Gleitkommadarstellung nach IEEE 754

$x_{1,2} = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$

Bitverteilung (single/double) normalisiert: 1,xxx

Vorzeichen s(1) Exponent e (8/11) Mantisse f (23/52)

Wert $z = (-1)^s \cdot 1,f \cdot 2^{e-127}$ Genauigkeit: z^{-f}

$00000000_2 = 0 = -127 + 127$ $00000001_2 = 1 = -126 + 127$ minimal

$11111111_2 = \infty = +128 + 127$ $11111110_2 = 254 = +127 + 127$ maximal

$\frac{11}{10} \cdot 1 = \frac{11}{10} = 1 + \frac{1}{10}$

$\frac{1}{10} \cdot 2 = \frac{2}{10} = 0 + \frac{2}{10}$

$\frac{2}{10} \cdot 2 = \frac{4}{10} = 0 + \frac{4}{10}$

$\frac{4}{10} \cdot 2 = \frac{8}{10} = 0 + \frac{8}{10}$

$\frac{8}{10} \cdot 2 = \frac{16}{10} = 1 + \frac{6}{10}$

$\frac{6}{10} \cdot 2 = \frac{12}{10} = 1 + \frac{2}{10}$

$-\frac{11}{10} = -1,\overline{0001}_2 = -1,0001\overline{1001}_2$

Vorzeichen $s = 1$ Exponent $e = 0 + 127 = 127$

Mantisse $f = 00011001100110011001101$

letzte Mantissebit ergibt sich nach Rundungsregel

$y \mid 0x$ mit x "beliebig" → Abrunden

$y \mid 1x$ mit x "nicht nur Nullen" → Aufrunden

$y1 \mid 1x$ mit x "nur Nullen" → Aufrunden

$y0 \mid 1x$ mit x "nur Nullen" → Abrunden

Maschinengenauigkeit = kleinste positive Zahl ε, sodass $rd(1+\varepsilon) = 1$

$\varepsilon = \frac{1}{2} \cdot B^{1-t}$ bei IEEE: $\varepsilon = \frac{1}{2} \cdot 2^{1-24} = 2^{-24}$

$(a \; op_M \; b) = rd_M(a \; op \; b) = (a \; op \; b) \cdot (1+\varepsilon_i)$ mit $|\varepsilon_i| \leq \varepsilon_{ma}$, $\varepsilon_i \varepsilon_j = 0$

Rundungsfehler absolute Fehler $f_{abs} = |x - rd(x)|$ relativ Fehler $f_{rel} = \left|\frac{f_{abs}}{x}\right|$

relativ Eingabefehler $f_{rel\&in} = \left|\frac{rd(x)-x}{x}\right|$ verstärkte Faktor $= \frac{f_{rel\&in}}{f_{rel\&out}}$

relativ Ausgabefehler $f_{rel\&out} = \left|\frac{rd(f)(x) - f(x)}{f(x)}\right| = \begin{cases} \infty & \text{instabil} \\ \text{sonst} & \text{stabil} \end{cases}$

Kondition: Fehlerverhältnis zwischen Eingabe u. Ausgabe

$cond(f,x) = \frac{x \cdot f'(x)}{f(x)}$ ≤ 1 gut konditioniert ≥ 1 schlecht konditioniert

Ein Algorithmus kann instabil trotz guter Kondition sein, und auch anderstrum, also stabil trotz schlechter Kondition

Interpolation (Polynom ist eindeutig bestimmt)

Basisfunktion $g_0(x) = 1$, $g_1(x) = x$, $g_2(x) = x^2$, $g_3(x) = x^3$ ...

$$\begin{bmatrix} g_0(x_0) & g_1(x_0) & \dots & g_{n-1}(x_0) \\ g_0(x_1) & g_1(x_1) & \dots & g_{n-1}(x_1) \\ \vdots & \vdots & \vdots & \vdots \\ g_0(x_{n-1}) & g_1(x_{n-1}) & \dots & g_{n-1}(x_{n-1}) \end{bmatrix} \cdot \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-2} \\ c_{n-1} \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-2} \\ y_{n-1} \end{pmatrix}$$

$p(x) = c_0 \cdot g_0(x) + c_1 \cdot g_1(x) + \dots + c_{n-1} \cdot g_{n-1}(x)$

Lagrange-Basis $p(x) = \sum_{j=0}^{n-1} y_j \cdot L_j(x)$ mit $L_j(x) = \prod_{i=0, i\neq j}^{n-1} \frac{x - x_i}{x_j - x_i}$

Interpolation nach Newton $c_{i,k} = \frac{c_{i+1,k-1} - c_{i,k-1}}{x_{i+k} - x_i}$

x_i	$i \backslash k$	0	1	2	...
x_0	0	$c_{0,0} = f(x_0) = y_0$	→ $c_{0,1}$	→ $c_{0,2}$	→ ...
x_1	1	$c_{1,0} = f(x_1) = y_1$	→ $c_{1,1}$	→ ⋮	$c_{0,n} \neq 0$ ⇒ Polynom update
x_2	2	$c_{2,0} = f(x_2) = y_2$	→ ⋮		da n-te Stützpunkt nicht auf
⋮	⋮	⋮			ursprüngliche Polynom liegt

$p(x) = c_{0,0} + c_{0,1} \cdot (x - x_0) + \dots + c_{0,n} \cdot \prod_{i=0}^{n-1} (x - x_i)$

Interpolation nach Aitken-Neville (für wenige Punkte direkt auswerten)

x_i	$i \backslash k$	0	1	2	...
x_0	0	$p[0,0] = y_0$	→ $p[0,1]$	→ $p[0,2]$	(...)
x_1	1	$p[1,0] = y_1$	→ $p[1,1]$	→ ⋮	Wert von p(x) an Stelle x
x_2	2	$p[2,0] = y_2$	→ ⋮		steht nach Algo-Abschluss in p[0,n]
⋮	⋮	⋮			(ganz rechts oben)

$p[i,k] = p[i,k-1] + \frac{x - x_i}{x_{i+k} - x_i} \cdot \left(p[i+1,k-1] - p[i,k-1]\right)$

Runge-Effekt

Die Ränder von Interpolationspolynom mit vielen äquidistante Stützstelle werden extrem ungenau. Unser Interpolationspolynom hat aufgrund der hohen Anzahl an Stützstellen einen hohen Grad und somit kommt es zu Schwingungen an den Rändern des Interpolationsbereiches

Interpolation-Fehlerabschätzung (an Stelle x)

$\left|f(x) - p(x)\right| = \left|\frac{f^{(n)}(\xi)}{n!} \cdot \prod_{k=0}^{n-1} (x - x_k)\right|$ mit $\xi \in \{x_0, \dots, x_n\}$, sodass Fehler maximal wird $n = \#$Stützstellen $f^{(n)} = n$-te Ableitung von f

Hermite-Interpolation

eine Funktion $p(x)$ zu erhalten, die überall stetig differenzierbar ist ($p \in C^1$ = keine Knicke)

1) mit Standard Polynom (3. Grad) $p(t) = c_3t^3 + c_2t^2 + c_1t + c_0$ und dessen Ableitung $p'(t) = 3c_3t^2 + 2c_2t + c_1$ ⇒ LGS aufstellen und lösen

$p(t=0) = c_0 = y_0$

$p(t=1) = c_3 + c_2 + c_1 + c_0 = y_1$

$p'(t=0) = c_1 = y_0'$

$p'(t=1) = 3c_3 + 2c_2 + c_1 = y_1'$

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 3 & 2 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} c_3 \\ c_2 \\ c_1 \\ c_0 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_1' \\ y_0' \\ y_0 \end{pmatrix} \Rightarrow \begin{matrix} c_3 = \dots \\ c_2 = \dots \\ c_1 = \dots \\ c_0 = \dots \end{matrix}$$

2) dann Werte von c_3, c_2, c_1, c_0 in Polynom einsetzen und sortiert nach y_0, y_1, y_0', y_1' in Form $p(t) = y_0 \cdot H_0(t) + y_1 \cdot H_1(t) + y_0' \cdot H_2(t) + y_1' \cdot H_3(t)$ sodass Basispolynom Hi durch Koeffizientvergleich bestimmt wird für $x_0 = 0$ $x_1 = 1$

3) Um in $[x_i, x_{i+1}]$ die Basispolynom $H_0(t)$ bis $H_3(t)$ $t \in [0,1]$ nutzen zu können, müssen wir unsere Variable x vorher noch mit $t_i(x)$ auf $[0;1]$ transformieren. Das erreichen wir mit $t_i(x) = \frac{x - x_i}{x_{i+1} - x_i} = \frac{x - x_i}{h} \in [0;1]$

$$\begin{pmatrix} v_0 \\ v_1 \\ v_2 \\ \vdots \\ v_{n-1} \end{pmatrix} = V = M_{IDFT,n} \cdot C = \begin{pmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & \omega^{1\cdot1} & \omega^{2\cdot1} & \omega^{3\cdot1} & \dots & \omega^{(n-1)\cdot1} \\ 1 & \omega^{1\cdot2} & \omega^{2\cdot2} & \omega^{3\cdot2} & \dots & \omega^{(n-1)\cdot2} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{1\cdot(n-1)} & \omega^{2\cdot(n-1)} & \omega^{3\cdot(n-1)} & \dots & \omega^{(n-1)\cdot(n-1)} \end{pmatrix} \cdot \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{n-1} \end{pmatrix}$$

4) um die transformierte Funktion abzuleiten, müssen wir Kettenregel nutzen

$\frac{d}{dx}(f(t_i(x))) = \frac{d}{dt}(f(t_i(x))) \cdot \frac{dt}{dx} = \frac{d}{dt}(f(t_i(x))) \cdot \frac{1}{x_{i+1} - x_i} = \frac{d}{dt}(f(t_i(x))) \cdot \frac{1}{h_i}$

5) um nun das kubische Teilpolynom $p_i(t(x))$ festzulegen, müssen wir Faktor h_i dazunehmen. somit erhalten wir für Teilintervall $[x_i, x_{i+1}]$:

$p_i(t_i(x)) = y_i \cdot H_0(t_i(x)) + y_{i+1} \cdot H_1(t_i(x)) + h_i \cdot y_i' \cdot H_2(t_i(x)) + h_i \cdot y_{i+1}' \cdot H_3(t_i(x))$

Interpolation mit Splines

man möchte eine interpolierende Funktion $s(x)$ erhalten, die ähnlich wie bei Hermite-Interpolation aus kubischen Teilpolynom $p_i(x)$ auf Teilintervalle $[x_i, x_{i+1}]$ besteht. Allerdings soll $s(x)$ überall zweimal differenzierbar sein ($s \in C^2$) und dafür keine anderen Informationen benötigen als die n+1 zu interpolierenden Stützpunkte $(x_0, y_0), \dots, (x_n, y_n)$

$$\begin{cases} s(x_i) = y_i \text{ für } i = 0, \dots, n \\ s'(x_{i+1}) = y'_{i+1} \text{ für } i = 0, \dots, n-2 \\ s''(x_{i+1}) = y''_{i+1} \text{ für } i = 0, \dots, n-2 \end{cases}$$

mit $x_{i+1} - x_i = h = \frac{x_n - x_0}{n}$, $i = 0, 1, \dots, n-1$ löse das System:

$$\begin{bmatrix} 4 & 1 & & & \\ 1 & 4 & \ddots & & \\ & \ddots & \ddots & 1 & \\ & & 1 & 4 \end{bmatrix} \cdot \begin{pmatrix} y_1' \\ y_2' \\ \vdots \\ y_{n-2}' \\ y_{n-1}' \end{pmatrix} = \frac{3}{h} \begin{pmatrix} y_2 - y_0 \\ y_3 - y_1 \\ \vdots \\ y_{n-1} - y_{n-3} \\ y_n - y_{n-2} \end{pmatrix} - \begin{pmatrix} y_0' \\ 0 \\ \vdots \\ 0 \\ y_n' \end{pmatrix} = \frac{3}{h} \begin{pmatrix} y_2 - y_0 - \frac{h}{3} y_0' \\ y_3 - y_1 \\ \vdots \\ y_{n-1} - y_{n-3} \\ y_n - y_{n-2} - \frac{h}{3} y_n' \end{pmatrix}$$

Bestimmen mithilfe dieser Gleichung die Ableitungen aller Stützwerte außer den an den Rändern. Dann hat man alle Informationen, um für jedes Paar benachbarter Stützstellen die entsprechenden Hermite-Funktion aufzustellen

i	0	1	2	3
x_i	-1	0	1	2
$s(x_i) = y_i$	2	0	2	3
$s(x_i) = y_i'$	9	-3	3	0

$$\begin{pmatrix} 4 & 1 \\ 1 & 4 \end{pmatrix} \begin{pmatrix} y_1' \\ y_2' \end{pmatrix} = \frac{3}{h} \begin{pmatrix} y_2 - y_0 - \frac{h}{3} y_0' \\ y_3 - y_1 - \frac{h}{3} y_3' \end{pmatrix} = \begin{pmatrix} -9 \\ 9 \end{pmatrix} \Rightarrow \boxed{\begin{pmatrix} y_1' \\ y_2' \end{pmatrix} = \begin{pmatrix} -3 \\ 3 \end{pmatrix}}$$

$$s(x) = \begin{cases} p_0(t_0(x)) \text{ für } x \in [-1,0[\text{ mit } t_0(x) = \frac{x-(-1)}{1} = x+1 \\ p_1(t_1(x)) \text{ für } x \in [0,1[\text{ mit } t_1(x) = \frac{x-0}{1} = x \\ p_2(t_2(x)) \text{ für } x \in [1,2[\text{ mit } t_2(x) = \frac{x-1}{1} = x-1 \end{cases}$$

mit $\begin{cases} p_0(t) = 2 \cdot H_0(t) + 0 \cdot H_1(t) + 9 \cdot H_2(t) - 3 \cdot H_3(t) \\ p_1(t) = 0 \cdot H_0(t) + 2 \cdot H_1(t) - 3 \cdot H_2(t) + 3 \cdot H_3(t) \\ p_2(t) = 2 \cdot H_0(t) + 3 \cdot H_1(t) + 3 \cdot H_2(t) + 0 \cdot H_3(t) \end{cases}$

