Horner-Schema:

$$\text{Berechne } f(x)=\sum_{i=0}^{n} a_i \cdot x^i \qquad \text{mit} \qquad f(x)=(((a_n \cdot x+a_{n-1})\cdot x+a_{n-2})\ldots+a_1)\cdot x+a_0$$

Matrix für 2D-Laplace-System:

Matrix der Dimension n^2 x n^2; -4 auf der HD; 1 auf der n-ten ND; 1 auf der 1.ND, aber jeder n-te Wert ist 0; Bsp für n = 3;

$$A_3=\begin{pmatrix} -4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -4 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & -4 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & -4 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & -4 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & -4 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & -4 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -4 \end{pmatrix}$$

Lineare Filterung:

Faltung einer Folge, eines Bildes etc. mit einer Filterfolge bzw., Filtermaske. Diese nicht umdrehen!

2D-Filterung:

Komplexität bei Filter der Größe M x N:
nicht separierbar => O(M * N); separierbar => O(M + N);
Separierbare Filtermasken haben Rang 1!

FFT: (ich weiß, in SISY ist die anders^^)

Fourier-Transformation einer Folge der Länge n (=2^k). Es gilt $\omega_n=\sqrt[n]{e^{j2\pi}}$ (n-te Einheitswurzel)
Bsp für n = 8:

$$f=[1\,|\,2\,|\,3\,|\,4\,|\,5\,|\,6\,|\,7\,|\,8]$$

$$\downarrow$$

$$f^{(1)}=[f_0+f_4|f_1+f_5|f_2+f_6|f_3+f_7] \qquad f^{(2)}[f_0-f_4|(f_1-f_5)\cdot\omega_8|(f_2-f_6)\cdot\omega_8^2|(f_3-f_7)\cdot\omega_8^3]$$

$$f^{(1)}=[6\,|\,8\,|\,10\,|\,12] \qquad f^{(2)}=[-4\,|\,\frac{4+4\cdot j}{\sqrt{2}}\,|\,-4\cdot j\,|\,\frac{4-4\cdot j}{\sqrt{2}}] \quad \text{mit } \omega_8=\frac{1+j}{\sqrt{2}}$$

$$\downarrow$$

$$f^{(11)}=[f_0^1+f_2^1\,|\,f_1^1+f_3^1] \quad f^{(12)}=[f_0^1-f_2^1\,|\,(f_1^1-f_3^1)\cdot\omega_4] \quad f^{(21)} \text{und } f^{(22)} \text{ analog mit } \omega_4=j$$

$$f^{(11)}=[16\,|\,20] \quad f^{(12)}=[-4\,|\,-4\cdot j] \quad f^{(21)}=[-4-4\cdot j\,|\,\frac{8}{\sqrt{2}}] \quad f^{(22)}=[-4+4\cdot j\,|\,-\frac{8}{\sqrt{2}}]$$

$$\downarrow$$

$$g:=[f_1^{(11)}+f_2^{(11)}|f_1^{(11)}-f_2^{(11)}|f_1^{(12)}+f_2^{(12)}|f_1^{(12)}-f_2^{(12)}|f_1^{(21)}+f_2^{(21)}|f_1^{(21)}-f_2^{(21)}|f_1^{(22)}+f_2^{(22)}|f_1^{(22)}-f_2^{(22)}]$$

$$g=[36\,|\,-4\,|\,-4-4\cdot j\,|\,-4+4\cdot j\,|\,4\cdot(\sqrt{2}-1-j)\,|\,4\cdot(-\sqrt{2}-1-j)\,|\,4\cdot(-\sqrt{2}-1+j)\,|\,4\cdot(\sqrt{2}-1+j)]$$

$$\downarrow$$

Umsortieren durch Bit-reversed-indexing, *d.h.* $0=000_2\rightarrow 000_2=0$; $1=001_2\rightarrow 100_2=4$ *etc.*

$$\downarrow$$

$$\text{FFT}(f)=[g_0\,|\,g_4\,|\,g_2\,|\,g_6\,|\,g_1\,|\,g_5\,|\,g_3\,|\,g_7] =$$

$$[36\,|\,4\cdot(\sqrt{2}-1-j)\,|\,-4-4\cdot j\,|\,4\cdot(-\sqrt{2}-1+j)\,|\,-4\,|\,4\cdot(-\sqrt{2}-1-j)\,|\,-4+4\cdot j\,|\,4\cdot(\sqrt{2}-1+j)]$$

Komplexität $\in O(\frac{3}{2}\cdot n\cdot ld(n))$

CRS (Compressed Row Storage):

„value“ ist Folge der von 0 verschiedener Matrixelemente und „col-ind“ deren Spaltenindex.
„row-ptr(i)“ ist Index des ersten Elements in der Zeile i.
Es gilt row-ptr(n+1)=(Anzahl der von 0 verschiedenen Elemente in der Matrix) + 1 und
Anzahl der von 0 versch. Elemente der Zeile i = row-ptr(i+1) – row-ptr(i)
Bsp:

$$4\text{x}3-\text{Matrix } A=\begin{pmatrix}3 & 0 & 4\\ 0 & 0 & 0\\ 1 & 7 & 0\\ 0 & 2 & 6\end{pmatrix}$$

$$value=[3\,|\,4\,|\,1\,|\,7\,|\,2\,|\,6] \qquad col-ind=[1\,|\,3\,|\,1\,|\,2\,|\,2\,|\,3] \qquad row-ptr=[1\,|\,3\,|\,3\,|\,5\,|\,7]$$

CCS-Format ähnlich, nur mit „row-ind“ und „col-ptr“.
Es gilt CRS(A) = CCS(A^T)

Vektor-Normen:

$$\|\vec{x}\|_1=\sum_{i=1}^{n}|x_i| \quad (=\text{Summennorm / Manhattan-Norm}) \qquad \|\vec{x}\|_2=\sqrt{\sum_{i=1}^{n}x_i^2} \qquad \|\vec{x}\|_\infty=\max_i|x_i|$$

Matrix-Normen: A hat Dimension N x M

$$|||A|||_1=\max_j\sum_{i=1}^{n}|a_{i,j}| \quad (\text{max. Spaltensumme}) \qquad |||A|||_\infty=\max_i\sum_{j=1}^{m}|a_{i,j}| \quad (\text{max. Zeilensumme})$$

$$|||A|||_2=\sqrt{\max_i\lambda_i} \quad (\text{Spektralnorm: } \lambda_i \text{ sind Eigenwerte von } A^T\cdot A \text{ es gilt: } \lambda_i\geq 0)$$

Falls A symmetrisch ⇒
$|||A|||_2=$ Betrag des betragsmäßig größten Eigenwertes von A (= Spektralradius von A)

Bsp:

$$A=\begin{pmatrix}5 & 3\\ 2 & -7\\ 3 & 0\end{pmatrix} \qquad |||A|||_1=5+2+3=3+|-7|+0=10\,; \qquad |||A|||_\infty=2+|-7|=9\,;$$

$$B:=A^T\cdot A=\begin{pmatrix}38 & 1\\ 1 & 58\end{pmatrix} \qquad EW(B)\approx[37.9501\,|\,58.0499] \quad \Rightarrow \quad |||A|||_2\approx\sqrt{58.0499}\approx 7.619$$

Konditionszahlen:

Je größer die Konditionszahl einer Matrix desto anfälliger ist das Ergebnis auf Schwankungen in den Eingangsdaten.
Es gilt: $\kappa(A)=|||A|||\cdot|||A^{-1}|||$ (es kann jede beliebige Matrixnorm verwendet werden)
Wenn A symmetrisch ist, dann gilt für die Konditionszahl zur 2-Norm: $\kappa(A)=\left|\frac{\lambda_{max}(A)}{\lambda_{min}(A)}\right|$

Lösungsverfahren für LGS

- **iterative Verfahren:**

Für das LGS Ax = b wird in jedem Schritt eine immer bessere Näherungslösung bestimmt.
Diese Verfahren konvergieren nur für diagonal-dominante Matrizen, also muss gelten

$$|a_{i,i}| > \sum_{j=1, j\neq i}^{n} |a_{i,j}| \quad \forall\ i \quad \wedge \quad |a_{j,j}| > \sum_{i=1, i\neq j}^{n} |a_{i,j}| \quad \forall\ j$$

