \quad \text{für } t \to \infty \quad \text{(Asymptotische Stabilität)}\,.$$

↔ Diskrete Evolution von s-stufigem Runge-Kutta-Verfahren (→ Def. 5.3.3) def:rk $\vdash \begin{array}{c|c} \mathbf{c} & \mathfrak{A} \\ \hline & \mathbf{b}^T \end{array}$?

$$k_i = \lambda(y + h\sum_{j=1}^{s} a_{ij}k_j)\,,$$
$$y_h(t+h) = y + h\sum_{i=1}^{s} b_i k_i$$
$$\Rightarrow \quad \boxed{\Psi_h^{h,0} y = (1 + \lambda h \mathbf{b}^T(\mathbf{I} - \lambda h\mathfrak{A})^{-1}\mathbf{1})y}\,.$$

(Kontinuierliche) Evolution: $\boxed{\Phi^{h,0} = e^{\lambda h}}$

Wann erbt Lösung $\{y_k\}_{k=0}^{\infty}$, $y_{k+1} = \Psi_h^{h,0} y_k$, eines RK-ESV auf (unendlichem) äquidistantem Gitter (Maschenweite h) asymptotische Stabilität ?

$$\lim_{k\to\infty} y_k = 0 \quad \Leftrightarrow \quad z := \lambda h \text{ so, dass } \ |\Psi_h^{h,0}(z)| < 1\,.$$

5.6
p. 608

Definition 5.6.1 (Stabilitätsgebiet eines Einschrittverfahrens). *Das Stabilitätsgebiet eines ESV für das AWP (eq:stiffmp) auf der Grundlage der diskreten Evolution $\Psi_h^{h,0} =: S(z)$, $z := \lambda h$, $S : D_S \subset \mathbb{C} \mapsto \mathbb{C}$, ist*

$$\mathcal{S}_\Psi := \{z \in \mathbb{C}: |S(z)| < 1\} \subset \mathbb{C} .$$

Theorem 5.6.2 (Stabilitätsfunktion von Runge-Kutta-Verfahren). *Ist $\Psi_h^{h,0}$ diskrete Evolution zu einem s-stufigen Runge-Kutta-Verfahren (→ Def. 5.3.3) mit Butcher-Schema $\begin{array}{c|c} \mathbf{c} & \mathfrak{A} \\ \hline & \mathbf{b}^T \end{array}$ für das AWP (eq:stiffmp), dann*

$$\Psi_h^{h,0} = \underbrace{1 + z\mathbf{b}^T(\mathbf{I} - z\mathfrak{A})^{-1}\mathbf{1}}_{\text{STABFN}} , \quad z := \lambda h , \quad \mathbf{1} = (1, \ldots, 1)^T \in \mathbb{R}^s .$$

Stabilitätsfunktion $S(z)$

Übung 5.7. Das allgemeine Runge-Kutta-Verfahren aus Definition 5.3.3 werde angewandt auf das skalare Modellproblem (5.6.1), Schrittweite $h > 0$.

(i) Schreibe $\boldsymbol{k} \in \mathbb{R}^s$ für den Vektor $(k_1, \ldots, k_s)^T$ der Inkremente und $y_1 := y_h(t_0 + h)$. Zeige, dass

$$\begin{pmatrix} \mathbf{I} - z\mathfrak{A} & 0 \\ -z\mathbf{b}^T & 1 \end{pmatrix} \begin{pmatrix} \boldsymbol{k} \\ y_1 \end{pmatrix} = y_0 \begin{pmatrix} \mathbf{1} \\ 1 \end{pmatrix} .$$

(ii) Benutze die Cramersche Regel, um die alternative Darstellung

$$S(z) = \frac{\det(\mathbf{I} - z\mathfrak{A} + z\mathbf{1}\mathbf{b}^T)}{\det(\mathbf{I} - z\mathfrak{A})}$$

herzuleiten.

- Explizites Eulerverfahren: $\begin{array}{c|c} 0 & 0 \\ \hline & 1 \end{array}$ ➢ $S(z) = 1 + z$.
- Implizites Eulerverfahren: $\begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array}$ ➢ $S(z) = \frac{1}{1-z}$.

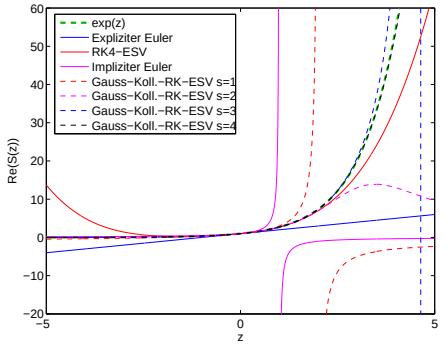
- Explizite Trapezregel: $\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1 & 0 \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$ ➢ $S(z) = 1 + z + \frac{1}{2}z^2$.
- Implizite Mittelpunktsregel: $\begin{array}{c|c} \frac{1}{2} & \frac{1}{2} \\ \hline & 1 \end{array}$ ➢ $S(z) = \frac{1 + \frac{1}{2}z}{1 - \frac{1}{2}z}$.
- RK4-Verfahren (eq:rk4): $\begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ \hline & \frac{1}{6} & \frac{2}{6} & \frac{2}{6} & \frac{1}{6} \end{array}$ ➢ $S(z) = 1 + z + \frac{1}{2}z^2 + \frac{1}{6}z^3 + \frac{1}{24}z^4$.

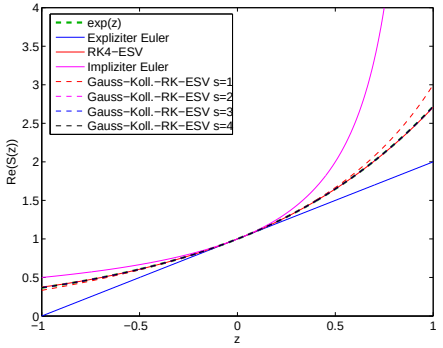
Korollar 5.7.1.

Explizite Runge-Kutta-Verfahren ➤ $S(z) \in \mathcal{P}_s$,

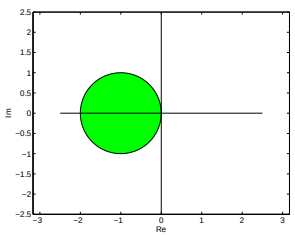
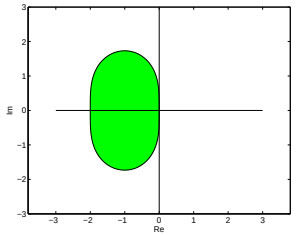
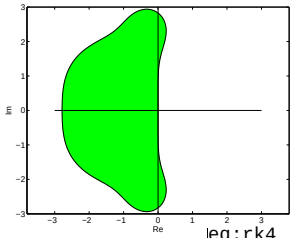
Allgemeine Runge-Kutta-Verfahren ➤ $S(z) = \frac{P(z)}{Q(z)}$, $P, Q \in \mathcal{P}_s$.

EXSTABFN
Beispiele: Verhalten von Stabilitätsfunktionen (für reelles Argument z):





EXSTABDOM
Beispiele: Stabilitätsgebiete $\mathcal{S}$ expliziter RK-ESV:

$\mathcal{S}_\Psi$: expliziter Euler (eq:exEul 5.3.1) $\mathcal{S}_\Psi$: explizite Trapezregel Mittelpunktsregel $\mathcal{S}_\Psi$: RK4-Verfahren (eq:rk4 5.3.8) Kuttas 3/8-Regel (eq:38 5.3.9)

Das Stabilitätsgebiet expliziter RK-Verfahren ist beschränkt !

$\lim_{k\to\infty} y_k = 0$ für von explizitem RK-Verfahren für AWP (eq:stiffmp 5.6.1) erzeugte Folge $\{y_k\}_{k=0}^{\infty}$ nur wenn $|\lambda h|$ hinreichend klein ($\lambda h \in \mathcal{S}_\Psi$) .

Nicht durch Genauigkeitsanforderungen bedingte Schrittweitenbeschränkung **!**
(für explizite (RK)-ESV)

5.7.1 Steifheit

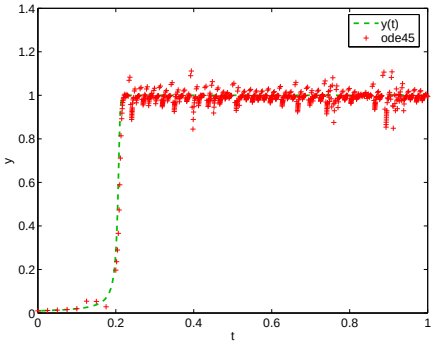
Konzept 5.7.2 (Steifes Anfangswertproblem)**.** *Ein AWP heisst steif (engl.* stiff*), falls für explizite ESV (→ Def. def:expl 5.3.2) Stabilität eine wesentlich kleinere Schrittweite verlangt als die Genauigkeitsanforderungen.*

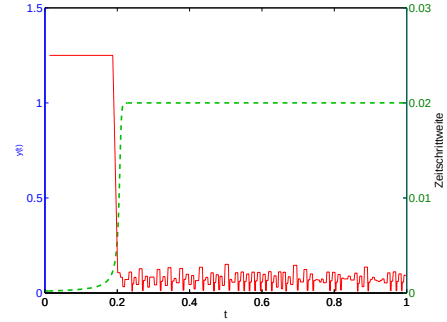
Beispiel 233 (Adaptive explizite Einschrittverfahren für steifes Problem).

$$\dot{y}(t) = \lambda y^2(1-y)\,, \quad \lambda = 500 \quad , \quad y(0) = \tfrac{1}{100}\,.$$

MATLAB-CODE : Adaptives ESV für steifes Problem

```
fun = @(t,x) 500*x^2*(1-x); tspan = [0 1]; y0 = 0.01;
options = odeset('reltol',0.1,'abstol',0.001,'stats','on');
[t,y] = ode45(fun,tspan,y0,options);
plot(t,y,'r+');
```



➢ Schrittweitensteuerung realisiert Schrittweitenbeschränkung !
(`186 successful steps, 55 failed attempts, 1447 function evaluations`)

$y = 1$ stark attraktiver Fixpunkt ➢ Extreme Schrittweitenbeschränkung für expliziten Integrator `ode45`

Welche Anfangswertprobleme sind steif **?**

☞ ODE-Modelle für Systeme mit schnell relaxierenden Komponenten
(mit stark unterschiedlichen Zeitkonstanten)

Beispiel 234 (Steife Schaltkreisgleichungen im Zeitbereich).

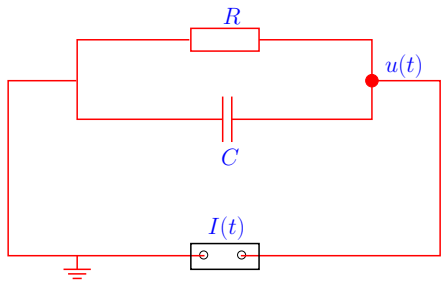
Schaltkreisanalyse im Zeitbereich:
(vgl. Frequenzbereichsanalyse Bsp. ex:elnet 66)

Bauelementgleichungen (Zeitbereich):

- Widerstand: $i(t) = R^{-1}u(t)$
- Kondensator: $i(t) = C\dot{u}(t)$

Dgl. aus Knotenanalyse:

$$C\dot{u}(t) = -R^{-1}u(t) + I(t)\,.$$



Konkret: $C = 1pF$, $R = 1k\Omega$, $I(t) = \sin(2\pi\,1\mathrm{Hz}\,t)\mathrm{mA}$, $u(0) = 0V$

Skalierte (dimensionslose) Dgl. : $\dot{u}(t) = -10^9\,u(t) + 10^9\sin(2\pi t) \;\Rightarrow\; u(t) \approx \sin(2\pi t)$.

Steife AWP ➢ (Geeignete) implizite RK-ESV (→ Def. def:expl 5.3.2) sind zu verwenden **!**

Übung 5.8. Das Anfangswerproblem $y(0) = 1$, $\dot{y}(0) = 0$ für die lineare Differentialgleichung 2. Ordnung

$$\ddot{y} + \dot{y} + y = 0 \tag{5.8.1}$$

soll mit einem 2-stufigen sogenannten SDIRK-Verfahren (singly diagonally implicit Runge-Kutta method) numerisch gelöst werden. Dies ist ein Runge-Kutta-Verfahren beschrieben durch das Butcher-Tableau

$$\begin{array}{c|cc} \gamma & \gamma & 0 \\ 1-\gamma & 1-2\gamma & \gamma \\ \hline & 1/2 & 1/2 \end{array} . \tag{5.8.2}$$

(i) Gib die Gleichungen für die Inkremente k_1 und k_2 des Runge-Kutta-Verfahrens (5.8.2) angewandte auf ein Anfangswertproblem zur Differentialgleichung $\dot{y} = f(t, y)$ an.

(ii) Bestimme die Stabilitätsfunktion $S(z)$ des SDIRK-Verfahrens (5.8.2) für $\gamma = 1$. Ist das Verfahren A/L-stabil?

(iii) Formuliere (5.8.1) als Anfangswertproblem für ein lineares System 1. Ordnung für die Funktion $\mathbf{z}(t) = (y(t), \dot{y}(t))^T$.

(iv) Implementiere eine MATLAB-Funktion

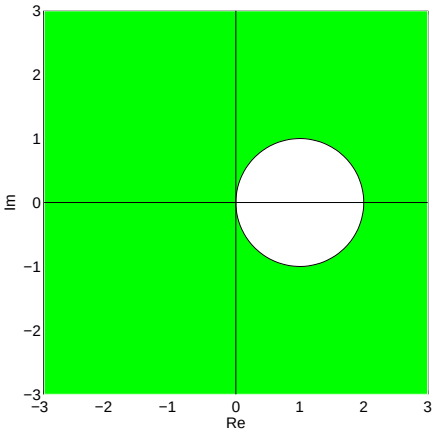
```
z = sdirkstep(z0,h),
```

die die numerische Evolution des Einschrittverfahrens (5.8.2) für die in (ii) erhaltene Differentialgleichung realisiert.