Komplexe Zahlen

$z = x + iy$ $\bar{z} = x - iy$ $Re(z) = x$ $Im(z) = y$ $i^2 = -1$ $\overline{z\omega} = \bar{z} \cdot \bar{\omega}$

$|z| = \sqrt{Re(z)^2 + Im(z)^2} = \sqrt{x^2 + y^2} = \sqrt{z \cdot \bar{z}}$ $exp(-z) = \frac{1}{exp(z)}$

$e^{it} = \cos(t) + i\sin(t)$ durchläuft den Einheitskreis gegen Uhrzeigerrichtung (beginnend bei $e^0 = (1,0)$). außerdem 2π-periodisch $e^{i\cdot 0} = e^{i \cdot 2k\pi} = 1$ $e^{-i\pi} = -1$

$\varphi = \begin{cases} \arccos(x/r) & y \geq 0 \\ -\arccos(x/r) & y < 0 \end{cases}$ $z^n = r^n(\cos(n\varphi) + i\sin(n\varphi))$

α	0	30	45	60	90	120	150	180	270	360
	0	$\frac{\pi}{6}$	$\frac{\pi}{4}$	$\frac{\pi}{3}$	$\frac{\pi}{2}$	$\frac{2\pi}{3}$	$\frac{5\pi}{6}$	π	$\frac{3\pi}{2}$	2π
$\sin(\alpha)$	0	$\frac{1}{2}$	$\frac{\sqrt{2}}{2}$	$\frac{\sqrt{3}}{2}$	1	$\frac{\sqrt{3}}{2}$	$\frac{1}{2}$	0	-1	0
$\cos(\alpha)$	1	$\frac{\sqrt{3}}{2}$	$\frac{\sqrt{2}}{2}$	$\frac{1}{2}$	0	$-\frac{1}{2}$	$-\frac{\sqrt{3}}{2}$	-1	0	1
$\tan(\alpha)$	0	$\frac{\sqrt{3}}{3}$	1	$\sqrt{3}$	∞	$-\sqrt{3}$	$-\frac{\sqrt{3}}{3}$	0	∞	0

$\sin(-x) = -\sin(x)$

$\cos(-x) = \cos(x)$

	n=3
$\omega^4 = \bar{\omega}^2 = \omega$	$-\frac{1}{2} + \frac{\sqrt{3}}{2}i$
$\bar{\omega}^4 = \omega^2 = \bar{\omega}$	$-\frac{1}{2} - \frac{\sqrt{3}}{2}i$
$\omega + \bar{\omega}$	-1
$\omega\bar{\omega}$	0

Frequenzanalyse

Diskrete Fourier-Transformation (DFT): mit n komplexe Eingabedaten $V = (v_0 \dots v_{n-1})^T$ und $\omega = exp(1 \cdot \frac{2\pi}{n}) = e^{i\frac{2\pi}{n}}$ gilt $c_k = DFT(v)_k = \frac{1}{n} \sum_{j=0}^{n-1} v_j \cdot \bar{\omega}^{jk}$

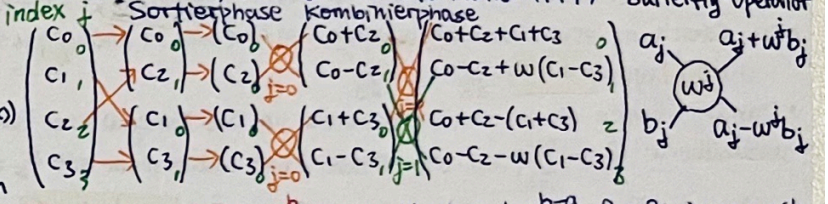
$$\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{n-1} \end{pmatrix} = C = M_{DFT,n} \cdot V = \frac{1}{n} \begin{pmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & \bar{\omega}^{1\cdot1} & \bar{\omega}^{2\cdot1} & \bar{\omega}^{3\cdot1} & \dots & \bar{\omega}^{(n-1)\cdot1} \\ 1 & \bar{\omega}^{1\cdot2} & \bar{\omega}^{2\cdot2} & \bar{\omega}^{3\cdot2} & \dots & \bar{\omega}^{(n-1)\cdot2} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \bar{\omega}^{1\cdot(n-1)} & \bar{\omega}^{2\cdot(n-1)} & \bar{\omega}^{3\cdot(n-1)} & \dots & \bar{\omega}^{(n-1)\cdot(n-1)} \end{pmatrix} \cdot \begin{pmatrix} v_0 \\ v_1 \\ v_2 \\ \vdots \\ v_{n-1} \end{pmatrix}$$

Inverse diskrete Fourier-Transformation (IDFT)

$v_j = IDFT(c)_k = \sum_{k=0}^{n-1} c_k \cdot \omega^{jk}$

Schnelle (Inverse) Fourier-Transformation (FFT)

Butterfly-Operator



Quadratur

$I(f) = \int_a^b f(x)\,dx$ $H = b-a$ $h = \frac{b-a}{n}$ $f_i = f(x_i)$ $x = a + ih$

Allgemeines Quadraturproblem $\int_a^b f(x)\,dx \approx \sum_{i=1}^{n} f(x_i) \cdot \omega_i$

mit Lagrange $f(x) \approx p(x) = \sum_{i=1}^{n} f(x_i) \cdot L_i(x)$

$\rightarrow \int_a^b f(x)\,dx \approx \int_a^b p(x)\,dx = \int_a^b \sum_{i=1}^{n} f(x_i) \cdot L_i(x)\,dx$

Trapezregel (für 2 Stützpunkte) $Q_T(f) = H \cdot \frac{1}{2}(f(a) + f(b))$

Trapezsumme $Q_{TS}(f;h) = h \cdot \left(\frac{f_0}{2} + f_1 + \dots + f_{N-1} + \frac{f_N}{2}\right)$

$Q_{TS}(f;h) = \frac{Q_{TS}(f;2h)}{2} + h(f_1 + f_3 + \dots + f_{N-3} + f_{N-1})$

Simpsonregel bzw. Keplersche Fassregel (für 3 Stützpunkte) $Q_S(f) = Q_F(f) = H \cdot \frac{1}{6}\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right)$

Simpsonsumme $Q_{SS}(f;h) = \frac{h}{3} \cdot (f_0 + 4f_1 + 2f_2 + 4f_3 + \dots + 2f_{n-2} + 4f_{n-1} + f_n)$

Restglied = Fehlerabschätzung, also im Grunde $R_{TS} \approx Q_{TS} - I(f)$

$R_{TS}(f;h) = h^2 \cdot H \cdot \frac{f^{(2)}(\xi)}{12}$ $R_{SS}(f;h) = h^4 \cdot H \cdot \frac{f^{(4)}(\xi)}{180}$ $\xi \in [a,b]$

ξ kommt aus dem Mittelwertsatz und wird so gewählt, dass Fehler maximal

lokale Diskretisierungsfehler: Fehler zwischen analytischer u. nummerischer approximierte Lösung eines ODEs nach 1 Zeitschritt $|y_{k+1} - y(t_{k+1})|$ mit $y_k = y(t_k)$

globale Diskretisierungsfehler: größte Fehler zwischen analytischer Lösung und nummerische Approximation für alle Zeitschritte $|y_n - y(t_N)|$ wobei $I = [0, N]$

Quadratur nach Romberg: man kombiniert zwei Trapezsummen unterschiedlicher Schrittweiten $Q_{TS}(f;h_1)$, $Q_{TS}(f;h_2)$ um das exakte Ergebnis $I(f)=\lim_{h\to 0} Q_{TS}(f,h)$ besser zu approximieren

Kombinationsregel $Q_{i,k}=Q_{i,k-1}+\frac{Q_{i,k-1}-Q_{i-1,k-1}}{\frac{h_{i-k}^2}{h_i^2}-1}$, $1\le i\le k$

h_i	$i\backslash k$	0	1		n
$\frac{b-a}{1}$	0	$Q_{TS}(f;\frac{b-a}{1})=Q_{00}$			
$\frac{b-a}{2}$	1	$Q_{TS}(f;\frac{b-a}{2})=Q_{10}$	Q_{11}		
$\frac{b-a}{4}$	2	$Q_{TS}(f;\frac{b-a}{4})=Q_{20}$	Q_{21}	…	
…	…	…	…	…	
$\frac{b-a}{2^n}$	n	$Q_{TS}(f;\frac{b-a}{2^n})=Q_{n0}$	Q_{n1}	…	Q_{nn}

falls 3 $Q_{i,k}$ die gleiche Werte haben, dann Integral bereits exakt. fertig

Gauß Quadratur: die Summe von Funktionswerten mal Gewichtungen soll gleich dem Integralwert der Funktion entsprechen

$\sum_{i=0}^{n-1} f_k(x_i)\cdot w_i=\int_a^b f_k(x)\,dx$ mit Polynombasis $f_k(x)=x^k$ für Grad k

- Die Gleichung für alle Grade k aufstellen ($k\in\{0,\dots,2n-1\}$) mit n Stützstellen

Beispiel $b=1$, $a=-1$

Grad 0 mit $f(x)=x^0=1$ $f(x_0)\cdot w_0+f(x_1)\cdot w_1=w_0+w_1=\int_{-1}^{1}1\,dx=[x]_{-1}^{1}=2$

Grad 1 mit $f(x)=x^1=x$ $f(x_0)\cdot w_0+f(x_1)\cdot w_1=w_0x_0+w_1x_1=\int_{-1}^{1}x=[\frac{1}{2}x^2]_{-1}^{1}=0$

Grad 2 mit $f(x)=x^2$ $f(x_0)\cdot w_0+f(x_1)\cdot w_1=w_0x_0^2+w_1x_1^2=\int_{-1}^{1}x^2=[\frac{1}{3}x^3]_{-1}^{1}=\frac{2}{3}$

Grad 3 mit $f(x)=x^3$ $f(x_0)\cdot w_0+f(x_1)\cdot w_1=w_0x_0^3+w_1x_1^3=\int_{-1}^{1}x^3=[\frac{1}{4}x^4]_{-1}^{1}=0$

- Gleichung umstellen/zusammensetzen, um auf alle x_i und w_i zu schließen

$w_0=1$ $w_1=1$ $x_0=-\frac{1}{\sqrt{3}}$ $x_1=\frac{1}{\sqrt{3}}$

- Danach können wir x_i und w_i benutzen um das Integral einer Funktion f anzunähern: $Q_G=\sum_{i=0}^{n-1} f(x_i)\cdot w_i$ Diese f hat nichts mehr mit f_k zu tun

$Q_G(x^3-x^2)=1\cdot(-\frac{1}{3\sqrt{3}}-\frac{1}{3})+1\cdot(\frac{1}{3\sqrt{3}}-\frac{1}{3})=-\frac{2}{3}$

lineare Gleichungssystem

lineare Ausgleichsproblem: ein überbestimmtes LGS $Ax=b$ mit $A\in\mathbb{R}^{m\times n}$ versucht man über die Minimierung des Residuums $\min_z|Az-b|_2$ zu lösen. Die Lösung entspricht der Lösung der Normalengleichung $A^TAx=A^Tb$

- gegeben sind die Punkte p_1 bis p_n mit den Koordinaten (x_1,y_1) bis (x_n,y_n) Ergebnisgerade entspricht $y=mx+b$
- $A\cdot\begin{pmatrix}t\\m\end{pmatrix}=b$ mit $A=\begin{pmatrix}1&x_0\\\vdots&\vdots\\1&x_n\end{pmatrix}$ und $b=\begin{pmatrix}y_0\\\vdots\\y_n\end{pmatrix}$ aufstellen
- Normalengleichung $A^TAx=A^Tb$ mit $x=\begin{pmatrix}t\\m\end{pmatrix}$ aufstellen und lösen

Pivotelement: das Element ganz links oben in den noch übrig gebliebenen Matrix-Teil

Spalten-Pivotsuche: Zeilen der Matrix vertauschen, dass betragsmäßig größte Element zum Pivot wird

LR-Zerlegung Verfahren: zum Lösen eines LGS der Form $Ax=b$

1) LR-Zerlegung der Matrix: $A=LR=\begin{pmatrix}1&0&0\\\vdots&\ddots&0\\1&\dots&1\end{pmatrix}\cdot\begin{pmatrix}\cdot&\dots&\cdot\\0&\ddots&\vdots\\0&0&\cdot\end{pmatrix}$

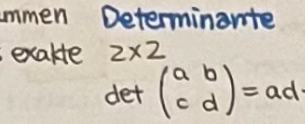
2) Vorwärtssubstitution: $Ly=b$ $\Rightarrow y=(y_1\ y_2\ y_3)^T$

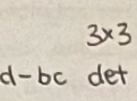
3) Rückwärtssubstitution: $Rx=y$ $\Rightarrow x=(x_1\ x_2\ x_3)^T$

Spektralnorm einer Matrix A entspricht $\|A\|_2=\max_{x\neq 0}\frac{\|Ax\|_2}{\|x\|_2}$ mit Vektornorm $\|x\|_2=\sqrt{x_1^2+\dots+x_n^2}=\sqrt{x^Tx}$

