Grundlagen der Numerischen Mathematik

a.o.Univ.Prof. Mag.Dr. Stephen Keeling
Homepage

Literatur:
Numerical Analysis, R.L. Burden und J.D. Faires

Unterlagen:
Keeling, Lehre, Universität Graz

Inhaltsverzeichnis I

Inhaltsverzeichnis II

Inhaltsverzeichnis III

Inhaltsverzeichnis IV

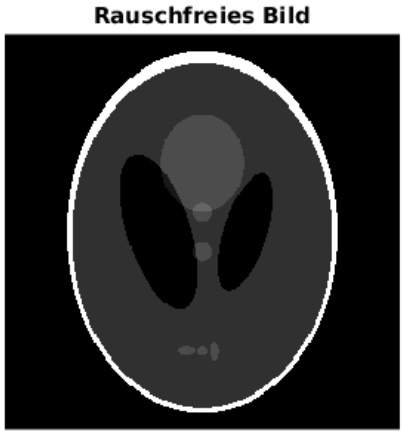
Inhaltsverzeichnis V

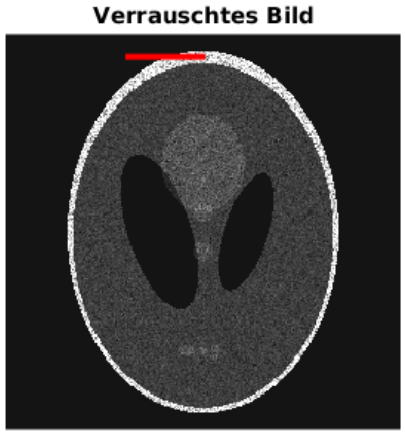
Was ist Numerische Mathematik?

- Experiment zu Hause:
  - Im Taschenrechner:
    - 2 eingeben,
    - dann 10-Mal $\sqrt{\cdot}$ drücken,
    - dann 10-Mal x^2 drücken,
    - und schließlich 2 subtrahieren.
  - Ergebnis $\neq 0$. Warum?
  - Zahlen werden nicht genau gespeichert!
- Wie sollen solche Probleme im Computer gelöst werden?
  - $A\boldsymbol{x} = \boldsymbol{b}$,
  - $f(x) = 0$,
  - $u''(x) = f(x)$.
- Antwort: Es geht nur Annäherungsweise, und
- eine gute *numerische* Methode soll nicht erlauben, dass Fehler sich schlecht aufbauen!
- *Numerische Mathematik* hat mit der Entwicklung solcher Methoden und mit der Analysis unvermeidlicher Fehler zu tun.

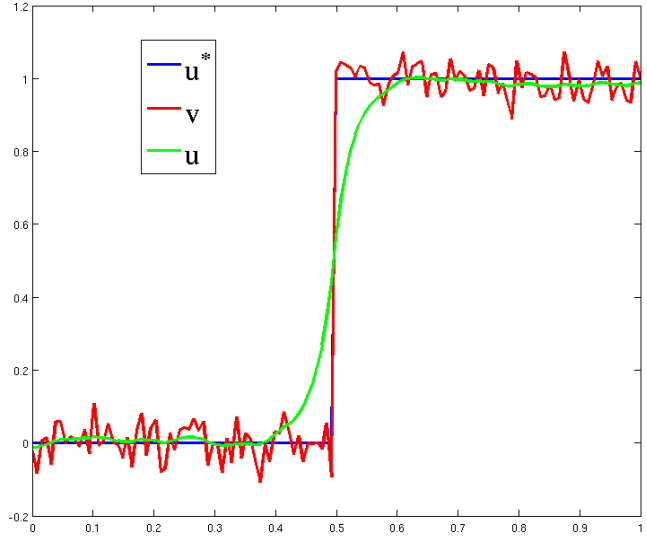
Motivierendes Thema: Signal- und Bildverarbeitung

- Das folgende Beispiel wird mehrmals im Skriptum in verschiedenen Kontexten analysiert.
- $v(x)$ ist ein gegebenes verrauschtes Signal.
- $u^\star(x)$ ist das gewünschte Signal.
- $u(x)$ ist eine Abschätzung von $u^\star(x)$.





Profil durch die Strecke $\longrightarrow$



Motivierendes Thema: Signal- und Bildverarbeitung

- Wie kann $u \approx u^\star$ abgeschätzt werden?
- Ansatz: $\Omega = (0,1)$, $u^\star \approx u = \operatorname{argmin}_w J(w)$,

$$J(u) = \int_\Omega |u - v|^2 dx + \mu \int_\Omega |u'|^2 dx$$

- Qualitativ:
  - $\mu \to \infty \Rightarrow u =$ Mittelwert von v (übergeglättet)
  - $\mu \to 0 \Rightarrow u = v$ (untergeglättet)

  Das beste $\mu^\star$ liegt zwischen diesen Extremen.
- Optimalitätsbedingung zur Bestimmung von u?
- Diskreter Ansatz: $J(u) \approx J_h(\boldsymbol{u})$,

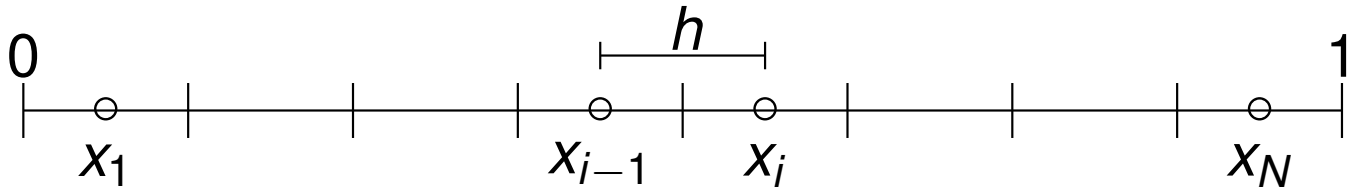
$$J_h(\boldsymbol{u}) = h \sum_{i=1}^{N} (u_i - v_i)^2 + \mu h \sum_{i=1}^{N-1} \left[\frac{u_{i+1} - u_i}{h} \right]^2$$

Details unten.

Motivierendes Thema: Signal- und Bildverarbeitung

- Daten werden üblicherweise diskret gemessen, z.B. $v_i = \frac{1}{h}\int_{x_i \pm h/2} v(x)dx$ oder $v_i = v(x_i)$ auf einem Gitter

$$x_i = (i - \tfrac{1}{2})h, \quad i = 1, \ldots, N, \quad h = 1/N$$



- Fragestellung: Unter welchen Bedingungen gelten diese?

$$h\sum_{i=1}^{N}[u(x_i) - v(x_i)]^2 \quad \xrightarrow{h\to 0} \quad \int_\Omega |u(x) - v(x)|^2 dx$$

$$\mu h \sum_{i=1}^{N-1}\left[\frac{u(x_{i+1}) - u(x_i)}{h}\right]^2 \quad \xrightarrow{h\to 0} \quad \mu\int_\Omega |u'(x)|^2 dx$$

- Sätze und Definitionen aus der Analysis werden jetzt für das notwendige Werkzeug hier in der Einführung gesammelt...

Motivierendes Thema: Signal- und Bildverarbeitung

Satz (Mittelwertsatz): Wenn $f \in \mathcal{C}([a,b])$ differenzierbar in (a,b) ist, dann $\exists c \in (a,b)$ mit $f'(c) = [f(b) - f(a)]/(b-a)$.

- Also $\exists \xi_i \in (x_i, x_{i+1})$ mit

$$\frac{u(x_{i+1}) - u(x_i)}{h} = \frac{u(x_{i+1}) - u(x_i)}{x_{i+1} - x_i} = u'(\xi_i)$$

und

$$\mu h \sum_{i=1}^{N-1} \left[\frac{u(x_{i+1}) - u(x_i)}{h} \right]^2 = \mu \sum_{i=1}^{N-1} [u'(\xi_i)]^2 h \approx \mu \int_\Omega |u'(x)|^2 dx$$

Satz (Mittelwertsatz für Integrale): Wenn $f \in \mathcal{C}([a,b])$ und $g(x) \geq 0$, $\forall x \in [a,b]$, dann $\exists c \in (a,b)$ mit

$$\int_a^b f(x)g(x)dx = f(c) \int_a^b g(x)dx$$

- Themen: Approximation von Ableitungen und Integralen.

Motivierendes Thema: Signal- und Bildverarbeitung

- Für Daten $\{v_i\}$ und Approximationen $\{u_i \approx u(x_i)\}$ seien $\boldsymbol{u} = \langle u_1, \ldots, u_N \rangle^\top$ und $\boldsymbol{v} = \langle v_1, \ldots, v_N \rangle^\top$. Zu minimieren:

$$J_h(\boldsymbol{u}) = h \sum_{i=1}^{N} (u_i - v_i)^2 + \mu h \sum_{i=1}^{N-1} \left[\frac{u_{i+1} - u_i}{h} \right]^2$$

- Optimalität: $\nabla J_h(\boldsymbol{u}) = \nabla_{\boldsymbol{u}} J_h(\boldsymbol{u}) = 0$.

$$\underbrace{\frac{\partial J_h}{\partial u_k}}_{1<k<N} = 2h(u_k - v_k) + \mu h/h^2 [2 \underbrace{(u_k - u_{k-1})}_{i=k-1} - 2 \underbrace{(u_{k+1} - u_k)}_{i=k}]$$

$$\frac{\partial J_h}{\partial u_1} = 2h(u_1 - v_1) + \mu/h [- 2 \underbrace{(u_2 - u_1)}_{i=1}]$$

$$\frac{\partial J_h}{\partial u_N} = 2h(u_N - v_N) + \mu/h [2 \underbrace{(u_N - u_{N-1})}_{i=N-1}]$$

Motivierendes Thema: Signal- und Bildverarbeitung

- oder $\nabla J_h(\boldsymbol{u}) = 2h(\boldsymbol{u} - \boldsymbol{v}) + 2(\mu/h)D\boldsymbol{u}$ wobei

$$D = \begin{bmatrix} 1 & -1 & & & 0 \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ 0 & & & -1 & 1 \end{bmatrix} \in \mathbb{R}^{N \times N}$$

- Optimalität:

$$0 = \frac{1}{2h}\nabla J_h(\boldsymbol{u}) = \boldsymbol{u} - \boldsymbol{v} + \frac{\mu}{h^2} D\boldsymbol{u}$$

oder das lineare Gleichungssystem:

$$\left[\frac{\mu}{h^2} D + I\right] \boldsymbol{u} = \boldsymbol{v}$$

mit $\mathbf{1} = \{1, \ldots, 1\} \in \mathbb{R}^N$ und $I = \mathrm{diag}(\mathbf{1})$.

Motivierendes Thema: Signal- und Bildverarbeitung

- Optimalitätssystem komponentenweise:

$$
\begin{aligned}
-\mu\frac{u_{i+1}-2u_i+u_{i-1}}{h^2} &+ u_i = v_i, \quad i=2,\ldots,N-1\\
-\mu\frac{u_2-u_1}{h^2} &+ u_1 = v_0, \quad i=1\\
-\mu\frac{-u_N+u_{N-1}}{h^2} &+ u_N = v_N, \quad i=N
\end{aligned}
$$

  wobei in den Gleichungen $i=1$ und $i=N$,
  “$u_1 = u_0$” bzw. “$u_N = u_{N+1}$”
  (virtuelle Randbedingungen $u'=0$) zu sehen sind.

- Kontinuum-Ansatz für Optimalität: Richtungableitung sind Null für alle Störungen w:

$$0 = \frac{\delta J}{\delta u}(u;w) = \lim_{\epsilon\to 0}\frac{d}{d\epsilon}J(u+\epsilon w)$$

$$= 2\int_0^1 [w(u-v)+\mu u'w']dx = 2\int_0^1 [u-v-\mu u'']wdx + wu'|_0^1$$

Motivierendes Thema: Signal- und Bildverarbeitung

- Gilt wenn u das Randwertproblem erfüllt:

$$\begin{cases} -\mu u'' + u &= v, & \Omega \\ u' &= 0, & \partial\Omega \end{cases}$$

- Vergleich mit dem diskreten Ergebnis zeigt:

$$u''(x_i) \approx \frac{u_{i+1} - 2u_i + u_{i-1}}{h^2}$$

  Approximation der zweiten Ableitung.

- Hausaufgabe: Schreibe eine Approximation $J_h(\boldsymbol{u})$ für

$$J(u) = \tfrac{1}{2} \int_\Omega (u - v)^2 dx + \mu \int_\Omega \sqrt{|u'|^2 + \varepsilon^2} dx$$

  und leite die Optimalitätsbedingung $\nabla J_h(\boldsymbol{u}) = 0$ her.

Zusammenfassung unserer Fragestellungen

- Wie werden Formeln zur Approximation eines Integrals hergeleitet?
- Formeln für Ableitungen?
- Wann wird die Konvergenz dieser Formeln mit Verfeinerung gewahrleistet, d.h. mit $h \to 0$?
- Welche Methode konvergiert am schnellsten?
- Wie löst man die entstehenden Gleichungssysteme?
- Wenn das Integral $\int_\Omega |u'|^2 d\boldsymbol{x}$ in J mit $\int_\Omega |u'| d\boldsymbol{x}$ ersetzt wird,
  - ist das Entrauschen-Ergebnis viel besser, aber
  - das Gleichungssystem ist nicht linear.
  - Wie wird das nicht lineare Problem gelöst?

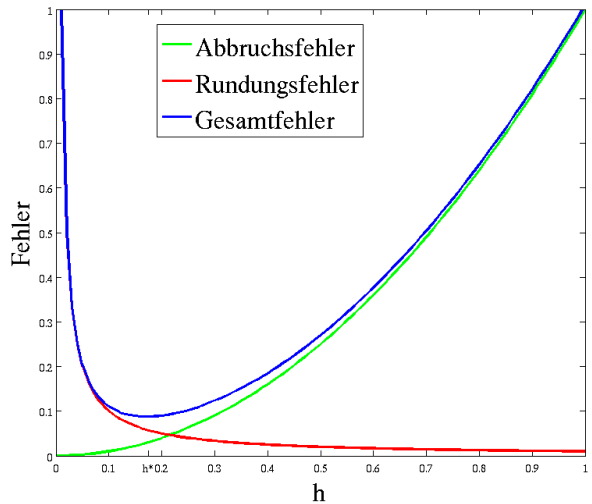
  Solche Probleme werden iterativ gelöst.
- Unter welchen Bedingungen konvergiert das iterative Verfahren? Wie schnell?

Fehleranalyse

- Fehler heissen *Abbruchsfehler*, wenn sie durch eine Diskretisierung und trotz exakter Arithmetik entstehen, z.B. das Residuum in

$$u''(x_i) - \frac{1}{h^2}[u(x_{i+1}) - 2u(x_i) + u(x_{i-1})]$$

  wird nicht berechnet sondern *abgebrochen*.
- Fehler heissen *Rundungsfehler*, wenn Zahlen einer Rechnung nicht exakt gespeichert werden können.
- Diese können üblicherweise so grafisch dargestellt werden:



- Abbruchsfehler werden folgendermaßen abschätzt.

Fehleranalyse

Satz (Taylor): Sei $u \in C^k([a,b])$ und $u^{(k+1)}$ existiert in (a,b). Sei $x_0 \in [a,b]$. Dann $\forall x \in [a,b]$ gilt $u(x) = P(x) + R(x)$, wobei das Taylor-Polynom P und das Restglied R gegeben sind durch

$$P(x) = \sum_{m=0}^{k} u^{(m)}(x_0)\frac{(x-x_0)^m}{m!}, \quad R(x) = \int_{x_0}^{x} u^{(k+1)}(t)\frac{(x-t)^k}{k!}dt$$

und $R(x) = u^{(k+1)}(\xi)\dfrac{(x-x_0)^{k+1}}{(k+1)!}$ für ein ξ zwischen x und x_0.

- Anwendung für die zweite Ableitung: $x_{i+1} - x_i = h$

$$u(x_{i+1}) = u(x_i) + u^{(1)}(x_i)h + u^{(2)}(x_i)\frac{h^2}{2} + u^{(3)}(x_i)\frac{h^3}{6} + u^{(4)}(\xi_{i+1})\frac{h^4}{24}$$

$$u(x_{i-1}) = u(x_i) - u^{(1)}(x_i)h + u^{(2)}(x_i)\frac{h^2}{2} - u^{(3)}(x_i)\frac{h^3}{6} + u^{(4)}(\xi_{i-1})\frac{h^4}{24}$$

Summe:

$$\frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{h^2} = u''(x_i) + F(h)$$

Fehleranalyse

- Für eine glatte Lösung u ist der Abbruchsfehler:

$$F(h) = [u^{(4)}(\xi_{i+1}) + u^{(4)}(\xi_{i-1})]\frac{h^2}{24}$$

- Laut der folgenden Definition gilt $F(h) = \mathcal{O}(h^2)$.

Def: Es gilt $F(x) = \mathcal{O}(G(x))$ für $x \to x_0$ ($x_0 \in \mathbb{R} \cup \{\pm\infty\}$) wenn $\limsup_{x\to x_0} |F(x)/G(x)| < \infty$, und es gilt $F(x) = o(G(x))$ wenn $\lim_{x\to x_0} |F(x)/G(x)| = 0$.

- Wenn die exakte Lösung erfüllt $|u^{(4)}(x)| \leq c$, $\forall x$, dann gilt:

$$\begin{aligned} |F(h)| &\leq |u^{(4)}(\xi_{i+1}) + u^{(4)}(\xi_{i-1})|\frac{h^2}{24} \\ &\leq (|u^{(4)}(\xi_{i+1})| + |u^{(4)}(\xi_{i-1})|)\frac{h^2}{24} \\ &\leq 2c\frac{h^2}{24} \leq \frac{ch^2}{12} \quad \Rightarrow \quad \lim_{h\to 0} |F(h)/h^2| \leq \frac{c}{12} \end{aligned}$$

Es gelten $F(h) = \mathcal{O}(h^2)$ und $F(h) = o(h)$.

Konvergenz

Bemerkung: Es ist gezeigt worden, der (lokale) Abbruchsfehler dieser Approximation der zweiten Ableitung konvergiert zu Null mit Verfeinerung. Es ist aber noch nicht gezeigt worden,

$$\left[\frac{\mu}{h^2}D + I\right]\boldsymbol{u} = \boldsymbol{v} \quad \ldots \quad \begin{cases} -\mu u'' + u &= v, \quad \Omega \\ u' &= 0, \quad \partial\Omega \end{cases}$$

dass die Lösung des linearen Gleichungssystems zur exakten Lösung des Randwertproblems mit Verfeinerung konvergiert,

$$\max_i |u_i - u(x_i)| \to 0,\ h \to 0$$

aber der Taylorsatz gehört zum Werkzeug.

Bemerkung: Die Optimalitätsbedingung $\nabla J_h(\boldsymbol{u}) = 0$ für

$$J(u) = \frac{1}{2}\int_\Omega |u - v|^2 dx + \mu \int_\Omega \sqrt{|u'|^2 + \varepsilon^2} dx$$

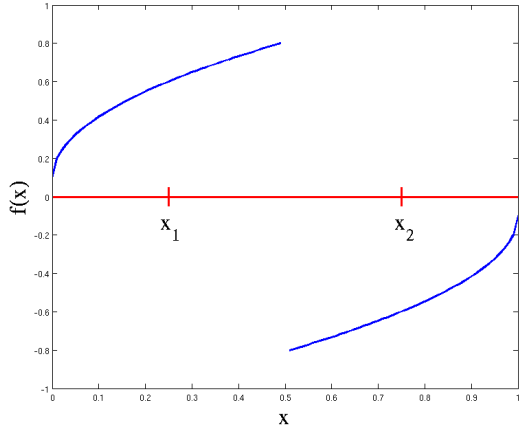
ist nicht linear. Benötigt wird ein Verfahren, mit dem die Nullstelle der Gleichung $\nabla J_h(\boldsymbol{u}) = 0$ bestimmt werden kann...

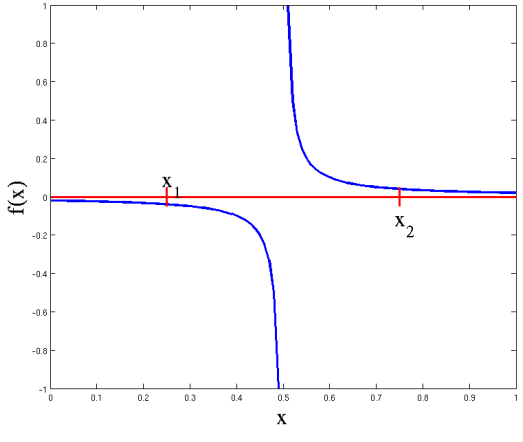
Nullstellen

- Einfaches Beispiel: Eine Nullstelle x_0 von $f : \mathbb{R} \to \mathbb{R}$ soll gefunden werden, wobei durch Versuch und Irrtum x_1 und x_2 vorhanden sind, die $f(x_1)f(x_2) < 0$ erfüllen. Liegt eine Nullstelle zwischen x_1 und x_2?

Satz (Zwischenwert): Wenn $f \in C([a, b])$ und k zwischen $f(a)$ und $f(b)$ liegt, dann existiert $c \in (a, b)$ mit $f(c) = k$ gilt.

- Gegenbeispiele:





Nullstellen

- Bekannte Verfahren zur Bestimmung einer Nullstelle:
  - Bisektion

$$f(a)f(b) < 0 \ldots (a+b)/2 = c \rightarrow \begin{cases} a, & f(c)f(b) < 0 \\ b, & f(a)f(c) < 0 \end{cases}$$

  - Newton

$$x_{k+1} = x_k - f(x_k)/f'(x_k), \quad k = 1, 2, \ldots$$

- Fragestellungen zu solchen iterativen Verfahren:
  - Ist ein Verfahren konvergent?
  - Was soll das Abbruchskriterium sein?
  - Was ist die Geschwindigkeit der Konvergenz?
  - Ist das Verfahren stabil?

Konvergenzgeschwindigkeit

Def: Die Folge $\{x_k\}$ konvergiert für $k \to \infty$ zu x^* mit einer Konvergenzgeschwindigkeit (nicht langsamer als) $\mathcal{O}(y_k)$ wenn

$$x_k = x^* + \mathcal{O}(y_k).$$

Beispiele:

- $x_k = (k+1)/k^2$ konvergiert für $k \to \infty$ zu 0 mit einer Konvergenzgeschwindigkeit $\mathcal{O}(1/k)$,

$$\begin{aligned} \lim_{k\to\infty} |x_k/(1/k)| &= \lim_{k\to\infty} (k+1)/k = 1 \in (0,\infty) \\ \Rightarrow \quad x_k &= 0 + \mathcal{O}(1/k) \end{aligned}$$

- $y_k = 1 + (2k+3)/k^3$ konvergiert für $k \to \infty$ zu 1 mit einer Konvergenzgeschwindigkeit $\mathcal{O}(1/k^2)$,

$$\begin{aligned} \lim_{k\to\infty} |(y_k - 1)/(1/k^2)| &= \lim_{k\to\infty} (2k+3)/k = 2 \in (0,\infty) \\ \Rightarrow \quad y_k &= 1 + \mathcal{O}(1/k^2) \end{aligned}$$

Stabilität

Def: Sei E_k der Fehler eines Verfahrens nach k Operationen, wenn der Anfangsfehler E_0 ist. Wenn gilt

$$|E_k| \leq ck|E_0| \quad 0 < c \neq c(k)$$

ist das Fehlerwachstum linear. Wenn nur gezeigt werden kann,

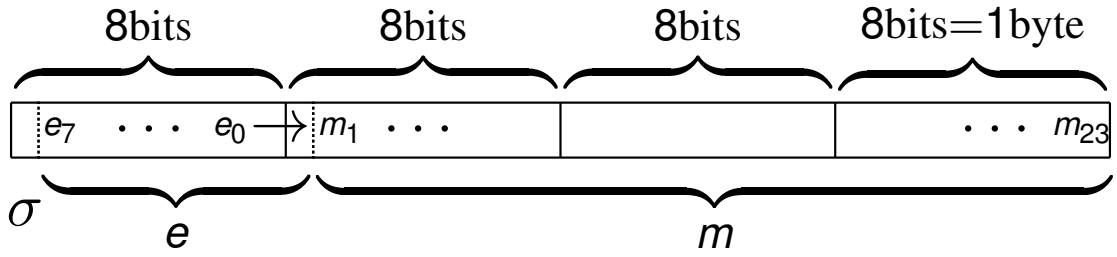
$$|E_k| \leq ca^k|E_0| \quad 0 < c \neq c(k), a > 1$$

ist das Fehlerwachstum exponentiell.
Da Fehlerwachstum in der Regel unvermeidlich ist, wird ein Verfahren stabil genannt, wenn das Fehlerwachstum höchstens linear ist.

- Beispiele:
  - Rekursion 1: $p_0 = 1$, $p_1 = \frac{1}{3}$, $p_k = \frac{10}{3}p_{k-1} - p_{k-2}$, $p_k \approx (\frac{1}{3})^k$.
    Matlab: $|p_k - (\frac{1}{3})^k| \approx \frac{1}{50} \cdot (\frac{7}{3})^k$, instabil!
  - Rekursion 2: $p_0 = 1$, $p_1 = \frac{1}{3}$, $p_k = 2p_{k-1} - p_{k-2}$, $p_k \approx 1 - \frac{2}{3}k$.
    Matlab: $|p_k - (1 - \frac{2}{3}k)| \approx \frac{2}{3}k - \frac{8}{3}$, stabil!
- Die Fehler in diesen Beispielen sind *Rundungsfehler*. Wie entstehen sie genau?

Darstellung von Zahlen im Computer

- Beispiel: IEEE Fließkommazahl *einfacher Genauigkeit* (32 bits, 4 bytes).



$\sigma \rightarrow$Vorzeichen, $e \rightarrow$Exponent, $m \rightarrow$Mantissa

$$e = \langle e_0, \ldots, e_7 \rangle, \quad m = \langle m_1, \ldots, m_{23} \rangle, \quad e_k, m_k \in \{0, 1\}$$

$$V = (-1)^\sigma, \quad E = -127 + \sum_{k=0}^{7} e_k 2^k, \quad M = 1 + \sum_{k=1}^{23} m_k 2^{-k}$$

Fließkommazahl: $x = V \cdot 2^E \cdot M$

Darstellung von Zahlen im Computer

- Beispiel: $x = 1$

$x = (1)_{10} = (1.0)_2 \times 2^0 \qquad (M \in [1,2))$
$M = (1)_2 \Rightarrow m_k = 0,\ k = 1, \ldots, 23$
$0 = E = -(127)_{10} + e \Rightarrow e = (127)_{10}$

Wie wird $(127)_{10}$ binär dargestellt?

Algorithmus für die dezimalen Ziffer von 127:
$\frac{127}{10} = 12 + \frac{7}{10}, \quad \frac{12}{10} = 1 + \frac{2}{10}, \quad \frac{1}{10} = 0 + \frac{1}{10}$
Algorithmus für die binären Ziffer von 127:
$\frac{127}{2} = 63 + \frac{1}{2}, \quad \frac{63}{2} = 31 + \frac{1}{2}, \quad \frac{31}{2} = 15 + \frac{1}{2},$
$\frac{15}{2} = 7 + \frac{1}{2}, \quad \frac{7}{2} = 3 + \frac{1}{2}, \quad \frac{3}{2} = 1 + \frac{1}{2}, \quad \frac{1}{2} = 0 + \frac{1}{2}$

Also $e = (127)_{10} = (1111111)_2$

x (IEEE): 00111111 10000000 00000000 00000000

Darstellung von Zahlen im Computer

- Beispiel: $x = 0.375$

  Algorithmus für die dezimalen Ziffer von 0.375:
  $0.375 \times 10 = 3 + 0.75,$
  $0.750 \times 10 = 7 + 0.50, \quad 0.50 \times 10 = 5 + 0$
  Algorithmus für die binären Ziffer von 0.375:
  $0.375 \times 2 = 0 + 0.75,$
  $0.750 \times 2 = 1 + 0.50, \quad 0.50 \times 2 = 1 + 0$
  Also
  $x = (0.375)_{10} = (0.011)_2 = (1.1)_2 \times 2^{-2} \qquad (M \in [1, 2))$
  $M = (1.1)_2 \Rightarrow m_1 = 1, m_k = 0, k = 2, \ldots, 23$
  $e - (127)_{10} = E = -(2)_{10} \Rightarrow e = (125)_{10} = (01111101)_2$

  x (IEEE): 00111110 11000000 00000000 00000000

Darstellung von Zahlen im Computer

- Beispiel: $x = 12.375$.

  $x = (12)_{10} + (0.375)_{10}$
  $x = (1100)_2 + (0.011)_2 = (1100.011)_2$

  $x = (1.100011)_2 \times 2^3, \qquad (M \in [1, 2))$
  $M = (1.100011)_2 \Rightarrow m_1 = 1, m_5 = 1, m_6 = 1$, sonst $m_k = 0$
  $e - (127)_{10} = E = (3)_{10} \Rightarrow e = (130)_{10} = (10000010)_2$

  x (IEEE): 01000001 01000110 00000000 00000000

- Sonderfälle:
  - Für $x = 0$ sind alle E-Bits 0 und alle M-Bits 0.
  - Für $|x| = \infty$ sind alle E-Bits 1 und alle M-Bits 0.
  - Für $x =$ NaN sind alle E-Bits 1 und nicht alle M-Bits 0.

Darstellung von Zahlen im Computer

- Beispiel: $x = -0.1$

  Algorithmus für die binären Ziffer von 0.1:
  $0.1 \times 2 = 0 + 0.2$, $0.2 \times 2 = 0 + 0.4$,
  $0.4 \times 2 = 0 + 0.8$, $0.8 \times 2 = 1 + 0.6$,
  $0.6 \times 2 = 1 + 0.2 \rightarrow$ zurück zum Anfang

  Also $x = -(0.1)_{10} = -(0.0\overline{0011})_2 = -(1.1\overline{0011})_2 \times 2^{-4}$

  Probe: $x = -\sum_{k=1}^{\infty} 2^{-4k} - \frac{1}{2}\sum_{k=1}^{\infty} 2^{-4k} = -\frac{1}{10}$

  $M = (1.1\overline{0011})_2 \Rightarrow m_1 = 1, \{m_{4k} = m_{4k+1} = 1\}_{k=1}^{5}$,
  sonst $m_k = 0$?
  $e - (127)_{10} = E = -(4)_{10} \Rightarrow e = (123)_{10} = (1111011)_2$

  Mit Aufrundung ($m_{23} = 1$)
  x (IEEE): 10111101 11001100 11001100 11001101
  Mit Abrundung ($m_{23} = 0$)
  x (IEEE): 10111101 11001100 11001100 11001100

Darstellung von Zahlen im Computer

- Fortsetzung des Beispiels: $x = -0.1$

  Mit Aufrundung ($m_{23} = 1$)
  x (IEEE): 10111101 11001100 11001100 11001101
  $\rightarrow (0.100000001490116119384765625)_{10}$
  Mit Abrundung ($m_{23} = 0$)
  x (IEEE): 10111101 11001100 11001100 11001100
  $\rightarrow (0.099999994039535522460937 5)_{10}$
  Mit einfacher Genauigkeit gibt es keine speicherbaren Zahlen zwischen diesen!

- IEEE Fließkommazahl *doppelter Genauigkeit* (64 bits, 8 bytes)

$$x = V \cdot 2^E \cdot M, \quad E = -1023 + \sum_{k=0}^{10} e_k 2^k$$

$$V = (-1)^\sigma, \quad M = 1 + \sum_{k=1}^{52} m_k 2^{-k}$$

Darstellung von Zahlen im Computer

Fehler in der Darstellung?

Def: Wenn $\tilde{p}$ eine Approximation zu p ist, ist der absolute Fehler $|p - \tilde{p}|$. Der relative Fehler ist $|p - \tilde{p}|/|p|$ wenn $p \neq 0$.

Def: $\tilde{p}$ approximiert p zu s *signifikanten Dezimalziffern* wenn $s \in \mathbb{N}$ die größte Zahl ist, bei der der relative Fehler kleiner als 5×10^{-s} ist.

- Beispiel: $x = -0.1$, $\tilde{x} = -0.10000000149\cdots$ (Aufrundung)

$$5 \times 10^{-9} < \frac{|x - \tilde{x}|}{|x|} = \frac{1.49\cdots \times 10^{-9}}{10^{-1}} < 5 \times 10^{-8} \Rightarrow s = 8$$

Darstellung von Zahlen im Computer

- Relativer Fehler der Fließkommazahl-Darstellung $\mathrm{fl}(x)$, mit b-binärziffriger Genauigkeit und Abrundung,

$$\frac{|x - \mathrm{fl}(x)|}{|x|} = \frac{|V \cdot 2^E \sum_{k=0}^{\infty} m_k 2^{-k} - V \cdot 2^E \sum_{k=0}^{b-1} m_k 2^{-k}|}{|V \cdot 2^E \sum_{k=0}^{\infty} m_k 2^{-k}|}$$

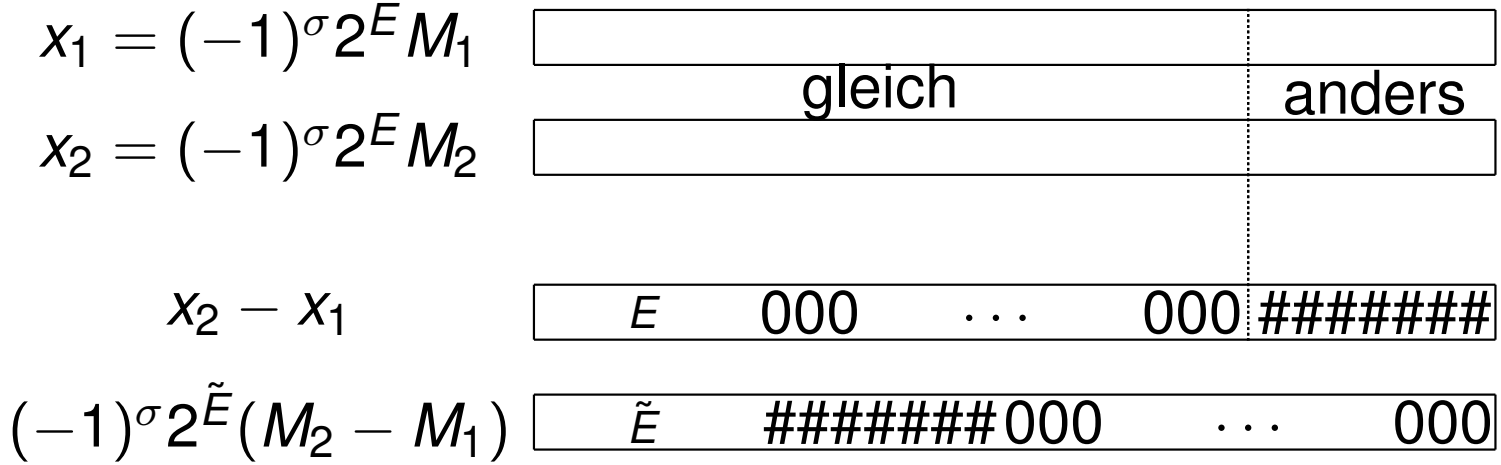
$$= \left|\sum_{k=b}^{\infty} m_k 2^{-k}\right| \Big/ \left|\sum_{k=0}^{\infty} m_k 2^{-k}\right|_{\geq 1} \leq \sum_{k=b}^{\infty} m_k 2^{-k}$$

$$\leq \sum_{k=b}^{\infty} 2^{-k} = 2^{-b} \sum_{k=0}^{\infty} 2^{-k} = \frac{2^{-b}}{1 - 2^{-1}} = 2^{-b+1} \quad (\overset{!}{<} \mathrm{tol})$$

- IEEE einfache Genauigkeit, $b = 24$,
  $2^{-b+1} \approx 1.19 \times 10^{-7}$, $s = 7$
- IEEE doppelte Genauigkeit, $b = 53$,
  $2^{-b+1} \approx 2.22 \times 10^{-16}$, $s = 16$

Darstellung von Zahlen im Computer

- Verlust von signifikanten Ziffern: *Auslöschung*.
  Bildlich gesehen:



  Nur die Ziffer ######## überleben Subtraktion. In exakter Arithmetik wäre der letzte Bereich $000\cdots000$ mit richtigen Ziffern besetzt.

Darstellung von Zahlen im Computer

- d-dezimalziffrige Genauigkeit mit Aufrundung der Zahl $x = (-1)^\sigma 10^E \sum_{k=0}^{\infty} \underset{m_k \in \{0,\ldots,9\}}{m_k} 10^{-k}$ ist mit Matlab `round(x,d,'s')` =

$$(-1)^\sigma 10^E \left(\sum_{k=0}^{d-2} m_k 10^{-k} + \tilde{m}_{d-1} 10^{-(d-1)} \right), \quad \tilde{m}_{d-1} = \begin{cases} m_{d-1}, & m_d < 5 \\ m_{d-1}+1, & m_d \geq 5. \end{cases}$$

- Beispiel: Berechne die Nullstellen von

$$p(x) = x^2 + 62.10x + 1$$

mit 4-dezimalziffriger Genauigkeit und Aufrundung.

$$\frac{-62.10 + [(62.10)^2 - 4.000 \cdot 1.000 \cdot 1.000]^{\frac{1}{2}}}{2.000}$$

$$\rightarrow \frac{-62.10 + [3856 - 4.000]^{\frac{1}{2}}}{2.000} \rightarrow \frac{-0.04000}{2.000} = -0.02000$$

$$\neq x_1 = -0.01610723$$

- Besserer Weg: $A - B = (A^2 - B^2)/(A + B) \Rightarrow$

$$\frac{-b + \sqrt{b^2 - 4ac}}{2a} = \frac{-b^2 + (b^2 - 4ac)}{2a[b + \sqrt{b^2 - 4ac}]} \rightarrow -0.01610$$

Darstellung von Zahlen im Computer

- Anfällig für Auslöschung: Polynomauswertung und Skalarprodukt
- Beispiel: Auswertung des Polynoms
  $$p(x) = x^3 - 6x^2 + 3x - 0.149, \quad x = 4.71$$
  mit 3-dezimalziffriger Genauigkeit und Aufrundung.
  exakt:
  $104.487111 - 133.1046 + 14.13 - 0.149 = -14.636489$
  3-ziffrig:
  $105 - 133 + 14.1 - 0.149 \to -14.0$, 4% relativer Fehler
- Horner Algorithmus:

  $$p(x) = x[x^2 + 6x + 3] - 0.149 = x[x(x - 6) + 3] - 0.149$$

  $$p \leftarrow x - 6, p \leftarrow x \cdot p, p \leftarrow p + 3, p \leftarrow x \cdot p, p \leftarrow p - 0.149$$

  Ergebnis: -14.6, 0.25% relativer Fehler
- Skalarprodukt: 2-dezimalziffrige Genauigkeit und Aufrundung,

  $$(20, -20, 0.1) \cdot (1, 1, 1) \to 0.1 \neq 0 \leftarrow (20, 0.1, -20) \cdot (1, 1, 1)$$

Lineare Gleichungssysteme: Direkte Methoden

Gaußsche Elimination zur Lösung von $A\boldsymbol{x} = \boldsymbol{b}$. Beispiel:

- Das System:

$$\begin{array}{rcrcrcrl} 3x_1 & + & 6x_2 & & 9x_3 & = & -12 & G_1 \\ 2x_1 & + & 5x_2 & - & 8x_3 & = & -11 & G_2 \\ x_1 & - & 4x_2 & - & 7x_3 & = & -10 & G_3 \end{array}$$

- $G_2 \leftarrow G_2 - \frac{2}{3}G_1$, $G_3 \leftarrow G_3 - \frac{1}{3}G_1$,

$$\begin{array}{rcrcrcr} 3x_1 & + & 6x_2 & + & 9x_3 & = & -12 \\ & & x_2 & - & 14x_3 & = & -3 \\ & - & 6x_2 & - & 10x_3 & = & -6 \end{array}$$

- $G_3 \leftarrow G_3 - \frac{-6}{+1}G_2$,

$$\begin{array}{rcrcrcr} 3x_1 & + & 6x_2 & + & 9x_3 & = & -12 \\ & & x_2 & - & 14x_3 & = & -3 \\ & & & - & 94x_3 & = & -24 \end{array}$$

Gaußsche Elimination und Rückwärts Substitution

- Rückwärts Substitution:

$$\begin{array}{rcrcrcl} 3x_1 & + & 6x_2 & + & 9x_3 & = & -12 \\ & & x_2 & - & 14x_3 & = & -3 \\ & & & - & 94x_3 & = & -24 \end{array} \quad \begin{array}{c} \Rightarrow \\ \Rightarrow \\ \Rightarrow \end{array} \quad \begin{array}{l} x_1 = \frac{1}{3}(-12 - 6x_2 - 9x_3) = -\frac{278}{47} \\ x_2 = -3 + 14x_3 = \frac{27}{47} \uparrow \\ x_3 = \frac{12}{47} \uparrow \end{array}$$

- Zur Lösung von $A\boldsymbol{x} = \boldsymbol{b}$ können die Schritte auf der erweiterten Matrix $[A|\boldsymbol{b}]$ durchgeführt werden:

$$[A|b] = \left[\begin{array}{ccc|c} 1 & 1 & 0 & 4 \\ 2 & 1 & -1 & 1 \\ 3 & -1 & -1 & -13 \end{array}\right] \in \mathbb{R}^{3\times 4} \qquad (\mathbb{R}^{n\times(n+1)})$$

- Wenn es mehrere rechte Seiten gibt,

$$A\boldsymbol{x}^i = \boldsymbol{b}^i, \quad i = 1, 2, \ldots, m \quad \text{oder}$$
$$AX = B, \quad X = [\boldsymbol{x}^1, \ldots, \boldsymbol{x}^m], \quad B = [\boldsymbol{b}^1, \ldots, \boldsymbol{b}^m]$$

können die Schritte gleichzeitig auf der erweiterten Matrix $[A|B]$ durchgeführt werden, wenn B von X unabhängig ist.

Diskretisierung einer PDG

- Die Diskretisierung $[\frac{\mu}{h^2} D + I]\boldsymbol{u} = \boldsymbol{v}$ des Randwertproblems,

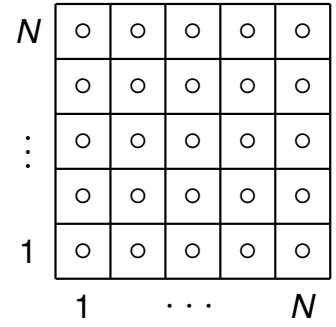
$$\begin{cases} -\mu u'' + u &= v, & \Omega \\ u' &= 0, & \partial\Omega \end{cases}$$

  wird sehr groß mit Verfeinerung, aber sie ist trotzdem immer tridiagonal. Gaußsch Elimination für dieses System ist daher ziemlich *billig*.

- Bessere Motivation: Randwertproblem auf $\Omega = (0,1)^2$,

$$\begin{cases} -\mu(u_{xx} + u_{yy}) + u &= v, & \Omega \\ u_n &= 0, & \partial\Omega \end{cases}$$

$$u_n = (-1)^{y+1} u_x, \quad y = 0, 1$$
$$u_n = (-1)^{x+1} u_y, \quad x = 0, 1$$



$$u_{xx} + u_{yy} \approx \frac{u_{i+1,j} - 2u_{ij} + u_{i-1,j}}{h^2} + \frac{u_{i,j+1} - 2u_{ij} + u_{i,j-1}}{h^2}$$

Diskretisierung einer PDG

- Gleichungssystem:

$$\begin{bmatrix} \ddots & & \ddots & \ddots & \ddots & & \ddots & 0 \\ & -\frac{\mu}{h^2} & & -\frac{\mu}{h^2} & \frac{4\mu}{h^2}+1 & -\frac{\mu}{h^2} & & -\frac{\mu}{h^2} \\ 0 & & \ddots & & \ddots & \ddots & \ddots & \ddots \end{bmatrix} \begin{bmatrix} u_{1,1} \\ \vdots \\ u_{N,1} \\ \vdots \\ u_{1,N} \\ \vdots \\ u_{N,N} \end{bmatrix} = \begin{bmatrix} v_{1,1} \\ \vdots \\ v_{N,1} \\ \vdots \\ v_{1,N} \\ \vdots \\ v_{N,N} \end{bmatrix}$$

$\mathcal{O}(N)$ $\mathcal{O}(N)$

- Obwohl die Matrix dünn besetzt ist (*sparse*, d.h. ganz wenig nicht triviale Elemente), führt Gaußsche Elimination zur Auffüllung (*fill-in*) der Zwischendiagonalen.
- Beispiel:

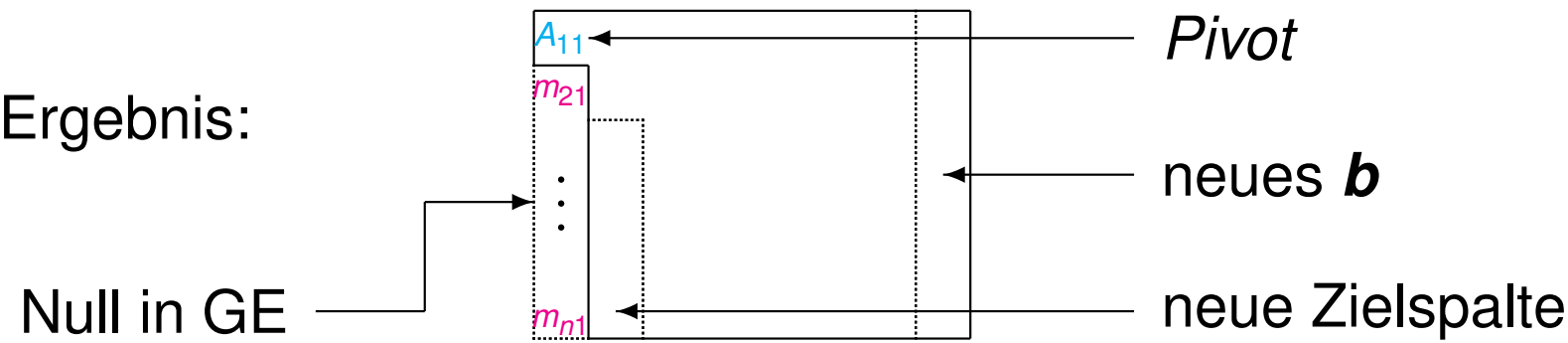
```
A = spdiags(ones(N,3),[-p,0,q],N,N) + speye(N);
[L,U] = lu(A);
% GE-Dreiecksform ist U, 0 bis q aufgefuellt
```

Automatisierung der Gaußschen Elimination

- Um die Lösungsschritte der Gaußschen Elimination zu automatisieren, werden diese symbolisch geschrieben:

$$
\begin{aligned}
A &\leftarrow [A|\boldsymbol{b}] \\
\text{2. Zeile:} \quad & \begin{cases} A_{21} = A_{21}/A_{11} & \text{(Multiplikator: } m_{21}\text{)} \\ A_{2j} = A_{2j} - A_{21} \cdot A_{1j}, \quad j = 2,\ldots,n(+1) \end{cases} \\
\text{3. Zeile:} \quad & \begin{cases} A_{31} = A_{31}/A_{11} & \text{(Multiplikator: } m_{31}\text{)} \\ A_{3j} = A_{3j} - A_{31} \cdot A_{1j}, \quad j = 2,\ldots,n(+1) \end{cases} \\
& \vdots \\
i\text{. Zeile:} \quad & \begin{cases} A_{i1} = A_{i1}/A_{11} & \text{(Multiplikator: } m_{i1}\text{)} \\ A_{ij} = A_{ij} - A_{i1} \cdot A_{1j}, \quad j = 2,\ldots,n(+1) \end{cases} \\
& i = 2,\ldots,n
\end{aligned}
$$

- Ergebnis:



Automatisierung der Gaußschen Elimination

- Mit der nächsten Zielspalte:

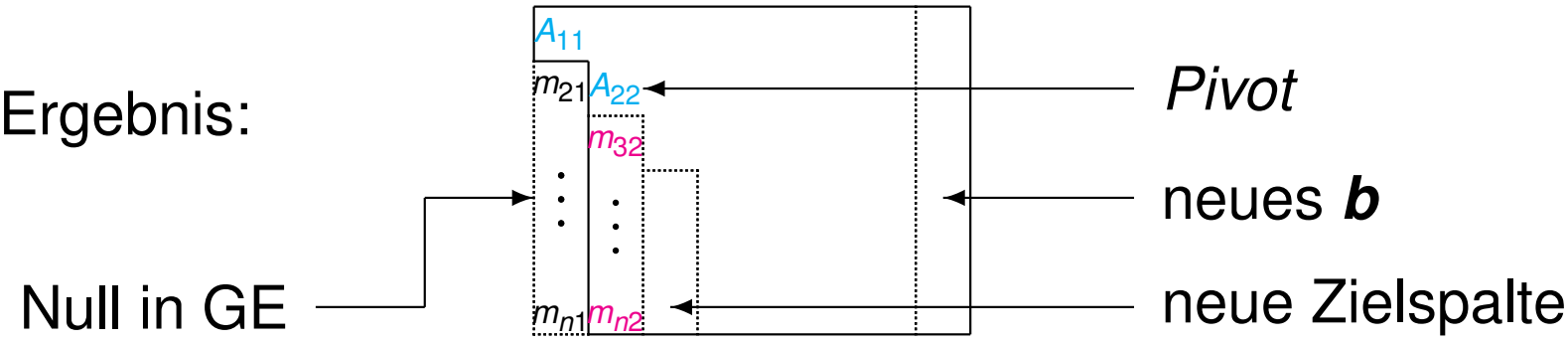
$$\text{3. Zeile:} \quad \begin{cases} A_{32} = A_{32}/A_{22} & \text{(Multiplikator: } m_{32}\text{)} \\ A_{3j} = A_{3j} - A_{32} \cdot A_{2j}, & j = 3, \ldots, n(+1) \end{cases}$$

$$\vdots$$

$$i\text{. Zeile:} \quad \begin{cases} A_{i2} = A_{i2}/A_{22} & \text{(Multiplikator: } m_{i2}\text{)} \\ A_{ij} = A_{ij} - A_{i2} \cdot A_{2j}, & j = 3, \ldots, n(+1) \end{cases}$$

$i = 3, \ldots, n$

- Ergebnis:



- Wird mit Zielspalten fortgesetzt, bis die Multiplikatoren eine Dreiecksmatrix bilden.

Automatisierung der Gaußschen Elimination

- Pseudo-Code zur Gaußschen Elimination:

```
A <- [A,b]                                  % Erweiterung
for k=1,...,n-1                             % Pivot-Index
   for i=k+1,...,n                          % Zeilen-Index
      A(i,k) = A(i,k) / A(k,k)              % Multiplikator
      for j=k+1,...,n+1                     % Spalten-Index
         A(i,j) = A(i,j) - A(i,k)*A(k,j)
      end       % (vektorisiert mit impliziten Schleifen!)
   end
end
```

- Pseudo-Code zur Rückwärts Substitution:

```
x(n) = A(n,n+1) / A(n,n)
for i=n-1,...,1
   s=0
   for j=i+1,...,n
      s = s + A(i,j)*x(j)
   end       % (vektorisiert mit impliziten Schleifen!)
   x(i) = (A(i,n+1) - s) / A(i,i)
end
```

`flops` der Gaußschen Elimination

- Wie viele Operationen kostet Gaußsche Elimination?
  - Divisionen

$$\left.\begin{array}{l}\texttt{k} = 1, \ldots, \texttt{n}-1 \\ \quad \texttt{i} = \texttt{k}+1, \ldots, \texttt{n} \quad \rightarrow (n-k)\end{array}\right\} \sum_{k=1}^{n-1}(n-k)$$

  - Multiplikationen

$$\left.\begin{array}{l}\texttt{k} = 1, \ldots, \texttt{n}-1 \\ \quad \texttt{i} = \texttt{k}+1, \ldots, \texttt{n} \quad \rightarrow \times(n-k) \\ \quad\quad \texttt{j} = \texttt{k}+1, \ldots, \texttt{n}(+1) \quad \rightarrow (n+1-k)\end{array}\right\} \sum_{k=1}^{n-1}(n-k)(n+1-k)$$

  - Subtraktionen

$$\ldots\text{ebenso} : \sum_{k=1}^{n-1}(n-k)(n+1-k)$$

- Nützliche Summen: (vgl. $\int_0^m k^{p-1}\,dk = m^p/p$)

$$\sum_{k=1}^{m} 1 = m, \quad \sum_{k=1}^{m} k = \frac{m(m+1)}{2}, \quad \sum_{k=1}^{m} k^2 = \frac{m(m+1)(2m+1)}{6}$$

`flops` der Gaußschen Elimination

- Kosten für Gaußsche Elimination:
  - Divisionen

$$\sum_{k=1}^{n-1}(n-k) = n\sum_{k=1}^{n-1}1 - \sum_{k=1}^{n-1}k = n(n-1) - \frac{(n-1)n}{2} = \frac{n(n-1)}{2}$$

  - Multiplikationen

$$\sum_{k=1}^{n-1}\underbrace{(n-k)(n+1-k)}_{n(n+1)-(2n+1)k+k^2} = (n^2+n)\sum_{k=1}^{n-1}1 - (2n+1)\sum_{k=1}^{n-1}k + \sum_{k=1}^{n-1}k^2 =$$

$$(n^2+n)(n-1) - (2n+1)\frac{(n-1)n}{2} + \frac{(n-1)n[2(n-1)+1]}{6} = \frac{n(n^2-1)}{3}$$

  - Subtraktionen: auch $n(n^2-1)/3$
- Gesamte `flops` (*floating point operations*):
  $n(n-1)/2 + 2n(n^2-1)/3 = n(n-1)(4n+7)/6 = \mathcal{O}(n^3)$

`flops` der Gaußschen Elimination

- Kosten für Rückwärts Substitutionen:
  - Divisionen: n
  - Subtraktionen: $n-1$
  - Additionen:

$$\left.\begin{array}{l}\texttt{i} = \texttt{n}-1,\ldots,1 \\ \quad \texttt{j} = \texttt{i}+1,\ldots,\texttt{n}\end{array}\right. \quad \rightarrow (n-i) \quad \Bigg\} \sum_{i=1}^{n-1}(n-i) = \frac{n(n-1)}{2}$$

  - Multiplikationen: auch $n(n-1)/2$
- Gesamte `flops`:
  $2n-1+2n(n-1)/2 = n^2+n-1 = \mathcal{O}(n^2)$
- Botschaft: Rückwärts Substitution ist viel billiger als Gaußsche Elimination.
- Hausaufgabe: Einen Pseudo-Code für Vorwärts Substitution schreiben und die Anzahl der `flops` bestimmen.