A wird aufgeteilt in A = L + D + R;

$$Bsp: A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \quad \Rightarrow \quad L = \begin{pmatrix} 0 & 0 & 0 \\ 4 & 0 & 0 \\ 7 & 8 & 0 \end{pmatrix} \quad D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{pmatrix} \quad R = \begin{pmatrix} 0 & 2 & 3 \\ 0 & 0 & 6 \\ 0 & 0 & 0 \end{pmatrix}$$

Der Start-Vektor x^0 wird frei gewählt und als Abbruch-Bedingung wird das Residuum im Schritt i ($r^i = b - Ax^i$) betrachtet. Falls die euklidische Norm des Residuums kleiner als eine bestimmte Schranke ist, wird das Ergebnis akzeptiert.

a) Jacobi-Verfahren:

Hier gilt: $\vec{x}^{i+1} = D^{-1} \cdot (\vec{b} - (L+R) \cdot \vec{x}^i) \quad \Rightarrow \quad Iterationsmatrix = -D^{-1} \cdot (L+R)$

oder für die Vektorkomponenten: $x_k^{i+1} = x_k^i + (b - \sum_{j=1}^{n} a_{k,j} \cdot x_j^i) \ / \ a_{k,k}$

b) Gauß-Seidel-Verfahren:

Hier gilt: $\vec{x}^{i+1} = (L+D)^{-1} \cdot (\vec{b} - R \cdot \vec{x}^i) \quad \Rightarrow \quad Iterationsmatrix = -(L+D)^{-1} \cdot R$

Für die Vektorkomponenten: $x_k^{i+1} = (b - \sum_{j=1}^{k-1} a_{k,j} \cdot x_j^{i+1} - \sum_{j=k+1}^{n} a_{k,j} \cdot x_j^i) \ / \ a_{k,k}$

Einfache Berechnung durch Verwendung eines einzelnen Vektors:

$$x_k = x_k + (b - \sum_{j=1}^{n} a_{k,j} \cdot x_j) \ / \ a_{k,k}$$

Dieses Verfahren konvergiert in etwa doppelt so schnell wie das Jacobi-Verfahren.

c) SOR-Verfahren:

Der Relaxationsparameter ω muss hier richtig gewählt werden

Hier gilt: $\vec{x}^{i+1} = (L + \frac{1}{\omega} \cdot D)^{-1} \cdot (\vec{b} - (R + (1 - \frac{1}{\omega}) \cdot D) \cdot \vec{x}^i) \quad \Rightarrow \quad R_{Iter} = -(L + \frac{1}{\omega} \cdot D)^{-1} \cdot (R + (1 - \frac{1}{\omega}) \cdot D)$

Für die Vektorkomponenten: $x_k^{i+1} = (1-\omega) \cdot x_k^i + \omega \cdot (b - \sum_{j=1}^{k-1} a_{k,j} \cdot x_j^{i+1} - \sum_{j=k+1}^{n} a_{k,j} \cdot x_j^i) \ / \ a_{k,k}$

Wenn der Spektralradius der Iterationsmatrix < 1 ist, dann konvergiert das Verfahren sicher.

- **direkte Verfahren:**

  Komplexität: Lösen eines LGS mit einer Dreiecksmatrix ist in $O(n^2/2)$

  a) LR-Zerlegung:

  Die Matrix A wird durch Gauss-Elimination so auf A = L*R zerlegt, dass R die Koeffizienten des LGS nach der Zerlegung enthält.
  Die untere Dreiecksmatrix enthält auf der Hauptdiagonalen 1-er und an den anderen Stellen die Koeffizienten mit denen die Zeilen bei der Elimination vor der Subtraktion multipliziert werden.

Bsp: $A=\begin{pmatrix} 1 & 2 & 0 \\ 2 & 2 & 8 \\ -3 & -2 & 12 \end{pmatrix} \Rightarrow \quad II-2\cdot I \Rightarrow l_{2,1}=2 \quad III-(-3)\cdot I \Rightarrow l_{3,1}=-3 \quad III+2\cdot II \Rightarrow l_{3,2}=-2$

$$L=\begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -3 & -2 & 1 \end{pmatrix} \quad R=\begin{pmatrix} 1 & 2 & 0 \\ 0 & -2 & 8 \\ 0 & 0 & 28 \end{pmatrix}$$

  Anschließend gilt Ax = LRx = b. => Löse erst Ly = b, dann Rx = y.

  Komplexität ist $O(n^3)$ für die Zerlegung und $O(n^2)$ fürs Lösen.

  b) QR-Zerlegung:

  Die Matrix A wird so in A=Q*R zerlegt, dass R eine obere Dreiecksmatrix und Q eine orthonormale Matrix ist (d.h. Spaltenvektoren sind paarweise orthogonal und haben die euklidische Norm 1).
  Diese Zerlegung lässt sich über sequentielle Householder-Spiegelungen oder Givens-Rotationen durchführen.

1.) Householder-Spiegelungen:

Es seien a_k die k-ten Spaltenvektoren der Matrix A. Bei dem k-ten Schritt dieses Verfahrens wird A mit einer Matrix H_k multipliziert, wodurch die k-te Spalte von A auf

den Vektor $\tilde{\vec{a}}_k = \|\vec{a}_k\|_2 \cdot \tilde{\vec{e}}_k = \frac{\|\vec{a}_k\|_2}{\sqrt{k}} \cdot \sum_{i=0}^{k} \vec{e}_k$, wobei $\vec{e}_k$ der k-te Einheitsvektor ist, abgebildet wird.

Hierbei gilt $H_k = E - 2 \cdot \vec{v}_k \cdot \vec{v}_k^T$ *wobei* $\vec{v}_k = \frac{\vec{a}_k - \lambda \cdot \tilde{\vec{e}}_k}{\|\vec{a}_k - \lambda \cdot \tilde{\vec{e}}_k\|_2}$ *mit* $\lambda = \|\vec{a}_k\|_2$

Bsp:

$$A = \begin{pmatrix} 1 & -1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 6 \end{pmatrix} \Rightarrow \tilde{\vec{e}}_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \ \|\vec{a}_1\| = 2 \Rightarrow \vec{v}_1 = \frac{\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} - 2 \cdot \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}}{\left\| \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} - 2 \cdot \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \right\|} = \frac{1}{2} \cdot \begin{pmatrix} -1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \Rightarrow H_1 = \frac{1}{2} \cdot \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}$$

$$A_1 = H_1 \cdot A = \begin{pmatrix} 2 & 5 \\ 0 & -4 \\ 0 & -3 \\ 0 & 0 \end{pmatrix} \Rightarrow \tilde{\vec{e}}_2 = \frac{1}{\sqrt{2}} \cdot \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \ \|\vec{a}_2\| = 5 \cdot \sqrt{2} \Rightarrow \vec{v}_2 = \frac{1}{3 \cdot \sqrt{10}} \begin{pmatrix} 0 \\ -9 \\ -3 \\ 0 \end{pmatrix} \Rightarrow H_2 = \frac{1}{5} \cdot \begin{pmatrix} 5 & 0 & 0 & 0 \\ 0 & -4 & -3 & 0 \\ 0 & -3 & 4 & 0 \\ 0 & 0 & 0 & 5 \end{pmatrix}$$

$$A_2 = H_2 \cdot A_1 = \underline{\begin{pmatrix} 2 & 5 \\ 0 & 5 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} =: R}$$

Die Matrizen H_k sind orthonormal und symmetrisch $\Rightarrow H_k^{-1} = H_k^T = H_k$

$$\Rightarrow R = H_2 \cdot H_1 \cdot A \Leftrightarrow A = (H_2 \cdot H_1)^{-1} \cdot R = H_1^{-1} \cdot H_2^{-1} \cdot R = H_1 \cdot H_2 \cdot R$$

$$\Rightarrow Q := H_1 \cdot H_2 = \underline{\frac{1}{10} \cdot \begin{pmatrix} 5 & -7 & 1 & 5 \\ 5 & -1 & -7 & -5 \\ 5 & 1 & 7 & -5 \\ 5 & 7 & -1 & 5 \end{pmatrix}}$$

$$\Rightarrow Q = \prod_{k=1}^{n} H_k \quad \wedge \quad R = Q^T \cdot A$$

2.) Givens-Rotationen (auch Jacobi-Rotationen):

Eine Z x S Matrix A wird sukzessive mit sog. Z x Z Drehmatrizen G_k multipliziert, wobei bei jedem Schritt ein Eintrag von A zu Null gesetzt wird. Soll der Eintrag in der i-ten Zeile und j Spalte von A, also $a_{i,j}$, genullt werden. So gilt für G, dass $G_{i,i} = G_{j,j} = c$, $G_{j,i} = s$ und $G_{i,j} = -s$. Die restlichen Einträge auf der HD von G sind 1 und alle anderen Einträge sind Null, d.h. Es werden die i-te und j-te Zeile von A verändert.