Determinante

2×2 $\det\begin{pmatrix}a&b\\c&d\end{pmatrix}=ad-bc$

3×3 

4×4, n×n 

det(Diagonalmatrix) = Produkt der Diagonalwerte

$\det(\text{Blockmatrix})=\det\begin{pmatrix}A&C\\0&B\end{pmatrix}=\det\begin{pmatrix}A&0\\C&B\end{pmatrix}=\det(A)\cdot\det(B)$

$\det(A)=\det(A^T)$ $\det(AB)=\det(A)\cdot\det(B)$ $\det(A^{-1})=\det(A)^{-1}$

Orthogonale Matrix Matrix ist orthogonal wenn $A^TA=E_n$

Differentialgleichungen

Gewöhnliche Differentialgleichung (ODE): eine Gleichung, die eine Funktion durch ihre Ableitung beschreibt $y'(t)=f(t,y)$

→ Separation der Variablen gegeben $y'(t)=f(t,y)$ bsp $y'(t)=z\cdot y(t)$

1) in Leibniz-Notation schreiben $\frac{dy}{dt}=zy$

2) Separation der Variablen alle y auf eine Seite $\frac{dy}{zy}=dt$

3) Integration $\int_{y_0}^{y}\frac{1}{zy}dy=\int_{t_0}^{t}1\,dt$ $\Rightarrow\frac{1}{z}[\ln|y|]_{y_0}^{y}=[t]_{t_0}^{t}$

$\Rightarrow\ln(y)-\ln(y_0)=z(t-t_0)$ $\Rightarrow e^{\ln(y)-\ln(y_0)}=\frac{e^{\ln(y)}}{e^{\ln(y_0)}}=\frac{y(t)}{y_0}=e^{z(t-t_0)}$

4) nach y auflösen $y(t)=\pm y_0\cdot e^{z(t-t_0)}$

→ verkürzte Version: gegeben $y'(t)=f(t)\cdot g(y(t))$

1) man stellt folgende Gleichung auf $\int_{y_0}^{y}\frac{1}{g(y(\eta))}d\eta=\int_{t_0}^{t}f(\tau)\,d\tau$

2) man löst die entstandene Gleichung nach $y(t)$ auf und dann fertig

Anfangswertproblem (AWP): Anfangsbedingung $y_0=y(t_0)$ und t_0 in die allgemeine Lösung von ODE einsetzen $y(t)=+y_0\cdot e^{zt}$ mit $t_0=0$

→ Kondition eines AWPs: y_0 durch $y_\varepsilon=y_0+\varepsilon$ in $y(t)$ ersetzen und umstellen $\Rightarrow y_\varepsilon(t)=y(t)+\text{Fehlerterl}$

$y_\varepsilon(0)=y_0+\varepsilon$ $y_\varepsilon(t)=(y_0+\varepsilon)\cdot e^{zt}=y_0\cdot e^{zt}+\varepsilon\cdot e^{zt}=y(t)+\varepsilon\cdot e^{zt}$

$|y_\varepsilon(t)-y(t)|=|\varepsilon|\cdot e^{zt}\to\infty$ für $t\to\infty$ $\Rightarrow$ schlecht konditioniertes Problem

(Explizite) Einschrittverfahren: Einschrittsverfahren bieten die Möglichkeit, ODE schrittweise anzunähern. Im Folgenden sei st immer die Schrittweite, $f(t_k,y_k)=y'(t_k)=t_k\cdot y_k(t_k)$

$t_k=t_0+k\cdot\delta t$

→ explizite Euler-Verfahren (1. Ordnung)

$y_{k+1}=y_k+\delta t\cdot f(t_k,y_k)$

→ Heun-Verfahren (2. Ordnung)

$y_{k+1}=y_k+\frac{\delta t}{2}\cdot(f(t_k,y_k)+f(t_{k+1},y_k+\delta t\cdot f(t_k,y_k)))$

→ klassisches Runge-Kutta-Verfahren (4. Ordnung)

$T_1=f(t_k,y_k)$

$T_2=f(t_k+\frac{\delta t}{2},y_k+\frac{\delta t}{2}T_1)$

$T_3=f(t_k+\frac{\delta t}{2},y_k+\frac{\delta t}{2}T_2)$

$T_4=f(t_{k+1},y_k+\delta tT_3)$

$y_{k+1}=y_k+\frac{\delta t}{6}(T_1+2T_2+2T_3+T_4)$

besser approximiert

Implizite Euler-Verfahren

$y_{k+1}=y_k+\delta t\cdot f(t_{k+1},y_{k+1})$

(gleiche Genauigkeit wie explizit Euler)

Mehrschrittverfahren

→ mittelpunktsregel $y_{k+1}=y_{k-1}+2\delta t\cdot f(t_k,y_k)$ $k=1,\dots,N-1$

y_0 gegeben, y_1 durch explizit Einschrittverfahren bestimmt

Quadratur und AWP-Lösung

$y(t_{k+1})-y(t_k)=\int_{t_k}^{t_{k+1}}y'(t)\,dt=\int_{t_k}^{t_{k+1}}f(t,y(t))\,dt$

dumme Rechteckesregel $\int_a^b f(t)\,dt\approx(b-a)\cdot f(a)$

Iterative Verfahren für LGS

Fixpunkt: Punkt x einer Funktion f, sodass $f(x)=x$ gilt

Fixpunktverfahren versuchen (wie eben Newton) mit einem Fixpunkt x^*, der Iterationsvorschrift $\phi(x)$ und einem Startwert x_0 mit $x^*=\phi(x^*)$ Gleichungen der folgenden Form zu lösen $x_{i+1}=\phi(x_i)$

Vektoriteration $\phi(x_k)=x_k+M^{-1}(b-Ax)=x_{k+1}$

Jacobi Verfahren $M=\text{diag}(A)$

k-te Jacobi-Iteration: $x^{(k+1)}=x^{(k)}+\text{diag}(A)^{-1}(b-Ax^{(k)})$

Wir teilen das in folgenden Rechenschritte:

1 $r^{(k)}=b-Ax^{(k)}$

2 $y_i^{(k)}=\frac{1}{a_{ii}}r_i^{(k)}$ $\forall i\in[1,\dots,n]$ $A\in\mathbb{R}^{n\times n}$

3 $x^{(k+1)}=x^{(k)}+y^{(k)}=x^{(k)}+(y_1^{(k)}\dots y_n^{(k)})^T$

Gerschgorin-Kreis

$|z-a_{jj}|\le\sum_{k=1,k\neq j}^{n}|a_{jk}|$

$\begin{pmatrix}7&2&1&1\\2&5&0&0\\1&0&4&0\\1&0&0&3\end{pmatrix}$ $|\lambda-7|\le 4$ [3,11]; $|\lambda-5|\le 2$ [3,7]; $|\lambda-4|\le 1$ [3,5]; $|\lambda-3|\le 1$ [2,4]

$\lambda\in[3,11]\cup[3,7]\cup[3,5]\cup[2,4]=[2,11]$

Gauß-Seidel Verfahren

k-te Gauß-Seidel Iteration (für alle i von 1 bis n)

1 $r_i^{(k)}=b_i-\sum_{m=1}^{i-1}a_{i,m}x_m^{(k+1)}-\sum_{m=i}^{n}a_{i,m}x_m^{(k)}$

2 $y_i^{(k)}=\frac{1}{a_{ii}}\cdot r^{(k)}$

3 $x_i^{(k+1)}=x_i^{(k)}+y_i^{(k)}$

Verfahren des steilsten Abstiegs (weitere Alternative, um iterativ ein LGS lösen zu können. Dazu muss A symmetrisch u. positiv definit sein (alle Eigenwerte > 0, also $x^TAx>0$ sein)

Grundidee: Funktion $f(x)=x^TAx-b^Tx+c$ definieren, dann ableiten, erhalten wir $\nabla f(x)=Ax-b$

In jeder Iteration berechnet man folgende Schritte:

1 aktuelles Residuum (Schrittrichtung): $r^{(k)}=b-Ax^{(k)}$

2 optimale Schrittweite: $a^{(k)}=\frac{r^{(k)T}\cdot r^{(k)}}{r^{(k)T}\cdot Ar^{(k)}}$

3 aktuell Zwischenergebnis: $x^{(k+1)}=x^{(k)}+a^{(k)}r^{(k)}$

Newton Verfahren kann die Nullstellen von $f(x)$ approximieren und bewerkstelligt dies mithilfe von Tangente $x_{k+1}=x_k-\frac{f(x_k)}{f'(x_k)}$

Eigenwertproblem $Av=\lambda v$ wobei v Eigenvektor, λ Eigenwert

1 Löse Gleichung $\det(A-\lambda_iE_n)=0$, um auf alle Eigenwerte λ_i zu schließen

2 für jeden Eigenwert die Gleichung $(A-\lambda_iE_n)v_i=0$ lösen, um alle Eigenvektoren v_i zu bestimmen **Rayleigh Quotient** $\lambda=\frac{v^TAv}{v^Tv}$

Power Iteration Eigenwert/Eigenvektor annähern

direkte Power Iteration

1 $w_k=A\cdot x_k$

2 $\lambda_k=\frac{x_k^TAx_k}{x_k^Tx_k}=x_k^Tw_k$ (da $x_k^Tx_k=1$)

3 $x_{k+1}=\frac{w_k}{\|w_k\|_2}$

Konvergenzrate $p=\frac{|\lambda_{\text{zweitgrößte}}|}{|\lambda_{\text{größte}}|}$

Shifted Power Iteration (shift um μ)

1 $w_k=(A-\mu I)x_k$

2 $\tilde{\lambda}_k=\frac{x_k^T(A-\mu I)x_k}{x_k^Tx_k}=x_k^Tw_k$

3 $x_{k+1}=\frac{w_k}{\|w_k\|_2}$ 4 $\lambda_k=\tilde{\lambda}_k+\mu$

Konvergenzrate $p=\frac{|\lambda-\mu|_{\text{zweitgrößte}}}{|\lambda-\mu|_{\text{größte}}}$

Konvergenz: kleine Schrittweite δt → kleiner globale Diskretisierungsfehler $\lim_{h\to 0,n\to\infty}y_k=y(t_k)$

Konsistenz: kleine Schrittweite δt → kleine lokale Diskretisierungsfehler lokale Fehler $\ell(\delta t)\to 0$ wenn $\delta t\to 0$

Stabilität: die Summe kleiner lokaler Fehler ergibt einen kleinen globalen Fehler (kein Verstärken kleiner lokale Fehler) Euler, Heun, Runge Kutta stabil

Steifheit: Konvergenz, Konsistenz und Stabilität gelten nur für sehr kleine δt, so nennt man das dazugehörige ODE steif

Einschrittverfahren (ESV): Konvergenz ⇒ Konsistenz

Mehrschrittverfahren (MSV): Konsistenz + Stabilität ⇒ Konvergenz

Numerische Methoden der Elektrotechnik

1. Grundlagen

1.1. Numerik $f(x) = y \quad \Rightarrow \quad \tilde{f}(\tilde{x}) = \tilde{y}$

liefert eine zahlenmäßige Lösung eines Problems mit einem Algorithmus

f Mathematisches Problem	$\tilde{f}$ Numerischer Algorithmus
x exakte Problemdaten	$\tilde{x}$ gerundete Problemdaten
y exaktes Ergebnis	$\tilde{y}$ gerundetes Ergebnis

1.2. Fehlertypen

Datenfehler: Eingabedaten aus ungenauer Messung
Verfahrensfehler: Diskretisierung von Gleichungen, Endliche Iteration
Rundungsfehler: (Zwischen-)Ergebnisse nur mit Maschinengenauigkeit

1.3. Numerische Qualitätsmerkmale

Kondition: Wie stark schwankt das Problem bei Störung $f(\tilde{x}) \to y$?
Konsistenz: Wie gut löst das Verfahren tatsäch. das Problem $\tilde{f}(x) \to y$?
Residuum $R < C \cdot h^p$ Schrittweite h, Konsistenzordnung p
Stabilität: Wie stark schwankt das Verfahren bei Störung $\tilde{f}(\tilde{x}) \to f(x)$
Konvergenz: Algorithmus stabil und konsistent: $\tilde{f}(\tilde{x}) \to f(x) \to y$

1.4. Spektralradius

Spektralradius $\rho(\underset{\sim}{A})$ einer Matrix $\underset{\sim}{A}$: Betragsmäßig größter Eigenwert.
Konvergenzbeweis aller Verfahren: Gershgorinkreise um die Null mit $r \leq 1$

$$\rho(\underset{\sim}{A}) = \max_i |\lambda_i|$$

1.5. Diagonaldominanz

Diagonalelemente sind größer als die restlichen Elemente der selben Zeile:

$$\underset{\sim}{A} = (a_{ij}) \quad \begin{matrix}\text{diagonaldominant} \\ \text{strikt diagonaldominant}\end{matrix} \Leftrightarrow |a_{ii}| \begin{matrix}\geq \\ > \end{matrix} \sum_{j=1, j\neq i}^{n} |a_{ij}| \quad \forall i$$

1.6. Definitheit

$$\underset{\sim}{A} \quad \begin{matrix}\text{positiv definit} \\ \text{positiv semidefinit}\end{matrix} \Leftrightarrow \lambda_i \begin{matrix}> \\ \geq \end{matrix} 0 \ \forall i \qquad \text{bzw.} \qquad \vec{x}^T \underset{\sim}{A} \vec{x} \begin{matrix}> \\ \geq \end{matrix} 0 \ \forall \vec{x} \neq 0$$

1.7. Kondition

Ein Maß wie stark sich Eingabefehler auf die Ausgabe auswirken.

$$\kappa_{\text{abs}}(x) = |f'(x)| \qquad \kappa_{\text{rel}}(x) = \left|\frac{f'(x)}{\frac{f(x)}{x}}\right| = \frac{|f'(x)| \cdot |x|}{|f(x)|}$$

Falls $\kappa_{\text{rel}} \ll 100$: gute Konditionierung
Verkettung $h = g(f(x)) \quad \kappa^h_{\text{abs}}(x) = \kappa^g_{\text{abs}}(f(x)) \kappa^f_{\text{abs}}(x)$

$$\text{für } \vec{y} = \underset{\sim}{A}\vec{x} \quad \Rightarrow \quad \text{cond}(\underset{\sim}{A}) = \left\|\underset{\sim}{A}^{-1}\right\| \cdot \left\|\underset{\sim}{A}\right\|$$

$\text{cond}(\underset{\sim}{A}) \to \infty$ schlecht, $\text{cond}(\underset{\sim}{A}) \to 1$ gut

1.8. Fehler

Absolut: $\left\|\tilde{f}(x) - f(x)\right\|$ Relativ: $\frac{\left\|f(x) - \tilde{f}(x)\right\|}{\|f(x)\|}$

1.9. Residuum $\vec{r} = \vec{b} - \underset{\sim}{A}\vec{x}$

bezeichnet die Abweichung vom gewünschten Ergebnis, wenn Näherungslösungen eingesetzt werden. $\vec{r}$ klein $\Rightarrow$ rel. Fehler $\ll 1$.