Rekursive Lösungen: Evolutionsgleichung

- Wenn viele Systeme mit $\boldsymbol{b}_k = \boldsymbol{b}_k(\boldsymbol{x}_{k-1})$ hintereinander gelöst werden müssen,
$$A\boldsymbol{x}_k = \boldsymbol{b}_k, \quad k = 1, 2, \ldots, m$$
sind die folgenden Schritte vorteilhaft:
  - Gaußsche Elimination nur einmal mit Kosten $\mathcal{O}(n^3)$ durchführen,
  - das Ergebnis speichern $(A = LU)$ und dann
  - m-Mal Rückwärts und Vorwärts Substitutionen durchführen, zwar mit Kosten $\mathcal{O}(mn^2)$.

  Sonst sind die Kosten $\mathcal{O}(mn^3)$!
- Beispiel: Evolutionsgleichung, $t \in [0, T]$,

$$\begin{cases} \boldsymbol{u}'(t) &= B\boldsymbol{u}(t) \\ \boldsymbol{u}(0) &= \boldsymbol{u}_0 \end{cases} \quad \overset{\text{zeitliche Diskretisierung}}{\longrightarrow} \quad \frac{\boldsymbol{u}(t^k) - \boldsymbol{u}(t^{k-1})}{t^k - t^{k-1}} = B\boldsymbol{u}(t^k)$$

mit $t^k = k\tau$, $\boldsymbol{u}^k \approx \boldsymbol{u}(t^k)$, $\boldsymbol{u}^m \approx \boldsymbol{u}(T)$ und $A = [I - \tau B]$. Es kosten die Iterationen

$$A\boldsymbol{u}^k = \boldsymbol{u}^{k-1}, \quad k = 1, \ldots, m$$

$\mathcal{O}(mn^3)$, aber mit $A = LU$ ist $\mathcal{O}(n^3 + mn^2)$ möglich.

LU Zerlegung

- Wie kann die Arbeit für Gaußsche Elimination so gespeichert werden (d.h. $A = LU$), dass sie später vorteilhaft verwendet werden kann?
- Definiere $A^{(1)} = A$. Das Ergebnis nach der Elimination der ersten Zielspalte ist $A^{(2)} = M^{(1)} A^{(1)}$ wobei:

$$A^{(2)} = \begin{bmatrix} & & \\ 0 & & \\ \vdots & & \\ 0 & & \end{bmatrix} \qquad M^{(1)} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ -m_{21} & & & \\ \vdots & & I^{(n-1)} & \\ -m_{n1} & & & \end{bmatrix}$$

Probe: Sei $A = \begin{bmatrix} \boldsymbol{z}_1 \\ \vdots \\ \boldsymbol{z}_n \end{bmatrix}$, $\quad M^{(1)} A = \begin{bmatrix} \boldsymbol{z}_1 \\ \boldsymbol{z}_2 - m_{21}\boldsymbol{z}_1 \\ \vdots \end{bmatrix}$

LU Zerlegung

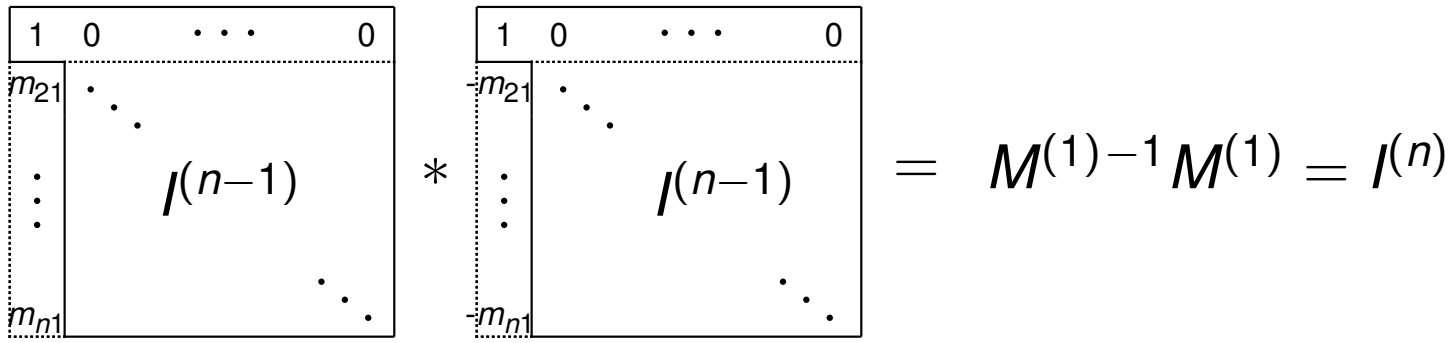
- Das Ergebnis nach der Elimination der zweiten Zielspalte ist $A^{(3)} = M^{(2)} A^{(2)}$ wobei:

$$A^{(3)} = \begin{pmatrix} & & & \\ & 0 & & \\ & & 0 & \\ & \vdots & \vdots & \\ & 0 & 0 & \end{pmatrix} \qquad M^{(2)} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & 0 \cdots & 0 \\ & -m_{32} & \ddots & \\ & \vdots & & I^{(n-2)} \\ & -m_{n2} & & \end{pmatrix}$$

- Am Ende der Gaußschen Elimination,
$$A^{(n)} = M^{(n-1)} A^{(n-1)} = M^{(n-1)} \dots M^{(1)} A$$
ist $A^{(n)}$ eine obere Dreiecksmatrix.
- Mit $U = A^{(n)}$ folgt
$$A = M^{(1)-1} \dots M^{(n-1)-1} U$$
- Die Matrizen $M^{(i)-1}$ lassen sich leicht explizit darstellen:

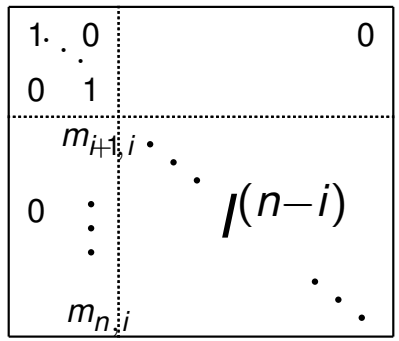
LU Zerlegung

- Erstens für $i = 1$,

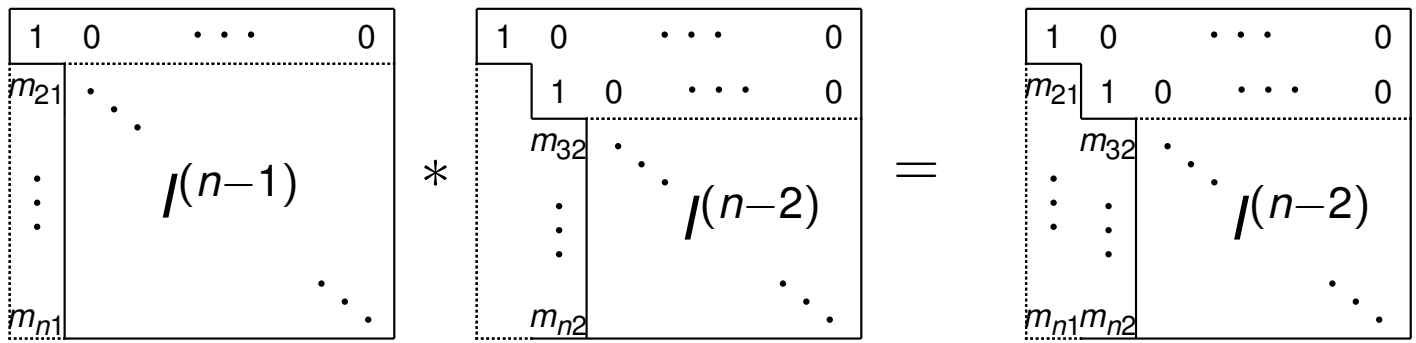


- Weiters gelten im Allgemeinen,

$$M^{(i)-1} = \quad i = 1, \ldots, n-1$$



- Für $M^{(1)-1} M^{(2)-1}$ gilt



LU Zerlegung

- Weiters gilt für die ganze Kette,

$$M^{(1)-1} \dots M^{(n-1)-1} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ m_{21} & 1 & 0 & \cdots & 0 \\ & m_{32} & 1 & & \vdots \\ \vdots & & \ddots & \ddots & \\ & \vdots & & \ddots & 0 \\ m_{n1} & m_{n2} & \cdots & m_{n,n-1} & 1 \end{pmatrix} = L$$

- So ergibt sich der folgende Satz:

Satz: Seien $\{a_{ij}^{(k)}\} = A^{(k)} = M^{(k-1)} A^{(k-1)}$, $1 \leq k \leq n$, die Matrizen, die sich durch Gaußsche Elimination der Matrix A ergeben, wobei $a_{kk}^{(k)} \neq 0$, $1 \leq k \leq n$. Dann mit

$$U = \{u_{ij}\}, \quad u_{ij} = \begin{cases} a_{ij}^{(n)}, & 1 \leq i \leq n, \quad i \leq j \leq n \\ 0, & 2 \leq i \leq n, \quad 1 \leq j < i \end{cases}$$

$$L = \{\ell_{ij}\}, \quad \ell_{ij} = \begin{cases} m_{ij}, & 2 \leq i \leq n, \quad 1 \leq j < i \\ 1, & 1 \leq i \leq n, \quad j = i \\ 0, & 1 \leq i \leq n, \quad i < j \leq n \end{cases}$$

folgt die Faktorisierung $A = LU$ in Dreiecksmatrizen.

Rekursive Lösungen mit *LU*

- Mit dieser Zerlegung werde mehrere sequentielle Systeme günstig so gelöst: $A\boldsymbol{x}_k = \boldsymbol{b}_k \Leftrightarrow L\boldsymbol{y}_k = \boldsymbol{b}_k, U\boldsymbol{x}_k = \boldsymbol{y}_k$.
- Beispiel: Evolutionsgleichung, $t \in [0, T]$,

$$\begin{cases} \boldsymbol{u}'(t) &= B\boldsymbol{u}(t) \\ \boldsymbol{u}(0) &= \boldsymbol{u}_0 \end{cases} \quad \overset{\text{zeitliche}}{\underset{\text{Diskretisierung}}{\longrightarrow}} \quad \frac{\boldsymbol{u}(t^k) - \boldsymbol{u}(t^{k-1})}{t^k - t^{k-1}} = B\boldsymbol{u}(t^k)$$

mit $t^k = k\tau$, $\boldsymbol{u}^k \approx \boldsymbol{u}(t^k)$, $\boldsymbol{u}^m \approx \boldsymbol{u}(T)$ und $A = [I - \tau B]$,

$$A\boldsymbol{u}^k = \boldsymbol{u}^{k-1}, \quad k = 1, \ldots, m$$

- Durch die *LU* Zerlegung $A = LU$,

$$\begin{array}{rlll} \text{Vorwärts Substitution:} & L\boldsymbol{v} &= \boldsymbol{u}^{k-1}, & \mathcal{O}(n^2) \\ \text{Rückwärts Substitution:} & U\boldsymbol{u}^k &= \boldsymbol{v}, & \mathcal{O}(n^2) \end{array} \qquad k = 1, \ldots, m$$

sind die Kosten $\mathcal{O}(n^3 + mn^2)$.

Pivot Strategien

Pivot Strategien:

- Beispiel:

$$\begin{bmatrix} 0.003000 & 59.14 \\ 5.291 & -6.130 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 59.17 \\ 46.78 \end{bmatrix}$$

- Exakte Lösung:

$$x_1^* = 10.00, \quad x_2^* = 1.000$$

- Mit 4-dezimalziffriger Genauigkeit und Aufrundung,

$$m_{21} = \frac{5.291}{0.003000} = 1763.6\bar{6} \rightarrow 1764$$

- Erster Schritt der Gaußschen Elimination,

$$\begin{bmatrix} 0.003000 & 59.14 & 59.17 \\ (1764) & -104300 & -104400 \end{bmatrix}$$

- Mit exakter Arithmetik wäre es gewesen,

$$\begin{bmatrix} 0.003000 & 59.14 & 59.17 \\ (1763.6\bar{6}) & -104309.376\bar{6} & -104309.376\bar{6} \end{bmatrix}$$

Pivot Strategien

- Nun Rückwärts Substitution mit 4-dezimalziffriger Genauigkeit und Aufrundung,

$$x_2 = \frac{-104400}{-104300} \to 1.001 \approx x_2^*$$

$$x_1 = \frac{59.17 - (59.14)(1.001)}{0.003000} \to -10.00 \neq 10.00 = x_1^*$$

- Hausaufgabe: Zeige, wenn die Zeilen getauscht werden,

$$\begin{bmatrix} 5.291 & -6.130 & 46.78 \\ 0.003000 & 59.14 & 59.17 \end{bmatrix}$$

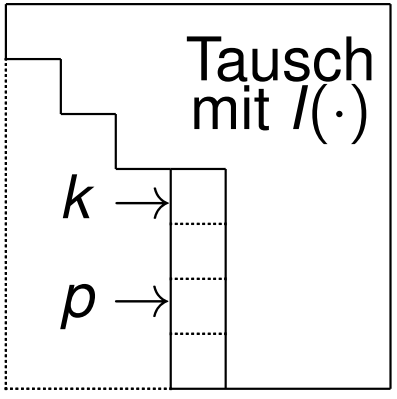
bekommt man mit 4-dezimalziffriger Genauigkeit und Aufrundung,

$$x_2 \to 1.000, \quad x_1 \to 10.00.$$

Pivot Strategien

- Pseudo-Code zur GE mit einfacher Pivot-Suche:

```
A <- [A,b]                                          % Erweiterung
I(i)=i, i=1,...,n                                   % Zeiger Initialisierung
for k=1,...,n-1                                     % Pivot-Index
    p=argmax_{k<=i<=n} |A(I(i),k)|                  % Größtes Pivot-Element
    if A(I(p),k) = 0 then stop                      % Test für Regularität
    It = I(k), I(k) = I(p), I(p) = It               % Virtueller Zeilentausch
    for i=k+1,...,n                                 % Zeilen-Index
        A(I(i),k) = A(I(i),k) / A(I(k),k)           % Multiplikator
        for j=k+1,...,n+1                           % Spalten-Index
            A(I(i),j) = A(I(i),j) - A(I(i),k)*A(I(k),j)
        end
    end
end
```



- Pseudo-Code zur Rückwärts Substitution:

```
x(n) = A(I(n),n+1) / A(I(n),n)
for i=n-1,...,1
    s=0
    for j=i+1,...,n
        s = s + A(I(i),j)*x(j)
    end
    x(i) = (A(I(i),n+1) - s) / A(I(i),i)
end
```

LU Zerlegung mit Permutation

- Wie kann die Matrix mit Pivot-Suche faktorisiert werden?

Def: Eine Permutationsmatrix P hat genau einen nicht trivialen Eintrag, zwar mit dem Wert 1, in jeder Spalte und in jeder Zeile, und es gilt $PP^\top = I$.

- Beispiel:

$$P = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad A = \begin{bmatrix} 0 & 1 \\ 3 & 2 \end{bmatrix} \quad PA = \begin{bmatrix} 3 & 2 \\ 0 & 1 \end{bmatrix}$$

Bemerkung: Sei $\{l(i) : i = 1, \ldots, n\}$ eine gegebene Permutation von den Zahlen $\{1, \ldots, n\}$. Die Permutationsmatrix

$$P_{i,j} = \delta_{l(i),j} \quad \text{erfüllt} \quad \tilde{A} = PA \quad \text{wobei} \quad \tilde{A}_{ij} = A_{l(i),j}.$$

- Mit dem obigen Satz ergibt sich $PA = \tilde{A} = LU$ vom Algorithmus mit Pivot-Suche, wobei

$$P_{i,j} = \delta_{l(i),j} \quad L_{ij} = \begin{cases} A^{(n)}_{l(i),j}, & i > j \\ 1, & i = j \\ 0, & i < j \end{cases} \quad U_{ij} = \begin{cases} A^{(n)}_{l(i),j}, & i \leq j \\ 0, & i > j \end{cases}$$

Skalierte Pivotsuche

Weitere Pivotsuche-Strategien

- Das 2×2 Beispiel lässt sich so umschreiben:

$$\begin{bmatrix} 30.00 & 591400 \\ 5.291 & -6.130 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 591700 \\ 46.78 \end{bmatrix} \qquad (\leftarrow \times 10^4)$$

- Erster Schritt der Gaußschen Elimination (kein Tausch!)

$$m_{21} = \frac{5.291}{30.00} \to 0.1764 \qquad \begin{bmatrix} 30.00 & 591400 & 591700 \\ (0.1764) & -104300 & -104400 \end{bmatrix}$$

führt zur gleichen Lösung wie vorher ohne Pivotsuche:

$$x_2 \to 1.001, \quad x_1 \to -10.00 \neq 10.00 = x_1^*$$

- Neue Strategie: Die Zeilen sollen skaliert werden.

Skalierte Pivotsuche

- Pseudo-Code zur GE mit skalierter Pivotsuche:

```
A <- [A,b]                                          % Erweiterung
s(i)=max_j |A(i,j)|, i=1,...,n                      % Zeilenskalen
I(i)=i, i=1,...,n                                   % Zeiger Initialisierung
for k=1,...,n-1                                     % Pivot-Index
    p=argmax_{k<=i<=n} |A(I(i),k)|/s(I(i))          % Größtes Pivot-Element
    if A(I(p),k) = 0 then stop                      % Test für Regularität
    It = I(k), I(k) = I(p), I(p) = It               % Virtueller Zeilentausch
    for i=k+1,...,n                                 % Zeilen-Index
        A(I(i),k) = A(I(i),k) / A(I(k),k)           % Multiplikator
        for j=k+1,...,n+1                           % Spalten-Index
            A(I(i),j) = A(I(i),j) - A(I(i),k)*A(I(k),j)
        end
    end
end
```

- Pseudo-Code zur Rückwärts Substitution:

```
x(n) = A(I(n),n+1) / A(I(n),n)
for i=n-1,...,1
    s=0
    for j=i+1,...,n
        s = s + A(I(i),j)*x(j)
    end
    x(i) = (A(I(i),n+1) - s) / A(I(i),i)
end
```

Totale Pivotsuche

Bemerkung: Kosten für Tauschen und Skalieren sind $\mathcal{O}(n^2)$

Weitere Pivotsuche-Strategien: Totale Pivotsuche

- Virtueller Zeilentausch mit einem Zeiger $I(\cdot)$.
- Virtueller Spaltentausch mit einem Zeiger $J(\cdot)$.
- Bestimmung des Pivot-Elements:

```
(p,q)=argmax_(k≤i,j≤n) |A(I(i),J(j))|
```

- Zusätzliche Arbeit ist $\mathcal{O}(n^3)$.
- Geeignet wenn das Problem sehr verschiedene Skalen hat, d.h. wenn die Matrix A sehr *steif* ist.
- Verwendung:

$$\begin{array}{rrcrcrcl} \text{Zeilentausch:} & A\boldsymbol{x} = \boldsymbol{b} & \rightarrow & P_1 A\boldsymbol{x} & = & LU\boldsymbol{x} & = & P_1\boldsymbol{b} \\ \text{Spaltentausch:} & A\boldsymbol{x} = \boldsymbol{b} & \rightarrow & AP_2P_2^\top\boldsymbol{x} & = & LUP_2^\top\boldsymbol{x} & = & \boldsymbol{b} \end{array}$$

Beides: $(P_1AP_2)P_2^\top\boldsymbol{x} = LUP_2^\top\boldsymbol{x} = P_1\boldsymbol{b}$

$$L\boldsymbol{y} = P_1\boldsymbol{b}, \quad U\boldsymbol{z} = \boldsymbol{y}, \quad \boldsymbol{x} = P_2\boldsymbol{z}$$

Determinante und Inverse

Bemerkung: Falls keine Pivotsuche notwendig ist, folgt aus $A^{(n)} = M^{(n-1)} \dots M^{(1)} A$ und $\det(M^{(i)}) = 1$, $i = 1, \dots, n-1$, dass $\det(A) = \det(A^{(n)})$ gilt, d.h.

$$\det(A) = \det \begin{pmatrix} a_{11} & \cdots & \cdots & a_{1n} \\ 0 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & \cdots & a_{nn}^{(n)} \end{pmatrix} = a_{11}^{(1)} \cdot a_{22}^{(2)} \cdots a_{nn}^{(n)}$$

und daher kann die Determinante durch Gaußsche Elimination mit Kosten $\mathcal{O}(n^3)$ berechnet werden. Die direkte Berechnung ist viel teurer, zwar $\mathcal{O}(n!)$ durch Induktion.

Bemerkung: Falls die Inverse notwendig ist, kann sie günstig durch die Lösung des folgenden Systems berechnet werden:

$$AX = I \quad \Rightarrow \quad X = A^{-1} \quad \mathcal{O}(n^4)$$

Die direkte Berechnung mit Determinanten ist viel teurer $\mathcal{O}(n!)$.

Diagonal Dominante Matrizen

Wann wird keine Pivotsuche notwendig?

Def: Eine Matrix $\{a_{ij}\} = A \in \mathbb{R}^{n\times n}$ ist *streng diagonal dominant* wenn

$$|a_{ii}| > \sum_{i\neq j=1}^{n} |a_{ij}|, \quad i = 1, \ldots, n$$

Bei Gleichheit ist sie *schwach diagonal dominant*.

- Beispiel: Für das motivierende Beispiel,

$$\begin{cases} -\mu u'' + u &= v, & \Omega \\ u' &= 0, & \partial\Omega \end{cases} \longrightarrow \quad A\boldsymbol{u} = \left[\frac{\mu}{h^2} D + I\right] \boldsymbol{u} = \boldsymbol{v}$$

$N \in \mathbb{N}$, $h = 1/N$, $\mathbf{1} \in \mathbb{R}^N$, $I = \mathrm{diag}(\mathbf{1})$, $\mu > 0$,

$$A = \mu D/h^2 + I, \quad D = \begin{bmatrix} 1 & -1 & & & 0 \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ 0 & & & -1 & 1 \end{bmatrix} \in \mathbb{R}^{N\times N}$$

SPD Matrizen

ist A streng diagonal dominant:

$$\begin{array}{lllll} (x_1, x_N) & i = 0, N: & \frac{\mu}{h^2} + 1 & > & |-\frac{\mu}{h^2}| \quad \surd \\ (x_i) & 1 < i < N: & \frac{2\mu}{h^2} + 1 & > & |-\frac{\mu}{h^2}| + |-\frac{\mu}{h^2}| \quad \surd \end{array}$$

Satz: Für eine streng diagonal dominante Matrix, kann Gaußsche Elimination ohne Pivotsuche stabil durchgeführt werden.

Def: Eine symmetrische Matrix $A \in \mathbb{R}^{n \times n}$ ist positiv definit (SPD) wenn $\boldsymbol{x}^\top A \boldsymbol{x} > 0$, $\forall \boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{x} \neq 0$.

Bemerkung: Wenn $A \in \mathbb{R}^{n \times n}$ symmetrisch ist, sind die Eigenwerte reel und es gilt

$$\min\{\lambda \in \sigma(A)\} \leq \frac{\boldsymbol{x}^\top A \boldsymbol{x}}{\boldsymbol{x}^\top \boldsymbol{x}} \leq \max\{\lambda \in \sigma(A)\}$$

Wenn $\min\{\lambda \in \sigma(A)\} > 0$ gezeigt werden kann, ist A SPD.

Gerschgorin Satz

- Dies lässt sich für das obige Beispiel mit dem folgenden Satz zeigen.

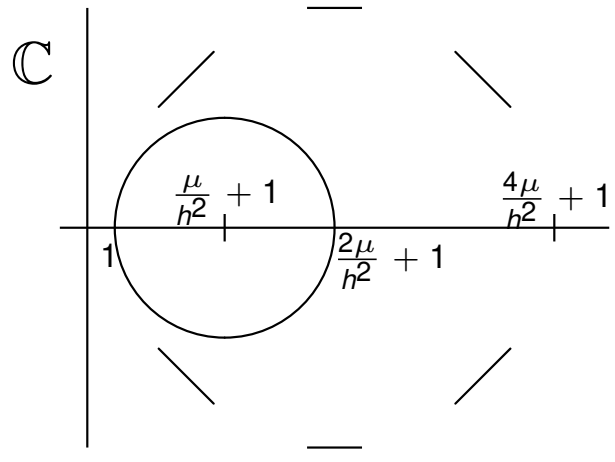
Satz (Gerschgorin): Für $\{a_{ij}\} = A \in \mathbb{R}^{n\times n}$ gilt

$$\sigma(A) \subset \bigcup_{i=1}^{n} R_i \quad \text{wobei} \quad R_i = \left\{ z \in \mathbb{C} : |z - a_{ii}| \leq \sum_{i\neq j=1}^{n} |a_{ij}| \right\}$$

Weiters liegen genau k Eigenwerte in $\cup_{j=1}^{k} R_{i_j}$ wenn diese Menge einen leeren Schnitt mit den anderen Scheiben hat.

- Beispiel: Wo liegen die Eigenwerte für das Beispiel?

$$R_1 = B(\tfrac{\mu}{h^2} + 1, \tfrac{\mu}{h^2})$$
$$R_i = B(\tfrac{2\mu}{h^2} + 1, \tfrac{2\mu}{h^2})$$
$$R_N = B(\tfrac{\mu}{h^2} + 1, \tfrac{\mu}{h^2})$$



Notwendigkeit einer Pivotsuche

- Für das Beispiel ist A symmetrisch und es gilt

$$1 \leq \min\{\lambda \in \sigma(A)\} \leq \frac{\boldsymbol{x}^\top A \boldsymbol{x}}{\boldsymbol{x}^\top \boldsymbol{x}}$$

  also ist A SPD.

Satz: Für eine SPD Matrix kann Gaußsche Elimination ohne Pivotsuche stabil durchgeführt werden.

Botschaft: Für viele Matrizen in Anwendungen ist eine Pivotsuche nicht notwendig, aber es ist sicher nicht immer so.

Folgesatz: Eine Matrix $A \in \mathbb{R}^{n\times n}$ ist SPD genau dann wenn $\exists L \in \mathbb{R}^{n\times n}$ eine untere Dreiecksmatrix mit $A = LL^\top$.

- Das System $LL^\top = A = \{a_{ij}\}$, $L = \{\ell_{ij}\}$, komponentweise

$$a_{ij} = \sum_{k=1}^{\min(i,j)} \ell_{ik}\ell_{jk} \qquad \begin{aligned} a_{11} &= \ell_{11}^2 \\ a_{21} &= \ell_{21}\ell_{11} \qquad \dots \\ a_{12} &= \ell_{11}\ell_{21} \end{aligned}$$

  führt zum folgenden Algorithmus:

Pseudo-Code für den Cholesky Algorithmus

```
L(1,1) = √A(1,1)
for i=2,...,n
    L(i,1) = A(i,1) / L(1,1)
end
for j=2,...,n-1
    s = 0
    for k=1,...,j-1
        s = s + L(j,k) * L(j,k)
    end
    L(j,j) = √A(j,j) − s
    for i=j+1,...,n
        s = 0
        for k=1,...,j-1
            s = s + L(i,k) * L(j,k)
        end
        L(i,j) = [A(i,j) − s]/L(j,j)
    end
end
s = 0
for k=1,...,n-1
    s = s + L(n,k) * L(n,k)
end
L(n,n) = √A(n,n) − s
```

- Hausaufgabe: Leite diese Methode her.
- Bemerke: Der Cholesky Algorithmus ist nur für SPD Matrizen.
- Hausaufgabe: Schätze die `flops` des Codes ab.

Bandmatrizen

- Die Bandbreite einer solchen Matrix,

$$A = \begin{bmatrix} a_{1,1} & \cdots & a_{1,q+1} & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ a_{p+1,1} & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & a_{n-q,n} \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & a_{n,n-p} & \cdots & a_{n,n} \end{bmatrix} \in \mathbb{R}^{n \times n}$$

  ist $p + q + 1$, die kleinste Anzahl der benachbarten Diagonalen, innerhalb welcher die nicht trivialen Elemente von A zu finden sind.

- Bemerke:

$$p \leq n - 1, \quad q \leq n - 1, \quad \text{Bandbreite} \leq 2n - 1.$$

  Wenn $p + q + 1 \ll 2n - 1$ ist A eine *Bandmatrix*.

Bandmatrizen

- Um Speicherplatz zu reduzieren, wird A als Liste der Diagonalen gespeichert:

$$B = \begin{bmatrix} 0 & \cdots & 0 & a_{1,1} & \cdots & \cdots & a_{1,q+1} \\ \vdots & \ddots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & \ddots & \vdots & \vdots & \vdots & \vdots & a_{n-q,n} \\ a_{p+1,1} & \vdots & \vdots & \vdots & \vdots & \ddots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ a_{n,n-p} & \cdots & \cdots & a_{n,n} & 0 & \cdots & 0 \end{bmatrix} \in \mathbb{R}^{n\times(p+q+1)}$$

- Bemerke: Die nicht trivialen Elemente von $A = \{a_{i,j}\}$ lassen sich bezüglich der Elemente von $B = \{b_{\alpha,\beta}\}$ so darstellen:

$$a_{i,j} = b_{i,j-i+p+1}$$

Sparse Speicherformat

- Hausaufgabe: Schreibe einen Pseudo-Code zur Implementierung des Gauß Algorithmus für A bezüglich B mit möglichst wenig `flops`.
- Es gibt auch *sparse format*:

$$\texttt{A} = \texttt{sparse}(\texttt{i},\texttt{j},\texttt{w},\texttt{m},\texttt{n}) \quad \Rightarrow \quad A \in \mathbb{R}^{m\times n}, \quad A_{i(k),j(k)} = w(k)$$

- Beispiel: $N \in \mathbb{N}$, $h = 1/N$, $\mathbf{1} \in \mathbb{R}^N$, $I = \mathrm{diag}(\mathbf{1})$, $\mu > 0$,

$$A = \mu D/h^2 + I, \quad D = \begin{bmatrix} 1 & -1 & & & 0 \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ 0 & & & -1 & 1 \end{bmatrix} \in \mathbb{R}^{N\times N}$$

```
d = [-ones(N,1),2*ones(N,1),-ones(N,1)];
      d(1,2)=1;    d(N,2)=1;
D = spdiags(d,[-1,0,+1],N,N);    I = speye(N);
A = mu*D/h^2 + I;
```

Lineare Gleichungssysteme: Iterative Methoden

Iterative Verfahren zur Lösung von $A\boldsymbol{x} = \boldsymbol{b}$.

- Um $A\boldsymbol{x} = \boldsymbol{b}$ annäherungsweise zu lösen, werden Iterationen der folgenden Form formuliert,

$$\boldsymbol{x}^{(k+1)} = T\boldsymbol{x}^{(k)} + \boldsymbol{c}, \quad k = 0, 1, \ldots$$

  mit dem Fixpunkt $\boldsymbol{x} = A^{-1}\boldsymbol{b}$.

- Um Konvergenz zu untersuchen, wird ein gewisses Werkzeug benötigt.

Def: Eine Vektornorm ist eine Funktion $\|\cdot\| : \mathbb{R}^n \to \mathbb{R}$ mit den Eigenschaften:

1. $\|\boldsymbol{x}\| \geq 0, \forall \boldsymbol{x} \in \mathbb{R}^n$ ($\sum_{i=1}^n x_i$ keine Norm)
2. $\|\boldsymbol{x}\| = 0 \Leftrightarrow \boldsymbol{x} = 0,$ ($|x_1|$ keine Norm für $n > 1$)
3. $\|\alpha\boldsymbol{x}\| = |\alpha|\|\boldsymbol{x}\|, \forall \boldsymbol{x} \in \mathbb{R}^n, \forall \alpha \in \mathbb{R}$ ($\sum_{i=1}^n x_i^2$ keine Norm)
4. $\|\boldsymbol{x} + \boldsymbol{y}\| \leq \|\boldsymbol{x}\| + \|\boldsymbol{y}\|, \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$
   ($f(\sqrt{\sum_{i=1}^n x_i^2})$ keine Norm wenn f nicht konvex)

Vektornormen

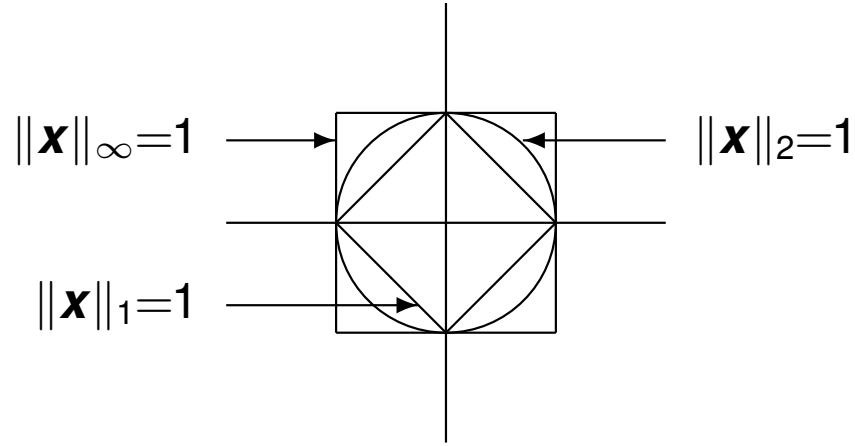
Satz: Die folgenden sind bekannte Normen:

$$\|\boldsymbol{x}\|_p = \left[\sum_{i=1}^{n} |x_i|^p\right]^{\frac{1}{p}}, 1 \leq p < \infty, \qquad \|\boldsymbol{x}\|_\infty = \max_{1\leq i\leq n} |x_i|$$

Satz: $\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ gilt

$$|\boldsymbol{x} \cdot \boldsymbol{y}| \leq \sum_{i=1}^{n} |x_i y_i| \leq \|\boldsymbol{x}\|_p \|\boldsymbol{y}\|_q$$

wobei $1/p + 1/q = 1$.



Beweis (für $p = q = 2$): Mit $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ beliebig, seien $\tilde{\boldsymbol{x}} = \langle |x_1|, \ldots, |x_n| \rangle$ und $\tilde{\boldsymbol{y}} = \langle |y_1|, \ldots, |y_n| \rangle$. Dann gilt

$$\begin{aligned}
\|\boldsymbol{x}\|_2^2 + 2\left|\textstyle\sum_{i=1}^n x_i y_i\right| + \|\boldsymbol{y}\|_2^2 &\leq \\
\|\boldsymbol{x}\|_2^2 + 2\textstyle\sum_{i=1}^n |x_i y_i| + \|\boldsymbol{y}\|_2^2 &= \|\tilde{\boldsymbol{x}}\|_2^2 + 2\tilde{\boldsymbol{x}} \cdot \tilde{\boldsymbol{y}} + \|\tilde{\boldsymbol{y}}\|_2^2 \\
= \|\tilde{\boldsymbol{x}} + \tilde{\boldsymbol{y}}\|_2^2 &\leq [\|\tilde{\boldsymbol{x}}\|_2 + \|\tilde{\boldsymbol{y}}\|_2]^2 \\
&= \|\tilde{\boldsymbol{x}}\|_2^2 + 2\|\tilde{\boldsymbol{x}}\|_2 \|\tilde{\boldsymbol{y}}\|_2 + \|\tilde{\boldsymbol{y}}\|_2^2 \quad \blacksquare
\end{aligned}$$

Äquivalenz von Normen

Satz: In $\mathbb{R}^n$ sind alle Normen äquivalent, d.h. für 2 beliebige Normen $\|\boldsymbol{x}\|_{\mathrm{A}}$ und $\|\boldsymbol{x}\|_{\mathrm{B}}$ auf $\mathbb{R}^n$ $\exists c_1, c_2 > 0$ mit

$$c_1\|\boldsymbol{x}\|_{\mathrm{A}} \leq \|\boldsymbol{x}\|_{\mathrm{B}} \leq c_2\|\boldsymbol{x}\|_{\mathrm{A}} \quad \forall \boldsymbol{x} \in \mathbb{R}^n$$

Insbesondere:

Satz: $\forall \boldsymbol{x} \in \mathbb{R}^n$,

$$\|\boldsymbol{x}\|_\infty \leq \|\boldsymbol{x}\|_2 \leq \sqrt{n}\|\boldsymbol{x}\|_\infty$$

Beweis: Für $\boldsymbol{x} \in \mathbb{R}^n$ sei $|x_k| = \|\boldsymbol{x}\|_\infty$. Dann gilt

$$\|\boldsymbol{x}\|_\infty^2 = |x_k|^2 \leq \sum_{i=1}^{n} |x_i|^2 = \|\boldsymbol{x}\|_2^2 \leq \sum_{i=1}^{n} |x_k|^2 = n|x_k|^2 = n\|\boldsymbol{x}\|_\infty^2 \quad \blacksquare$$

Hausaufgabe: Zeige Äquivalenz der Normen $\|\cdot\|_1$ und $\|\cdot\|_2$.

Hausaufgabe: Beweise den folgenden Satz.

Satz: Für $A \in \mathbb{R}^{n\times n}$ SPD ist $\|\boldsymbol{x}\|_A = (\boldsymbol{x}^\top A\boldsymbol{x})^{\frac{1}{2}}$ eine Norm auf $\mathbb{R}^n$.

Qualitative Eigenschaften der Normen

- Soll ein Fehler $(\boldsymbol{x} - \boldsymbol{x}^*)$ mit der 1-, 2- oder ∞-Norm quantifiziert werden? Mit einer anderen Norm?
- Antwort: Es hängt vom Kontext ab!
- Beispiel: Die gemessenen pH-Werte einer chemischen Lösung sind $\boldsymbol{f} = \langle a, b, \ldots, b \rangle \in \mathbb{R}^{n+1}$.
- Augenscheinlich ist b die beste Abschätzung vom pH.
- Realistischer wäre $\boldsymbol{f}$ + Rauschen, aber die einfache Form von $\boldsymbol{f}$ betont den *Ausreißer* a und vereinfacht Rechnungen. Das Ergebnis ist qualitativ gleich mit Rauschen:
- Hausaufgabe: Für $\mathbf{1} = \langle 1, \ldots, 1 \rangle$ zeige

$$\frac{a + nb}{1 + n} = \operatorname*{argmin}_{c \in \mathbb{R}} \|\boldsymbol{f} - c\mathbf{1}\|_2^2 \quad \text{während} \quad b = \operatorname*{argmin}_{c \in \mathbb{R}} \|\boldsymbol{f} - c\mathbf{1}\|_1$$

- Ausreißer beeinflussen das Ergebnis stärker mit $\|\cdot\|_2$.
- Statistisch robuster ist $\|\cdot\|_1$. Siehe `Maximum Likelihood`.

Matrixnormen

Ein Konzept der Größe einer Matrix gehört auch zum Werkzeug.

Def: Eine Matrixnorm ist eine Funktion $\|\cdot\| : \mathbb{R}^{n\times n} \to \mathbb{R}$ mit den Eigenschaften:

1. $\|A\| \geq 0, \forall A \in \mathbb{R}^{n\times n}$
2. $\|A\| = 0 \Leftrightarrow A = 0$
3. $\|\alpha A\| = |\alpha|\|A\|, \forall A \in \mathbb{R}^{n\times n}, \forall \alpha \in \mathbb{R}$
4. $\|A + B\| \leq \|A\| + \|B\|, \forall A, B \in \mathbb{R}^{n\times n}$
5. $\|AB\| \leq \|A\|\|B\|, \forall A, B \in \mathbb{R}^{n\times n}$

Def: Für eine Vektornorm $\|\cdot\|_\natural$ ist die entsprechende *induzierte* (oder *natürliche*) Matrixnorm $\|\cdot\|_\natural$ definiert durch:

$$\|A\|_\natural = \max_{\|\boldsymbol{x}\|_\natural = 1} \|A\boldsymbol{x}\|_\natural$$

Hausaufgabe: Zeige dass diese Funktion $\|\cdot\|_\natural : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$ eine Matrixnorm ist.

Charakterisierung bekannter Matrixnormen

Satz: Die Matrixnormen $\|\cdot\|_1$, $\|\cdot\|_2$ und $\|\cdot\|_\infty$ sind für eine Matrix $\{A_{ij}\} = A \in \mathbb{R}^{n\times n}$ explizit so gegeben:

$$\begin{aligned}
\|A\|_1 &= \max_{\|\boldsymbol{x}\|_1=1} \|A\boldsymbol{x}\|_1 &&= \max_{1\le j\le n} \sum_{i=1}^{n} |a_{ij}| \\
\|A\|_2 &= \max_{\|\boldsymbol{x}\|_2=1} \|A\boldsymbol{x}\|_2 &&= \left[\max\{\lambda \in \sigma(A^\top A)\}\right]^{\frac{1}{2}} \\
\|A\|_\infty &= \max_{\|\boldsymbol{x}\|_\infty=1} \|A\boldsymbol{x}\|_\infty &&= \max_{1\le i\le n} \sum_{j=1}^{n} |a_{ij}|
\end{aligned}$$

Def: Eine Matrixnorm $\|\cdot\|_{\mathrm{M}}$ ist mit einer Vektornorm $\|\cdot\|_{\mathrm{V}}$ *verträglich* (oder *kompatibel*) wenn gilt

$$\|A\boldsymbol{x}\|_{\mathrm{V}} \le \|A\|_{\mathrm{M}}\|\boldsymbol{x}\|_{\mathrm{V}}, \quad \forall \boldsymbol{x} \in \mathbb{R}^n, \quad \forall A \in \mathbb{R}^{n\times n}$$

Hausaufgabe: Zeige dass die Funktion $\|\cdot\|_{\mathrm{F}} : \mathbb{R}^{n\times n} \to \mathbb{R}$ eine Matrixnorm ist, nämlich die *Frobenius* Matrixnorm,

$$\|A\|_{\mathrm{F}} = \Big[\sum_{i,j=1}^{n} |A_{ij}|^2\Big]^{\frac{1}{2}}$$

die mit der Vektornorm $\|\cdot\|_2$ verträglich ist. Sie ist aber von keiner Vektornorm induziert: $\|I\|_{\mathrm{F}} = \sqrt{n} > 1 = \|I\|_{\natural}$.

Spektralradius

Def: Sei $\mathbb{R}^n$ mit einer beliebigen Vektornorm $\|\cdot\|_{\mathrm{V}}$ versehen. Für $\boldsymbol{x}^{(k)}, \boldsymbol{x} \in \mathbb{R}^n$ gilt $\lim_{k\to\infty} \boldsymbol{x}^{(k)} = \boldsymbol{x}$ genau dann wenn $\forall \epsilon > 0$, $\exists K, \ni \forall k \geq K, \|\boldsymbol{x} - \boldsymbol{x}^{(k)}\|_{\mathrm{V}} < \epsilon$.

Satz: Für $\boldsymbol{x}^{(k)} = \langle x_1^{(k)}, \ldots, x_n^{(k)} \rangle, \boldsymbol{x} = \langle x_1, \ldots, x_n \rangle \in \mathbb{R}^n$ gilt $\lim_{k\to\infty} \boldsymbol{x}^{(k)} = \boldsymbol{x}$ genau dann wenn $\lim_{k\to\infty} x_i^{(k)} = x_i$.

Beweis: Hausaufgabe. ■

Def: Die Eigenwerte einer Matrix $A \in \mathbb{R}^{n\times n}$ sind die Nullstellen des charakteristischen Polynoms

$$p(\lambda) = \det(A - \lambda I), \quad \sigma(A) = \{\lambda : p(\lambda) = 0\}$$

und die entsprechenden Eigenvektoren sind die nicht trivialen Lösungen ($\boldsymbol{x} \neq 0$) der Gleichung $A\boldsymbol{x} = \lambda\boldsymbol{x}$.

Def: Der Spektralradius einer Matrix $A \in \mathbb{R}^{n\times n}$ ist

$$\rho(A) = \max\{|\lambda| : \lambda \in \sigma(A)\}$$

Spektralradius

- $\rho(A)$ ist der größte *Skalierungsfaktor*.
  Beispiel: $A = S\Lambda S^{-1}$, $S = [\boldsymbol{x}_1, \boldsymbol{x}_2]$,

$$\Lambda = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}, \qquad \begin{aligned} \Lambda\hat{\boldsymbol{e}}_1 &= \hat{\boldsymbol{e}}_1, & A\boldsymbol{x}_1 &= \boldsymbol{x}_1 \\ \Lambda\hat{\boldsymbol{e}}_2 &= 2\hat{\boldsymbol{e}}_2, & A\boldsymbol{x}_2 &= 2\boldsymbol{x}_2 \quad \leftarrow \end{aligned}$$

Satz: Sei $\|\cdot\|_\natural$ eine beliebige Matrixnorm auf $\mathbb{R}^{n\times n}$, die von der Vektornorm $\|\cdot\|_\natural$ auf $\mathbb{R}^n$ induziert ist. Dann gilt

$$\rho(A) \leq \|A\|_\natural, \quad \forall A \in \mathbb{R}^{n\times n}$$

Beweis: Wähle λ, $\boldsymbol{x}$ so aus, dass $A\boldsymbol{x} = \lambda\boldsymbol{x}$, $|\lambda| = \rho(A)$ und $\|\boldsymbol{x}\|_\natural = 1$ gelten. Dann gilt

$$\rho(A) = |\lambda| = |\lambda|\|\boldsymbol{x}\|_\natural = \|\lambda\boldsymbol{x}\|_\natural = \|A\boldsymbol{x}\|_\natural \leq \|A\|_\natural\|\boldsymbol{x}\|_\natural = \|A\|_\natural. \quad \blacksquare$$

- Wie wird eine Lösungsiteration

$$\boldsymbol{x}^{(k+1)} = T\boldsymbol{x}^{(k)} + \boldsymbol{c}$$

von $A\boldsymbol{x} = \boldsymbol{b}$ hergeleitet? Zerlegung: $A = M + N$, wobei

Konvergente Matrizen

Systeme mit M leicht zu lösen sind:

$$M\boldsymbol{x}^{(k+1)} = \boldsymbol{b} - N\boldsymbol{x}^{(k)}$$

so $\boldsymbol{c} = M^{-1}\boldsymbol{b}$ und $T = -M^{-1}N$. Ein Fixpunkt ist $\boldsymbol{x}^\star = A^{-1}\boldsymbol{b}$,

$$M\boldsymbol{x}^\star = \boldsymbol{b} - N\boldsymbol{x}^\star.$$

Satz: Sei $\boldsymbol{x}^\star \in \mathbb{R}^n$ ein Fixpunkt der Iteration $\boldsymbol{x}^{(k+1)} = T\boldsymbol{x}^{(k)} + \boldsymbol{c}$. Angenommen gilt $\|T\|_\natural < 1$, wobei $\|\cdot\|_\natural$ eine induzierte Matrixnorm ist. Dann folgt $\boldsymbol{x}^{(k)} \overset{k\to\infty}{\longrightarrow} \boldsymbol{x}^\star$, $\forall \boldsymbol{x}^{(0)} \in \mathbb{R}^n$.

Beweis: Es ergibt sich $(\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^\star) = T(\boldsymbol{x}^{(k)} - \boldsymbol{x}^\star)$ aus $\boldsymbol{x}^{(k+1)} = T\boldsymbol{x}^{(k)} + \boldsymbol{c}$ und $\boldsymbol{x}^\star = T\boldsymbol{x}^\star + \boldsymbol{c}$. Es folgt $\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^\star\|_\natural \le \|T\|_\natural \|\boldsymbol{x}^{(k)} - \boldsymbol{x}^\star\|_\natural \le \cdots \le \|T\|_\natural^k \|\boldsymbol{x}^{(0)} - \boldsymbol{x}^\star\|_\natural$. Da $\|T\|_\natural < 1$ gilt, folgen $\|T\|_\natural^k \overset{k\to\infty}{\longrightarrow} 0$ und $\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^\star\|_\natural \overset{k\to\infty}{\longrightarrow} 0$. ■

Def: $T \in \mathbb{R}^{n\times n}$ ist *konvergent* wenn gilt

$$\lim_{k\to\infty} (T^k)_{ij} = 0, \quad \forall 1 \le i, j \le n$$

Konvergente Matrizen

- Beispiele:

$$A = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}, \qquad A^k = \begin{bmatrix} 2^k & 0 \\ 0 & 1 \end{bmatrix} \not\to \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{3} \end{bmatrix}, \qquad B^k = \begin{bmatrix} \frac{1}{2^k} & 0 \\ 0 & \frac{1}{3^k} \end{bmatrix} \longrightarrow \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$C = \begin{bmatrix} \frac{1}{2} & 0 \\ \frac{1}{4} & \frac{1}{2} \end{bmatrix}, \qquad C^k = \begin{bmatrix} \frac{1}{2^k} & 0 \\ \frac{k}{2^{k+1}} & \frac{1}{2^k} \end{bmatrix} \longrightarrow \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$D = \begin{bmatrix} -\frac{3}{2} & -\frac{16}{5} \\ \frac{6}{5} & \frac{5}{2} \end{bmatrix}, \qquad D^k = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^{-1} \begin{bmatrix} (\frac{9}{10})^k & 0 \\ 0 & (\frac{1}{10})^k \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \longrightarrow \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

- $\|D\|_1 = 5.7$, $\|D\|_2 = 4.4922$, $\|D\|_\infty = 4.7$ und $\rho(D) = 0.9$.
- Gibt es eine Matrixnorm $\|\cdot\|_{\mathrm{M}}$ wobei $\|D\|_{\mathrm{M}} < 1$?

Satz: Sei $A \in \mathbb{R}^{n\times n}$ gegeben. $\forall \epsilon > 0$ gibt es eine induzierte Matrixnorm $\|\cdot\|_\epsilon$ wobei gilt $\rho(A) + \epsilon > \|A\|_\epsilon$.

- Bemerkung: Diese Matrixnorm $\|\cdot\|_\epsilon$ hängt von A ab.

Zerlegungen für Iterationen

Satz: Die folgenden sind äquivalent:

1. $T \in \mathbb{R}^{n\times n}$ ist konvergent
2. $\lim_{k\to\infty} \|T^k\|_\natural = 0$ für eine induzierte Matrixnorm $\|\cdot\|_\natural$
3. $\lim_{k\to\infty} \|T^k\|_\natural = 0$ für alle induzierten Matrixnormen $\|\cdot\|_\natural$
4. $\rho(T) < 1$
5. $\lim_{k\to\infty} T^k \boldsymbol{x} = 0,\ \forall \boldsymbol{x} \in \mathbb{R}^n$

- In der Zerlegung $A = M + N$,

$$M\boldsymbol{x}^{(k+1)} = \boldsymbol{b} - N\boldsymbol{x}^{(k)}$$

welches M ist geeignet? Besonders wenn A diagonal dominant ist, ist die folgende Zerlegung natürlich:

$$A = D + L + U, \quad M = D, \quad N = L + U.$$

Hier sind D eine diagonale Matrix, L eine streng untere Dreiecksmatrix und U eine streng obere Dreiecksmatrix.

Jacobi Methode

- Die Jacobi Methode: $M_{\mathrm{J}} = D$, $T_{\mathrm{J}} = -D^{-1}(L+U)$,

$$\begin{aligned} \boldsymbol{x}^{(k+1)} &= D^{-1}[\boldsymbol{b} - (L+U)\boldsymbol{x}^{(k)}] \\ &= D^{-1}[\boldsymbol{b} - \underbrace{(L+U+D)}_{=A}\boldsymbol{x}^{(k)} + D\boldsymbol{x}^{(k)}] \\ &= \boldsymbol{x}^{(k)} + D^{-1}[\boldsymbol{b} - A\boldsymbol{x}^{(k)}] \\ &= \boldsymbol{x}^{(k)} + D^{-1}\boldsymbol{r}^{(k)} \end{aligned}$$

  wobei $\boldsymbol{r} = \boldsymbol{b} - A\boldsymbol{x}$ das *Residuum* für gegebenes $\boldsymbol{x}$ ist.