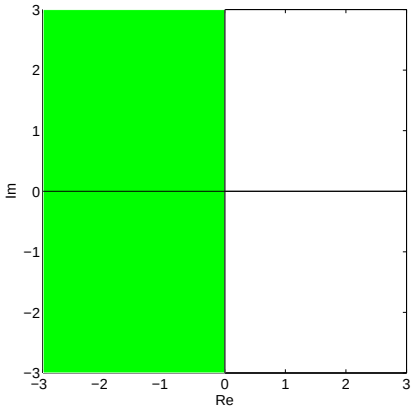
(v) Führe mit Hilfe von MATLAB ein numerisches Experiment aus, das einen Hinweis auf die Ordnung des Verfahrens (mit $\gamma = \frac{3+\sqrt{3}}{6}$) für das Anfangswertproblem aus Teilaufgabe (ii) liefert.

5.8.1 Einschrittverfahren für steife AWP

Beispiele: Stabilitätsgebiete (→ Def. 5.6.1) impliziter Einschrittverfahren (→ Def. 5.3.2)



$\mathcal{S}_\Psi$: implizites Eulerverfahren (5.3.1)



$\mathcal{S}_\Psi$: implizite Trapezregel

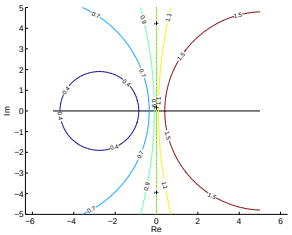
AWP (5.6.1) & $\lambda \gg 1$: Keine stabilitätsbedingte Schrittweitenbeschränkung für beide Verfahren (vgl. Verhalten von $S(z)$, $\operatorname{Re} z < 0$)

Definition 5.8.1 (A-stabiles Einschrittverfahren).

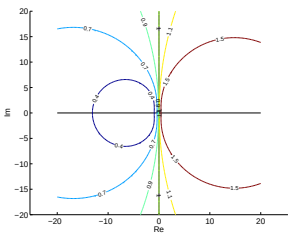
ESV A-stabil :⇔ $\{z \in \mathbb{C}: \operatorname{Re} z < 0\} \subset \mathcal{S}_\Psi$ ($\mathcal{S}_\Psi$ = *Stabilitätsgebiet, Def. 5.6.1)*

Autonome Dgl. $\dot{\mathbf{y}} = f(\mathbf{y})$, Stationärer Punkt $f(\mathbf{y}^*) = 0$ $\wedge$ $\operatorname{Re}\{\sigma(Df(\mathbf{y}^*))\} \subset \mathbb{R}^-$ ($\hat{=}$ Stabilität von $\mathbf{y}^*$) $\Rightarrow$ Falls $\|\mathbf{y}_0 - \mathbf{y}^*\| < \delta$, dann $\lim_{k\to\infty} \mathbf{y}_k = 0$.

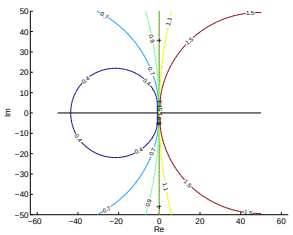
Stabilität von Gauss-Kollokations-Einschrittverfahren (→ Abschnitt 5.3.1):



Implizite Mittelpunktsregel



$s = 2$ (Ordnung 4)



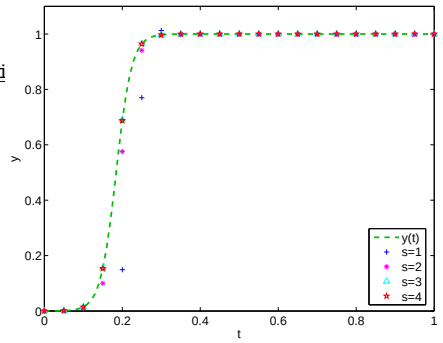
$s = 4$ (Ordnung 8)

Niveaulinien von $|S(z)|$ für Gauss-Kollokations-Einschrittverfahren

Beispiel 235 (Gauss-Kollokationsverfahren für logistische Differentialgleichung). → Bsp. 232, ex:logexi

Logistische Differentialgleichung $\dot{y} = f(y)$, $f(y) = \lambda y(1-y)$ → (5.3.4), eg:logdgl $\lambda = 50$, Anfangswert $y_0 = 10^{-4}$, Zeitintervall $[0,1]$.

Kollokations-Einschrittverfahren (→ Abschnitt 5.3.1) sec:kollokation auf äquidistantem Gitter, $h = \frac{1}{20}$.



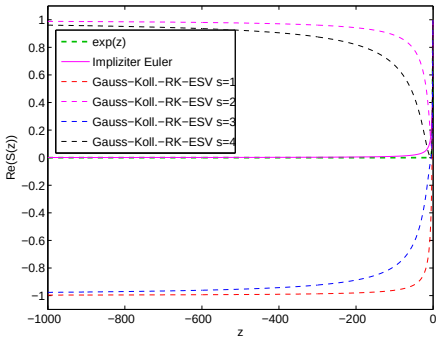
Theorem 5.8.2 (Stabilität von Gauss-Kollokations-Einschrittverfahren).

Alle Gauss-Kollokations-RK-ESV (eg:rkimpl 5.3.2) *sind A-stabil.*

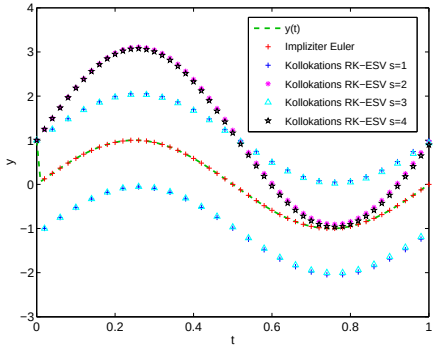
Beispiel 236 (Implizite RK-ESV bei schnellen Transienten). → Bsp. ex:odecircuit 234

AWP: $\dot{y} = -\lambda y + \beta \sin(2\pi t)$, $\lambda = 10^6$, $\beta = 10^6$, $y(0) = 1$.

RK-ESV, äquidistantes Gitter auf $[0,1]$, $h = \frac{1}{40}$:



Stabilitätsfunktionen für $\operatorname{Re} z \ll 1$



Diskrete Evolutionen (Zeitverlauf)

➢ Ungenügende Dämpfung der Anfangsstörung bei Kollokations-RK-ESV **!**
(Oszillationen für ungerades s → vgl. Stabilitätsfunktionen)

Definition 5.8.3 (L-Stabilität).

ESV L-stabil :⇔ $\{z \in \mathbb{C}: \operatorname{Re} z < 0\} \subset \mathcal{S}_\Psi$ **&** $\lim_{\operatorname{Re} z \to -\infty} |S(z)| = 0$

Kurz: L-stabil :⇔ A-stabil **&** „$S(-\infty) = 0$"

Wie findet man L-stabile RK-ESV **?**

$S(-\infty) = 1 - \mathbf{b}^T \mathfrak{A}^{-1} \mathbf{1}$: Falls $\mathbf{b}^T = \mathbf{a}_{s,j}^T$ (Zeile von $\mathfrak{A}$) ⇒ $S(-\infty) = 0$.

Idee:
- Wähle $c_s = 1$ im Kollokations-RK-ESV (eg:rkimpl 5.3.2)
- Wähle $c_1, \ldots, c_{s-1}$ als Knoten einer Quadraturformel maximaler Ordnung.
  (→ Gauss-Radau-Quadratur, Ordnung $2s-1$)

RADAU

Implizite s-stufige L-stabile Radau-ESV, Konvergenzordnung $2s-1$ (→ Thm. thm:cvgkoll 5.4.4)

$$\begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array}$$

Impliziter Euler

$$\begin{array}{c|cc} \frac{1}{3} & \frac{5}{12} & -\frac{1}{12} \\ 1 & \frac{3}{4} & \frac{1}{4} \\ \hline & \frac{3}{4} & \frac{1}{4} \end{array}$$

Radau-ESV, Ordnung 3

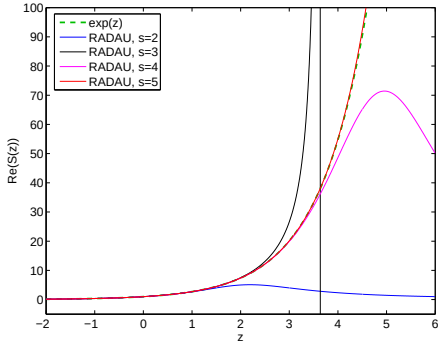
$$\begin{array}{c|ccc} \frac{4-\sqrt{6}}{10} & \frac{88-7\sqrt{6}}{360} & \frac{296-169\sqrt{6}}{1800} & \frac{-2+3\sqrt{6}}{225} \\ \frac{4+\sqrt{6}}{10} & \frac{296+169\sqrt{6}}{1800} & \frac{88+7\sqrt{6}}{360} & \frac{-2-3\sqrt{6}}{225} \\ 1 & \frac{16-\sqrt{6}}{36} & \frac{16+\sqrt{6}}{36} & \frac{1}{9} \\ \hline & \frac{16-\sqrt{6}}{36} & \frac{16+\sqrt{6}}{36} & \frac{1}{9} \end{array}$$

Radau-ESV, Ordnung 5

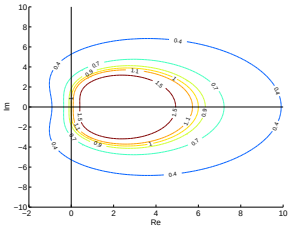
Stabilitätsfunktion s-stufiger Radau-Kollokations-RK-ESVs:

$$S(z) = \frac{P(z)}{Q(z)}, \quad P \in \mathcal{P}_{s-1}, Q \in \mathcal{P}_s .$$

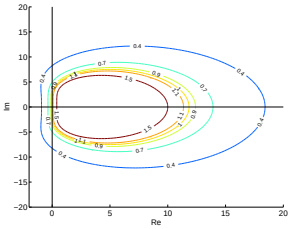
Vorsicht: Auch „$S(\infty) = 0$"



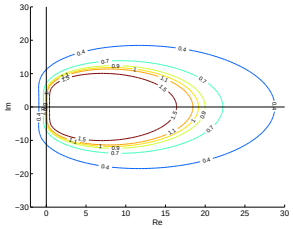
Niveaus der Stabilitätsfunktionen von s-stufigen Radau-Kollokations-RK-ESVs:



$s = 2$



$s = 3$

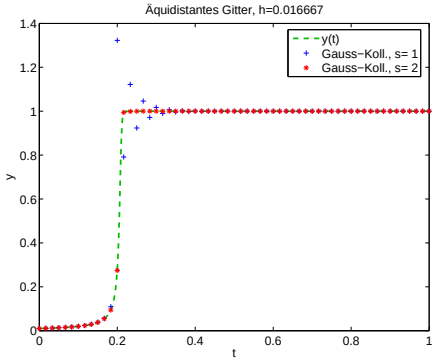


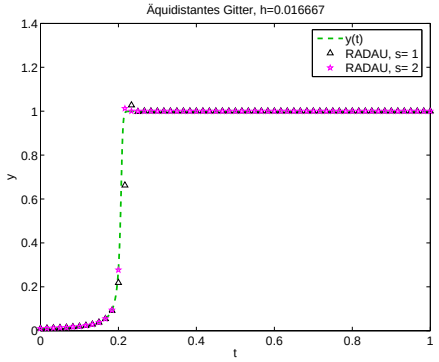
$s = 4$

Beispiel 237 (Implizite ESV für steifes Problem). → Bsp. 233 ex:ode45stiff

$$\dot{y}(t) = \lambda y^2(1-y)\,, \quad \lambda = 500 \quad , \quad y(0) = \frac{1}{100}\,.$$

Qualitatives Verhalten von Gauss-Kollokations-ESV (→ Abschnitt 5.3.1) & RADAU-ESV auf äquidistantem Gitter: sec:kollokation





➢ Falsche Oszillationen bei Gauss-Kollokations-ESV niedriger Ordnung

5.8.2 Implementierung impliziter Runge-Kutta-Einschrittverfahren

Nichtlineares Gleichungssystem für Inkremente $\mathbf{k}_i$ (→ Def. 5.3.3): def:rk

$$\mathbf{k}_i = f(t_0 + c_i h, \mathbf{y}_0 + h\sum_{j=1}^{s} a_{ij}\mathbf{k}_j) \quad \blacktriangleright \quad s \cdot d \text{ Unbekannte !}$$

$$\vec{\mathbf{g}} = (\mathbf{g}_1, \ldots, \mathbf{g}_s)^T\,,$$
$$\mathbf{g}_j := h\sum_{j=1}^{s} a_{ij} f(t_0 + c_j h, \mathbf{y}_0 + \mathbf{g}_j) \quad \blacktriangleright \quad F(\vec{\mathbf{g}}) = \vec{\mathbf{g}} - h\mathfrak{A}\begin{pmatrix} f(t_0 + c_1 h, \mathbf{y}_0 + \mathbf{g}_1) \\ \vdots \\ f(t_0 + c_s h, \mathbf{y}_0 + \mathbf{g}_s) \end{pmatrix} \stackrel{!}{=} 0\,.$$

Approximative Lösung mit vereinfachtem Newton-Verfahren → Bem. 48: Startnäherung $\vec{\mathbf{g}} = 0$ rem:simplNewt

Jacobimatrix $DF(0) = \begin{pmatrix} \mathbf{I} - ha_{11}\frac{\partial f}{\partial \mathbf{y}}(t_0, \mathbf{y}_0) & \cdots & -ha_{1s}\frac{\partial f}{\partial \mathbf{y}}(t_0, \mathbf{y}_0) \\ \vdots & \ddots & \vdots \\ -ha_{s1}\frac{\partial f}{\partial \mathbf{y}}(t_0, \mathbf{y}_0) & \cdots & \mathbf{I} - ha_{ss}\frac{\partial f}{\partial \mathbf{y}}(t_0, \mathbf{y}_0) \end{pmatrix}$.