1.10. Parametrisierung einer Geraden

$$g(x) = ax + b \qquad \begin{matrix} y_1 = g(x_1) \\ y_2 = g(x_2) \end{matrix} \qquad \begin{matrix} a = \frac{y_1 - y_2}{x_1 - x_2} \\ b = \frac{x_1 y_2 - x_2 y_1}{x_1 - x_2} \end{matrix}$$

1.11. Schnittpunkt zweier Geraden

$$\begin{matrix} a_{11}x_1 + a_{12}x_2 = b_1 \\ a_{21}x_1 + a_{22}x_2 = b_2 \end{matrix} \qquad \begin{matrix} x_1 = \frac{a_{22}b_1 - a_{12}b_2}{a_{11}a_{22} - a_{12}a_{21}} \\ x_2 = \frac{-a_{21}b_1 + a_{11}b_2}{a_{11}a_{22} - a_{12}a_{21}} \end{matrix}$$

1.12. Matrizen $\underset{\sim}{A} \in \mathbb{K}^{m \times n}$

$$(\underset{\sim}{A} + \underset{\sim}{B})^\top = \underset{\sim}{A}^\top + \underset{\sim}{B}^\top \qquad (\underset{\sim}{A} \cdot \underset{\sim}{B})^\top = \underset{\sim}{B}^\top \cdot \underset{\sim}{A}^\top$$
$$(\underset{\sim}{A}^\top)^{-1} = (\underset{\sim}{A}^{-1})^\top \qquad (\underset{\sim}{A} \cdot \underset{\sim}{B})^{-1} = \underset{\sim}{B}^{-1}\underset{\sim}{A}^{-1}$$

1.12.1 Dimensionen

Bildraum	Nullraum
$\text{Bild}\ \underset{\sim}{A} = \{\underset{\sim}{A}\vec{x} \mid \vec{x} \in \mathbb{K}^n\}$	$\ker \underset{\sim}{A} = \{\vec{x} \in \mathbb{K}^n \mid \underset{\sim}{A}\vec{x} = \vec{0}\}$
$\text{rang}\ \underset{\sim}{A} = \dim(\text{Bild}\ \underset{\sim}{A})$	$\text{def}\ \underset{\sim}{A} = \dim(\ker \underset{\sim}{A})$

$\text{rang}\ \underset{\sim}{A} = r$ ist Anzahl. lin. unab. Spaltenvektoren.
$\underset{\sim}{A}$ erzeugt $\mathbb{K} \Leftrightarrow r = n \qquad \underset{\sim}{A}$ ist Basis von $\mathbb{K} \Leftrightarrow r = n = m$
$\dim \mathbb{K} = n = \text{rang}\ \underset{\sim}{A} + \dim \ker \underset{\sim}{A} \qquad \text{rang}\ \underset{\sim}{A} = \text{rang}\ \underset{\sim}{A}^\top$

1.12.2 Quadratische Matrizen $A \in \mathbb{K}^{n \times n}$

regulär/invertierbar/nicht-singulär $\Leftrightarrow \det(\underset{\sim}{A}) \neq 0 \Leftrightarrow \text{rang}\ \underset{\sim}{A} = n$
singulär/nicht-invertierbar $\Leftrightarrow \det(\underset{\sim}{A}) = 0 \Leftrightarrow \text{rang}\ \underset{\sim}{A} \neq n$
orthogonal $\Leftrightarrow \underset{\sim}{A}^\top = \underset{\sim}{A}^{-1} \Rightarrow \det(\underset{\sim}{A}) = \pm 1$
symmetrisch: $\underset{\sim}{A} = \underset{\sim}{A}^\top$ schiefsymmetrisch: $\underset{\sim}{A} = -\underset{\sim}{A}^\top$

1.12.3 Determinante von $\underset{\sim}{A} \in \mathbb{K}^{n \times n}$: $\det(\underset{\sim}{A}) = |\underset{\sim}{A}|$

$$\det \begin{bmatrix} \underset{\sim}{A} & \underset{\sim}{0} \\ \underset{\sim}{C} & \underset{\sim}{D} \end{bmatrix} = \det \begin{bmatrix} \underset{\sim}{A} & \underset{\sim}{B} \\ \underset{\sim}{0} & \underset{\sim}{D} \end{bmatrix} = \det(\underset{\sim}{A}) \det(\underset{\sim}{D})$$

$\det(\underset{\sim}{A}) = \det(\underset{\sim}{A}^T) \qquad \det(\underset{\sim}{A}^{-1}) = \det(\underset{\sim}{A})^{-1}$
$\det(\underset{\sim}{A}\underset{\sim}{B}) = \det(\underset{\sim}{A}) \det(\underset{\sim}{B}) = \det(\underset{\sim}{B}) \det(\underset{\sim}{A}) = \det(\underset{\sim}{B}\underset{\sim}{A})$
Hat $\underset{\sim}{A}$ 2 linear abhäng. Zeilen/Spalten $\Rightarrow |\underset{\sim}{A}| = 0$

Entwicklung. n. $j$ter Zeile: $|\underset{\sim}{A}| = \sum_{i=1}^{n} (-1)^{i+j} \cdot a_{ij} \cdot |\underset{\sim}{A}_{ij}|$

1.12.4 Eigenwerte λ und Eigenvektoren $\underset{\sim}{v}$

$$\underset{\sim}{A}\vec{v} = \lambda\vec{v} \qquad \det \underset{\sim}{A} = \prod \lambda_i \qquad \text{Sp}\ \underset{\sim}{A} = \sum a_{ii} = \sum \lambda_i$$

Eigenwerte: $\det(\underset{\sim}{A} - \lambda\underset{\sim}{1}) = 0$ Eigenvektoren: $\ker(\underset{\sim}{A} - \lambda_i\underset{\sim}{1}) = \vec{v}_i$
EW von Dreieck/Diagonal Matrizen sind die Elem. der Hauptdiagonale.

1.12.5 Spezialfall 2×2 Matrix A

$$\det(\underset{\sim}{A}) = ad - bc \qquad \begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{\det \underset{\sim}{A}} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$
$$\text{Sp}(\underset{\sim}{A}) = a + d$$
$$\lambda_{1/2} = \frac{\text{Sp}\ \underset{\sim}{A}}{2} \pm \sqrt{\left(\frac{\text{sp}\underset{\sim}{A}}{2}\right)^2 - \det \underset{\sim}{A}}$$

1.12.6 Spezielle Matrizen

Diagonalmatrix $\underset{\sim}{D}$: $\det \underset{\sim}{D} = \prod d_i$

$$\underset{\sim}{D}^{-1} = \text{diag}\,(d_1, \ldots, d_n)^{-1} = \text{diag}\left(d_1^{-1}, \ldots, d_n^{-1}\right)$$

1.13. Norm $||\cdot||$

Definition: Zahl, die die „Größe" eines Objekts $\mathcal{X}$ beschreibt.
Jede Norm muss folgende 3 Axiome erfüllen::

1. Definitheit: $\|\mathcal{X}\| \geq 0$ mit $\|\mathcal{X}\| = 0 \Leftrightarrow \mathcal{X} = 0$
2. absolute Homogenität: $\|\alpha \cdot \mathcal{X}\| = |\alpha| \cdot \|\mathcal{X}\|$ (α ist skalar)
3. Dreiecksungleichung: $\|\mathcal{X} + \mathcal{Y}\| \leq \|\mathcal{X}\| + \|\mathcal{Y}\|$

1.13.1 Vektornormen: ($\vec{x} \in \mathbb{K}^n, \sum$ von $i = 0$ bis n)

Summen	$\|\vec{x}\|_1 = \sum \|x_i\|$	Euklidische	$\|\vec{x}\|_2 = \sqrt{\sum \|x_i\|^2}$
Maximum	$\|\vec{x}\|_\infty = \max \|x_i\|$	Alg. p-Norm	$\|\vec{x}\|_p = (\sum \|x_i\|^p)^{1/p}$

1.13.2 Matrixnormen ($\underset{\sim}{A} \in \mathbb{K}^{m \times n}, i \in [0, m], j \in [0, n]$)

Für Matrixnormen gilt zu den 3 Standard Axiomen zusätzlich:

4. Submultiplikativität: $\left\|\underset{\sim}{A} + \underset{\sim}{B}\right\| \leq \left\|\underset{\sim}{A}\right\| \cdot \left\|\underset{\sim}{B}\right\|$

Gesamtnorm (Maxnorm $\|\underset{\sim}{A}\|_M$)	$\|\underset{\sim}{A}\|_G = \sqrt{mn} \cdot \max_{i,j} \|a_{ij}\|$
Zeilennorm (max Zeilensumme)	$\|\underset{\sim}{A}\|_\infty = \max_i \sum_{j=1}^{n} \|a_{ij}\|$
Spaltennorm (max Spaltensumme)	$\|\underset{\sim}{A}\|_1 = \max_j \sum_{i=1}^{n} \|a_{ij}\|$
Frobeniusnorm ($\|\|\underset{\sim}{1}\|\|_E = \sqrt{n}$) euklidische Norm	$\|\underset{\sim}{A}\|_E = \sqrt{\sum_{i=1}^{} \sum_{j=1}^{} \|a_{ij}\|^2}$
Spektralnorm, Hilbertnorm	$\|\underset{\sim}{A}\|_\lambda = \sqrt{\lambda_{\max}(\underset{\sim}{A}^\top \cdot \underset{\sim}{A})}$

1.14. Jacobi-Matrix $\vec{f} : \mathbb{R}^n \to \mathbb{R}^m$

$$\frac{\partial}{\partial \vec{x}} \vec{f}(\vec{x}) = \underset{\sim}{J}_f(x) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} = \begin{bmatrix} (\nabla f_1)^T \\ \vdots \\ (\nabla f_m)^T \end{bmatrix}$$

1.15. Wichtige Formeln

Dreiecksungleichung:	$\big\|\|x\| - \|y\|\big\| \leq \|x \pm y\| \leq |x| + |y|$
Cauchy-Schwarz-Ungleichung:	$\left\|\vec{x}^\top \cdot \vec{y}\right\| \leq \|\vec{x}\| \cdot \|\vec{y}\|$
Bernoulli-Ungleichung:	$(1 + x)^n \geq 1 + nx$
Arithmetische Summenformel	$\sum_{k=1}^{n} k = \frac{n(n+1)}{2}$
Geometrische Summenformel	$\sum_{k=0}^{n} q^k = \frac{1 - q^{n+1}}{1 - q}$
Binomialkoeffizient	$\binom{n}{k} = \binom{n}{n-k} = \frac{n!}{k! \cdot (n-k)!}$

1.16. Sinus, Cosinus $\sin^2(x) + \cos^2(x) = 1$

$e^{jx} = \cos(x) + j \cdot \sin(x)$

x	0	$\pi/6$	$\pi/4$	$\pi/3$	$\frac{1}{2}\pi$	π	$\frac{3}{2}\pi$	2π
φ	$0°$	$30°$	$45°$	$60°$	$90°$	$180°$	$270°$	$360°$
sin	0	$\frac{1}{2}$	$\frac{1}{\sqrt{2}}$	$\frac{\sqrt{3}}{2}$	1	0	-1	0
cos	1	$\frac{\sqrt{3}}{2}$	$\frac{1}{\sqrt{2}}$	$\frac{1}{2}$	0	-1	0	1
tan	0	$\frac{\sqrt{3}}{3}$	1	$\sqrt{3}$	$\pm\infty$	0	$\mp\infty$	0

Additionstheoreme
$\cos(x - \frac{\pi}{2}) = \sin x$
$\sin(x + \frac{\pi}{2}) = \cos x$
$\sin 2x = 2 \sin x \cos x$
$\cos 2x = 2 \cos^2 x - 1$
$\sin(x) = \tan(x) \cos(x)$

Stammfunktionen
$\int x \cos(x)\,dx = \cos(x) + x \sin(x)$
$\int x \sin(x)\,dx = \sin(x) - x \cos(x)$
$\int \sin^2(x)\,dx = \frac{1}{2}(x - \sin(x)\cos(x))$
$\int \cos^2(x)\,dx = \frac{1}{2}(x + \sin(x)\cos(x))$
$\int \cos(x) \sin(x) = -\frac{1}{2}\cos^2(x)$