- Die Jacobi Methode, komponentenweise,

$$\sum_{j<i} a_{ij} x_j^{(k)} + a_{ii} x_i^{(k+1)} + \sum_{j>i} a_{ij} x_j^{(k)} = b_i, \quad 1 \le i \le n$$

- Im folgenden Pseudo-Code wird x_i sequentiell in einer Schleife $i = 1, \ldots, n$ berechnet:

Pseudo-Code der Jacobi Methode

```
d(i) = A(i,i), i=1,...,n
x(i) = b(i), i=1,...,n
for it=1,...,itmax
    for i=1,...,n
        r(i) = b(i)
        for j=1,...,n
            r(i) = r(i) - A(i,j) * x(j)
        end                                   % Residuum
        dx(i) = r(i) / d(i)                   % Skalierung
    end
    for i=1,...,n
        x(i) = x(i) + dx(i)
    end                                       % Aktualisierung
    if (||dx|| < tol * ||x||)
        break                                 % Abbruch
    end
end

% Abbruchsparameter:  itmax ≪ O(n^3)
% tol = relative Fehlertoleranz
```

Jacobi Methode

- Wenn ein neuer Wert `x(i)+dx(i)` im Pseudo-Code verfügbar ist, warum nicht in `x(i)` sofort speichern? So entfernt man die Zeilen zwischen den folgenden

```
dx(i) = r(i) / d(i)
x(i) = x(i) + dx(i)
```

  damit diese neben einander innerhalb einer einzigen `i`-Schleife stehen, oder noch einfacher:

```
x(i) = x(i) + r(i)/d(i)
```

- Wenn im Pseudo-Code die eine `i`-Schleife so laufen sollte, würde die neue Iteration so aussehen:

$$\sum_{j<i} a_{ij} x_j^{(k+1)} + a_{ii} x_i^{(k+1)} + \sum_{j>i} a_{ij} x_j^{(k)} = b_i, \quad 1 \leq i \leq n$$

  oder

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left[b_i - \sum_{j<i} a_{ij} x_j^{(k+1)} - \sum_{j>i} a_{ij} x_j^{(k)} \right]$$

Gauß-Seidel Methode

- In Matrixnotation,

$$\boldsymbol{x}^{(k+1)} = D^{-1}[\boldsymbol{b} - L\boldsymbol{x}^{(k+1)} - U\boldsymbol{x}^{(k)}]$$

  oder die *Gauß-Seidel* Methode,

$$(D + L)\boldsymbol{x}^{(k+1)} = \boldsymbol{b} - U\boldsymbol{x}^{(k)}$$

- Die Zerlegung $A = M + N$ ist $N = U$ und

$$M_{\mathrm{J}} = D + L, \quad T_{\mathrm{GS}} = -(D + L)^{-1}U.$$

- Umgeschrieben,

$$\begin{aligned}
\boldsymbol{x}^{(k+1)} &= \underbrace{(D + L)^{-1}}_{\text{Vorwärts Substitution}}\,[\boldsymbol{b} - U\boldsymbol{x}^{(k)}] \\
&= (D + L)^{-1}[\boldsymbol{b} - \underbrace{(D + L + U)}_{=A}\boldsymbol{x}^{(k)} + (D + L)\boldsymbol{x}^{(k)}] \\
&= \boldsymbol{x}^{(k)} + (D + L)^{-1}[\boldsymbol{b} - A\boldsymbol{x}^{(k)}] = \boldsymbol{x}^{(k)} + (D + L)^{-1}\boldsymbol{r}^{(k)}
\end{aligned}$$

Symmetrische Gauß-Seidel Methode

- Es gibt eine gewisse Asymmetrie bei der Gauß-Seidel Methode.
- Es wäre genauso sinnvoll, wenn die eine `i`-Schleife rückwärts (`i=n,...,1`) anstatt vorwärts (`i=1,...,n`) laufen sollte.
- Mit so einer rückwärts laufenden `i`-Schleife ergibt sich $\boldsymbol{x}^{(k+1)} = (D+U)^{-1}[\boldsymbol{b} - L\boldsymbol{x}^{(k)}]$.
- Mit *beiden* Schleifen bekommt man die *Symmetrische Gauß-Seidel* Methode,

$$\begin{cases} \boldsymbol{x}^{(k+\frac{1}{2})} &= (D+L)^{-1}[\boldsymbol{b} - U\boldsymbol{x}^{(k)}] \\ \boldsymbol{x}^{(k+1)} &= (D+U)^{-1}[\boldsymbol{b} - L\boldsymbol{x}^{(k+\frac{1}{2})}] \end{cases}$$

- Hausaufgabe: Finde die Iterationsmatrix T_{SGS} für die Symmetrische Gauß-Seidel Methode.

Approximierte Inversen

Bemerkung: Die Zerlegung $A = M + N$ kann so umgeschrieben werden:

$$\begin{aligned} \boldsymbol{x}^{(k+1)} &= -M^{-1}N\boldsymbol{x}^{(k)} + M^{-1}\boldsymbol{b} \\ &= [-M^{-1}(M+N) + I]\boldsymbol{x}^{(k)} + M^{-1}\boldsymbol{b} \\ &= [I - M^{-1}A]\boldsymbol{x}^{(k)} + M^{-1}\boldsymbol{b} \end{aligned}$$

und so heisst M^{-1} die *approximierte Inverse*, da $\|I - M^{-1}A\|$ klein sein sollte.

Hausaufgabe: Sei $A = D + L + U$ eine streng diagonal dominante Matrix mit D diagonal, L streng unterdreickig und U streng oberdreickig.

- Bestimme die approximierten Inversen M_{J} und M_{GS} für die Jacobi-Methode bzw. Gauß-Seidel-Methode und zeige, es gelten $\rho(I - M_{\mathrm{J}}^{-1}A) < 1$ und $\rho(I - M_{\mathrm{GS}}^{-1}A) < 1$.
- Bestimme die Approximierte Inverse M_{SGS} für die symmetrische Gauß-Seidel Methode.
- Zeige, wenn A symmetrisch ist, ist M_{SGS} symmetrisch, obwohl M_{GS} im Allgemeinen nicht symmetrisch ist.

SOR Methode

- Gauß-Seidel noch einmal komponentenweise,

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left[b_i - \sum_{j<i} a_{ij} x_j^{(k+1)} - \sum_{j>i} a_{ij} x_j^{(k)} \right] \omega + x_i^{(k)} (1-\omega)$$

  Das alte und das neue können so gewichtet werden.

- In Matrix-Notation,

$$\begin{aligned}
\boldsymbol{x}^{(k+1)} &= D^{-1}[\boldsymbol{b} - L\boldsymbol{x}^{(k+1)} - U\boldsymbol{x}^{(k)}]\omega + \boldsymbol{x}^{(k)}(1-\omega) \\
D\boldsymbol{x}^{(k+1)} &= [\boldsymbol{b} - L\boldsymbol{x}^{(k+1)} - U\boldsymbol{x}^{(k)}]\omega + D\boldsymbol{x}^{(k)}(1-\omega) \\
(D+\omega L)\boldsymbol{x}^{(k+1)} &= [\boldsymbol{b} - U\boldsymbol{x}^{(k)}]\omega + D\boldsymbol{x}^{(k)}(1-\omega) \\
\boldsymbol{x}^{(k+1)} &= (D+\omega L)^{-1}[(1-\omega)D - \omega U]\boldsymbol{x}^{(k)} + \omega(D+\omega L)^{-1}\boldsymbol{b}
\end{aligned}$$

- Diese Methode heisst *Successive Over-Relaxation* (SOR)
  - $0 < \omega < 1 \Rightarrow$ *under relaxed*, d.h. sie konvergiert wegen *Dämpfung*, wenn sie bei $\omega = 1$ möglicherweise nicht konvergiert.
  - $\omega > 1 \Rightarrow$ *over relaxed*, d.h. sie konvergiert schneller wegen *Extrapolation*, wenn sie bei $\omega = 1$ schon zuverlässig konvergiert.

Konvergenz von Iterativen Verfahren

- Die *Symmetrische Successive Over-Relaxation* (SSOR) Methode:

$$\begin{cases} \boldsymbol{x}^{(k+\frac{1}{2})} &= (D+\omega L)^{-1}[(1-\omega)D-\omega U]\boldsymbol{x}^{(k)}+\omega(D+\omega L)^{-1}\boldsymbol{b} \\ \boldsymbol{x}^{(k+1)} &= (D+\omega U)^{-1}[(1-\omega)D-\omega L]\boldsymbol{x}^{(k+\frac{1}{2})}+\omega(D+\omega U)^{-1}\boldsymbol{b} \end{cases}$$

- Hausaufgabe: Bestimme die Iterationsmatrix T_{SSOR} und die approximierte Inverse M_{SSOR} für SSOR.

Satz: Wenn für das System $A\boldsymbol{x}^\star = \boldsymbol{b}$ gilt $\boldsymbol{x}^\star = T\boldsymbol{x}^\star + \boldsymbol{c}$, dann erfüllt das iterative Verfahren $\boldsymbol{x}^{(k+1)} = T\boldsymbol{x}^{(k)} + \boldsymbol{c}$ die Konvergenz $\lim_{k\to\infty} \boldsymbol{x}^{(k)} = \boldsymbol{x}^\star$, genau dann wenn $\rho(T) < 1$.

Bemerkung: Wenn für eine induzierte Norm $\|T\|_\natural < 1$ gezeigt werden kann, folgt $\rho(T) < 1$.

Satz: Wenn A streng diagonal dominant ist, gilt $\lim_{k\to\infty} \boldsymbol{x}^{(k)} = \boldsymbol{x}^\star$, $\forall \boldsymbol{x}^{(0)} \in \mathbb{R}^n$, für die Jacobi und Gauß-Seidel Methoden.

Konvergenz von Iterativen Verfahren

- Das Gewicht ω wird für SOR so ausgewählt, dass $\rho(T_{\text{SOR}}) < 1$ gilt, wobei $T_{\text{SOR}} = (D + \omega L)^{-1}[(1-\omega)D - \omega U]$.

Satz (Kahan): Sei $\{a_{ij}\} = A \in \mathbb{R}^n$ gegeben. Wenn $a_{ii} \neq 0$, $1 \leq i \leq n$, gilt, folgt $\rho(T_{\text{SOR}}) \geq |\omega - 1|$.

Deswegen gilt $\rho(T_{\text{SOR}}) < 1$ nur für $\omega \in (0, 2)$. Umgekehrt:

Satz (Ostrowski-Reich): Wenn A SPD ist und $0 < \omega < 2$ gilt, folgt für SOR die Konvergenz $\lim_{k\to\infty} \boldsymbol{x}^{(k)} = \boldsymbol{x}^\star$, $\forall \boldsymbol{x}^{(0)} \in \mathbb{R}^n$.

- Beispiel: Für die Diskretisierung $A\boldsymbol{u} = [\mu D/h^2 + I]\boldsymbol{u} = \boldsymbol{v}$ des Randwertproblems,

$$\begin{cases} -\mu u'' + u &= v, & x \in (0,1) \\ u' &= 0, & x \in \{0,1\} \end{cases}$$

ist A streng diagonal dominant, sogar SPD. Deswegen konvergieren Jacobi, Gauß-Seidel und SOR, $\forall \omega \in (0, 2)$.

Konjugierte Gradienten

Die Methode der Konjugierten Gradienten

Satz: Seien $f \in \mathcal{C}^2(\Omega)$ und $\boldsymbol{x}^\star \in \Omega$. Wenn $\nabla f(\boldsymbol{x}^\star) = 0$ erfüllt ist und $\nabla^2 f(\boldsymbol{x}^\star)$ SPD ist, ist $\boldsymbol{x}^\star$ ein lokales Minimum für f.

Explizite Beispiele:

- Für $f(\boldsymbol{x}) = \boldsymbol{b}^\top \boldsymbol{x}$ gelten $\nabla f(\boldsymbol{x}) = \boldsymbol{b}$ und $\nabla^2 f(\boldsymbol{x}) = 0$.

$$\frac{\partial}{\partial x_k} \sum_{i=1}^{n} b_i x_i = \sum_{i=1}^{n} b_i \frac{\partial x_i}{\partial x_k} = \sum_{i=1}^{n} b_i \delta_{ik} = b_k$$

- Für $f(\boldsymbol{x}) = \boldsymbol{x}^\top M \boldsymbol{x}$, $M \in \mathbb{R}^{n \times n}$, gelten $\nabla f(\boldsymbol{x}) = (M + M^\top)\boldsymbol{x}$ und $\nabla^2 f(\boldsymbol{x}) = M + M^\top$.

$$\frac{\partial}{\partial x_k} \sum_{i=1}^{n} \sum_{j=1}^{n} x_i M_{ij} x_j = \sum_{i=1}^{n} \sum_{j=1}^{n} \left(\frac{\partial x_i}{\partial x_k} M_{ij} x_j + x_i M_{ij} \frac{\partial x_j}{\partial x_k} \right)$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{n} \left(\delta_{ik} M_{ij} x_j + x_i M_{ij} \delta_{jk} \right) = \sum_{j=1}^{n} M_{kj} x_j + \sum_{i=1}^{n} M_{ik} x_i$$

Konjugierte Gradienten

Satz: Sei $A \in \mathbb{R}^{n \times n}$ SPD. Dann löst $\boldsymbol{x}^\star$ das System $A\boldsymbol{x} = \boldsymbol{b}$ genau dann wenn $\boldsymbol{x}^\star$ die Funktion $g(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^\top A\boldsymbol{x} - \boldsymbol{b}^\top \boldsymbol{x}$ global minimiert.

Beweis: ($\Rightarrow$) Aus $A\boldsymbol{x}^\star = \boldsymbol{b}$ folgt $\nabla g(\boldsymbol{x}^\star) = A\boldsymbol{x}^\star - \boldsymbol{b} = 0$. Weil A SPD ist, hat g genau einen kritischen Punkt. Da $\nabla^2 g(\boldsymbol{x}) = A$ SPD $\forall \boldsymbol{x} \in \mathbb{R}^n$ ist, ist g global konvex, und $\boldsymbol{x}^\star$ ist ein lokales Minimum. Aus der konvexen Analysis ist $\boldsymbol{x}^\star$ ein globales Minimum. ($\Leftarrow$) Mit einem globalen Minimum $g(\boldsymbol{x}^\star)$ von $g \in \mathcal{C}^\infty(\mathbb{R}^n)$ gilt notwendigerweise $\nabla g(\boldsymbol{x}^\star) = 0$. Da A SPD ist, $\exists! \boldsymbol{x}^\star \ni \nabla g(\boldsymbol{x}^\star) = A\boldsymbol{x}^\star - \boldsymbol{b} = 0$. ■

Vorausgesetzt ist A SPD, und $A\boldsymbol{x} = \boldsymbol{b}$ soll gelöst werden.

Entwicklung der Methode der Konjugierten Gradienten:

- Sei $\boldsymbol{x}^{(0)}$ eine Approximation der Lösung $\boldsymbol{x}^\star$.
- Sei $\boldsymbol{v}^{(1)}$ eine gegebene Suchrichtung zur Minimierung von g.
- Wo ist das Minimum von g entlang $\boldsymbol{v}^{(1)}$ von $\boldsymbol{x}^{(0)}$ weg?

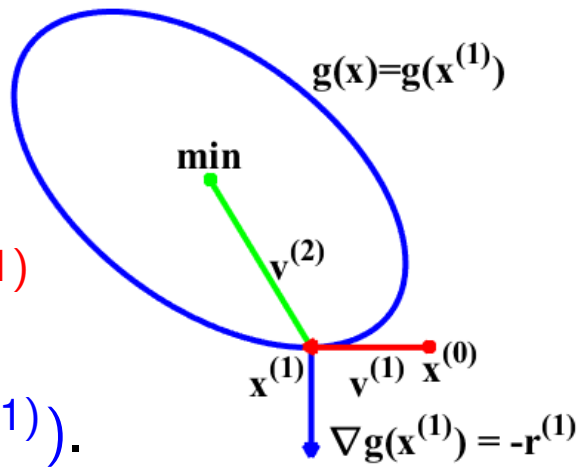
Konjugierte Gradienten

- Antwort durch die Ketten-Regel,

$$\frac{d}{dt} g(\boldsymbol{x}^{(0)} + t\boldsymbol{v}^{(1)}) = \nabla g(\boldsymbol{x}^{(0)} + t\boldsymbol{v}^{(1)}) \cdot \boldsymbol{v}^{(1)} = [A(\boldsymbol{x}^{(0)} + t\boldsymbol{v}^{(1)}) - \boldsymbol{b}] \cdot \boldsymbol{v}^{(1)} \stackrel{!}{=} 0$$

und das Minimum ist in $\boldsymbol{x}^{(1)} = \boldsymbol{x}^{(0)} + t^{(1)}\boldsymbol{v}^{(1)}$ wobei

$$t^{(1)} = \frac{(\boldsymbol{b} - A\boldsymbol{x}^{(0)}) \cdot \boldsymbol{v}^{(1)}}{\boldsymbol{v}^{(1)} \cdot A\boldsymbol{v}^{(1)}} = \frac{\boldsymbol{r}^{(0)} \cdot \boldsymbol{v}^{(1)}}{\boldsymbol{v}^{(1)} \cdot A\boldsymbol{v}^{(1)}}$$



- Eigenschaft des Gradienten:
  $\nabla g(\boldsymbol{x}^{(1)}) \perp$ Niveau Kurve $g(\boldsymbol{x}) = g(\boldsymbol{x}^{(1)})$
- Orthogonalität $-\boldsymbol{r}^{(1)} \cdot \boldsymbol{v}^{(1)} = \nabla g(\boldsymbol{x}^{(1)}) \cdot \boldsymbol{v}^{(1)}$
  $= [A(\boldsymbol{x}^{(0)} + t^{(1)}\boldsymbol{v}^{(1)}) - \boldsymbol{b}] \cdot \boldsymbol{v}^{(1)} = 0$
  bedeutet $\boldsymbol{v}^{(1)} \parallel$ Niveau Kurve $g(\boldsymbol{x}) = g(\boldsymbol{x}^{(1)})$.
- Wie soll die Suchrichtung $\boldsymbol{v}^{(2)}$ ausgewählt werden?
- Antwort: Minimiere g über span$\{\boldsymbol{x}^{(1)} + \boldsymbol{r}^{(1)}, \boldsymbol{x}^{(1)} + \boldsymbol{v}^{(1)}\}$.

Konjugierte Gradienten

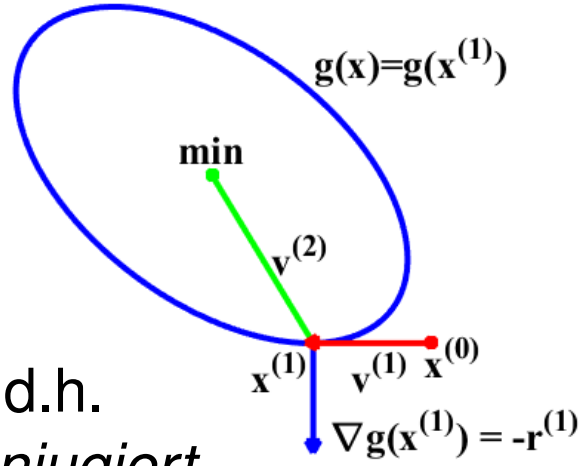
- Die Funktion $(t, \sigma) \mapsto g(\boldsymbol{x}^{(1)} + t\boldsymbol{r}^{(1)} + \sigma \boldsymbol{v}^{(1)})$ wird minimiert:

$$\frac{d}{d\sigma} g(\boldsymbol{x}^{(1)} + t\boldsymbol{r}^{(1)} + \sigma \boldsymbol{v}^{(1)}) = \nabla g(\boldsymbol{x}^{(1)} + t\boldsymbol{r}^{(1)} + \sigma \boldsymbol{v}^{(1)}) \cdot \boldsymbol{v}^{(1)} \overset{\sigma = ts}{=} \cdots =$$

$$[A(\boldsymbol{x}^{(1)} + t\boldsymbol{r}^{(1)} + ts\boldsymbol{v}^{(1)}) - \boldsymbol{b}] \cdot \boldsymbol{v}^{(1)} = \underbrace{(A\boldsymbol{x}^{(1)} - \boldsymbol{b}) \cdot \boldsymbol{v}^{(1)}}_{-\boldsymbol{r}^{(1)} \cdot \boldsymbol{v}^{(1)} = 0} + tA \underbrace{(\boldsymbol{r}^{(1)} + s\boldsymbol{v}^{(1)})}_{= \boldsymbol{v}^{(2)}} \cdot \boldsymbol{v}^{(1)} \overset{!}{=} 0$$

- Nimm $\boldsymbol{v}^{(2)} = \boldsymbol{r}^{(1)} + s^{(1)} \boldsymbol{v}^{(1)}$ wobei

$$s^{(1)} = -\frac{\boldsymbol{r}^{(1)} \cdot A\boldsymbol{v}^{(1)}}{\boldsymbol{v}^{(1)} \cdot A\boldsymbol{v}^{(1)}}$$



- Bemerke:
  $\boldsymbol{v}^{(2)} \cdot A\boldsymbol{v}^{(1)} = (\boldsymbol{r}^{(1)} + s^{(1)} \boldsymbol{v}^{(1)}) \cdot A\boldsymbol{v}^{(1)} = 0$, d.h.
  $\boldsymbol{v}^{(1)}$ und $\boldsymbol{v}^{(2)}$ sind *A-orthogonal* oder *A-konjugiert*.
- $\frac{d}{dt} g(\boldsymbol{x}^{(1)} + t\boldsymbol{r}^{(1)} + ts^{(1)} \boldsymbol{v}^{(1)}) = \frac{d}{dt} g(\boldsymbol{x}^{(1)} + t\boldsymbol{v}^{(2)}) \overset{!}{=} 0$ führt zu $t^{(2)}$.
- Diese Formeln lassen sich für jede Iteration verallgemeinern:

Konjugierte Gradienten

Zusammenfassung der Formeln:

$\boldsymbol{x}^{(0)}$ sei gegeben

$\boldsymbol{v}^{(1)} = \boldsymbol{r}^{(0)} = \boldsymbol{b} - A\boldsymbol{x}^{(0)}$

$\boldsymbol{r}^{(k)} \cdot \boldsymbol{v}^{(k)} = 0$

$\boldsymbol{v}^{(k+1)} \cdot A\boldsymbol{v}^{(k)} = 0$

$$
\begin{aligned}
\text{Für } k &= 1, 2, \ldots \\
t^{(k)} &= \boldsymbol{r}^{(k-1)} \cdot \boldsymbol{v}^{(k)} / \boldsymbol{v}^{(k)} \cdot A\boldsymbol{v}^{(k)} \\
\boldsymbol{x}^{(k)} &= \boldsymbol{x}^{(k-1)} + t^{(k)} \boldsymbol{v}^{(k)} \\
\boldsymbol{r}^{(k)} &= \boldsymbol{b} - A\boldsymbol{x}^{(k)} \\
s^{(k)} &= -\boldsymbol{r}^{(k)} \cdot A\boldsymbol{v}^{(k)} / \boldsymbol{v}^{(k)} \cdot A\boldsymbol{v}^{(k)} \\
\boldsymbol{v}^{(k+1)} &= \boldsymbol{r}^{(k)} + s^{(k)} \boldsymbol{v}^{(k)}
\end{aligned}
$$

- Vereinfachungen:

$$
\boldsymbol{r}^{(k)} \cdot \boldsymbol{v}^{(k)} = 0 \Rightarrow t^{(k)} = \frac{\boldsymbol{r}^{(k-1)} \cdot [\boldsymbol{r}^{(k-1)} + s^{(k-1)} \boldsymbol{v}^{(k-1)}]}{\boldsymbol{v}^{(k)} \cdot A\boldsymbol{v}^{(k)}} = \frac{\boldsymbol{r}^{(k-1)} \cdot \boldsymbol{r}^{(k-1)}}{\boldsymbol{v}^{(k)} \cdot A\boldsymbol{v}^{(k)}}
$$

Also

$$
\begin{aligned}
\boldsymbol{r}^{(k)} = \boldsymbol{b} - A\boldsymbol{x}^{(k)} = \boldsymbol{b} - A(\boldsymbol{x}^{(k-1)} + t^{(k)} \boldsymbol{v}^{(k)}) = \boldsymbol{r}^{(k-1)} - t^{(k)} A\boldsymbol{v}^{(k)} \\
\Rightarrow \boldsymbol{r}^{(k)} \cdot \boldsymbol{r}^{(k-1)} = \boldsymbol{r}^{(k-1)} \cdot \boldsymbol{r}^{(k-1)} - t^{(k)} (\boldsymbol{v}^{(k)} - s^{(k-1)} \boldsymbol{v}^{(k-1)}) \cdot A\boldsymbol{v}^{(k)} = 0 \\
\Rightarrow \boldsymbol{r}^{(k)} \cdot \boldsymbol{r}^{(k)} = \boldsymbol{r}^{(k)} \cdot \boldsymbol{r}^{(k-1)} - t^{(k)} \boldsymbol{r}^{(k)} \cdot A\boldsymbol{v}^{(k)} = -t^{(k)} \boldsymbol{r}^{(k)} \cdot A\boldsymbol{v}^{(k)}
\end{aligned}
$$

und

$$
s^{(k)} = -\frac{\boldsymbol{r}^{(k)} \cdot A\boldsymbol{v}^{(k)}}{\boldsymbol{v}^{(k)} \cdot A\boldsymbol{v}^{(k)}} = \frac{\boldsymbol{r}^{(k)} \cdot \boldsymbol{r}^{(k)} / t^{(k)}}{\boldsymbol{r}^{(k-1)} \cdot \boldsymbol{r}^{(k-1)} / t^{(k)}} = \frac{\boldsymbol{r}^{(k)} \cdot \boldsymbol{r}^{(k)}}{\boldsymbol{r}^{(k-1)} \cdot \boldsymbol{r}^{(k-1)}}
$$

Konjugierte Gradienten

Die Methode der Konjugierten Gradienten:

$\boldsymbol{x}^{(0)}$ sei gegeben
$\boldsymbol{v}^{(1)} = \boldsymbol{r}^{(0)} = \boldsymbol{b} - A\boldsymbol{x}^{(0)}$

$$
\begin{aligned}
\text{Für } k &= 1, 2, \ldots \\
t^{(k)} &= \boldsymbol{r}^{(k-1)} \cdot \boldsymbol{r}^{(k-1)} / \boldsymbol{v}^{(k)} \cdot A\boldsymbol{v}^{(k)} \\
\boldsymbol{x}^{(k)} &= \boldsymbol{x}^{(k-1)} + t^{(k)} \boldsymbol{v}^{(k)} \\
\boldsymbol{r}^{(k)} &= \boldsymbol{r}^{(k-1)} - t^{(k)} A\boldsymbol{v}^{(k)} \\
s^{(k)} &= \boldsymbol{r}^{(k)} \cdot \boldsymbol{r}^{(k)} / \boldsymbol{r}^{(k-1)} \cdot \boldsymbol{r}^{(k-1)} \\
\boldsymbol{v}^{(k+1)} &= \boldsymbol{r}^{(k)} + s^{(k)} \boldsymbol{v}^{(k)}
\end{aligned}
$$

- Hausaufgabe: Schreibe einen effizienten Pseude-Code zur Implementierung dieser Methode.

Vorausgesetzt dass A SPD ist:

Satz: Es gelten $\boldsymbol{r}^{(k)} \cdot \boldsymbol{r}^{(l)} = 0$, $\boldsymbol{v}^{(k)} \cdot A\boldsymbol{v}^{(l)} = 0$, $1 \leq l < k$.

Satz: Mit exakter Arithmetik gilt $A\boldsymbol{x}^{(k)} = \boldsymbol{b}$ für ein $k \leq n$.

Präkonditionierung

- Angenommen $A \approx C^2$ gilt für ein SPD C, und Systeme $C\boldsymbol{y} = \boldsymbol{c}$ sind leichter zu lösen als Systeme $A\boldsymbol{x} = \boldsymbol{b}$.
- Das Problem $A\boldsymbol{x} = \boldsymbol{b}$ kann so umgeschrieben werden:

$$(C^{-1}AC^{-1})(C\boldsymbol{x}) = C^{-1}\boldsymbol{b}$$

- Falls die SPD Matrix $(C^{-1}AC^{-1})$ besser *konditioniert* ist als die Matrix A, konvergiert die Methode der Konjugierten Gradienten schneller für das transformierte System als für das ursprüngliche System.

Def: Sei $\|\cdot\|_\star$ eine induzierte Matrixnorm. Die *Konditionszahl* einer Matrix A bezüglich dieser Norm ist $\kappa_\star(A) = \|A\|_\star\|A^{-1}\|_\star$.

Bemerkung: $1 = \|I\|_\star = \|AA^{-1}\|_\star \leq \|A\|_\star\|A^{-1}\|_\star = \kappa_\star(A)$.
Beispiel:

$$A = \begin{bmatrix} 1 & 1-\epsilon \\ 1-\epsilon & 1 \end{bmatrix}_{\epsilon\in(0,1)} \qquad \begin{aligned} \sigma(A) &= \{\epsilon, 2-\epsilon\} \\ \kappa_2(A) &= (2-\epsilon)/\epsilon \end{aligned}$$

Fehler Abschätzungen

Satz: Eine gegebene Vektornorm und die entsprechende induzierte Matrixnorm sei mit $\|\cdot\|$ bezeichnet, und κ ist die entsprechende Konditionszahl. Dann für $A \in \mathbb{R}^{n\times n}$ und $\boldsymbol{b}, \boldsymbol{x}^\star = A^{-1}\boldsymbol{b}, \tilde{\boldsymbol{x}}, \in \mathbb{R}^n$ gilt:

$$\frac{\|\boldsymbol{x}^\star - \tilde{\boldsymbol{x}}\|}{\|\boldsymbol{x}^\star\|} \leq \kappa(A)\frac{\|\boldsymbol{b} - A\tilde{\boldsymbol{x}}\|}{\|\boldsymbol{b}\|}$$

Beweis: Aus $A\boldsymbol{x}^\star = \boldsymbol{b}$ folgt

$$\|\boldsymbol{b}\| = \|A\boldsymbol{x}^\star\| \leq \|A\|\|\boldsymbol{x}^\star\| \quad \Rightarrow \quad \frac{1}{\|\boldsymbol{x}^\star\|} \leq \frac{\|A\|}{\|\boldsymbol{b}\|}$$

Mit $\boldsymbol{e} = \boldsymbol{x}^\star - \tilde{\boldsymbol{x}}$ und $\boldsymbol{r} = \boldsymbol{b} - A\tilde{\boldsymbol{x}} = A\boldsymbol{x}^\star - A\tilde{\boldsymbol{x}} = A\boldsymbol{e}$ gilt

$$\|\boldsymbol{e}\| = \|A^{-1}A\boldsymbol{e}\| \leq \|A^{-1}\|\|A\boldsymbol{e}\| = \|A^{-1}\|\|\boldsymbol{r}\|$$

und daher

$$\frac{\|\boldsymbol{e}\|}{\|\boldsymbol{x}^\star\|} \leq \|A^{-1}\|\|A\|\frac{\|\boldsymbol{r}\|}{\|\boldsymbol{b}\|}$$

■

Fehler Abschätzungen

- Ist $(\|\boldsymbol{b} - A\boldsymbol{x}^{(k)}\| < \tau \cdot \|\boldsymbol{b}\|)$ ein gutes Abbruchskriterium einer Iteration $\boldsymbol{x}^{(k)} \overset{k\to\infty}{\longrightarrow} \boldsymbol{x}^\star$?
- Anhand der obigen Abschätzung ist es theoretisch möglich, dass $\|\boldsymbol{x}^\star - \boldsymbol{x}^{(k)}\|/\|\boldsymbol{x}^\star\|$ sehr groß werden kann.
- Pessimistisch? Nein. Beispiel:

$$A = \left[\begin{array}{cc} 1 & 1-\epsilon \\ 1-\epsilon & 1 \end{array}\right]_{\epsilon\in(0,1)} \qquad \begin{array}{l} \sigma(A) = \{\epsilon, 2-\epsilon\} \\ \kappa_2(A) = (2-\epsilon)/\epsilon \end{array}$$

$$\boldsymbol{x}^\star = \left[\begin{array}{c} \cos(\theta) \\ \sin(\theta) \end{array}\right]_{\theta=\frac{\pi}{4}} \qquad \boldsymbol{b} = A\boldsymbol{x}^\star = \left[\begin{array}{c} \cos(\theta) + (1-\epsilon)\sin(\theta) \\ \sin(\theta) + (1-\epsilon)\cos(\theta) \end{array}\right]$$

$$\|\boldsymbol{x}^\star\|_2 = 1, \quad \|\boldsymbol{b}\|_2^2 = 1 + (1-\epsilon)^2 + \underbrace{4\cos(\theta)\sin(\theta)}_{2\sin(2\theta)=2}(1-\epsilon) = (2-\epsilon)^2$$

$$\tilde{\boldsymbol{x}} = \boldsymbol{x}^\star + \frac{\boldsymbol{z}}{\delta}, \quad \boldsymbol{z} = \frac{1}{\sqrt{2}}\left[\begin{array}{c} 1 \\ -1 \end{array}\right], \quad A\boldsymbol{z} = \epsilon\boldsymbol{z}, \quad \|\boldsymbol{z}\|_2 = 1$$

$$\|\boldsymbol{x}^\star - \tilde{\boldsymbol{x}}\|_2 = \frac{\|\boldsymbol{z}\|_2}{\delta} = \frac{1}{\delta}, \quad \|\boldsymbol{b} - A\tilde{\boldsymbol{x}}\|_2 = \frac{\|A\boldsymbol{z}\|_2}{\delta} = \frac{\epsilon}{\delta}$$

Fehler Abschätzungen

- Zusammenfassung: mit $\delta = \sqrt{\epsilon}$,

$$\underbrace{\frac{\|\boldsymbol{x}^\star - \tilde{\boldsymbol{x}}\|_2}{\|\boldsymbol{x}^\star\|_2}}_{1/\delta \overset{\delta\to 0}{\longrightarrow} \infty} = \underbrace{\kappa_2(A)}_{(2-\delta^2)/\delta^2 \overset{\delta\to 0}{\longrightarrow} \infty} \underbrace{\frac{\|\boldsymbol{b} - A\tilde{\boldsymbol{x}}\|_2}{\|\boldsymbol{b}\|_2}}_{\delta/(2-\delta^2) \overset{\delta\to 0}{\longrightarrow} 0}$$

  d.h. es kann doch sein, dass das Residuum sehr klein wird, während der Fehler groß bleibt.

- Abbruchskriterium? Zu empfehlen ist:

$$\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}\| < \tau \cdot \|\boldsymbol{x}^{(k)}\|$$

  wobei die Toleranz τ rein von der Genauigkeit der Fließkommazahl-Darstellung gewählt werden kann, z.B. ca 2^{-23} für einfache Genauigkeit und ca 2^{-53} für doppelte Genauigkeit.

Eigenwerte und Eigenvektoren

Ein nützliches Werkzeug:

Satz: Wenn $A \in \mathbb{R}^{n\times n}$ die *verschiedenen* Eigenwerte $\{\lambda_i\}_{i=1}^k$, ($k \leq n$) mit den entsprechenden Eigenvektoren $\{\boldsymbol{x}_i\}_{i=1}^k$ hat, dann sind diese Eigenvektoren linear unabhängig. Wenn $k = n$ gilt, dann bilden die Eigenvektoren eine *Basis* für $\mathbb{R}^n$.

Def: $P \in \mathbb{R}^{n\times n}$ ist *orthogonal* wenn $P^{-1} = P^\top$ gilt.

Def: $A, B \in \mathbb{R}^{n\times n}$ sind *ähnlich* wenn $\exists S, S^{-1} \in \mathbb{R}^{n\times n} \ni A = S^{-1}BS$.

Satz: $A = S^{-1}BS \Rightarrow \sigma(A) = \sigma(B)$ und $A\boldsymbol{x} = \lambda\boldsymbol{x} \Rightarrow BS\boldsymbol{x} = \lambda S\boldsymbol{x}$.

Satz: Ist $A \in \mathbb{R}^{n\times n}$ symmetrisch mit $\sigma(A) = \{\lambda_i\}_{i=1}^n$ und $\Lambda = \mathrm{diag}(\{\lambda_i\}_{i=1}^n)$, dann $\exists P \in \mathbb{R}^{n\times n}$ orthogonal mit $\Lambda = P^\top AP$.

Satz: Ist $A \in \mathbb{R}^{n\times n}$ symmetrisch, gilt $\sigma(A) \subset \mathbb{R}$.

Satz: Ist $A \in \mathbb{R}^{n\times n}$ symmetrisch, dann bilden die Eigenvektoren eine *orthonormale Basis* für $\mathbb{R}^n$.

Satz: A ist SPD $\Leftrightarrow$ A ist symmetrisch und $\sigma(A) \subset \mathbb{R}_+$.

Vektoriteration

Zur Bestimmung eines dominanten Eigenwerts

- Der Plan: Für $A \in \mathbb{R}^{n \times n}$ und einen Startvektor $\boldsymbol{x}^{(0)}$, soll $A^k \boldsymbol{x}^{(0)}$ einigermaßen zu einem Eigenvektor konvergieren, der zum dominanten Eigenwert gehört.
- Angenommen $A \in \mathbb{R}^{n \times n}$ hat die Eigenwerte $\{\lambda_i\}_{i=1}^n$ mit den entsprechenden Eigenvektoren $\{\boldsymbol{v}_i\}_{i=1}^n$, die eine Basis für $\mathbb{R}^n$ bilden.
- Zusätzlich gelten

$$|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n| \geq 0.$$

  d.h. λ_1 ist *dominant*.
- Ein beliebiger Startvektor für die Vektoriteration kann so dargestellt werden:

$$\boldsymbol{x}^{(0)} = \sum_{i=1}^{n} \alpha_i \boldsymbol{v}_i$$

- Angenommen gilt $\alpha_1 \neq 0$ und $\boldsymbol{x}^{(0)}$ ist so normalisiert, dass $\|\boldsymbol{x}^{(0)}\|_\infty = 1$ gilt.

Vektoriteration

- Die Vektoriteration: Für $k = 1, 2, \ldots$

$$\begin{cases} \boldsymbol{y}^{(k)} &= A\boldsymbol{x}^{(k-1)} \\ \boldsymbol{x}^{(k)} &= \boldsymbol{y}^{(k)} / y_{p_k}^{(k)} \\ \mu_k &= y_{p_{k-1}}^{(k)} \end{cases} \quad \text{wobei} \quad \begin{cases} |y_{p_k}^{(k)}| = \|\boldsymbol{y}^{(k)}\|_\infty \\ \text{d.h. } \|\boldsymbol{x}^{(k)}\|_\infty = 1 \end{cases}$$

- Behauptungen:

$$\mu_k \overset{k\to\infty}{\longrightarrow} \lambda_1 \quad \text{und} \quad \exists\{s_k = \pm 1\}_{k=0}^{\infty} \quad \text{mit} \quad s_k \boldsymbol{x}^{(k)} \overset{k\to\infty}{\longrightarrow} \frac{\boldsymbol{v}_1}{\|\boldsymbol{v}_1\|_\infty}$$

- Zuerst eine Darstellung von $\boldsymbol{x}^{(k)}$,

$$\boldsymbol{x}^{(k)} = \frac{\boldsymbol{y}^{(k)}}{y_{p_k}^{(k)}} = \frac{A\boldsymbol{x}^{(k-1)}}{y_{p_k}^{(k)}} = \frac{A\boldsymbol{y}^{(k-1)}}{y_{p_k}^{(k)} y_{p_{k-1}}^{(k-1)}} = \cdots = \frac{A^{k-1}\boldsymbol{y}^{(1)}}{y_{p_k}^{(k)} \cdots y_{p_1}^{(1)}} = \frac{A^k \boldsymbol{x}^{(0)}}{\prod_{i=1}^{k} y_{p_i}^{(i)}}$$

- Dann eine Darstellung für μ_k,

$$\mu_k = \frac{y_{p_{k-1}}^{(k)}}{x_{p_{k-1}}^{(k-1)}} = \frac{(A\boldsymbol{x}^{(k-1)})_{p_{k-1}}}{x_{p_{k-1}}^{(k-1)}} = \frac{(A^k \boldsymbol{x}^{(0)})_{p_{k-1}} / \prod_{i=1}^{k-1} y_{p_i}^{(i)}}{(A^{k-1}\boldsymbol{x}^{(0)})_{p_{k-1}} / \prod_{i=1}^{k-1} y_{p_i}^{(i)}} = \frac{(A^k \boldsymbol{x}^{(0)})_{p_{k-1}}}{(A^{k-1}\boldsymbol{x}^{(0)})_{p_{k-1}}}$$

Vektoriteration

- Weil λ_1 dominant ist, gilt

$$\mu_k = \frac{\lambda_1^k \left[\alpha_1 \boldsymbol{v}_1 + \sum_{i=2}^{n} \left(\frac{\lambda_i}{\lambda_1} \right)^k_{\downarrow 0} \boldsymbol{v}_i \right]_{p_{k-1}}}{\lambda_1^{k-1} \left[\alpha_1 \boldsymbol{v}_1 + \sum_{i=2}^{n} \left(\frac{\lambda_i}{\lambda_1} \right)^{k-1}_{\downarrow 0} \boldsymbol{v}_i \right]_{p_{k-1}}} \xrightarrow{k\to\infty} \lambda_1$$

- Wegen der obigen Darstellung von $\boldsymbol{x}^k$ gilt

$$\boldsymbol{y}^{(k)} = y^{(k)}_{p_k} \boldsymbol{x}^{(k)} = y^{(k)}_{p_k} \frac{A^k \boldsymbol{x}^{(0)}}{\prod_{i=1}^{k} y^{(i)}_{p_i}} = \underbrace{\frac{\lambda_1^k}{\prod_{i=1}^{k-1} y^{(i)}_{p_i}}}_{Q = 1/(\alpha_1 \nu_k)} \left[\alpha_1 \boldsymbol{v}_1 + \sum_{i=2}^{n} \alpha_i \left(\frac{\lambda_i}{\lambda_1} \right)^k_{\downarrow 0} \boldsymbol{v}_i \right]$$

Mit $|v_{1,p}| = \|\boldsymbol{v}_1\|_\infty$ gilt

$$\frac{|y^{(k)}_{p_k}|}{|y^{(k)}_p|} = \frac{\|\boldsymbol{y}^{(k)}\|_\infty}{|y^{(k)}_p|} = \frac{\|\nu_k \boldsymbol{y}^{(k)}\|_\infty}{|\nu_k y^{(k)}_p|} \xrightarrow{k\to\infty} \frac{\|\boldsymbol{v}_1\|_\infty}{|v_{1,p}|} = 1$$

Inverse Vektoriteration

- Also konvergieren die Vektoren $\boldsymbol{x}^{(k)}$ wie behauptet:

$$\left.\frac{y_{p_k}^{(k)}}{y_p^{(k)}}\right|_{\to\pm 1} \boldsymbol{x}^{(k)} = \frac{\boldsymbol{y}^{(k)}}{y_p^{(k)}} = \frac{A\boldsymbol{x}^{(k-1)}}{(A\boldsymbol{x}^{(k-1)})_p} = \frac{A^k\boldsymbol{x}^{(0)}/\prod_{i=1}^{k-1} y_{p_i}^{(i)}}{(A^k\boldsymbol{x}^{(0)})_p/\prod_{i=1}^{k-1} y_{p_i}^{(i)}}$$

$$= \frac{A^k\boldsymbol{x}^{(0)}}{(A^k\boldsymbol{x}^{(0)})_p} = \frac{\lambda_1^k\left[\alpha_1\boldsymbol{v}_1 + \sum_{i=2}^{n}\alpha_i\left(\frac{\lambda_i}{\lambda_1}\right)^k_{\downarrow 0}\boldsymbol{v}_i\right]}{\lambda_1^k\left[\alpha_1\boldsymbol{v}_1 + \sum_{i=2}^{n}\alpha_i\left(\frac{\lambda_i}{\lambda_1}\right)^k_{\downarrow 0}\boldsymbol{v}_i\right]_p} \overset{k\to\infty}{\to} \frac{\boldsymbol{v}_1}{v_{1,p}}$$

Inverse Vektoriteration

- Um λ_k zu berechnen, wird die Vektoriteration auf $(A - qI)^{-1}$ angewendet, wobei $|\lambda_k - q| < |\lambda_i - q|, \forall i \neq k$.
- Hausaufgabe: Es gilt $\mu_k \overset{k\to\infty}{\longrightarrow} (\lambda_k - q)^{-1}$ für die Iteration,

$$\begin{cases} (A-qI)\boldsymbol{y}^{(k)} &= \boldsymbol{x}^{(k-1)} \\ \boldsymbol{x}^{(k)} &= \boldsymbol{y}^{(k)}/y_{p_k}^{(k)} \\ \mu_k &= y_{p_{k-1}}^{(k)} \end{cases} \quad \text{wobei} \quad \begin{cases} |y_{p_k}^{(k)}| = \|\boldsymbol{y}^{(k)}\|_\infty \\ \text{d.h. } \|\boldsymbol{x}^{(k)}\|_\infty = 1 \end{cases}$$

Berechnung *aller* Eigenwerte

- Die Vektoriteration und die Inverse Vektoriteration sind nur zur Berechnung bestimmter Eigenwerte geeignet.
- Nun suchen wir Verfahren, die alle Eigenwerte und ihre entsprechenden Eigenvektoren liefern.
- Nach dem Abel-Ruffini Satz gibt es für $n \geq 5$ keine allgemeine endliche Formel mit Arithmetik und Wurzeln zur Bestimmung der Nullstellen des charakteristischen Polynoms.
- Daher müssen Eigenwerte iterativ berechnet werden.
- Der Plan vom QR-Algorithmus: $A^{(1)} = A$ (SPD)

  Für $k = 1, 2 \ldots \quad A^{(k)} = Q^{(k)} R^{(k)}, \quad A^{(k+1)} = R^{(k)} Q^{(k)}$ wobei

  $Q^{(k)}$ orthogonal ist und $R^{(k)}$ eine obere Dreiecksmatrix ist.
- Aber diese Iteration ist viel billiger, wenn zuerst so transformiert wird: $\tilde{A} = PAP^\top$, wobei P orthogonal und $\tilde{A}$ SPD tridiagonal sind.

Die Householder Methode

- Zur Bestimmung der Transformation $\tilde{A} = PAP^\top$, A SPD, wobei P orthogonal und $\tilde{A}$ SPD tridiagonal sind:
- Mit $P = P^{(n-2)} \dots P^{(1)}$, $A^{(1)} = A$, soll gelten

$$A^{(2)} = P^{(1)} A^{(1)} P^{(1)} = \begin{bmatrix} \square & \square & 0 & \cdots & 0 \\ \square & + & - & - & + \\ 0 & | & & & | \\ \vdots & | & & & | \\ 0 & + & - & - & + \end{bmatrix}$$

- und im nächsten Schritt

$$A^{(3)} = P^{(2)} A^{(2)} P^{(2)} = \begin{bmatrix} \square & \square & 0 & \cdots & 0 \\ \square & \square & \square & 0 & \cdots \\ 0 & \square & + & - & + \\ \vdots & 0 & | & & | \\ 0 & \vdots & + & - & + \end{bmatrix}$$

- usw bis $\tilde{A} = A^{(n-1)}$.

Householder Transformationen

Def: Sei $\hat{\boldsymbol{w}} \in \mathbb{R}^n$ mit $\hat{\boldsymbol{w}}^\top \hat{\boldsymbol{w}} = 1$. Dann heisst $P = I - 2\hat{\boldsymbol{w}}\hat{\boldsymbol{w}}^\top$ eine *Householder Transformation*.

Satz: Eine Householder Transformation ist symmetrisch und orthogonal.

$$P^\top P = P^2 = (I - 2\hat{\boldsymbol{w}}\hat{\boldsymbol{w}}^\top)(I - 2\hat{\boldsymbol{w}}\hat{\boldsymbol{w}}^\top) = I - 4\hat{\boldsymbol{w}}\hat{\boldsymbol{w}}^\top + 4\hat{\boldsymbol{w}}\hat{\boldsymbol{w}}^\top\hat{\boldsymbol{w}}\hat{\boldsymbol{w}}^\top = I$$

Satz: Seien $0 \neq \boldsymbol{u} \in \mathbb{R}^n$ und $\theta = \frac{1}{2}\|\boldsymbol{u}\|_2^2$. Dann ist $P = I - \boldsymbol{u}\boldsymbol{u}^\top/\theta$ eine Householder Transformation.

$$\hat{\boldsymbol{w}} = \boldsymbol{u}/\|\boldsymbol{u}\|_2 \Rightarrow P = I - 2\hat{\boldsymbol{w}}^\top\hat{\boldsymbol{w}}$$

Satz: Seien $\boldsymbol{x} \in \mathbb{R}^n$ und $\sigma = \pm\|\boldsymbol{x}\|_2$. Angenommen $\boldsymbol{x} \neq -\sigma\hat{\boldsymbol{e}}_1$. Seien $\boldsymbol{u} = \boldsymbol{x} + \sigma\hat{\boldsymbol{e}}_1$ und $\theta = \frac{1}{2}\|\boldsymbol{u}\|_2^2$. Dann ist $P = I - \boldsymbol{u}\boldsymbol{u}^\top/\theta$ eine Householder Transformation und es gilt $P\boldsymbol{x} = -\sigma\hat{\boldsymbol{e}}_1$.

Beweis: $\boldsymbol{x} \neq -\sigma\hat{\boldsymbol{e}}_1 \Rightarrow \boldsymbol{u} = \boldsymbol{x} + \sigma\hat{\boldsymbol{e}}_1 \neq 0$. Nach dem obigen Satz ist P eine Householder Transformation. Dann gilt

Householder Transformationen

$$\begin{aligned}\theta = \tfrac{1}{2}\|\boldsymbol{u}\|_2^2 &= \tfrac{1}{2}(\boldsymbol{x}+\sigma\hat{\boldsymbol{e}}_1)^\top(\boldsymbol{x}+\sigma\hat{\boldsymbol{e}}_1)\\ &= \tfrac{1}{2}(\boldsymbol{x}^\top\boldsymbol{x}+2\sigma\boldsymbol{x}^\top\hat{\boldsymbol{e}}^{(1)}+\sigma^2) \quad (\sigma^2=\|\boldsymbol{x}\|_2^2)\\ &= \tfrac{1}{2}(\sigma^2+2\sigma\boldsymbol{x}^\top\hat{\boldsymbol{e}}_1+\sigma^2) = \sigma^2+\sigma x_1\end{aligned}$$

Daher gilt

$$\begin{aligned}(I-\boldsymbol{u}\boldsymbol{u}^\top/\theta)\boldsymbol{x} &= \boldsymbol{x}-\boldsymbol{u}\boldsymbol{u}^\top\boldsymbol{x}/\theta\\ &= \boldsymbol{x}-(\boldsymbol{x}+\sigma\hat{\boldsymbol{e}}_1)\frac{(\boldsymbol{x}+\sigma\hat{\boldsymbol{e}}_1)^\top\boldsymbol{x}}{\sigma^2+\sigma x_1}\\ &= \boldsymbol{x}-(\boldsymbol{x}+\sigma\hat{\boldsymbol{e}}_1\frac{(\boldsymbol{x}^\top\boldsymbol{x}+\sigma x_1)}{\sigma^2+\sigma x_1} = -\sigma\hat{\boldsymbol{e}}_1 \quad \blacksquare\end{aligned}$$

Bemerkung: Um Auslöschung zu vermeiden, soll σ mit dem gleichen Vorzeichen wie x_1 ausgewählt werden. P wird auch nicht verändert, wenn $\boldsymbol{x}$ zur Sicherheit skaliert wird.