➢ In jedem (vereinfachten) Newton-Schritt: Lösen eines linearen GLeichungssystems mit Matrix $DF(0) \in \mathbb{R}^{sd,sd}$.

Adaptive MATLAB-Integratoren für steife Probleme: (Schrittweitensteuerung wie in Abschnitt 5.4.1) sec:sws

```
opts = odeset('abstol',atol,'reltol',rtol,'Jacobian',@J)
[t,y] = ode15s/ode23s(odefun,tspan,y0,opts);
```

Beispiel 238 (Adaptives semi-implizites RK-ESV für steifes Problem). Bsp. 233, 237 ex:ode45stiff

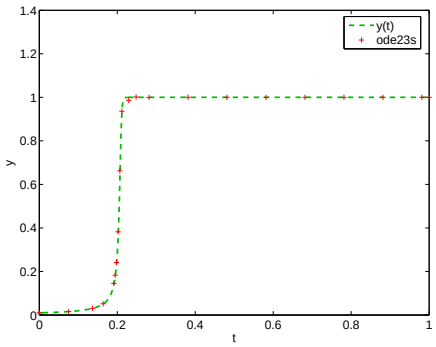
$$\dot{y}(t) = \lambda y^2(1-y)\,, \quad \lambda = 500 \quad , \quad y(0) = \frac{1}{100}\,.$$

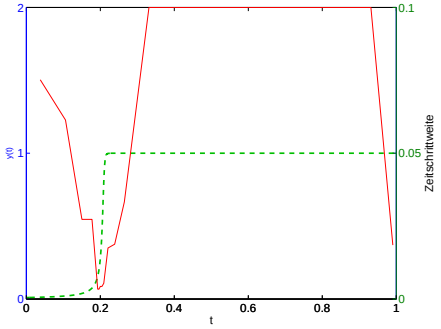
MATLAB-CODE : Semi-Implizites ESV für steifes Problem

```
lambda = 500; tspan = [0 1]; y0 = 0.01;
fun = @(t,x) lambda*x^2*(1-x);
Jac = @(t,x) lambda*(2*x*(1-x)-x^2);
o = odeset('reltol',0.1,'abstol',0.001,'stats','on','Jacobian',Jac);
[t,y] = ode23s(fun,[0 1],y0,o);
```

Statistik:

```
20 successful steps 4 failed attempts 70 function
evaluations
```





➤ Effizientes Verfahren (vgl. Bsp. 233): Keine Schrittweitenbeschränkung für $y \approx 1$ ex:ode45stiff

5.9 Differentiell-Algebraische Anfangswertprobleme

[File: section-differentiell-algebraische-anfangswertprobleme.tex, SVN: section-differentiell-algebraische-anfangswertprobleme.tex 1061 2006-10-20 16:58:14]

Gegeben: „Rechte Seiten" $h : \Omega_1 \times \Omega_2 \subset \mathbb{R}^p \times \mathbb{R}^q \mapsto \mathbb{R}^p$, $g : \Omega_1 \times \Omega_2 \mapsto \mathbb{R}^q$ hinreichend glatt, $p, q \in \mathbb{N}$, $\mathbf{u}_0 \in \Omega_1$, $\mathbf{v}_0 \in \Omega_2$

☞ Differentiell-algebraisches Anfangswertproblem (DAE): Algebraische Nebenbedingung

$$\begin{aligned} \dot{\mathbf{u}} &= h(\mathbf{u}, \mathbf{v}) \,, \\ 0 &= g(\mathbf{u}, \mathbf{v}) \end{aligned} \quad , \quad \begin{aligned} \mathbf{u}(0) &= \mathbf{u}_0 \,, \\ \mathbf{v}(0) &= \mathbf{v}_0 \end{aligned} \quad , \quad \boxed{g(\mathbf{u}_0, \mathbf{v}_0) = 0} \,. \tag{5.9.1}$$ eq:DAE

Konsistente Anfangswerte erforderlich !

Beispiel 239 (Deskriptorform der Pendelgleichung). → Bsp. 216 ex:odemech

Pendelgleichung in nicht-minimalen Koordinaten (Ursprung $\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ an Aufhängung):

$$m \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{p}_1 \\ \dot{p}_2 \end{pmatrix} = \begin{pmatrix} p_1 \\ p_2 \\ -\lambda x_1 \\ -\lambda x_2 - g \end{pmatrix} \quad , \quad \boxed{x_1^2 + x_2^2 = l^2} \,.$$

λ = Lagrange-Multiplikator (↔ $\mathbf{v}$ in (5.9.1)) eq:DAE ➢ Zwangskraft Zwangsbedingung

Differenzieren der Zwangsbedingung ➤ $x_1 p_1 + x_2 p_2 = 0$, (5.9.2) eq:diffnb

Nochmaliges Differenzieren ➤ $(p_1^2 + p_2^2) - \lambda(x_1^2 + x_2^2) - g x_2 = 0$. (5.9.3) eq:diffnb

Annahme: (trifft auf Bsp. 239 nur für differenzierte Nebenbedingung (5.9.3) zu !) ex:daependulum eq:diffnb2

$\frac{\partial g}{\partial \mathbf{v}} \in \mathbb{R}^{q,q}$ regulär entlang einer Lösungskurve von (5.9.1) eq:DAE

➤ Lokale Auflösbarkeit: $\mathbf{v} = G(\mathbf{u})$: (5.9.1) eq:DAE $\Rightarrow$ $\dot{\mathbf{u}} = f(\mathbf{u}, G(\mathbf{u})) \hat{=}$ (5.1.1). eq:awp

DAE von höherem Index: Konzept zur Klassifikation von DAEs HAW91 [26]

Einschrittverfahren für (5.9.1) eq:DAE ?

Idee: Singuläre Störungstechnik

① Betrachte AWPs für Dgl. $\mathbf{M}\dot{\mathbf{y}} = f(\mathbf{y})$, $\mathbf{M} \in \mathbb{R}^{d,d}$, $d = p + q$

② Unter Annahme $\mathbf{M}$ regulär: RK-ESV für $\dot{\mathbf{y}} = \mathbf{M}^{-1} f(\mathbf{y})$

③ Macht Verfahren noch Sinn für singuläres $\mathbf{M}$? Wenn ja → 

➤ Inkrementgleichungen, vgl. Def. 5.3.3: def:rk

$$\mathbf{M}\mathbf{k}_i = f(\mathbf{y}_0 + h \sum_{j=1}^{s} a_{ij} \mathbf{k}_j) \,. \tag{5.9.4}$$ eq:Minc

➢ Falls $\mathbf{M}$ singulär: (5.9.4) eq:Minc nur sinnvoll für *implizite* RK-ESV !

Anwendung auf (5.9.1): eq:DAE

$$\mathbf{M} = \begin{pmatrix} \mathbf{I} & 0 \\ 0 & 0 \end{pmatrix} \quad , \quad \mathbf{I} \in \mathbb{R}^{p,p} \quad , \quad \mathbf{y} = \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \end{pmatrix} \quad , \quad f(\mathbf{y}) = \begin{pmatrix} h(\mathbf{u}, \mathbf{v}) \\ g(\mathbf{u}, \mathbf{v}) \end{pmatrix} \,.$$

Eignen sich alle impliziten ESV ?

Modellproblem: $\begin{aligned} \dot{u} &= v f(u) \,, \\ 0 &= 1 - v \end{aligned}$ ➤ $\begin{aligned} \dot{u} &= v f(u) \,, \\ \epsilon \dot{v} &= 1 - v \end{aligned}$ $0 < \epsilon \ll 1$.

Singulär gestörte Dgl.

Heuristik: ESV tauglich für Modellproblem ↔ ESV tauglich für singulär gestörtes Problem $\forall \epsilon \approx 0$

➤ ESV muss für AWP $\dot{v} = \epsilon^{-1}(1 - v)$, $v(0) = 1$, $0 < \epsilon \ll 1$, Folge $\{v_k\}_{k=0}^{\infty}$ mit $\lim_{k \to \infty} v_k = 1$ liefern !

➤ Notwendig: $\boxed{S(\infty) = 0}$ für Stabilitätsfunktion (→ Thm. 5.6.2) $S(z)$ des ESV thm:SRK

Kandidaten für RK-ESV für (5.9.1): eq:DAE Radau-Verfahren

5.10 Strukturerhaltung

[File: section-strukturerhaltung.tex, SVN: section-strukturerhaltung.tex 1010 2006-09-11 15:44:43Z hiptmair]

(Wir betrachten nur autonome Anfangswertprobleme $\dot{\mathbf{y}} = f(\mathbf{y}), \mathbf{y}(0) = \mathbf{y}_0$, Phasenraum $\Omega \subset \mathbb{R}^d$)

Struktur = Essentielle Qualitative Eigenschaften der Evolution $\boldsymbol{\Phi}^t$ zur ODE
(Hervorragende Darstellung in Monographie [23])

Ziel: „Vererbung" von Struktur an diskrete Evolution $\boldsymbol{\Psi}_h^h$

Wichtig für Langzeitintegration zur Berechnung einer „qualitativ richtigen Lösung"
(im Sinne der Rückwärtsanalyse → Abschnitt 1.8)

5.10.1 Nichtexpansivität

Sei $\mathbf{M} \in \mathbb{R}^{d,d}$ s.p.d.(→ Def. 3.2.8) ➢ Norm $\|\mathbf{y}\|_M := (\mathbf{y}^T\mathbf{M}\mathbf{y})^{1/2}$ auf $\mathbb{R}^d$.

Definition 5.10.1 (Nichtexpansivität). *Eine Evolution $\boldsymbol{\Phi}^t$ zu einer autonomen Dgl. heisst nichtexpansiv, falls*

$$\left\|\boldsymbol{\Phi}^t\mathbf{y} - \boldsymbol{\Phi}^t\tilde{\mathbf{y}}\right\|_M \leq \|\mathbf{y} - \tilde{\mathbf{y}}\|_M \quad \forall \mathbf{y}, \tilde{\mathbf{y}} \in \Omega \, .$$

☞ Analoges Konzept für diskrete Evolutionen zu ESV [11, Abschn. 6.3.3]

Theorem 5.10.2 (B-stabile ESV). *Die diskreten Evolutionen zu Gauss-Kollokations- und Radau-RK-ESV erben die Nichtexpansivität der (exakten) Evolution.*

Beispiel 240 (Gradientenfluss ➞ „Kriechvorgänge").

Potential $V : \mathbb{R}^d \mapsto \mathbb{R}$ strikt konvex:

$$\exists \alpha > 0: \quad (\mathbf{grad}\, V(\mathbf{x}) - \mathbf{grad}\, V(\mathbf{y}))^T(\mathbf{x} - \mathbf{y}) \geq \alpha \|\mathbf{x} - \mathbf{y}\|_2^2 \quad \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^d \, .$$

Gradientenfluss-AWP: $\dot{\mathbf{y}}(t) = -\mathbf{grad}\, V(\mathbf{y}(t)) \quad , \quad \mathbf{y}(0) = \mathbf{y}_0 \in \mathbb{R}^d \, .$

$$\chi(t) := \left\|\boldsymbol{\Phi}^t\mathbf{y} - \boldsymbol{\Phi}^t\tilde{\mathbf{y}}\right\|_2^2 \quad \Rightarrow \quad \dot{\chi}(t) = -2\underbrace{(\mathbf{grad}\, V(\boldsymbol{\Phi}^t\mathbf{y}) - \mathbf{grad}\, V(\boldsymbol{\Phi}^t\tilde{\mathbf{y}}))^T(\boldsymbol{\Phi}^t\mathbf{y} - \boldsymbol{\Phi}^t\tilde{\mathbf{y}})}_{\geq \alpha\|\boldsymbol{\Phi}^t\mathbf{y} - \boldsymbol{\Phi}^t\tilde{\mathbf{y}}\|_2^2} \leq 0 \, .$$

5.10.2 Quadratische erste Integrale

Definition 5.10.3 (Quadratisches erstes Integral). *Eine Funktion $I : \mathbb{R}^d \mapsto \mathbb{R}$, $I(\mathbf{x}) = \mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{q}^T\mathbf{x} + q$, $\mathbf{Q} \in \mathbb{R}^{d,d}$, $\mathbf{q} \in \mathbb{R}^d$, $q \in \mathbb{R}$ ist quadratisches erstes Integral der Evolution $\boldsymbol{\Phi}^t$ zur autonomen Dgl. $\dot{\mathbf{y}} = f(\mathbf{y})$, falls*

$$I(\boldsymbol{\Phi}^t\mathbf{y}) = I(\mathbf{y}) \quad \forall \mathbf{y} \in \Omega, \forall t \in J(\mathbf{y}) \, .$$

Theorem 5.10.4 (Erhaltung quadratischer erster Integrale). *Quadratische erste Integrale der (exakten) Evolution werden auch von der von Gauss-Kollokations-ESV erzeugten diskreten Evolution erhalten.*

Beispiel 241 (Präzession einer Magnetnadel).