Es gilt: $c = \frac{a_{j,j}}{\sqrt{a_{j,j}^2 + a_{i,j}^2}} \quad \wedge \quad s = \frac{a_{i,j}}{\sqrt{a_{j,j}^2 + a_{i,j}^2}}$

Bsp:

$$A = \begin{pmatrix} 3 & 5 \\ 5 & 5 \\ 4 & 5 \end{pmatrix} \Rightarrow c_1 = \frac{3}{\sqrt{3^2+4^2}} = \frac{3}{5} \quad s_1 = \frac{4}{\sqrt{3^2+4^2}} = \frac{4}{5} \Rightarrow G_1 = \begin{pmatrix} \frac{3}{5} & 0 & \frac{4}{5} \\ 0 & 1 & 0 \\ \frac{-4}{5} & 0 & \frac{3}{5} \end{pmatrix} \Rightarrow A^{(1)} = G_1 \cdot A = \begin{pmatrix} 5 & 7 \\ 5 & 5 \\ 0 & -1 \end{pmatrix}$$

$$\Rightarrow c_2 = \frac{5}{\sqrt{5_2+5_2}} = \frac{1}{\sqrt{2}} \quad s_2 = \frac{5}{\sqrt{5_2+5_2}} = \frac{1}{\sqrt{2}} \Rightarrow G_2 = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{pmatrix} A^{(2)} = G_2 \cdot A^{(1)} = \begin{pmatrix} 5\cdot\sqrt{2} & 6\cdot\sqrt{2} \\ 0 & -\sqrt{2} \\ 0 & -1 \end{pmatrix}$$

$$\Rightarrow c_3 = \frac{-\sqrt{2}}{\sqrt{(-\sqrt{2})^2+(-1)^2}} = -\sqrt{\frac{2}{3}} \quad s_3 = \frac{-1}{\sqrt{(-\sqrt{2})^2+(-1)^2}} = -\frac{1}{\sqrt{3}} \Rightarrow G_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -\sqrt{\frac{2}{3}} & -\frac{1}{\sqrt{3}} \\ 0 & \frac{1}{\sqrt{3}} & -\sqrt{\frac{2}{3}} \end{pmatrix}$$

$$\Rightarrow \underline{R := A^{(3)} = G_3 \cdot A^{(2)} = \begin{pmatrix} 5\cdot\sqrt{2} & 6\cdot\sqrt{2} \\ 0 & \sqrt{3} \\ 0 & 0 \end{pmatrix}}$$

$$G_3 \cdot G_2 \cdot G_1 \cdot A = R \Leftrightarrow A = (G_3 \cdot G_2 \cdot G_1)^{-1} \cdot R = G_1^T \cdot G_2^T \cdot G_3^T \cdot R =: Q \cdot R$$

$$\Rightarrow \underline{Q = \prod_{k=1}^{3} G_k^T = \begin{pmatrix} \frac{3}{5\cdot\sqrt{2}} & \frac{7}{5\cdot\sqrt{3}} & \frac{1}{\sqrt{6}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{3}} & \frac{1}{\sqrt{6}} \\ \frac{2\cdot\sqrt{2}}{5} & \frac{1}{5\cdot\sqrt{3}} & -\sqrt{\frac{2}{3}} \end{pmatrix}}$$

Lösen überbestimmter LGS:

Besitz in dem LGS A*x=b die Matrix A mehr Zeilen als Spalten, so ist das LGS überbestimmt und es kann i.A. nur eine möglichst gute Näherungslösung berechnet werden.
Hierzu wird dann das LGS $A^T*A*x=A^T*b$ gelöst.

Bsp.: Ermittle eine Parabel, die folgende Bedingungen erfüllt:
f(-2) = f(2) = 3; f(0) = 2; f '(1) = 1; mit $f(x) = ax^2 + bx + c$ und $f'(x) = 2ax + b$;

$$\Rightarrow A\cdot\vec{x}=\begin{pmatrix}4&-2&1\\4&2&1\\0&0&1\\2&1&0\end{pmatrix}\cdot\begin{pmatrix}a\\b\\c\end{pmatrix}=\begin{pmatrix}3\\3\\2\\1\end{pmatrix}=\vec{b}\,;\ \Rightarrow\ \tilde{A}=A^T\cdot A=\begin{pmatrix}36&2&8\\2&9&0\\8&0&3\end{pmatrix},\quad \tilde{\vec{b}}=A^T\cdot\vec{b}=\begin{pmatrix}26\\1\\8\end{pmatrix}$$

$$\Rightarrow\quad \vec{x}=\tilde{A}^{-1}\cdot\tilde{\vec{b}}\approx\begin{pmatrix}0.3125\\0.0417\\1.8333\end{pmatrix}\quad\Rightarrow\quad bestes\ f(x)=0.3125x^2+0.0417x+1.8333$$

Interpolationsverfahren

a) lokale Verfahren in 1D

- **Stückweise konstante Interpolation (Nearest Neighbour)**

  Die Funktion f(x) wird zwischen je zwei benachbarten Stützstellen x_i & x_{i+1} so interpoliert, dass f(x) den Wert bei der Stützstelle hat, die x am nächsten liegt.
  Bsp.: f(1) = 2, f(2) = 3; => f(1,49) = f(1) = 2; f(1,5) = f(2) = 3;

- **Stückweise lineare Interpolation**

  Die Funktion f(x) wird zwischen je zwei benachbarten Stützstellen x_i & x_{i+1} durch Streckenzüge $g_i(x)$ interpoliert. Es gilt:

  $$g_i(x)=\frac{f(x_i)\cdot(x_{i+1}-x)+f(x_{i+1})\cdot(x-x_i)}{x_{i+1}-x_i}=:(1-\alpha)\cdot f(x_i)+\alpha\cdot f(x_{i+1})\qquad mit\ \alpha=\frac{x-x_i}{x_{i+1}-x_i}$$

  Bsp: $f(1)=5,\ f(2)=3\ \Rightarrow\ \tilde{f}(x)=\frac{5\cdot(2-x)+3\cdot(x-1)}{2-1}=-2\cdot x+7$

- **Catmull-Rom Interpolation**

  Die Funktion f(x) wird zwischen je zwei benachbarten Stützstellen x_i & x_{i+1} durch kubische Polynome folgender Form interpoliert:

  $$p_i(x)=a_0\cdot(x_{i+1}-x)^3+a_1\cdot(x_{i+1}-x)^2\cdot(x-x_i)+a_2\cdot(x_{i+1}-x)^1\cdot(x-x_i)^2+a_3\cdot(x-x_i)^3=$$
  $$=\sum_{k=0}^{3}a_k\cdot(x_{i+1}-x)^{3-k}\cdot(x-x_i)^k\qquad wobei\ gilt$$

  $$a_0=\frac{y_i}{(x_{i+1}-x_i)^3};\quad a_1=3\cdot a_0+\frac{y_i'}{(x_{i+1}-x_i)^2};\quad a_2=3\cdot a_3-\frac{y_{i+1}'}{(x_{i+1}-x_i)^2};\quad a_3=\frac{y_{i+1}}{(x_{i+1}-x_i)^3};$$

  D.h. dass in jedem Punkt die Steigungen geschätzt werden müssen, dafür gibt es 3 Methoden:

  $y_i'=\frac{y_{i+1}-y_i}{x_{i+1}-x_i}$ (vorwärts) $\quad y_i'=\frac{y_i-y_{i-1}}{x_i-x_{i-1}}$ (rückwärts) $\quad y_i'=\frac{y_{i+1}-y_{i-1}}{x_{i+1}-x_{i-1}}$ (zentral)

b) globale Verfahren in 1D

- **Polynom-Interpolation**

  Eine Funktion mit n Bedingungen kann durch ein Polynom vom Grad n-1 interpoliert oder durch eins mit niedrigerem Grad approximiert werden. Verwendet man als Bedingungen nur die Stützstellen spricht man von „Lagrange"-Interpolation und bei zusätzlicher Verwendung der Ableitungen von „Hermite"-Interpolation.