Algorithmus:

$$\begin{aligned}\xi &= \|\boldsymbol{x}\|_\infty & \theta &= \rho(v_1+\rho)\\ \eta &= \frac{(\xi\not\approx 0)}{\xi+(\xi\approx 0)} & \psi &= \frac{(\theta\not\approx 0)}{\sqrt{\theta}+(\theta\approx 0)}\\ \boldsymbol{v} &= \eta\boldsymbol{x} & \boldsymbol{u} &= \psi(\boldsymbol{v}+\rho\hat{\boldsymbol{e}}_1)\\ \rho &= \mathrm{s}(v_1)\|\boldsymbol{v}\|_2 & \sigma &= \xi\rho\end{aligned}$$

$$\mathrm{s}(t) = \begin{cases} 1, & t \geq 0\\ -1, & \text{sonst}\end{cases}$$

$$P = I - \boldsymbol{u}\boldsymbol{u}^\top$$

$$P\boldsymbol{x} = -\sigma\hat{\boldsymbol{e}}_1$$

Hausaufgabe: $\|\boldsymbol{u}\|_2^2 = 2$

Transformation auf Tridiagonale Form

- Zur Bestimmung von $P^{(1)}$ sei

$$\boldsymbol{a}^{(1)} = \begin{bmatrix} a_{11}^{(1)} \\ a_{21}^{(1)} \\ \vdots \\ a_{n1}^{(1)} \end{bmatrix} = \begin{bmatrix} a_{11}^{(1)} \\ \\ \tilde{\boldsymbol{a}}^{(1)} \\ \\ \end{bmatrix}$$

$$(\tilde{\boldsymbol{u}}, \sigma) = \text{Householder}(\tilde{\boldsymbol{a}}^{(1)})$$

106

$$\tilde{P} = I - \tilde{\boldsymbol{u}}^\top \tilde{\boldsymbol{u}}, \quad \tilde{\boldsymbol{u}} \in \mathbb{R}^{n-1}$$

wobei $\tilde{P}\tilde{\boldsymbol{a}}^{(1)} = -\sigma \hat{\boldsymbol{e}}_1 \in \mathbb{R}^{n-1}$

- Dann

$$P = I - \boldsymbol{u}^\top \boldsymbol{u}, \quad \boldsymbol{u} = \begin{bmatrix} 0 \\ \tilde{\boldsymbol{u}} \end{bmatrix} \in \mathbb{R}^n$$

erfüllt

$$P\boldsymbol{a}^{(1)} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & + & - & + \\ \vdots & | & \tilde{P} & | \\ 0 & + & - & + \end{bmatrix} \begin{bmatrix} a_{11}^{(1)} \\ \\ \tilde{\boldsymbol{a}}^{(1)} \\ \\ \end{bmatrix} = \begin{bmatrix} a_{11}^{(1)} \\ -\sigma \\ 0 \\ \vdots \end{bmatrix}$$

Transformation auf Tridiagonale Form

- Zur Bestimmung von $P^{(2)}$ sei

$$\boldsymbol{a} = \begin{bmatrix} a_{12}^{(2)} \\ a_{22}^{(2)} \\ \vdots \\ a_{n2}^{(2)} \end{bmatrix} = \begin{bmatrix} a_{12}^{(2)} \\ a_{22}^{(2)} \\ \tilde{\boldsymbol{a}}^{(2)} \\ \\ \end{bmatrix}$$

$$(\tilde{\boldsymbol{u}}, \sigma) = \text{Householder}(\tilde{\boldsymbol{a}}^{(2)})$$

106

$$\tilde{P} = I - \tilde{\boldsymbol{u}}^\top \tilde{\boldsymbol{u}}, \quad \tilde{\boldsymbol{u}} \in \mathbb{R}^{n-2}$$

wobei $\tilde{P}\tilde{\boldsymbol{a}}^{(2)} = -\sigma \hat{\boldsymbol{e}}_1 \in \mathbb{R}^{n-2}$

- Dann

$$P = I - \boldsymbol{u}^\top \boldsymbol{u}, \quad \boldsymbol{u} = \begin{bmatrix} 0 \\ 0 \\ \tilde{\boldsymbol{u}} \end{bmatrix} \in \mathbb{R}^n$$

erfüllt

$$P\boldsymbol{a}^{(2)} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & + & - & + \\ \vdots & \vdots & | & \tilde{P} & | \\ 0 & 0 & + & - & + \end{bmatrix} \begin{bmatrix} a_{12}^{(2)} \\ a_{22}^{(2)} \\ \\ \tilde{\boldsymbol{a}}^{(2)} \\ \\ \end{bmatrix} = \begin{bmatrix} a_{12}^{(2)} \\ a_{22}^{(2)} \\ -\sigma \\ 0 \\ \vdots \end{bmatrix}$$

Hessenberg Matrizen

- Dann sehen die Matrizen $P^{(k)} \dots P^{(1)} A$ so aus:

$$P^{(1)}A = \begin{bmatrix} \square & + & - & - & + \\ \square & | & & & | \\ 0 & | & & & | \\ \vdots & | & & & | \\ 0 & + & - & - & + \end{bmatrix} \qquad P^{(2)}P^{(1)}A = \begin{bmatrix} \square & \square & \cdots & \cdots & \square \\ \square & \square & + & - & + \\ 0 & \square & | & & | \\ \vdots & 0 & | & & | \\ 0 & \vdots & + & - & + \end{bmatrix}$$

- Und $P^{(n-2)} \dots P^{(1)} A$ ist eine *obere Hessenberg* Matrix:

$$P^{(n-2)} \dots P^{(1)}A = \begin{bmatrix} \square & \cdots & \cdots & \cdots & \square \\ \square & \ddots & & & \vdots \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \square \\ 0 & \cdots & 0 & \square & \square \end{bmatrix}$$

Transformation auf Tridiagonale Form

- Wenn $A^{(1)} = A$ SPD ist, gilt $A^{(2)} = P^{(1)} A^{(1)} P^{(1)} =$

$$\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & + & - & + \\ \vdots & | & \tilde{P} & | \\ 0 & + & - & + \end{bmatrix} \begin{bmatrix} a_{11}^{(1)} & & \tilde{\boldsymbol{a}}^{(1)\top} & \\ & + & - & + \\ \tilde{\boldsymbol{a}}^{(1)} & | & \tilde{A} & | \\ & + & - & + \end{bmatrix} \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & + & - & + \\ \vdots & | & \tilde{P} & | \\ 0 & + & - & + \end{bmatrix}$$

$$= \begin{bmatrix} a_{11}^{(1)} & -\sigma & 0 & \cdots \\ -\sigma & + & - & + \\ 0 & | & \tilde{P}\tilde{A}\tilde{P} & | \\ \vdots & + & - & + \end{bmatrix}$$

- $P^{(2)}$ wird von $A^{(2)} = P^{(1)} A^{(1)} P^{(1)}$ wie vorher bestimmt, und $A^{(3)} = P^{(2)} A^{(2)} P^{(2)}$ hat zwei reduzierte Spalten und Zeilen.
- Achtung: Die Matrix $P^{(1)} A^{(1)} P^{(1)}$ hat eine andere zweite Spalte als $P^{(1)} A^{(1)}$, und $P^{(2)}$ wird entsprechend beeinflusst.
- Wenn $P^{(k+1)}$ durch Householder von $A^{(k+1)} = P^{(k)} A^{(k)} P^{(k)}$ bestimmt wird, ist $PAP^\top$ tridiagonal mit $P = P^{(n-2)} \dots P^{(1)}$.

Pseudo-Code zur Transformation auf Hessenberg oder Tridiagonale Form

```
H = A, Q = I, ASym = (A == A')
for k=1,...,n-2
   h(i) = H(k+i,k), i=1,...,n-k
   (u,sg) = Householder(h)  106          % (I-uu')h=-sg e1
   for l=k,...,n                 % H <- (I-ũũ')H, H = ...P2 P1 A
      sm = sum(H(k+i,l)*u(i):  i=1,...,n-k)
      H(k+i,l) = H(k+i,l) - sm*u(i), i=1,...,n-k
   end
   for l=1,...,n                 % Q <- Q(I-ũũ'), Q = I P1 P2 ...
      sm = sum(Q(l,k+i)*u(i):  i=1,...,n-k)
      Q(l,k+i) = Q(l,k+i) - sm*u(i), i=1,...,n-k
   end
   If ASym
      for l=k,...,n              % H <- H(I-ũũ'), H = ...P2 P1 A P1 P2...
         sm = sum(H(l,k+i)*u(i):  i=1,...,n-k)
         H(l,k+i) = H(l,k+i) - sm*u(i), i=1,...,n-k
      end
   end
end
% ASym = false => A=Q*H, ASym = true => A=Q*H*Q'
```

QR Algorithmus

- Für die Iteration, $A^{(k)} = Q^{(k)} R^{(k)}$, $A^{(k+1)} = R^{(k)} Q^{(k)}$, gilt

$$A^{(k+1)} = R^{(k)} Q^{(k)} = Q^{(k)\top} A^{(k)} Q^{(k)} = \cdots = \underbrace{Q^{(k)\top} \cdots Q^{(1)\top}}_{Q^\top} A \underbrace{Q^{(1)} \cdots Q^{(k)}}_{Q}$$

  also sind die Matrizen $\{A^{(k)}\}$ alle ähnlich zu A.
- Unter gewissen unten genannten Bedingungen gilt

$$A^{(k)} \overset{k}{\longrightarrow} \infty \; \Lambda = \operatorname{diag}(\sigma(A)).$$

- Der erste Bedarf ist, ein Algorithmus zur *QR*-Zerlegung.
- Householder Transformationen passen: $Q^{(1)}$ wird so konstruiert,

$$\boldsymbol{a}^{(1)} = \begin{bmatrix} a_{11}^{(1)} \\ a_{21}^{(1)} \\ \vdots \\ a_{n1}^{(1)} \end{bmatrix}$$

$$(\boldsymbol{u}, \sigma) = \text{Householder}(\boldsymbol{a}^{(1)})$$

106

$$Q^{(1)} = I - \boldsymbol{u}^\top \boldsymbol{u}, \quad \boldsymbol{u} \in \mathbb{R}^n$$

wobei $Q^{(1)} \boldsymbol{a}^{(1)} = -\sigma \hat{\boldsymbol{e}}_1 \in \mathbb{R}^n$

QR Algorithmus

- $Q^{(2)}$ wird so konstruiert,

$$\boldsymbol{a}^{(2)} = \begin{bmatrix} a_{12}^{(2)} \\ a_{22}^{(2)} \\ \vdots \\ a_{n2}^{(2)} \end{bmatrix} = \begin{bmatrix} a_{12}^{(2)} \\ \\ \tilde{\boldsymbol{a}}^{(2)} \\ \\ \end{bmatrix}$$

$$(\tilde{\boldsymbol{u}}, \sigma) = \text{Householder}(\tilde{\boldsymbol{a}}^{(2)})$$

106

$$\tilde{Q}^{(2)} = I - \tilde{\boldsymbol{u}}^\top \tilde{\boldsymbol{u}}, \quad \tilde{\boldsymbol{u}} \in \mathbb{R}^{n-1}$$

$$\text{wobei } \tilde{Q}^{(2)} \tilde{\boldsymbol{a}}^{(2)} = -\sigma \hat{\boldsymbol{e}}_1 \in \mathbb{R}^{n-1}$$

$$Q^{(2)} = I - \boldsymbol{u}^\top \boldsymbol{u}, \quad \boldsymbol{u} = \begin{bmatrix} 0 \\ \tilde{\boldsymbol{u}} \end{bmatrix}$$

- Es gelten

$$Q^{(1)}A = \begin{bmatrix} -\sigma_1 & + & - & - & + \\ 0 & | & & & | \\ \vdots & | & & & | \\ 0 & + & - & - & + \end{bmatrix} \qquad Q^{(2)}Q^{(1)}A = \begin{bmatrix} -\sigma_1 & \square & + & - & + \\ 0 & -\sigma_2 & | & & | \\ \vdots & 0 & | & & | \\ 0 & \vdots & + & - & + \end{bmatrix}$$

- Nach $n-1$ solchen Transformationen ergibt sich eine obere Dreiecksmatrix,

$$Q^\top A = Q^{(n-1)} \dots Q^{(1)} A = R$$

Givens Transformationen

- Aber wenn A schon tridiagonal ist, kann Q billiger konstruiert werden:

$$A = \begin{bmatrix} a_1 & b_1 & 0 & \cdots & 0 \\ b_1 & a_2 & b_2 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & b_{n-2} & a_{n-1} & b_{n-1} \\ 0 & \cdots & 0 & b_{n-1} & a_n \end{bmatrix}$$

Def: Eine *Givens Transformation* $G = G^{ij}(\theta)$ hat die Form

$$GG^\top = I, \quad G = \begin{bmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & & & & \\ & & & q_{ii} & & q_{ij} & \\ & & & & 1 & & \\ & & & & & \ddots & \\ & & & & & & 1 \\ & & & q_{ji} & & q_{jj} & \\ & & & & & & & 1 \\ & & & & & & & & \ddots \\ & & & & & & & & & 1 \end{bmatrix}$$

$$q_{ii} = \cos(\theta) = q_{jj}$$
$$q_{ij} = \sin(\theta) = -q_{ji}$$
für gegebene
i, j und θ

Givens Transformationen

- Für $x_1 = a_1$, $y_1 = b_1$, soll θ so ausgewählt werden, dass

$$\begin{bmatrix} \cos(\theta) & \sin(\theta) & & \\ -\sin(\theta) & \cos(\theta) & & \\ & & 1 & \\ & & & \ddots\ 1 \end{bmatrix} \begin{bmatrix} x_1 & y_1 & 0 & \\ b_1 & a_2 & b_2 & \\ & \ddots & \ddots & \ddots \\ & & b_{n-1} & a_n \end{bmatrix} = \begin{bmatrix} g_1 & d_1 & e_1 & & \\ 0 & x_2 & y_2 & & \\ & b_2 & a_3 & b_3 & \\ & & \ddots & \ddots & \ddots \\ & & & b_{n-1} & a_n \end{bmatrix}$$

- Mit $s_1 = \sin(\theta_1)$, $c_1 = \cos(\theta_1)$,

$$\left.\begin{aligned} -s_1 x_1 + c_1 b_1 &= 0 \\ s_1^2 + c_1^2 &= 1 \end{aligned}\right\} \quad \begin{aligned} s_1 &= b_1 / \sqrt{b_1^2 + x_1^2} \\ c_1 &= x_1 / \sqrt{b_1^2 + x_1^2} \end{aligned}$$

- Hausaufgabe: Für $g_1 > 0$ ist Q eindeutig bestimmt.

Die QR-Zerlegung

- Initialisierung: $R \leftarrow A$, $Q \leftarrow I$.
- Für $k = 1, \ldots, n-1$,

$$Q \leftarrow G^{k,k+1}(\theta_k) Q, \quad R \leftarrow G^{k,k+1}(\theta_k) R$$

wobei $\sin(\theta_k) = b_k / \sqrt{b_k^2 + x_k^2}$, $\cos(\theta_k) = x_k / \sqrt{b_k^2 + x_k^2}$.

- Es gilt

$$\begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & & \cos(\theta_k) & \sin(\theta_k) & \\ & & -\sin(\theta_k) & \cos(\theta_k) & \\ & & & & 1 \\ & & & & & \ddots \\ & & & & & & 1 \end{bmatrix} \begin{bmatrix} \ddots & \ddots & \ddots & & \\ & g_{k-1} & d_{k-1} & e_{k-1} & \\ & 0 & x_k & y_k & 0 \\ & & b_k & a_{k+1} & b_{k+1} \\ & & & \ddots & \ddots & \ddots \end{bmatrix}$$

$$= \begin{bmatrix} \ddots & \ddots & \ddots & & \\ & g_k & d_k & e_k & \\ & 0 & x_{k+1} & y_{k+1} & 0 \\ & & b_{k+1} & a_{k+2} & b_{k+2} \\ & & & \ddots & \ddots & \ddots \end{bmatrix}$$

- Am Ende setze $Q \leftarrow Q^\top$ und es gilt $A = QR$. Eindeutig?

QR Algorithmus

Satz: Sei $A \in \mathbb{R}^{n \times n}$ mit $\text{Rang}(A) = n$. Dann gibt es eine Zerlegung $A = QR$, wobei Q orthogonal ist und R eine obere Dreiecksmatrix ist. Weiters ist die Zerlegung eindeutig, wenn die diagonalen Einträge von R positiv sind.

Bemerkung: Für die Konvergenz des QR-Algorithmus zur Bestimmung der Eigenräume von A ist diese eindeutige Zerlegung nicht notwendig.

- Nun haben wir *eine* QR-Zerlegung $A^{(k)} = Q^{(k)} R^{(k)}$.
- Zu zeigen ist, dass

$$A^{(k+1)} = R^{(k)} Q^{(k)} = Q^{(k)\top} A^{(k)} Q^{(k)}$$

  wieder tridiagonal ist.
- Zuerst wird gezeigt, $Q^{(k)}$ ist obere Hessenberg.

Givens-Kette ist Hessenberg

- Bemerke:

$$\begin{bmatrix} 1 & & & & \\ & c_2 & s_2 & & \\ & -s_2 & c_2 & & \\ & & & 1 & \\ & & & & \ddots \\ & & & & & 1 \end{bmatrix} \begin{bmatrix} c_1 & s_1 & & & \\ -s_1 & c_1 & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{bmatrix} = \begin{bmatrix} c_1 & s_1 & & & \\ \square & \square & \square & & \\ \square & \square & \square & & \\ & & & 1 & \\ & & & & \ddots \\ & & & & & 1 \end{bmatrix}$$

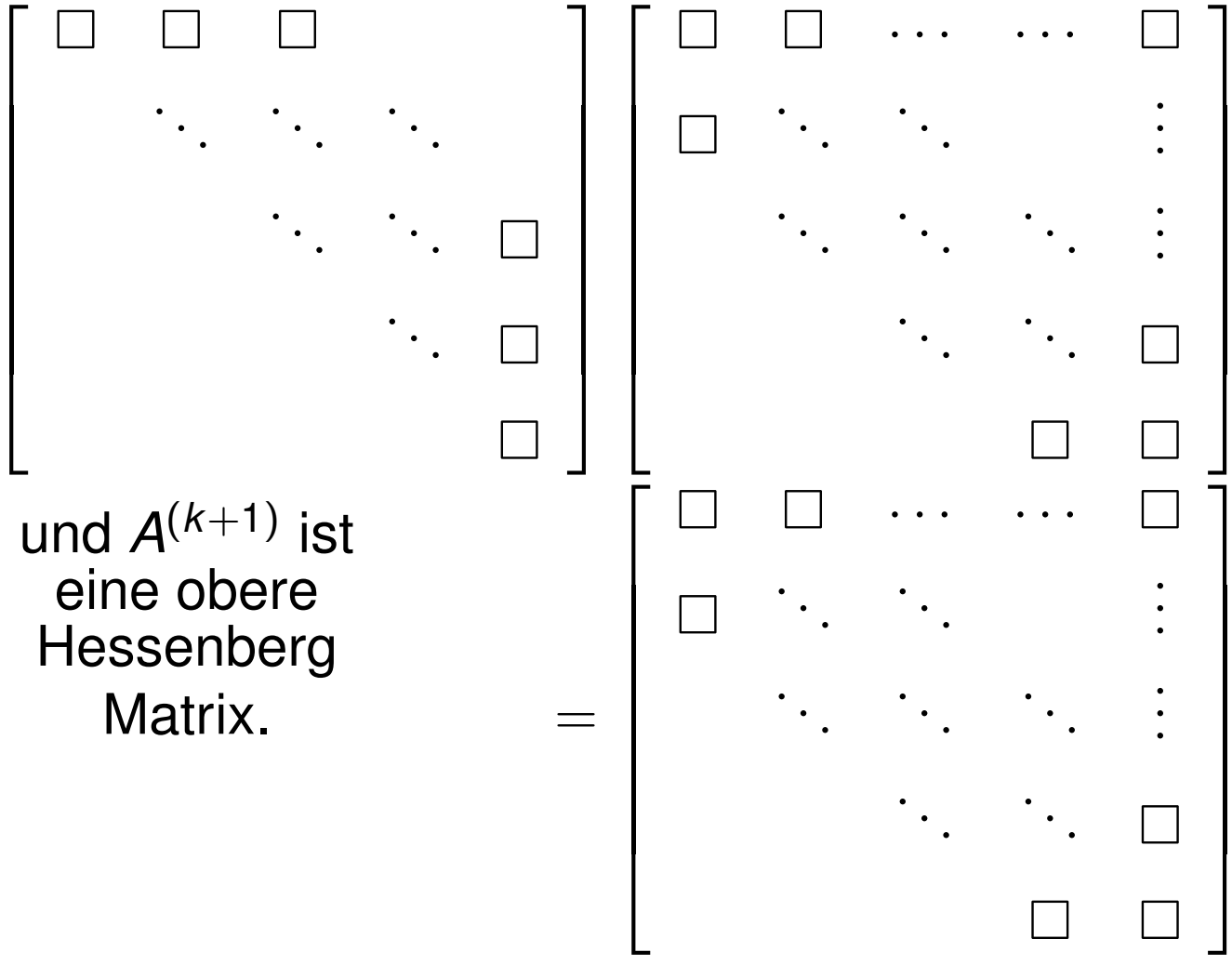
- Induktiv:

$$\begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & c_k & s_k & \\ & & -s_k & c_k & \\ & & & & \ddots \\ & & & & & 1 \end{bmatrix} \begin{bmatrix} \square & \square & & & \\ \square & \ddots & \ddots & & \\ \vdots & \ddots & \ddots & \ddots & \\ \vdots & & \ddots & \ddots & \square \\ \square & \cdots & \cdots & \square & \square \\ & & & & & 1 \\ & & & & & & \ddots \\ & & & & & & & 1 \end{bmatrix} = \begin{bmatrix} \square & \square & & & & \\ \square & \ddots & \ddots & & & \\ \vdots & \ddots & \ddots & \ddots & & \\ \vdots & & \ddots & \ddots & \square & \\ \square & \cdots & \cdots & \square & \square & \\ & & & & & \ddots \\ & & & & & & 1 \end{bmatrix}$$

- Also ist $G^{k,k+1}(\theta_k) \cdots G^{1,2}(\theta_1)$ untere Hessenberg.
- Daher ist $Q^{(k)}$ obere Hessenberg.
- $R^{(k)} (= \operatorname{diag}\{\boldsymbol{g}, \boldsymbol{d}, \boldsymbol{e}\})$ hat drei obere Diagonale.

Tridiagonale Form Bleibt Erhalten

- Daher haben wir folgende Struktur für $A^{(k+1)} = R^{(k)} Q^{(k)} =$



und $A^{(k+1)}$ ist eine obere Hessenberg Matrix.

- Aber $A^{(k+1)} = Q^{(k)\top} A^{(k)} Q^{(k)}$ ist symmetrisch und deswegen tridiagonal!

Konvergenz des *QR*-Algorithmus

Satz: Sei $A \in \mathbb{R}^{n\times n}$ symmetrisch mit Eigenwerten $\{\lambda_i\}_{i=1}^n$, die erfüllen $|\lambda_1| > |\lambda_2| > \cdots > |\lambda_n| > 0$. Seien $\{\boldsymbol{v}_i\}_{i=1}^n$ die entsprechenden Eigenvektoren, die eine orthonormale Basis bilden. Für die Matrix $Q = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n]$ soll es eine LU-Zerlegung $Q^\top = LU$ geben, ohne dass $Q^\top$ mit einer Permutationsmatrix multipliziert werden muss. Dann mit $A^{(1)} = A$ und einer (beliebigen) QR-Zerlegung $A^{(k)} = Q^{(k)}R^{(k)}$, konvergiert die Folge $A^{(k+1)} = R^{(k)}Q^{(k)}$ zur diagonalen Matrix $\Lambda = \mathrm{diag}(\{\lambda_i\}_{i=1}^n)$.

Bemerkung: Mit $A^{(k)} = \{a_{ij}^{(k)}\}$ gilt

$$|a_{i+1,i}^{(k)}| = \mathcal{O}\left(\left|\frac{\lambda_{i+1}}{\lambda_i}\right|^k\right) \quad \text{oder} \quad |a_{i+1,i}^{(k)}| = \mathcal{O}\left(\left|\frac{\lambda_{i+1}-q}{\lambda_i - q}\right|^k\right)$$

wenn man eine Verschiebung macht,

$$A^{(k)} - qI = Q^{(k)}R^{(k)}, \quad A^{(k+1)} = R^{(k)}Q^{(k)} + qI$$

Pseudo-Code für den *QR*-Algorithmus

```
a(i) = H(i,i), b(i) = H(i,i+1) = H(i+1,i), i=1,...,n-1
for k=1,...,kmax
   x(1) = a(1), y(1) = b(1)
   for j=1,...,n-1     % compute R, Q (storing c's & s's)
      g(j) = sqrt(x(j)^2 + b(j)^2)
      c(j) = x(j)/g(j), s(j) = b(j)/g(j)
      d(j) = c(j)*y(j) + s(j)*a(j+1)
      x(j+1) = c(j)*a(j+1) - s(j)*y(j)
      if j < (n-1)
         e(j) = s(j)*b(j+1), y(j+1) = c(j)*b(j+1)
      end
   end
   g(n) = x(n)
   a(1) = s(1)*d(1) + c(1)*g(1), b(1) = s(1)*g(2)
   for j=2,...,n-1     % compute R * Q, overwriting H
      a(j) = s(j)*d(j) + c(j)*c(j-1)*g(j)
      b(j) = s(j)*g(j+1)
   end
   a(n) = c(n-1)*g(n)
   if (||b|| < tol*||a||) then break
end
% a(i), i=1,...,n ≈ Eigenwerte von H
```

Ausgleichsprobleme

Beispiel (Lineare Regression): Gegeben seien $\{(x_i, y_i)\}_{i=1}^m$.
Was ist die *beste* Gerade durch diese Daten?

- Es soll minimiert werden: $\boldsymbol{x} = \{x_i\}_{i=1}^m \quad \boldsymbol{y} = \{y_i\}_{i=1}^m$

$$
\begin{aligned}
E(a,b) &= \sum_{i=1}^{m} [y_i - (a + bx_i)]^2 \qquad \boldsymbol{c} = (a,b)^\top \quad V = [\mathbf{1}, \boldsymbol{x}] \\
&= \|\boldsymbol{y} - V\boldsymbol{c}\|_2^2 = (\boldsymbol{y} - V\boldsymbol{c})^\top (\boldsymbol{y} - V\boldsymbol{c}) \\
&= \boldsymbol{c}^\top V^\top V \boldsymbol{c} - 2\boldsymbol{c}^\top V^\top \boldsymbol{y} + \boldsymbol{y}^\top \boldsymbol{y}
\end{aligned}
$$

- $E(\boldsymbol{c}^\star) = \min \Rightarrow 0 = \nabla_{\boldsymbol{c}} E(\boldsymbol{c}^\star) = 2V^\top V \boldsymbol{c}^\star - 2V^\top \boldsymbol{y}$
- Hausaufgabe: $\{x_i\}_{i=1}^m$ verschieden $\Rightarrow V^\top V$ ist SPD.
- Wenn $\boldsymbol{c}^\star$ die *normalen Gleichungen* löst,

$$V^\top V \boldsymbol{c} = V^\top \boldsymbol{y}$$

folgt $E(\boldsymbol{c}^\star) = \min$.

Lineare und Polynomiale Regression

- Hausaufgabe: Zeige, die Lösung für lineare Regression ist explizit so gegeben:

$$a^\star = \overline{\boldsymbol{y}} - b^\star \overline{\mathbf{x}}, \quad b^\star = \frac{\overline{\boldsymbol{xy}} - \overline{\boldsymbol{x}} \cdot \overline{\boldsymbol{y}}}{\overline{\boldsymbol{x}^2} - \overline{\boldsymbol{x}}^2} \quad \text{wobei z.B. } \overline{\boldsymbol{xy}} = \frac{1}{m} \sum_{i=1}^{m} x_i y_i$$

- Polynomiale Regression: $\boldsymbol{x}^k = \{x_i^k\}_{i=1}^m$, $\boldsymbol{c} = \{c_k\}_{k=0}^{n-1}$,

$$\begin{aligned} E(\boldsymbol{c}) &= \sum_{i=1}^{m} [y_i - P(x_i; \boldsymbol{c})]^2 \qquad & P(x; \boldsymbol{c}) = \sum_{k=0}^{n-1} c_k x^k \\ &= \|\boldsymbol{y} - V\boldsymbol{c}\|_2^2 & V = [\mathbf{1}, \boldsymbol{x}, \ldots, \boldsymbol{x}^{n-1}] \\ &= \boldsymbol{c}^\top V^\top V \boldsymbol{c} - 2\boldsymbol{c}^\top V^\top \boldsymbol{y} + \boldsymbol{y}^\top \boldsymbol{y} \end{aligned}$$

- V ist die *Vandermonde* Matrix.
- Lösung gegeben durch die normalen Gleichungen,

$$V^\top V \boldsymbol{c} = V^\top \boldsymbol{y}$$

aber $V^\top V$ kann für hohes n schlecht konditioniert sein.

Singulärwert Zerlegung

- Ein alternativer Ansatz ist durch die *Singulärwert Zerlegung* gegeben.

Satz (SWZ): Für $A \in \mathbb{R}^{m\times n}$ $\exists$ orthogonale Matrizen $U \in \mathbb{R}^{m\times m}$, $V \in \mathbb{R}^{n\times n}$ sodass $U^\top AV = \Sigma = \text{diag}\{\sigma_1, \ldots, \sigma_p\} \in \mathbb{R}^{m\times n}$ wobei $p = \min\{m, n\}$ und $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0$.

- Also $U^\top AV = \Sigma \Rightarrow A = UU^\top AVV^\top = U\Sigma V^\top$.

Def: $A = U\Sigma V^\top$ ist die Singulärwert Zerlegung (SWZ) von A, und $\{\sigma_i\}_{i=1}^p$ sind die Singulärwerte von A.

Satz: Sei $A = U\Sigma V^\top$ die SWZ für $A \in \mathbb{R}^{m\times n}$, wobei $m > n$ und $\sigma_1 \geq \cdots \geq \sigma_r > \sigma_{r+1} = \cdots = \sigma_n = 0$ gelten. Dann wird $\|A\boldsymbol{x} - \boldsymbol{b}\|_2$ durch $\boldsymbol{x}^\star = V\Sigma^\dagger U^\top \boldsymbol{b} \in \mathbb{R}^n$ minimiert, wobei $\Sigma^\dagger = \text{diag}\{\sigma_1^{-1}, \ldots, \sigma_r^{-1}, 0, \ldots, 0\} \in \mathbb{R}^{n\times m}$. Wenn $\tilde{\boldsymbol{x}} \in \mathbb{R}^n$ auch minimierend ist und $\tilde{\boldsymbol{x}} \neq \boldsymbol{x}^\star$ gilt, dann gilt $\|\boldsymbol{x}^\star\|_2 < \|\tilde{\boldsymbol{x}}\|_2$.

Eigenschaften der SWZ

Satz (Eigenschaften der SWZ): Sei $A = U\Sigma V^\top$ die SWZ für $A \in \mathbb{R}^{m\times n}$ mit $p = \min\{m, n\}$. Es gelten:

- $U = \{\boldsymbol{u}^1, \ldots, \boldsymbol{u}^m\}$, $V = \{\boldsymbol{v}^1, \ldots, \boldsymbol{v}^n\} \Rightarrow$
  $A\boldsymbol{v}^i = \sigma_i \boldsymbol{u}^i$, $A^\top \boldsymbol{u}^i = \sigma_i \boldsymbol{v}^i$, $i = 1, \ldots, p$.
- $\sigma_1 \geq \cdots \geq \sigma_r > \sigma_{r+1} = \cdots = \sigma_n = 0 \Rightarrow$
  $\mathcal{N}(A) = \operatorname{span}\{\boldsymbol{v}^i\}_{i=r+1}^n$, $\mathcal{R}(A) = \operatorname{span}\{\boldsymbol{u}^i\}_{i=1}^r$.
- $\|A\|_2 = \sigma_1$.
- $\|A\|_{\mathrm{F}}^2 = \sigma_1^2 + \cdots + \sigma_p^2$.
- $\kappa_\star(A) = \max\limits_{\|\boldsymbol{x}\|_\star=1} \|A\boldsymbol{x}\|_\star / \min\limits_{\|\boldsymbol{x}\|_\star=1} \|A\boldsymbol{x}\|_\star \overset{m\geq n}{\Rightarrow} \kappa_2(A) = \sigma_1/\sigma_n$.
- $A^\top A = V\Sigma^\top \Sigma V^\top$ und $AA^\top = U\Sigma\Sigma^\top U^\top$.
- $\{\sigma_i^2\}_{i=1}^p$ sind die eindeutigen Nullstellen von $\det(\lambda I - A^\top A)$ für $p = n$ oder die eindeutigen Nullstellen von $\det(\lambda I - AA^\top)$ für $p = m$.

Pseudoinverse

Def: Sei $A = U\Sigma V^\top$ die SWZ für $A \in \mathbb{R}^{m\times n}$. Dann ist ist $A^\dagger = V\Sigma^\dagger U^\top$ die *Pseudo-Inverse* von A.

Satz: Für $A \in \mathbb{R}^{m\times n}$, $\mathcal{N}(A) = 0 \Rightarrow A^\dagger = (A^\top A)^{-1}A^\top$.

Beweis: Es gilt $\boldsymbol{x}^\top A^\top A\boldsymbol{x} = \|A\boldsymbol{x}\|_2^2 \geq 0$ und zusätzlich folgt $\|A\boldsymbol{x}\|_2^2 = 0 \Leftrightarrow \boldsymbol{x} = 0$ sowohl $m \geq n$ aus $\mathcal{N}(A) = 0$. Daher ist $A^\top A$ SPD. Aus $A^\top A = (U\Sigma V^\top)^\top(U\Sigma V^\top) = V\Sigma^\top U^\top U\Sigma V^\top = V\Sigma^\top\Sigma V^\top$ folgt, dass die positiven Eigenwerte von $A^\top A$ durch $0 < \Sigma^\top\Sigma = \mathrm{diag}\{\sigma_1^2, \ldots, \sigma_n^2\} \in \mathbb{R}^{n\times n}$ gegeben sind. Es folgt $(A^\top A)^{-1}A^\top = V[\Sigma^\top\Sigma]^{-1}V^\top V\Sigma^\top U^\top = V[\Sigma^\top\Sigma]^{-1}\Sigma^\top U^\top = V\Sigma^\dagger U^\top = A^\dagger$. ■

Satz: Für $A \in \mathbb{R}^{m\times n}$ und orthogonale Matrizen $P \in \mathbb{R}^{m\times m}$ und $Q \in \mathbb{R}^{n\times n}$ haben A und PAQ die gleichen Singulärwerte.

Beweis: Für $m > n$ sind die Eigenwerte $\{\sigma_i^2\}_{i=1}^n$ von $A^\top A = V\Sigma^\top\Sigma V^\top$ die Eigenwerte von $(PAQ)^\top(PAQ) = (Q^\top V)\Sigma^\top\Sigma(V^\top Q)$. Für $n > m$ sind die Eigenwerte $\{\sigma_i^2\}_{i=1}^m$ von $AA^\top = U\Sigma\Sigma^\top U^\top$ die Eigenwerte von $(PAQ)^\top(PAQ) = (Q^\top U)\Sigma\Sigma^\top(U^\top Q)$. ■

Konditionierung und die SWZ

- Für $A \in \mathbb{R}^{m\times n}$, $m > n$, mit SWZ, $A = U^\top \Sigma V$,
$$\Sigma = \begin{pmatrix} \hat{\Sigma} \\ 0 \end{pmatrix} \in \mathbb{R}^{m\times n}, \quad \hat{\Sigma} = \mathrm{diag}\{\sigma_i\}_{i=1}^n, \quad \sigma_i > 0$$
sei die Projektion $\Pi = \Sigma\Sigma^\dagger$ von $\mathbb{R}^m$ in $\mathbb{R}^n$ definiert.
- Für ein gegebenes $\boldsymbol{b}$ sei die Zerlegung definiert:
$$U\boldsymbol{b} = \langle \boldsymbol{c}, \boldsymbol{d}\rangle^\top, \quad \boldsymbol{c} \in \mathbb{R}^r, \quad \boldsymbol{d} \in \mathbb{R}^{m-r}$$
$$\Pi U\boldsymbol{b} = \langle \boldsymbol{c}, \boldsymbol{0}\rangle^\top, \quad (I-\Pi)U\boldsymbol{b} = \langle \boldsymbol{0}, \boldsymbol{d}\rangle^\top$$
- Das Ausgleichsproblem kann so umformuliert werden:
$$\begin{aligned} \|A\boldsymbol{x} - \boldsymbol{b}\|_2^2 &= \|U^\top(\Sigma V\boldsymbol{x} - U\boldsymbol{b})\|_2^2 = \|\Sigma V\boldsymbol{x} - U\boldsymbol{b}\|_2^2 \\ &= \|\Pi(\Sigma V\boldsymbol{x} - U\boldsymbol{b})\|_2^2 + \|(I-\Pi)(\Sigma V\boldsymbol{x} - U\boldsymbol{b})\|_2^2 \\ &+2\underbrace{\langle \Pi(\cdots), (I-\Pi)(\cdots)\rangle}_{=0} = \|\hat{\Sigma} V\boldsymbol{x} - \boldsymbol{c}\|_2^2 + \|\boldsymbol{d}\|_2^2 \end{aligned}$$
- Da $\kappa_2(\hat{\Sigma}) = \kappa_2(\hat{\Sigma}V)$ gilt, erfüllen die Lösungen der Systeme $\hat{\Sigma}V\boldsymbol{x} = \boldsymbol{c}$ und $\hat{\Sigma}V\tilde{\boldsymbol{x}} = \tilde{\boldsymbol{c}}$ die Fehlerabschätzung
$$\frac{\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|_2}{\|\boldsymbol{x}\|_2} \le \kappa_2(\hat{\Sigma}) \frac{\|\boldsymbol{c} - \tilde{\boldsymbol{c}}\|_2}{\|\boldsymbol{c}\|_2}$$
- Vergleiche mit $\kappa_2(A^\top A) = \kappa_2(\hat{\Sigma})^2$ für die normalen Gleichungen.

Berechnung der SWZ

Berechnung der SWZ (zur Lösung eines Ausgleichsproblems)

- Da die Singulärwerte durch die Eigenwerte von $A^\top A$ ($m > n$) gegeben sind, könnten wir den QR-Algorithmus auf $A^\top A$ anwenden, um $\Sigma^\top \Sigma$ zu bestimmen.
- Der folgende Algorithmus mit bidiagonalen Matrizen ist effizienter und stabiler.
- Der Plan:
  - Vereinfachung $PAQ = [B; Z]$, $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{n\times n}$ obere bidiagonal, $Z = 0 \in \mathbb{R}^{(m-n)\times n}$, $P \in \mathbb{R}^{m\times m}$, $Q \in \mathbb{R}^{n\times n}$ orthogonal.
  - Mit $B^{(1)} = B$ ist $B^{(l)\top} B^{(l)}$ tridiagonal.
  - Mit Givens konstruiere $Q^{(l)}$, mit dem $B^{(l)} Q^{(l)}$ unter bidiagonal ist.
  - Also ist $R^{(l)\top} = B^{(l)\top}(B^{(l)} Q^{(l)})$ unter tridiagonal.
  - Es gilt $B^{(l)\top} B^{(l)} = Q^{(l)} R^{(l)}$.
  - Mit Givens konstruiere $P^{(l)}$, mit dem $B^{(l+1)} = P^{(l)} B^{(l)} Q^{(l)}$ ober bidiagonal ist.
  - Es gilt $B^{(l+1)\top} B^{(l+1)} = R^{(l)} Q^{(l)}$!
  - Im Endeffekt ein Schritt des QR-Algorithmus.
  - Es folgt $|B^{(l)}| \to \mathrm{diag}\{\sigma_i\}_{i=1}^n$, $l \to \infty$.

Vereinfachung auf Bidiagonale Form

- Für die Vereinfachung $PAQ = [B; Z]$ setze
  $$A = A^{(1)} = [\boldsymbol{d}^{(1)} | C^{(1)}], \quad \boldsymbol{d}^{(1)} \in \mathbb{R}^{m\times 1}, \quad C^{(1)} \in \mathbb{R}^{m\times(n-1)}$$
- Für $k = 2, \ldots, n-1$,
  $$A^{(k)} = P^{(k-1)} A^{(k-1)} Q^{(k-1)} = \left(\begin{array}{c|c|c} B^{(k)} & \begin{matrix}0\\ a^{(k)}\end{matrix} & 0 \\ \hline 0 & \boldsymbol{d}^{(k)} & C^{(k)} \end{array}\right)$$
  $k > 2$
  wobei
  $$\begin{array}{llll} B^{(k)} & \in & \mathbb{R}^{(k-1)\times(k-1)} & a^{(k)} \in \mathbb{R}^1 \\ C^{(k)} & \in & \mathbb{R}^{(m-k+1)\times(n-k)} & \boldsymbol{d}^{(k)} \in \mathbb{R}^{(m-k+1)\times 1} \end{array}$$
- Es gelten $A^{(n-1)} = P^{(n-2)} \ldots P^{(1)} A Q^{(1)} \ldots Q^{(n-2)}$, $A^{(n)} = P^{(n-1)} A^{(n-1)}$ und $P^{(n)} A^{(n)} = A^{(n+1)} = [B^{(n+1)}; 0] = [B; 0]$.
- Schließlich $P = P^{(n+1)} \ldots P^{(1)}$ und $Q = Q^{(1)} \ldots Q^{(n-2)}$.
- Im nächsten Schritt,
  $$P^{(k)} = \left(\begin{array}{c|c} I^{(k-1)} & 0 \\ \hline 0 & \tilde{P}^{(k)} \end{array}\right) \quad Q^{(k)} = \left(\begin{array}{c|c} I^{(k)} & 0 \\ \hline 0 & \tilde{Q}^{(k)} \end{array}\right)$$
  wobei durch Householder Transformationen
  $$\begin{array}{rcl@{\qquad}rcl} \tilde{P}^{(k)} \boldsymbol{d}^{(k)} & = & -\sigma \hat{\boldsymbol{e}}_1 \in \mathbb{R}^{m-k+1}, & \boldsymbol{c}^{(k)} & = & \hat{\boldsymbol{e}}_1^\top \tilde{P}^{(k)} C^{(k)}, \\ \tilde{Q}^{(k)\top} \boldsymbol{c}^{(k)\top} & = & -\tau \hat{\boldsymbol{e}}_1 \in \mathbb{R}^{n-k}, & a^{(k+1)} & = & -\tau. \end{array}$$

Pseudo-Code zur Transformation auf Bidiagonale Form

```
d(i) = A(i,1), C(i,j) = A(i,j+1), i=1,...,m, j=1,...,n-1
P = I (m×m), Q = I (n×n)
for k=1,...,n
   % d is (m-k+1)×1
   (u,sg) = Householder(d)  106        % (I-uu')d=-sg e1
   B(k,k) = -sg                        % B grows
   if (k>1) then B(k-1,k) = a
   for l=1,...,n-k                     % C̃ <- (I-uu')C̃
      sm = sum(C(i,l)*u(i):  i=1,...,m-k+1)
      C(i,l) = C(i,l) - sm*u(i), i=1,...,m-k+1
   end
   for l=1,...,m                       % P̃ <- (I-uu')P̃
      sm = sum(P(k-1+i,l)*u(i):  i=1,...,m-k+1)
      P(k-1+i,l) = P(k-1+i,l) - sm*u(i), i=1,...,m-k+1
   end
   if (k < n-1)
      c(j) = C(1,j), j=1,...,n-k
      % c is (n-k)×1
```

Pseudo-Code zur Transformation auf Bidiagonale Form

```
        (v,ta) = Householder(c)  106          % (I-vv')c=-ta e1
        a = -tau
        for l=2,...,m-k+1                     % C̃ <- C̃(I-vv')
            sm = sum(C(l,i)*v(i):  i=1,...,n-k)
            C(l,i) = C(l,i) - sm*v(i), i=1,...,n-k
        end
        for l=1,...,n                         % Q̃ <- Q̃(I-vv')
            sm = sum(Q(l,k+i)*v(i):  i=1,...,n-k)
            Q(l,k+i) = Q(l,k+i) - sm*v(i), i=1,...,n-k
        end
        d(i) = C(i+1,1), i=1,...,m-k          % d & C shrink
        C(i,j) = C(i+1,j+1), i=1,...,m-k, j=1,...,n-k-1
    end
    if (k = n-1)
        d(i) = C(i+1,1), i=1,...,m-k          % d shrinks
        a = C(1,1)                            % no ta
    end
end

% P*A*Q = [B;0], B n×n
```

Diagonalisierende Iteration

- Bei $k = n - 1$,
$$A^{(n)} = P^{(n-1)} A^{(n-1)}$$
und $Q^{(n-1)}$ ist nicht notwendig, weil $C^{(n)}$ nur eine einzige Spalte hat, und $c^{(n-1)} \in \mathbb{R}^1$ ist schon $a^{(n)}$.
- Bei $k = n$,
$$A^{(n+1)} = P^{(n)} A^{(n)}$$
und $Q^{(n)}$ ist nicht notwendig, weil $C^{(n)}$ nicht existiert. Da $\boldsymbol{d}^{(n)} \in \mathbb{R}^{(m-n+1)\times 1}$, ist $P^{(n)}$ aber notwendig.
- Dann ist $B = B^{(n+1)}$ bidiagonal.
- Für die Diagonalisierung von B setze $B^{(1)} = B$.
- Mit Givens konstruiere $Q^{(l)}$, mit dem $B^{(l)} Q^{(l)}$ unter bidiagonal ist.
- Die Kette von Givens Transformationen ist eine obere Hessenberg Matrix 118,
$$Q^{(l)} = G^{(1,2)}(\theta_1) \cdots G^{(n-1,n)}(\theta_{n-1})$$

Diagonalisierende Iteration

- Die Givens Transformationen werden so ausgewählt, dass

$$B^{(l)}Q^{(l)} = \begin{pmatrix} \square & \square & & \\ & \ddots & \ddots & \\ & & \ddots & \square \\ & & & \square \end{pmatrix} \qquad Q^{(l)} = \begin{pmatrix} \square & \swarrow 0 & & \\ \square & \ddots & \ddots & \\ & \ddots & \ddots & \swarrow 0 \\ & & \square & \square \end{pmatrix}$$

  unter bidiagonal ist.
- Durch Multiplikation links mit $B^{(l)\top}$ ist das Ergebnis unter tridiagonal,

$$R^{(l)\top} = B^{(l)\top}(B^{(l)}Q^{(l)}) = \begin{pmatrix} \square & & & \\ \square & \ddots & & \\ & \ddots & \ddots & \\ & & \square & \square \end{pmatrix} \begin{pmatrix} \square & & & \\ \square & \ddots & & \\ & \ddots & \ddots & \\ & & \square & \square \end{pmatrix}$$

  und daher ist $R^{(l)}$ ober tridiagonal.

Folge der Bidiagonalen Matrizen

- Daher ist

$$B^{(l)\top} B^{(l)} = Q^{(l)} R^{(l)}$$

eine QR Zerlegung von der tridiagonale Matrix $B^{(l)\top} B^{(l)}$,

$$\begin{pmatrix} \square & \square & & \\ \square & \ddots & \ddots & \\ & \ddots & \ddots & \square \\ & & \square & \square \end{pmatrix} = B^{(l)\top} B^{(l)} = Q^{(l)} R^{(l)} =$$

$$\begin{pmatrix} \square & \square & \cdots & \cdots & \square \\ \square & \ddots & \ddots & & \vdots \\ & \ddots & \ddots & \ddots & \vdots \\ & & \ddots & \ddots & \square \\ & & & \square & \square \end{pmatrix} \begin{pmatrix} \square & \square & \square & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \square \\ & & & \ddots & \square \\ & & & & \square \end{pmatrix}$$

Folge der Bidiagonalen Matrizen

- Mit Givens konstruiere $P^{(l)}$, mit dem

$$B^{(l+1)} = P^{(l)}(B^{(l)}Q^{(l)}) = P^{(l)} \begin{pmatrix} \square & & & \\ \square & \ddots & & \\ & \ddots & \ddots & \\ & & \square & \square \end{pmatrix} = \begin{pmatrix} \square & \square & & \\ 0\nearrow & \ddots & \ddots & \\ & \ddots & \ddots & \square \\ & & 0\nearrow & \square \end{pmatrix}$$

  ober bidiagonal ist.
- Die Kette von Givens Transformationen ist eine untere Hessenberg Matrix 118,

$$P^{(l)} = G^{(n-1,n)}(\theta_{n-1}) \cdots G^{(1,2)}(\theta_1)$$

- Schließlich ist ein Schritt des QR Verfahrens vollständig.

$$\begin{aligned} B^{(l+1)\top}B^{(l+1)} &= (P^{(l)}B^{(l)}Q^{(l)})^\top (P^{(l)}B^{(l)}Q^{(l)}) \\ &= (Q^{(l)\top}B^{(l)\top}P^{(l)\top})(P^{(l)}B^{(l)}Q^{(l)}) \\ &= Q^{(l)\top}(B^{(l)\top}B^{(l)})Q^{(l)} = Q^{(l)\top}(Q^{(l)}R^{(l)})Q^{(l)} \\ &= R^{(l)}Q^{(l)} \end{aligned}$$

Pseudo-Code, *QR* Algorithmus zur SW-Zerlegung

```
Q = I, P = I, Bl = B
for l=1,...,lmax
    a(i) = Bl(i,i), i=1,...,n
    b(i) = Bl(i,i+1), i=1,...,n-1
    x(1) = a(1)
    for j=1...,(n-1)                    % Ql = G(c1,s1)G(c2,s2)...
        g(j) = sqrt(x(j)^2 + b(j)^2)
        c(j) = x(j)/g(j), s(j) = -b(j)/g(j)
        d(j) = -s(j)*a(j+1), x(j+1) = c(j)*a(j+1)
    end
    g(n) = x(n)                         % Bl*Ql = (lower) diags {d,g}
    Ql = I
    for j=1,...,(n-1)
        q1(i) = Ql(i,j)*c(j) - Ql(i,j+1)*s(j), i=1,...,n
        q2(i) = Ql(i,j+1)*c(j) + Ql(i,j)*s(j), i=1,...,n
        Ql(i,j) = q1(i), Ql(i,j+1) = q2(i), i=1,...,n
    end                                 % (Bl*Ql)'*Bl = Rl
    Q = Q*Ql
    a(i) = (Bl*Ql)(i,i) = g(i), i=1,...,n
    b(i) = (Bl*Ql)(i+1,i) = d(i), i=1,...,n-1
```

Pseudo-Code, *QR* Algorithmus zur SW-Zerlegung

```
    x(1) = a(1)
    for j=1...,(n-1)
       g(j) = √(x(j)^2 + b(j)^2)
       c(j) = x(j)/g(j), s(j) = b(j)/g(j)
       d(j) = s(j)*a(j+1), x(j+1) = c(j)*a(j+1)
    end
    g(n) = x(n)                    % B(l+1) = (upper) diags {g,d}
    Pl = I
    for i=1,...,(n-1)              % ...G(c2,s2)G(c1,s1) = Pl
       p1(j) = Pl(i,j)*c(i) + Pl(i+1,j)*s(i), j=1,...,n
       p2(j) = Pl(i+1,j)*c(i) - Pl(i,j)*s(i), j=1,...,n
       Pl(i,j) = p1(j), Pl(i+1,j) = p2(j), j=1,...,n
    end                            % Pl*(Bl*Ql) = B(l+1)
    P = Pl*P                       % B(l+1)'*B(l+1) = Rl*Ql
    Bl(i,i) = g(i), i=1,...,n
    Bl(i,i+1) = d(i), i=1,...,(n-1)
    if (||d|| < tol * ||g||) then break
end                                % |g(i)| ≈ Singulärwerte von B
Sn(i,i) = sign(g(i)), i=1,...,n
S(i,i) = |g(i)|, i=1,...,n
P=Sn*P                             % S = P*B*Q
```

Interpolation

- Für polynomiale Regression ist das Ausgleichsproblem mit $m > n$ entstanden:

$$E(\boldsymbol{c}) = \sum_{i=0}^{m-1} [y_i - P(x_i; \boldsymbol{c})]^2, \quad P(x; \boldsymbol{c}) = \sum_{k=0}^{n-1} c_k x^k$$

- Nun sei $n = m$, d.h. alle Datenpunkte werden *gespießt*.
- Die explizite Lösung von min $E(\boldsymbol{c})$ lässt sich bezüglich der *Lagrange Polynome* darstellen.
- Beispiel: Gegeben sind (x_0, y_0) und (x_1, y_1). Die linearen Lagrange Polynome sind:

$$L_{1,0}(x) = \frac{x - x_1}{x_0 - x_1} \quad L_{1,1}(x) = \frac{x - x_0}{x_1 - x_0}$$

die erfüllen:

$$\left.\begin{array}{ll} L_{1,0}(x_0) = 1 & L_{1,1}(x_0) = 0 \\ L_{1,0}(x_1) = 0 & L_{1,1}(x_1) = 1 \end{array}\right\} \text{ oder } L_{1,i}(x_j) = \delta_{ij}$$

und

$$P(x) = \sum_{i=0}^{1} y_i L_{1,i}(x) \quad \text{erfüllt} \quad P(x_j) = \sum_{i=0}^{1} y_i L_{1,i}(x_j) = \sum_{i=0}^{1} y_i \delta_{ij} = y_j$$

Lagrange Polynome

- Mit (x_0, y_0), (x_1, y_1), (x_2, y_2), mit den quadratischen

$$L_{2,0}(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} \quad L_{2,1}(x) = \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} \quad L_{2,2}(x) = \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}$$

und

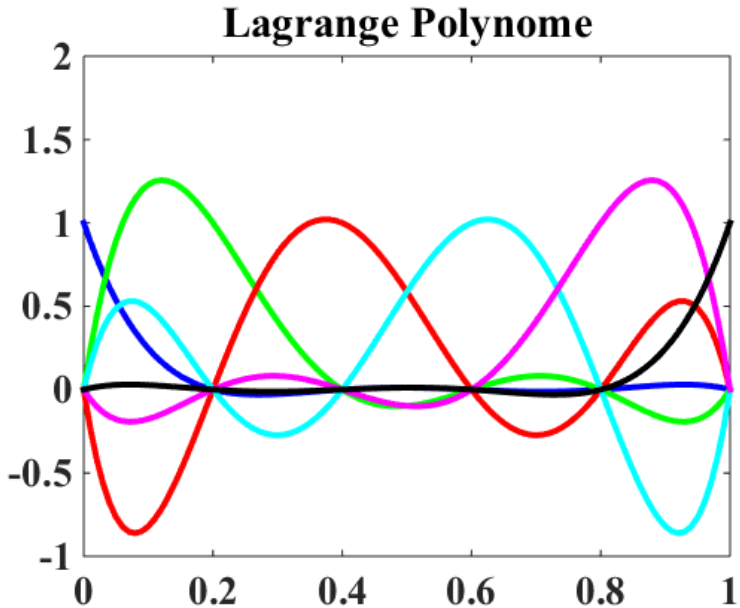
$$P(x) = \sum_{i=0}^{2} y_i L_{2,i}(x) \quad \text{folgt} \quad P(x_j) = \sum_{i=0}^{2} y_i L_{2,i}(x_j) = \sum_{i=0}^{2} y_i \delta_{ij} = y_j$$

- Im Allgemeinen werden die Lagrange Polynome bezüglich Stützstellen $\boldsymbol{x} = \{x_i\}_{i=0}^{n}$ so definiert:

$$L_{n,i}(x; \boldsymbol{x}) = \prod_{i \neq j=0}^{n} \frac{x - x_j}{x_i - x_j}$$

$$P_n(x) = \sum_{i=0}^{n} y_i L_{n,i}(x; \boldsymbol{x})$$

$$P_n(x_j) = \sum_{i=0}^{n} y_i L_{n,i}(x_j; \boldsymbol{x}) = \sum_{i=0}^{n} y_i \delta_{ij} = y_j$$



- Was ist die Qualität dieser Interpolation?