Sinus/Cosinus Hyperbolicus $\sinh, \cosh$
$\sinh x = \frac{1}{2}(e^x - e^{-x}) = -j \sin(jx) \qquad \cosh^2 x - \sinh^2 x = 1$
$\cosh x = \frac{1}{2}(e^x + e^{-x}) = \cos(jx) \qquad \cosh x + \sinh x = e^x$

Kardinalsinus $\text{si}(x) = \frac{\sin(x)}{x}$ **genormt:** $\text{sinc}(x) = \frac{\sin(\pi x)}{\pi x}$

1.17. Integrale $\int e^x\,dx = e^x = (e^x)'$

Partielle Integration: $\int uw' = uw - \int u'w$
Substitution: $\int f(g(x))g'(x)\,dx = \int f(t)\,dt$

$F(x)$	$f(x)$	$f'(x)$
$\frac{1}{q+1}x^{q+1}$	x^q	qx^{q-1}
$\frac{2\sqrt{ax^3}}{3}$	$\sqrt{ax}$	$\frac{a}{2\sqrt{ax}}$
$x\ln(ax) - x$	$\ln(ax)$	$\frac{1}{x}$
$\frac{1}{a^2}e^{ax}(ax - 1)$	$x \cdot e^{ax}$	$e^{ax}(ax + 1)$
$\frac{a^x}{\ln(a)}$	a^x	$a^x \ln(a)$
$-\cos(x)$	$\sin(x)$	$\cos(x)$
$\cosh(x)$	$\sinh(x)$	$\cosh(x)$
$-\ln\lvert\cos(x)\rvert$	$\tan(x)$	$\frac{1}{\cos^2(x)}$

$$\int e^{at}\sin(bt)\,dt = e^{at}\frac{a\sin(bt) + b\cos(bt)}{a^2 + b^2}$$

$\int \frac{dt}{\sqrt{at+b}} = \frac{2\sqrt{at+b}}{a}$
$\int te^{at}\,dt = \frac{at-1}{a^2}e^{at}$
$\int t^2 e^{at}\,dt = \frac{(ax-1)^2+1}{a^3}e^{at}$
$\int xe^{ax^2}\,dx = \frac{1}{2a}e^{ax^2}$

1.18. Exponentialfunktion und Logarithmus

$a^x = e^{x \ln a} \qquad \log_a x = \frac{\ln x}{\ln a} \qquad \ln x \leq x - 1$
$\ln(x^a) = a \ln(x) \qquad \ln(\frac{x}{a}) = \ln x - \ln a \qquad \log(1) = 0$

1.19. Reihen

$\sum_{n=1}^{\infty} \frac{1}{n} \to \infty$ Harmonische Reihe
$\sum_{n=0}^{\infty} q^n \overset{|q|\leq 1}{=} \frac{1}{1-q}$ Geometrische Reihe
$\sum_{n=0}^{\infty} \frac{z^n}{n!} = e^z$ Exponentialreihe

2. Lösung nichtlinearer Gleichungen

Exakte Lösung $x*$, Fehler $\epsilon = x - x*$

2.1. Intervallhalbierung (Nullstellensuche)

Problem: $f(x) = 0$, $f(x)$ stetig in $[a, b]$ und $f(a) \cdot f(b) < 0$
Gesucht: $x^* : f(x^*) = 0, a \leq x^* \leq b$
Konvergenz: $\varepsilon^{(k+1)} = \frac{1}{2}\varepsilon^{(k)} = \left(\frac{1}{2}\right)^{k+1}\varepsilon^{(0)}$
Iterationsschritte bis $\varepsilon < \tau$: $k = \left\lceil \text{ld}\left(\frac{\varepsilon^{(0)}}{\tau}\right)\right\rceil$

2.2. Fixpunktiteration (alg. Iterationsverfahren)

Jedes Problem $f(x) = g(x)$ lässt sich als Fixpunktproblem schreiben:
$x^* = \Phi(x^*) := f(x) - g(x) + x$

$$x^{(k+1)} = \Phi(x^{(k)})$$

$$x^\star = \Phi(x^\star)$$

Falls $|\Phi'(x^\star)| < 1 \Rightarrow$ Konvergenz bzw. stabiler Fixpunkt
Falls $0 < |\Phi'(x^\star)| < 1 \Rightarrow$ lineare Konvergenz mit

$$\varepsilon^{(k+1)} \approx \Phi'(x^\star)\varepsilon^{(k)}$$

Falls $\Phi'(x^\star) = 0$ und $\Phi''(x^\star) \neq 0 \Rightarrow$ quadratische Konvergenz
Allgemein: Konvergenzordnung $n \Leftrightarrow$
$\Phi'(x^\star) = \Phi''(x^\star) = \ldots = \Phi^{(n-1)}(x^\star) = 0$ und $\Phi^{(n)}(x^\star) \neq 0$

2.3. Newton-Raphson

Funktion durch Gerade annähern und Nullstelle bestimmen. An dieser Stelle den Vorgang wiederholen. Nur lokale Konvergenz
Ausgangsproblem:

$$f(x) = 0$$

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})} =: \Phi\left(x^{(k)}\right)$$

2.3.1 Konvergenz

Falls $f'(x^\star) \neq 0$ (einfache Nullstelle) $\Rightarrow$ quadratische Konvergenz mit

$$\varepsilon^{(k+1)} = \left(\frac{1}{2}\frac{f''(x^\star)}{f'(x^\star)}\right)\left(\varepsilon^{(k)}\right)^2$$

Falls $f'(x^\star) = 0$ (Nullstellengrad $n > 1$) $\Rightarrow$ lineare Konvergenz mit

$$\textbf{Konvergenzfaktor} = \frac{n-1}{n}$$

2.3.2 Sekanten-Methode

Falls die Auswertung von $f'(x)$ vermieden werden soll:

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})\left(x^{(k)} - x^{(k-1)}\right)}{f(x^{(k)}) - f(x^{(k-1)})}$$

2.3.3 Mehrdimensional

Theoretisch:

$$\vec{x}^{(k+1)} = \vec{x}^{(k)} - \underset{\sim}{J}_{\vec{f}}^{-1}\left(\vec{x}^{(k)}\right)\vec{f}(x^{(k)})$$

Praktisch:

$$\underset{\sim}{J}_{\vec{f}}\left(\vec{x}^{(k)}\right)\vec{x}^{(k+1)} = \underset{\sim}{J}_{\vec{f}}\left(\vec{x}^{(k)}\right) \cdot \left(\vec{x}^{(k)} - \vec{f}(x^{(k)})\right)$$

3. Lösung linearer Gleichungssysteme

Ausgangsproblem:

$$\underset{\sim}{A}\vec{x} = \vec{b}$$

$\underset{\sim}{A} = \underset{\sim}{M} - \underset{\sim}{N}$	Systemmatrix
$\underset{\sim}{D}$	Diagonalmatrix diag(diag($\underset{\sim}{A}$))
$\underset{\sim}{L}$	Linke untere Dreiecksmatrix tril($\underset{\sim}{A}$, -1)
$\underset{\sim}{U}$	Rechte obere Dreiecksmatrix triu($\underset{\sim}{A}$, 1)

$\underset{\sim}{A} = \underset{\sim}{D} + \underset{\sim}{L} + \underset{\sim}{U}$, keine LR-Zerlegung!

3.1. Allgemeines Iterationsverfahren

Mit beliebiger, invertierbarer Matrix $\underset{\sim}{C} \in \mathbb{R}^{n\times n}$ gilt Umformung:

$\underset{\sim}{A}\vec{x} = \vec{b} \Leftrightarrow \underset{\sim}{C}\vec{x} = \underset{\sim}{C}\vec{x} - \underset{\sim}{A}\vec{x} + \vec{b} \Leftrightarrow \vec{x} = (\underset{\sim}{1} - \underset{\sim}{C}^{-1}\underset{\sim}{A})\vec{x} + \underset{\sim}{C}^{-1}\vec{b}$

Wähle $\underset{\sim}{K} = (\underset{\sim}{1} - \underset{\sim}{C}^{-1}\underset{\sim}{A})$ (alles vor dem $\vec{x}$)

Verfahren konvergiert allgemein, wenn Spektralradius $\rho(\underset{\sim}{K}) < 1$

3.2. Jacobi-Verfahren

Konvergiert falls $\rho(\underset{\sim}{K}_{\text{Jac}}) < 1$ oder falls $\underset{\sim}{A}$ strikt diagonaldominant

Spektralradius $\rho(\underset{\sim}{K}_{\text{Jac}}) = \max|\lambda_i(\underset{\sim}{K}_{\text{Jac}})|$ mit λ_i EW.

$$\underset{\sim}{K}_{\text{Jac}} = \underset{\sim}{D}^{-1}(-\underset{\sim}{L} - \underset{\sim}{U})$$

Matrixdarstellung:

$$\vec{x}^{(k+1)} = \underset{\sim}{K}_{\text{Jac}}\vec{x}^{(k)} + \underset{\sim}{D}^{-1}\vec{b}$$

Komponentenweise:

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1, j\neq i}^{n} a_{ij}x_j^{(k)}\right)$$

Vorkonditionierung:

$$\underset{\sim}{P} = \underset{\sim}{D} \quad \Rightarrow \quad \underset{\sim}{P}^{-1}\underset{\sim}{A}\vec{x} = \underset{\sim}{P}^{-1}\vec{b}$$

3.3. Gauß-Seidel-Verfahren

Unterschied zu Jacobi: Komponentenweise Berechnung von $\vec{x}$ mit bereits iterierten Werten. (Kürzere Iterationszyklen)

$$\underset{\sim}{K}_{\text{GS}} = -(\underset{\sim}{D} + \underset{\sim}{L})^{-1}\underset{\sim}{U}$$

Matrixdarstellung:

$$\vec{x}^{(k+1)} = \underset{\sim}{K}_{\text{GS}}\vec{x}^{(k)} + (\underset{\sim}{D} + \underset{\sim}{L})^{-1}\vec{b}$$

Komponentenweise:

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

Konvergiert falls $\rho(\underset{\sim}{K}_{\text{GS}}) < 1$ oder $\underset{\sim}{A}$ strikt diagonaldominant oder $\underset{\sim}{A}$ positiv definit

Falls A tridiagonal und positiv definit

$$\rho(\underset{\sim}{K}_{\text{GS}}) = \rho(\underset{\sim}{K}_j)^2$$

Vorkonditionierung:

$$\underset{\sim}{P} = \underset{\sim}{D} + \underset{\sim}{L} \quad \Rightarrow \quad \underset{\sim}{P}^{-1}\underset{\sim}{A}\vec{x} = \underset{\sim}{P}^{-1}\vec{b}$$

3.4. Successive Over-Relaxation

$$\underset{\sim}{K}_{\text{SOR}} = (\underset{\sim}{D} + \omega\underset{\sim}{L})^{-1}(\underset{\sim}{D}(1-\omega) - \omega\underset{\sim}{U})$$

Matrixdarstellung:

$$\vec{x}^{(k+1)} = \underset{\sim}{K}_{\text{SOR}}\vec{x}^{(k)} + (\underset{\sim}{D} + \omega\underset{\sim}{L})^{-1}\omega\vec{b}$$

Komponentenweise:

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

Optimale Konvergenz für

$$\omega_{\text{opt}} = \arg\min_{\omega}\rho(\underset{\sim}{K}_{\text{SOR}})$$

Falls A positiv definit und tridiagonal $\Rightarrow \rho(\underset{\sim}{K}_{\text{GS}}) = \rho(\underset{\sim}{K}_{\text{Jac}})^2 < 1$:

$$\omega_{\text{opt}} = \frac{2}{1 + \sqrt{1 - \rho(\underset{\sim}{K}_j)^2}}$$

Vorkonditionierung: $\underset{\sim}{P} = \frac{1}{\omega}(\underset{\sim}{D} + \omega\underset{\sim}{L})$

3.5. Gradienten-Verfahren

Voraussetzungen: $\underset{\sim}{A} = \underset{\sim}{A}^T$ und $\underset{\sim}{A}$ positiv definit (alle EW > 0)

$$\Phi(\vec{x}) = \frac{1}{2}\vec{x}^T\underset{\sim}{A}\vec{x} - \vec{x}^T\vec{b}$$

$$\vec{r}^{(k)} = \vec{b} - \underset{\sim}{A}\vec{x}^{(k)}$$

$$\alpha^{(k)} = \frac{\vec{r}^{(k)T}\vec{r}^{(k)}}{\vec{r}^{(k)T}\underset{\sim}{A}\vec{r}^{(k)}}$$

Optimierung

$$\vec{r}^{(k+1)} = \vec{r}^{(k)} - \alpha^{(k)}\underset{\sim}{A}\vec{r}^{(k)}$$

Matrixdarstellung:

$$\vec{x}^{(k+1)} = \vec{x}^{(k)} + \alpha^{(k)}\vec{r}^{(k)}$$

3.5.1 Konjugierte Gradienten, CG-Verfahren

Vorraussetzung: $\underset{\sim}{A} = \underset{\sim}{A}^T$ und $\underset{\sim}{A}$ positiv definit

Konvergiert in n Schritten für $\underset{\sim}{A} \in \mathbb{R}^{n\times n}$

Vorkonditionierung (PCG): $\underset{\sim}{P}$ symm, pos def.