  1.) Polynom-Interpolation mit Vandermonde-Matrix

  Zur Bestimmung der Koeffizienten von $p(x)=\sum_{i=0}^{n-1} c_i \cdot x^i$ wird folgendes LGS gelöst:

$$\begin{pmatrix} 1 & x_1 & \cdots & x_1^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \cdots & x_n^{n-1} \end{pmatrix} \cdot \begin{pmatrix} c_0 \\ \vdots \\ c_{n-1} \end{pmatrix} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$$

  Komplexität in $O(n^3)$

  2.) Polynom-Interpolation mit Lagrange-Basis

  Hier wird das Polynom durch $p(x)=\sum_{k=1}^{n} y_k \cdot L_k(x)$ ausgedrückt, wobei für die sog. Lagrange-Polynome definiert sind durch: $L_k(x)=\prod_{i=1, i\neq k}^{n} \frac{x-x_i}{x_k-x_i}$

  Bsp.: Berechne das Interpolations-Polynom zu den Punkten $(1|6),\ (2|4),\ (4|12)$

$$p(x) = 6 \cdot \frac{(x-2)\cdot(x-4)}{(1-2)\cdot(1-4)} + 4 \cdot \frac{(x-1)\cdot(x-4)}{(2-1)\cdot(2-4)} + 12 \cdot \frac{(x-1)\cdot(x-2)}{(4-1)\cdot(4-2)} =$$

$$\underline{2\cdot(x-2)\cdot(x-4) - 2\cdot(x-1)\cdot(x-4) + 2\cdot(x-1)\cdot(x-2)}$$

  Komplexität in O(n)

  3.) Polynom-Interpolation mit Newton-Basis

  Hier wird das Polynom durch $p(x)=\sum_{k=0}^{n-1} c_k \cdot q_k(x)$ ausgedrückt, wobei für die sog. Newton-Polynome definiert sind durch: $q_k(x)=\prod_{i=1, i\neq k}^{n} (x-x_i)$

  Die Koeffizienten werden über den Aitken-Neville-Algorithmus berechnet:

  Es gilt: $a_{i,0} := y_i \qquad c_k = a_{1,k} \qquad a_{i,k} = \frac{a_{i+1,k-1} - a_{i,k-1}}{x_{i+k} - x_i}$

  Bsp: Gleiche Werte wie oben $\vec{a}_k := [a_{1,k},\ \cdots,\ a_{n-1,k}]^T$

- $\vec{a}_0 := \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} \mathbf{6} \\ 4 \\ 12 \end{pmatrix} \rightarrow \vec{a}_1 = \begin{pmatrix} \frac{4-6}{2-1} \\ \frac{12-4}{4-2} \\ 0 \end{pmatrix} = \begin{pmatrix} \mathbf{-2} \\ 4 \\ 0 \end{pmatrix} \rightarrow \vec{a}_2 = \begin{pmatrix} \frac{4-(-2)}{4-1} \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} \mathbf{2} \\ 0 \\ 0 \end{pmatrix} \rightarrow \vec{c} = \begin{pmatrix} 6 \\ -2 \\ 2 \end{pmatrix}$

$$\Rightarrow \underline{p(x) = 6 - 2\cdot(x-1) + 2\cdot(x-1)\cdot(x-2)}$$

  Komplexität in $O(n^2)$

c) Interpolation im Mehrdimensionalen

- **Multi-Lineare Interpolation**

Bei diesem Verfahren wird sukzessive in jede Raumrichtung linear interpoliert. Dabei muss insgesamt $2^N - 1$ mal interpoliert werden (2^{N-k} Interpolationen im k-ten Schritt). Die verwendeten Stützstellen bilden einen regelmäßigen, rechtwinkligen N-dimensionalen Körper (2D => Rechteck, 3D => Quader etc.).

Bsp.: Ges: Interpolant $\tilde{f}(x,y)$ zu den Punkten $A=(1|1),\ B=(3|1),\ C=(3|4)$ *und* $D(1|4)$ mit $f(A)=1,\ f(B)=3,\ f(C)=-2$ *und* $f(D)=2$;

$$\alpha_x(x) := \frac{x-x_A}{x_B-x_A} = \frac{x-1}{2};\quad P_{AB}(x) := (1-\alpha)\cdot f(A)+\alpha\cdot f(B) = (1-\frac{(x-1)}{2})\cdot 1+\frac{(x-1)}{2}\cdot 3 = x;$$

$$P_{DC}(x) := (1-\alpha)\cdot f(D)+\alpha\cdot f(C) = (1-\frac{(x-1)}{2})\cdot 2+\frac{(x-1)}{2}\cdot(-2) = -2\cdot x+4;$$

$$\alpha_y = \frac{y-y_A}{y_D-y_A} = \frac{y-1}{3};\Rightarrow\ \tilde{f}(x,y) = (1-\alpha_y)\cdot P_{AB}(x)+\alpha_y\cdot P_{DC}(x) =$$

$$= (1-\frac{y-1}{3})\cdot x+\frac{y-1}{3}\cdot(-2\cdot x+4) = \underline{-xy+2\cdot x+\frac{4}{3}\cdot y-\frac{4}{3}};$$

- **Lineare Interpolation mit baryzentrischen Koordinaten**

Für die Darstellung eines Punkts X mit N kartesischen Koordinaten x_k im baryzentrischen Raum werden N + 1 Bezugspunkte P_i und damit baryzentrische Koordinaten σ_k benötigt.
Sind die baryz. Koordinaten so normiert, dass ihre Summe gleich 1 (und nicht -1!) ist, dann gilt mit K_k := „der Körper, der durch alle P_i ohne P_k aufgespannt wird“, dass σ_k gleich dem Abstand von X zu K_k durch den Abstand von P_k zu K_k ist => $\underline{d(X, K_k) = \sigma_k * d(P, K_k).}$

Damit folgt auch: Ist $\sigma_k > 0$, dann liegen P_k und X auf der selben Seite von K_k !

Bsp.: 1.) A, B, C und D aus obigem Beispiel => D hat bzgl. A,B,C die baryz. Koord. (1,-1,1), da gilt:
d(D, BC) = d(A, BC), d(D, AC) = (-1) * d(B, AC) und d(D, AB) = d(C, AB).

2.) Der Ursprung hätte bzgl. A,B,D die baryz. Koordinaten $\left(\frac{11}{6}|-\frac{1}{2}|-\frac{1}{3}\right)$

Für die Interpolation gilt dann $f(X) = \sum_{k=1}^{N+1} \sigma_k \cdot f(P_k)$

Damit folgt in diesem Beispiel $f((0|0)) = \frac{11}{6}\cdot f(A)-\frac{1}{2}\cdot f(B)-\frac{1}{3}\cdot f(D) = -\frac{1}{3}$

Bezier-Kurven

Bezierkurven sind Funktionen, die das Intervall [0; 1] auf eine Kurve im mehrdimensionalen Raum abbilden. Das Aussehen einer Bezierkurve vom Grad N wird durch N+1 Kontrollpunkte festgelegt.

