1 Euklidische Approximation

Sei V ein reeller euklidischer Vektorraum.
Das Skalarprodukt in V wird mit $\langle \cdot, \cdot \rangle_V$ und die Norm mit $\| \cdot \|_V$ bezeichnet.
$V_N \subset V$ sei ein Teilraum der Dimension $N < \infty$ mit Basis $\{\phi_N^n\}_{n=1,\dots,N}$.

(1.1) Problemstellung: Sei $v \in V$. Bestimme $v_N^* \in V$ mit

$$\|v - v_N^*\|_V = \min_{v_N \in V_N} \|v - v_N\|_V .$$

(1.2) Die Matrix

$$A = \left(\langle \phi_N^m, \phi_N^n \rangle_V \right)_{m,n=1,\dots,N} \in \mathbb{R}^{N,N}$$

ist symmetrisch und positiv definit.

(1.3) Problem (1.1) ist eindeutig lösbar. Es gilt

$$v_N^* = \sum_{n=1}^{N} x_n^* \phi_N^n ,$$

wobei $x^* \in \mathbb{R}^N$ die eindeutige Lösung des linearen Gleichungssystems

$$Ax^* = b$$

mit $b = \left(\langle v, \phi_N^m \rangle_V \right)_{m=1,\dots,N} \in \mathbb{R}^N$ ist.

2 Direkte Lösungsverfahren für lineare Gleichungen

(2.1) Sei $x = (x_n)_{n=1,\dots,N} \in \mathbb{R}^N$, $A = (a_{m,n})_{m=1,\dots,M,\ n=1,\dots,N} \in \mathbb{R}^{M,N}$.

a) Sei $1 \le m \le n \le N$. Dann ist
$$x[m:n] = (x_k)_{k=m,\dots,n} \in \mathbb{R}^{1+n-m}$$
Teilvektor von x.

b) Seien $1 \le m_1 \le m_2 \le M$, $1 \le n_1 \le n_2 \le N$. Dann ist
$$A[m_1 : m_2, n_1 : n_2] = (a_{jk})_{j=m_1,\dots,m_2,\ k=n_1,\dots,n_2} \in \mathbb{R}^{1+m_2-m_1,1+n_2-n_1}$$
Untermatrix von A.

c) Die Matrix A lässt sich in $A = \mathrm{lower}(A) + \mathrm{diag}(A) + \mathrm{upper}(A)$ zerlegen. Dabei ist
- $\mathrm{lower}(A) \in \mathbb{R}^{M,N}$ eine strikte untere Dreiecksmatrix mit $\mathrm{lower}(A)[m,n] = A[m,n]$ für $m > n$,
- $\mathrm{diag}(A) \in \mathbb{R}^{M,N}$ eine Diagonalmatrix mit $\mathrm{diag}(A)[n,n] = A[n,n]$ für $n = 1,\dots,\min\{M,N\}$,
- $\mathrm{upper}(A) \in \mathbb{R}^{M,N}$ eine strikte obere Dreiecksmatrix mit $\mathrm{upper}(A)[m,n] = A[m,n]$ für $m < n$.

d) A ist eine untere Dreiecksmatrix, wenn $A = \mathrm{diag}(A) + \mathrm{lower}(A)$ gilt. A ist eine obere Dreiecksmatrix, wenn $A = \mathrm{diag}(A) + \mathrm{upper}(A)$ gilt. Eine Dreiecksmatrix A heißt normiert, wenn $\mathrm{diag}(A)[n,n] = 1$ für $n = 1,\dots \min\{M,N\}$ gilt.

f) Mit $0_N \in \mathbb{R}^N$ wird der Nullvektor bezeichnet, mit $e^n \in \mathbb{R}^N$ wird der n-te Einheitsvektor bezeichnet, und mit $I_N \in \mathbb{R}^{N,N}$ die Einheitsmatrix.