$\mathbf{y}(t)$ = Trajektorie der Spitze einer Magnetnadel (im äusseren Feld $\mathbf{h}$ ➢ Bewegungsgleichung

$$\dot{\mathbf{y}} = \mathbf{y} \times \mathbf{h} \quad , \quad \text{Kreuzprodukt} \quad \mathbf{y} \times \mathbf{h} = \begin{pmatrix} y_2h_3 - y_3h_2 \\ y_3h_1 - y_1h_3 \\ y_1h_2 - y_2h_1 \end{pmatrix} \perp \mathbf{y}$$

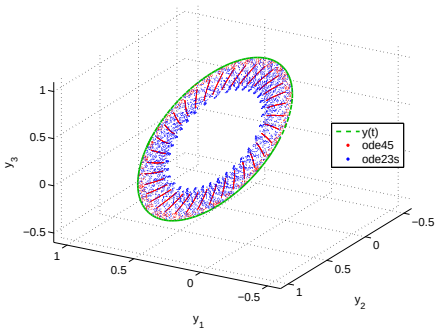
$$(\mathbf{y}(t) \times \mathbf{h}) \cdot \mathbf{y}(t) = 0 \quad \blacktriangleright \quad \frac{d}{dt}\|\mathbf{y}(t)\|_2^2 = 2\dot{\mathbf{y}} \cdot \mathbf{y}(t) = 0 \quad \Rightarrow \quad \|\mathbf{y}(t)\|_2^2 = \text{const} \, .$$

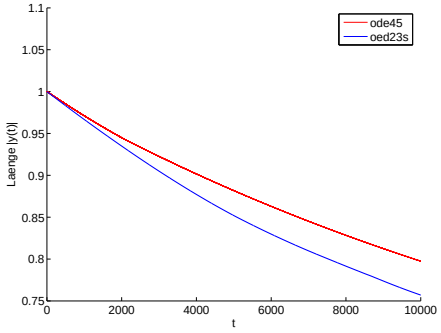
Anfangswert: $\mathbf{y}_0 = (\frac{1}{2}\sqrt{2}, 0, 1, \frac{1}{2}\sqrt{2})^T$

MATLAB-CODE : Berechnung des Präzession einer Magnetnadel

```
h = [-1;-1;-1]; tspan = [0 10000]; y0 = [0.5*sqrt(2);0;0.5*sqrt(2)];
fun = @(t,x) cross(x,h);
Jac = @(t,x) [0 h(3) -h(2); -h(3) 0 h(1); h(2) -h(1) 0];
options = odeset('reltol',0.001,'abstol',1e-4,'stats','on');
[t45,y45] = ode45(fun,tspan,y0,options);
options = odeset('reltol',0.001,'abstol',1e-4,'stats','on','Jacobian',Jac);
[t23,y23] = ode23s(fun,tspan,y0,options);
```

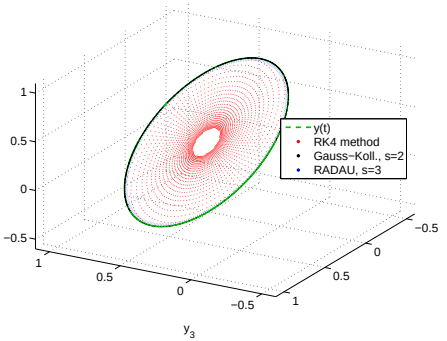
ode45: 24537 successful steps, 7432 failed attempts, 191815 function evaluations
ode23s: 93447 successful steps, 4632 failed attempts, 289607 function evaluations

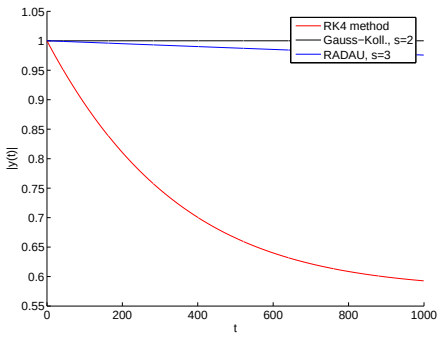




➢ Keine Erhaltung von $\|\mathbf{y}(t)\|_2$ über *lange Zeiten*

Einschrittverfahren auf äquidistanten Gittern:





5.10.3 Symplektizität

Gegeben: (Glatte) HAM Hamilton-Funktion $H : \Omega \subset \mathbb{R}^{2d} \mapsto \mathbb{R}$, $d \in \mathbb{N}$ (Energie)
→ Vorlesung „Theoretische Mechanik”

Oft: $H(\mathbf{p},\mathbf{q}) = \frac{1}{2}\|\mathbf{p}\|_2^2 + V(\mathbf{q})$, Potential $V : \Omega_{\mathbf{q}} \mapsto \mathbb{R}$ (5.10.1) eq:Hmec

Kinetische Energie — Potentielle Energie

HAMGL
Hamiltonsche (Bewegungs-)Gleichungen der klassischen Mechanik

$$\dot{\mathbf{p}} = -\frac{\partial H}{\partial \mathbf{q}}(\mathbf{p},\mathbf{q}) \quad , \quad \dot{\mathbf{q}} = \frac{\partial H}{\partial \mathbf{p}}(\mathbf{p},\mathbf{q}) \, . \tag{5.10.2}$$

eq:HAM

$\mathbf{q} \hat{=}$ (verallgemeinerte) Ortskoordinate, $\mathbf{p}$ (verallgemeinerte) Impulskoordinate

➢ Evolution $\mathbf{\Phi}_H^t$ zu (5.10.2) erfüllt $H(\mathbf{\Phi}_H^t(\mathbf{q},\mathbf{q})) = \text{const}$ für alle $(\mathbf{p},\mathbf{q})^T \in \Omega$ und Zeiten t
(➞ „Energieerhaltung”)

Zusätzliche Struktur: $\mathbf{\Phi}_H^t$ SYMPLECT symplektisch → HLW02 [23, Ch. VI]

Erklärung für $d = 1$:

5.10
p. 637

t fest ➢ $\mathbf{\Phi}_H^t : \Omega \mapsto \Omega$ Diffeomorphismus des Phasenraums: $(\mathbf{\Phi}_H^t)^{-1} = \mathbf{\Phi}_H^{-t}$

Lokale „Flächenverzerrung” ↔ $\det J(t;p,q)$, $J(t;p,q) := \dfrac{\partial \mathbf{\Phi}_H^t(p,q)}{\partial(p,q)}$

$t \mapsto J(t;p,q)$ erfüllt Variationsgleichung eq:VARG (5.2.1):

$$\dot{J}(t;p,q) = \begin{pmatrix} -\frac{\partial^2 H}{\partial p \partial q}(\mathbf{\Phi}_H^t(p,q)) & -\frac{\partial^2 H}{\partial q^2}(\mathbf{\Phi}_H^t(p,q)) \\ \frac{\partial^2 H}{\partial p^2}(\mathbf{\Phi}_H^t(p,q)) & \frac{\partial^2 H}{\partial p \partial q}(\mathbf{\Phi}_H^t(p,q)) \end{pmatrix} J(t;p,q) \, .$$

$$\blacktriangleright \quad \frac{d}{dt}\det J(t;p,q) = \frac{d}{dt}\left(\frac{\partial p(t)}{\partial p}\frac{\partial q(t)}{\partial q} - \frac{\partial p(t)}{\partial q}\frac{\partial q(t)}{\partial p}\right) = 0 \, .$$

Ableitungen nach Anfangswerten

$\blacktriangleright$ Evolution $\mathbf{\Phi}_H^t$ ist flächenerhaltend

Beispiel 242 (Symplektizität des Phasenflusses für Pendelgleichung). → Bsp. ex:odemech 216

$p \leftrightarrow$ Winkelgeschwindigkeit, $q \leftrightarrow$ Winkelvariable

5.10
p. 638

$$\begin{aligned} \dot{p} &= -\sin q \, , \\ \dot{q} &= p \end{aligned} \quad \blacktriangleright \quad H(p,q) = \frac{1}{2}p^2 - \cos q$$

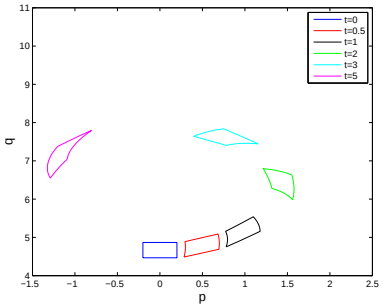
(Gesamtenergie, vgl. (5.10.1)) eq:Hmech eq:Hpend (5.10.3)

5.10
p. 639

5.10
p. 640

Volumenerhaltung im Phasenraum:

Evolution eines quadratischen Volumens ▷



◇

Theorem 5.10.5 (Symplektizität von Gauss-Kollokations-RK-ESV)**.** *Diskrete Evolutionen von Gauss-Kollokations-ESVs (→ Abschnitt 5.3.1) auf äquidistanten Gittern erhalten die Symplektizität einer Evolution (➥ „symplektische Integratoren”).*

sec:kollokation

5.10
p. 641

Bemerkung 243 (Warum symplektische Integratore ?).

Beobachtung: Symplektizität von Integrationsverfahren ➤ Hervorragendes qualitatives Langzeitverhalten für AWP (5.10.2)

eg:HAM

Beispiel 244 (Bewegungsleichung des starren Pendels). → Bsp. 216

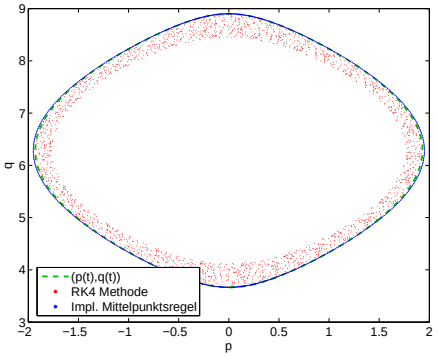
ex:odemech

AWP für (5.10.3) auf $[0, 1000]$, $p(0) = 0$, $q(0) = 7\pi/6$.

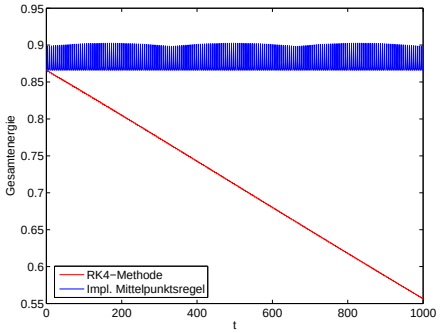
eg:Hpend

Vergleich von klassischem Runge-Kutta-Verfahren (5.3.8) (Ordnung 4) mit 1-stufigem Gauss-Kollokations-ESV (implizite Mittelpunktsregel 5.3.3), äquidistantes Gitter, $h = \frac{1}{2}$:

eg:rk4
eg:implmidprule



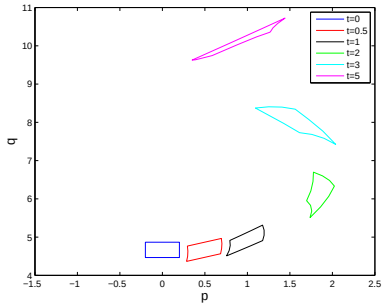
Orbits exakter/diskreter Evolutionen



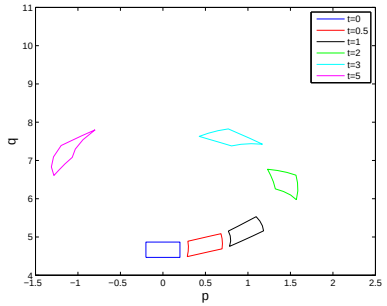
Energieerhaltung diskreter Evolutionen

➢ Keine Energiedrift bei impliziter Mittelpunktsregel (typisch für „symplektische Integratoren”)

5.10
p. 643



Evolution eines quadratischen Volumens
(Explizites Eulerverfahren)



Evolution eines quadratischen Volumens
(Implizite Mittelpunktsregel)

◇

Beispiel 245 (Symplektisches partitioniertes Eulerverfahren). → [23, Kap. VI]

HLW02

$$\begin{aligned} \dot{\mathbf{u}} &= f(\mathbf{u}, \mathbf{v}) \,, \\ \dot{\mathbf{v}} &= g(\mathbf{u}, \mathbf{v}) \,, \end{aligned} \quad \blacktriangleright \quad \begin{aligned} \mathbf{u}_h(t+h) &= \mathbf{u}_h(t) + hf(\mathbf{u}_h(t+h), \mathbf{v}_h(t)) \,, \\ \mathbf{v}_h(t+h) &= \mathbf{v}_h(t) + hg(\mathbf{u}_h(t+h), \mathbf{v}_h(t)) \,, \end{aligned} \tag{5.10.4}$$

eq:sympl

5.10
p. 642

5.10
p. 644

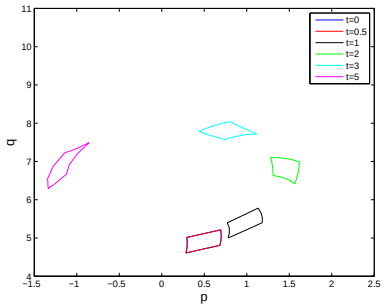
(5.10.2) mit H gemäss (5.10.1)

$$\begin{array}{ll} \mathbf{u}{:=}\mathbf{q} & \\ \mathbf{v}{:=}\mathbf{p} & \mathbf{q}_h(t+h) = \mathbf{q}_h(t) + h\mathbf{p}_h(t)\ , \\ \Rightarrow & \mathbf{p}_h(t+h) = \mathbf{p}_h(t) - h\,\mathbf{grad}\,V(\mathbf{q}_h(t+h))\ , \end{array} \tag{5.10.5}$$

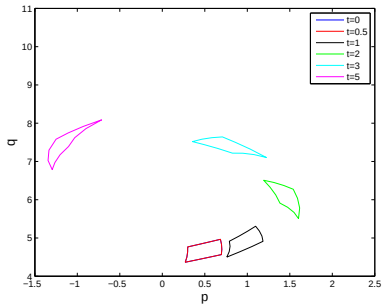
$$\begin{array}{ll} \mathbf{u}{:=}\mathbf{p} & \\ \mathbf{v}{:=}\mathbf{q} & \mathbf{p}_h(t+h) = \mathbf{p}_h(t) - h\,\mathbf{grad}\,V(\mathbf{q}_h(t))\ , \\ \Rightarrow & \mathbf{q}_h(t+h) = \mathbf{q}_h(t) + h\mathbf{p}_h(t+h)\ . \end{array} \tag{5.10.6}$$