Sie ist definiert als $\vec{f}(t)=\sum_{i=0}^{n}\vec{b}_i\cdot B_i^n(t)$ *mit* $B_i^n(t)=\binom{n}{i}\cdot(1-t)^{n-i}\cdot t^i=\frac{n!}{(n-i)!\cdot i!}\cdot(1-t)^{n-i}\cdot t^i$

Für die Bernstein-Polynome gilt

1) $B_0^n(0)=B_n^n(1)=1 \quad \wedge \quad B_i^n(0)=B_i^n(1)=0 \quad \forall\, 1\le i\le n-1$
2) $B_i^n(t)=0 \quad \forall\,(i<0)\vee(i>n)$
3) $\frac{d}{dt}B_i^n(t)=n\cdot(B_{i-1}^{n-1}(t)-B_i^{n-1}(t))$

Bezierkurven haben folgende Eigenschaften:

1) Der erste und letzte Kontrollpunkt liegt auf der Kurve und dort verläuft die Bezierkurve tangential zum Kontrollpolygon
2) Die Kurve liegt in der konvexen Hülle des Kontrollpolygons
3) Die Kurve schneidet die Gerade vom ersten zum letzten Kontrollpunkt maximal so oft wie das Kontrollpolygon.
4) Die Ableitung einer Bezierkurve vom Grad N ist eine Bezierkurve vom Grad N – 1 mit den Kontrollpunkte $B_i' = n * (B_{i+1} – B_i)$.

De Casteljau – Algorithmus

Zur Auswertung einer Bezierkurve am Parameterwert t kann an Stelle der Berechnung der Bernstein-Polynome der „de Casteljau“-Algorithmus angewendet werden. Hierbei gilt:

$\vec{c}_k^{\,0}=\vec{b}_k \qquad \vec{c}_k^{\,i}=(1-t)\cdot\vec{c}_{k-1}^{\,i-1}+t\cdot\vec{c}_k^{\,i-1} \qquad \underline{\vec{f}(t)=\vec{c}_n^{\,n}}$

Bsp: Bezierkurve mit den Kontrollpunkten $\vec{b}_0=(1|1)$, $\vec{b}_1=(3|1)$, $\vec{b}_2=(3|4)$ und $\vec{b}_3=(1|4)$ an der Stelle $t=\frac{1}{3.}$

$$C^0:=\begin{pmatrix}\vec{c}_0^{\,0T}\\ \vec{c}_1^{\,0T}\\ \vec{c}_2^{\,0T}\\ \vec{c}_3^{\,0T}\end{pmatrix}=\begin{pmatrix}1 & 1\\ 3 & 1\\ 3 & 4\\ 1 & 4\end{pmatrix} \rightarrow C^1=\begin{pmatrix}0 & 0\\ \frac{5}{3} & 1\\ 3 & 2\\ \frac{7}{3} & 4\end{pmatrix} \rightarrow C^2=\begin{pmatrix}0 & 0\\ 0 & 0\\ \frac{19}{9} & \frac{4}{3}\\ \frac{25}{9} & \frac{8}{3}\end{pmatrix} \rightarrow C^3=\begin{pmatrix}0 & 0\\ 0 & 0\\ 0 & 0\\ \frac{7}{3} & \frac{16}{9}\end{pmatrix} \Rightarrow \underline{\vec{f}(\frac{1}{3})=\begin{pmatrix}\frac{7}{3}\\ \frac{16}{9}\end{pmatrix}}$$

Dieser Algorithmus lässt sich auch graphisch anwenden, indem man die Verbindungsstrecken der Kontrollpunkte im Verhältnis t teilt und die entstehenden Punkte wieder verbindet. Dies macht man solange bis nur ein Punkt übrig ist, was dann der Funktionswert ist.

Bezier-Flächen

Bezierflächen sind definiert als $\vec{f}(s,t)=\sum_{i=0}^{n}\sum_{k=0}^{m}\vec{b}_{ik}\cdot B_i^n(s)\cdot B_k^m(t)$. Setzt man einen Wert für s ein, so erhält man eine Bezierkurve mit Parameter t (und umgekehrt).
Demnach kann man Bezierflächen durch doppelte Ausführung des „de Casteljau"-Algorithmus auswerten.

Coons-Patch

Sind die 4 Randkurven $C_0(s)$, $C_1(s)$, $D_0(t)$ und $D_1(t)$ mit $C_0(1) = D_1(0)$, $D_1(1) = C_1(1)$, $C_1(0) = D_0(1)$ und $D_0(0) = C_0(0)$ gegeben, dann ist das Coons-Patch ein 2D-Interpolant dieses Bereiches, der exakt mit dem Rand übereinstimmt.
Er ist definiert als $x(s,t):=x_t(s,t)+x_s(s,t)-x_{st}(s,t)$, wobei gilt:

$$x_t(s,t)=(1-s)\cdot x(0,t)+s\cdot(1,t) \qquad x_s(s,t)=(1-t)\cdot x(s,0)+t\cdot(s,1)$$
$$x_{st}(s,t):=\text{"bilinearer Interpolant der 4 Eckpunkte } x(0,0),\ x(1,0),\ x(0,1) \text{ und } x(1,1)\text{"}$$

Bsp:

$$C_0(s)=\begin{pmatrix}s\\0\\s^2\end{pmatrix},\quad C_1(s)=\begin{pmatrix}s\\1\\1-s\end{pmatrix},\quad D_0(t)=\begin{pmatrix}0\\t\\t^2\end{pmatrix},\quad D_1(t)=\begin{pmatrix}1\\t\\1-t\end{pmatrix}$$

$$x_t(s,t)=(1-s)\cdot D_0(t)+s\cdot D_1(t)=\begin{pmatrix}s\\t\\s+t^2-st-st^2\end{pmatrix},\ x_s(s,t)=(1-t)\cdot C_0(s)+t\cdot C_1(s)=\begin{pmatrix}s\\t\\s^2+t-st-s^2t\end{pmatrix}$$

$$x_{st}(s,t)=(1-t)\cdot[(1-s)\cdot C_0(0)+s\cdot C_0(1)]\ +\ t\cdot[(1-s)\cdot C_1(0)+s\cdot C_1(1)]=$$

$$=(1-t)\cdot\begin{pmatrix}s\\0\\s\end{pmatrix}\ +\ t\cdot\begin{pmatrix}s\\1\\1-s\end{pmatrix}=\begin{pmatrix}s\\t\\s+t-2\cdot st\end{pmatrix}\ \Rightarrow\ \underline{x(s,t)=\begin{pmatrix}s\\t\\s^2-s^2t-st^2+t^2\end{pmatrix}}$$

Numerische Integration

Man kann das Integral einer Funktion für n+1 äquidistante Stützstellen mittels eines Polynoms vom Grad n approximieren. Dies geschieht mit den sog. „Newton-Cotes“-Formeln:

$h := b-a$ $\quad$ $\epsilon :=$ größtmöglicher Fehler der Abschätzung $\quad$ $a \le \xi \le b$

n=0 $\Rightarrow \int_a^b f(x)\,dx \approx h \cdot f\left(\frac{a+b}{2}\right)$ (Mittelpunktsregel) $\Rightarrow \epsilon \le \frac{h^3}{24} \cdot max\left|f^{(2)}(\xi)\right|$

n=1 $\Rightarrow \int_a^b f(x)\,dx \approx \frac{h}{2} \cdot (f(a)+f(b))$ (Trapezregel) $\Rightarrow \epsilon \le \frac{h^3}{12} \cdot max\left|f^{(2)}(\xi)\right|$

n=2 $\Rightarrow \int_a^b f(x)\,dx \approx \frac{h}{6} \cdot \left(f(a)+4 \cdot f\left(\frac{a+b}{2}\right)+f(b)\right)$ (Simpson-Regel) $\Rightarrow \epsilon \le \frac{\left(\frac{h}{2}\right)^5}{180} \cdot max\left|f^{(4)}(\xi)\right|$

n=3 $\Rightarrow \int_a^b f(x)\,dx \approx \frac{h}{8} \cdot (f(a)+3 \cdot f(x_1)+3 \cdot f(x_2)+f(b))$ (3/8-Regel) $\Rightarrow \epsilon \le \frac{3 \cdot \left(\frac{h}{3}\right)^5}{80} \cdot max\left|f^{(4)}(\xi)\right|$

Die Fehlerabschätzungen für die Simpson-Regel ist zwar falsch, aber so kann man sich wenigstens aufs Skript berufen.....