2 Direkte Lösungsverfahren für lineare Gleichungen

(2.2) Sei $L \in \mathbb{R}^{N,N}$ eine normierte untere Dreiecksmatrix und $b \in \mathbb{R}^N$. Dann ist L regulär und das Lineare Gleichungssystem (LGS) $Ly = b$ ist mit $O(N^2)$ Operationen lösbar.
Entsprechend ist für eine reguläre obere Dreiecksmatrix $R \in \mathbb{R}^{N,N}$ das LGS $Rx = y$ in $O(N^2)$ Operationen lösbar.

(2.3) Die normierten unteren Dreiecksmatrizen bilden eine Gruppe.
Die regulären oberen Dreiecksmatrizen bilden eine Gruppe.

(2.4) Wenn eine Matrix $A \in \mathbb{R}^{N,N}$ eine LR-Zerlegung $A = LR$ mit einer normierten untere Dreiecksmatrix L und einer regulären obere Dreiecksmatrix R besitzt, dann ist A regulär und das LGS $Ax = b$ ist mit $O(N^2)$ Operationen lösbar.

(2.5) Eine Matrix $A \in \mathbb{R}^{N,N}$ besitzt genau dann eine LR-Zerlegung von A, wenn alle Hauptuntermatrizen $A[1:n, 1:n]$ regulär sind. Die LR-Zerlegung ist eindeutig und lässt sich mit $O(N^3)$ Operationen berechnen.

(2.6) Eine Matrix $A \in \mathbb{R}^{N,N}$ heißt *strikt diagonal-dominant*, falls $|a_{mm}| > \sum_{\substack{n=1 \\ n \neq m}}^{N} |a_{mn}|$.

(2.7) Wenn A strikt diagonal dominant ist, dann existiert eine LR-Zerlegung.

(2.8) Sei $A \in \mathbb{R}^{N,N}$ symmetrisch und positiv definit. Dann existiert genau eine Cholesky-Zerlegung $A = LDL^T$ mit einer normierten unteren Dreiecksmatrix L und einer Diagonalmatrix D.

2 Direkte Lösungsverfahren für lineare Gleichungen

(2.9) Sei $\pi \in S_N$ eine Permutation. Dann heißt $P_\pi = \left(e^{\pi(1)}|...|e^{\pi(N)}\right) \in \mathbb{R}^{N,N}$ Permutationsmatrix zu π. Wir schreiben $P_{(mn)}$ für die Permutationsmatrix mit der Vertauschung $\pi = (mn)$, d.h. $\pi(m) = n$, $\pi(n) = m$, $\pi(k) = k$ für $k \neq m, n$.

(2.10) Die Permutationsmatrizen in $\mathbb{R}^{N,N}$ bilden eine Gruppe.
Es gilt $P_\sigma P_\pi = P_{\pi\circ\sigma}$ und $P_{\pi^{-1}} = P_\pi^T$.

(2.11) Sei $A \in \mathbb{R}^{N,N}$ regulär. Dann existiert eine Permutationsmatrix P, so dass PA eine LR-Zerlegung $PA = LR$ besitzt und für die Einträge $|L[m,n]| \leq 1$ gilt.

Sei $|\cdot|$ eine Vektornorm, und sei $\|\cdot\|$ eine zugeordete Matrixnorm, d. h.,

$$|Ax| \leq \|A\|\,|x|\,, \qquad x \in \mathbb{R}^N,\ A \in \mathbb{R}^{M,N}\,.$$

(2.12) Sei $A \in \mathbb{R}^{N,N}$ regulär, und sei $\Delta A \in \mathbb{R}^{N,N}$ so klein, dass $\|\Delta A\| < \|A^{-1}\|^{-1}$ gilt.
Dann ist die Matrix $\tilde{A} = A + \Delta A$ regulär.
Sei $b \in \mathbb{R}^N$, $b \neq 0_N$, $\Delta b \in \mathbb{R}^N$ klein und $\tilde{b} = b + \Delta b$.
Dann gilt für die Lösungen $x \in \mathbb{R}^N$ von $Ax = b$ und $\tilde{x} \in \mathbb{R}^N$ von $\tilde{A}\tilde{x} = \tilde{b}$

$$\frac{|\Delta x|}{|x|} \leq \frac{\kappa(A)}{1 - \kappa(A)\frac{\|\Delta A\|}{\|A\|}} \left(\frac{|\Delta b|}{|b|} + \frac{\|\Delta A\|}{\|A\|}\right).$$