$\underset{\sim}{P}^{-\frac{1}{2}} \cdot \underset{\sim}{A} \cdot \underset{\sim}{P}^{\frac{1}{2},\top}$ $\qquad \underset{\sim}{P}^{\frac{1}{2},\top}\vec{x}$ $\qquad \underset{\sim}{P}^{-\frac{1}{2}}\vec{b}$

4. Matrix Zerlegung

4.1. LR-Zerlegung von Matrizen (Lower and Upper)

Geeignetes Lösungsverfahren für $\underset{\sim}{A}\vec{x} = \vec{b}$, falls $n < 500$

$\underset{\sim}{A} = \underset{\sim}{L} \cdot \underset{\sim}{R}$ mit $\underset{\sim}{R}$ obere Dreiecksmatrix ($\operatorname{rang}\underset{\sim}{A} = \operatorname{rang}\underset{\sim}{R}$)

4.1.1 Pivotisierung (Spaltenpivotsuche)

Permutationsmatrix $\underset{\sim}{P}^\top = \underset{\sim}{P}^{-1}$ vertauscht Zeilen, damit LR Zerlegung bei 0 Einträgen möglich ist. Tausche so, dass man durch die betragsmäßig größte Zahl dividiert (Pivotelement)

4.1.2 Rechenaufwand (FLOPS)

LU-Zerlegung	$\frac{2}{3}n^3 - \frac{1}{2}n^2 - \frac{1}{6}n$
Vorwärtseinsetzen	$n^2 - n$
Rückwärtseinsetzen	n^2

Bei symmetrischer Matrix für LU Zerlegung halbiert.

Aufwand für Berechnung von $\underset{\sim}{A}^{-1}$: $n^3 + n^2 \in \mathcal{O}(n^3)$

LR-Zerlegung mit Gaußverfahren $\underset{\sim}{A} = \underset{\sim}{L}\underset{\sim}{R}$; $\underset{\sim}{P}^{-1} = \underset{\sim}{P}^\top$

- Sortiere Zeilen von $\underset{\sim}{A}$ mit $\underset{\sim}{P}$ so dass $a_{11} > \ldots > a_{n1}$
- Zerlegen von $\underset{\sim}{P}\underset{\sim}{A}\vec{x} = \vec{b}$ zu $\underset{\sim}{L}(\underset{\sim}{R}\vec{x}) = \underset{\sim}{L}\vec{y} = \underset{\sim}{P}^\top\vec{b}$ mit Zerlegungsmatrix: (Beispiel für 2×2) $\underset{\sim}{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \rightarrow \begin{bmatrix} a & b \\ \frac{c}{a} & d - \frac{c}{a}b \end{bmatrix} = \underset{\sim}{A}^*$ mit den Eliminationsfaktoren $l_{ik} = \frac{a_{ik}}{a_{kk}} \overset{z.B.}{=} \frac{c}{a}$
- Für jede Spalte der unteren Dreiecksmatrix wiederholen. Für eine $n \times n$ Matrix braucht man $n-1$ Durchläufe
- $\underset{\sim}{R} = \text{triu}(\underset{\sim}{A}^*)$ (obere $\triangle$-Matr. von $\underset{\sim}{A}^*$, inkl. Diagonalelem.)
- $\underset{\sim}{L} = \text{tril}(\underset{\sim}{A}^*, -1) + \underset{\sim}{1}$ (untere $\triangle$-Matr. mit 1en auf Diag.)
- **Vorwärtseinsetzen:** $\underset{\sim}{L}\vec{y} = \vec{b}$ bzw. $\underset{\sim}{L}\vec{y} = \underset{\sim}{P}^\top\vec{b}$ (löse nach $\vec{y}$)
- **Rückwärtseinsetzen:** $\underset{\sim}{R}\vec{x} = \vec{y}$ (Löse nach $\vec{x}$)

4.2. QR-Zerlegung (existiert immer)

$\underset{\sim}{A} = \underset{\sim}{Q}\underset{\sim}{R}$ mit $\underset{\sim}{Q}^{-1} = \underset{\sim}{Q}^\top$

Berechnung (Verfahren): Housholder (numerisch stabil) , Gram-Schmidt, Givens Rotation.

$\underset{\sim}{A} \xrightarrow{EZF} \underset{\sim}{H}\underset{\sim}{A} \xrightarrow{EZF} \underset{\sim}{\tilde{H}}\underset{\sim}{H}\underset{\sim}{A} = \underset{\sim}{R} \Rightarrow \underset{\sim}{A} = \underset{\sim}{H}^\top\underset{\sim}{\tilde{H}}^\top\underset{\sim}{R}$

Aufgabe: Finde Vektor $\vec{v}$ der Senkrecht auf $\underset{\sim}{H}$ steht.

Lösen von LGSen mit der QR Zerlegung

Bestimme $\vec{x}$ durch Rückwärtssubsitution aus $\underset{\sim}{R}\vec{x} = \underset{\sim}{Q}^\top\vec{b}$

QR-Zerlegung mit Householder-Transformation für $\underset{\sim}{A} \in \mathbb{R}^{m\times n}$

- Setze $\vec{a} = \vec{s}_1$ (erste Spalte) und $\vec{v} = \vec{a} + \operatorname{sgn}(a_1)\|\vec{a}\|\vec{e}_1$
- Berechne *Householder*-Trafomatrix $\underset{\sim}{H}_{\vec{v}_1} = \underset{\sim}{1}_m - \frac{2}{\vec{v}^\top\vec{v}}\vec{v}\vec{v}^\top$
- Erhalte $\underset{\sim}{A}^* = \underset{\sim}{H}_{\vec{v}_1}\underset{\sim}{A}$ (ersten Spalte bis auf a_{11} nur Nullen)
- Wiederhole für $\underset{\sim}{A}^*$ ohne 1. Zeile und Spalte (Untermatr. $\underset{\sim}{A}^*_{11}$) Erweitere $\underset{\sim}{H}^*_{\vec{v}_2}$ oben mit $\underset{\sim}{1}_m$ zu $\underset{\sim}{H}_{\vec{v}_2}$ ($h_{11} = 1$)
- Nach $p = \min\{m-1, n\}$ Schritten: $\underset{\sim}{H}_{\vec{v}_p}\underset{\sim}{A}^* = \underset{\sim}{R}$weil
- $\underset{\sim}{Q}^\top = \underset{\sim}{H}_{\vec{v}_p}\cdots\underset{\sim}{H}_{\vec{v}_1}$ und $\underset{\sim}{Q}^\top\underset{\sim}{A} = \underset{\sim}{R}$

4.3. Orthogonalisierungsverfahren nach Gram-Schmidt

Berechnet zu n Vektoren $\vec{v}_i$ ein Orthogonalsystem $\vec{b}_i$ $\quad (i \in [1; n])$

$$\vec{b}_1 = \vec{v}_1 \qquad \vec{b}_i = \vec{v}_i - \sum_{k=1}^{i-1}\frac{\vec{b}_k^\top\cdot\vec{v}_i}{\vec{b}_k^\top\cdot\vec{b}_k}\vec{b}_k$$

Erhalte Orthonormalsystem durch $\vec{b}'_i = \vec{b}_i / \left\|\vec{b}_i\right\|$

QR-Zerlegung: $\underset{\sim}{A} = \underset{\sim}{Q}\underset{\sim}{R}$ mit $\underset{\sim}{Q} = [\vec{b}'_1, \ldots, \vec{b}'_n]$ $\quad \underset{\sim}{R} = \underset{\sim}{Q}^\top\underset{\sim}{A}$

4.4. Givens Rotation (Jacobi-Rotation) $\quad \underset{\sim}{G}^{-1} = \underset{\sim}{G}^\top$

Die orthogonale Givens-Rotationsmatrix $\underset{\sim}{G}$ entspricht der Einheitsmatrix wobei 4 Elemente die Form $\begin{bmatrix} c & s \\ -s & c \end{bmatrix}$ haben. Die c beliebig auf der Hauptdiagonalen und $s/-s$ in der gleichen Zeile/Spalte wie die c.

QR-Zerlegung mit Givens-Rotation für $\underset{\sim}{A} \in \mathbb{R}^{m\times n}$

- Initialisierung: Setze $\underset{\sim}{R} = \underset{\sim}{A}$ und $\underset{\sim}{G}_{\text{gesamt}} = \underset{\sim}{1}_m$
- Wiederhole folgende Schritte für alle Elemente r_{xy} in $\underset{\sim}{R}$, welche 0 werden müssen um obere Dreiecksmatrix zu erhalten. (Reihe x, Spalte y), verfahre spaltenweise (links nach rechts) und in jeder Spalte von oben nach unten:
- Setze $a = r_{yy}$ (Hauptdiagonalelement in dieser Spalte)
- Setze $b = r_{xy}$ (Wert, welcher durch 0 ersetzt werden soll)
- Berechne $c := \frac{a}{p}$ und $s := \frac{b}{p}$ mit $p := \sqrt{a^2 + b^2}$
- Setze $\underset{\sim}{G} = (g_{ij}) = \begin{cases} c & i = x, j = x \\ c & i = y, j = y \\ s & i = y, j = x \\ -s & i = x, j = y \\ \text{Einheitsmatrix} & \text{sonst} \end{cases}$
- Setze $\underset{\sim}{R} = \underset{\sim}{G}\underset{\sim}{R}$ und $\underset{\sim}{G}_{\text{gesamt}} = \underset{\sim}{G}\underset{\sim}{G}_{\text{gesamt}}$
- Fahre, falls nötig, mit nächstem Element in $\underset{\sim}{R}$ fort
- Erhalte $\underset{\sim}{Q} = \underset{\sim}{G}^\top_{\text{gesamt}}$ Löse

4.5. Dünnbesetzte Matrizen

Ziel: effizienteres Speichern von Matrizen mit vielen 0 Einträgen. $\quad \underset{\sim}{A} = \begin{bmatrix} a & 0 & 0 & 0 \\ 0 & b & c & 0 \\ 0 & 0 & 0 & d \\ e & 0 & f & 0 \end{bmatrix}$

	COO	CRS	CCS
row	$\{1,2,2,3,4,4\}$		$\{1,4,2,2,4,3\}$
rowptr		$\{1,2,4,5,7\}$	
col	$\{1,2,3,4,1,3\}$	$\{1,2,3,4,1,3\}$	
colptr			$\{1,3,5,6,7\}$
val	$\{a,b,c,d,e,f\}$	$\{a,b,c,d,e,f\}$	$\{a,b,c,d,e,f\}$

COO Zeilen und Spaltenindex von val

CRS rowptr(i) zeigt auf j-tes Element von col

CCS colptr(i) zeigt auf j-tes Element von row

rowptr(1)=1, rowptr(n)=n+1, gibt an, bei welchem element (in col) die neue Zeile beginnt.

5. Numerische Differentiation

5.1. Vorwärtsdifferenz

$$f'(x_0) \approx \tilde{f}'_{\text{Vor}}(x_0) = \frac{f(x_0+h)-f(x_0)}{h}$$

$$f'(x_0) - \tilde{f}'_{\text{Vor}}(x_0) \in \mathcal{O}(h)$$

5.2. Rückwärtsdifferenz

$$f'(x_0) \approx \tilde{f}'_{\text{Rück}}(x_0) = \frac{f(x_0)-f(x_0-h)}{h}$$

$$f'(x_0) - \tilde{f}'_{\text{Rück}}(x_0) \in \mathcal{O}(h)$$

5.3. Zentrale Differenz

$$f'(x_0) \approx \tilde{f}'_{\text{Zentral}}(x_0) = \frac{f(x_0+h)-f(x_0-h)}{2h}$$

$$f'(x_0) - \tilde{f}'_{\text{Zentral}}(x_0) \in \mathcal{O}(h^2)$$

$h_{opt} = \sqrt[3]{\frac{3\epsilon}{M}}$ Max. Rundungsfehler ϵ

6. Numerische Integration

6.1. Polynom-Ansätze

$$\int_a^b f(x)\,\mathrm{d}x \approx \int_a^b P(x)\,\mathrm{d}x$$

Genauigkeitsgrad n: Integration eines Polynoms vom Grad n ist exakt

6.1.1 Lagrange

$$P(x) = \sum_{k=0}^{n} L_{n,k}(x)\cdot f(x_k)$$

$$L_{n,k}(x) = \prod_{i=0, i\neq k}^{n} \frac{x-x_i}{x_k-x_i}$$

6.1.2 Differenzen

$f[x_i] = f(x_i)$ $\quad f[x_i, x_{i+1}] = \frac{f[x_{i+1}]-f[x_i]}{x_{i+1}-x_i}$

$$f[x_i,\ldots,x_j] = \frac{f[x_{i+1},\ldots,x_j]-f[x_i,\ldots x_{j-1}]}{x_j-x_i}$$

$$P(x) = f[x_0] + \sum_{k=1}^{n} f[x_0,\ldots,x_k](x-x_0)\ldots(x-x_{k-1})$$

6.2. Newton-Cotes

$$\int_a^b f(x)\,\mathrm{d}x \approx \sum_{i=0}^{n} g_i f(x_i)$$

$$h = \frac{b-a}{n}$$

6.2.1 Trapez (Lineares Lagrage Polynom)

falls $n = 1$:

$$\int_a^b f(x)\,\mathrm{d}x \approx (b-a)\,\frac{f(a)+f(b)}{2}$$

Allgemein zusammenges. Trapez: n =#Kanten = #Stützstellen - 1

$$\int_{x_0}^{x_n} f(x)\,\mathrm{d}x \approx \frac{h}{2}\left(f(x_0) + \sum_{k=1}^{n-1} 2f(x_k) + f(x_n)\right)$$