➢ Explizite Einschrittverfahren **!**

Anwendung auf Pendelgleichung (5.10.3) → Bsp. 244

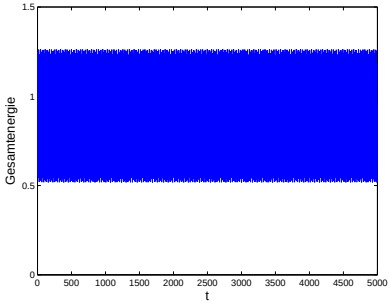


Evolution eines quadratischen Volumens (Symplektisches partitioniertes Eulerverfahren (5.10.5))



Evolution eines quadratischen Volumens (Symplektisches partitioniertes Eulerverfahren (5.10.6))

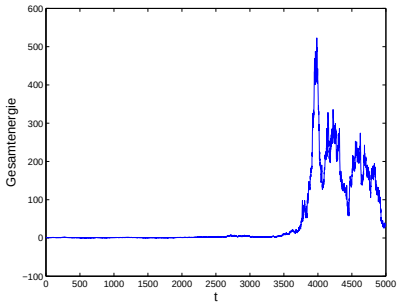
Energieerhaltung des symplektischen partitionierten Eulerverfahrens (5.10.6) ($p(0) = 0, q(0) = \frac{7\pi}{6}$)



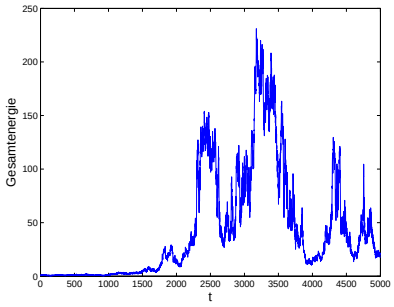
◇

Beispiel 246 (Symplektische Integratoren und variable Schrittweite). Fortsetzung Bsp. 245

Symplektisches Eulerverfahren (5.10.6) für (5.10.3) auf $[0, T]$, $T = 5000$. Erratische variable Schrittweite $h_i = 0.5(1 + 0.5(rand() - 0.5))$, $i = 1, \ldots, 10000$, $p(0) = 0$, $q(0) = 7\pi/6$



Energiedrift bei variabler Schrittweite (Symplektisches partitioniertes Eulerverfahren (5.10.5))



Energiedrift bei variabler Schrittweite (Symplektisches partitioniertes Eulerverfahren (5.10.6))

◇

5.10.4 Reversibilität

$\boldsymbol{\Phi}^{s,t} \hat{=}$ Evolutionsoperator zum AWP (5.1.1) → Sect. 5.1

Thm. 5.1.2 $\Rightarrow$ $\quad \forall (t,\mathbf{y}) \in \widehat{\Omega}\colon \quad (\boldsymbol{\Phi}^{t,s} \circ \boldsymbol{\Phi}^{s,t})\mathbf{y} = \mathbf{y} \quad \forall s \in J(t,\mathbf{y})\,.$ (5.10.7)

Definition 5.10.6 (Symmetrisches Einschrittverfahren)**.** *Ein ESV für das AWP (5.1.1) mit diskreter Evolution $\boldsymbol{\Psi}_h^{s,t}$ heisst symmetrisch, falls*

$$\forall (t,\mathbf{y}) \in \widehat{\Omega}\colon \quad (\boldsymbol{\Psi}_h^{t,t+h} \circ \boldsymbol{\Psi}_h^{t+h,t})\mathbf{y} = \mathbf{y} \quad \textit{für hinreichend kleines } h\,.$$

Wann ist s-stufiges, $s \in \mathbb{N}$, Runge-Kutta-Einschrittverfahren (→ Def. 5.3.3)

$$\mathbf{k}_i = f(\mathbf{y}_0 + h\sum_{j=1}^{s} a_{ij}\mathbf{k}_j) \quad , \quad \mathbf{y}_1 = \mathbf{y}_0 + h\sum_{i=1}^{s} b_i\mathbf{k}_i\,,$$

für autonomes AWP symmetrisch ? ⇨ Falls invariant gegen Vertauschung $\mathbf{y}_0 \leftrightarrow \mathbf{y}_1$, $h \leftrightarrow -h$!

Hinreichend: $\quad b_j = a_{ij} + a_{1+s-i,1+s-j}\,, \quad 1 \le i,j \le s\,.$

Wichtige Konsequenz für symmetrische RK-ESV:

autonomes AWP $\quad \dot{\mathbf{y}} = f(\mathbf{y}) \quad , \quad \mathbf{y}(0) = \mathbf{y}_0$ mit $\quad f(\mathsf{T}\mathbf{y}) = -\mathsf{T}f(\mathbf{y}) \quad \forall \mathbf{y} \in \Omega\,,$ (5.10.8)

$\mathsf{T} : \Omega \mapsto \Omega \hat{=}$ *bijektive Selbstabbildung* des Phasenraumes.

$\Rightarrow \quad \mathsf{T} \circ \boldsymbol{\Phi}^t = (\boldsymbol{\Phi}^t)^{-1} \circ \mathsf{T}$ für hinreichend kleines t . (5.10.9)

Lemma 5.10.7. *Die diskrete Evolution $\boldsymbol{\Psi}_h^t$ für ein Runge-Kutta-Einschrittverfahren (→ Def. 5.3.3) zu autonomem AWP (5.10.8) erfüllt*

$$\mathsf{T} \circ \boldsymbol{\Psi}_h^h = \boldsymbol{\Psi}_h^{-h} \circ \mathsf{T} \quad \textit{für kleine } h\,.$$

Z.B. expliziter Euler (5.3.1):

$$(\mathsf{T} \circ \boldsymbol{\Phi}_h^h)(\mathbf{y}) = \mathsf{T}\mathbf{y} + h\mathsf{T}f(\mathbf{y}) \overset{(5.10.8)}{=} \mathsf{T}\mathbf{y} - hf(\mathsf{T}\mathbf{y}) = (\boldsymbol{\Phi}_h^{-h} \circ \mathsf{T})(\mathbf{y}) \quad \forall \mathbf{y} \in \Omega\,.$$

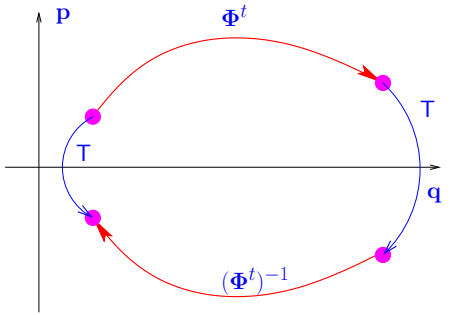
Korollar 5.10.8 (T-Reversibilität symmetrischer RK-ESV)**.** *Die diskrete Evolution $\boldsymbol{\Phi}_h^t$ für ein symmetrisches RK-ESV (→ Def. 5.10.6) zu autonomem AWP (5.10.8) erfüllt für hinreichend kleines h*

$$\mathsf{T} \circ \boldsymbol{\Psi}_h^h = (\boldsymbol{\Psi}_h^h)^{-1} \circ \mathsf{T}\,. \quad \leftrightarrow \quad (5.10.9)$$

Warum ist das interessant ?

Revesibilität mechanischer Systeme mit Hamilton-Funktion (5.10.1) → Abschn. 5.10.3:

$$\mathsf{T}\begin{pmatrix}\mathbf{p}\\ \mathbf{q}\end{pmatrix} = \begin{pmatrix}-\mathbf{p}\\ \mathbf{q}\end{pmatrix} \quad \blacktriangleright \quad (5.10.2) \;=\; \text{AWP der Klasse (5.10.8)}$$



Interpretation:
(5.10.7) = „Rückwärtsevolution" nach Umkehr der Geschwindigkeitsrichtung

Verwende $\binom{\mathbf{p}}{\mathbf{q}} \mapsto \binom{-\mathbf{p}}{\mathbf{q}}$-reversible Einschrittverfahren für konservative mechanische Systeme !

Kandidat: Implizite Mittelpunktsregel (5.3.3): symplektisch, symmetrisch 
implizit

Geht es auch explizit ? (vgl. Bsp. 2.4.5)

Idee: Hintereinanderschaltung symplektischer Eulerverfahren (5.10.5), (5.10.6) mit Schrittweiten $h \leftarrow h/2$

➤ Störmer-Verlet-Verfahren [24]

Für (5.10.2) mit $H(\mathbf{p},\mathbf{q}) = \frac{1}{2}\|\mathbf{p}\|_2^2 + V(\mathbf{q})$ gemäss (5.10.1)

$$\begin{aligned} \mathbf{q}_h(t+\tfrac{1}{2}h) &= \mathbf{q}_h(t) + \tfrac{h}{2}\mathbf{p}_h(t)\ , \\ \mathbf{p}_h(t+h) &= \mathbf{p}_h(t) - h\,\mathbf{grad}\,V(\mathbf{q}_h(t+\tfrac{1}{2}h))\ , \\ \mathbf{q}_h(t+h) &= \mathbf{q}_h(t+\tfrac{1}{2}h) + \tfrac{h}{2}\mathbf{p}_h(t+h)\ . \end{aligned} \qquad (5.10.10)$$

Symmetrisches, symplektisches, explizites Verfahren der Ordnung 2

Beispiel 247 (Molekulardynamik). (→ Vorlesung „rechnergestützte Chemie")

- Phasenraum für $n \in \mathbb{N}$ Atome in $d \in \mathbb{N}$ Dimensionen: $\Omega = \mathbb{R}^{2dn}$ (Positionen $\mathbf{q} = [\mathbf{q}^1,\ldots,\mathbf{q}^n]^T \in \mathbb{R}^{dn}$, Impulse $\mathbf{p} = [\mathbf{p}^1,\ldots,\mathbf{p}^n]^T \in \mathbb{R}^{dn}$)
- Gesamtenergie (Hamilton-Funktion): $H(\mathbf{p},\mathbf{q}) = \frac{1}{2}\|\mathbf{p}\|_2^2 + V(\mathbf{q})$

LJP
Lenard-Jones-Potential: $V(\mathbf{q}) = \sum_{j=1}^{n}\sum_{i\neq j}\mathcal{V}(\left\|\mathbf{q}^i - \mathbf{q}^j\right\|_2)$, $\mathcal{V}(\xi) = \xi^{-12} - \xi^{-6}$.

(5.10.11)

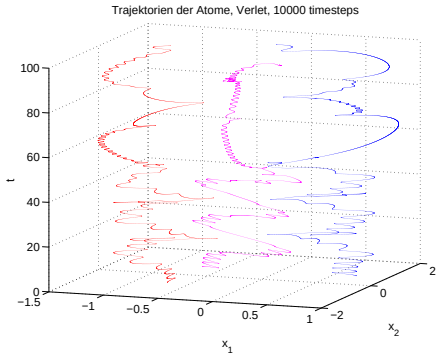
Bewegungsgleichungen (5.10.2):

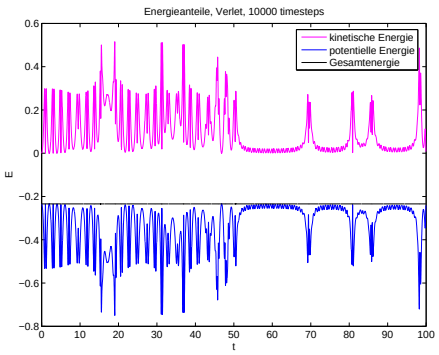
$$\dot{\mathbf{p}}^j = -\sum_{i\neq j}\mathcal{V}'(\left\|\mathbf{q}^j - \mathbf{q}^i\right\|_2)\frac{\mathbf{q}^j - \mathbf{q}^i}{\left\|\mathbf{q}^j - \mathbf{q}^i\right\|_2} \quad , \quad \dot{\mathbf{q}}^j = \mathbf{p}^j\ , j = 1,\ldots,n\ .$$

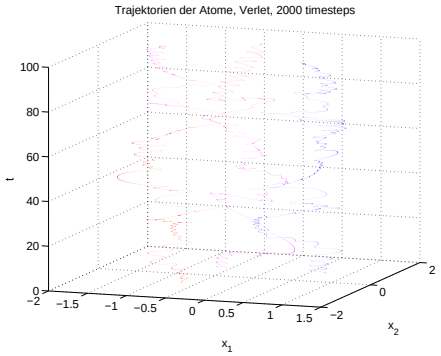
Störmer-Verlet-Verfahren (5.10.10):

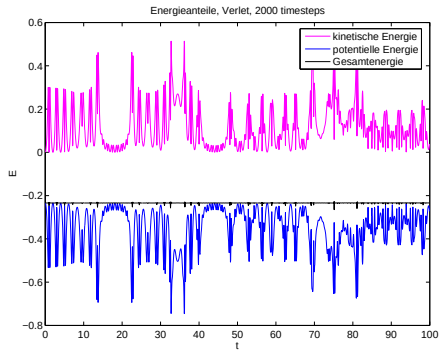
$$\begin{aligned} \mathbf{q}_h(t+\tfrac{1}{2}h) &= \mathbf{q}_h(t) + \tfrac{h}{2}\mathbf{p}_h(t)\ , \\ \mathbf{p}_h^j(t+h) &= \mathbf{p}_h^j(t) - h\sum_{i\neq j}\mathcal{V}'(\left\|\mathbf{q}_h^j(t+\tfrac{1}{2}h) - \mathbf{q}_h^i(t+\tfrac{1}{2}h)\right\|_2)\frac{\mathbf{q}_h^j(t+\tfrac{1}{2}h) - \mathbf{q}_h^i(t+\tfrac{1}{2}h)}{\left\|\mathbf{q}_h^j(t+\tfrac{1}{2}h) - \mathbf{q}_h^i(t+\tfrac{1}{2}h)\right\|_2}\ , \\ \mathbf{q}_h(t+h) &= \mathbf{q}_h(t+\tfrac{1}{2}h) + \tfrac{h}{2}\mathbf{p}_h(t+h)\ . \end{aligned}$$

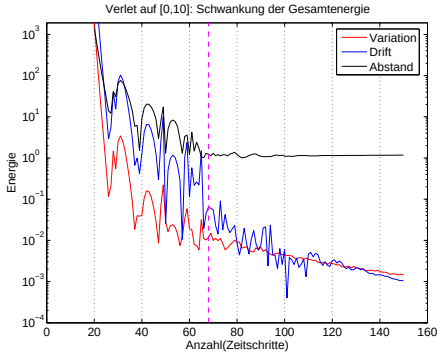
Simulation mit $d = 2$, $n = 3$, $\mathbf{q}^1(0) = \frac{1}{2}\sqrt{2}\binom{-1}{-1}$, $\mathbf{q}^2(0) = \frac{1}{2}\sqrt{2}\binom{1}{1}$, $\mathbf{q}^3(0) = \frac{1}{2}\sqrt{2}\binom{-1}{1}$, $\mathbf{p}(0) = 0$, Endzeitpunkt $T = 100$











$T = 10, d = 2, n = 3, \mathbf{q}^1(0) = \frac{1}{2}\sqrt{2}\binom{-1}{-1}, \mathbf{q}^2(0) = \frac{1}{2}\sqrt{2}\binom{1}{1}, \mathbf{q}^3(0) = \frac{1}{2}\sqrt{2}\binom{-1}{1}, \mathbf{p}(0) = 0.$

Variation $= \sum_{i=1}^{N-1}|E_{\mathrm{tot}}((i+1)h) - E_{\mathrm{tot}}(ih)|$,

Drift $= |E_{\mathrm{tot}}(T) - E_{\mathrm{tot}}(0)|$,

Abstand $= \max\{\left\|\mathbf{q}_h^j(T)\right\|_2, j = 1,2,3\}$.