Dabei ist $\Delta x = \tilde{x} - x$, $\frac{|\Delta x|}{|x|}$ der *relative Fehler*, und $\kappa(A) = \|A\|\,\|A^{-1}\|$ die *Kondition* von A.

2 Direkte Lösungsverfahren für lineare Gleichungen

(2.13) a) Ein Problem heißt *sachgemäß gestellt*, wenn es eindeutig lösbar ist und die Lösung stetig von den Daten abhängt.
b) Die *Kondition* eines Problems ist eine Maß dafür, wie stark die Abhängigkeit der Lösung von den Daten ist.
c) Die *Stabilität* eines numerischen Algorithmus ist eine Maß dafür, wie stark die Daten-Abhängigkeit der numerischen Lösung im Vergeich zu der tatsächlichen Lösung ist.

(2.14) Wir verwenden für $x \in \mathbb{R}^N$ und $A \in \mathbb{R}^{M,N}$

$$|x|_1 = \sum_{n=1}^{N} |x_n| \qquad |x|_2 = \sqrt{x^T x} \qquad |x|_\infty = \max_{n=1,\dots,N} |x_n|$$

und die zugeordnete Operatornorm $\|A\|_p = \sup_{x \neq 0_N} \frac{|Ax|_p}{|x|_p}$, d.h.

$$\|A\|_1 = \max_{n=1,\dots,N} \sum_{m=1}^{M} |a_{mn}|, \quad \|A\|_2 = \sqrt{\rho(A^T A)}, \quad \|A\|_\infty = \max_{m=1,\dots,M} \sum_{n=1}^{N} |a_{mn}|$$

mit *Spekralradius* $\rho(A) = \max\{|\lambda| : \lambda \in \sigma(A)\}$ und Spektrum $\sigma(A)$.

2 Direkte Lösungsverfahren für lineare Gleichungen

(2.16) Sei $Q \in \mathbb{R}^{N,N}$ orthogonal, d.h. $Q^T Q = I_N$. Dann ist $\kappa_2(Q) = 1$.

(2.17) Zu $x \in \mathbb{R}^N$ und $m \neq n$ mit $x_m^2 + x_n^2 > 0$ existiert eine *Givensrotation*

$$Q = \begin{pmatrix} 1 & & & & & & & & \\ & \ddots & & & & & & & \\ & & 1 & & & & & & \\ & & & c & & & s & & \\ & & & & 1 & & & & \\ & & & & & \ddots & & & \\ & & & & & & 1 & & \\ & & & -s & & & c & & \\ & & & & & & & \ddots & \end{pmatrix} \begin{matrix} \\ \\ \\ \longleftarrow m \\ \\ \\ \longleftarrow n \\ \\ \end{matrix}$$

(m ↓ über Spalte 3, n ↓ über Spalte 7)

mit $c^2 + s^2 = 1$ und $y_n = 0$ für $Qx = y$.

(2.18) Zu $x \in \mathbb{R}^N$, $x \neq 0$, existiert eine Householdertransformation

$Q = I_N - \frac{2}{w^T w} w w^T \in \mathbb{R}^{N,N}$ mit $w \in \mathbb{R}^N$, $w_1 = 1$, sodass $Qx = \sigma e^1$ mit $\sigma \in \mathbb{R}$.

(2.19) Zu $A \in \mathbb{R}^{M,N}$ existiert eine QR-Zerlegung $A = QR$ mit einer orthogonalen Matrix $Q \in \mathbb{R}^{M,M}$ und eine oberen Dreiecksmatrixmatrix $R \in \mathbb{R}^{M,N}$.