Genauigkeit globaler Interpolation

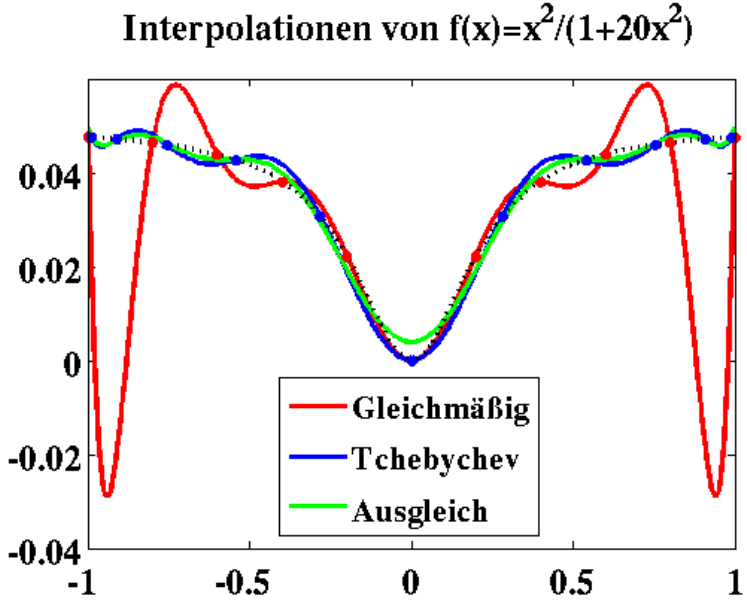
- Beispiel: Die Funktion $f(x) = x^2/(1+20x^2) \in C^\infty(\mathbb{R})$ wird mit $n = 10$ oder $m = 100$ Punkten folgendermaßen interpoliert:

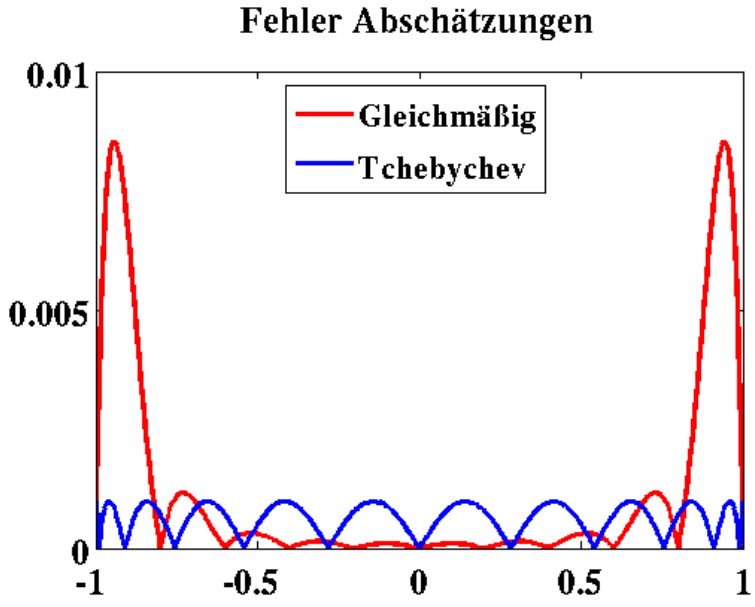
(Gleichmäßig)
- $P(x_i) = f(x_i)$, $x_i = -1 + 2i/n$, $i = 0, \ldots, n$

(Tchebychev)
- $Q(t_j) = f(t_j)$, $t_j = \cos(\pi(n - j + 1/2)/(n+1))$, $j = 0, \ldots, n$

(Ausgleich)
- $\sum_{k=0}^{m} |R(y_k) - f(y_k)|^2 = \min$, $y_k = -1 + 2k/m$, $k = 0, \ldots, m$

- Die Ergebnisse sehen so aus:





wobei theoretische Abschätzungen der Fehler rechts gezeigt werden.

Genauigkeit globaler Interpolation

Satz: Wenn $\{x_i\}_{i=0}^n \subset [a,b]$ und $f \in \mathcal{C}^{n+1}([a,b])$ dann $\forall x \in [a,b], \exists \xi(x) \in [a,b] \ni$

$$f(x) = P_n(x) + \frac{f^{(n+1)}(\xi(x))}{(n+1)!}\psi_{n+1}(x; x_0, \ldots, x_n)$$

wobei

$$P_n(x) = \sum_{i=0}^{n} f(x_i)L_{n,i}(x) \quad \psi_{n+1}(x; x_0, \ldots, x_n) = \prod_{i=0}^{n}(x - x_i)$$

Insbesondere gilt

$$|f(x) - P_n(x)| \leq B|\psi_{n+1}(x; x_0, \ldots, x_n)|$$

wenn $|f^{(n+1)}(\xi)| \leq B(n+1)!, \forall \xi \in [a,b]$.

- Wie wird $|\psi_{n+1}(x; x_0, \ldots, x_n)|$ minimiert?
- Diese Funktion ist oben grafisch dargestellt für die Fälle:
  - Gleichmäßig verteilte Stützstellen und
  - Tchebychev Stützstellen.
- Das zweite Ergebnis ist klar besser, aber in der Praxis sind die Stützstellen $\{x_i\}_{i=0}^n$ nicht immer frei auszuwählen.

Iterierte Interpolation

Def: Gegeben seien die Daten $\{(x_i, f_i)\}_{i=0}^n$ und verschiedene Werte $\{m_j\}_{j=1}^k$, $0 \leq m_j \leq n$. Dann wird mit $P_{m_1,\ldots,m_k}$ das Polynom bezeichnet, das die Daten $\{(x_{m_j}, f_{m_j})\}_{j=1}^k$ interpoliert.

- Beispiel: Mit (x_0, f_0), (x_1, f_1) und (x_2, f_2) gilt

$$P_{0,2}(x) = \frac{(x - x_2)}{(x_0 - x_2)} f_0 + \frac{(x - x_0)}{(x_2 - x_0)} f_2$$

Satz: Gegeben seien $\{(x_i, f_i)\}_{i=0}^n$, $P_{0,\ldots,j-1,j+1,\ldots,n}$ und $P_{0,\ldots,i-1,i+1,\ldots,n}$, $i \neq j$. Dann gilt

$$P_{0,\ldots,n}(x) = \frac{(x - x_j)P_{0,\ldots,j-1,j+1,\ldots,n}(x) - (x - x_i)P_{0,\ldots,i-1,i+1,\ldots,n}(x)}{x_i - x_j}$$

d.h. $P_{0,\ldots,n}(x_i) = P_{0,\ldots,j-1,j+1,\ldots,n}(x_i) = f_i$

und $P_{0,\ldots,n}(x_j) = P_{0,\ldots,i-1,i+1,\ldots,n}(x_j) = f_j$.

- Dieser Satz führt zu einem Algorithmus zur Berechnung des Polynoms $P_{0,\ldots,n}$.

Iterierte Interpolation

- Beispiel: Gegeben seien $\{(x_i, f_i)\}_{i=0}^{2}$ und eine Auswertungsstelle x für P_{012}. Man bekommt $P_{012}(x)$ aus der Tabelle:

$$
\begin{array}{lllll}
x_0 & P_0(x) = f_0 & & & \\
x_1 & P_1(x) = f_1 & P_{01}(x) = \frac{(x-x_1)P_0-(x-x_0)P_1}{x_0-x_1} & & \\
x_2 & P_2(x) = f_2 & P_{12}(x) = \frac{(x-x_2)P_1-(x-x_1)P_2}{x_1-x_2} & P_{012}(x) = \frac{(x-x_2)P_{01}-(x-x_0)P_{12}}{x_0-x_2} &
\end{array}
$$

- Wenn ein neuer Datenpunkt verfügbar wird, dann kann die Tabelle einfach ergänzt werden:

$$
\begin{array}{lllll}
x_0 & P_0(x) = f_0 & & & \\
x_1 & P_1(x) = f_1 & P_{01}(x) = \frac{(x-x_1)P_0-(x-x_0)P_1}{x_0-x_1} & & \\
x_2 & P_2(x) = f_2 & P_{12}(x) = \frac{(x-x_2)P_1-(x-x_1)P_2}{x_1-x_2} & P_{012}(x) = \frac{(x-x_2)P_{01}-(x-x_0)P_{12}}{x_0-x_2} & \\
x_3 & P_3(x) = f_3 & P_{23}(x) = \frac{(x-x_3)P_2-(x-x_2)P_3}{x_2-x_3} & P_{123}(x) = \frac{(x-x_3)P_{12}-(x-x_1)P_{23}}{x_1-x_3} & P_{0123}(x) = \frac{(x-x_3)P_{012}-(x-x_0)P_{123}}{x_0-x_3}
\end{array}
$$

Nevilles Algorithmus

Algorithmus (Neville): Eingaben sind Daten $\{(x_i, f_i)\}_{i=0}^{n}$ und die Auswertungsstelle x, Ausgabe ist der Wert $P_{0,\ldots,n}(x)$.

$Q_{i,0} = f_i,\ i = 0, \ldots, n$
for $i = 1, \ldots, n$
    for $j = 1, \ldots, i$

$$Q_{ij} = \frac{(x - x_{i-j})Q_{i,j-1} - (x - x_i)Q_{i-1,j-1}}{x_i - x_{i-j}}$$

    end
end
return Q_{nn}

Bildlich gesehen:

$$\begin{array}{lllll} x_0 & Q_{00} & & & \\ \vdots & \vdots & Q_{11} & & \\ \vdots & \vdots & & \ddots & \\ x_n & Q_{n0} & Q_{n1} & \cdots & Q_{nn} \end{array}$$

$$Q_{ij} = P_{i-j,\ldots,i}$$

$$Q_{nn} = P_{n-n,\ldots,n}$$

Dividierte Differenzen

Def: $P_n(x)$ bezeichnet hier das Polynom, das die Daten $\{(x_i, f_i)\}_{i=0}^n$ interpoliert.

- Nun wird nicht nur der Wert $P_n(x)$ an einer Auswertungsstelle x berechnet, sondern auch Koeffizienten des Polynoms bestimmt.
- $P_n(x)$ kann so dargestellt werden:

  $$P_n(x) = a_0 + a_1(x - x_0) + \cdots + a_n(x - x_0) \times \cdots \times (x - x_{n-1})$$

- Die Koeffizienten $\{a_i\}_{i=0}^n$ ergeben sich durch:

$$\begin{aligned} f_0 &= P_n(x_0) &&= a_0 =: f[x_0] \\ f_1 &= P_n(x_1) &&= a_0 + a_1(x_1 - x_0) = f_0 + a_1(x_1 - x_0) \\ & && \Rightarrow a_1 = \frac{f_1 - f_0}{x_1 - x_0} =: f[x_0, x_1] \end{aligned}$$

- Die restlichen Koeffizienten werden induktiv durch ähnliche Differenzen gegeben:

$$a_k = f[x_0, \ldots, x_k], \quad f[x_i, \ldots, x_{i+k}] = \frac{f[x_{i+1}, \ldots, x_{i+k}] - f[x_i, \ldots, x_{i+k-1}]}{x_{i+k} - x_i}$$

Dividierte Differenzen

- Das interpolierende Polynom lässt sich so darstellen,

$$P_n(x) = f[x_0] + \sum_{k=1}^{n} f[x_0, \ldots, x_k] \prod_{l=0}^{k-1} (x - x_l)$$

und soll mit dem Horner Algorithmus ausgewertet werden.

Satz: Wenn $f \in \mathcal{C}^n([a,b])$ und $\{x_i\}_{i=0}^n \subset [a,b]$ verschieden sind, dann $\exists \xi \in (a,b) \ni$

$$f[x_0, \ldots, x_n] = \frac{f^{(n)}(\xi)}{n!}$$

Def: $\mathcal{P}^k(\Omega)$ bezeichnet die Menge der Polynome eines Grades höchstens k auf einem Gebiet Ω.

Bemerkung: Seien $P_\mathrm{L}, P_\mathrm{D} \in \mathcal{P}^n([x_0, x_n])$ die durch Lagrange bzw. Dividierte Differenzen gegebenen Polynome, die die Daten $\{(x_i, f_i)\}_{i=0}^n$ interpolieren. Dann gilt $P_\mathrm{L}(x) \equiv P_\mathrm{D}(x)$, weil $Q = P_\mathrm{L} - P_\mathrm{D} \in \mathcal{P}^n([x_0, x_n])$, und Q hat die $n+1$ Nullstellen $\{x_i\}_{i=0}^n$.

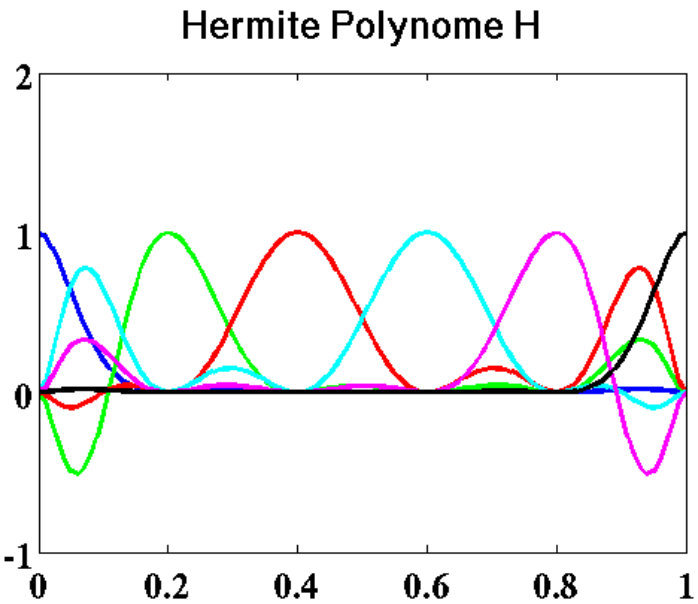
Hermite Interpolation

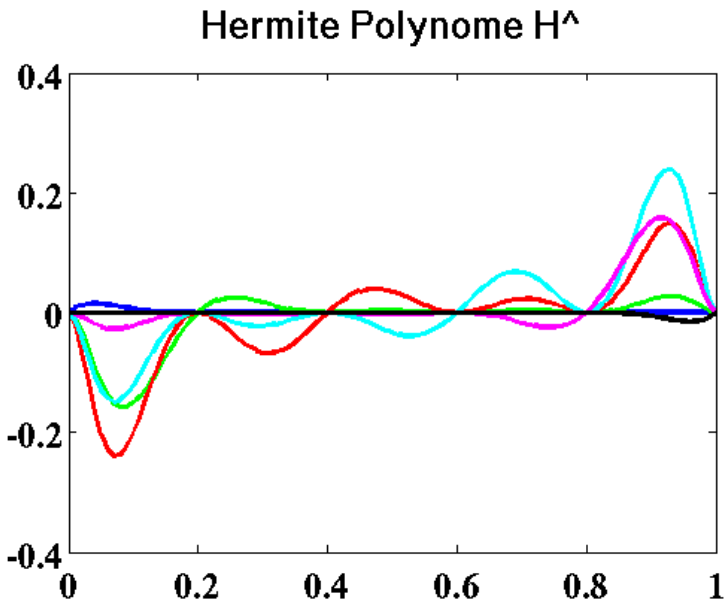
- Nun seien Daten $\{(x_i, f_i, \hat{f}_i)\}_{i=0}^n$ gegeben, und ein Polynom $H(x)$ soll konstruiert werden, die erfüllt

$$H(x_i) = f_i, \quad H'(x_i) = \hat{f}_i, \quad i = 0, \ldots, n.$$

- Für gegebene Stützstellen $\{x_i\}_{i=0}^n$ seien die Hermite Polynome in $\mathcal{P}^{2n+1}([x_0, x_n])$ bezüglich der Lagrange Polynome definiert durch

$$\begin{aligned} H_{n,i}(x) &= [1 - 2(x - x_i)L'_{n,i}(x_i)]L^2_{n,i}(x) \\ \hat{H}_{n,i}(x) &= (x - x_i)L^2_{n,i}(x) \end{aligned}$$





Hermite Interpolation

- Die Hermite Polynome bilden eine Basis für die Interpolationsaufgabe,

$$H_{n,i}(x_j) = \delta_{ij}, \quad \hat{H}_{n,i}(x_j) = 0,$$
$$H'_{n,i}(x_j) = 0, \quad \hat{H}'_{n,i}(x_j) = \delta_{ij}.$$

- Die Interpolante

$$H(x) = \sum_{i=0}^{n} \left[f_i H_{n,i}(x) + \hat{f}_i \hat{H}_{n,i}(x) \right]$$

erfüllt

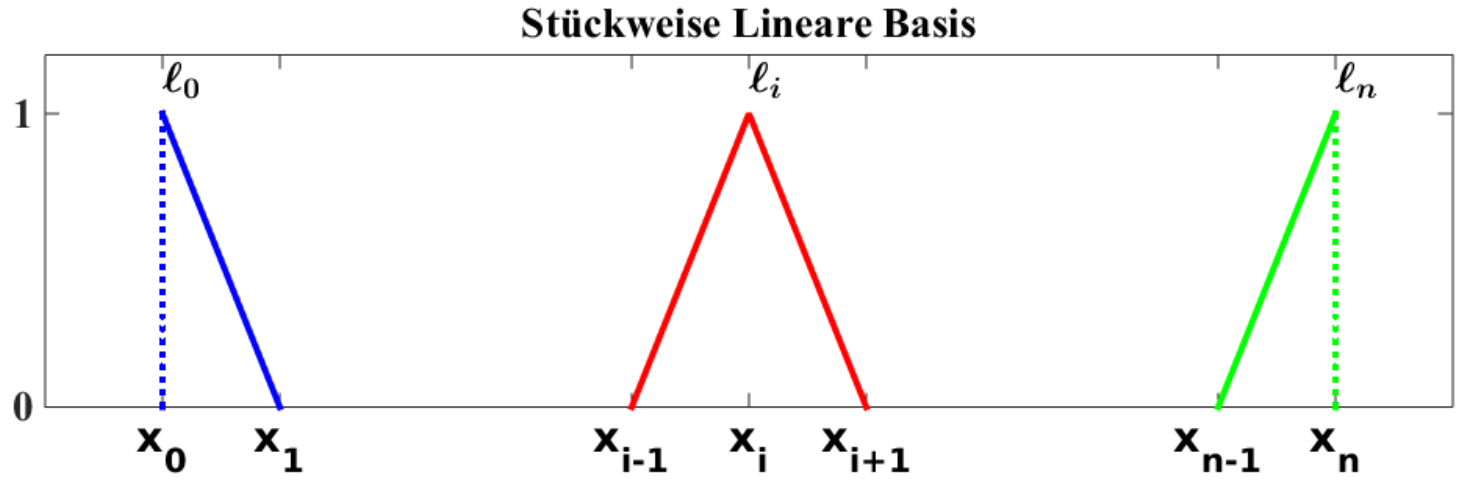
$$(\star) \qquad H(x_j) = f_j, \quad H'(x_j) = \hat{f}_j, \quad i = 0, \ldots, n$$

Satz: Wenn $\{x_i\}_{i=0}^n$ verschieden sind und $f \in \mathcal{C}^1([x_0, x_n])$ gilt, dann ist $H \in \mathcal{P}^{2n+1}([x_0, x_n])$ das eindeutige Polynom kleinsten Grades, das $(\star)$ mit $f_i = f(x_i)$, $\hat{f}_i = f'(x_i)$, $i = 0, \ldots, n$, erfüllt.

Stückweise Polynome Interpolation

- Seien Stützstellen $\{x_i\}_{i=0}^n$ gegeben.
- Die entsprechenden Basisfunktionen für stückweise lineare Interpolation sind

$$\ell_i(x) = \begin{cases} \dfrac{x - x_{i-1}}{x_i - x_{i-1}}, & x_{i-1} \le x \le x_i \\ \dfrac{x_{i+1} - x}{x_{i+1} - x_i}, & x_i \le x \le x_{i+1} \\ 0, & \text{sonst.} \end{cases}$$



- Es gilt $\ell_i(x_j) = \delta_{ij}$.

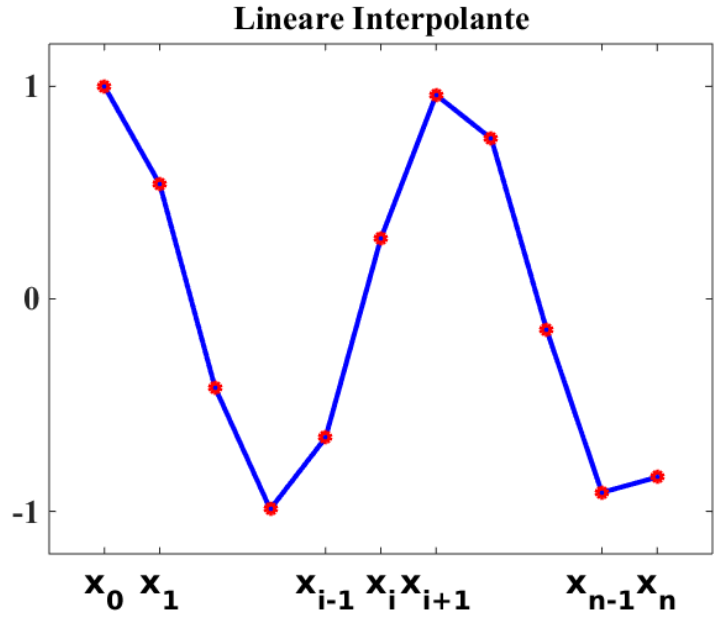
Stückweise Polynome Interpolation

- Die Funktion,

$$L(x) = \sum_{i=0}^{n} f_i \ell_i(x)$$

erfüllt

$$L(x_j) = \sum_{i=0}^{n} f_i \ell_i(x_j) = \sum_{i=0}^{n} f_i \delta_{ij} = f_j$$



Satz: Gegeben seien $\{x_i\}_{i=0}^{n}$ mit $h = \max_i(x_{i+1} - x_i)$ und $f \in \mathcal{C}^2([x_0, x_n])$. Dann $\exists c \neq c(h)$ mit

$$\max_{x\in[x_0,x_n]} |f(x) - L(x)| \leq ch^2 \max_{x\in[x_0,x_n]} |D^2 f(x)|$$
$$\max_{x\in[x_0,x_n]} |f'(x) - L'(x)| \leq ch \max_{x\in[x_0,x_n]} |D^2 f(x)|$$

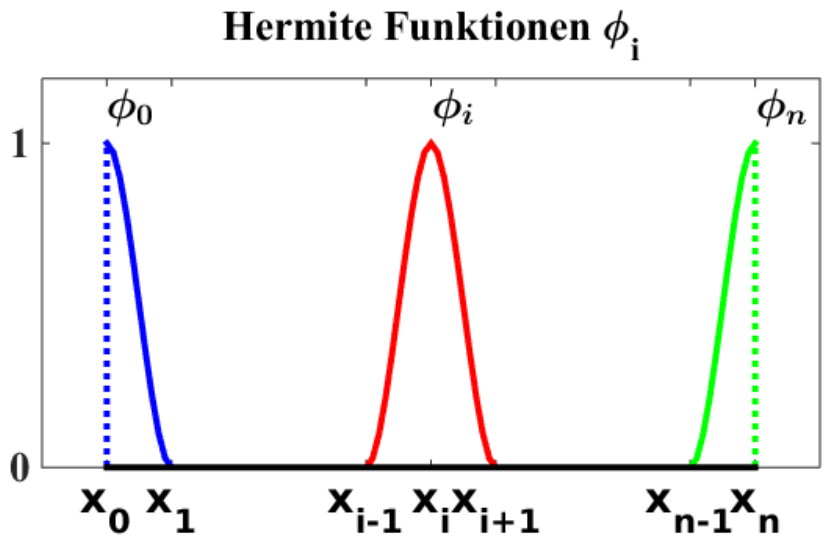
- Siehe `Details`

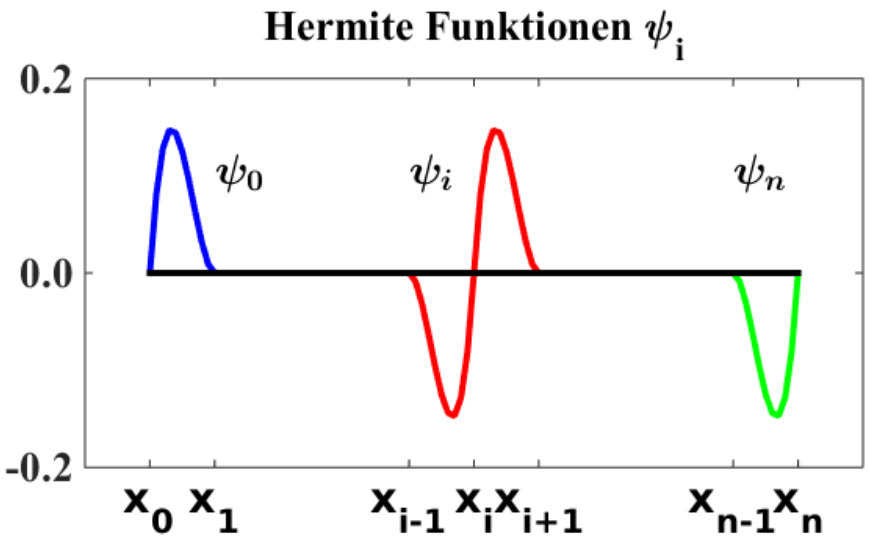
Stückweise Kubisch Hermite Interpolation

- Mit Daten $\{(x_i, f_i, \hat{f}_i)\}_{i=0}^n$ wird ein $H \in \mathcal{C}^1([x_0, x_n])$ mit $H \in \mathcal{P}^3([x_i, x_{i-1}])$, $i = 1, \ldots, n$, konstruiert, das erfüllt

$$H(x_i) = f_i, \quad H'(x_i) = \hat{f}_i, \quad i = 0, \ldots, n.$$

- Seien die Basisfunktionen $\{\phi_i\}_{i=0}^n$, $\{\psi_i\}_{i=0}^n \subset \mathcal{C}^1([x_0, x_n])$





definiert, die stückweise kubisch sind und erfüllen:

$$\phi_i(x_j) = \delta_{ij}, \quad \psi_i(x_j) = 0,$$
$$\phi_i'(x_j) = 0, \quad \psi_i'(x_j) = \delta_{ij}.$$

Stückweise Kubisch Hermite Interpolation

- Hausaufgabe: Konstruiere $\{\phi_i\}_{i=0}^n$, $\{\psi_i\}_{i=0}^n$ für ein regelmäßiges Gitter.
- Die Interpolante

$$H(x) = \sum_{i=0}^{n} \left[f_i \phi_i(x) + \hat{f}_i \psi_i(x) \right]$$

erfüllt

$$H(x_j) = f_j, \quad H'(x_j) = \hat{f}_j, \quad i = 0, \ldots, n$$

Satz: Gegeben seien $\{x_i\}_{i=0}^n$ mit $h = \max_i(x_{i+1} - x_i)$ und $f \in \mathcal{C}^4([x_0, x_n])$. Dann $\exists c_k \neq c_k(h)$ mit

$$\max_{x \in [x_0, x_n]} |D^k[f(x) - H(x)]| \leq c h^{4-k} \max_{x \in [x_0, x_n]} |D^4 f(x)|$$

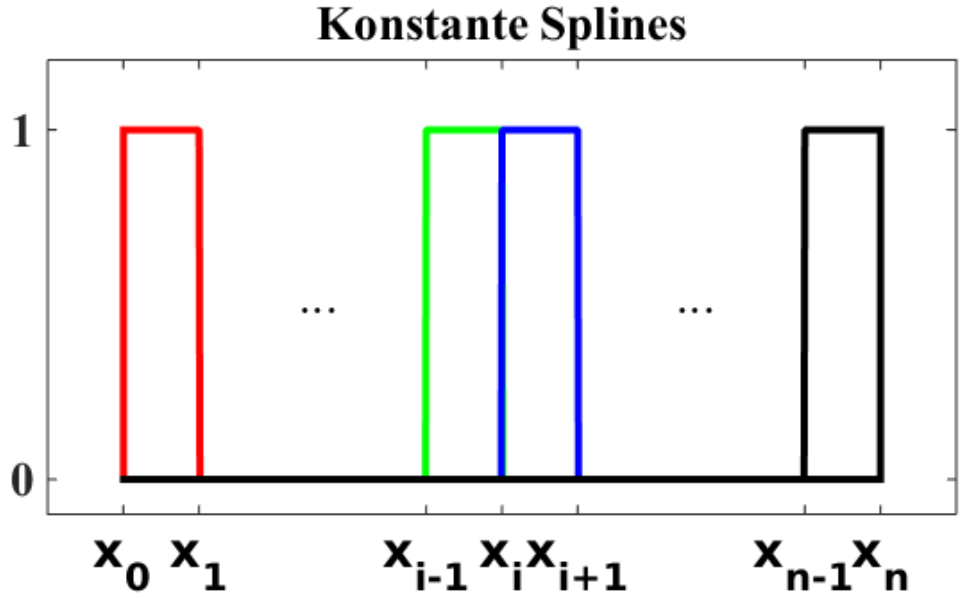
für $k = 0, \ldots, 3$.

Splines

- Spline Funktionen sind stückweise Polynome.
- Sei ein Gitter $\boldsymbol{x} = \{x_i\}_{i=0}^{n}$ gegeben.
- Die Glattheit an jeder Stelle x_i ist veränderlich, aber nur die *B-Splines* mit maximaler Glattheit werden hier untersucht.
- Für $k = 0, \ldots, 3$ wird eine *Basis* $\{s_i\}_{i=-k}^{n-1}$ dargestellt für

$$\mathcal{S}^k(\boldsymbol{x}) = \{\underbrace{s \in \mathcal{C}^{k-1}([x_0, x_n])}_{\text{nur für } k \geq 1} : s \in \mathcal{P}^k([x_{i-1}, x_i]), 1 \leq i \leq n\}$$

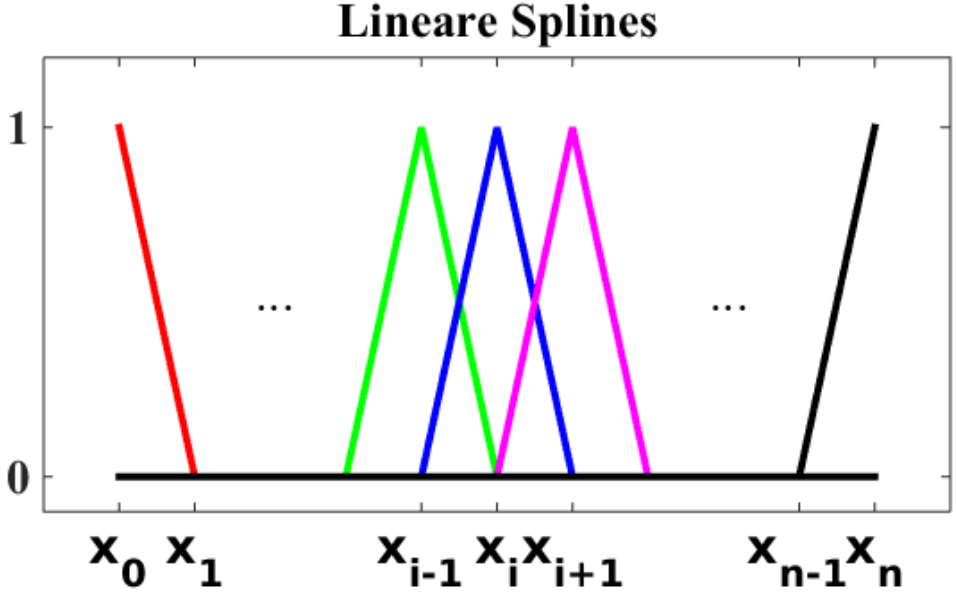
- Basis für stückweise konstante Funktionen:



$$\sum_{i=0}^{n-1} \alpha_i s_i(x) = f(x) \in \mathcal{S}^0(\boldsymbol{x})$$

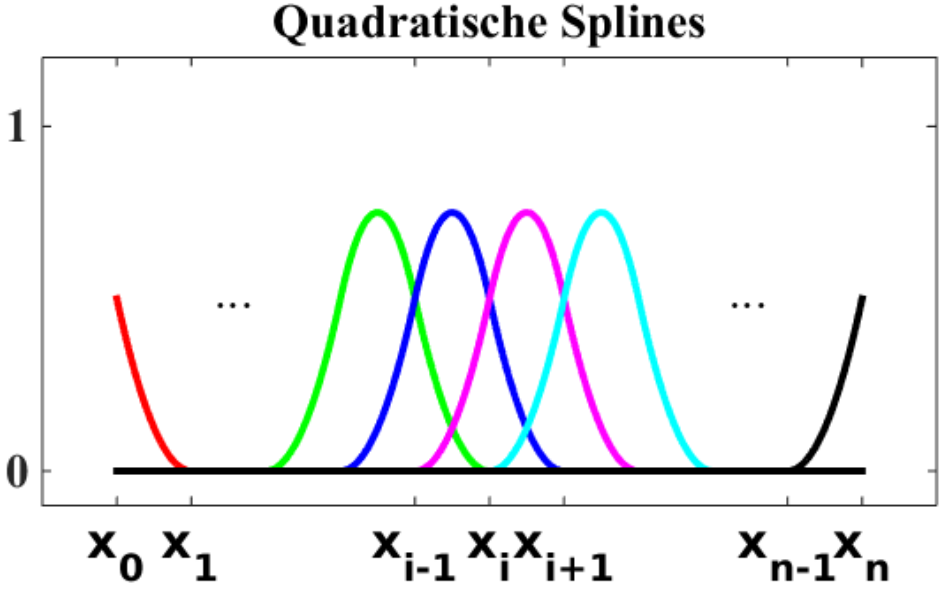
Splines

- Basis für stückweise lineare Funktionen in $\mathcal{C}^0([a,b])$:



$$\sum_{i=-1}^{n-1} \alpha_i s_i(x) = f(x) \in \mathcal{S}^1(\boldsymbol{x})$$

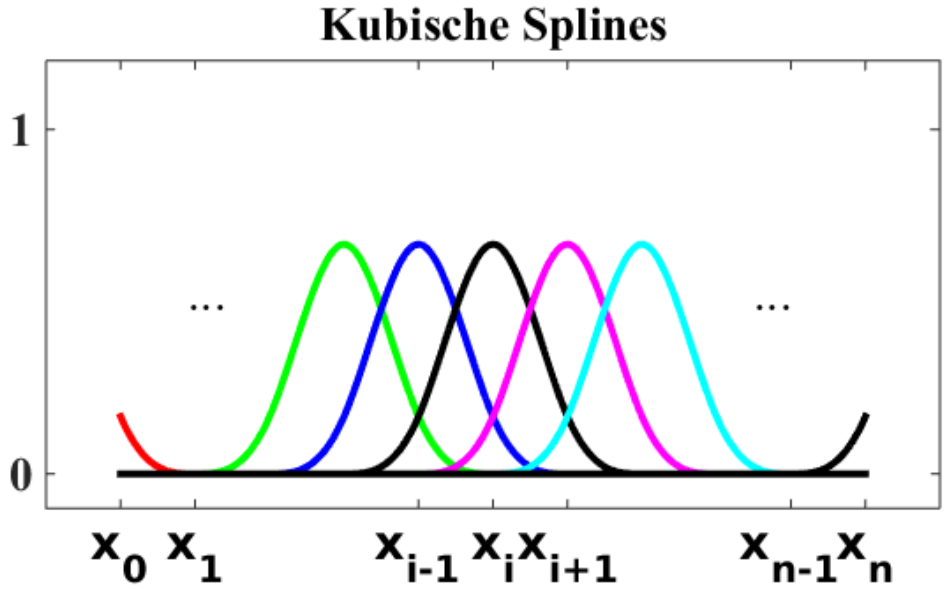
- Basis für stückweise quadratische Funktionen in $\mathcal{C}^1([a,b])$:



$$\sum_{i=-2}^{n-1} \alpha_i s_i(x) = f(x) \in \mathcal{S}^2(\boldsymbol{x})$$

Splines

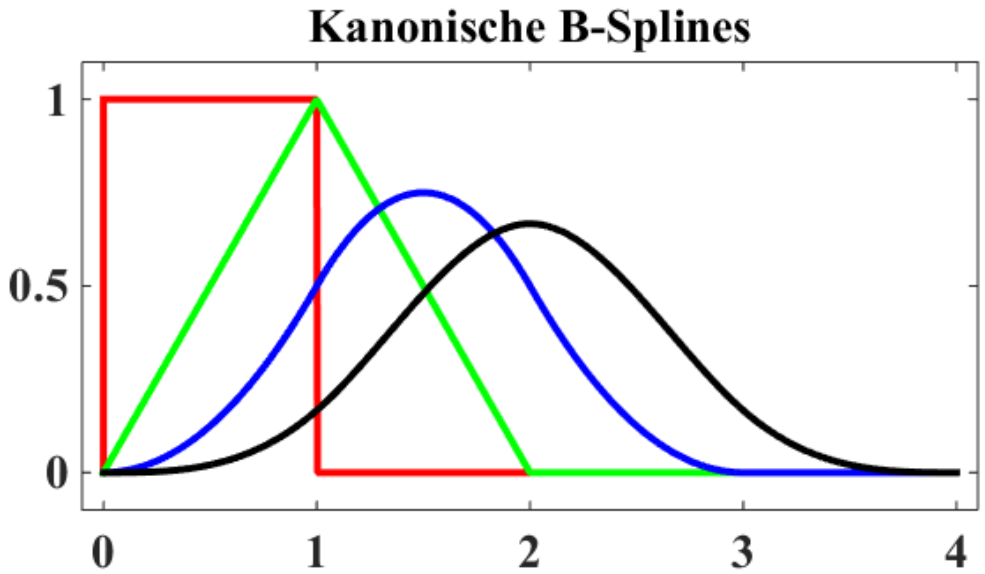
- Basis für stückweise kubische Funktionen in $\mathcal{C}^2([a,b])$:



$$\sum_{i=-3}^{n-1} \alpha_i s_i(x) = f(x) \in \mathcal{S}^3(\boldsymbol{x})$$

- Um jede Basis zu erzeugen,

$$\pi_0(x) = \chi_{[0,1]}(x)$$



$$\pi_k(x) = \int_{-\infty}^{+\infty} \pi_{k-1}(x-y)\pi_0(y)dy$$

$$\sum_{i=1}^{k} \pi_k(i) = 1$$

Die Kanonischen Splines

- Die Kanonischen Splines explizit:

$$\pi_0(x) = \begin{cases} 1 & x \in [0,1] \\ 0 & \text{sonst} \end{cases}$$

$$\pi_1(x) = \begin{cases} x & x \in [0,1] \\ 2-x & x \in [1,2] \\ 0 & \text{sonst} \end{cases} = \int_0^1 \pi_0(x-y)dy$$

$$\pi_2(x) = \begin{cases} \frac{1}{2}x^2 & x \in [0,1] \\ \frac{3}{4} - (x-\frac{3}{2})^2 & x \in [1,2] \\ \frac{1}{2}(x-3)^2 & x \in [2,3] \\ 0 & \text{sonst} \end{cases} = \int_0^1 \pi_1(x-y)dy \quad \text{usw}$$

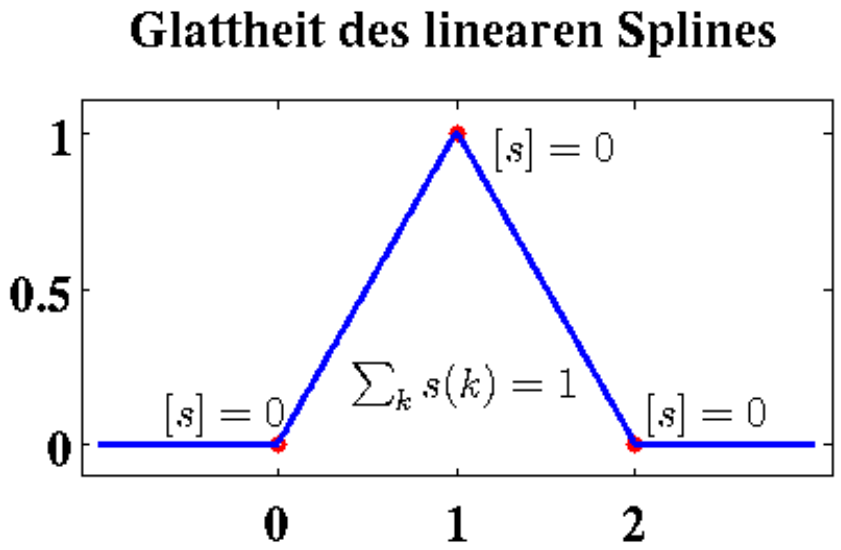
$$\pi_3(x) = \begin{cases} \frac{1}{6}x^3 & x \in [0,1] \\ \frac{1}{6}[1+3(x-1)+3(x-1)^2-3(x-1)^3] & x \in [1,2] \\ \frac{1}{6}[1+3(3-x)+3(3-x)^2-3(3-x)^3] & x \in [2,3] \\ \frac{1}{6}(4-x)^3 & x \in [3,4] \\ 0 & \text{sonst} \end{cases}$$

Glattheitsbedingungen

- Wenn das Gitter gleichmäßig ist, $\boldsymbol{x} = \{x_i\}_{i=0}^{n}$, $h = x_i - x_{i-1}$, ist eine Basis für $\mathcal{S}^k(\boldsymbol{x})$ explizit gegeben durch:

$$s_{k,i}(x) = \pi_k((x - x_i)/h), \quad i = -k, \ldots, n-1$$

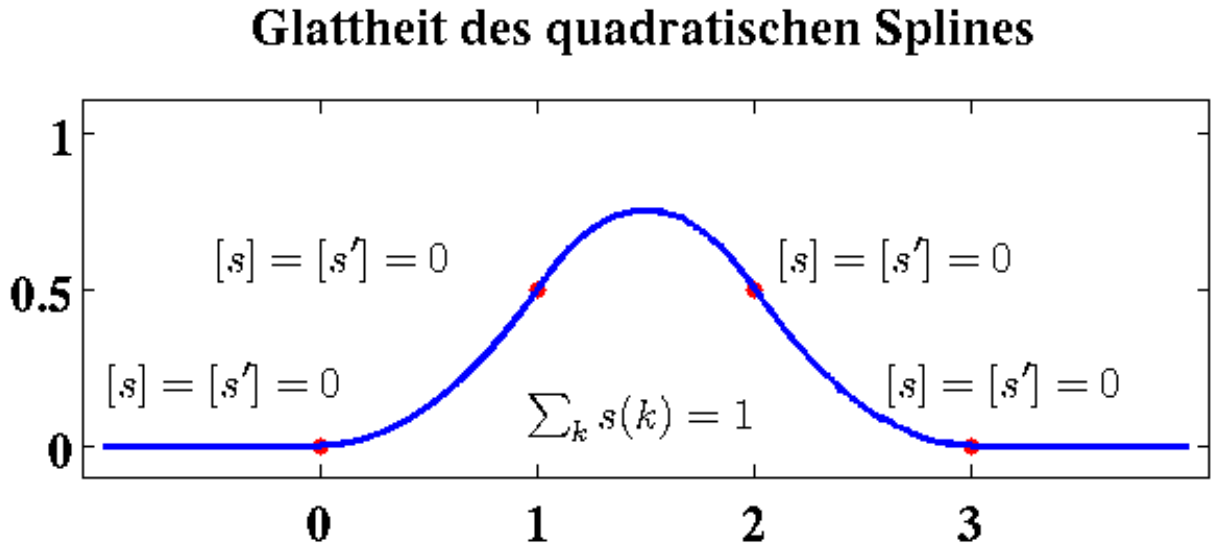
- Im Allgemeinen werden die Basisfunktionen durch Glattheitsbedingungen bestimmt.
- Für die linearen Splines: 4 Unbekannte, 4 Bedingungen,



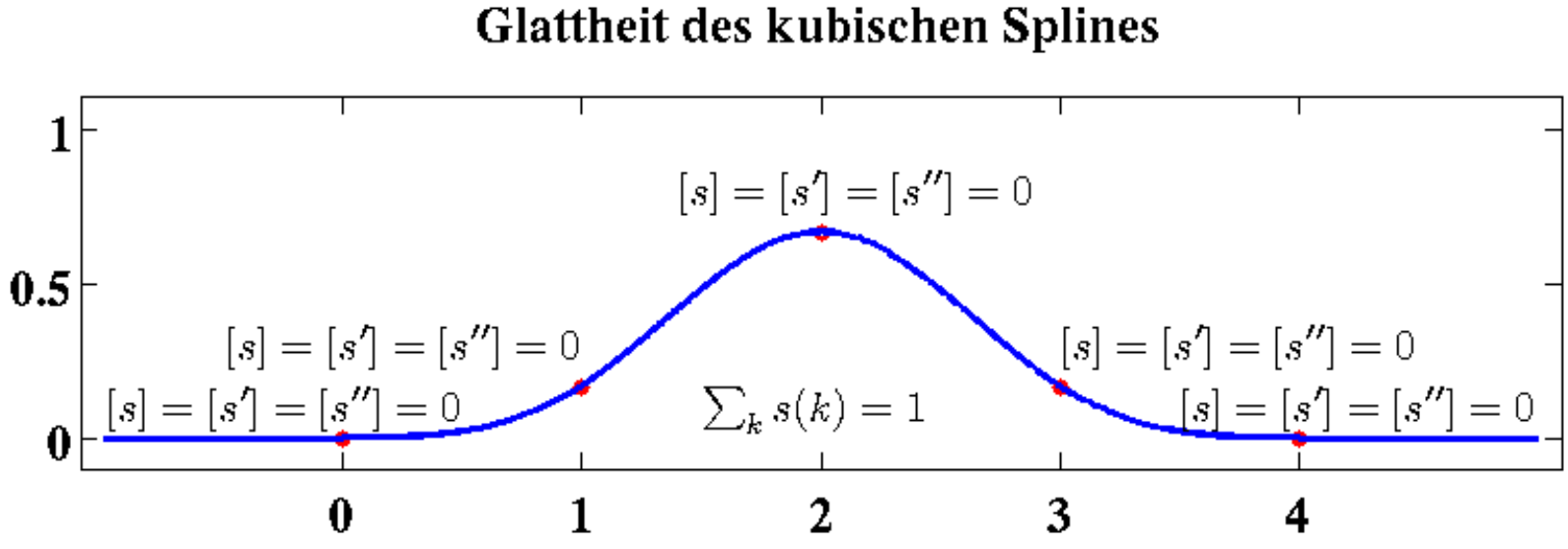
- Hausaufgabe: Schreibe das Gleichungssystem für die Koeffizienten der Splines Grades 1, 2 und 3.

Glattheitsbedingungen

- Für die quadratischen Splines: $9 + 9$



- Für die kubischen Splines: $16 + 16$



Interpolation mit Kubischen Splines

- Gegeben seien die Daten $\{(x_i, f_i)\}_{i=0}^n$ mit

$$\{x_i\}_{i=0}^n = \boldsymbol{x} \in \mathbb{R}^{n+1}.$$

- Zur Vereinfachung wird angenommen, dass das Gitter gleichmäßig ist, obwohl es nicht notwendig ist.
- Die Dimension von $\mathcal{S}^3(\boldsymbol{x})$ ist $n+3$.
- Im Datensatz fehlen noch 2 Informationsstücke, um eine Interpolante

$$s(x) = \sum_{i=-3}^{n-1} \alpha_i s_i(x) \in \mathcal{S}^3$$

zu bestimmen.

- Eine mögliche Ergänzung sind die Randbedingungen:

$$0 = s'(x_0) = \underbrace{s'_{-3}(x_0)}_{=-1/(2h)} \alpha_{-3} + \underbrace{s'_{-2}(x_0)}_{=0} \alpha_{-2} + \underbrace{s'_{-1}(x_0)}_{=1/(2h)} \alpha_{-3} = \frac{\alpha_{-1} - \alpha_{-3}}{2h}$$

und $s'(x_n) = 0$ oder

$$\alpha_{-3} = \alpha_{-1} \text{ und } \alpha_{n-1} = \alpha_{n-3}.$$

Interpolation mit Kubischen Splines

- Mit den Randbedingungen:

$$0 = s''(x_0) = \underbrace{s''_{-3}(x_0)}_{=1/h^2}\alpha_{-3} + \underbrace{s''_{-2}(x_0)}_{=-2/h^2}\alpha_{-2} + \underbrace{s''_{-1}(x_0)}_{=1/h^2}\alpha_{-1} = \frac{\alpha_{-1} - 2\alpha_{-2} + \alpha_{-3}}{h^2}$$

und $s''(x_n) = 0$ ergeben sich

$$\alpha_{-3} = 2\alpha_{-2} - \alpha_{-1} \quad \text{und} \quad \alpha_{n-1} = 2\alpha_{n-2} - \alpha_{n-3}.$$

Durch

$$\begin{aligned} f_0 = s(x_0) &= s_{-3}(x_0)\alpha_{-3} + s_{-2}(x_0)\alpha_{-2} + s_{-1}(x_0)\alpha_{-3} \\ &= \tfrac{1}{6}\alpha_{-3} + \tfrac{2}{3}\alpha_{-2} + \tfrac{1}{6}\alpha_{-3} \\ &= \tfrac{1}{6}(2\alpha_{-2} - \alpha_{-1}) + \tfrac{2}{3}\alpha_{-2} + \tfrac{1}{6}\alpha_{-3} = \alpha_{-2} \end{aligned}$$

ergibt sich mit $\alpha_{-1} - \alpha_{-3} = \alpha_{-1} - (2\alpha_{-2} - \alpha_{-1})$,

$$s'(x_0) = \frac{\alpha_{-1} - \alpha_{-3}}{2h} = \frac{\alpha_{-1} - \alpha_{-2}}{h} = \frac{\alpha_{-1} - f_0}{h} \approx f'(x_0)$$

und ähnlich $s'(x_n) \approx f'(x_n)$.

Interpolation mit Kubischen Splines

- Um die Interpolante $s = \sum_{i=-3}^{n-1} \alpha_i s_i(x)$ zu bestimmen, werden die Koeffizienten $\{\alpha_i\}_{i=-3}^{n-1}$ durch die Lösung des linearen Gleichungssystems berechnet:

$$f_j = s(x_j) = \alpha_{-3} s_{-3}(x_j) + \sum_{i=-2}^{n-2} \alpha_i s_i(x_j) + \alpha_{n-1} s_{n-1}(x_j), \quad j = 0, \ldots, n$$

mit Randbedingungen, z.B.

$$\begin{array}{rclcrcl} \alpha_{-3} & = & \alpha_{-1} & \text{oder} & \alpha_{-3} & = & 2\alpha_{-2} - \alpha_{-1} \\ \alpha_{n-1} & = & \alpha_{n-3} & & \alpha_{n-1} & = & 2\alpha_{n-2} - \alpha_{n-3} \end{array}$$

- Bemerke:

$$s_i(x_j) = 0, j < i+1; \quad s_i(x_{i+1}), s_i(x_{i+2}), s_i(x_{i+3}) \neq 0; \quad s_i(x_j) = 0, j > i+3$$

Die Matrix

$$\{s_i(x_j) : -3 \leq i \leq n-1, -1 \leq j \leq n+1\}$$

ist tridiagonal!

- Die entsprechende Matrix für lineare Splines ist diagonal!

Interpolation mit Kubischen Splines

- Das System für kubische Splines sieht so aus:

$$\begin{bmatrix} f_0 \\ \vdots \\ \vdots \\ f_n \end{bmatrix} = \begin{bmatrix} 1-\theta & \theta & & & \\ \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & & \\ & \ddots & \ddots & \ddots & \\ & & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\ & & & \theta & 1-\theta \end{bmatrix} \begin{bmatrix} \alpha_{-2} \\ \vdots \\ \vdots \\ \alpha_{n-2} \end{bmatrix}$$

wobei $\theta = \frac{1}{3}$ für die Randbedingungen $s'(x_0) = 0 = s'(x_n)$
und $\theta = 0$ für die Randbedingungen $s''(x_0) = 0 = s''(x_n)$.