Bsp: Berechne $\int_0^2 e^x\,dx \Rightarrow h=2$ $\quad$ Def.: $N^{(i)} :=$ Newton-Cotes-Formel vom Grad i

$N^{(0)} \approx 2 \cdot e^1 \approx 5{,}43656$ $\quad$ $N^{(1)} \approx \frac{2}{2} \cdot (e^0+e^2) \approx 8{,}38906$

$N^{(2)} \approx \frac{2}{6} \cdot (e^0+4 \cdot e^1+e^2) \approx 6{,}42073$ $\quad$ $N^{(3)} \approx \frac{2}{8} \cdot (e^0+3 \cdot e^{\frac{2}{3}}+3 \cdot e^{\frac{4}{3}}+e^2) \approx 6{,}40332$

Wenn mehr Stützstellen bekannt sind, dann ist es besser iterierte als höhergradige Newton-Cotes-Formeln zu verwenden, d.h. man stellt das Integral als Summe von Newton-Cotes-Formeln dar, deren Randpunkte im Inneren doppelt gewichtet sind (sie sind sowohl Anfang- als auch Endpunkt eines Integrals).

$h := \frac{b-a}{n}$ $\quad$ $n+1 =$ Zahl der Stützstellen $\quad$ $a \le \xi \le b$

(iter. Trapez-R.) $\Rightarrow \int_a^b f(x)\,dx \approx T^h_{[a,b]} = \frac{h}{2} \cdot \left(f(a)+2 \cdot \sum_{i=1}^{n-1} f(a+i \cdot h)+f(b)\right) \Rightarrow \epsilon \le \frac{h^2}{12} \cdot max\left|f^{(2)}(\xi)\right|$

(iter. Simpson-R.) $\Rightarrow \int_a^b f(x)\,dx \approx S^h_{[a,b]} = \frac{h}{3} \cdot \left(f(a)+4 \cdot \sum_{i=1}^{\frac{n}{2}} f(a+(2 \cdot i-1) \cdot h)+2 \cdot \sum_{i=1}^{\frac{n}{2}-1} f(a+2 \cdot i \cdot h)+f(b)\right)$

$\Rightarrow \epsilon \le \frac{h^4}{180} \cdot max\left|f^{(4)}(\xi)\right|$ $\quad$ Beachte: Bei der iterierten Simpson-Regel muss (n+1) ungerade sein

Bsp: Berechne $\int_0^2 e^x\,dx$ mit Schrittweite $h=0.5$

(Trapezsumme) $\quad T^{0.5}_{[0,2]}(e^x) = \frac{0.5}{2} \cdot \left(e^0+2 \cdot (e^{0.5}+e^1+e^{1.5})+e^2\right) \approx 6{,}52161$

(Simpsonsumme) $\quad S^{0.5}_{[0,2]}(e^x) = \frac{0.5}{3} \cdot \left(e^0+4 \cdot e^{0.5}+2 \cdot e^1+4 \cdot e^{1.5}+e^2\right) \approx 6{,}39121$

Romberg-Quadratur-Verfahren

Mit diesem Verfahren lässt sich der Fehler einer Integral-Berechnung verkleinern, wenn die Trapezsummen zu bestimmten Schrittweiten bekannt sind. Für dieses Verfahren gilt:

$$h_j = \frac{b-a}{2^k} \qquad \int_a^b f(x)dx = R(n,n) \text{ bei n Iterationschritten}$$

$$R(j,0) = T^{h_j}_{[a,b]}(f(x)) \qquad R(j,k) = \frac{4^k \cdot R(j,k-1) - R(j-1,k-1)}{4^k - 1}$$

Bsp.: Berechne $\int_0^2 e^x\,dx$ mit minimaler Schrittweite $h = 0.5$ *Def.:* $\vec{r}^{(k)} = [R(0,k), \cdots, R(n,k)]^T$

$$h_0 = 2; \qquad h_1 = 1; \qquad h_2 = 0.5; \qquad R(0,0) = T^2_{[0,2]}(e^x) = \frac{2}{2} \cdot \left(e^0 + e^2\right) \approx 8{,}38906;$$

$$R(1,0) = T^1_{[0,2]}(e^x) = \frac{1}{2} \cdot \left(e^0 + 2 \cdot e^1 + e^2\right) \approx 6{,}91281; \qquad R(2,0) = T^{0.5}_{[0,2]}(e^x) = s.\,oben \approx 6{,}52161;$$

$$\Rightarrow \vec{r}^{(0)} = \begin{pmatrix} 8{,}38906 \\ 6{,}91281 \\ 6{,}52161 \end{pmatrix} \Rightarrow \vec{r}^{(1)} = \begin{pmatrix} 0 \\ \frac{4 \cdot 6{,}91281 - 8{,}38906}{4-1} \\ \frac{4 \cdot 6{,}52161 - 6{,}91281}{4-1} \end{pmatrix} = \begin{pmatrix} 0 \\ 6{,}42073 \\ 6{,}39121 \end{pmatrix} \Rightarrow \vec{r}^{(2)} = \begin{pmatrix} 0 \\ 0 \\ \frac{16 \cdot 6{,}39121 - 6{,}42073}{16-1} \end{pmatrix};$$

$$\int_0^2 e^x\,dx \approx R(2,2) = r_2^{(2)} \approx 6{,}38924$$

Singulärwert-Zerlegung – SVD (Singular Value Decomposition)

Eine beliebige Z x S Matrix A wird durch die SVD auf die Form A = U * S * V^T gebracht. Hierbei ist S eine Z x S Matrix, auf deren Hauptdiagonale die (absteigend sortierten) Singulärwerte von A stehen.

Für den i-ten Singulärwert von gilt $\sigma_i = \sqrt{\lambda_i}$, mit $\lambda_i = EW_i\{A \cdot A^T\} = EW_i\{A^T \cdot A\}$.

Die Spalten von U sind die **normierten** Eigenvektoren von A*A^T und die Spalten von V die **normierten** Eigenvektoren von A^T*A.

$$\text{Bsp: } A = \begin{pmatrix} 2 & 3 \\ 1 & -6 \end{pmatrix} \Rightarrow A \cdot A^T = \begin{pmatrix} 13 & -16 \\ -16 & 37 \end{pmatrix} \wedge A^T \cdot A = \begin{pmatrix} 5 & 0 \\ 0 & 45 \end{pmatrix} \Rightarrow \lambda_1 = 45 \;\; \lambda_2 = 5$$

$$\Rightarrow S = \begin{pmatrix} \sqrt{45} & 0 \\ 0 & \sqrt{5} \end{pmatrix} = \begin{pmatrix} 3\cdot\sqrt{5} & 0 \\ 0 & \sqrt{5} \end{pmatrix}; \quad \textit{Berechnung der EVen} \Rightarrow U = \frac{1}{\sqrt{5}} \cdot \begin{pmatrix} 1 & 2 \\ -2 & 1 \end{pmatrix} \wedge V = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

$$\Rightarrow A = U \cdot S \cdot V^T = \frac{1}{\sqrt{5}} \cdot \begin{pmatrix} 1 & 2 \\ -2 & 1 \end{pmatrix} \cdot \begin{pmatrix} 3\cdot\sqrt{5} & 0 \\ 0 & \sqrt{5} \end{pmatrix} \cdot \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}^T$$

Vertauscht man die Reihenfolge der Singulärwerte, so muss man auch zugehörigen Spaltenvektoren von U und V vertauschen:

$$A = \frac{1}{\sqrt{5}} \cdot \begin{pmatrix} 1 & 2 \\ -2 & 1 \end{pmatrix} \cdot \begin{pmatrix} 3\cdot\sqrt{5} & 0 \\ 0 & \sqrt{5} \end{pmatrix} \cdot \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}^T = \frac{1}{\sqrt{5}} \cdot \begin{pmatrix} 2 & 1 \\ 1 & -2 \end{pmatrix} \cdot \begin{pmatrix} \sqrt{5} & 0 \\ 0 & 3\cdot\sqrt{5} \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}^T$$

Rang, Kern und Bild

Der Rang einer Matrix entspricht der Zahl **ihrer von Null verschiedenen** Singulärwerte.