2 Direkte Lösungsverfahren für lineare Gleichungen

(2.20) Sei $A \in \mathbb{R}^{M,N}$ und $b \in \mathbb{R}^M$. Dann gilt:
$x \in \mathbb{R}^N$ minimiert $|Ax - b|_2 \quad \Longleftrightarrow \quad A^T Ax = A^T b$.

(2.21) Zu $A \in \mathbb{R}^{M,N}$ mit $R = \operatorname{rang}(A)$ existieren Singulärwerte $\sigma_1, \ldots, \sigma_R > 0$ und eine Singulärwertzerlegung

$$A = V \Sigma U^T$$

mit $V \in \mathbb{R}^{M,M}$, $U \in \mathbb{R}^{N,N}$ orthogonal und $\Sigma \in \mathbb{R}^{M,N}$ mit $\Sigma[r,r] = \sigma_r$ für $r = 1, \ldots, R$ und $\Sigma[m,n] = 0$ sonst.

(2.22) $A^+ = U \Sigma^+ V^T$ ist die *Pseudo-Inverse*
mit $\Sigma^+ \in \mathbb{R}^{N,M}$ mit $\Sigma^+[r,r] = 1/\sigma_r$ für $r = 1, \ldots, R$ und $\Sigma^+[m,n] = 0$ sonst.

(2.22) $x = A^+ b$ löst die Normalengleichung $A^T Ax = A^T b$.

(2.23) Sei $A \in \mathbb{R}^{M,N}$ und $b \in \mathbb{R}^M$.
Dann gilt für die Tikhonov-Regularisierung mit $\alpha > 0$:
$x \in \mathbb{R}^N$ minimiert $|Ax - b|_2^2 + \alpha |x|_2^2 \quad \Longleftrightarrow \quad (A^T A + \alpha I_N) x = A^T b$.

(2.24) Es gilt $\lim_{\alpha \longrightarrow 0} (A^T A + \alpha I_N)^{-1} A^T b = A^+ b$.

3 Eigenwertberechnung

(3.2) Eine Matrix $H \in \mathbb{R}^{N,N}$ heißt *Hessenberg*-Matrix, wenn
$H[n+2:N,n] = 0_{N-n-1}$ für $n = 1,...,N-2$.

(3.3) Sei $A \in \mathbb{R}^{N,N}$. Dann existiert eine orthogonale Matrix $Q \in \mathbb{R}^{N,N}$, so dass
$H = QAQ^T$ eine Hessenberg-Matrix ist.
Die Berechnung benötigt $O(N^3)$ Operationen.
Wenn A symmetrisch ist, dann ist H eine Tridiagonalmatrix.

(3.4) Sei $A \in \mathbb{R}^{N,N}$ symmetrisch, tridiagonal, und irreduzibel, d.h.
$A[n-1,n] = A[n,n-1] \neq 0$ und $A[n+2:N,n] = A[n,n+2:N]^T = 0_{N-n-1}$.
Die charakteristischen Polynome $P_n(t) = \det\big(A[1:n,1:n] - tI_n\big)$ der Hauptuntermatrizen lassen sich durch eine Dreitermrekursion berechnen:
Setze $P_0 \equiv 1$. Dann gilt $P_1(t) = A[1,1] - t$ und

$$P_n(t) = (A[n,n] - t)P_{n-1}(t) - A[n-1,n]^2 P_{n-2}(t).$$

Sie bilden eine *Sturmsche Kette*: Für die Nullstellen $\lambda_1^n \leq \lambda_2^n \leq \cdots \leq \lambda_n^n$ gilt

$$\lambda_{k-1}^{n-1} < \lambda_k^n < \lambda_k^{n-1}$$

(mit $\lambda_0^n = \underline{\lambda}$ und $\lambda_{n+1}^n = \overline{\lambda}$, falls $\sigma(A) \subset (\underline{\lambda}, \overline{\lambda})$), und es gilt für $t \in (\underline{\lambda}, \overline{\lambda})$

$$\lambda_k^n < t \leq \lambda_{k+1}^n$$

mit $k = W_n(t)$ und $W_n(t) = \#\big\{i \in \{1,...,n\} : P_i(t)P_{i-1}(t) < 0 \text{ oder } P_i(t) = 0\big\}$.