Satz: Gegeben seien $\{x_i\}_{i=0}^n$ mit $h = \max_i(x_{i+1} - x_i)$ und $f \in \mathcal{C}^1([x_0, x_n])$. Dann $\exists! s \in \mathcal{S}^3(\boldsymbol{x})$ mit $s(x_i) = f(x_i)$, $0 \leq i \leq n$, und $s'(x_0) = f'(x_0)$, $s'(x_n) = f'(x_n)$. Für $f \in \mathcal{C}^4([x_0, x_n])$ $\exists c_r \neq c_r(h)$ mit

$$\max_{x \in [x_0, x_n]} |D^r[f(x) - s(x)]| \leq c_r h^{4-r} \max_{x \in [x_0, x_n]} |D^4 f(x)|, \quad 0 \leq r \leq 3.$$

Tensor Produkte für 2D Interpolation

- Gegeben seien Daten $\{(x_i, y_j, f_{ij})\}_{i,j=0}^{n}$ auf dem 2D Gitter $x_i = ih$, $i = 0, \ldots, n$, $y_i = jh$, $j = 0, \ldots, n$, $h = 1/n$.
- Die Interpolante ist

$$s(x, y) = \sum_{i=-3}^{n-1} \sum_{j=-3}^{n-1} \alpha_{ij} s_i(x) s_j(y)$$

wobei die Koeffizienten $\{\alpha_{ij}\}_{i,j=-3}^{n-1}$ durch Lösung des Gleichungssystems bestimmt werden:

$$f_{kl} = s(x_k, y_l) = \sum_{i=-3}^{n-1} \sum_{j=-3}^{n-1} \alpha_{ij} s_i(x_k) s_j(y_l), \quad 0 \le k, l \le n$$

mit z.B. Randbedingungen, $\theta = 1$ oder 2, $\phi = 1 - \theta$,

$$\begin{aligned} \alpha_{-3,j} &= \theta\alpha_{-2,j} + \phi\alpha_{-1,j}, \quad \alpha_{n-1,j} = \theta\alpha_{n-2,j} + \phi\alpha_{n-3,j}, \quad -3 \le j \le n-1 \\ \alpha_{i,-3} &= \theta\alpha_{i,-2} + \phi\alpha_{i,-1}, \quad \alpha_{i,n-1} = \theta\alpha_{i,n-2} + \phi\alpha_{i,n-3}, \quad -3 \le i \le n-1 \end{aligned}$$

- Solche *Tensor Produkte* können für andere Basisfunktionen ähnlich definiert werden, wie z.B. Stückweise Kubisch Hermite Interpolanten.

Nicht Lineare Gleichungen

- Das Ziel ist, eine Nullstelle zu finden,
$$f(x) = 0$$
wobei f nicht linear ist.
- Wenn $f(x) = F'(x)$ gilt, kann das Ziel sein, F zu minimieren oder maximieren.
- Gegeben seien a und b mit $f(a)f(b) < 0$. Nach dem Zwischenwertsatz $\exists \xi \in (a, b)$ mit $f(\xi) = 0$ wenn $f \in \mathcal{C}([a, b])$.
- Die einfachste Methode ist das Bisektionsverfahren:

$$\begin{array}{c} f(a_k)f(b_k) < 0 \\ c_k = (a_k + b_k)/2 \end{array} \qquad \begin{cases} f(c_k)f(b_k) < 0: & a_{k+1} = c_k, \quad b_{k+1} = b_k \\ f(a_k)f(c_k) < 0: & b_{k+1} = c_k, \quad a_{k+1} = a_k \end{cases}$$

$$\text{Halt mit:} \quad f(c_k) = 0 \quad \text{oder} \quad |b_k - a_k| < \texttt{tol} \cdot |c_k|$$

- In einem Code soll c entweder a oder b überschreiben.
- Solang $f(a)f(b) < 0$ gilt, ist es egal welcher Wert positiv oder negativ ist.

Bisektionsverfahren

- Pseudo-Code für das Bisektionsverfahren

```
fa = f(a), fb = f(b)
if (fa * fb > 0)
   error:  sign(fa) = sign(fb)
end
for k=1,...,kmax
   c = (a + b)/2
   fc = f(c)
   if (|b-a| < tol*|c|) or (fc = 0)
      return c
   end
   if (fa * fc > 0)
      a = c % sign(fa) = sign(fc)
   else
      b = c % sign(fb) = sign(fc)
   end
end
error:  failed to converge to relative
difference tol after kmax iterations
```

Bisektionsverfahren

Satz: Sei $f \in \mathcal{C}([a_1, b_1])$ mit $f(a_1)f(b_1) < 0$. Dann konvergieren die Iterierten $\{c_k\}$ des Bisektionsverfahrens zu einer Nullstelle $x^\star \in (a_1, b_1)$, $f(x^\star) = 0$, und zwar mit Geschwindigkeit:

$$|x^\star - c_k| \leq 2^{-k}|b_1 - a_1|, \quad k \geq 1$$

Beweis: Sei $[a_k, b_k]$ das $k$te Bisektionsintervall mit $c_k = (a_k + b_k)/2$. Es gilt

$$(b_1 - a_1) = 2(b_2 - a_2) = \cdots = 2^{k-1}(b_k - a_k).$$

Aus $a_{k+1} \in \{a_k, c_k\}$ und $b_{k+1} \in \{c_k, b_k\}$ folgen

$$(a_{k+1} - a_k) \leq \tfrac{1}{2}(b_k - a_k) \text{ und } (b_k - b_{k+1}) \leq \tfrac{1}{2}(b_k - a_k)$$

und daher

$$|c_{k+1} - c_k| = \tfrac{1}{2}|(a_{k+1} - a_k) - (b_k - b_{k+1})| \leq$$
$$\tfrac{1}{2}[|a_{k+1} - a_k| + |b_k - b_{k+1}|] \leq \tfrac{1}{2}(b_k - a_k) = 2^{-k}(b_1 - a_1).$$

Also für $k > l$ gilt

$$|c_k - c_l| \leq |c_k - c_{k-1}| + \cdots |c_{l+1} - c_l| = \sum_{m=l}^{k-1} |c_{m+1} - c_m| \leq$$
$$\sum_{m=l}^{k-1} 2^{-m}(b_1 - a_1) = (b_1 - a_1)(2^{-l} - 2^{-k})/(1 - 2^{-1}) \overset{k,l\to\infty}{\longrightarrow} 0.$$

Bisektionsverfahren

Deswegen ist $\{c_k\}$ eine Cauchy Folge. Sei $x^\star$ der Limes.
Aus $c_k \in (a_k, b_k)$ folgen

$$|x^\star - a_k| \leq |x^\star - c_k| + |c_k - a_k| \leq |x^\star - c_k| + |b_k - a_k| \overset{k\to\infty}{\longrightarrow} 0.$$

und

$$|x^\star - b_k| \leq |x^\star - c_k| + |c_k - b_k| \leq |x^\star - c_k| + |a_k - b_k| \overset{k\to\infty}{\longrightarrow} 0.$$

Daher aus $f \in \mathcal{C}([a_1, b_1])$ folgen

$$\lim_{k\to\infty} f(a_k) = f(x^\star) = \lim_{k\to\infty} f(b_k).$$

Aus $f(a_k)f(b_k) \leq 0$, $\forall k$, folgt

$$\lim_{k\to\infty} f(a_k)f(b_k) \leq 0.$$

Diese Limiten implizieren

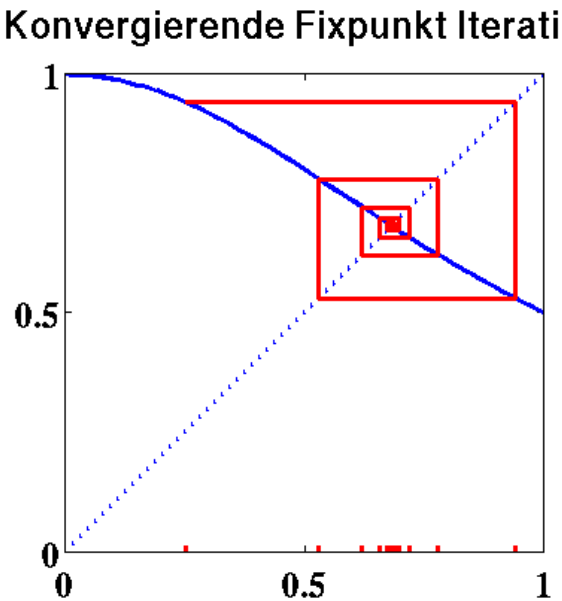
$$0 \leq f(x^\star)^2 = \lim_{k\to\infty} f(a_k)f(b_k) \leq 0$$

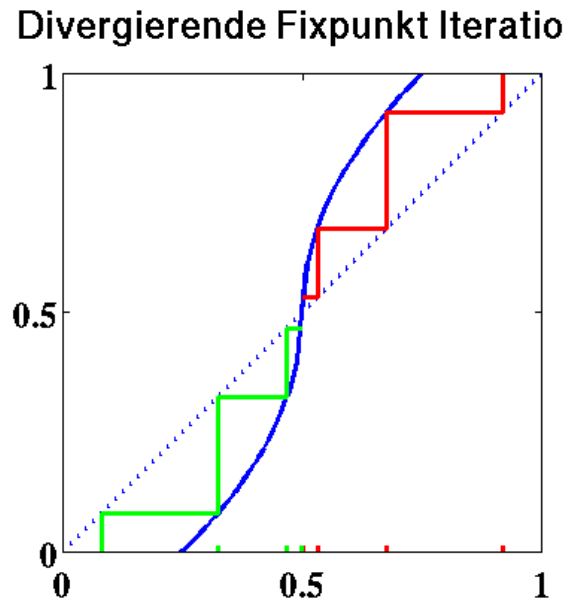
und es folgt $f(x^\star) = 0$. ■

Bemerkung: Obwohl Konvergenz für das Bisektionsverfahren garantiert ist, ist die Geschwindigkeit typischerweise niedriger als die für das Newton Verfahren, das aber nur *lokal* konvergiert.

Fixpunktiteration

- Die Nullstelle in der Gleichung $f(x^\star) = 0$ kann bezüglich eines Fixpunkts $x^\star = g(x^\star)$ einer Funktion $g(x)$ in verschiedenen Weisen umgeschrieben werden:
$$g(x) = x - f(x), \quad g(x) = x - f(x)/f'(x)$$
- Dann gibt es eine natürliche *Fixpunktiteration*:
$$x_{k+1} = g(x_k), \quad k = 0, 1, 2, \ldots$$
- Wann konvergiert diese Iteration? Grafische Darstellung:





- Wichtige Eigenschaft: $|g'(x^\star)| < 1$.

Fixpunktiteration

Satz: Sei $g \in \mathcal{C}([a,b])$ mit $g([a,b]) \subset [a,b]$. Dann $\exists x^\star \in [a,b]$ mit $g(x^\star) = x^\star$. Falls $g \in \mathcal{C}^1([a,b])$ und $\exists \gamma$ mit $|g'(x)| \leq \gamma < 1$, $\forall x \in [a,b]$, dann ist $x^\star$ ein eindeutiger Fixpunkt.

Beweis: $g([a,b]) \subset [a,b] \Rightarrow f(x) := g(x) - x$ erfüllt $f(a) = g(a) - a \geq a - a = 0$ und $f(b) = g(b) - b \leq b - b = 0$. Wenn $f(a) = 0$ gelten sollte, ist $x^\star = a$ ein Fixpunkt, und analog für $x^\star = b$. Nimm an, $f(b) < 0 < f(a)$. Nach dem Zwischenwertsatz $\exists x^\star \in [a,b]$ mit $0 = f(x^\star) = g(x^\star) - x^\star$, da f stetig ist. Wenn es einen zweiten Fixpunkt $\tilde{x} \in [a,b]$ gäbe, dann würde ein Widerspruch durch den Mittelwertsatz entstehen:

$$\begin{aligned} |x^\star - \tilde{x}| &= |g(x^\star) - g(\tilde{x})| = |g'(\xi)(x^\star - \tilde{x})| \\ &\leq \gamma |x^\star - \tilde{x}| \\ &< |x^\star - \tilde{x}| \end{aligned}$$

wobei ξ zwischen $x^\star$ und $\tilde{x}$ in $[a,b]$ liegen würde. ■

Fixpunktiteration

Satz: Sei $g \in \mathcal{C}^1([a,b])$ mit $g([a,b]) \subset [a,b]$. Wenn $\exists\gamma$ mit $|g'(x)| \leq \gamma < 1$, $\forall x \in [a,b]$, dann $\forall x_0 \in [a,b]$ konvergiert die Fixpunktiteration $x_{k+1} = g(x_k)$ zum eindeutigen Fixpunkt $x^\star$.

Beweis: $g([a,b]) \subset [a,b] \Rightarrow x_{k+1}(= g(x_k)) \in [a,b]$, $\forall k$. Nach dem Mittelwertsatz gilt

$$\begin{aligned} |x_{k+1} - x^\star| &= |g(x_k) - g(x^\star)| = |g'(\xi_k)(x_k - x^\star)| \\ &\leq \gamma|x_k - x^\star| \leq \gamma^2|x_{k-1} - x^\star| \leq \cdots \\ &\leq \gamma^{k+1}|x_0 - x^\star| \overset{k\to\infty}{\longrightarrow} 0 \end{aligned}$$

wobei ξ_k zwischen $x^\star$ und x_k in $[a,b]$ liegt. ■

Satz: Unter den Bedingungen des letzten Satzes gilt

$$|x^\star - x_k| \leq \frac{\gamma^k}{1-\gamma}|x_1 - x_0|, \quad k \geq 1$$

Beweis: Hausaufgabe. ■

Satz: Für $g \in \mathcal{C}^1([a,b])$ mit $g(x^\star) = x^\star \in (a,b)$ und $|g'(x)| > 1$, $\forall x \in [a,b]$, konvergiert $\{x_{k+1} = g(x_k)\}$ zu $x^\star$ nicht wenn $x_0 \neq x^\star$.

Fixpunktiteration

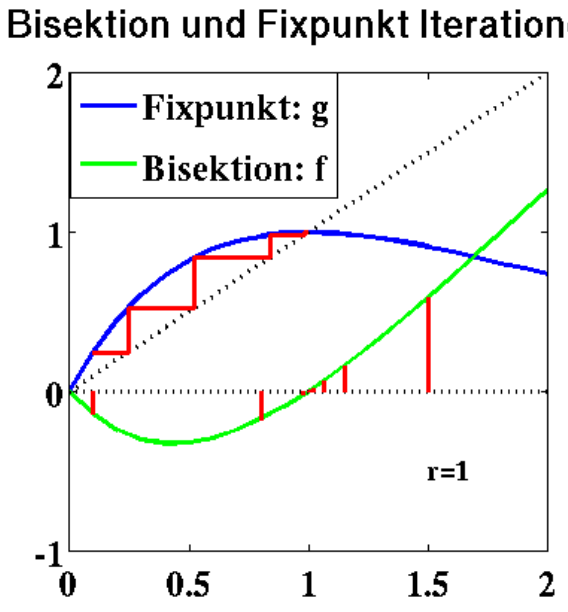
- Pseudo-Code für eine Fixpunktiteration

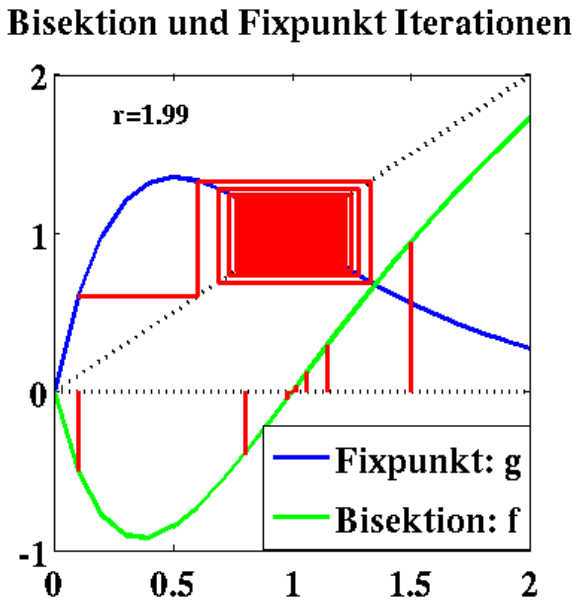
```
x0 gegeben
for k=1,...,kmax
   x = g(x0)
   if (|x-x0| < tol*|x|)
      return x
   end
   x0 = x
end
error:  failed to converge to relative
difference tol after kmax iterations
```

- Welche Iteration ist schneller, Bisektion oder Fixpunkt?
  Hängt von der Funktion ab: $g'(x^\star) = 0$ favorisiert Fixpunkt.
- Beispiel: $g(x) = xe^{r(1-x)}$, $r \geq 1$, $f(x) = x - g(x)$.
  Entspricht einem diskreten `Populations-Modell`.
- Hausaufgabe: Zeige, für $r \in [1, 2)$,
  - $\exists \epsilon > 0 \ni g(B(1, \epsilon)) \subset B(1, \epsilon)$ und
  - $\exists \gamma \in (0, 1) \ni |g'(x)| \leq \gamma$, $\forall x \in B(1, \epsilon)$.

Vergleich zweichen Bisektions- und Fixpunktiteration

- Es gelten $g(x) = xe^{r(1-x)}$ und $f(x) = x - g(x)$.
- Für die erste Grafik ($r = 1, g'(1) = 0$) verlangt Bisektion 12 Iterationen und Fixpunkt 6 Iterationen, um $x = 0.99995$ bzw. $x = 0.999999995$ zu berechnen.





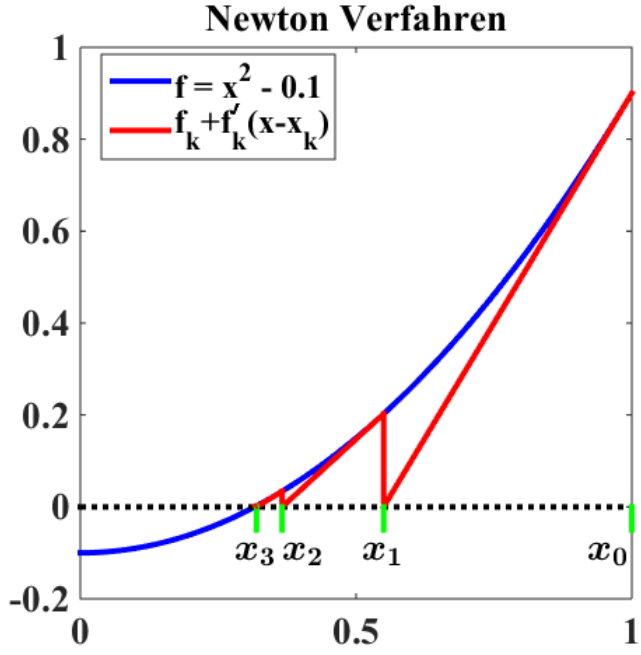
- Für die zweite Grafik ($r = 1.99, g'(x^\star) = -0.99$) verlangt Bisektion 12 Iterationen und Fixpunkt 542 Iterationen, um $x = 0.99995$ bzw. $x = 1.0005$ zu berechnen.

Newton Verfahren

- Die Strategie: Eine Nullstelle wird durch eine Folge von linearen Approximationen der Funktion bestimmt:
- Für ein gegebenes x_0 ist

$$y - f(x_0) = f'(x_0)(x - x_0)$$

die zu f tangente Gerade an der Stelle $(x_0, f(x_0))$.



- Die Nullstelle $(x_1, 0)$ dieser Gerade erfüllt

$$-f(x_0) = f'(x_0)(x_1 - x_0)$$

oder

$$x_1 = x_0 - f(x_0)/f'(x_0)$$

- Im Allgemeinen werden die Newton Iterierten so gegeben:

$$x_{k+1} = x_k - f(x_k)/f'(x_k), \quad k = 0, 1, 2, \ldots$$

- Wichtig ist, dass $f'(x^\star) \neq 0$.

Newton Verfahren

- Pseudo-Code für das Newton Verfahren

```
f(x), fp(x) = f'(x), x0 gegeben
for k=1,...,kmax
   f0 = f(x0)
   fp0 = fp(x0)
   if (fp0 =/= 0)
      x1 = x0 - f0/fp0
   else
      x1 = x0
   end
   if (|x1-x0| < tol*|x1|)
      return x1
   end
   x0 = x1
end
error:  failed to converge to relative
difference tol after kmax iterations
```

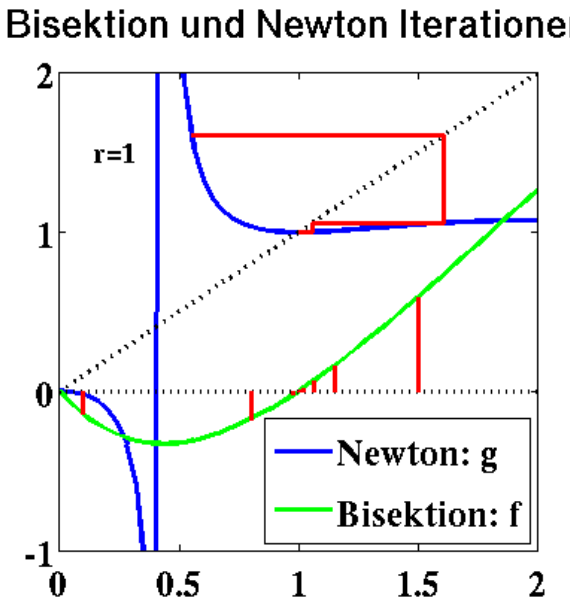
Sekant Verfahren

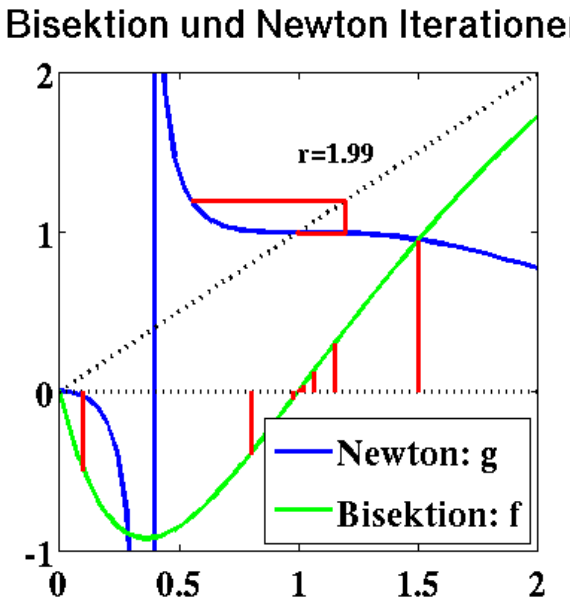
- Pseudo-Code für das Sekant Verfahren

```
f(x), x0 =/= x1 gegeben
f1 = f(x0)
for k=1,...,kmax
   f0 = f1
   f1 = f(x1)
   if (f1 =/= f0)
      x2 = x1 - f1 * (x1 - x0)/(f1 - f0)
   else
      x2 = x1
   end
   if (|x2-x1| < tol*|x2|)
      return x2
   end
   x0 = x1
   x1 = x2
end
error:  failed to converge to relative
difference tol after kmax iterations
```

Vergleich zweichen Newton und Bisektionsiteration

- Es gelten $f(x) = x - xe^{r(1-x)}$ und $g(x) = x - f(x)/f'(x)$.
- Für die erste Grafik ($r = 1$) verlangt Bisektion 12 Iterationen und Newton 5 Iterationen, um $x = 0.99995$ bzw. $x = 1.000000000006$ zu berechnen.





- Für die zweite Grafik ($r = 1.99$) verlangt Bisektion 12 Iterationen und Newton 4 Iterationen, um $x = 0.99995$ bzw. $x = 1.\bar{0}!$ zu berechnen.

Konvergenz des Newton Verfahrens

Satz: Sei $f \in \mathcal{C}^2([a,b])$ mit $f(x^\star) = 0$ und $f'(x^\star) \neq 0$ für $x^\star \in (a,b)$. Dann $\exists \epsilon > 0$ wobei $\forall x_0 \in B(x^\star, \epsilon)$ die Newton Iterierten $\{x_n\}$ zu $x^\star$ konvergieren.

Beweis: Die Funktion $g(x) = x - f(x)/f'(x)$ erfüllt $g(x^\star) = x^\star$. Da $f'(x^\star) \neq 0$ und $f' \in \mathcal{C}([a,b])$, $\exists \delta > 0$ mit $f'(x) \neq 0$, $\forall x \in B(x^\star, \delta)$. Deswegen ist $g(x)$ in $B(x^\star, \delta)$ wohl definiert und stetig. Aus der Rechnung

$$g'(x) = 1 - (f'(x)^2 - f(x)f''(x))/f'(x)^2 = f(x)f''(x)/f'(x)^2$$

folgt $g' \in \mathcal{C}(B(x^\star, \delta))$ und $g'(x^\star) = 0$. Deswegen $\exists \epsilon > 0$ und $\gamma \in (0,1)$ mit $|g'(x)| \leq \gamma, \forall x \in B(x^\star, \epsilon)$. Nun sei $x \in B(x^\star, \epsilon)$. Nach dem Mittelwertsatz $\exists \xi$ zwischen x und $x^\star$ mit

$$\begin{aligned} |g(x) - g(x^\star)| &= |g'(\xi)(x - x^\star)| \\ &\leq \gamma |x - x^\star| < \epsilon \end{aligned}$$

Es folgt $g(B(x^\star, \epsilon)) \subset B(x^\star, \epsilon)$. So sind die Bedingungen im Satz 170 für Konvergenz zum eindeutigen Fixpunkt erfüllt. ■

Asymptotische Konvergenzrate

Bemerkung: Für eine Fixpunktiteration mit $g''(x^\star) \neq 0$ gibt es die bestmögliche lokale Konvergenz wenn $g'(x^\star) = 0$ gilt.

Bemerkung: Das Newton Verfahren konvergiert typischerweise nur lokal, d.h. nur wenn $|x^\star - x_0|$ ausreichend klein ist. Das Bisektionsverfahren konvergiert für beliebige Startwerte, solang die anfänglichen Vorzeichen verschieden sind.

Bemerkung: Um die jeweiligen Vorteile des Bisektions- und des Newton Verfahrens zu kombinieren, kann zwischen diesen gewechselt werden. Eine dieser zwei Methoden soll verwendet werden, bis eine niedrige *Konvergenzrate* erkannt wird, und dann soll auf die andere Methode umgeschaltet werden.

Def: Angenomen gilt $x_k \overset{k\to\infty}{\longrightarrow} x^\star$. Wenn für $\alpha, \lambda \in (0, \infty)$ gilt

$$\lim_{k\to\infty} \frac{|x^\star - x_{k+1}|}{|x^\star - x_k|^\alpha} = \lambda$$

ist α die *Konvergenzordnung* mit *asymptotischer Fehlerkonstante* λ.

Asymptotische Konvergenzrate

- Beispiele:

  $\alpha = 1 \Rightarrow$ aymptotisch lineare Konvergenzrate,

  z.B. $x_{k+1} = g(x_k)$, $0 < |g'(x^\star)| < 1$

  $\alpha = 2 \Rightarrow$ aymptotisch quadratische Konvergenzrate,

  z.B. $x_{k+1} = g(x_k)$, $|g'(x^\star)| = 0$

Satz: Sei $g \in \mathcal{C}^1([a,b])$ mit $g([a,b]) \subset [a,b]$, $g(x^\star) = x^\star \in (a,b)$ und $|g'(x)| \leq \gamma < 1$, $\forall x \in [a,b]$. Wenn $g'(x^\star) \neq 0$, konvergiert $\{x_{k+1} = g(x_k)\}$ asymptotisch nur linear zu $x^\star$.

Beweis: Sei $x_0 \in [a,b]$. Aus $g([a,b]) \subset [a,b]$ folgt $\{x_k\} \subset [a,b]$. Nach dem Mittelwertsatz, $\exists \xi_k \in [a,b]$ zwischen x_k und $x^\star$ mit

$$|x_{k+1} - x^\star| = |g(x_k) - g(x^\star)| = |g'(\xi_k)||x_k - x^\star|$$

Nach dem Satz 170 gilt $x_k \overset{k\to\infty}{\longrightarrow} x^\star$, und daher gilt auch $\xi_k \overset{k\to\infty}{\longrightarrow} x^\star$. Mit $g \in \mathcal{C}^1([a,b])$ folgt $g'(\xi_k) \overset{k\to\infty}{\longrightarrow} g'(x^\star)$ und daher

$$\frac{|x^\star - x_{k+1}|}{|x^\star - x_k|} = |g'(\xi_k)| \overset{k\to\infty}{\longrightarrow} |g'(x^\star)| \in (0,1). \qquad \blacksquare$$

Asymptotische Konvergenzrate

Satz: Sei $g \in \mathcal{C}^2([a,b])$ mit $g(x^\star) = x^\star \in (a,b)$ und $g'(x^\star) = 0$. Dann $\exists \epsilon > 0$ wobei $\forall x_0 \in B(x^\star, \epsilon)$ die Folge $\{x_{k+1} = g(x_k)\}$ mindestens asymptotisch quadratisch zu $x^\star$ konvergiert.

Beweis: Wähle $\epsilon > 0$ so aus, dass $|g'(x)| \leq \gamma < 1$, $\forall x \in B(x^\star, \epsilon) \subset [a,b]$. Nach dem Mittelwertsatz $\exists \xi$ zwischen $x \in B(x^\star, \epsilon)$ und $x^\star$ mit

$$|g(x) - x^\star| = |g(x) - g(x^\star)| = |g'(\xi)||x - x^\star| \leq \gamma\epsilon < \epsilon.$$

Es folgt $g(B(x^\star, \epsilon)) \subseteq B(x^\star, \epsilon)$. Daher gilt $\{x_k\}_{k\in\mathbb{N}} \subset B(x^\star, \epsilon)$ wenn $x_0 \in B(x^\star, \epsilon)$. Nach dem Satz 170 gilt $x_k \overset{k\to\infty}{\longrightarrow} x^\star$. Nach dem Taylorsatz $\exists \eta_k \in B(x^\star, \epsilon)$ zwischen x_k und $x^\star$ mit

$$x_{k+1} = g(x_k) = \underbrace{g(x^\star)}_{=x^\star} + \underbrace{g'(x^\star)}_{=0}(x_k - x^\star) + \tfrac{1}{2}g''(\eta_k)(x_k - x^\star)^2$$

Mit $|\eta_k - x^\star| \leq |x_k - x^\star| \overset{k\to\infty}{\longrightarrow} 0$ und $g \in \mathcal{C}^2([a,b])$ folgt

$$\frac{|x^\star - x_{k+1}|}{|x^\star - x_k|^2} = \tfrac{1}{2}|g''(\eta_k)| \overset{k\to\infty}{\longrightarrow} \tfrac{1}{2}|g''(x^\star)| = \lambda < \infty. \qquad \blacksquare$$

Bemerkung: Mit $g''(x^\star) = 0$ ist α noch höher.

Nullstellen höherer Multiplizität

- Beispiel: $f'(x^\star) \neq 0 \Rightarrow g(x) = x - f(x)/f'(x)$ erfüllt
$$g'(x) = 1 - \frac{f'(x)^2 - f(x)f''(x)}{f'(x)^2} = \frac{f(x)f''(x)}{f'(x)^2} \xrightarrow{x \to x^\star} 0$$
Was ist wenn $f'(x^\star) = 0$?

Def: Die Nullstelle, $f(x^\star) = 0$, hat *Multiplizität m*, wenn m die höchste Zahl ist, die erfüllt $\lim_{x \to x^\star} |f(x)|/|x - x^\star|^m \in (0, \infty)$. Bei $m = 1$ ist die Nullstelle *einfach*.

- Kann Newton asymptotisch quadratisch konvergieren, wenn $x^\star$ keine einfache Nullstelle für f ist? Im Allgemeinen *nicht*.
- Beispiel: $f(x) = e^x - x - 1$, $x^\star = 0$, $m = 2$, $g(x) = x - f(x)/f'(x)$ aber $g'(0) = \frac{1}{2} \neq 0$!
- Wenn die Nullstelle $f(x^\star) = 0$ Multiplizität $m > 1$ hat, verwende das modifizierte Newton Verfahren:
$$g_m(x) = x - mf(x)/f'(x)$$
- Hausaufgabe: Zeige asymptotisch quadratische Konvergenz für g_m.

Systeme von Nicht Linearen Gleichungen

- Beispiel: Minimiere

$$\begin{aligned} J_h(\boldsymbol{u}) &= \frac{h}{2}\sum_{i=1}^{N}(u_i - v_i)^2 + \mu h \sum_{i=1}^{N-1}\left[\left(\frac{u_{i+1}-u_i}{h}\right)^2 + \epsilon^2\right]^{\frac{1}{2}} \\ &\approx \frac{1}{2}\int_\Omega (u-v)^2 dx + \mu\int_\Omega \sqrt{|u'|^2+\epsilon^2}\,dx \end{aligned}$$

- Für $\nabla J_h(\boldsymbol{u}) = 0$ sei $D_i(\boldsymbol{u}) = \left[\left(\frac{u_{i+1}-u_i}{h}\right)^2 + \epsilon^2\right]^{-\frac{1}{2}} = D_i$

$$\underbrace{\frac{\partial J_h}{\partial u_k}}_{1<k<N} = h(u_k - v_k) + \mu/h[\underbrace{(u_k - u_{k-1})D_{k-1}}_{i=k-1} - \underbrace{(u_{k+1}-u_k)D_k}_{i=k}]$$

$$\frac{\partial J_h}{\partial u_1} = h(u_0 - v_0) + \mu/h[\quad - \underbrace{(u_2-u_1)D_1}_{i=1}]$$

$$\frac{\partial J_h}{\partial u_N} = h(u_N - v_N) + \mu/h[\underbrace{(u_N - u_{N-1})D_{N-1}}_{i=N-1}\quad]$$

Optimalitätssystem

- Die Optimalitätsbedingung ist

$$0 = \frac{1}{h}\nabla J_h(\boldsymbol{u}) = \boldsymbol{u} - \boldsymbol{v} + \frac{\mu}{h^2} D(\boldsymbol{u})\boldsymbol{u} =: \boldsymbol{F}(\boldsymbol{u})$$

wobei

$$D(\boldsymbol{u}) = \begin{bmatrix} D_1 & -D_1 & & & 0 \\ -D_1 & D_1 + D_2 & -D_2 & & \\ & \ddots & \ddots & \ddots & \\ & & -D_{N-2} & D_{N-2} + D_{N-1} & -D_{N-1} \\ 0 & & & -D_{N-1} & D_{N-1} \end{bmatrix} \in \mathbb{R}^{N \times N}$$

- Gesucht wird eine Nullstelle $\boldsymbol{F}(\boldsymbol{u}^\star) = 0$.
- Ein Ansatz ist, eine Fixpunktiteration $\boldsymbol{u}_{k+1} = \boldsymbol{G}(\boldsymbol{u}_k)$ zu verwenden, wobei $\boldsymbol{G}(\boldsymbol{u}) = \boldsymbol{u} - \boldsymbol{F}(\boldsymbol{u})$ oder

$$\boldsymbol{G}(\boldsymbol{u}) = \boldsymbol{v} - \frac{\mu}{h^2} D(\boldsymbol{u})\boldsymbol{u}$$

Konvergenz der Fixpunktiteration

Satz: Für $\bar{B} = B \Subset \mathbb{R}^N$ sei $\boldsymbol{G} \in \mathcal{C}(B, \mathbb{R}^N)$ mit $\boldsymbol{G}(B) \subseteq B$. Dann $\exists \boldsymbol{u}^\star \in B$ mit $\boldsymbol{G}(\boldsymbol{u}^\star) = \boldsymbol{u}^\star$. Wenn $\boldsymbol{G} \in \mathcal{C}^1(B, \mathbb{R}^N)$ und $\rho(\boldsymbol{G}'(\boldsymbol{u})) \leq \gamma < 1$, $\forall \boldsymbol{u} \in B$, gelten, ist $\boldsymbol{u}^\star$ ein eindeutiger Fixpunkt in B. Wenn die Abschätzung $\|\boldsymbol{G}'(\boldsymbol{u})\|_* \leq \gamma < 1$, $\forall \boldsymbol{u} \in B$, für eine induzierte Matrixnorm $\|\cdot\|_*$ gilt, folgt für die Fixpunktiteration $\{\boldsymbol{u}_{k+1} = \boldsymbol{G}(\boldsymbol{u}_k)\}$,

$$\|\boldsymbol{u}^\star - \boldsymbol{u}_k\|_* \leq \frac{\gamma^k}{1-\gamma}\|\boldsymbol{u}_1 - \boldsymbol{u}_0\|_* \quad k \geq 1$$

- Beispiel: Es wird für $\boldsymbol{G}(\boldsymbol{u}) = \boldsymbol{v} - (\mu/h^2)D(\boldsymbol{u})\boldsymbol{u}$ gezeigt, die Bedingungen dieses Satzes werden mit $B = \bar{B}_\infty(\boldsymbol{v}, \frac{2\mu}{h})$ erfüllt, und es gilt $\|\boldsymbol{G}'(\boldsymbol{u})\|_\infty \leq \frac{4\mu}{\epsilon h^2}$, $\forall \boldsymbol{u} \in B$.
- Mit $|D_i(u_{i+1} - u_i)/h| \leq 1$ und

$$|[\boldsymbol{G}(\boldsymbol{u}) - \boldsymbol{v}]_i| = \left|-\frac{\mu}{h^2}[D(\boldsymbol{u})\boldsymbol{u}]_i\right| = \frac{\mu}{h}\left|D_{i-1}\frac{u_i - u_{i-1}}{h}\delta_{i>1} - D_i\frac{u_{i+1} - u_i}{h}\delta_{i<N}\right| \leq \frac{2\mu}{h}$$

folgt $\|\boldsymbol{G}(\boldsymbol{u}) - \boldsymbol{v}\|_\infty \leq \frac{2\mu}{h}$, $\forall \boldsymbol{u} \in \mathbb{R}^N$, d.h. $\boldsymbol{G}(B) \subseteq B$.

Konvergenz der Fixpunktiteration

- Um $\|\boldsymbol{G}'(\boldsymbol{u})\|_\infty \leq \frac{4\mu}{\epsilon h^2}$ zu zeigen, wird abgeleitet,

$$G_i = v_i - \frac{\mu}{h}\left[D_{i-1}\frac{u_i - u_{i-1}}{h}\delta_{i>1} - D_i\frac{u_{i+1} - u_i}{h}\delta_{i<N}\right]$$

- Beispielsweise für u_{i-1},

$$\begin{aligned}\frac{\partial G_i}{\partial u_{i-1}} &= -\frac{\mu}{h}\left[D_{i-1}^3\left(\frac{u_i - u_{i-1}}{h}\right)^2\frac{1}{h} - D_{i-1}\frac{1}{h}\right]\delta_{i>1} \\ &= \frac{\mu}{h^2}D_{i-1}^3\left[D_{i-1}^{-2} - \left(\frac{u_i - u_{i-1}}{h}\right)^2\right]\delta_{i>1} = \frac{\mu\epsilon^2}{h^2}D_{i-1}^3\delta_{i>1}\end{aligned}$$

- Ähnlicherweise,

$$\frac{\partial G_i}{\partial u_{i+1}} = \frac{\mu\epsilon^2}{h^2}D_{i+1}^3\delta_{i<N}, \qquad \frac{\partial G_i}{\partial u_i} = -\frac{\mu\epsilon^2}{h^2}(D_i^3\delta_{i>1} + D_{i+1}^3\delta_{i<N})$$

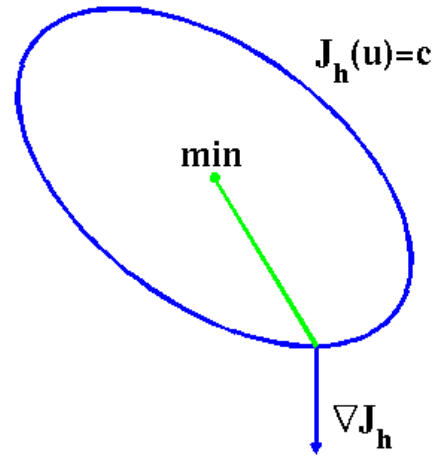
- Mit $\epsilon D_i \leq 1$, folgt $\|\boldsymbol{G}'(\boldsymbol{u})\|_\infty \leq \frac{4\mu}{\epsilon h^2}$, $\forall \boldsymbol{u} \in \mathbb{R}^N$.

Abstiegsverfahren

- Nach dem Satz 184 gibt es genau ein $\boldsymbol{u}^\star \in B$ mit $\boldsymbol{G}(\boldsymbol{u}^\star) = \boldsymbol{u}^\star$, und es gilt

$$\|\boldsymbol{u}^\star - \boldsymbol{u}_k\|_\infty \leq \frac{\gamma^k}{1-\gamma}\|\boldsymbol{u}_1 - \boldsymbol{u}_0\|_\infty$$

solang $4\mu/(\epsilon h^2) \leq \gamma < 1$.



Minimierung durch den steilsten Abstieg

- Die Suchrichtung ist $-\nabla J_h(\boldsymbol{u})/h$
$$= \boldsymbol{F}(\boldsymbol{u}) = \boldsymbol{u} - \boldsymbol{G}(\boldsymbol{u})$$

$$\begin{aligned} \boldsymbol{u}_{k+1} &= \boldsymbol{u}_k - \alpha\boldsymbol{F}(\boldsymbol{u}_k) \\ &= (1-\alpha)\boldsymbol{u}_k + \alpha\boldsymbol{G}(\boldsymbol{u}_k) \\ &=: \boldsymbol{G}_{\mathrm{A}}(\boldsymbol{u}_k) \end{aligned}$$

- Hausaufgabe: Zeige für $\alpha \in (0,1)$, es gilt $\boldsymbol{G}_{\mathrm{A}}(B) \subseteq B$ für $B = \bar{B}_\infty(\boldsymbol{v}, \frac{2\mu}{h})$ und $\|\boldsymbol{G}'_{\mathrm{A}}(\boldsymbol{u})\|_\infty \leq 1 - \alpha + \frac{4\alpha\mu}{\epsilon h^2}$, $\forall \boldsymbol{u} \in B$.

Implizite Fixpunktiteration

- Die Optimalitätsbedingung,

$$0 = \frac{1}{h}\nabla J_h(\boldsymbol{u}) = \boldsymbol{u} - \boldsymbol{v} + \frac{\mu}{h^2} D(\boldsymbol{u})\boldsymbol{u}$$

  kann auch so umgeschrieben werden:

$$\left[I + \frac{\mu}{h^2} D(\boldsymbol{u})\right] \boldsymbol{u} = \boldsymbol{v}$$

- Dann ist die Iteration

$$\left[I + \frac{\mu}{h^2} D(\boldsymbol{u}_k)\right] \boldsymbol{u}_{k+1} = \boldsymbol{v}$$

  eine implizite Fixpunktiteration mit der Fixpunktfunktion:

$$\boldsymbol{G}_{\mathrm{I}}(\boldsymbol{u}) = \left[I + \frac{\mu}{h^2} D(\boldsymbol{u})\right]^{-1} \boldsymbol{v}$$

- `Hausaufgabe`: Zeige, es gilt $\boldsymbol{G}_{\mathrm{I}}(B) \subseteq B$ für $B = \bar{B}_2(0, \|\boldsymbol{v}\|_2)$ und $\|\boldsymbol{G}'_{\mathrm{I}}(\boldsymbol{u})\|_2 \leq \frac{4\mu}{h^3\epsilon^2}\|\boldsymbol{v}\|_2$, $\forall \boldsymbol{u} \in B$.

Newton Verfahren für Systeme

- Das Newton Verfahren ist gegeben durch

$$\boldsymbol{F}'(\boldsymbol{u}_k)(\boldsymbol{u}_{k+1} - \boldsymbol{u}_k) = -\boldsymbol{F}(\boldsymbol{u}_k), \quad k = 0, 1, 2, \ldots$$

Es gelten

$$\boldsymbol{F}(\boldsymbol{u}) = \boldsymbol{u} - \boldsymbol{v} + \frac{\mu}{h^2} D(\boldsymbol{u})\boldsymbol{u} = \boldsymbol{u} - \boldsymbol{G}(\boldsymbol{u}), \quad \boldsymbol{F}'(\boldsymbol{u}) = I - \boldsymbol{G}'(\boldsymbol{u})$$

und

$$-\boldsymbol{G}'(\boldsymbol{u}) = \frac{\mu\epsilon^2}{h^2} E(\boldsymbol{u}), \quad E(\boldsymbol{u}) = \begin{bmatrix} D_1^3 & -D_1^3 & & & \\ -D_1^3 & D_1^3 + D_2^3 & -D_2^3 & & \\ & \ddots & \ddots & \ddots & \\ & & -D_{N-2}^3 & D_{N-2}^3 + D_{N-1}^3 & -D_{N-1}^3 \\ & & & -D_{N-1}^3 & D_{N-1}^3 \end{bmatrix}$$

- Die Gerschgorin Scheiben für $\boldsymbol{F}'(\boldsymbol{u})$ sind $B(z_i, r_i)$ wobei

$$\begin{aligned} z_1 &= 1 + \tfrac{\mu\epsilon^2}{h^2} D_1^3 & r_1 &= \tfrac{\mu\epsilon^2}{h^2} D_1^3 \\ z_i &= 1 + \tfrac{\mu\epsilon^2}{h^2}(D_{i-1}^3 + D_i^3) & r_i &= \tfrac{\mu\epsilon^2}{h^2}(D_{i-1}^3 + D_i^3) \\ z_N &= 1 + \tfrac{\mu\epsilon^2}{h^2} D_{N-1}^3 & r_N &= \tfrac{\mu\epsilon^2}{h^2} D_{N-1}^3 \end{aligned}$$

- Da $\epsilon D_i \leq 1$ gilt, folgt $1 \leq z_i \pm r_i \leq 1 + 4\mu/(\epsilon h^2)$, $\forall i$.
- Es folgt, $\boldsymbol{F}'(\boldsymbol{u}) = I + \frac{\mu\epsilon^2}{h^2} E(\boldsymbol{u})$ ist SPD $\forall \boldsymbol{u} \in \mathbb{R}^N$.

Newton Verfahren für Systeme

Satz: Für $\bar{B} = B \Subset \mathbb{R}^N$ sei $\boldsymbol{G} \in \mathcal{C}^2(B, \mathbb{R}^N)$ mit $\boldsymbol{G}(B) \subseteq B$. Dann $\exists \boldsymbol{u}^\star \in B$ mit $\boldsymbol{G}(\boldsymbol{u}^\star) = \boldsymbol{u}^\star$. Wenn $\rho(\boldsymbol{G}'(\boldsymbol{u}^\star)) = 0$ gilt, dann $\exists \epsilon > 0$, wobei $\forall \boldsymbol{u}_0 \in B(\boldsymbol{u}^\star, \epsilon)$ die Fixpunktiteration $\{\boldsymbol{u}_{k+1} = \boldsymbol{G}(\boldsymbol{u}_k)\}$ quadratisch zu $\boldsymbol{u}^\star$ konvergiert.

- Mit $\boldsymbol{F}(\boldsymbol{u}^\star) = 0$ und $\epsilon > 0$ seien $\boldsymbol{F}(\boldsymbol{u})$ und $\boldsymbol{F}'(\boldsymbol{u})^{-1}$ für $\boldsymbol{u} \in \bar{B}(\boldsymbol{u}^\star, \epsilon) = B$ ausreichend glatt, dass die Funktion

$$\boldsymbol{G}_{\mathrm{N}}(\boldsymbol{u}) = \boldsymbol{u} - \boldsymbol{F}'(\boldsymbol{u})^{-1}\boldsymbol{F}(\boldsymbol{u})$$

  $\boldsymbol{G}_{\mathrm{N}} \in \mathcal{C}^2(B, \mathbb{R}^N)$ erfüllt. Dann gilt

$$\boldsymbol{G}'_{\mathrm{N}}(\boldsymbol{u}) = I - [\partial_{\boldsymbol{u}}\boldsymbol{F}'(\boldsymbol{u})^{-1}] \underbrace{\boldsymbol{F}(\boldsymbol{u})}_{\to 0, \boldsymbol{u} \to \boldsymbol{u}^\star} - \boldsymbol{F}'(\boldsymbol{u})^{-1}\boldsymbol{F}'(\boldsymbol{u}) \overset{\boldsymbol{u} \to \boldsymbol{u}^\star}{\longrightarrow} 0$$

  und die Bedingungen des Satzes werden erfüllt.
- Für die glatte Abbildung $\boldsymbol{F}(\boldsymbol{u}) = \nabla_{\boldsymbol{u}} J_h(\boldsymbol{u})/h$ ist $F'(\boldsymbol{u})$ SPD $\forall \boldsymbol{u} \in \mathbb{R}^N$, und die Bedingungen des Satzes werden für das Newton Verfahren erfüllt.

Vergleich der Iterativen Verfahren

- Die Dimension der Daten in J_h wird reduziert, um die verschiedenen Ansätze zu vergleichen und zu visualisieren.
- Nimm $N = 2$, $h = 1/2$, $\boldsymbol{u} = (u_1, u_2)$ und $\boldsymbol{v} = (v_1, v_2)$,

$$J_h(\boldsymbol{u}) = \tfrac{1}{4}(u_1 - v_1)^2 + \tfrac{1}{4}(u_2 - v_2)^2 + \mu\sqrt{(u_2 - u_1)^2 + \tfrac{1}{4}\epsilon^2}$$

- Das Abstiegsverfahren ist $\boldsymbol{u}_{k+1} = \boldsymbol{G}_{\mathrm{A}}(\boldsymbol{u}_k)$ oder

$$\boldsymbol{u}_{k+1} = (1 - \alpha)\boldsymbol{u}_k + \alpha\left(\boldsymbol{v} - \frac{\mu}{h^2} D(\boldsymbol{u}_k)\boldsymbol{u}_k\right)$$

- Die Implizite Fixpunktiteration ist $\boldsymbol{u}_{k+1} = \boldsymbol{G}_{\mathrm{I}}(\boldsymbol{u}_k)$ oder

$$\left[I + \frac{\mu}{h^2} D(\boldsymbol{u}_k)\right] \boldsymbol{u}_{k+1} = \boldsymbol{v}$$

- Die Newton Iteration ist $\boldsymbol{u}_{k+1} = \boldsymbol{G}_{\mathrm{N}}(\boldsymbol{u}_k)$ oder

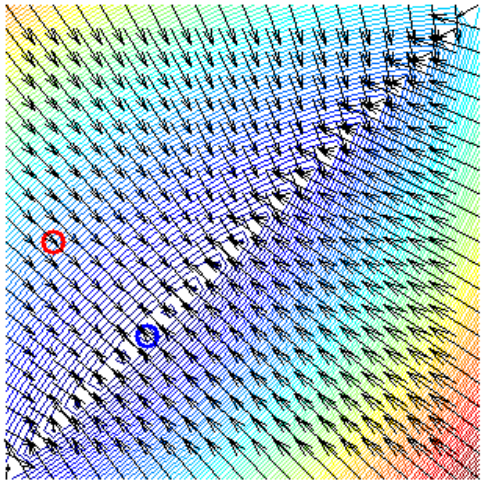
$$\left[I + \frac{\mu\epsilon^2}{h^2} E(\boldsymbol{u}_k)\right](\boldsymbol{u}_{k+1} - \boldsymbol{u}_k) = -\left[\boldsymbol{u}_k - \boldsymbol{v} + \frac{\mu}{h^2} D(\boldsymbol{u}_k)\boldsymbol{u}_k\right]$$

- Die Implizite Fixpunktiteration funktioniert hier am besten.

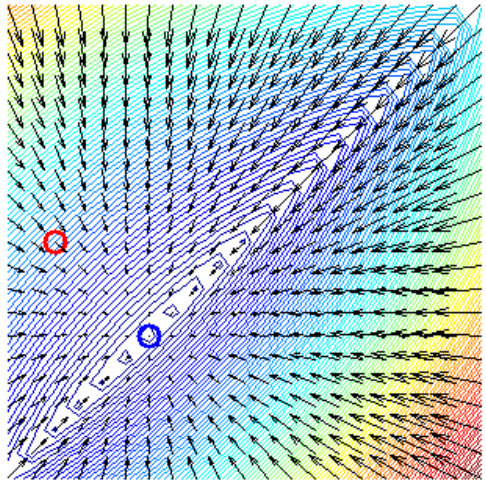
Richtungsfelder der Lösungsverfahren

- Für $N = 2$, $h = \frac{1}{2}$, $v_1 = \frac{1}{10}$, $v_2 = \frac{1}{2}$, $h^2\epsilon^2 = 10^{-4}$ und $\mu = h$ sind die Richtungsfelder $\boldsymbol{u}_{k+1} - \boldsymbol{u}_k$, $\boldsymbol{u}_k \in (0,1)^2$, der jeweiligen Methoden hier grafisch dargestellt:

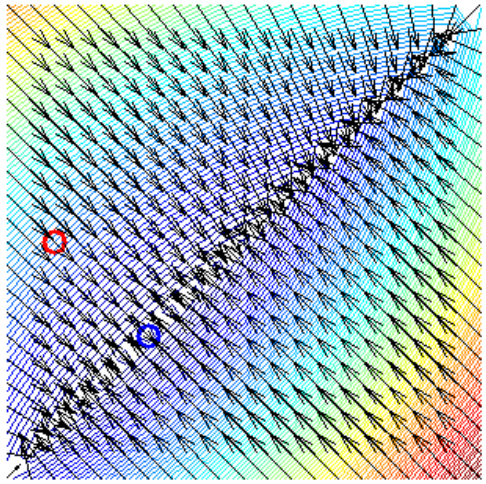
Abstiegsverfahren



Implizites Fixpunkt Verfahren



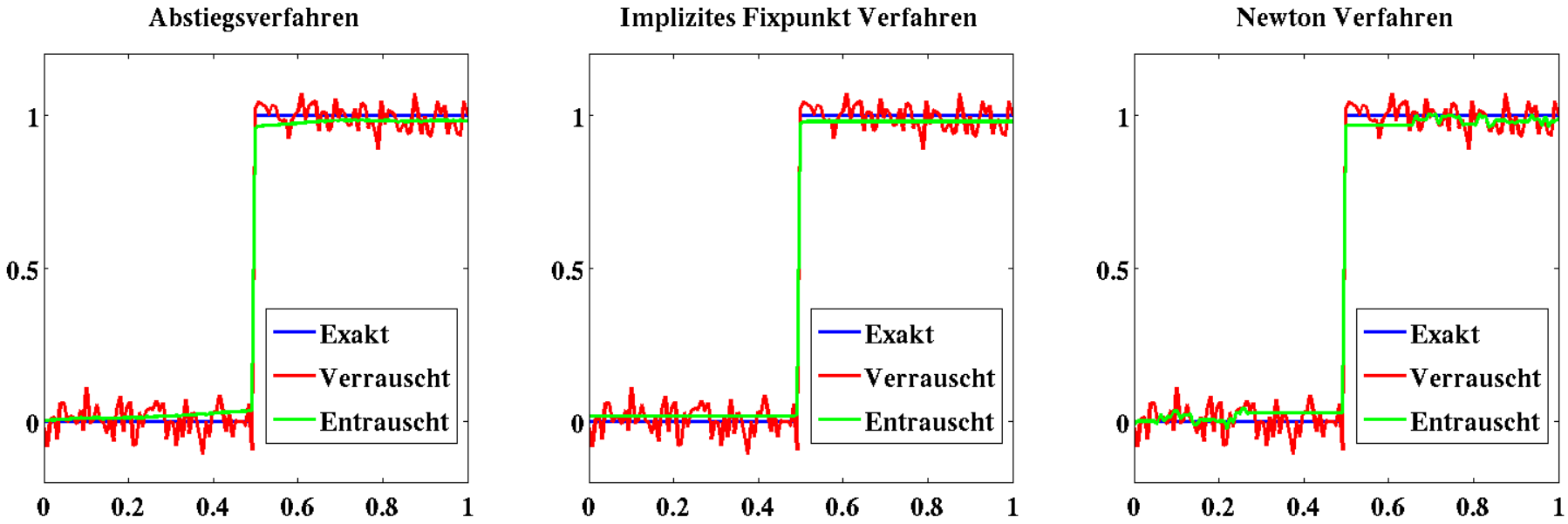
Newton Verfahren



- Der Punkt ○ markiert $\boldsymbol{v}$.
- Für ϵ klein und μ groß, soll das Minimum $\boldsymbol{u}$ sich näher bei $u_1 = u_2$ d.h. dem Punkt ○ befinden.
- Nur das Richtungsfeld für das Implizite Fixpunkt Verfahren zeigt überall zuverlässig in die Richtung des Minimums.