Das Bild einer Matrix ist der kleinste Vektorraum, in dem alle $\{\vec{b} \mid \vec{b} = A \cdot \vec{x}\}$ liegen, also alle möglichen **Ergebnisse** von A * x.
Das Bild von A besteht aus den Spalten von U, die zu von Null versch. Singulärwerten gehören.

Der Kern einer Matrix ist der Vektorraum, in dem alle $\{\vec{x} \mid A \cdot \vec{x} = \vec{0}\}$ liegen, also der Vektorraum der durch A * x auf den Nullvektor abgebildet wird. *Beachte*: Eine Z x S Matrix vom Rang S hat einen leeren Kern.

Der Kern von A besteht aus den Spalten von V, die **nicht** zu von Null versch. Singulärwerten gehören.

$$\text{Bsp: } A = U \cdot S \cdot V^T = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix} \cdot \begin{pmatrix} 5 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \end{pmatrix}^T$$

$$\Rightarrow Rang(A) = 2 \quad \wedge \quad Bild(A) = \left\{ \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}; \; \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \right\} \quad \wedge \quad Kern(A) = \left\{ \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}; \; \begin{pmatrix} 0 \\ 0 \\ -1 \\ 0 \end{pmatrix} \right\}$$

Lösen singulärer LGS

LGS mit singulären Matrizen kann man über Multiplikation mit der Pseudoinversen loesen:

$$A\cdot\vec{x}=(U\cdot S\cdot V^T)\cdot\vec{x}=\vec{b} \quad \rightarrow \quad \vec{x}=A^{\sim 1}\cdot\vec{b}=(V\cdot S^{\sim 1}\cdot U^T)\cdot\vec{b}$$

Gehört b zum Bild von A, dann wird das LGS exakt gelöst, ansonsten wird x so bestimmt, dass gilt:

$\left\|A\cdot\vec{x}-\vec{b}\right\|_2\rightarrow min$. In beiden Fällen gilt: $\left\|\vec{x}\right\|_2=min$.

Ist nach allen Vektoren x gefragt, die dieses LGS lösen, dann müssen alle Vektoren, die zum Kern gehören, mit Variablen gewichtet, hinzuaddiert werden.

Fuer $S^{\sim 1}$ gilt: Wenn A eine Z x S Matrix ist, dann ist $S^{\sim 1}$ eine S x Z Matrix, die auf der Hauptdiagonalen die Kehrwerte der von Null verschiedenen Singulärwerte hat.

Bsp: $S=\begin{pmatrix}3&0&0\\0&2&0\\0&0&0\\0&0&0\end{pmatrix} \Rightarrow S^{\sim 1}=\begin{pmatrix}\frac{1}{3}&0&0&0\\0&\frac{1}{2}&0&0\\0&0&0&0\end{pmatrix}$

Hauptachsentransformation – PCA (Principal Component Analysis)

Die Hauptachsentransformierte $C\in\mathbb{R}^{Z\times Z}$ einer Z x S Matrix A lässt sich leicht über deren SVD bestimmen. Es gilt:

$$C=\frac{1}{(S-1)}\cdot A\cdot A^T=\frac{1}{(S-1)}\cdot(U\cdot S\cdot V^T)\cdot(U\cdot S\cdot V^T)^T=\frac{1}{(S-1)}\cdot U\cdot S\cdot V^T\cdot V\cdot S^T\cdot U^T=\underline{\frac{1}{(S-1)}\cdot U\cdot S\cdot S^T\cdot U^T}$$

Die Spaltenvektoren der Matrix U werden Hauptachsen genannt.

Bsp: *genaue Rechnung siehe SVD*

$$A=\begin{pmatrix}2&3\\1&-6\end{pmatrix}=\frac{1}{\sqrt{5}}\cdot\begin{pmatrix}1&2\\-2&1\end{pmatrix}\cdot\begin{pmatrix}3\cdot\sqrt{5}&0\\0&\sqrt{5}\end{pmatrix}\cdot\begin{pmatrix}0&1\\1&0\end{pmatrix}^T=:U\cdot S\cdot V^T$$

$$C=\frac{1}{(2-1)}\cdot A\cdot A^T=U\cdot S\cdot S^T\cdot U^T=\frac{1}{\sqrt{5}}\cdot\begin{pmatrix}1&2\\-2&1\end{pmatrix}\cdot\begin{pmatrix}3\cdot\sqrt{5}&0\\0&\sqrt{5}\end{pmatrix}\cdot\begin{pmatrix}3\cdot\sqrt{5}&0\\0&\sqrt{5}\end{pmatrix}\cdot\frac{1}{\sqrt{5}}\cdot\begin{pmatrix}1&-2\\2&1\end{pmatrix}=$$

$$\underline{\frac{1}{\sqrt{5}}\cdot\begin{pmatrix}1&2\\-2&1\end{pmatrix}\cdot\begin{pmatrix}45&0\\0&5\end{pmatrix}\cdot\begin{pmatrix}1&-2\\2&1\end{pmatrix}\cdot\frac{1}{\sqrt{5}}}$$

Optimierung

Bei der Optimierung eines Problems F(x) gilt es, das Minimum zu finden. Die Probleme können linear und nicht linear sein und eventuell Nebenbedingungen haben.

Exakte Lösung nicht linearer Optimierungsprobleme mit Nebenbedingungen

Hier werden die Nebenbedingungen mit sog. Lagrange-Multiplikatoren multipliziert und zu F(x) hinzuaddiert. Von der so entstanden Funktion wird der Gradient gleich Null gesetzt und das entstehende Gleichungssystem gelöst.
Dieses Verfahren wird im Skript "Mathematik für Ingenieure A2" im Kapitel „Restringierte Optimierung" sehr gut erklärt.

Numerische Lösung nicht linearer Optimierungsprobleme ohne Nebenbedingungen

Gesucht ist ein lokales Minimum eines nicht-linearen (im Allgemeinen vektorwertigen) Problems, z.B.:
$F(\vec{x}) = \vec{x}^T \cdot A \cdot \vec{x} + 2 \cdot \vec{b}_T \cdot \vec{x} + c \rightarrow min \qquad (Beachte: \ \nabla F(\vec{x}) = 2 \cdot A \cdot \vec{x} + 2 \cdot \vec{b})$

Für die numerische Lösung gibt es drei Verfahren:

1.) Gradientenverfahren – **iterativ**

Ausgehend von einer Startlösung x_0 zieht man solange den gewichteten Gradienten im aktuellen Punkt ab, bis der Gradient genügend nahe an 0 ist.
Beachte: Man muss immer vom Gradienten weg gehen!

$$\Rightarrow \ \vec{s}^{(k)} := \nabla F(\vec{x}^{(k)}) \qquad \vec{x}^{(k+1)} = \vec{x}^{(k)} - t \cdot \vec{s}^{(k)} \qquad \text{mit} \ \ t = \underset{t}{argmin} \{ F(\vec{x}^{(k)} - t \cdot \vec{s}^{(k)}) \} > 0$$

Abbruch bei $\| \nabla F(\vec{x}^{(k+1)}) \|_2 < \epsilon$

Bsp.: $F(\vec{x}) = \vec{x}^T \cdot A \cdot \vec{x} + 2 \cdot \vec{b}^T \cdot \vec{x} + c \qquad mit \qquad A = \begin{pmatrix} 1 & 1 \\ 1 & 9 \end{pmatrix}, \ \vec{b} = \begin{pmatrix} -10 \\ -10 \end{pmatrix}, \ c = 100$

1) *Bestimme* $t(\vec{x})$*:*

$$\nabla F(\vec{x} - t \cdot \vec{s}) = 2 \cdot A \cdot (\vec{x} - t \cdot \vec{s}) + 2 \cdot \vec{b} \overset{!}{=} 0 \ \Leftrightarrow \ \vec{s}^T \cdot A \cdot (\vec{x} - t \cdot \vec{s}) + \vec{s}^T \cdot \vec{b} = 0$$

$$\Leftrightarrow \ t \cdot \vec{s}^T \cdot A \cdot \vec{s} = \vec{s}^T \cdot A \cdot \vec{x} + \vec{s}^T \cdot \vec{b} \ \Rightarrow \ \underline{t(\vec{x}) = \frac{\vec{s}^T \cdot A \cdot \vec{x} + \vec{s}^T \cdot \vec{b}}{\vec{s}^T \cdot A \cdot \vec{s}}} \ \ mit \ \ \vec{s} = \nabla F(\vec{x})$$