3 Eigenwertberechnung

Sei $A \in \mathbb{R}^{N,N}$ symmetrisch mit Eigenwerten $\lambda_1, ..., \lambda_N$ und ONB aus Eigenvektoren $v^1, ..., v^N$. Dann gilt

$$A = \sum_n \lambda_n v^n (v^n)^T .$$

(3.5) Der *Rayleigh-Quotient* ist

$$r(A,x) = \frac{x^T A x}{x^T x}, \qquad x \in \mathbb{R}^N,\ x \neq 0_N .$$

(3.6) Sei $|\lambda_1| = \rho(A)$ und $|\lambda_n| < |\lambda_1|$ für $n = 2, ..., N$.
Dann gilt für alle $w \in \mathbb{R}^N$ mit $w^T v^1 > 0$

$$\lim_{k \longrightarrow \infty} r(A, A^k w) = \lambda_1 , \qquad \lim_{k \longrightarrow \infty} \frac{1}{|A^k w|_2} A^k w = v^1 .$$

(3.7) Sei $|w|_2 = 1$ und $s = r(A,w)$. Dann gilt

$$\min_{n=1,...,N} |\lambda_n - s| \leq |Aw - sw|_2 .$$

3 Eigenwertberechnung

(3.8) Eine konvergente Folge (d^k) in $\mathbb{R}$ mit Grenzwert d^* konvergiert

a) *linear*, wenn $c \in (0,1)$ und $k_0 > 0$ existieren mit
$$|d^{k+1} - d^*| \le c\,|d^k - d^*| \qquad \text{für } k \ge k_0$$

b) *superlinear*, wenn zu jedem $\varepsilon > 0$ ein $k_0 > 0$ existiert mit
$$|d^{k+1} - d^*| \le \varepsilon\,|d^k - d^*| \qquad \text{für } k \ge k_0$$

c) *von der Ordnung* $p > 1$, wenn wenn $C > 0$ existiert mit
$$|d^{k+1} - d^*| \le C\,|d^k - d^*|^p .$$

(3.9) Inverse Iteration mit variablem shift

S0) Wähle $z^0 \in \mathbb{R}^N$, $z^0 \neq 0_N$, $\varepsilon \ge 0$. Setze $k = 0$.
S1) Setze $w^k = \frac{1}{|z_k|_2} z^k$, $s_k = r(A, w^k)$.
S2) Falls $|Aw^k - s_k w^k|_2 \le \varepsilon$ STOP.
S3) Berechne $z^{k+1} = (A - s_k I_N)^{-1} w^k$.
S4) Setze $k := k+1$, gehe zu S1).

Wenn der Startvektor z^0 hinreichend nahe bei einem Eigenvektor v^m mit isoliertem Eigenwert λ_m liegt, konvergiert die Iteration kubisch (d.h. von der Ordnung $p = 3$).

3 Eigenwertberechnung: QR-Iteration mit shift

Sei $A \in \mathbb{R}^{N,N}$ symmetrisch.
S0) Berechne $A_0 = QAQ^T$ tridiagonal (Hessenberg-Transformation).
Wähle $\varepsilon \geq 0$. Setze $k = 0$.

S1) Falls $|A_k[n+1,n]| \leq \varepsilon$ für ein n:
getrennte Eigenwertberechnung für $A_k[1:n,1:n]$ und $A_k[n+1:N,n+1:N]$.