Entrauschen Ergebnisse

- Nun $N = 128$, $\epsilon = 10^{-3}$, $\mu = 7 \cdot 10^{-3}$ und $\boldsymbol{v}$ ist rot:



- Zusammenfassung der (notwendigen) Parameter:

Methode	Dämpfung	Iterationen	Fehler	Zeit
Abstieg:	$\alpha = 10^{-3}$	10^4 (max)	$2 \cdot 10^{-3}$	1.1
Implizit:	0	22	$9 \cdot 10^{-5}$	0.01
Newton:	$\omega = 10^{-2}$	10^4 (max)	$2 \cdot 10^{-3}$	3.7

wobei:

Fehler = $\|\boldsymbol{u}_{k+1} - \boldsymbol{u}_k\|_2 / \|\boldsymbol{u}_{k+1}\|_2$, Zeit = Sekunden

Newton Dämpfung: $\boldsymbol{u}_{k+1} \leftarrow \omega \boldsymbol{u}_{k+1} + (1 - \omega)\boldsymbol{u}_k$

Numerisches Differenzieren und Integrieren

Zuerst werden Approximationen von Ableitungen untersucht.

- Vorwärts Differenzen:

$$f'(x) = \frac{f(x+h) - f(x)}{h} + \mathcal{O}(h)$$

wobei mit dem Taylorsatz $\exists \xi \in [x, x+h]$ mit

$$f(x+h) = f(x) + f'(x)h + \underbrace{\tfrac{1}{2} f''(\xi) h^2}_{h\mathcal{O}(h)}$$

- Rückwärts Differenzen:

$$f'(x) = \frac{f(x) - f(x-h)}{h} + \mathcal{O}(h)$$

wobei mit dem Taylorsatz $\exists \eta \in [x-h, x]$ mit

$$f(x-h) = f(x) - f'(x)h + \underbrace{\tfrac{1}{2} f''(\eta) h^2}_{h\mathcal{O}(h)}$$

Approximation der Ableitungen

- Zentrale Differenzen:

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + \mathcal{O}(h^2)$$

wobei mit dem Taylorsatz $\exists \eta \in [x-h, x]$, $\xi \in [x, x+h]$ mit

$$\begin{aligned} f(x+h) &= f(x) + f^{(1)}(x)h + \tfrac{1}{2}f^{(2)}(x)h^2 + \tfrac{1}{6}f^{(3)}(\xi)h^3 \\ f(x-h) &= f(x) - f^{(1)}(x)h + \tfrac{1}{2}f^{(2)}(x)h^2 - \tfrac{1}{6}f^{(3)}(\eta)h^3 \end{aligned}$$

und $$\frac{f(x+h) - f(x-h)}{2h} - f'(x) = \underbrace{\frac{1}{12}f^{(3)}(\xi)h^2 + \frac{1}{12}f^{(3)}(\eta)h^2}_{\mathcal{O}(h^2)}$$

- Ähnlich für die zweite Ableitung, wie auf 18 gezeigt,

$$\begin{aligned} f''(x) &= \frac{f'(x+\frac{h}{2}) - f'(x-\frac{h}{2})}{h} - \frac{h^2}{24}f^{(4)}(x) + \mathcal{O}(h^4) \\ &= \frac{1}{h}\left[\frac{f(x+h) - f(x)}{h} - \frac{f(x) - f(x-h)}{h} - \frac{h^3}{24}f^{(4)}(x)\right] + \mathcal{O}(h^2) \\ &= \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} + \mathcal{O}(h^2) \end{aligned}$$

Approximation der Ableitungen

- Für die vierte Ableitung,

$$
\begin{aligned}
f^{(4)}(x) &= \frac{f^{(2)}(x+h) - 2f^{(2)}(x) + f^{(2)}(x-h)}{h^2} - \frac{h^2}{12} f^{(6)}(x) + \mathcal{O}(h^2) \\
&= \left[\frac{f(x+2h) - 2f(x+h) + f(x)}{h^4} - \frac{f^{(4)}(x)}{12} \right] + \\
&\quad \left[\frac{f(x+h) - 2f(x) + f(x-h)}{h^4} - \frac{f^{(4)}(x)}{12} \right] (-2) + \\
&\quad \left[\frac{f(x) - 2f(x-h) + f(x-2h)}{h^4} - \frac{f^{(4)}(x)}{12} \right] + \mathcal{O}(h^2) \\
&= \frac{f(x-2h) - 4f(x-h) + 6f(x) - 4f(x+h) + f(x+2h)}{h^4} + \mathcal{O}(h^2)
\end{aligned}
$$

- Für Ableitungen höherer Ordnung,

$$
\frac{\delta_h^n f(x)}{h^n} = \frac{1}{h^n} \sum_{k=0}^{n} (-1)^k \binom{n}{k} f(x + (n/2 - k)h) = f^{(n)}(x) + \mathcal{O}(h^2)
$$

  Bei n ungerade $\delta_h^n f(x \pm \frac{h}{2})$ mitteln, um f auf $h\mathbb{Z}$ auszuwerten.

- Fehler-Norm $\|F(\boldsymbol{x})\|_{p,h} = h^{1/p} \|F(\boldsymbol{x})\|_p$, z.B. für $f(x) = \sqrt{x}$.

Approximation der Ableitungen

- Allgemeiner: Die Koeffizienten $\{a_{-1}, a_0, a_1\}$ einer Approximation der ersten Ableitung werden bestimmt,

$$f'(x) = a_{-1} f(x-h) + a_0 f(x) + a_1 f(x+h) + \mathcal{O}(h^p)$$

  um die Genauigkeit $\mathcal{O}(h^p)$ zu maximieren.
- Mit dem Taylorsatz, (ξ, η zwischen x und $x-h$ bzw. $x+h$)

$$\begin{aligned} f(x-h) &= f(x) - f^{(1)}(x)h + \tfrac{1}{2} f^{(2)}(x)h^2 - \tfrac{1}{6} f^{(3)}(\xi)h^3 \\ f(x+h) &= f(x) + f^{(1)}(x)h + \tfrac{1}{2} f^{(2)}(x)h^2 + \tfrac{1}{6} f^{(3)}(\eta)h^3 \end{aligned}$$

  ergibt sich die gewichtete Summe,

$$\begin{aligned} a_{-1} f(x-h) + a_0 f(x) + a_1 f(x+h) = (a_{-1} + a_0 + a_1) f(x) \\ + (-a_{-1} + a_1) f^{(1)}(x)h + (a_{-1} + a_1) \tfrac{1}{2} f^{(2)}(x)h^2 \\ - a_{-1} \tfrac{1}{6} f^{(3)}(\xi)h^3 + a_1 \tfrac{1}{6} f^{(3)}(\eta)h^3 \end{aligned}$$

  mit der ein System für die Koeffizienten hergeleitet wird:

$$\begin{bmatrix} 1 & 1 & 1 \\ -1 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_{-1} \\ a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} 0 \\ h^{-1} \\ 0 \end{bmatrix} \Rightarrow \begin{bmatrix} a_{-1} \\ a_0 \\ a_1 \end{bmatrix} = \frac{1}{2h} \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

Approximationen der Ableitungen

- Aus

$$-\frac{a_{-1}}{6}f^{(3)}(\xi)h^3+\frac{a_1}{6}f^{(3)}(\eta)h^3=\frac{1}{12}[f^{(3)}(\xi)+f^{(3)}(\eta)]h^2$$

  folgt die Genauigkeit $\mathcal{O}(h^2)$ oder $p=2$.
- Ähnlich können die Koeffizienten $\{a_0,a_1,a_2\}$ der Vorwärts-Approximation bestimmt werden,

$$f'(x)=a_0f(x)+a_1f(x+h)+a_2f(x+2h)+\mathcal{O}(h^q)$$

  für ein möglichst hohes q.
- Mit dem Taylorsatz, (ξ,η zwischen x und $x-h$ bzw. $x+h$)

$$\begin{aligned}f(x+h) &= f(x)+f^{(1)}(x)h+\tfrac{1}{2}f^{(2)}(x)h^2+\tfrac{1}{6}f^{(3)}(\xi)h^3\\ f(x+2h) &= f(x)+f^{(1)}(x)2h+\tfrac{1}{2}f^{(2)}(x)4h^2+\tfrac{1}{6}f^{(3)}(\eta)8h^3\end{aligned}$$

  ergibt sich die gewichtete Summe,

$$\begin{aligned}a_0f(x)+a_1f(x+h)+a_2f(x+2h)&=(a_0+a_1+a_2)f(x)\\ &+(a_1+2a_2)f^{(1)}(x)h+(a_1+4a_2)\tfrac{1}{2}f^{(2)}(x)h^2\\ &+a_1\tfrac{1}{6}f^{(3)}(\xi)h^3+a_2\tfrac{1}{6}f^{(3)}(\eta)8h^3\end{aligned}$$

Approximationen der Ableitungen

- mit der ein System für die Koeffizienten hergeleitet wird:

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \\ 0 & 1 & 4 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 0 \\ h^{-1} \\ 0 \end{bmatrix} \Rightarrow \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \frac{1}{2h} \begin{bmatrix} -3 \\ 4 \\ -1 \end{bmatrix}$$

- Die Genauigkeit $\mathcal{O}(h^2)$ folgt aus

$$\tfrac{1}{6} a_1 f^{(3)}(\xi) h^3 + \tfrac{1}{6} a_1 f^{(3)}(\eta) 8 h^3 = \tfrac{1}{3}[f^{(3)}(\xi) - 2 f^{(3)}(\eta)] h^2.$$

- Hausaufgabe: Bestimme die Koeffizienten, mit denen gilt

$$f''(x) = a_{-2} f(x - 2h) + \cdots + a_{+2} f(x + 2h) + \cdots + \mathcal{O}(h^4)$$

- Mit Vorwärts- und Rückwärts-Differenzen,

$$\frac{\Delta_h^n f(x)}{h^n} = \frac{1}{h^n} \sum_{k=0}^{n} (-1)^{n-k} \binom{n}{k} f(x + kh) = f^{(n)}(x) + \mathcal{O}(h)$$

$$\frac{\nabla_h^n f(x)}{h^n} = \frac{1}{h^n} \sum_{k=0}^{n} (-1)^{k} \binom{n}{k} f(x - kh) = f^{(n)}(x) + \mathcal{O}(h)$$

Richardson Extrapolation

- Sei eine Formel $N_0(h)$ zur Approximation einer Größe M mit einer gewissen Genauigkeit verfügbar. Die Genauigkeit soll *billig* erhöht werden.
- Wenn gilt

$$M = N_0(h) + a_0 h^2 + \mathcal{O}(h^4)$$

dann folgt

$$M = N_0(h/2) + a_0 (h/2)^2 + \mathcal{O}(h^4).$$

Durch eine gewichtete Summe dieser Formeln ergibt sich

$$M = \underbrace{\frac{4N_0(h/2) - N_0(h)}{4-1}}_{=:N_1(h)} + \mathcal{O}(h^4)$$

und $N_1(h)$ ist eine Approximation höherer Genauigkeit.

- Wenn allgemeiner gilt

$$M = N_0(h) + a_0 h^{k_0} + a_1 h^{k_1} + \cdots$$

Richardson Extrapolation

dann für fixiertes $t > 1$, z.B. $t = 2$ wie oben,

$$N_{i+1}(h) = \frac{t^{k_i} N_i(h/t) - N_i(h)}{t^{k_i} - 1}, \quad i = 0, 1, 2, \ldots$$

und es folgt

$$M = N_{i+1}(h) + \mathcal{O}(h^{k_{i+1}}), \quad i = 0, 1, 2, \ldots$$

- Beispiel: Mit

$$N_0(h) = \frac{f(x_0 + 2h) - f(x_0)}{2h} = f'(x_0) + hf^{(2)}(x_0) + \frac{2h^2}{3} f^{(3)}(x_0) + \cdots$$

$t = 2$, $k_0 = 1$ und $k_1 = 2$ gilt $N_1(h) =$

$$\frac{2N_0(h/2) - N_0(h)}{2 - 1} = \frac{-3f(x_0) + 4f(x_0 + h) - f(x_0 + 2h)}{2h} = f'(x_0) + \mathcal{O}(h^2)$$

- Beispiel: Für

$$N_0(h) = \frac{f(x_0 + h) - f(x_0 - h)}{2h} = f'(x_0) + \frac{h^2}{2!} f^{(3)}(x_0) + \frac{h^4}{4!} f^{(5)}(x_0) \cdots$$

wird mit $f(x) = xe^x$, $x_0 = 2$ und $h = 0.2$ explizit gerechnet:

Richardson Extrapolation

Der bekannte Wert ist $f'(x_0) = 22.167168\cdots$

- Mit Extrapolation,

$$\begin{aligned} N_0(0.2) &= [f(2.2) - f(1.8)]/[2(0.2)] &&= 22.414160 \\ N_0(0.1) &= [f(2.1) - f(1.9)]/[2(0.1)] &&= 22.228786 \\ N_0(0.05) &= [f(2.05) - f(1.95)]/[2(0.05)] &&= 22.182564 \end{aligned}$$

konvergiert mit $\mathcal{O}(h^2)$.

- Erweiterte Tabelle, $N_1(h)$ konvergiert mit $\mathcal{O}(h^4)$,

$$\begin{aligned} N_0(0.2) &= 22.414160 \\ N_0(0.1) &= 22.228786 & N_1(0.2) &= [4N_0(0.1) - N_0(0.2)]/3 = 22.166995 \\ N_0(0.05) &= 22.182564 & N_1(0.1) &= [4N_0(0.05) - N_0(0.1)]/3 = 22.167157 \end{aligned}$$

- Noch weiter, $N_2(h)$ konvergiert mit $\mathcal{O}(h^6)$,

$$\begin{aligned} N_0(0.2) &= 22.414160 \\ N_0(0.1) &= 22.228786 & N_1(0.2) &= 22.166995 \\ N_0(0.05) &= 22.182564 & N_1(0.1) &= 22.167157 & N_2(0.2) &= [16N_1(0.1) - N_1(0.2)]/15 \\ & & & & &= 22.167168 \end{aligned}$$

- Die Methode wird auch für numerische Integration verwendet.

Numerische Integration

- Die meisten Integrale können nicht analytisch ausgerechnet werden, z.B.

$$\operatorname{erf}(z) = \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2} dt$$

- Integrale werden mit gewichteten Summen von Funktionswerten approximiert:

$$\int_a^b f(x)dx \approx \sum_{i=0}^{n} a_i f(x_i)$$

wobei die Gewichte $\{a_i\}$ und die *Quadraturpunkte* $\{x_i\}$ auszuwählen sind.

- Von Interpolation sind die Gewichte gegeben durch

$$f(x) \approx \sum_{i=0}^{n} f(x_i) L_{n,i}(x) \quad \Rightarrow \quad \int_a^b f(x)dx \approx \sum_{i=0}^{n} f(x_i) \underbrace{\int_a^b L_{n,i}(x)dx}_{=:a_i}$$

- Für eine *Newton-Cotes* Formel sind die Quadraturpunkte $\{x_i\}$ gleichmäßig verteilt.

Trapez-Regel

- Der Approximationsfehler

$$E(f) = \int_a^b f(x)dx - \sum_{i=0}^{n} f(x_i) \int_a^b L_{n,i}(x)dx$$

kann mit dem Satz 141 so dargestellt werden,

$$E(f) = \frac{1}{(n+1)!} \int_a^b f^{(n+1)}(\xi(x)) \prod_{i=0}^{n} (x - x_i)dx$$

und entsprechend abgeschätzt werden.

- Beispiel: *Trapez-Regel*. Gegeben sind $\{x_0, x_1\}$, $n = 1$, $h = x_1 - x_0$ und

$$f(x) \approx f(x_0)\frac{x - x_1}{x_0 - x_1} + f(x_1)\frac{x - x_0}{x_1 - x_0}$$

Dann mit $f_i = f(x_i)$ gilt

$$\int_{x_0}^{x_1} f(x)dx = f_0 \int_{x_0}^{x_1} \frac{x - x_1}{x_0 - x_1}dx + f_1 \int_{x_0}^{x_1} \frac{x - x_0}{x_1 - x_0}dx + E(f)$$

Trapez-Regel

oder

$$\begin{aligned}\int_{x_0}^{x_1} f(x)dx &= \frac{f_0}{x_0 - x_1}\frac{(x-x_1)^2}{2}\Big|_{x_0}^{x_1} + \frac{f_1}{x_1 - x_0}\frac{(x-x_0)^2}{2}\Big|_{x_0}^{x_1} + E(f) \\ &= \frac{f_0 + f_1}{2}(x_1 - x_0) + E(f)\end{aligned}$$

- Der Approximationsfehler ist:

$$E(f) = \frac{1}{2}\int_{x_0}^{x_1} f^{(2)}(\xi(x))(x - x_0)(x - x_1)dx$$

Nach dem Mittelwertsatz 11 für Integrale $\exists \hat{x} \in [x_0, x_1]$ wobei

$$E(f) = \frac{f^{(2)}(\xi(\hat{x}))}{2}\int_{x_0}^{x_1}(x - x_0)(x - x_1)dx = -\frac{h^3}{6}f^{(2)}(\xi(\hat{x}))$$

- Für die Trapez-Regel gilt daher

$$\int_{x_0}^{x_1} f(x)dx = \frac{f_0 + f_1}{2}(x_1 - x_0) + \mathcal{O}(h^3)$$

Simpson's Regel

- Beispiel: *Simpson's Regel*. Gegeben sind $\{x_0, x_1, x_2\}$, $n = 2$, $x_1 = (x_0 + x_2)/2$, $h = x_1 - x_0$ und

$$f(x) \approx f(x_0)\frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} + f(x_1)\frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} + f(x_2)\frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}$$

Dann mit $f_i = f(x_i)$ gilt

$$\int_{x_0}^{x_2} f(x)dx = \frac{x_2 - x_0}{6}[f_0 + 4f_1 + f_2] + \mathcal{O}(h^5)$$

mit $E(f) = \frac{f^{(3)}(x_0)}{3!} \underbrace{\int_{x_0}^{x_2} (x-x_0)(x-x_1)(x-x_2)dx}_{=0} + \frac{f^{(4)}(\xi(\hat{x}))}{4!}\int_{x_0}^{x_2}(x-x_0)^2(x-x_1)(x-x_2)dx$

- Bemerkung: Es gibt einen Sprung in Genauigkeit von $\mathcal{O}(h^3)$ für Trapez ($n = 1$, ungerade!) zu $\mathcal{O}(h^5)$ für Simpson ($n = 2$, gerade!): Der Fehlerintegrand wird ungerade!

Def: Der *Genauigkeitsgrad* einer Integrationsformel ist n, wenn $E(P) = 0, \forall P \in \mathcal{P}^n$ aber $E(\tilde{P}) \neq 0$ für ein $\tilde{P} \in \mathcal{P}^{n+1}$.

Genauigkeit einer Newton-Cotes Formel

Def: Eine Newton-Cotes Formel heisst *geschlossen* wenn $a = x_0$ und $b = x_n$ gelten. Sonst heisst sie *offen*.

Satz: Sei eine *geschlossene* Newton-Cotes Formel gegeben,

$$\int_a^b f(x)dx = \sum_{i=0}^{n} a_i f(x_i) + E(f), \quad a_i = \int_a^b L_{n,i}(x)dx$$

mit $x_i = a + ih$, $i = 0, \ldots, n$, $h = (b-a)/n$. Es gelten

- Wenn n ungerade ist und $f \in C^{n+1}([a,b])$, ist der Genauigkeitsgrad n und $\exists \xi \in [a,b]$ mit

$$E(f) = h^{n+2} \frac{f^{(n+1)}(\xi)}{(n+1)!} \int_0^n t(t-1)\cdots(t-n)dt$$

- Wenn n gerade ist und $f \in C^{n+2}([a,b])$, ist der Genauigkeitsgrad $n+1$ und $\exists \eta \in [a,b]$ mit

$$E(f) = h^{n+3} \frac{f^{(n+2)}(\eta)}{(n+2)!} \int_0^n t^2(t-1)\cdots(t-n)dt$$

Geschlossene Newton-Cotes Formeln

- Beispiel: Es gibt einen Sprung im Genauigkeitsgrad von $n = 1$ mit $E = \mathcal{O}(h^{n+2})$ für Trapez zu $n + 1 = 3$ mit $E = \mathcal{O}(h^{n+3})$ für Simpson.
- Botschaft: Es ist vorteilhaft wenn n gerade ist.
- Für die nächsten zwei Integrationsformeln seien die Quadraturpunkte gegeben durch

$$x_i = a + ih, \quad f_i = f(x_i), \quad i = 0, \ldots, n, \quad h = (b - a)/n.$$

Für $n = 3$, exakt für $f \in \mathcal{P}^3$

$$\int_{x_0}^{x_3} f(x)dx = \frac{3h}{8}[f_0 + 3f_1 + 3f_2 + f_3] + \mathcal{O}(h^5)$$

Für $n = 4$, exakt für $f \in \mathcal{P}^5$

$$\int_{x_0}^{x_4} f(x)dx = \frac{2h}{45}[7f_0 + 32f_1 + 12f_2 + 32f_3 + 7f_4] + \mathcal{O}(h^7)$$

- Hier gibt es einen Sprung im Genauigkeitgrad von $n = 3$ mit $E = \mathcal{O}(h^{n+2})$ zu $n + 1 = 5$ mit $E = \mathcal{O}(h^{n+3})$.

Genauigkeit einer offenen Newton-Cotes Formel

Satz: Sei eine *offene* Newton-Cotes Formel gegeben,

$$\int_a^b f(x)dx = \sum_{i=0}^{n} a_i f(x_i) + E(f), \quad a_i = \int_a^b L_{n,i}(x)dx$$

mit $x_i = a+(i+1)h$, $i = 0,\ldots,n$, $h = (b-a)/(n+2)$. Es gelten

- Wenn n ungerade ist und $f \in C^{n+1}([a,b])$, ist der Genauigkeitsgrad n und $\exists \xi \in [a,b]$ mit

$$E(f) = h^{n+2}\frac{f^{(n+1)}(\xi)}{(n+1)!}\int_{-1}^{n+1} t(t-1)\cdots(t-n)dt$$

- Wenn n gerade ist und $f \in C^{n+2}([a,b])$, ist der Genauigkeitsgrad $n+1$ und $\exists \eta \in [a,b]$ mit

$$E(f) = h^{n+3}\frac{f^{(n+2)}(\eta)}{(n+2)!}\int_{-1}^{n+1} t^2(t-1)\cdots(t-n)dt$$

Mittelpunkt-Regel

- Beispiel: Mittelpunkt-Regel, $n = 0$, $h = x_1 - x_0$,

$$\int_{x_{-1}}^{x_1} f(x)dx = 2hf(x_0) + \mathcal{O}(h^3)$$

- Exakt für $f \in \mathcal{P}^1$. Direkt zeigen: (ξ zwischen x und x_0)

$$f(x) = f(x_0) + (x - x_0)f'(x_0) + \frac{1}{2}(x - x_0)^2 f''(\xi)$$

(η zwischen x_{-1} und x_1)

$$\int_{x_{-1}}^{x_1} f(x)dx = \underbrace{\int_{x_{-1}}^{x_1} f(x_0)dx}_{=(x_1-x_{-1})f(x_0)} + f'(x_0) \underbrace{\int_{x_{-1}}^{x_1} (x - x_0)dx}_{=0}$$

$$+ \frac{f''(\eta)}{2} \underbrace{\int_{x_{-1}}^{x_1} (x - x_0)^2 dx}_{=2h^3/3}$$

- Der Fehlerintegrand wird ungerade!

Offene Newton-Cotes Formeln

- Weitere offene Newton-Cotes Formeln: $h = x_1 - x_0$

$n = 1$ exakt für $f \in \mathcal{P}^1$

$$\int_{x_{-1}}^{x_2} f(x)dx = \frac{3h}{2}[f_0 + f_1] + \mathcal{O}(h^3)$$

$n = 2$ Milne's Regel exakt für $f \in \mathcal{P}^3$

$$\int_{x_{-1}}^{x_3} f(x)dx = \frac{4h}{3}[2f_0 - f_1 + 2f_2] + \mathcal{O}(h^5)$$

$n = 3$ exakt für $f \in \mathcal{P}^3$

$$\int_{x_{-1}}^{x_4} f(x)dx = \frac{5h}{24}[11f_0 + f_1 + f_2 + 11f_3] + \mathcal{O}(h^5)$$

- Hier gibt es einen Sprung im Genauigkeitsgrad von $n = 1$ mit $E = \mathcal{O}(h^{n+2})$ zu $n + 1 = 3$ mit $E = \mathcal{O}(h^{n+3})$.
- Auch keinen Sprung von $n = 2$ zu $n = 3$.

Zusammengesetzte Mittelpunkt-Regel

- Hier wird ein Integral so zerlegt,

$$\int_a^b f(t)dt = \sum_{j=1}^{m} \int_{t_j}^{t_{j+1}} f(t)dt$$

wobei mit $h = (b-a)/m$,

$$t_j = a + (j-1)h, \quad j = 1, \ldots, m+1.$$

- Eine Grundregel wird in jedem Teilintervall angewendet:

$$\int_a^b f(t)dt = \sum_{j=1}^{m} \int_0^h f(t_j+x)dx = \sum_{j=1}^{m} \left[\sum_{i=0}^{n} a_i f(t_j + x_i) + \mathcal{O}(h^p) \right]$$

- Für die Mittelpunkt-Regel mit $\bar{t}_j = (t_j + t_{j+1})/2$, $(\bar{t}_j - t_j = h/2)$

$$\int_a^b f(t)dt = \sum_{j=1}^{m} \left[hf(\bar{t}_j) + \mathcal{O}(h^3) \right]$$

Zusammengesetzte Simpsons-Regeln

oder mit $mh^3 = (mh)h^2 = (b-a)h^2 = \mathcal{O}(h^2)$,

$$\int_a^b f(t)dt = h\sum_{j=1}^{m} f(\bar{t}_j) + \mathcal{O}(h^2)$$

Satz: $E(f) = \mathcal{O}(h^2)$ gilt für die Zusammengesetzte Mittelpunkt-Regel wenn $f \in \mathcal{C}^2([a,b])$.

- Nun mit $\bar{t}_j = (t_j + t_{j+1})/2$, $\bar{f}_j = f(\bar{t}_j)$ und $f_j = f(t_j)$,

$$\int_a^b f(t)dt = \sum_{j=1}^{m} \int_{t_j}^{t_{j+1}} f(t)dt = \sum_{j=1}^{m} \left[\frac{h}{6}(f_j + 4\bar{f}_j + f_{j+1}) + \mathcal{O}(h^5) \right]$$

$$= \frac{h}{6}\sum_{j=1}^{m} f_j + \frac{4h}{6}\sum_{j=1}^{m} \bar{f}_j + \frac{h}{6}\sum_{j=1}^{m} f_{j+1} + \mathcal{O}(mh^5) =$$

$$\frac{h}{6}\left[f(a) + \sum_{j=2}^{m} f_j \right] + \frac{4h}{6}\sum_{j=1}^{m} \bar{f}_j + \frac{h}{6}\left[\sum_{j=2}^{m} f_j + f(b) \right] + \mathcal{O}(h^4)$$

wobei $mh^5 = (mh)h^4 = \mathcal{O}(h^4)$.

Zusammengesetzte Trapez-Regeln

- Für die Simpsons-Regel ergibt sich: $(\bar{t}_j - t_j = h/2)$

$$\int_a^b f(t)dt = \frac{h}{6}\left[f(a) + 4\sum_{j=1}^{m} f(\bar{t}_j) + 2\sum_{j=2}^{m} f(t_j) + f(b) \right] + \mathcal{O}(h^4)$$

Satz: $E(f) = \mathcal{O}(h^4)$ gilt für die Zusammengesetzte Simpsons-Regel wenn $f \in \mathcal{C}^4([a,b])$.

- Hausaufgabe: Leite die Zusammengesetzte Trapez-Regel her:

$$\int_a^b f(t)dt = \frac{h}{2}\left[f(a) + 2\sum_{j=2}^{m} f(t_j) + f(b) \right] + \mathcal{O}(h^2)$$

Satz: $E(f) = \mathcal{O}(h^2)$ gilt für die Zusammengesetzte Trapez-Regel wenn $f \in \mathcal{C}^2([a,b])$.

Romberg Integration

- Mit der zusammengesetzten Trapez-Regel,

$$\int_a^b f(t)dt = N_0(h) + c(f^{(2)})h^2 + c(f^{(4)})h^4 + \cdots$$

$$N_0(h) = \frac{h}{2}\left[f(a) + 2\sum_{j=2}^{m} f(t_j) + f(b) \right]$$

$$t_j = a + (j-1)h, \quad j = 1, \ldots, m+1, \quad h = (b-a)/m$$

wird Richardson Extrapolation durchgeführt,

$$N_j(h) = \frac{4^j N_{j-1}(h/2) - N_{j-1}(h)}{4^j - 1}, \quad j = 1, 2, \ldots$$

- Die erste Spalte der Richardson Tabelle,

$$\begin{array}{ll} N_0(h) & \\ N_0(h/2) & N_1(h) = [4N_0(h/2) - N_0(h)]/3 \\ N_0(h/4) & N_1(h/2) = [4N_0(h/4) - N_0(/2)]/3 \quad N_2(h) = [16N_1(h/2) - N_1(h)]/15 \ldots \end{array}$$

kann effizient berechnet werden, da die Auswertungen in $N_0(h/2^{k-1})$ wieder für $N_0(h/2^k)$ verwendet werden können.

Romberg Integration

- Um diese Behiehung angenehm darzustellen, bemerke

$$t_j^{(m)} = a + (j-1)\frac{b-a}{m}, \quad \bar{t}_j^{(m)} = \frac{t_j + t_{j+1}}{2}, \quad \{t_j^{(2m)}\} = \{t_l^{(m)}, \bar{t}_l^{(m)}\}.$$

- Dann mit $m \in 2^{\mathbb{N}_0}$ und $h = (b-a)/m$ gelten

$$\begin{aligned} N_0(h/2) &= \frac{h/2}{2}\left[f(a) + 2\sum_{j=2}^{2m} f(t_j^{(2m)}) + f(b) \right] \\ &= \frac{h/2}{2}\left[f(a) + 2\sum_{l=2}^{m} f(t_l^{(m)}) + 2\sum_{l=1}^{m} f(\bar{t}_l^{(m)}) + f(b) \right] \\ &= \frac{1}{2}\left[N_0(h) + h\sum_{l=1}^{m} f(\bar{t}_l^{(m)}) \right] \end{aligned}$$

$$N_0(h/4) = \frac{1}{2}\left[N_0(h/2) + h/2\sum_{l=1}^{2m} f(\bar{t}_l^{(2m)}) \right]$$

usw.

$$N_0(h/8) = \frac{1}{2}\left[N_0(h/4) + h/4\sum_{l=1}^{4m} f(\bar{t}_l^{(4m)}) \right]$$

Romberg Integration

- Beispiel: $\int_0^\pi \sin(t)dt = 2$, $f(t) = \sin(t)$.
  Nimm $a = 0$, $b = \pi$, $m = 1$, $h = (b-a)/m = \pi$.

$$
\begin{aligned}
N_0(h) &= \tfrac{h}{2}[f(a) + f(b)] = \tfrac{\pi}{2}[\sin(0) + \sin(\pi)] = 0 \\
N_0(h/2) &= \tfrac{1}{2}[N_0(h) + h\sum_{l=1}^{1} f(\underbrace{\bar{t}_l^{(1)}}_{\pi/2})] \approx 1.571 \\
N_0(h/4) &= \tfrac{1}{2}[N_0(h/2) + h/2\sum_{l=1}^{2} f(\underbrace{\bar{t}_l^{(2)}}_{\pi/4,\, 3\pi/4})] \approx 1.896 \\
N_0(h/8) &= \tfrac{1}{2}[N_0(h/4) + h/4\sum_{l=1}^{4} f(\underbrace{\bar{t}_l^{(4)}}_{\pi/8,\, 3\pi/8,\, 5\pi/8,\, 7\pi/8})] \approx 1.974
\end{aligned}
$$

- Die Tabelle:

0.000000			
1.570796	2.094395		
1.896119	2.004560	1.998571	
1.974232	2.000269	1.999983	2.000006
...			

Gauß Quadratur

- Nun wird untersucht, ob die Quadraturpunkte $\{x_i\}$ vorteilhaft ausgewählt werden können.
- Das Resultat: Der Genauigkeitsgrad einer Quadraturformel

$$\int_a^b f(x)dx \approx \sum_{i=0}^{n} a_i f(x_i), \quad a_i = \int_a^b L_{n,i}(x)dx$$

ist maximal bei $2n+1$, wenn die Quadraturpunkte $\{x_i\}_{i=0}^n$ Nullstellen von ϕ_{n+1} aus $\{\phi_k \in \mathcal{P}^k\}_{k\in\mathbb{N}_0}$ sind, wobei diese Polynome folgendermaßen `orthogonal` sind:

- Eine Menge $\{\phi_k\}_{k\in\mathbb{N}_0}$ von Polynomen heissen orthogonal auf $[a,b]$ bezüglich des Gewichts w, wenn die folgenden Eigenschaften erfüllt sind:
  - $\phi_k \in \mathcal{P}^k$
  - $w : [a,b] \to [0,\infty]$ ist auf $[a,b]$ summierbar und

$$\int_a^b w(x)\phi_k(x)\phi_l(x)dx = \alpha_k \delta_{kl}, \quad 0 \le k,l \le n.$$

Sie sind orthonormal wenn $\alpha_k = 1$.

Gauß Quadratur

Satz: Sei $\{\phi_k\}_{k=0}^n$ orthogonale Polynome auf $[a, b]$ bezüglich des Gewichts w. Dann hat ϕ_k genau k Nullstellen in (a, b).

- *Legendre* Polynome: $[a, b] = [-1, 1]$, $w(x) = 1$,

$$P_0(x) = 1, \quad P_1(x) = x, \quad P_2(x) = \tfrac{1}{2}(3x^2 - 1),$$
$$P_3(x) = \tfrac{1}{2}(5x^3 - 3x), \quad P_4(x) = \tfrac{1}{8}(35x^4 - 30x^2 + 3),$$
$$P_5(x) = \tfrac{1}{8}(63x^5 - 70x^3 + 15x),$$

und im Allgemeinen,

$$P_k(x) = \frac{1}{2^k k!} \frac{d^k}{dx^k}[(x^2 - 1)^k]$$

- *Tchebyshev* Polynome: $[a, b] = [-1, 1]$, $w(x) = 1/\sqrt{1 - x^2}$,

$$T_0(x) = 1, \quad T_1(x) = x, \quad T_2(x) = 2x^2 - 1,$$
$$T_3(x) = 4x^3 - 3x, \quad T_4(x) = 8x^4 - 8x^2 + 1,$$
$$T_5(x) = 16x^5 - 20x^3 + 5x,$$

und im Allgemeinen,

$$T_k(x) = \cos[k \cos^{-1}(x)]$$

Gauß Quadratur

Def: Die Nullstellen $\{x_i\}_{i=0}^n$ des Legendre Polynoms P_{n+1} sind die *Gauß-Legendre Punkte*.

Satz: Wenn $\{x_i\}_{i=0}^n$ die Gauß-Legendre Punkte sind, ist der Genauigkeitsgrad der Quadraturformel

$$\int_{-1}^{1} f(x)dx \approx \sum_{i=0}^{n} a_i f(x_i), \quad a_i = \int_{-1}^{+1} L_{n,i}(x)dx$$

maximal bei $2n+1$.

Bemerkung: Mit $2n+2$ Freiheitsgraden $\{a_i\}_{i=0}^n$ und $\{x_i\}_{i=0}^n$ kann für maximal $2n+2$ polynomiale Koeffizienten in

$$\int_{-1}^{1} P(x)dx = \sum_{i=0}^{n} a_i P(x_i), \quad \forall P \in \mathcal{P}^{2n+1}$$

allgemein gelten. Wenn $R \in \mathcal{P}^n$ folgt $E(R) = 0$ aus $D_x^{n+1} R = 0$. Wenn $P \in \mathcal{P}^{2n+1}$ gilt $P = QP^{n+1} + R$ für $Q, R \in \mathcal{P}^n$ und das Legendre Polynom P^{n+1}. Aus Orthogonalität $\int_{-1}^{+1} QP^{n+1} = 0$ und $E(R) = 0$ folgt $E(P) = 0$.

Gauß Quadratur

Bemerkung: Mit der Koordinatentransformation

$$h = b - a, \quad \tau(x) = a + (x+1)h/2, \quad g(x) = f(\tau(x))h/2$$

lässt sich diese Quadraturformel für ein allgemeines Integral anzuwenden, $c_n = (n!)^4/((2n+1)((2n)!)^3)$, $\xi \in (a,b)$

$$\int_a^b f(t)dt = \int_{-1}^{1} g(x)dx = \sum_{i=0}^{n} a_i g(x_i) + c_{n+1} f^{(2n+2)}(\xi) h^{2n+3}$$

- Beispiel: $\int_1^{1.5} e^{-t^2} dt = \frac{\sqrt{\pi}}{2}[\mathrm{erf}(1.5) - \mathrm{erf}(1)] \approx 0.1093643$

Romberg-Trapez:

0.1183197			
0.1115627	0.1093104		
0.1099114	0.1093610	$\lvert F\rvert$ =6.9e-8	
0.1095009	0.1093641	$\lvert F\rvert$ =1.2e-9	$\lvert F\rvert$ =1.0e-10

Zusammengesetzte Gauß Quadratur

Gauß-Legendre: $\int_1^{1.5} e^{-t^2} dt = \int_{-1}^{1} g(x)dx$, $g(x) = \frac{1}{4}\exp[-\frac{(x+5)^2}{16}]$

$$n = 1 \quad \begin{array}{ll} x_0 = -\sqrt{1/3} & a_0 = 1 \\ x_1 = +\sqrt{1/3} & a_1 = 1 \end{array} \qquad \sum_{i=0}^{1} a_i g(x_i) = 0.1094003$$

$$n = 2 \quad \begin{array}{ll} x_0 = -\sqrt{3/5} & a_0 = 5/9 \\ x_1 = 0 & a_1 = 8/9 \\ x_2 = +\sqrt{3/5} & a_2 = 5/9 \end{array} \qquad \sum_{i=0}^{2} a_i g(x_i) = 0.1093642$$

$$n = 3 \quad x_i \in \{\pm\sqrt{\tfrac{3}{7} \pm \tfrac{2}{7}\sqrt{\tfrac{6}{5}}}\} \quad a_i \in \{\tfrac{18 \pm \sqrt{30}}{36}\} \quad |F| = 2.9\text{e}{-10}$$

Zusammengesetzt: $h = (b-a)/m$, $t_j = a + (j-1)h$,
$\tau_j(x) = t_j + (x+1)h/2$, $g_j(x) = f(\tau_j(x))h/2$,

$$\int_a^b f(t)dt = \sum_{j=1}^{m} \int_{-1}^{+1} g_j(x)dx = \sum_{j=1}^{m} \sum_{i=0}^{n} a_i g_j(x_i) + \mathcal{O}(h^{2n+2}).$$

Gewöhnliche Differentialgleichungen

Zu lösen ist das Anfangswertproblem:

$$(\star) \qquad \begin{cases} \boldsymbol{y}'(t) &= \boldsymbol{f}(t, \boldsymbol{y}(t)), \quad t \in [t_0, T] \\ \boldsymbol{y}(t_0) &= \boldsymbol{y}_0 \end{cases}$$

wobei $\boldsymbol{y} \in \mathcal{C}^1([t_0, T], \mathbb{R}^n)$ und $\boldsymbol{f} \in \mathcal{C}(D, \mathbb{R}^n)$ Lipschitz stetig auf $D = [t_0, T] \times \mathbb{R}^n$ ist, d.h. $\exists L > 0$ mit

$$\|\boldsymbol{f}(t, \boldsymbol{y}_2) - \boldsymbol{f}(t, \boldsymbol{y}_1)\| \leq L\|\boldsymbol{y}_2 - \boldsymbol{y}_1\|, \quad \forall (t, \boldsymbol{y}_1), (t, \boldsymbol{y}_2) \in D.$$

Satz: Wenn $\boldsymbol{f}$ Lipschitz stetig auf D ist, gibt es genau eine Lösung $\boldsymbol{y} \in \mathcal{C}^1([t_0, T], \mathbb{R}^n)$ zu $(\star)$. (Siehe `Details`)

- Analytische Lösung? Selten.
- Diskretisierung: Zur Vereinfachung wird ein zeitliches Gitter verwendet, das aus gleichmäßig verteilten Stützstellen besteht:

$$t_k = t_0 + k\tau, \quad k = 0, 1, \ldots, M, \quad \tau = (T - t_0)/M.$$

Die Euler Methode

- Für ein ξ_k zwischen t_k und t_{k+1} erfüllt die Lösung $\boldsymbol{y}(t)$

$$\left\| \frac{\boldsymbol{y}(t_{k+1}) - \boldsymbol{y}(t_k)}{\tau} - \boldsymbol{f}(t_k, \boldsymbol{y}(t_k)) \right\| \leq \int_{t_k}^{t_{k+1}} \left\| \frac{\boldsymbol{f}(s, \boldsymbol{y}(s)) - \boldsymbol{f}(t_k, \boldsymbol{y}(t_k))}{\tau} \right\| ds$$
$$\leq L \int_{t_k}^{t_{k+1}} \left\| \frac{\boldsymbol{y}(s) - \boldsymbol{y}(t_k)}{\tau} \right\| ds \leq L \int_{t_k}^{t_{k+1}} \left\| \boldsymbol{y}'(\xi_k(s)) \right\| ds = \mathcal{O}(\tau)$$

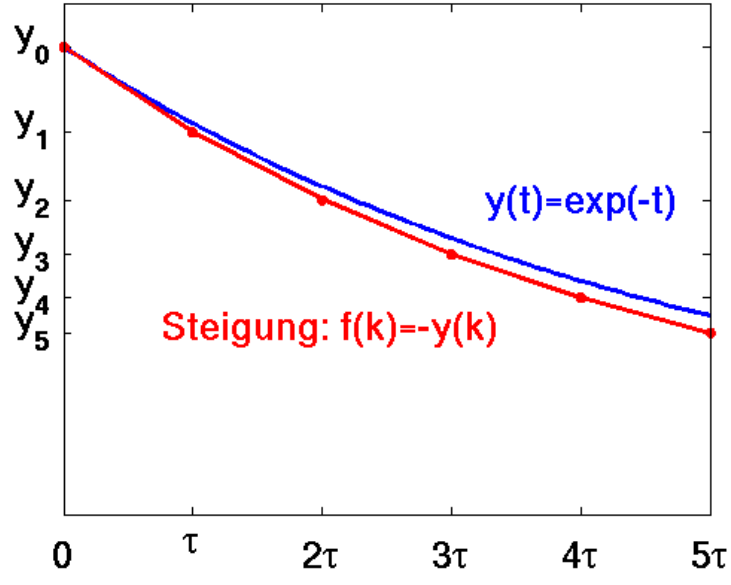
- Mit $\boldsymbol{y}_k \approx \boldsymbol{y}(t_k)$ basiert die Vorwärts Euler Methode darauf:

$$\boldsymbol{y}_{k+1} = \boldsymbol{y}_k + \tau \boldsymbol{f}(t_k, \boldsymbol{y}_k), \quad k = 0, 1, \ldots, M-1$$

- Beispiel: $f(t, y) = -y$, $y_0 = 1$, $t_0 = 0$, $T > 0$,

$$\begin{cases} y' &= -y, \quad t \in [0, T] \\ y(0) &= 1 \end{cases}$$

$\Rightarrow y(t) = e^{-t}$. Mit $\tau = 0.2$:



Die Euler Methode

Satz: Angenommen, f erfüllt die Bedingungen des Satzes 222. Seien $\boldsymbol{y}$ die eindeutige Lösung zu $(\star)$ und $\{\boldsymbol{y}_k\}_{k=0}^{M}$ durch die Vorwärts Euler Methode gegeben. Es gelten (Siehe Details)

$$\|\boldsymbol{y}(t_{k+1}) - \boldsymbol{y}_{k+1}\| \leq (1 + L\tau)\|\boldsymbol{y}(t_k) - \boldsymbol{y}_k\| + CL\tau^2, \quad C = \sup_{t\in[t_0,T]} \|\boldsymbol{y}'(t)\|$$

$$\max_{k=0,\ldots,M} \|\boldsymbol{y}(t_k) - \boldsymbol{y}_k\| \leq C\tau[\exp(L(T - t_0)) - 1]$$

- Die *Vorwärts Euler Methode* ist konvergent,
$$\boldsymbol{y}_{k+1} = \boldsymbol{y}_k + \tau \boldsymbol{f}(t_k, \boldsymbol{y}_k), \quad k = 0, 1, \ldots, M - 1$$
aber die nächsten Beispiele zeigen, die Zeitschrittweite τ muss oft sehr klein sein. Im Vergleich ist
- der *Rückwärts Euler Methode* (auch $\mathcal{O}(\tau)$) vorteilhaft:
$$\boldsymbol{y}_{k+1} = \boldsymbol{y}_k + \tau \boldsymbol{f}(t_{k+1}, \boldsymbol{y}_{k+1}), \quad k = 0, 1, \ldots, M - 1$$
- Die Vorwärts Methode ist *explizit*, weil $\boldsymbol{y}_{k+1}$ bezüglich schon berechneter Größen explizit gegeben ist.
- Die Rückwärts Methode ist *implizit*, weil $\boldsymbol{y}_{k+1}$ nur implizit definiert ist, was eine iterative Lösung kosten kann!

Wärmegleichung

- Beispiel: Eine räumliche Diskretisierung der Wärmegleichung,

$$\begin{cases} u_t &= \kappa u_{xx}, & \Omega \times (0,\infty) \\ u' &= 0, & \partial\Omega \times (0,\infty) \\ u &= u_0, & \Omega \times \{t=0\} \end{cases} \qquad \Omega = (0,1)$$

ist mit $u_i(t) \approx u(x_i,t)$, $x_i = (i-\frac{1}{2})h$, $i = 1,\dots,N$, $h = 1/N$:

$$\begin{cases} \boldsymbol{u}' &= A\boldsymbol{u} \\ \boldsymbol{u}(0) &= \boldsymbol{u}_0 \end{cases}$$

$$A = -\frac{\kappa}{h^2} D$$

$$D = \begin{bmatrix} 1 & -1 & & & 0 \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ 0 & & & -1 & 1 \end{bmatrix} \in \mathbb{R}^{N\times N}$$

- Die Lösung $\boldsymbol{u}$ erfüllt:

$$\lim_{t\to\infty} \boldsymbol{u}(t) = \text{Mittelwert}(\boldsymbol{u}_0) \cdot \mathbf{1}$$

Laut Gerschgorin gilt $\sigma(A) \le 0$, und daher gilt

$$\frac{1}{2}\frac{d}{dt}\|\boldsymbol{u}\|_2^2 = \boldsymbol{u}^\top \boldsymbol{u}' = \boldsymbol{u}^\top A\boldsymbol{u} \le 0 \quad \Rightarrow \quad \|\boldsymbol{u}(t)\|_2 \le \|\boldsymbol{u}(0)\|_2$$

Wärmegleichung

- Vorwärts Euler:

$$\boldsymbol{u}_k = \boldsymbol{u}_{k-1} + \tau A \boldsymbol{u}_{k-1} = \left(I - \frac{\kappa\tau}{h^2} D\right) \boldsymbol{u}_{k-1} = \cdots = \left(I - \frac{\kappa\tau}{h^2} D\right)^k \boldsymbol{u}_0$$

- Damit die Methode stabil ist, muss gelten:

$$\sigma\left(I - \frac{\kappa\tau}{h^2} D\right) \subset \bar{B}(0,1).$$

Die Gerschgorin Scheiben in $\mathbb{C}$ sind $B(z_i, r_i)$ wobei

$$z_1 = 1 - \frac{\kappa\tau}{h^2}, r_1 = \frac{\kappa\tau}{h^2}; \quad z_i = 1 - \frac{2\kappa\tau}{h^2}, r_i = \frac{2\kappa\tau}{h^2}; \quad z_N = 1 - \frac{\kappa\tau}{h^2}, r_N = \frac{\kappa\tau}{h^2}$$

- Die Stabilitätsbedingung ist:

$$-1 \leq 1 - \frac{4\kappa\tau}{h^2} \ (\leq 1)$$

die auffällig umständlich ist:

$$\frac{1}{N^2} = h^2 \geq 2\kappa\tau = 2\kappa \frac{T - t_0}{M} \quad \Rightarrow \quad M = \mathcal{O}(N^2)$$

Konservative Methoden

- Rückwärts Euler:

$$\boldsymbol{u}_k = \boldsymbol{u}_{k-1} + \tau A \boldsymbol{u}_k \quad \Rightarrow \quad \left(I + \frac{\kappa\tau}{h^2} D\right) \boldsymbol{u}_k = \boldsymbol{u}_{k-1}$$

Laut Gerschgorin gilt $\sigma(D) \geq 0$, und daher

$$\|\boldsymbol{u}_k\|_2^2 = \boldsymbol{u}_k^\top \boldsymbol{u}_k \leq \boldsymbol{u}_k^\top \left(I + \frac{\kappa\tau}{h^2} D\right) \boldsymbol{u}_k = \boldsymbol{u}_k^\top \boldsymbol{u}_{k-1} \leq \|\boldsymbol{u}_k\|_2 \|\boldsymbol{u}_{k-1}\|_2$$

- Es folgt $\|\boldsymbol{u}_k\|_2 \leq \|\boldsymbol{u}_{k-1}\|_2$ ohne Bedingungen auf τ!
- Wenn $B^\top = -B$ und daher $\sigma(B) \subset \imath\mathbb{R}$ gelten, ergibt sich

$$\left\{ \begin{array}{rcl} \boldsymbol{u}' & = & B\boldsymbol{u} \\ \boldsymbol{u}(0) & = & \boldsymbol{u}_0 \end{array} \right. \quad \Rightarrow \quad \frac{1}{2}\frac{d}{dt}\|\boldsymbol{u}\|_2^2 = \boldsymbol{u}^\top B \boldsymbol{u} = \frac{1}{2}\left(\boldsymbol{u}^\top B \boldsymbol{u} + \boldsymbol{u}^\top B^\top \boldsymbol{u}\right) = 0$$

und $\|\boldsymbol{u}(t)\|_2$ bleibt erhalten bei $\|\boldsymbol{u}(0)\|_2$.

- Für ein solches Problem ist eine numerische Methode mit der folgenden Eigenschaft bevorzugt.