Instabilität für grosse Zeitschritte **?**

Beispiel 248 (Harmonischer Oszillator).

Modellproblem für (5.10.2): $p, q \in \mathbb{R}$, $\Omega = \mathbb{R}^2$, $H(p,q) = \frac{1}{2}p^2 + \frac{1}{2}aq^2$

(5.10.2) $\dot{p} = -aq$, $\dot{q} = p \quad \Leftrightarrow \quad \ddot{q} = -aq^2$. (5.10.12)

Äquivalente Formulierung des Störmer-Verlet-Verfahrens (5.10.10) für (5.10.12)

$$q_h(t-\tfrac{1}{2}h) = q_h(t-h) + \tfrac{1}{2}hp_h(t-h)\,, \qquad q_h(t+\tfrac{1}{2}h) = q_h(t) + \tfrac{1}{2}hp_h(t)\,,$$
$$p_h(h) = p_h(t-h) - ahq_h(t-\tfrac{1}{2}h)\,, \quad \& \quad p_h(t+h) = p_h(t) - ahq_h(t+\tfrac{1}{2}h)\,,$$
$$q_h(h) = q_h(t-\tfrac{1}{2}h) + \tfrac{1}{2}hp_h(t)\,, \qquad q_h(t+h) = q_h(t+\tfrac{1}{2}h) + \tfrac{1}{2}hp_h(t+h)\,,$$

$$\Updownarrow$$

$$q_h(t+h) - 2q_h(t) + q_h(t-h) = -ah^2q_h(t) \tag{5.10.13}$$

eq:stv3t

Dreitermrekursion

Ansatz: $q_h(kh) = \xi^k$ ➢ (5.10.13) ⇒ $\xi^2 - (2-ah^2)\xi + 1 = 0$

CHAREQ

Characteristische Gleichung of (5.10.13)

$$\exists \xi_* \in \mathbb{C}\colon \quad \xi_*^2 - (2-ah^2)\xi_* + 1 = 0 \quad \wedge \quad |\xi_*| > 1 \iff ah^2 > 4\,.$$

∃ Lösungen von (5.10.13): $|q_h(kh)| \to \infty$ für $k \to \infty$ ➢ Instabilität

5.11 Splittingverfahren

[File: section-splittingverfahren.tex, SVN: section-splittingverfahren.tex 1300 2007-02-01 15:58:07Z kalai]

Autonomes AWP mit additiv zerlegter rechter Seite:

$$\dot{\mathbf{y}} = f(\mathbf{y}) + g(\mathbf{y}) \quad , \quad \mathbf{y}(0) = \mathbf{y}_0\,, \tag{5.11.1}$$

eq:splitav

mit $f : \Omega \subset \mathbb{R}^d \mapsto \mathbb{R}^d$, $g : \Omega \subset \mathbb{R}^d \mapsto \mathbb{R}^d$ Lipschitz-stetig.

Evolutionen: $\boldsymbol{\Phi}_f^t \leftrightarrow$ AWP für $\dot{\mathbf{y}} = g(\mathbf{y})$,
$\boldsymbol{\Phi}_g^t \leftrightarrow$ AWP für $\dot{\mathbf{y}} = h(\mathbf{y})$.

Annahme: $\boldsymbol{\Phi}_f^t$, $\boldsymbol{\Phi}_g^t$ (analytisch) bekannt

Idee: Konstruiere Einschrittverfahren gemäss

$$\mathbf{y}_h(t+h) = (\boldsymbol{\Phi}_g^h \circ \boldsymbol{\Phi}_f^h)\,\mathbf{y}_h(t)\,, \tag{5.11.2}$$

eq:lietrott

$$\mathbf{y}_h(t+h) = (\boldsymbol{\Phi}_f^{h/2} \circ \boldsymbol{\Phi}_g^h \circ \boldsymbol{\Phi}_f^{h/2})\,\mathbf{y}_h(t)\,. \tag{5.11.3}$$

eq:strang

Terminologie: Splitting (5.11.2) = LIETROTTER Lie-Trotter-Splitting
Splitting (5.11.3) = STRANG Strang-Splitting

Theorem 5.11.1 (Konsistenzordnung einfacher Splittingverfahren). *Die ESV (5.11.2) und (5.11.3) haben die Konsistenzordnungen (→ Def. 5.4.2) 1 bzw. 2.*

eq:lietrotter eq:strang def:ssmloccons

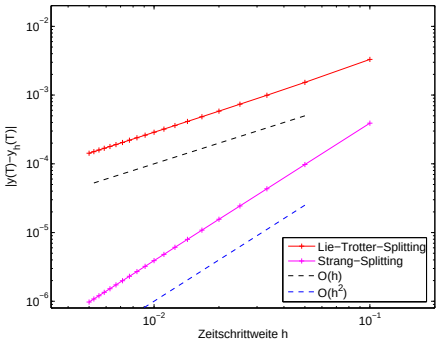
Beispiel 249 (Konvergenz einfacher Splittingverfahren).

$$\dot{y} = \underbrace{\lambda y(1-y)}_{=:f(y)} + \underbrace{\sqrt{1-y^2}}_{=:g(y)} \quad , \quad y(0) = 0\,.$$

- $\boldsymbol{\Phi}_f^t y = \dfrac{1}{1+(y^{-1}-1)e^{-\lambda t}}\,, t > 0, y \in]0,1]$ (Logistische Differentialgleichung (5.3.4))

eq:logdgl

- $\boldsymbol{\Phi}_g^t y = \begin{cases} \sin(t + \arcsin(y)) & \text{, falls } t + \arcsin(y) < \frac{\pi}{2}\,, \\ 1 & \text{, sonst,} \end{cases} \quad t > 0, y \in [0,1]\,.$



Numerisches Experiment:

$T = 1$, $\lambda = 1$, Vergleich von Splittingverfahren (konstante Schrittweite) mit hochgenauer numerischer Lösung erhalten durch

```
f=@(t,x) λ*x*(1-x)+sqrt(1-x^2);
options=odeset('reltol',1.0e-10,...
               'abstol',1.0e-12);
[t,yex]=ode45(f,[0,1],y0,options);
```

◁ Fehlerverhalten zum Endzeitpunkt $T = 1$

Idee: Ersetze

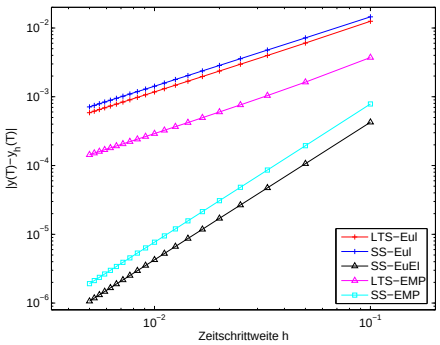
Exakte Evolutionen ⟶ diskrete Evolutionen
$\boldsymbol{\Phi}_g^h$, $\boldsymbol{\Phi}_f^h$ ⟶ $\boldsymbol{\Psi}_{h,g}^h$, $\boldsymbol{\Psi}_{h,f}^h$

Beispiel 250 (Inexakte Splittingverfahren). Forsetzung Bsp. 249

ex:splitcvg

ex:splitcvg

AWP von Bsp. 249, inexakte Splittingverfahren auf der Grundlage verschiedener inexakter Basisverfahren:



LTS-Eul Explizites Eulerverfahren (5.3.1) $\to$ $\Psi^h_{h,g}$, $\Psi^h_{h,f}$ + Lie-Trotter-Splitting (5.11.2)

SS-Eul Explizites Eulerverfahren (5.3.1) $\to$ $\Psi^h_{h,g}$, $\Psi^h_{h,f}$ + Strang-Splitting (5.11.3)

SS-EuEI Strang-Splitting (5.11.3): Explizites Eulerverfahren (5.3.1) $\circ$ exakte Evolution $\Phi^h_g \circ$ implizites Eulerverfahren (5.3.1)

LTS-EMP Explizite Mittelpunktsregel (5.3.6) $\to$ $\Psi^h_{h,g}$, $\Psi^h_{h,f}$ + Lie-Trotter-Splitting (5.11.2)

SS-EMP Explizite Mittelpunktsregel (5.3.6) $\to$ $\Psi^h_{h,g}$, $\Psi^h_{h,f}$ + Strang-Splitting (5.11.3)

Beachte:

SS-EuEI $=$ symmetrisches Einschrittverfahren $\to$ Def. 5.10.6

Warum das Ganze **?**

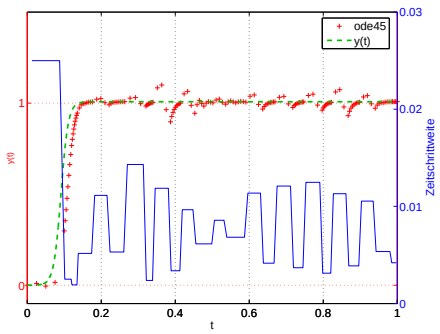
$\dot{\mathbf{y}} = f(\mathbf{y}) + g(\mathbf{y})$ „schwierig” $\dot{\mathbf{y}} = f(\mathbf{y})$ $\to$ Steifheit, aber analytisch integrierbar

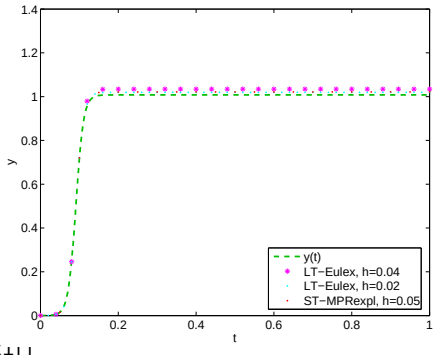
(z.B. steif $\to$ Abschn. 5.7.1) $\dot{\mathbf{y}} = g(\mathbf{y})$ „einfach” (explizit) numerisch integrierbar

Beispiel 251 (Abspaltung steifer Anteile). vgl. Bsp. 233

AWP $\dot{y} = \lambda y(1-y) + \alpha \sin(y)$, $\lambda = 100$, $\alpha = 1$, $y(0) = 10^{-4}$.

Kleine Störung





Lösung aus ode45, siehe Bsp. 233

inexakte Splittingverfahren: Näherungslösungen

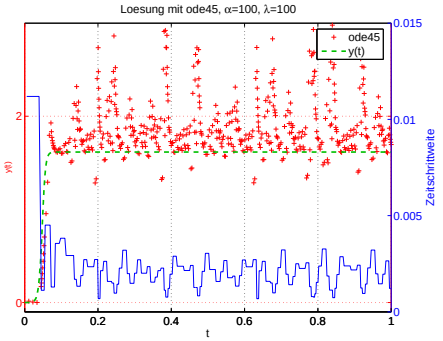
Gesamtzahl der Zeitschritte	ode45:	152
	LT-Eulex, $h = 0.04$:	25
	LT-Eulex, $h = 0.02$:	50
	ST-MPRexpl, $h = 0.05$:	20

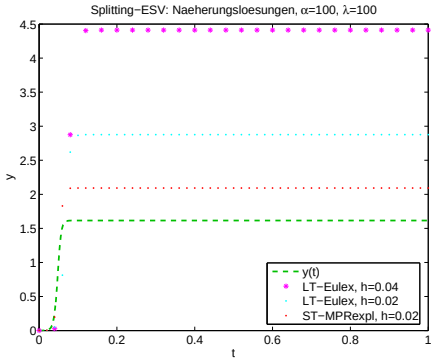
LT-Eulex: $\dot{y} = \lambda y(1-y) \to$ exakte Evolution, $\dot{y} = \alpha \sin y \to$ expl. Euler (5.3.1) **&** Lie-Trotter-Splitting (5.11.2)

ST-MPRexpl: $\dot{y} = \lambda y(1-y) \to$ exakte Evolution, $\dot{y} = \alpha \sin y \to$ expl. Mittelpunktsregel (5.3.6) **&** Strang-Splitting (5.11.3)

Klar: Störung $\uparrow$ $\Rightarrow$ obige Splittingverfahren schlechter

$\lambda = 100$ und $\alpha = 100$:





5.12 Verfahren für oszillatorische Differentialgleichungen

[File: section-verfahren-fuer-oszillatorische-probleme.tex, SVN: section-verfahren-fuer-oszillatorische-probleme.tex 1183 2006-11-29 10:46:03Z hiptmair]

Prototyp: $\ddot{y} = -\omega^2 y \quad , \quad y(0) = y_0,\ \dot{y}(0) = v_0$

$$\blacktriangleright \quad y(t) = \alpha\cos(\omega t) + \beta\sin(\omega t)\ , \quad \alpha, \beta \in \mathbb{R}$$

Verallgemeinerung (skalar): $\ddot{y} = -\omega^2 y + g(y) \quad , \quad y(0) = y_0,\ \dot{y}(0) = v_0\ ,$ (5.12.1) eq:scaosc

mit Lipschitz-stetiger Störung $g : \mathbb{R} \mapsto \mathbb{R}$.