2) *Berechne iterativ* $\vec{x}^{(k)}$ *z.B. mit* $\vec{x}^{(0)} = (0, \ 0)^T$

$$\vec{s}^{(0)} = \nabla F(\vec{x}^{(0)}) = 2 \cdot b = \begin{pmatrix} -20 \\ -20 \end{pmatrix} \quad t^{(0)} = t(\vec{x}^{(0)}) = \frac{\vec{s}^{(0)T} \cdot A \cdot \vec{x}^{(0)} + \vec{s}^{(0)T} \cdot \vec{b}}{\vec{s}^{(0)T} \cdot A \cdot \vec{s}^{(0)}} = \frac{0 + 400}{4800} = \frac{1}{12}$$

$$\vec{x}^{(1)} = \vec{x}^{(0)} - t^{(0)} \cdot \vec{s}^{(0)} = \begin{pmatrix} \frac{5}{3} \\ \frac{5}{3} \end{pmatrix} \qquad \vec{s}^{(1)} = \nabla F(\vec{x}^{(1)}) \quad etc.$$

2.) Newton-Verfahren – **iterativ**

Hier bestimmt man mit Hilfe des Newton-Verfahrens nur Nullstellenbestimmung die Nullstelle des Gradienten (entspricht lokalem Minimum / Maximum). Es gilt:

$$\vec{x}^{(k+1)}=\vec{x}^{(k)}-H_F(\vec{x}^{(k)})^{-1}\cdot\nabla F(\vec{x}^{(k)}) \qquad \text{mit } H_F(\vec{x})=\text{ Hesse-Matrix von } F(\vec{x})$$

Bsp: Selbe Werte wie oben

$$H_F(\vec{x})=\begin{pmatrix}\frac{\delta\nabla F(\vec{x})_1}{\delta x_1} & \frac{\delta\nabla F(\vec{x})_1}{\delta x_2}\\ \frac{\delta\nabla F(\vec{x})_2}{\delta x_1} & \frac{\delta\nabla F(\vec{x})_2}{\delta x_2}\end{pmatrix}=\begin{pmatrix}2 & 2\\ 2 & 18\end{pmatrix} \quad\Rightarrow\quad H_F^{-1}(\vec{x})=\frac{1}{16}\cdot\begin{pmatrix}9 & -1\\ -1 & 1\end{pmatrix}$$

$$\vec{x}^{(1)}=\vec{x}^{(0)}-H_F^{-1}(\vec{x}^{(0)})\cdot\nabla F(\vec{x}^{(0)})=\begin{pmatrix}0\\0\end{pmatrix}-\frac{1}{16}\cdot\begin{pmatrix}9 & -1\\ -1 & 1\end{pmatrix}\cdot\begin{pmatrix}-20\\-20\end{pmatrix}=\begin{pmatrix}10\\0\end{pmatrix}$$

3.) cg-Verfahren (conjugate gradients) – **exakt**

Anstatt wie beim Gradientenverfahren immer den Gradienten als Suchrichtung zu verwenden, berechnet man nach jedem Schritt den "konjugierten" Gradienten, was dazu führt, dass dieses Verfahren wesentlich schneller konvergiert.
Es gilt:

$$\vec{g}^{(0)}=A\cdot\vec{x}^{(0)}+b \qquad \vec{s}^{(0)}=-\vec{g}^{(0)} \qquad \vec{x}^{(0)}\ \textit{beliebig}$$

$$\alpha_k=\underset{t}{argmin}\{F(\vec{x}^{(k)}+t\cdot\vec{s}^{(k)})\} \qquad \beta_k=\frac{\vec{g}^{(k+1)T}\cdot\vec{g}^{(k+1)}}{\vec{g}^{(k)T}\cdot\vec{g}^{(k)}}$$

$$\vec{x}^{(k+1)}=\vec{x}^{(k)}+\alpha_k\cdot\vec{s}^{(k)} \qquad \vec{g}^{(k+1)}=\vec{g}^{(k)}+\alpha_k\cdot A\cdot\vec{s}^{(k)} \qquad \vec{s}^{(k+1)}=-\vec{g}^{(k+1)}+\beta_k\cdot\vec{s}^{(k)}$$

Numerische Lösung linearer Optimierungsprobleme mit Nebenbedingungen

Gegeben sei die Funktion $F(\vec{x})=\sum_{i=1}^{S} c_i \cdot x_i = \vec{c}^T \cdot \vec{x}$ mit den Nebenbedingungen $A \cdot \vec{x} \le \vec{b}$ und $\vec{x} \ge 0$, wobei A eine Z x S Matrix ist.
Um das Ungleichungssystem in ein Gleichungssystem umzuformen wird in jeder Nebenbedingung (also jeder Zeile von Ax) eine sog. „Schlupfvariable“ x_{S+j} eingeführt. D.h. Der Vektor y entsteht durch Anhängen der Z Schlupfvariablen an den Vektor x und die Matrix B entsteht durch Anhängen der Z x Z Einheitsmatrix an A.

$$A \cdot \vec{x} \le \vec{b} \quad \rightarrow \quad B \cdot \vec{y} = \vec{b}$$

$$\vec{x}=\begin{pmatrix} x_1 \\ \vdots \\ x_S \end{pmatrix} \quad \rightarrow \quad \vec{y}=\begin{pmatrix} x_1 \\ \vdots \\ x_S \\ \boldsymbol{x_{S+1}} \\ \vdots \\ \boldsymbol{x_{S+Z}} \end{pmatrix} \quad \wedge \quad A=\begin{pmatrix} a_{11} & \cdots & a_{1S} \\ \vdots & \ddots & \vdots \\ a_{Z1} & \cdots & a_{ZS} \end{pmatrix} \quad \rightarrow \quad B=\begin{pmatrix} a_{11} & \cdots & a_{1S} & \mathbf{1} & 0 & 0 \\ \vdots & \ddots & \vdots & 0 & \ddots & 0 \\ a_{Z1} & \cdots & a_{ZS} & 0 & 0 & \mathbf{1} \end{pmatrix}$$

Die Lösung des Problems $\tilde{\vec{y}} = argmin\left\{\vec{c}^T \cdot \vec{y} \,:\, B \cdot \vec{y} \le \vec{b} \ \wedge \ \vec{y} \ge \vec{b}\right\}$ ist das Ergebnis von $\vec{y} = B^{\sim 1} \cdot \vec{b}$, wobei $B^{\sim 1}$ die Pseudoinverse von B ist (s. SVD).

Bsp: Optimiere $F(\vec{x}) = x_1 - x_2 + 2x_3$ unter folgenden NB

1) $2 \cdot x_2 - x_1 \ge 1 \quad \Rightarrow \quad x_1 - 2 \cdot x_2 \le 1$ 2) $x_2 \le 2$ 3) $x_1 + x_2 + x_3 \le 4$

$$A \cdot \vec{x} = \begin{pmatrix} 1 & -2 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \le \begin{pmatrix} 1 \\ 2 \\ 4 \end{pmatrix} = \vec{b}$$

Einführen der Schlupfvariablen:

1) $x_1 - 2 \cdot x_2 + x_4 = 1$ 2) $x_2 + x_5 = 2$ 3) $x_1 + x_2 + x_3 + x_6 = 4$

$$B \cdot \vec{y} = \begin{pmatrix} 1 & -2 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 4 \end{pmatrix} = \vec{b} \quad \Rightarrow \quad \tilde{\vec{y}} = B^{\sim 1} \cdot \vec{b} = \begin{pmatrix} 3 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \quad \Rightarrow \quad \tilde{\vec{x}} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 3 \\ 1 \\ 0 \end{pmatrix}$$

Simplexverfahren

Beim Simplexverfahren beginnt man in irgendeiner Ecke des Polyeders und läuft an einer Kante entlang, an der die Funktion F(x) größer wird. Existiert so eine Kante nicht, dann ist man im Optimum.
Das numerische Lösungsverfahren wird in der Klausur nicht gefordert, deshalb lasse ich es aus Motivationsgründen mal weg....