Def: Eine Methode zur Approximation $y_k \approx y(t_k)$ der Lösung von $y' = \lambda y$ mit $y(0) = y_0$, $t_{k+1} > t_k$ heisst konservativ, wenn für beliebiges $\lambda \in \imath\mathbb{R}$ gilt

$$|y_{k+1}| = |y_k|, \quad k \in \mathbb{N}_0.$$

Wellengleichung

- Beispiel: Eine räumliche Diskretisierung der Wellengleichung,

$$\begin{cases} u_{tt} &= \nu^2 u_{xx}, & \Omega \times (0,\infty) \\ u_x &= 0, & \partial\Omega \times (0,\infty) \\ u &= u_0, & \Omega \times \{t=0\} \\ u_t &= u_1, & \Omega \times \{t=0\} \end{cases} \qquad \begin{pmatrix} \nu u_x \\ u_t \end{pmatrix}_t = \begin{pmatrix} 0 & \nu\partial_x \\ \nu\partial_x & 0 \end{pmatrix} \begin{pmatrix} \nu u_x \\ u_t \end{pmatrix}$$

ist mit $x_i = (i - \frac{1}{2})h$, $i = 1, \ldots, N$, $h = 1/N$,

$$\boldsymbol{U} = \{U_i\}_{i=1}^{2N-1} \approx \begin{bmatrix} \{\nu\partial_x u(x_{i+\frac{1}{2}}, t)\}_{i=1}^{N-1} \\ \{\partial_t u(x_i, t)\}_{i=1}^{N} \end{bmatrix}, \quad \boldsymbol{U}_0 \approx \begin{bmatrix} \{\nu\partial_x u_0(x_{i+\frac{1}{2}})\}_{i=1}^{N-1} \\ \{u_1(x_i)\}_{i=1}^{N} \end{bmatrix}$$

gegeben durch:

$$Q \in \mathbb{R}^{(N-1)\times N}$$

$$\begin{cases} \boldsymbol{U}' &= B\boldsymbol{U} \\ \boldsymbol{U}(0) &= \boldsymbol{U}_0 \end{cases} \qquad B = \frac{\nu}{h}\begin{pmatrix} 0 & Q \\ -Q^\top & 0 \end{pmatrix} \qquad Q = \begin{bmatrix} -1 & +1 & & 0 \\ & \ddots & \ddots & \\ 0 & & -1 & +1 \end{bmatrix}$$

- Die Lösung $\boldsymbol{U}$ erfüllt: $\|\boldsymbol{U}(t)\|_2 = \|\boldsymbol{U}(0)\|_2$.

Crank-Nicholson Methode

- Die Crank-Nicholson Methode (nun $\mathcal{O}(\tau^2)$) für $(\star)$ ist:

$$\frac{\boldsymbol{y}_{k+1} - \boldsymbol{y}_k}{\tau} = \frac{\boldsymbol{f}(t_k, \boldsymbol{y}_k) + \boldsymbol{f}(t_{k+1}, \boldsymbol{y}_{k+1})}{2}$$

- Für die Wellengleichung,

$$\frac{\boldsymbol{U}_{k+1} - \boldsymbol{U}_k}{\tau} = \frac{B\boldsymbol{U}_k + B\boldsymbol{U}_{k+1}}{2} \quad \Rightarrow \quad \left[I - \frac{\tau}{2}B\right]\boldsymbol{U}_{k+1} = \left[I + \frac{\tau}{2}B\right]\boldsymbol{U}_k$$

- Wegen $B^\top = -B$ gelten

$$\boldsymbol{U}^\top B\boldsymbol{U} = \tfrac{1}{2}(\boldsymbol{U}^\top B\boldsymbol{U} + \boldsymbol{U}^\top B^\top \boldsymbol{U}) = 0$$

$$\begin{aligned} \boldsymbol{U}_{k+1}^\top \boldsymbol{U}_{k+1} &= \boldsymbol{U}_{k+1}^\top [I - \tfrac{\tau}{2}B]\boldsymbol{U}_{k+1} = \boldsymbol{U}_{k+1}^\top [I + \tfrac{\tau}{2}B]\boldsymbol{U}_k \\ &= \boldsymbol{U}_{k+1}^\top \boldsymbol{U}_k + \tfrac{\tau}{2}\boldsymbol{U}_{k+1}^\top B\boldsymbol{U}_k \end{aligned}$$

und

$$\begin{aligned} \boldsymbol{U}_k^\top \boldsymbol{U}_k &= \boldsymbol{U}_k^\top [I + \tfrac{\tau}{2}B]\boldsymbol{U}_k = \boldsymbol{U}_k^\top [I - \tfrac{\tau}{2}B]\boldsymbol{U}_{k+1} \\ &= \boldsymbol{U}_k^\top \boldsymbol{U}_{k+1} - \tfrac{\tau}{2}\boldsymbol{U}_k^\top B\boldsymbol{U}_{k+1} = \boldsymbol{U}_{k+1}^\top \boldsymbol{U}_k - \tfrac{\tau}{2}\boldsymbol{U}_{k+1}^\top B^\top \boldsymbol{U}_k \end{aligned}$$

und daher gilt $\|\boldsymbol{U}_{k+1}\|_2 = \|\boldsymbol{U}_k\|_2$.

- Hausaufgabe: Zeige mit $\boldsymbol{U}_{k+1} = r(\tau B)\boldsymbol{U}_k$, $B = S^*\Lambda S$, $S^*S = I$, $\|\boldsymbol{U}_{k+1}\|_2 = \|S\boldsymbol{U}_{k+1}\|_2 = \|r(\tau\Lambda)S\boldsymbol{U}_k\|_2 = \|S\boldsymbol{U}_k\|_2 = \|\boldsymbol{U}_k\|_2$.

Konsistenz, Stabilität, Konvergenz

- Für das Anfangswertproblem ($\star$) sei eine allgemeine Einzelschritt-Methode gegeben:

$$\boldsymbol{y}_{k+1} = \boldsymbol{y}_k + \tau\phi(t_k, \boldsymbol{y}_k, \tau)$$

  wobei ϕ die gesamte Abbildung $(t_k, \boldsymbol{y}_k, \tau) \to (\boldsymbol{y}_{k+1} - \boldsymbol{y}_k)/\tau$ darstellt.
- `Beispiele`: Vorwärts Euler,

$$\phi(t, \boldsymbol{y}, \tau) = \boldsymbol{f}(t, \boldsymbol{y})$$

  Rückwärts Euler,

$$\phi(t, \boldsymbol{y}, \tau) = \boldsymbol{f}(t + \tau, \boldsymbol{y} + \tau\phi(t, \boldsymbol{y}, \tau))$$

Def: Die Methode heisst *konsistent* mit ($\star$) wenn

$$\phi(t, \boldsymbol{y}, 0) = \boldsymbol{f}(t, \boldsymbol{y})$$

Def: Die *Genauigkeitsordnung* ist ν wenn

$$\frac{\boldsymbol{y}(t + \tau) - \boldsymbol{y}(t)}{\tau} - \phi(t, \boldsymbol{y}(t), \tau) = \mathcal{O}(\tau^\nu)$$

Konsistenz, Stabilität, Konvergenz

- Beispiel: Wie auf Seite 223 gezeigt, hat die Vorwärts Euler Methode hat Genauigkeitsordnung $\nu = 1$.
- Hausaufgabe: Für die Crank-Nicholson Methode gelten

$$\|\phi(t, \boldsymbol{y}_2, \tau) - \phi(t, \boldsymbol{y}_1, \tau)\| \leq \Lambda \|\boldsymbol{y}_2 - \boldsymbol{y}_1\| \quad \text{mit} \quad \Lambda = \frac{L}{1 - \tau L/2} > 0$$

und Genauigkeitsordnung $\nu = 2$,

$$\left\| \frac{\boldsymbol{y}(t+\tau) - \boldsymbol{y}(t)}{\tau} - \phi(t, \boldsymbol{y}(t), \tau) \right\| \leq G\Lambda\tau^2, \quad G\Lambda = \frac{7}{24} \max_{t \in [t_0, T]} \|\boldsymbol{y}^{(3)}(t)\|$$

Def: Mit der Einzelschritt-Methode, $\boldsymbol{y}_{k+1} = \boldsymbol{y}_k + \tau\phi(t_k, \boldsymbol{y}_k, \tau)$, $\tau = (T - t_0)/M$, seien $\{\boldsymbol{y}_k\}_{k=0}^M$ und $\{\boldsymbol{z}_k\}_{k=0}^M$ gegeben mit Anfangswerten $\boldsymbol{y}_0$ bzw. $\boldsymbol{z}_0$. Die Methode heisst stabil wenn $\exists \tau_0 > 0$ und $c \neq c(M)$ wobei $\forall \tau \in (0, \tau_0)$,

$$\max_{0 \leq k \leq M} \|\boldsymbol{y}_k - \boldsymbol{z}_k\| \leq c \|\boldsymbol{y}_0 - \boldsymbol{z}_0\|.$$

- Bemerkung: Stabilität ist eine Konsequenz der Lipschitz Stetigkeit von ϕ.

Konsistenz, Stabilität, Konvergenz

Satz: Für das Anfangswertproblem $(\star)$ sei eine Einzelschritt-Methode gegeben: $\boldsymbol{y}_0 = \boldsymbol{y}(t_0)$,

$$\boldsymbol{y}_{k+1} = \boldsymbol{y}_k + \tau\phi(t_k, \boldsymbol{y}_k, \tau), \quad k = 0, \ldots, M-1$$

wobei für $L, \Lambda > 0$, $D = [t_0, T] \times \boldsymbol{R}^n$ und $E = D \times [0, \tau_0]$,

$$\|\boldsymbol{f}(t, \boldsymbol{y}_1 - \boldsymbol{f}(t, \boldsymbol{y}_2)\| \leq L\|\boldsymbol{y}_1 - \boldsymbol{y}_2\|, \quad \forall (t, \boldsymbol{y}_1), (t, \boldsymbol{y}_2) \in D$$

$$\|\phi(t, \boldsymbol{y}_1, \tau) - \phi(t, \boldsymbol{y}_2, \tau)\| \leq \Lambda\|\boldsymbol{y}_1 - \boldsymbol{y}_2\|, \quad \forall (t, \boldsymbol{y}_1, \tau), (t, \boldsymbol{y}_2, \tau) \in E.$$

Es gelten: (Siehe Details)

1. Die Methode ist stabil.
2. Die Methode ist konvergent genau dann wenn sie konsistent ist.
3. Mit
$$T(\tau) = \sup_{t \in [t_0, T-\tau]} \left\| \frac{\boldsymbol{y}(t+\tau) - \boldsymbol{y}(t)}{\tau} - \phi(t, \boldsymbol{y}(t), \tau) \right\|$$
folgt
$$\max_{k=0,\ldots,M} \|\boldsymbol{y}(t_k) - \boldsymbol{y}_k\| \leq T(\tau) \left(\frac{\exp(\Lambda(T - t_0)) - 1}{\Lambda} \right).$$

Bedeutung der Stabilität

- Für das Anfangswertproblem $(\star)$ seien $\{\boldsymbol{y}_k\}_{k=0}^M$ und $\{\boldsymbol{z}_k\}_{k=0}^M$ gegeben durch die Vorwärts Euler Methode mit Anfangswerten $\boldsymbol{y}_0$ bzw. $\boldsymbol{z}_0$. Es gelten:

  $$[\boldsymbol{y}_k - \boldsymbol{z}_k] = [\boldsymbol{y}_{k-1} - \boldsymbol{z}_{k-1}] + \tau[\boldsymbol{f}(t_{k-1}, \boldsymbol{y}_{k-1}) - \boldsymbol{f}(t_{k-1}, \boldsymbol{z}_{k-1})]$$

  und

$$\begin{aligned}
\|\boldsymbol{y}_k - \boldsymbol{z}_k\| &\leq \|\boldsymbol{y}_{k-1} - \boldsymbol{z}_{k-1}\| + \tau L\|\boldsymbol{y}_{k-1} - \boldsymbol{z}_{k-1}\| \\
&\leq (1 + L\tau)^k\|\boldsymbol{y}_0 - \boldsymbol{z}_0\| \leq (1 + L\tau)^M\|\boldsymbol{y}_0 - \boldsymbol{z}_0\| \\
&= (1 + L(T - t_0)/M)^M\|\boldsymbol{y}_0 - \boldsymbol{z}_0\| \\
&\leq \exp(L(T - t_0))\|\boldsymbol{y}_0 - \boldsymbol{z}_0\|
\end{aligned}$$

- Die Euler Methode ist stabil weil $\exists c > 0$ mit

  $$\max_{0\leq k\leq M} \|\boldsymbol{y}_k - \boldsymbol{z}_k\| \leq c\|\boldsymbol{y}_0 - \boldsymbol{z}_0\|$$

  aber die Konstante $c = \exp((T - t_0)L)$ kann riesig sein.

Bedeutung der Stabilität

- Beispiel: $y' = f(t, y)$, $y_0 = 0$, $[t_0, T] = [0, 1]$ und
$$f(t, y) = -1000y + 1000t^2 + 2t$$
Die exakte Lösung ist $y(t) = t^2$, $t \in [0, 1]$. Seien $\{y_k\}_{k=0}^M$ und $\{z_k\}_{k=0}^M$ gegeben durch die Vorwärts Euler Methode mit Anfangswerten $y_0 \approx z_0$.
Es gelten:

$$[y_k - z_k] = [y_{k-1} - z_{k-1}] - 1000\tau[y_{k-1} - z_{k-1}]$$

$$|y_k - z_k| \leq (1 - 1000\tau)^k |y_0 - z_0| \leq (1 - 1000\tau)^M |y_0 - z_0|$$

- Mit $\tau < 0.002$ gilt $y_M \approx 1 = y(t_M)$.
- Mit $\tau = 0.01$ werden Fehler bei jedem Schritt mit einem Faktor $|1 - 1000 \cdot 0.01| = 9$ vergrößert.
- Die Stabilitäts-Abschätzung wird nicht verletzt, aber $c = e^{1000}$.

Solche Beispiele motivieren die folgenden Definitionen.

Absolute Stabilität

Def: Eine numerische Methode zur Lösung von $(\star)$ heisst *absolut stabil*, wenn die numerische Lösung $\{y_k\}$ des Problems $y' = \lambda y$, $\Re[\lambda] < 0$, erfüllt $\lim_{k\to\infty} y_k = 0$. Die Menge der Werte $\{z = \tau\lambda \in \mathbb{C} : \Re[z] < 0, \lim_{k\to\infty} y_k = 0\}$ heisst die *Menge der absoluten Stabilität*.

- Die Menge der absoluten Stabilität für die Vorwärts Euler Methode ist $\{z \in \mathbb{C} : |z + 1| < 1\}$:

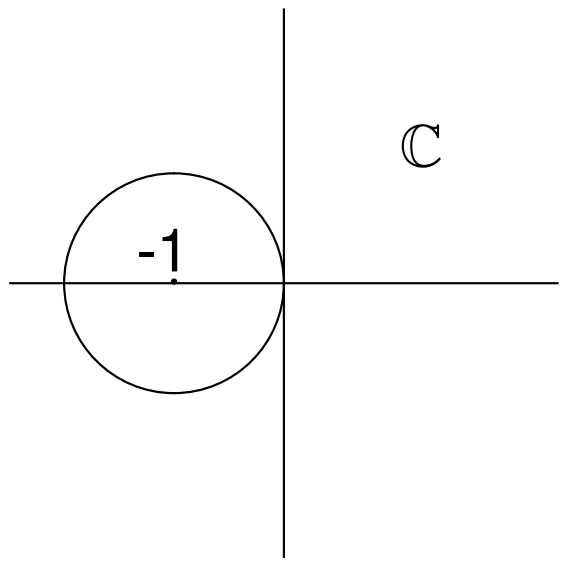
$$y_k = y_{k-1} + \tau\lambda y_{k-1} = \cdots = (1 + \tau\lambda)^k y_0$$

und

$$y_k \overset{k\to\infty}{\longrightarrow} 0 \quad \Leftrightarrow \quad |1 + \tau\lambda| < 1$$

oder mit $x = \Re[\tau\lambda]$, $y = \Im[\tau\lambda]$,

$$[(x + 1)^2 + (y - 0)^2] < 1$$



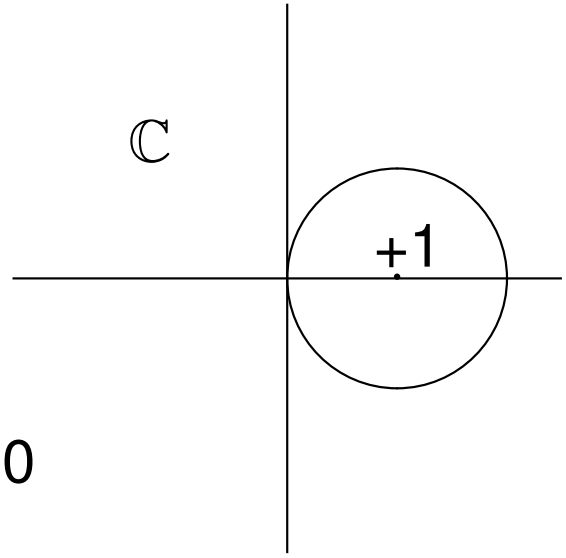
A-Stabilität

- Die Menge der absoluten Stabilität für die Rückwärts Euler Methode ist $\{z \in \mathbb{C} : \Re[z] < 0\}$:

$$y_{k+1} = y_k + \tau\lambda y_{k+1} \Rightarrow (1-\tau\lambda)y_{k+1} = y_k \Rightarrow y_{k+1} = y_k/(1-\tau\lambda)$$

und

$$y_k = y_0/(1-\tau\lambda)^k \overset{k\to\infty}{\longrightarrow} 0 \quad \Leftrightarrow$$
$$|1-\tau\lambda|^{-1} < 1 \quad \Leftrightarrow \quad 1 < |1-\tau\lambda|$$



oder mit $x = \Re[\tau\lambda]$, $y = \Im[\tau\lambda]$,

$$\left.\begin{array}{c} x < 0 \quad \text{und} \\ 1 < [(x-1)^2 + (y-0)^2] \end{array}\right\} \Rightarrow \quad x < 0$$

Def: Sei $y_{k+1} = r(\tau\lambda)y_k$, $k \in \mathbb{N}_0$, eine numerische Lösung des Problems $y' = \lambda y$, $\Re[\lambda] \leq 0$. Die Methode heisst A-stabil wenn die Funktion $r(z)$ erfüllt $|r(z)| \leq 1$, $\forall \Re[z] \leq 0$.

- Rückwärts Euler und Crank-Nicholson sind A-stabil.
- Solche Konzepte sind geeignet für folgende Probleme.

Steife Systeme

- Ein Prozess heisst *steif* wenn mindestens zwei sehr unterschiedliche Zeitskalen dabei sind.
- Physikalisches Beispiel: Strömung einer Flüssigkeit, in der chemische Reaktionen stattfinden. Es ist üblich, dass die Reaktionen viel schneller als die Strömung laufen.
- Mathematisches Beispiel:

$$\boldsymbol{x}'(t) = A\boldsymbol{x}(t), \quad AS = S\Lambda, \quad \Lambda = \mathrm{diag}\{\lambda_1, \lambda_2\}$$

$$\lambda_1, \lambda_2 < 0 \quad \text{aber} \quad |\lambda_1| \gg |\lambda_2|$$

$\boldsymbol{y}(t) = S^{-1}\boldsymbol{x}(t)$ erfüllt

$$\boldsymbol{y}'(t) = S^{-1}\boldsymbol{x}'(t) = S^{-1}A\boldsymbol{x}(t) = S^{-1}AS\boldsymbol{y}(t) = \Lambda\boldsymbol{y}(t)$$

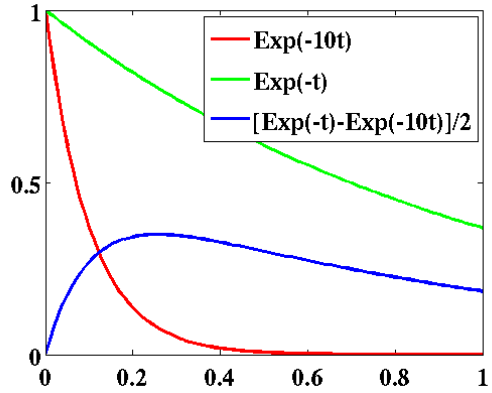
$\boldsymbol{y}(t) = \exp(\Lambda t)\boldsymbol{y}(0) \Rightarrow \boldsymbol{x}(t) = S\exp(\Lambda t)S^{-1}\boldsymbol{x}(0)$ oder

$$x_1(t) = Ae^{\lambda_1 t} + Be^{\lambda_2 t}, \quad x_2(t) = Ce^{\lambda_1 t} + De^{\lambda_2 t}$$

Jedes $x_i(t)$ enthält eine Komponente $e^{\lambda_1 t}$, die viel schneller zerfällt als die Komponente $e^{\lambda_2 t}$.

Steife Systeme

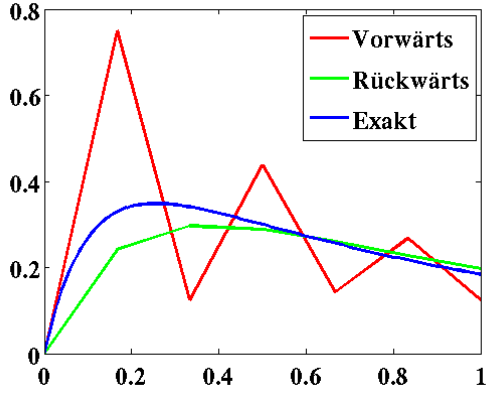
- Mit $S = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$, $\Lambda = \begin{bmatrix} -10 & 0 \\ 0 & -1 \end{bmatrix}$, $\boldsymbol{x}(0) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, gilt $x_1(t) = (e^{-t} - e^{-10t})/2$:

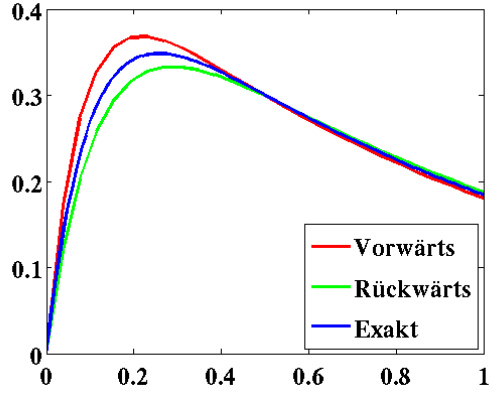


x_1 hat eine schnellere Komponente e^{-10t} und eine langsamere Komponents e^{-t}.

- Numerische Ergebnisse in $[t_0, T] = [0, 1]$ mit

$M = 7$





$M = 27$.

- Auch für steife Probleme ist ein kleines τ notwendig, wo die Lösung sich schnell ändert.

Runge-Kutta Methoden

- Zur numerischen Lösung des Anfangswertproblems $(\star)$ ist eine Runge-Kutta (RK) Methode

$$\boldsymbol{y}_{k+1} = \boldsymbol{y}_k + \phi(t_k, \boldsymbol{y}_k, \tau), \quad k = 0, \ldots, M-1$$

durch das System mit q Stufen definiert:

$$\begin{cases} \boldsymbol{y}_{k,i} &= \boldsymbol{y}_k + \tau \sum_{j=1}^{q} a_{ij} \boldsymbol{f}(t_k + \theta_j \tau, \boldsymbol{y}_{k,j}), \quad i = 1, \ldots, q \\ \boldsymbol{y}_{k+1} &= \boldsymbol{y}_k + \tau \sum_{i=1}^{q} b_i \boldsymbol{f}(t_k + \theta_i \tau, \boldsymbol{y}_{k,i}) \end{cases}$$

wobei ϕ die gesamte Abbildung $(t_k, \boldsymbol{y}_k, \tau) \to (\boldsymbol{y}_{k+1} - \boldsymbol{y}_k)/\tau$ darstellt.

- Die Methode wird mit einer (Butcher) Tabelle dargestellt:

$$\begin{array}{c|c} \boldsymbol{\theta} & A \\ \hline & \boldsymbol{b}^\top \end{array} \quad \text{wobei} \quad A \in \mathbb{R}^{q \times q}, \quad \boldsymbol{b}, \boldsymbol{\theta} \in \mathbb{R}^q$$

- Die Methode heisst *explizit* wenn $a_{ij} = 0$ für $i \leq j$ gilt, und sie heisst *implizit* wenn $a_{ii} \neq 0$ für mindestens ein i gilt.

Runge-Kutta Methoden

- Eine wohl bekannte RK-Methode ist:

$$\begin{array}{c|c} \boldsymbol{\theta} & A \\ \hline & \boldsymbol{b}^\top \end{array} \quad \rightarrow \quad \begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array}$$

oder

$$\begin{cases} \boldsymbol{y}_{k,1} &= \boldsymbol{y}_k \\ \boldsymbol{y}_{k,2} &= \boldsymbol{y}_k + \frac{\tau}{2}\boldsymbol{f}(t_k, \boldsymbol{y}_{k,1}) \\ \boldsymbol{y}_{k,3} &= \boldsymbol{y}_k + \frac{\tau}{2}\boldsymbol{f}(t_k + \frac{\tau}{2}, \boldsymbol{y}_{k,2}) \\ \boldsymbol{y}_{k,4} &= \boldsymbol{y}_k + \tau\boldsymbol{f}(t_k + \frac{\tau}{2}, \boldsymbol{y}_{k,3}) \\ \boldsymbol{y}_{k+1} &= \boldsymbol{y}_k + \tau[\frac{1}{6}\boldsymbol{f}(t_k, \boldsymbol{y}_{k,1}) + \frac{1}{3}\boldsymbol{f}(t_k + \frac{\tau}{2}, \boldsymbol{y}_{k,2}) \\ & \qquad + \frac{1}{3}\boldsymbol{f}(t_k + \frac{\tau}{2}, \boldsymbol{y}_{k,3}) + \frac{1}{6}\boldsymbol{f}(t_k + \tau, \boldsymbol{y}_{k,4})] \end{cases}$$

- Wenn $\boldsymbol{y} \in \mathcal{C}^{(5)}([t_0, T])$ gilt, erfüllt die Methode

$$\max_{k=0,\ldots,M} \|\boldsymbol{y}(t_k) - \boldsymbol{y}_k\| = \mathcal{O}(\tau^4).$$

Adaptive Runge-Kutta Methoden

- Nun sei das Gitter nicht notwendigerweise gleichmäßig.
- Die Zeitschrittweite wird automatisch bestimmt.
- Eine gröbere RK-Methode (mit $\tilde{\boldsymbol{b}}$) wird in eine genauere RK-Methode (mit $\boldsymbol{b}$) *eingebettet*: (Siehe `Beispiel`)

$$\begin{array}{c|c} \boldsymbol{\theta} & A \\ \hline & \boldsymbol{b}^\top \\ & \tilde{\boldsymbol{b}}^\top \end{array}$$

$$\begin{aligned} \boldsymbol{y}_{k,i} &= \boldsymbol{y}_k + \tau \textstyle\sum_{j=1}^{q} a_{ij} \boldsymbol{f}(t_k + \theta_j \tau, \boldsymbol{y}_{k,j}) \\ \boldsymbol{y}_{k+1} &= \boldsymbol{y}_k + \tau \textstyle\sum_{i=1}^{q} b_i \boldsymbol{f}(t_k + \theta_i \tau, \boldsymbol{y}_{k,i}) \\ \tilde{\boldsymbol{y}}_{k+1} &= \boldsymbol{y}_k + \tau \textstyle\sum_{i=1}^{q} \tilde{b}_i \boldsymbol{f}(t_k + \theta_i \tau, \boldsymbol{y}_{k,i}) \end{aligned}$$

- Gegeben sei $\boldsymbol{y}_k$, und ein einziger Schritt mit der genaueren Methode ($\mathcal{O}(\tau^{\nu+1})$) bzw. mit der gröberen Methode ($\mathcal{O}(\tau^{\nu})$) werden folgendermaßen dargestellt:

$$\boldsymbol{y}_{k+1} = \boldsymbol{y}_k + \tau\phi(t_k, \boldsymbol{y}_k, \tau), \quad \tilde{\boldsymbol{y}}_{k+1} = \boldsymbol{y}_k + \tau\tilde{\phi}(t_k, \boldsymbol{y}_k, \tau).$$

- $\boldsymbol{y}_{k+1}$ und $\tilde{\boldsymbol{y}}_{k+1}$ werden verwendet, um das geeignete τ zu bestimmen.

Adaptive Runge-Kutta Methoden

- Für den einzigen $k$ten Schritt sei $\boldsymbol{z}(t)$ die Lösung von $(\star)$ mit $(t_0, \boldsymbol{y}_0)$ ersetzt durch $(t_k, \boldsymbol{y}_k)$.
- Seien die jeweiligen Abbruchsfehler so bezeichnet:

$$\begin{aligned} \mathcal{O}(\tau^{\nu+1}) = T_{k+1}(\tau) &= [\boldsymbol{z}(t_{k+1}) - \boldsymbol{z}(t_k)]/\tau - \phi(t_k, \boldsymbol{z}(t_k), \tau) \\ &= [\boldsymbol{z}(t_{k+1}) - \boldsymbol{y}_k]/\tau - \phi(t_k, \boldsymbol{y}_k, \tau) \end{aligned}$$

bzw.

$$\begin{aligned} \mathcal{O}(\tau^{\nu}) = \tilde{T}_{k+1}(\tau) &= [\boldsymbol{z}(t_{k+1}) - \boldsymbol{z}(t_k)]/\tau - \tilde{\phi}(t_k, \boldsymbol{z}(t_k), \tau) \\ &= [\boldsymbol{z}(t_{k+1}) - \boldsymbol{y}_k]/\tau - \tilde{\phi}(t_k, \boldsymbol{y}_k, \tau) \end{aligned}$$

- Wenn nach ϕ bzw. $\tilde{\phi}$ durch die letzten Gleichungen aufgelöst wird, ergeben sich

$$\mathcal{O}(\tau^{\nu+1}) = \tilde{T}_{k+1}(\tau) = \frac{\boldsymbol{z}(t_{k+1}) - \boldsymbol{y}_{k+1}}{\tau}$$

und

$$\mathcal{O}(\tau^{\nu}) = \tilde{T}_{k+1}(\tau) = \frac{\boldsymbol{z}(t_{k+1}) - \tilde{\boldsymbol{y}}_{k+1}}{\tau}.$$

Adaptive Runge-Kutta Methoden

- Die Abbruchsfehler können so kombiniert werden:

$$\begin{aligned}\mathcal{O}(\tau^\nu) = \tilde{T}_{k+1}(\tau) &= \frac{\boldsymbol{z}(t_{k+1}) - \tilde{\boldsymbol{y}}_{k+1}}{\tau} \\ &= \frac{\boldsymbol{z}(t_{k+1}) - \boldsymbol{y}_{k+1}}{\tau} + \frac{\boldsymbol{y}_{k+1} - \tilde{\boldsymbol{y}}_{k+1}}{\tau} \\ &= \underbrace{T_{k+1}(\tau)}_{\mathcal{O}(\tau^{n+1})} + \underbrace{\frac{\boldsymbol{y}_{k+1} - \tilde{\boldsymbol{y}}_{k+1}}{\tau}}_{\text{signifikanter}}\end{aligned}$$

- Mit

$$C\tau^\nu \approx \tilde{T}_{k+1}(\tau) \approx \frac{\boldsymbol{y}_{k+1} - \tilde{\boldsymbol{y}}_{k+1}}{\tau} = \sum_{i=1}^{q} (b_i - \tilde{b}_i)\boldsymbol{f}(t_k + \theta_i\tau, \boldsymbol{y}_{k,i})$$

ergibt sich

$$\tilde{T}_{k+1}(\delta \cdot \tau) \approx C(\delta \cdot \tau)^\nu = \delta^\nu C\tau^\nu \approx \delta^\nu \frac{\boldsymbol{y}_{k+1} - \tilde{\boldsymbol{y}}_{k+1}}{\tau}$$

- Um die Genauigkeit $\|\tilde{T}_{k+1}(\delta \cdot \tau)\| \leq \varepsilon$ zu erreichen, wird δ ausgewählt, um die Ungleichung zu erfüllen,

$$\delta^\nu \|\boldsymbol{y}_{k+1} - \tilde{\boldsymbol{y}}_{k+1}\| \leq \varepsilon\tau$$

aber zur Sicherheit unter den Einschränkungen $\delta \in [\delta_{\min}, \delta_{\max}]$.

Adaptive Runge-Kutta Methoden

Die adaptive Strategie:

- Gegeben seien:
  - Absolute Fehlertoleranz ε,
  - Max- und Min-Werte $\tau_{\max}$ bzw. $\tau_{\min}$ für τ,
  - Max- und Min-Werte $\delta_{\max}$ bzw. $\delta_{\min}$ für δ.
- Anfänglich gelten $\boldsymbol{y}_0 = \boldsymbol{y}(t_0)$ und $\tau_0 = \tau_{\max}$.
- Für $k \geq 0$ sei ein τ_k vom letzten Zeitschritt gegeben.
- Dies wird verwendet, um $\boldsymbol{y}_{k+1}$ zu berechnen.
- Berechne (effizient)

$$\delta = \max\left\{\delta_{\min}, \min\left\{\delta_{\max}, \left(\frac{\varepsilon\tau_k}{\|\boldsymbol{y}_{k+1} - \tilde{\boldsymbol{y}}_{k+1}\|}\right)^{1/\nu}\right\}\right\}.$$

und

$$\tau = \min\{T - t_{k+1}, \tau_{\max}, \max\{\tau_{\min}, \delta \cdot \tau_k\}\}.$$

- Falls $\delta < 1$ gilt, berechne $\boldsymbol{y}_{k+1}$ erneut mit $\tau_k = \tau$, bis $\delta \geq 1$ gilt.
- Sonst setze $t_{k+1} = t_k + \tau_k$ und $\tau_{k+1} = \tau$ weiter, bis $t_{k+1} = T$.

Randwertprobleme

- Zu lösen ist das *Sturm-Liouville* Randwertproblem:

$$(\dagger) \quad \begin{cases} -(pu')' + qu &= f, \quad x \in \Omega \\ \alpha u + \omega D_n u &= g, \quad x \in \partial\Omega \end{cases} \qquad \Omega = [a, b]$$

  wobei $D_n u = -u'|_{x=a}, u'|_{x=b}$ und

$$0 < \check{p} \le p(x) \le \hat{p}, \quad 0 < \check{q} \le q(x) \le \hat{q}.$$

- Für verschiedene Werte von α und ω hat man folgende Arte von Randbedingungen (RB):

  $\omega = 0$: Dirichlet, $\alpha = 0$: Neumann, $\alpha, \omega \neq 0$: Robin.

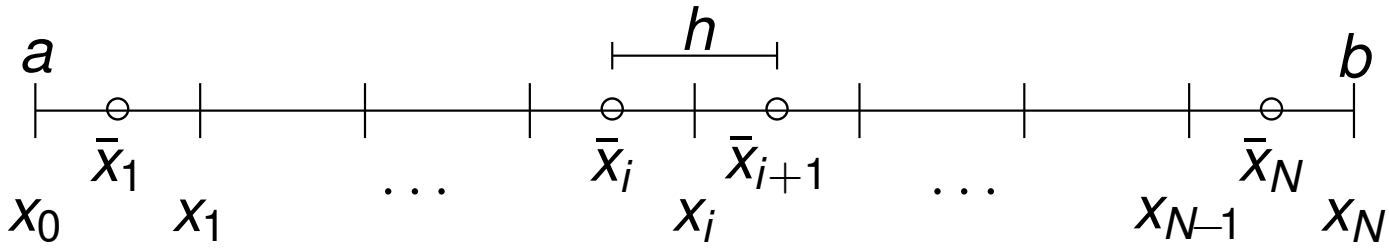
- Seien v mit $\alpha v + \omega D_n v = g$ auf $\partial\Omega$ und $\tilde{f} = f + (pv')' - qv$. Dann mit $f \to \tilde{f}$ und $g \to 0$ wird $(\dagger)$ von $\tilde{u} = u - v$ erfüllt.
- Also sei $g = 0$ in $(\dagger)$.
- Ergibt sich durch Minimierung des Funktionals,

$$J(u) = \int_\Omega [q(u - f/q)^2 + p|u'|^2]dx, \quad u \in H^1(\Omega)$$

  wobei mit $H^0 = L^2$ für die Funktionenräume gilt
  $\|v\|^2_{L^2(\Omega)} = \int_\Omega |v|^2 dx, \quad \|v\|^2_{H^k(\Omega)} = \|v\|^2_{H^{k-1}(\Omega)} + \|v^{(k)}\|_{L^2(\Omega)}$

Finite Differenzen für Randwertprobleme

- Das Problem (†) mit $\alpha = 0 = g$ und $\omega > 0$ (Neumann RB) wird durch Finite Differenzen folgendermaßen diskretisiert.
- Das Gitter: $x_i = a + ih$, $h = (b-a)/N$, $\bar{x}_i = (x_{i-1} + x_i)/2$,



- Seien $\boldsymbol{x} = \{x_i\}_{i=0}^N$, $\bar{\boldsymbol{x}} = \{\bar{x}_i\}_{i=1}^N$ und $\bar{u}_h(x) = \bar{u}_1,\ x \in [a, \bar{x}_1]; \quad \bar{u}_N,\ x \in [\bar{x}_N, b]; \quad \sum_{i=1}^N \bar{u}_i \ell(x; \bar{\boldsymbol{x}}),$ sonst.
- Mit den Notationen $v_i \approx v(x_i)$ und $\bar{v}_i \approx v(\bar{x}_i)$ wird (†) mit zentralen Differenzen für $i = 2, \ldots, N-1$ so diskretisiert:

$$\begin{cases} (p\bar{u}')_i' & \approx & \dfrac{(pu')_i - (pu')_{i-1}}{h} \\ & \approx & \dfrac{1}{h}\left[p_i \dfrac{\bar{u}_{i+1} - \bar{u}_i}{h} - p_{i-1}\dfrac{\bar{u}_i - \bar{u}_{i-1}}{h}\right] \\ (q\bar{u})_i & \approx & \bar{q}_i \bar{u}_i \end{cases}$$

Finite Differenzen für Randwertprobleme

- Für $i = 1, N$ mit $(pu')_0 = 0 = (pu')_N$,

$$(p\bar{u}')'_1 \approx p_1 \frac{\bar{u}_2 - \bar{u}_1}{h^2}, \quad (p\bar{u}')'_N \approx -p_{N-1} \frac{\bar{u}_N - \bar{u}_{N-1}}{h^2}$$

- Seien $\boldsymbol{p} = \{p_i\}_{i=1}^{N-1}$, $\bar{\boldsymbol{q}} = \{\bar{q}_i\}_{i=1}^{N}$, $\bar{\boldsymbol{u}} = \{\bar{u}_i\}_{i=1}^{N}$, $\bar{\boldsymbol{f}} = \{\bar{f}_i\}_{i=1}^{N}$.
- Das algebraische System für diese Diskretisierung ist:

$$(\dagger)_h \quad A\bar{\boldsymbol{u}} = \bar{\boldsymbol{f}}$$

wobei:

$$A = Q^\top \mathrm{D}(\boldsymbol{p}) Q + \mathrm{D}(\bar{\boldsymbol{q}})$$

$$\mathrm{D}(\boldsymbol{v}) = \mathrm{diag}(\boldsymbol{v} \in \mathbb{R}^n) \in \mathbb{R}^{n \times n}$$

$$Q = \frac{1}{h} \begin{bmatrix} -1 & 1 & & 0 \\ & \ddots & \ddots & \\ 0 & & -1 & 1 \end{bmatrix} \in \mathbb{R}^{(N-1) \times N}.$$

Satz: Für $p \in \mathcal{C}^3(\Omega)$, $q \in \mathcal{C}^0(\Omega)$, $f \in \mathcal{C}^0(\Omega)$, $g = 0$, $\alpha = 0$ und $\omega = 1$ seien $u \in \mathcal{C}^4(\Omega)$ die Lösung von $(\dagger)$ und $\boldsymbol{u}$ die Lösung von $(\dagger)_h$. Dann gilt $\|u - \bar{u}_h\|_{H^1} = \mathcal{O}(h)$.

- Siehe `Details` und bemerke $\|u - \bar{u}_h\|_{L^2} = \mathcal{O}(h^2)$.
- `Hausaufgabe`: Diskretisiere das Problem $(\dagger)$ mit den anderen Randbedingungen auf einem geeigneten Gitter.

Rayleigh-Ritz Methode

- Das Randwertproblem (†) mit Neumann Randbedingungen ergibt sich durch Minimierung das Funktionals,

$$J(u) = \int_\Omega [q(u - f/q)^2 + p|u'|^2]dx$$

- Die Richtungsableitungen des Funktionals an der Stelle u in die Richtung einer Störung v sind gegeben durch:

$$\begin{aligned}
\frac{\delta J}{\delta u}(u; v) &= \lim_{\epsilon \to 0} \frac{J(u + \epsilon v) - J(u)}{\epsilon} = \lim_{\epsilon \to 0} \frac{d}{d\epsilon} J(u + \epsilon v) \\
&= 2 \int_\Omega [(qu - f)v + pu'v']dx \\
&= 2 \int_\Omega [(qu - f) - (pu')']vdx + 2pu'v|_{\partial\Omega}
\end{aligned}$$

- Wenn $J(u) = \min$ gilt, muss $\delta J/\delta u(u; v) = 0$ für alle zulässigen Störungen v gelten.

Variationelle Methoden

- Für ein beliebiges $B(x_0, \delta) \subset \Omega$, sei

$$v_0^\delta(x) = \begin{cases} 1/(2\delta), & x \in B(x_0, \delta) \\ 0, & \text{sonst} \end{cases}$$

Laut dem Mittelwertsatz für Integrale [11] $\exists \xi \in B(x_0, \delta)$ mit

$$0 = \frac{1}{2}\frac{\delta J}{\delta u}(u; v_0^\delta) = [(qu - f) - (pu')'](\xi) \underbrace{\int_\Omega v_0^\delta \, dx}_{=1} \xrightarrow{\delta \to 0} [(qu - f) - (pu')'](x_0)$$

oder $qu - f - (pu')' = 0$ in Ω,
d.h. die Differentialgleichung in (†) ergibt sich.

- Für ein beliebiges $x_1 \in \partial\Omega$, sei

$$v_1^\delta(x) = \begin{cases} 1, & x \in B(x_1, \delta) \cap \Omega \\ 0, & \text{sonst} \end{cases}$$

Da u die Differentialgleichung erfüllt folgt

$$0 = \frac{1}{2}\frac{\delta J}{\delta u}(u; v_1^\delta) = pu' v_1^\delta|_{\partial\Omega} + \int_\Omega \underbrace{[(qu - f) - (pu')']}_{=0} v_1^\delta \, dx \xrightarrow{\delta \to 0} [pu'](x_1)$$

oder $u' = 0$ auf $\partial\Omega$, d.h.
die Neumann Randbedingung in (†) ergibt sich *natürlich*.

Schwache Formulierung von Randwertproblemen

- Wenn andere Randbedingungen, z.B. Dirichlet, vorgegeben werden, darf es keine Störung am Rand geben, und so muss $v = 0$ am $\partial\Omega$ gelten. Dann ist das letzte Argument über die Randbedingung hinfällig.
- Nachdem partielle Integration wie oben durchgeführt wird, bis keine Ableitungen von v verbleiben, ergibt sich die *starke Formulierung* (†) der Optimalitätsbedingung für J, für welche höhere Regularität von u verlangt wird.
- Wenn keine partielle Integration durchgeführt wird, und v und u müssen gleiche aber niedrigere Glattheit haben, ergibt sich die *schwache Formulierung* der Optimalitätsbedingung für J:

$$(\ddagger) \quad \int_\Omega [quv + pu'v']dx = \int_\Omega fvdx, \quad \forall v \in H^1(\Omega).$$

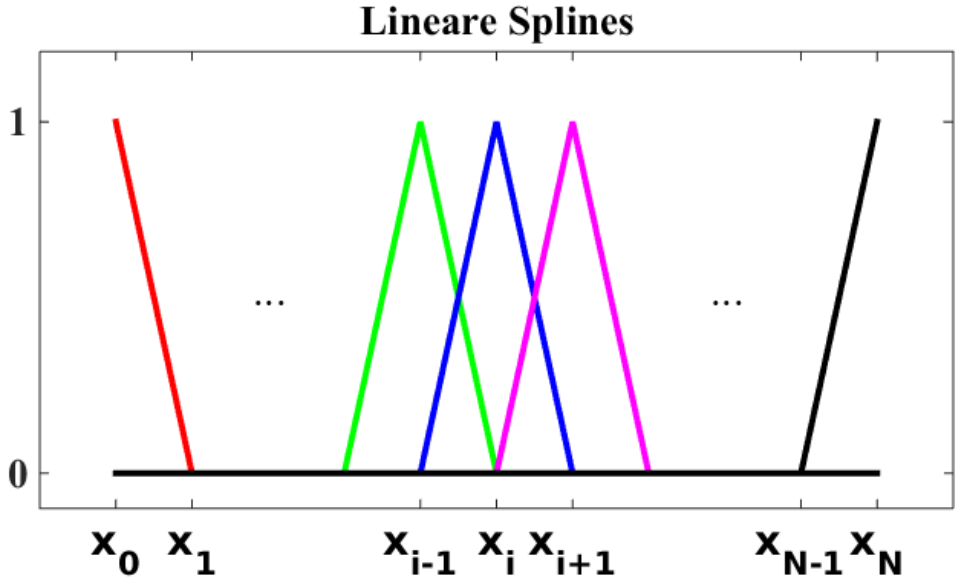
Finite Elemente für Randwertprobleme

- Mit dem Gitter

$$\boldsymbol{x} = \{x_i\}_{i=0}^{N}, \quad x_i = a + hi, \quad h = (b-a)/N$$

wird $(\ddagger)$ über N Elemente formuliert, jedes mit einer Zelle $[x_{i-1}, x_i]$ und mit einer linearen Darstellung der Lösung, d.h.

- Basisfunktionen aus $\mathcal{S}^1(\Omega) = \mathcal{S}^1(\boldsymbol{x})$ werden verwendet:



$$u_h(x) = \sum_{i=0}^{N} u_i s_i(x)$$

$$u_i = u_h(x_i)$$

$$\boldsymbol{u} = \{u_i\}_{i=0}^{N} = u_h(\boldsymbol{x})$$

- Die diskrete Formulierung von $(\ddagger)$ ist

$$(\ddagger)_h \qquad \int_\Omega [q u_h s_h + p u_h' s_h'] dx = \int_\Omega f s_h dx, \quad \forall s_h \in \mathcal{S}^1(\Omega)$$

Finite Elemente für Randwertprobleme

- Das System $(\ddagger)_h$ für die Koeffizienten $\boldsymbol{u} = u_h(\boldsymbol{x})$ wird mit

$$\mathcal{A}(v,w) = \int_\Omega [qvw + pv'w']dx, \quad \mathcal{F}(v) = \int_\Omega fvdx$$

  explizit so umgeschrieben

$$\boldsymbol{A}\boldsymbol{u} = \boldsymbol{f}, \quad A = \{\mathcal{A}(s_i, s_j)\}_{i,j=0}^N, \quad \boldsymbol{f} = \{\mathcal{F}(s_j)\}_{j=0}^N.$$

Satz: Für $0 < \check{p} \leq p(x) \leq \hat{p}$, $0 < \check{q} \leq q(x) \leq \hat{q}$, $\forall x \in \Omega$, $p, q, f \in L^2(\Omega)$ seien $u \in H^2(\Omega)$ und $u_h \in \mathcal{S}^1(\Omega)$ die Lösungen von $(\ddagger)$ bzw. $(\ddagger)_h$. Dann gilt $\|u - u_h\|_{H^1} = \mathcal{O}(h)$.

- Siehe `Details` und bemerke $\|u - u_h\|_{L^2} = \mathcal{O}(h^2)$.
- Nun wird das Gleichungssystem mit einer einfachen Mittelpunkt Regel für die jeweiligen Integrale approximiert. Gebräuchlich wird Gauß Quadratur verwendet.

Finite Elemente mit Quadratur für Randwertprobleme

- Die Approximation $\tilde{u}_h \approx u_h$ wird bezeichnet mit

$$\tilde{u}_h(x) = \sum_{i=0}^{N} \tilde{u}_i s_i(x), \quad \tilde{\boldsymbol{u}} = \{\tilde{u}_i\}_{i=0}^{N} = \tilde{u}_h(\boldsymbol{x}).$$

- Wegen des lokalen Trägers der Splines gilt für $|i-j| > 1$,

$$\int_\Omega s_i s_j dx = \int_\Omega s_i' s_j' dx = 0.$$

- Für $i = 0, \ldots, N-1$ mit $\bar{q}_{i+1} = q(\bar{x}_{i+1})$,

$$\int_\Omega q s_i s_{i+1} dx \approx \bar{q}_{i+1} \int_0^h \left(1 - \frac{x}{h}\right)\left(\frac{x}{h}\right) dx = \frac{h}{6}\bar{q}_{i+1}$$

und mit $\bar{p}_{i+1} = p(\bar{x}_{i+1})$,

$$\int_\Omega p s_i' s_{i+1}' dx \approx \bar{p}_{i+1} \int_0^h \left(-\frac{1}{h}\right)\left(+\frac{1}{h}\right) dx = -\frac{1}{h}\bar{p}_{i+1}$$

- Diese sind die Fälle für $|i-j| = 1$. Verblieben sind nur die Fälle für $|i-j| = 0$:

Finite Elemente mit Quadratur für Randwertprobleme

- Für $i = 0, \ldots, N$ mit $\bar{q}_0 = 0 = \bar{q}_{N+1}$ für auslaufende Teilintervalle,

$$\int_\Omega q s_i^2 dx = \bar{q}_i \int_0^h \left(\frac{x}{h}\right)^2 dx + \bar{q}_{i+1} \int_h^{2h} \left(2 - \frac{x}{h}\right)^2 dx = \frac{h}{3}(\bar{q}_i + \bar{q}_{i+1})$$

und ebenfalls mit $\bar{p}_0 = 0 = \bar{p}_{N+1}$,

$$\int_\Omega p s_i'^2 dx = \bar{p}_i \int_0^h \left(\frac{1}{h}\right)^2 dx + \bar{p}_{i+1} \int_h^{2h} \left(-\frac{1}{h}\right)^2 dx = \frac{1}{h}(\bar{p}_i + \bar{p}_{i+1})$$

- Für $j = 0, \ldots, N$ mit $\bar{f}_{j+1} = f(\bar{x}_{j+1})$ und $\bar{f}_0 = 0 = \bar{f}_{N+1}$ für auslaufende Teilintervalle,

$$\begin{aligned} \int_\Omega f s_j dx &= \sum_{i=1}^{N} \int_{x_{i-1}}^{x_i} f s_j dx \approx \bar{f}_j \int_{x_{j-1}}^{x_j} s_j dx + \bar{f}_{j+1} \int_{x_j}^{x_{j+1}} s_j dx \\ &= \bar{f}_j \int_0^h \frac{x}{h} dx + \bar{f}_{j+1} \int_h^{2h} \left(2 - \frac{x}{h}\right) dx = \frac{h}{2}(\bar{f}_j + \bar{f}_{j+1}) \end{aligned}$$

Finite Elemente mit Quadratur für Randwertprobleme

- Seien

$$\bar{\boldsymbol{p}} = \{\bar{p}_i\}_{i=1}^N,\ \bar{\boldsymbol{q}} = \{\bar{q}_i\}_{i=1}^N,\ \bar{\boldsymbol{f}} = \{\bar{f}_i\}_{i=1}^N \ \in \mathbb{R}^N.$$

und

$$\mathrm{D}(\boldsymbol{v}) = \mathrm{diag}(\boldsymbol{v}) \in \mathbb{R}^n \quad \text{für} \quad \boldsymbol{v} \in \mathbb{R}^n.$$

- Das Gleichungssystem für $\tilde{\boldsymbol{u}} = \tilde{u}_h(\boldsymbol{x})$ ist

$$(\tilde{\ddagger})_h \qquad A\tilde{\boldsymbol{u}} = \tfrac{1}{2} R^\top \boldsymbol{f}$$

mit

$$A = Q^\top \mathrm{D}(\boldsymbol{p}) Q + \tfrac{1}{6} R^\top \mathrm{D}(\boldsymbol{q}) R + \tfrac{1}{6} \mathrm{D}(R^\top \boldsymbol{q})$$

wobei $Q, R \in \mathbb{R}^{N \times (N+1)}$ gegeben sind durch

$$Q = \frac{1}{h} \begin{bmatrix} -1 & +1 & & 0 \\ & \ddots & \ddots & \\ 0 & & -1 & +1 \end{bmatrix}, \quad R = \begin{bmatrix} 1 & 1 & & 0 \\ & \ddots & \ddots & \\ 0 & & 1 & 1 \end{bmatrix}.$$

Finite Elemente mit Quadratur für Randwertprobleme

Satz: Für $p \in \mathcal{C}^3(\Omega)$, $q \in \mathcal{C}^2(\Omega)$ und $f \in \mathcal{C}^2(\Omega)$ seien $u \in \mathcal{C}^4(\Omega)$ die Lösung von $(\ddagger)$ und $\tilde{u}_h \in \mathcal{S}^1(\Omega)$ die Lösung von $(\tilde{\ddagger})_h$. Dann gilt $\|u - \tilde{u}_h\|_{H^1} = \mathcal{O}(h)$.

- Siehe `Details` und bemerke $\|u - \tilde{u}_h\|_{L^2} = \mathcal{O}(h^2)$.
- `Hausaufgabe`: Minimiere das Funktional

$$K(u) = \int_\Omega [q(u - f/q)^2 + p(u'')^2]dx$$

über den `Funktionenraum` $H^2(\Omega)$ und zeige, wenn $u \in C^4(\Omega)$ gilt, muss das minimierende u erfüllen

$$\begin{cases} (pu'')'' + qu &= f, & x \in \Omega \\ (pu'') = (pu'')' &= 0, & x \in \partial\Omega \end{cases}$$

Weiters für das über $\mathcal{S}^2(\Omega)$ minimierende u_h gilt

$$\|u - u_h\|_{L^2} = \mathcal{O}(h^2).$$

- Für weitere Details siehe das `Skriptum` über Numerik für Differentialgleichungen.