Verallgemeinerung (vektoriell) $\ddot{\mathbf{y}} = -\mathbf{A}\mathbf{y} + g(\mathbf{y}) \quad , \quad \mathbf{y}(0) = \mathbf{y}_0,\ \dot{\mathbf{y}}(0) = \mathbf{v}_0\ ,$ (5.12.2) eq:vecosc

$\mathbf{A} \in \mathbb{R}^{d,d}$ symmetrisch positiv definit (→ Def. 3.2.8), $g : \mathbb{R}^d \mapsto \mathbb{R}^d$ def:spd

Bemerkung 252.

(5.12.1) eq:scaosc $v:=\dot{y}$ ⇒ $\dfrac{d}{dt}\begin{pmatrix} y \\ v \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -\omega^2 & 0 \end{pmatrix}\begin{pmatrix} y \\ v \end{pmatrix} + \begin{pmatrix} 0 \\ g(y) \end{pmatrix}$. (5.12.3) eq:osc1

Lösung von (5.12.3) eq:osc1 durch Variation der Konstanten:

$$\begin{pmatrix} y(t) \\ v(t) \end{pmatrix} = \begin{pmatrix} \cos t\omega & \omega^{-1}\sin t\omega \\ -\omega\sin t\omega & \cos t\omega \end{pmatrix}\begin{pmatrix} y_0 \\ v_0 \end{pmatrix} + \int_0^t \begin{pmatrix} \omega^{-1}\sin(t-s)\omega \\ \cos(t-s)\omega \end{pmatrix} g(y(s))\,\mathrm{d}s \qquad (5.12.4)$$ eq:solosc

Bemerkung 253. $y(t)$ löst (5.12.1) eq:scaosc & $G' = g$ ➡ $\frac{1}{2}|\dot{y}|^2 + \frac{1}{2}\omega^2 y(t) - G(y(t)) \equiv \mathrm{const}$.

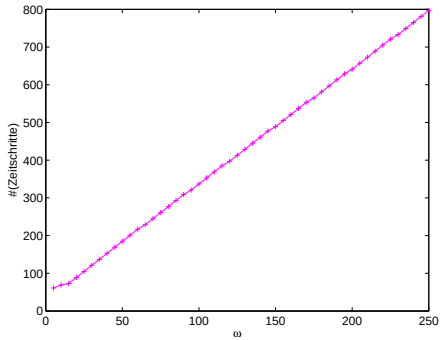
„Energie" für ODE (5.12.1) eq:scaosc

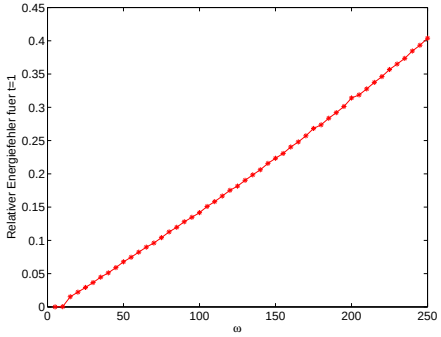
Beispiel 254 (Standardintegratoren für oszillatorische Differentialgleichung).

Adaptives explizites RK-ESV (Abschn. 5.4.1) sec:sws für (5.12.1) eq:scaosc:

```
y0=[1;0]; f=@(t,x) [0,1;-omega^2,0]*x + 20*[0;sin(x(1))];
options=odeset('reltol',1.0e-2,'abstol',1.0e-5);
[t,y]=ode45(f,[0,1],y0,options);
```

(„Energie" $\hat{=}$ Invariante aus Bem. 253) rem:Hconsosc





$\omega \uparrow \quad \Rightarrow \quad$ Oszillationen in $y(t) \uparrow \quad \Rightarrow \quad$ Anzahl Zeitschritte $\uparrow$

Ziel: *Effiziente* numerische Integration von (5.12.1)/(5.12.2) eq:scaosc eq:vecosc auch für $\omega \gg 1$ bzw. $\lambda_{\max}(\mathbf{A}) \gg 1$

Idee: Verwende analytische Lösungsdarstellung (5.12.4) eq:solosc zur numerischen Integration:

$$y(t \pm h) = \cos(h\omega)y(t) \pm \tfrac{\sin h\omega}{\omega}\dot{y}(t) + \int_0^{\pm h} \tfrac{\sin(\pm h - s)\omega}{\omega} \cdot g(y(t+s))\,\mathrm{d}s \qquad (5.12.5)$$ eq:solos

$g \equiv \mathrm{const}$ eq:solosc2 eq:scaosc eq:ccg ⇒ $y(t+h) - 2\cos(h\omega)y(t) + y(t-h) = h^2\left(\dfrac{\sin(\frac{1}{2}h\omega)}{\frac{1}{2}h\omega}\right)^2 g\ .$ (5.12.6) eq:solosc

➢ Gautschis Zweischrittverfahren ($y_h(t+h)$ aus $y_h(t), y_h(t-h)$) für (5.12.1) eq:scaosc

$$y_h(t+h) - 2\cos(h\omega)y_h(t) + y_h(t-h) = h^2\left(\frac{\sin(\frac{1}{2}h\omega)}{\frac{1}{2}h\omega}\right)^2 g(y_h(t))\ . \qquad (5.12.7)$$ eq:Gauts

Notwendig: Startschritt aus (5.12.4) eq:solosc

$$y_h(h) = \cos(h\omega)y_0 + \frac{\sin h\omega}{\omega}v_0 + \tfrac{1}{2}h^2\left(\frac{\sin(\frac{1}{2}h\omega)}{\frac{1}{2}h\omega}\right)^2 g(y_0)\ . \qquad (5.12.8)$$ eq:starts

Ableitungsnäherung: Aus (5.12.5) eq:solosc2 für $g \equiv \mathrm{const}$:

$$\blacktriangleright \quad y(t+h) - y(t-h) = 2h\,\frac{\sin h\omega}{h\omega}\dot{y}(t) \quad \Rightarrow \quad v_h(t) = \frac{h\omega}{\sin h\omega} \cdot \frac{y_h(t+h) - y_h(t-h)}{2h}\ . \qquad (5.12.9)$$ eq:ydotre

Bemerkung 255. Gautschi-Verfahren (5.12.7) eq:Gauts, (5.12.8) eq:starts für vektorielles Problem (5.12.2) eq:vecosc **?**

Ersetze $\cos(h\omega) \mapsto \cos h\mathbf{A}\ , \quad \left(\dfrac{\sin(\frac{1}{2}h\omega)}{\frac{1}{2}h\omega}\right)^2 \mapsto 4(h\mathbf{A})^{-2}\sin^2(\tfrac{1}{2}h\mathbf{A})\ .$

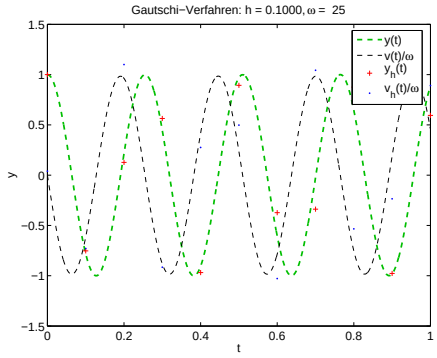
Erinnerung: Funktionalkalkül für Matrizen:

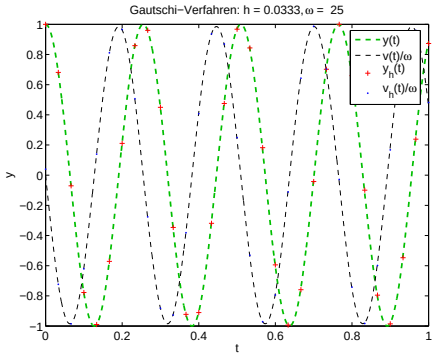
$$\mathbf{S}^{-1}\mathbf{A}\mathbf{S} = \mathrm{diag}(\lambda_1, \ldots, \lambda_n) \quad \blacktriangleright \quad f(\mathbf{A}) = \mathbf{S}\,\mathrm{diag}(f(\lambda_1), \ldots, f(\lambda_n))\mathbf{S}^{-1}\ .$$

Beispiel 256 (Gautschis Zweischrittverfahren).

Anfangswertproblem vom Typ (5.12.1) eq:scaosc auf $[0, 1]$:

$$\ddot{y} = -\omega^2 y + \sin y \quad , \quad y(0) = 1\ , \quad \dot{y}(0) = 0\ .$$

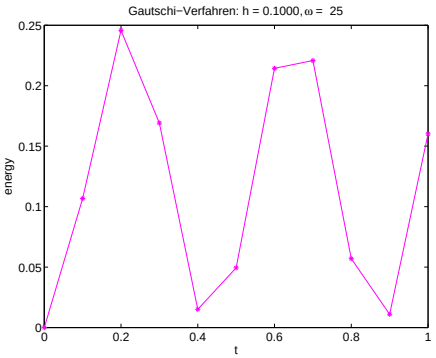




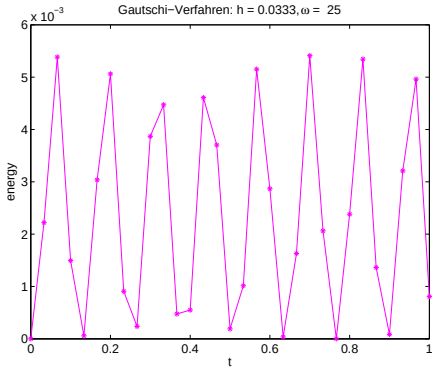
$\omega = 25$: y_h folgt (oszillatorischer Lösung), auch wenn $h \approx \frac{2\pi}{\omega}$

Relativer Fehler in „Energie” (→ Bem. 253) für $t = 1$:

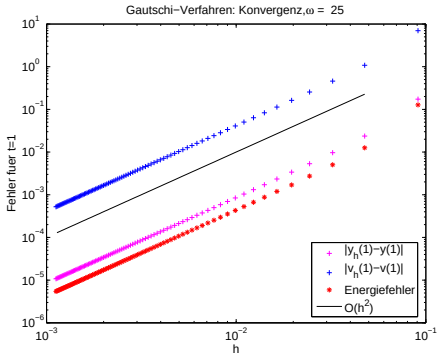
```
rem:Hconsosc
```

Zeitschritt $h = 0.1$



Zeitschritt $h = 0.033$



Beobachtung:

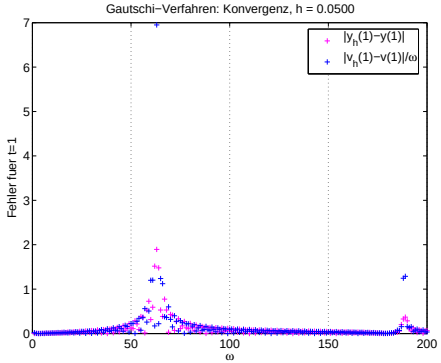
Alle Fehler $\approx O(h^2)$

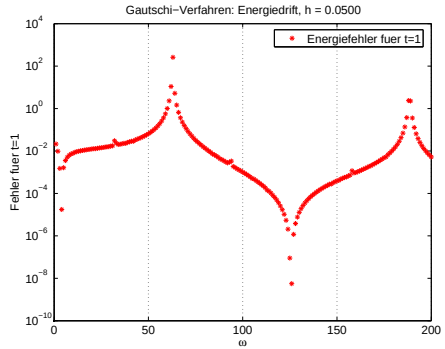
$\Updownarrow$

Konvergenzordnung 2

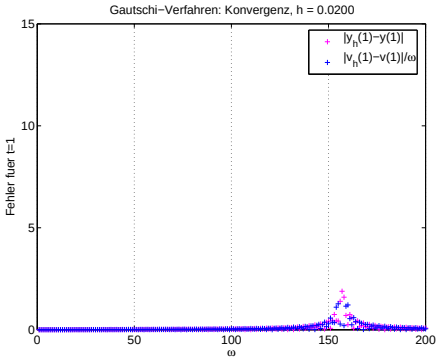
Ziel erreicht **?**

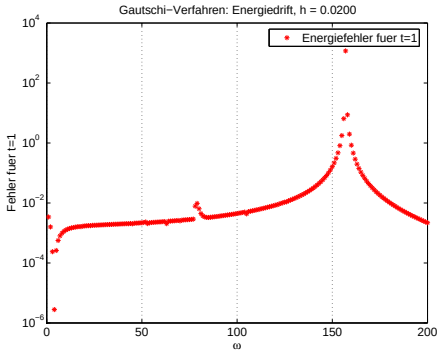
Fehler für fixes h in Abhängigkeit von ω:





$h = 0.05$: Was pasiert für $\omega \approx 61$, $\omega \approx 123$, $\omega \approx 185$ **?** Instabilität ?