Approximationsfehler $= -\frac{h^3}{12} f''(\xi) \propto \frac{1}{n^3}$

6.3. Simpson-Regel (Quadratisches Lagrage Polynom)

Simpson $\frac{1}{3}$ (Fassregel) (falls $n = 2$):

$$\int_a^b f(x)\,\mathrm{d}x \approx \frac{b-a}{6}(f(a)+4f(a+h)+f(b))$$

Simpson $\frac{3}{8}$ (falls $n = 3$):

$$\int_a^b f(x)\,\mathrm{d}x \approx \frac{3h}{8}(f(a)+3f(a+h)+3f(a+2h)+f(b))$$

Allgemein (zusammengesetzte Simpsonregel $\frac{1}{3}$):

$$\int_a^b f(x)\,\mathrm{d}x \approx \frac{h}{3}\left(f(a)+f(b)+\sum_{k=1}^{n-1} a_k f(a+k\cdot h)\right)$$

$$a_k = 3 + (-1)^{k+1}$$

Approximationsfehler $\propto -h^5 f^{(4)}(\xi) \propto \frac{1}{n^5}$

Rundungsfehler konstant: $e(n) = (b-a)\varepsilon$

6.4. Kubische Splines $S(x)$

Stückweise Approximation von $f(x)$ durch n kubische Polynome mit $S(x_i) = f(x_i)$. Bestimme Parameter a, b, c, d für jedes Teilstück:

$$S_j(x) = a_j + b_j(x-x_j) + c_j(x-x_j)^2 + d_j(x-x_j)^3$$

$$S'_j(x) = b_j + 2c_j(x-x_j) + 3d_j(x-x_j)^2$$

Für $j = 0, 1, \ldots, n-1$:

$$S_j(x_j) = f(x_j) \wedge S_j(x_{j+1}) = f(x_{j+1})$$

Für $j = 0, 1, \ldots, n-2$:

$$S_j(x_{j+1}) \overset{!}{=} S_{j+1}(x_{j+1}) \equiv a_{j+1}$$

$$S'_j(x_{j+1}) \overset{!}{=} S'_{j+1}(x_{j+1}) \equiv b_{j+1}$$

$$S''_j(x_{j+1}) \overset{!}{=} S''_{j+1}(x_{j+1}) \equiv 2c_{j+1}$$

Freier bzw. natürlicher Rand: $S''(x_0) = S''(x_n) = 0$

Eingespannter Rand: $S'(x_0) = f'(x_0) \wedge S'(x_n) = f'(x_n)$

Parameterbestimmung

- $a_j = f(x_j)$ und $h_j = x_{j+1} - x_j$
- Löse LGS für $\vec{c}$: $\underset{\sim}{A}\vec{c} = \vec{l}$

$$\underset{\sim}{A} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ h_0 & 2(h_0+h_1) & h_1 & & \vdots \\ \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & h_{n-2} & 2(h_{n-2}-h_{n-1}) & h_{n-1} \\ 0 & \cdots & 0 & 0 & 1 \end{bmatrix}$$

$$\vec{l} = \begin{bmatrix} 0 \\ \frac{3}{h_1}(a_2-a_1) - \frac{3}{h_0}(a_1-a_0) \\ \vdots \\ \frac{3}{h_{n-1}}(a_n-a_{n-1}) - \frac{3}{h_{n-2}}(a_{n-1}-a_{n-2}) \\ 0 \end{bmatrix}$$

- $b_j = \frac{1}{h_j}(a_{j+1}-a_j) - \frac{h_j}{3}(2c_j + c_{j+1})$
- $d_j = \frac{1}{3h_j}(c_{j+1}-c_j)$

7. Least Squares

7.1. Ausgleichsrechnung

Gegeben: n Datenpunkte (x_i, y_i), Gesucht: Eine Polynom-Funktion f welche die Datenpunkte möglichst gut (kleinstes Fehlerquadrat) approximiert. Es gilt: $\vec{f}_{\vec{\alpha}}(\vec{x}) = \vec{y} + \vec{r} \approx \vec{y}$ mit Residum $\vec{r}$

Bestimme k Parameter α_j so, dass Fehlerquadrat $\vec{r}^{\top}\vec{r}$ minimiert wird.

Erstelle $\underset{\sim}{A} \in \mathbb{R}^{n\times k} = \begin{bmatrix}\vec{x}^{k-1} & \ldots & \vec{x} & \vec{1}\end{bmatrix}$ (Polynom Grad $k-1$)

$$\min_{\alpha}\vec{r} = \min_{\alpha}\left\|\underset{\sim}{A}\vec{\alpha} - \vec{y}\right\|_2^2 = \min_{\alpha}\left\|\vec{y} - \underset{\sim}{A}\vec{\alpha}\right\|_2^2$$

Minimierung durch Ableitung: $\forall j \in [1,k] : \frac{\partial(\vec{r})^2}{\partial\alpha_j} \overset{!}{=} 0$

Dadurch ergibt sich: $\underset{\sim}{A}^{\top}\underset{\sim}{A}\vec{\alpha} = \underset{\sim}{A}^{\top}\vec{y}$

7.1.1 Lineare Ausgleichsrechnung ($k = 2$)

$f_{\vec{\alpha}}(x) = \alpha_1 x + \alpha_0$ $\qquad \underset{\sim}{A} = [\vec{x}\ \ \vec{1}]$ $\qquad \vec{\alpha} = \begin{bmatrix}\alpha_1 \\ \alpha_0\end{bmatrix}$

$$\underset{\sim}{A}^{\top}\underset{\sim}{A} = \begin{bmatrix}\vec{x}^{\top}\vec{x} & \sum x_i \\ \sum x_i & \vec{1}^{\top}\vec{1} = n\end{bmatrix} \qquad \vec{\alpha} = (\underset{\sim}{A}^{\top}\underset{\sim}{A})^{-1}(\underset{\sim}{A}^{\top}\vec{y})$$

$$\arg\min_{\alpha_1,\alpha_0} E(\alpha_1,\alpha_0) = \sum_{i=1}^{n}(y_i - (\alpha_1 x_i + \alpha_0))^2$$

7.1.2 Polynomial Least Squares

$$f_{\vec{\alpha}}(x) = P(x,\vec{\alpha}) = \alpha_k x^k + \ldots + \alpha_1 x + \alpha_0$$

$$\arg\min_{\alpha_0,\ldots,\alpha_{k-1}} E_n(\alpha_0,\ldots,\alpha_{k-1}) = \sum_{i=1}^{n}(y_i - P(\vec{\alpha}, x_i))^2$$

Minimierung durch Ableitung: $\forall i \in [0, k-1] : \frac{\partial E_n}{\partial\alpha_j} \overset{!}{=} 0$

Lösen der Normalengleichung

- Bestimme eine reduzierte QR-Zerlegung $\underset{\sim}{A} = \underset{\sim}{\tilde{Q}}\underset{\sim}{\tilde{R}}$ mit $\underset{\sim}{\tilde{Q}} \in \mathbb{R}^{n\times k}$, $\underset{\sim}{\tilde{R}} \in \mathbb{R}^{k\times k}$
- Löse $\underset{\sim}{\tilde{R}}\vec{x} = \underset{\sim}{\tilde{Q}}^{\top}\vec{y}$

$$\left\|\vec{b} - \underset{\sim}{A}\vec{x}\right\|^2 = \left\|\underset{\sim}{Q}^{\top}(\vec{b} - \underset{\sim}{A}\vec{x})\right\|^2 = \left\|\tilde{\vec{b}} - \underset{\sim}{\tilde{R}}\vec{x}\right\|^2 + \|\vec{c}\|^2 \geq \left\|\vec{c}^2\right\|$$

8. Numerischer Rechenaufwand

8.1. Elementare Rechenoperationen

$\underset{\sim}{A} \in \mathbb{R}^{m\times n}$ $\quad \underset{\sim}{B} \in \mathbb{R}^{n\times k}$ $\quad \vec{x} \in \mathbb{R}^n$ $\quad \vec{b} \in \mathbb{R}^m$

Was?	Addition	Multiplikation	Sqrt
$\|\vec{x}\|$	$n-1$	n	1
$\vec{x}^{\top}\vec{x}$	$n-1$	n	
$\vec{x}\vec{b}^{\top}$		nm	
$\underset{\sim}{A}\vec{x}$	$n(n-1)$	n^2	
$\underset{\sim}{A}\underset{\sim}{A}^{\top}$	$m^2(n-1)$	$m^2\cdot n$	
$\underset{\sim}{A}\underset{\sim}{B}$	$m(n-1)k$	mnk	

9. Numerische Lösung von Differentialgleichungen

Ausgangsproblem: DGL

$$\dot{x}(t) = f(x(t))$$

Idee: Anstatt die Funktion $x(t)$ zu bestimmen, wird versucht die Lösung $x(t = t^*)$ für ein bestimmtes t^* zu finden. Man kennt bereits eine Lösung $x(t_0)$ und hangelt sich von dort mit Schritten $x(t_0 + \Delta t\nu)$ (Schrittweite Δt, ν-ter Schritt) nach vorne bis man $x(t^*)$ erreicht.

$$\hat{x}(\nu) \overset{\wedge}{=} \hat{x}^{(\nu)} = x(t_0 + \Delta t\nu)$$

$$\hat{f}(\nu) \overset{\wedge}{=} f(t_0 + \Delta t\nu)$$

9.1. Expliziter Euler

$$\hat{x}^{(\nu+1)} = \hat{x}^{(\nu)} + \Delta t\cdot\hat{f}\left(\hat{x}^{(\nu)}\right)$$

stabil für $0 < \Delta t < 2$, instabil für $\Delta t > 2$

9.2. Impliziter Euler

$$\hat{x}^{(\nu+1)} = \hat{x}^{(\nu)} + \Delta t\cdot\hat{f}\left(\hat{x}^{(\nu+1)}\right)$$

Löse Gleichung nach $\hat{x}^{(\nu+1)}$

9.3. Trapez

$$\hat{x}(\nu+1) = \hat{x}(\nu) + \frac{\Delta t}{2}(\hat{f}(\nu) + \hat{f}(\nu+1))$$

9.4. Gear $\mathcal{O}((\Delta t)^2)$

$$\hat{x}(\nu+2) = \frac{4}{3}\hat{x}(\nu+1) - \frac{1}{3}\hat{x}(\nu) + \frac{2}{3}\Delta t\hat{f}(\nu+2)$$

9.5. Heun

$$\hat{x}^{[P]}(\nu+1) = \hat{x}(\nu) + \Delta t\hat{f}(\nu, \hat{x}(\nu))$$

$$\hat{x}(\nu+1) = \hat{x}(\nu) + \frac{\Delta t}{2}\left(\hat{f}(\nu,\hat{x}(\nu)) + \hat{f}(\nu+1, \hat{x}^{[P]}(\nu+1))\right)$$

9.6. k-Schritt-Adams-Bashforth

$$\hat{x}(\nu+k) = \hat{x}(\nu+k-1) + \Delta t\sum_{i=0}^{k-1} b_{k,i}\hat{f}(\nu+i)$$

$b_{i,k}$	$i=0$	$i=1$	$i=2$	$i=3$
$k=1$	1			
$k=2$	$-\frac{1}{2}$	$\frac{3}{2}$		
$k=3$	$\frac{5}{12}$	$-\frac{16}{12}$	$\frac{23}{12}$	
$k=4$	$-\frac{9}{24}$	$\frac{37}{24}$	$-\frac{59}{24}$	$\frac{55}{24}$

9.7. Finite Differenzen $\dot{x}(t) + \alpha x(t) = g(t) + c$

Stützstellen $t_0, \ldots t_n$

Vorwärtsdifferenz mit $x(t_n)$ bekannt: $\dot{x}(t_0) = \frac{1}{h}(x(t_1)) - x(t_0))$

$$\frac{1}{h}\begin{bmatrix}-1 & 1+\alpha h & & \\ & -1 & 1+\alpha h & \\ & & .. & .. \\ & & & h\end{bmatrix}\begin{pmatrix}x(t_0)\\ x(t_1)\\ ..\\ x(t_n)\end{pmatrix} = \begin{pmatrix}g(t)+c\\ g(t)+c\\ ..\\ x_n\end{pmatrix}$$

Rückwärtsdifferenz mit $x(t_0)$ bekannt: $\dot{x}(t_1) = \frac{1}{h}(x(t_1)) - x(t_0))$

$$\frac{1}{h}\begin{bmatrix}h & & & \\ -1 & 1+\alpha h & & \\ & .. & .. & \\ & & -1 & 1+\alpha h\end{bmatrix}\begin{pmatrix}x(t_0)\\ x(t_1)\\ ..\\ x(t_n)\end{pmatrix} = \begin{pmatrix}x_0\\ g(t)+c\\ ..\\ g(t)+c\end{pmatrix}$$