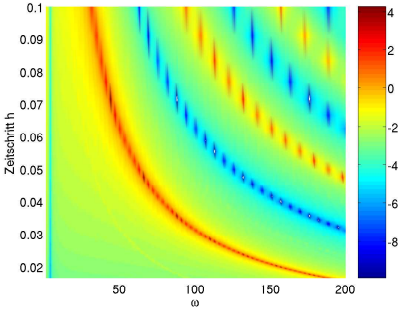
➢ h-Abhängigkeit kritischer Frequenzen

h-Abhängigkeit kritischer Frequenzen

Logarithmus $\log_{10}$ des relativen Energiefehlers zum Endzeitpunkt $t = 0$ ▷

Beobachtung:

$h\omega$-Abhängigkeit kritischer Frequenzen



Modellproblem:

$$\ddot{y} = -\omega^2 y + \alpha y\ , \quad \alpha \ll \omega^2\ . \tag{5.12.10}$$

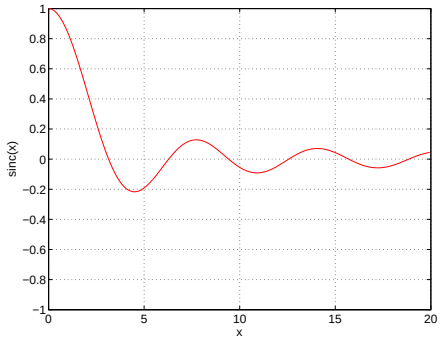
eq:oscm

➢ Gautschi-Verfahren (5.12.7), Schrittweite h: Dreitermrekursion, vgl. (5.10.13),

eq:Gautschi eq:sty3t

$$y_h(t+h) - \left\{2\cos(h\omega) + h^2\alpha\,\mathrm{sinc}^2(\tfrac{1}{2}h\omega)\right\} y_h(t) + y_h(t-h) = 0\ . \tag{5.12.11}$$

eq:gautsc



Die sinc-Funktion:

$$\mathrm{sinc}(x) := \frac{\sin x}{x}$$

➢ Analytisch auf $\mathbb{R}$

➢ $|\mathrm{sinc}(x)| \le 1$ mit globalem Maximum in $x = 0$

Analyse von (5.12.11): Ansatz $y_h(kh) = \xi^k$ ↔ Charakteristische (quadratische) Gleichung

eq:gautschimp

∃ Lösungen $y_h(kh)$ von (5.12.11): $\lim_{k\to\infty} y_h(hk) = \pm\infty$ ⇔ $\left|2\cos(h\omega) + h^2\alpha\,\mathrm{sinc}^2(\tfrac{1}{2}h\omega)\right| > 2$

➤ $(\cos h\omega \approx 1 \Leftrightarrow h\omega \approx 2\pi l,\ l \in \mathbb{Z}) \Rightarrow y_h(hk) \to \pm\infty$ auch für $h \ll 1$.

HOL98a

Abhilfe [30]: „Filterung": Dämpfung von α, falls $h\omega \approx 2\pi l$:

eq:Gautschistartstep

In (5.12.7), (5.12.8) ersetze: $g(y_h(t)) \mapsto g(\psi(h\omega)y_h(t))$, $\psi(\xi) := \mathrm{sinc}^2\xi(1 + \frac{1}{2}(1-\cos\xi))$

➢ Modifiziertes Gautschi-Verfahren:

$$y_h(t+h) - 2\cos(h\omega)y_h(t) + y_h(t-h) = h^2\left(\frac{\sin(\frac{1}{2}h\omega)}{\frac{1}{2}h\omega}\right)^2 g(\psi(h\omega)y_h(t))\ . \tag{5.12.12}$$

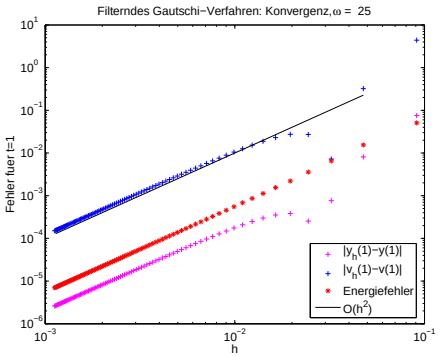
eq:gaufilt

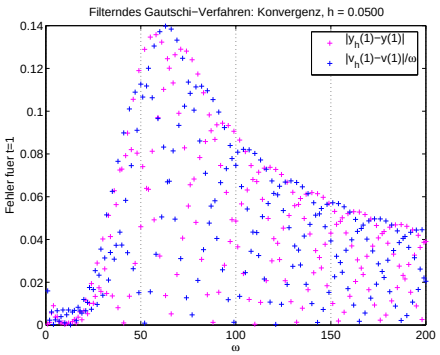
Beispiel 257 (Modifiziertes Gautschi-Verfahren).

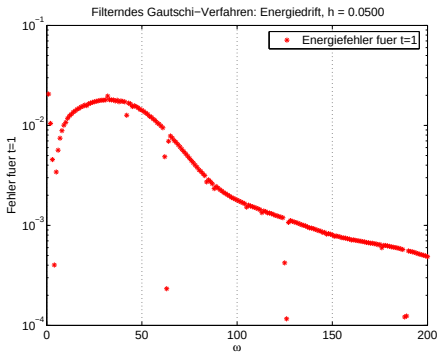
AWP aus Bsp. 5.12.7, Integration gemäss (5.12.12), Filterfunktion $\psi(\xi) := \mathrm{sinc}^2\xi(1 + \frac{1}{2}(1 - \cos\xi))$

eq:Gautschi eq:gaufilt

Konvergenzordnung 2 ▷



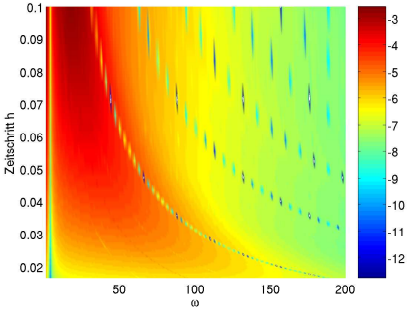




$h\omega$-Abhängigkeit der Energiedrift ▷

Beobachtung:
(5.12.12): eq:gaufilt

- Keine Instabilität
- Im Vergleich zu (5.12.7) (eq:Gautschi ⟶ Bsp. ex:Gautschi 256) deutlich reduzierte Energiedrift (Skala!)



5.12
p. 677

Verzeichnisse

5.12
p. 679

Index

5.12
p. 678

5.12
p. 680

5.12
p. 685

5.12
p. 686

5.12
p. 687

5.12
p. 688

Beispiele und Bemerkungen

5.12
p. 701

Definitionen und Konzepte

5.12
p. 702

5.12
p. 703

5.12
p. 704

MATLAB-CODE-Fragmente

Symbolverzeichnis

Literaturverzeichnis

[1] M. Abramowitz und I. Stegun, *Handbook of Mathematical Functions*, Dover Publications, New York, 1970.

[2] P. Amestoy, T. Davis und I. Duff, *An approximate minimum degree ordering algorithm*, SIAM Journal on Matrix Analysis and Applications, 17 (1996), S. 886–905.

[3] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine und H. V. der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM, Philadelphia, PA, 2nd ed., 1994.

[4] S. Börm, L. Grasedyck und W. Hackbusch, *Introduction to hierarchical matrices with applications*, Engineering Analysis with Boundary Elements, 27 (2003), S. 405–422.

[5] R. Brent, C. Percival und P. Zimmermann, *Error bounds on complex floating-point multiplication*, Math. Comp., (2007).

5.12 p. 710

[6] Q. Chen und I. Babuska, *Approximate optimal points for polynomial interpolation of real functions in an interval and in a triangle*, Comp. Meth. Appl. Mech. Engr., 128 (1995), S. 405–417.

[7] C. Chui, *Wavelets and spline interpolation*, in Advances in numerical analysis. Vol. 2: Wavelets, subdivision algorithms, and radial basis functions, Proc. 4th Summer Sch., Lancaster/UK 1990, Clarendon Press, oxford ed., 1992, S. 1–35.

[8] D. Coppersmith und T. Rivlin, *The growth of polynomials bounded at equally spaced points*, SIAM J. Math. Anal., 23 (1992), S. 970–983.

[9] P. Davis, *Interpolation and Approximation*, Dover, New York, 1975.

[10] P. Deuflhard, *Newton Methods for Nonlinear Problems*, Band 35 in Springer Series in Computational Mathematics, Springer, Berlin, 2004.

[11] P. Deuflhard und F. Bornemann, *Numerische Mathematik II*, DeGruyter, Berlin, 2 ed., 2002.

[12] P. Deuflhard und A. Hohmann, *Numerische Mathematik I*, DeGruyter, Berlin, 3 ed., 2002.

[13] P. Duhamel und M. Vetterli, *Fast fourier transforms: a tutorial review and a state of the art*, Signal Processing, 19 (1990), S. 259–299.

[14] A. Dutt und V. Rokhlin, *Fast Fourier transforms for non-equispaced data II*, Appl. Comput. Harmon. Anal., 2 (1995), S. 85–100.

[15] F. Fritsch und R. Carlson, *Monotone piecewise cubic interpolation*, SIAM J. Numer. Anal., 17 (1980), S. 238–246.

5.12 p. 709

5.12 p. 711

[16] M. Gander, W. Gander, G. Golub und D. Gruntz, *Scientific Computing: An introduction using MATLAB*, Springer, 2005. In Vorbereitung.

[17] W. Gander und W. Gautschi, *Adaptive quadrature - revisited*, BIT, 40 (2000), S. 84–101.

[18] J. Gilbert, C.Moler und R. Schreiber, *Sparse matrices in MATLAB: Design and implementation*, SIAM Journal on Matrix Analysis and Applications, 13 (1992), S. 333–356.

[19] G. Golub und C. Van Loan, *Matrix computations*, John Hopkins University Press, Baltimore, London, 2nd ed., 1989.

[20] L. Greengard und V. Rokhlin, *A new version of the fast multipole method for the Laplace equation in three dimensions*, Acta Numerica, (1997), S. 229–269.

[21] W. Hackbusch, *Iterative Lösung großer linearer Gleichungssysteme*, B.G. Teubner–Verlag, Stuttgart, 1991.

[22] W. Hackbusch und S. Börm, *Data-sparse approximation by adaptive $\mathcal{H}^2$-matrices*, Computing, 69 (2002), S. 1–35.

[23] E. Hairer, C. Lubich und G. Wanner, *Geometric numerical integration*, Band 31 in Springer Series in Computational Mathematics, Springer, Heidelberg, 2002.

[24] ——, *Geometric numerical integration illustrated by the Störmer-Verlet method*, Acta Numerica, 12 (2003), S. 399–450.

[25] E. Hairer, S. Norsett und G. Wanner, *Solving Ordinary Differential Equations I. Nonstiff Problems*, Springer-Verlag, Berlin, Heidelberg, New York, 2 ed., 1993.

5.12 p. 712

[26] E. HAIRER UND G. WANNER, *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems*, Springer-Verlag, Berlin, Heidelberg, New York, 1991.

[27] C. HALL UND W. MEYER, *Optimal error bounds for cubic spline interpolation*, J. Approx. Theory, 16 (1976), S. 105–122.

[28] M. HANKE-BOURGEOIS, *Grundlagen der Numerischen Mathematik und des Wissenschaftlichen Rechnens*, Mathematische Leitfäden, B.G. Teubner, Stuttgart, 2002.

[29] N. HIGHAM, *Accuracy and Stability of Numerical Algorithms*, SIAM, Philadelphia, PA, 2 ed., 2002.

[30] M. HOCHBRUCK UND C. LUBICH, *A Gautschi-type method for oscillatory second-order differential equations*, Numer. Math., 83 (1999), S. 403–426.

[31] D. MCALLISTER UND J. ROULIER, *An algorithm for computing a shape-preserving osculatory quadratic spline*, ACM Trans. Math. Software, 7 (1981), S. 331–347.

[32] J. MELENK, *Vorlesungsaufzeichnungen numerik*, skriptum, Unversität Regensburg, 2003.

[33] K. NEYMEYR, *A geometric theory for preconditioned inverse iteration applied to a subspace*, Tech. Report 130, SFB 382, Universität Tübingen, Tübingen, Germany, November 1999. Submitted to Math. Comp.

[34] ——, *A geometric theory for preconditioned inverse iteration: III. Sharp convergence estimates*, Tech. Report 130, SFB 382, Universität Tübingen, Tübingen, Germany, November 1999.

[35] M. OVERTON, *Numerical Computing with IEEE Floating Point Arithmetic*, SIAM, Philadelphia, PA, 2001.

[36] A. D. H.-D. QI, L.-Q. QI UND H.-X. YIN, *Convergence of Newton's method for convex best interpolation*, Numer. Math., 87 (2001), S. 435–456.

[37] C. RADER, *Discrete Fourier transforms when the number of data samples is prime*, Proceedings of the IEEE, 56 (1968), S. 1107–1108.

[38] R. RANNACHER, *Einführung in die numerische mathematik*. Vorlesungsskriptum Universität Heidelberg, 2000. http://gaia.iwr.uni-heidelberg.de/.

[39] J. RIVLIN, *The Chebyshev Polynomials*, Wiley-Interscience, 1984.

[40] V. ROKHLIN, *Rapid solution of integral equations of classical potential theory*, J. Comp. Phys., 60 (1985), S. 187–207.

[41] H. SCHWARZ, *Methode der finiten Elemente*, Band 47 in Leitfäden der angewandten Mathematik und Mechanik, Teubner-Verlag, Stuttgart, 3rd ed., 1991.

[42] M. STEWART, *A superfast toeplitz solver with improved numerical stability*, SIAM J. Matrix Analysis Appl., 25 (2003), S. 669–693.

[43] J. STOER, *Einführung in die Numerische Mathematik*, Heidelberger Taschenbücher, Springer, 4 ed., 1983.

[44] F. TISSEUR UND K. MEERBERGEN, *The quadratic eigenvalue problem*, SIAM Review, 43 (2001), S. 235–286.

[45] H. WERNER UND R. SCHABACK, *Praktische Mathematik II*, Hochschulktext, Springer, Berlin, 1972.