Für zweite Ableitung immer -1, 2, -1 in einer Zeile

10. Matlab Sample Code

```
function x = gaussVerfahren(A, b)
    [L, U, P] = LUZerlegung(A);
    [y] = vorwaertsSubstitution(L, P, b);
    [x] = rueckwaertsSubstitution(U, y);
end
```

```
function [L, U, P] = LUZerlegung(A)
    n = size(A, 1);
    L = zeros(n, n);
    P = eye(n);

    for i = 1:n-1
        [pivot, pivotIndex] = max(abs(A(i:n, i)));
        pivotIndex = pivotIndex + (i - 1);
        pivot = A(pivotIndex, i);
        Psub = eye(n);
        Psub(:, [i, pivotIndex]) = Psub(:, [pivotIndex, i]);
        A([i, pivotIndex], :) = A([pivotIndex, i], :);
        L([i, pivotIndex], :) = L([pivotIndex, i], :);
        P = Psub*P;
        pivotRow = A(i, i+1:n);
        for j = i+1:n
            factor = A(j, i)/pivot;
            L(j, i) = factor;
            currentRow = A(j, i+1:n);
            A(j, i+1:n) = currentRow - factor*pivotRow;
            A(j, i) = 0;
        end
    end

    U = A;
    L = L + eye(n);
end
```

```
function [y] = vorwaertsSubstitution(L, P, b)
    n = size(L, 1);
    y = zeros(n, 1);
    b = P*b;
    y(1) = b(1)/L(1, 1);

    for i = 2:n
        rowSum = L(i, 1:i-1)*y(1:i-1);
        y(i) = (b(i) - rowSum)/L(i, i);
    end
end
```

```
function [x] = rueckwaertsSubstitution(U, y)
    n = size(U, 1);
    x = zeros(n, 1);
    x(n) = y(n)/U(n, n);

    for i = n-1:-1:1
        rowSum = U(i, i+1:n)*x(i+1:n);
        x(i) = (y(i) - rowSum)/U(i, i);
    end
end
```

```
function [ x_k,r_k,alpha_k ] = conjugateGradientIteration( A, b,x0,N )
    x_k = zeros(length(x0),N+1);
    r_k = zeros(length(x0),N+1);
    p_k = zeros(length(x0),N+1);

    alpha_k = zeros(1,N);
    beta_k = zeros(1,N);

    x_k(:,1) = x0;
    r_k(:,1) = b-A*x0;
    p_k(:,1) = r_k(:,1);

    for i = 1:N
        Ap = A*p_k(:,i);
        alpha_k(i) = (p_k(:,i)'*r_k(:,i))./(p_k(:,i)'*Ap);
        x_k(:,i+1) = x_k(:,i) + alpha_k(i).*p_k(:,i);
        r_k(:,i+1) = r_k(:,i) -alpha_k(i).*Ap;
        beta_k(i) = (Ap'*r_k(:,i+1))./(Ap'*p_k(:,i));
        p_k(:,i+1) = r_k(:,i+1) - beta_k(i).*p_k(:,i);
    end
end
```

```
function [x_k,r_k,alpha_k] = gradientIteration(A,b,x0,N)
    x_k = zeros(length(x0),N+1);
    r_k = zeros(length(x0),N);
    alpha_k = zeros(1,N);

    x_k(:,1) = x0;
    for i = 1:N
        r_k(:,i) = b - A*x_k(:,i);
        alpha_k(i) = (r_k(:,i)'*r_k(:,i))./(r_k(:,i)'*A*r_k(:,i));
        x_k(:,i+1) = x_k(:,i) + alpha_k(i).*r_k(:,i);
    end
end
```

```
function [Q, R] = householder(A)
    n = size(A, 1);
    identity = eye(n);
    Q = eye(n);

    for i=1:(n-1)
        a = zeros(n, 1);
        a(i:end) = A(i:end, i);
        v = a + sign(a(i))*norm(a)*identity(:, i);
        Qpartial = identity - 2/(v'*v)*(v*v');
        Q = Qpartial*Q;
        A = Qpartial*A;
    end

    R = A;
    Q = Q';
end
```

```
function [Q, R] = givensRotation(A)
    n = size(A, 1);
    Q = eye(n);
    R = A;

    for i = 1:(n-1)
        for j = i+1:n;
            G = createGivensRotation(R, j, i);
            Q = G*Q;
            R = G*R;
        end
    end

    Q = Q';
end
```

```
function [G] = createGivensRotation(A, row, col)
    a1 = A(col, col);
    a2 = A(row, col);
    p = sqrt(a1*a1 + a2*a2);
    c = a1/p;
    s = a2/p;
    G = eye(size(A, 1));
    G(row, row) = c;
    G(col, col) = c;
    G(row, col) = (-1)*s;
    G(col, row) = s;
end
```

```
function [a,b,c,d] = splineParameter(xi,f)
    n = max(size(xi));% Anzahl der Stuetzstellen
    a = f(xi);
    h = zeros(n-1,1);% Schrittweite

    for i=1:n-1
       h(i) = xi(i+1)-xi(i);
    end
    A = sparse(zeros(n,n));% Matrix fuer LGS
    bs = zeros(n,1);% rechte Seite fuer LGS
    for i=2:n-1
       A(i,i) = 2*(h(i)+h(i-1));
       A(i,i-1) = h(i-1);
       A(i,i+1) = h(i);
       bs(i) = (3/h(i))*(a(i+1)-a(i)) - (3/h(i-1))*(a(i)-a(i-1));
    end
    A(1,1) = 1;
    A(n,n) = 1;
    c = A\bs;% Loesung des LGS
    b = zeros(n,1);% Parameter b fuer Splines
    d = zeros(n,1);% Parameter d fuer Splines
    for i=1:n-1
        b(i) = (1/h(i))*(a(i+1)-a(i))-(h(i)/3)*(2*c(i)+c(i+1));
        d(i) = (1/(3*h(i)))*(c(i+1)-c(i));
    end
end
```

```
    function b = multCRS(m,n,nonzeros,rowptr,col,val,x)
    b = zeros(m,1);
    for i=1:m
        for j=rowptr(i):rowptr(i+1)-1
            b(i) = b(i) + val(j)*x(col(j));
        end
    end
end
```

11. Blabla Fragen

1. **Nennen Sie einen Vorteil der Dividierten Differenzen gegenüber der Lagrange-Interpolation.**
   - geringerer Aufwand
   - keine komplette Neuberechnung bei neuer Stützstelle
2. **Nennen Sie einen Nachteil der Polynominterpolation gegenüber der Spline-Interpolation.**
   - Oszillation am Intervallrand $\Rightarrow$ großer Fehler am Rand
3. **Nennen Sie zwei Vorteile des Adams-Bashfort-3-Schrittverfahrens gegenüber der Trapez- Methode zum Lösen nichtlinearer Differentialgleichungen.**
   - höhere Genauigkeit (lokaler Fehler kleiner bei gleicher Schrittweite)
   - explizites Verfahren (geringerer Rechenaufwand)
4. **Nennen Sie zwei Vorteile des Gauß-Verfahrens gegenüber dem Jacobi-Verfahren.**
   - für alle nicht-singulären Matrizen lösbar
   - geringerer Aufwand, wenn das gleiche Gleichungssystem mit verschiedenen rechten Seiten gelöst werden soll.
5. **Nennen Sie drei numerische Integrationsverfahren, die die gleiche (lokale) Fehlerordnung wie das Trapezverfahren besitzen.**
   - Gear
   - Taylor-Verfahren zweiter Ordnung
   - Zweischritt Adams Bashfort
6. **Nennen Sie einen Vorteil des Jacobi-Verfahrens gegenüber dem Gauß-Seidel-Verfahren.**
   - leicht parallelisierbar
7. **Nennen Sie einen Nachteil des Jacobi-Verfahrens gegenüber dem Gauß-Seidel-Verfahren.**
   - langsamere Konvergenz
8. **Geben Sie an, welche numerischen Probleme bei Anwendung der Sekantenmethode zur Bestimmung der Nullstelle von $F(x)$ in der Nähe der Nullstelle x_0 auftreten können.**
   - In der Nähe der Nullstelle ist $F(x^{(k)}) \approx 0$, weshalb in der Iterationsvorschrift näherungsweise der Term $\frac{0}{0}$ auftreten kann. Dementsprechend können Auslöschungsfehler auftreten.

12. Sonstiges

12.1. Graphen $G = (V, E)$

m Knoten (vertices) v_i, n Kanten (edges) e_j
einfach/multi: nur eine/mehrere Kanten zwischen zwei Knoten
gerichtet: Kanten nur in eine Richtung. gewichtet: Kanten haben Werte
Zyklus: Gleicher Start und Endknoten $v_{\text{start}} = v_{\text{end}}$
Pfad: Alle Knoten verschieden $v_k \neq v_l$
Kreis: Zyklus und Pfad zusammen

Adjazenzmatrix $\underset{\sim}{\boldsymbol{A}} = (a_{ij}) \in \mathbb{B}^{|V|\times|V|}$:

$$a_{ij} = \begin{cases} 1 & \text{falls } v_i \text{ mit } v_j \text{ verbunden ist} \\ 0 & \text{sonst}(\nexists e : (v_i, v_j) \in e) \end{cases}$$

$\underset{\sim}{\boldsymbol{A}}$ immer symmetrisch, geht nur für ungerichtete Graphen!

Inzidenzmatrix $\underset{\sim}{\boldsymbol{B}} = (b_{ij}) \in \mathbb{B}^{|V|\times|E|}$:

$$b_{ij} = \begin{cases} -1 & \text{falls } e_j \text{ bei } v_i \text{ startet } (e_j = (v_i, v_x)) \\ 1 & \text{falls } e_j \text{ bei } v_i \text{ endet } (e_j = (v_x, v_i)) \\ 0 & \text{sonst } (v_i \notin e_j) \end{cases}$$

Jede Zeile (für jede Kante) enthält genau einmal 1 und -1
Rang von $\underset{\sim}{\boldsymbol{B}}$: $|V|-$#zusammenhängende Gebiete. Maximal $|V| - 1$!
Nullraum $\ker \underset{\sim}{\boldsymbol{B}}$: Vektoren der Gebiete. Also mindestens $\vec{1} \in \ker \underset{\sim}{\boldsymbol{B}}$

Laplacematrix $\underset{\sim}{\boldsymbol{L}} = \underset{\sim}{\boldsymbol{B}}^\top \underset{\sim}{\boldsymbol{B}} = (l_{ij}) \in \mathbb{B}^{|V|\times|V|}$

$$l_{ij} = \begin{cases} d_i & \text{falls } i = j \text{ und genau } d_i \text{ Kanten von } v_i \text{weggehen} \\ -1 & \text{falls } i \neq j \text{ und die Kante } (v_i, v_j) \text{ existiert} \\ 0 & \text{sonst} \end{cases}$$

12.2. Kirchhoff (Inzidenzmatrix B)

KCL: $\underset{\sim}{\boldsymbol{B}}^\top \vec{i} = \vec{0}$ KVL: $\vec{u} - \underset{\sim}{\boldsymbol{B}}\, \vec{u}_{\text{knoten}} = \vec{0}$ Ohm: $\vec{i} = \underset{\sim}{\boldsymbol{G}} \cdot \vec{u}$

12.3. Komplexität / Landau-Notation

Definiert Zeit und Platzbedarf von Algorithmen ($\exists c \in \mathbb{R}\, \forall n > n_0$)

$f \in \mathcal{O}(g(n))$	$\Rightarrow$	$0 \leq f(n) \leq c \cdot g(n)$
$f \in \Omega(g(n))$	$\Rightarrow$	$f(n) \geq c \cdot g(n) \geq 0$
$f \in \Theta(g(n))$	$\Rightarrow$	$c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$

Schrankenfunktionen (für große $n \in \mathbb{N}$)
$1 < \log_{10}(n) < \ln(n) < \log_2(n) < \sqrt{n} < n < n \cdot \ln(n) < (\log n)! < n^2 < e^n < n! < n^n < 2^{2^n}$

12.4. Gleitkommadarstellung nach IEEE 754

Bitverteilung(single/double):

$s(1)$	$e(8/11)$	$f(23/52)$

s: Vorzeichen, e: Exponent, f: Mantisse Normalisiert: 1.xxx
Wert $Z = (-1)^s \cdot 1.f \cdot 2^{e-127}$ Genauigkeit: $M = 2^{-f}$

12.5. Singulärwertszerlegung $\underset{\sim}{\boldsymbol{A}} \in \mathbb{K}^{m\times n} = \underset{\sim}{\boldsymbol{U}}\underset{\sim}{\boldsymbol{\Sigma}}\underset{\sim}{\boldsymbol{V}}^\top$

Zerlegung in zwei Rotationen und eine Streckung:
$\underset{\sim}{\boldsymbol{U}} \in \mathbb{K}^{m\times m}$, $\underset{\sim}{\boldsymbol{V}} \in \mathbb{K}^{n\times n}$: orthonormale Rotationsmatrizen
$\underset{\sim}{\boldsymbol{\Sigma}} \in \mathbb{K}^{m\times n}$: $\underset{\sim}{\boldsymbol{\sigma}}$ ergänzt mit 0en damit $\dim \underset{\sim}{\boldsymbol{\Sigma}} = \dim \underset{\sim}{\boldsymbol{A}}$

Singulärwertszerlegung

- Bestimme n EW λ_i von $\underset{\sim}{\boldsymbol{A}}^\top \underset{\sim}{\boldsymbol{A}}$, sortiere $\lambda_1 \geq \ldots \geq \lambda_n \geq 0$
  Erhalten n Singulärwerte $\sigma_i = \sqrt{\lambda_i}$
- Bestimme ONB $\underset{\sim}{\boldsymbol{V}} = [\vec{v}_1, \ldots, \vec{v}_n]$ aus EV von $\underset{\sim}{\boldsymbol{A}}^\top \cdot \underset{\sim}{\boldsymbol{A}}$
- Bestimme ONB $\underset{\sim}{\boldsymbol{U}} = [\vec{u}_1, \ldots, \vec{u}_k]$ mit $\vec{u}_i = \frac{1}{\sigma_i} \underset{\sim}{\boldsymbol{A}} \vec{v}_i$
  $k = \min(m, n)$, falls $n < m$: Ergänze $\underset{\sim}{\boldsymbol{U}}$ zu ONB des $\mathbb{K}^m$
- Berechne $\underset{m\times n}{\underset{\sim}{\boldsymbol{\Sigma}}} = \underset{m\times m}{\underset{\sim}{\boldsymbol{U}}^\top} \cdot \underset{m\times n}{\underset{\sim}{\boldsymbol{A}}} \cdot \underset{n\times n}{\underset{\sim}{\boldsymbol{V}}}$ ($\underset{\sim}{\boldsymbol{U}}, \underset{\sim}{\boldsymbol{V}}$ sind orthogonal)

Stabilität: Falls Fehler ¡ $\kappa\sigma\epsilon$ und σ kleiner als ausgeführte Iterationen