S2) Berechne $d_k = \frac{1}{2}(A_k[N-1,N-1] - A_k[N,N])$ und

$$s_k = A_k[N,N] + d_k - \operatorname{sgn}(d_k)\sqrt{d_k^2 + A_K[N-1,N]^2}\,.$$

S3) Berechne QR-Zerlegung

$$Q_k R_k = A_k - s_k I_N$$

und setze

$$A_{k+1} = R_k Q_k + s_k I_N\,.$$

S4) Setze $k := k+1$, gehe zu S1).

Es gilt $A_{k+1} = Q_k^T A_k Q_k$.
Falls der shift $s_k = A_k[N,N]$ gewählt wird, entspricht die QR-Iteration der Inversen Iteration mit variablem shift und Startvektor $z^0 = e^N$.

3 Eigenwertberechnung

(3.10) Gershgorin

Zu $A \in \mathbb{R}^{N,N}$ sind die Gershgorin-Kreise durch

$$K_n = \left\{ \lambda \in \mathbb{C} \colon |\lambda - A[n,n]| \leq \sum_{k \neq n} |A[n,k]| \right\}, \qquad n = 1, \ldots, N$$

defininiert. Dann gilt

$$\sigma(A) \subset \bigcup_{n=1}^{N} K_n .$$

(3.11) Sei $A \in \mathbb{R}^{N,N}$ symmetrisch mit Eigenwerten $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N$.

$$\begin{aligned} \lambda_n &= \max_{\dim S = n} \min_{0_N \neq x \in S} r(A,x), \\ \lambda_{N+1-n} &= \min_{\dim S = n} \max_{0_N \neq x \in S} r(A,x). \end{aligned}$$

4 Iterative Lösungsverfahren für lineare Gleichungen

(4.1) Sei $A, B \in \mathbb{R}^{N,N}$ mit $\rho(I_N - BA) < 1$. Dann ist A invertierbar, und es gilt für alle $b \in \mathbb{R}^N$ und alle Startvektoren $x^0 \in \mathbb{R}^N$ konvergiert die Iteration

$$x^{k+1} = x^k + B(b - Ax^k), \qquad k = 0, 1, 2, \ldots$$

gegen $\lim_{k \longrightarrow \infty} x^k = A^{-1} b$.

(4.2) Sei $K \in \mathbb{R}^{N,N}$ und $\varepsilon > 0$
Dann exisitiert eine Matrix-Norm $\|\cdot\|$, so dass $\|K\| \leq \rho(K) + \varepsilon$ gilt.

Anwendung: Es gilt $|x - x^k| \leq \|I_N - BA\|^k |x - x^0|$ (lineare Konvergenz).

(4.3) Konvergenz des Gauß-Seidel-Verfahrens
Sei $A \in \mathbb{R}^{N,N}$ symmetrisch positiv definit und sei $B = \big(\mathrm{diag}(A) + \mathrm{lower}(A)\big)^{-1}$.
Dann gilt bezüglich der Energienorm $|x|_A = \sqrt{x^T Ax}$ und $\|K\|_A = \sup_{x \neq 0_N} \frac{|Kx|_A}{|x_A|}$

$$\|I_N - BA\|_A < 1 .$$

4 Iterative Lösungsverfahren: Krylov-Verfahren

Sei $\langle \cdot, \cdot \rangle_V$ ein Skalarprodukt in $V = \mathbb{R}^N$.

S0) Wähle $x^0 \in \mathbb{R}^N$.
Berechne $r^0 = b - Ax^0$, $z^1 = Br^0$, $h_{10} = |z^1|_V$ und $v^1 = \frac{1}{h_{10}} z^1$. Setze $k = 1$.

S1) Berechne

$$\begin{aligned} w^k &= BAv^k \\ z^{k+1} &= w^k - \sum_{j=1}^{k} h_{jk} v^j \text{ mit } h_{jk} = \langle v^j, w^k \rangle_V \\ v^{k+1} &= \frac{1}{h_{k+1,k}} z^{k+1} \text{ mit } h_{k+1,k} = |z^{k+1}|_V \end{aligned}$$

Dann ist $v^1, ..., v^k$ eine Orthonormalbasis von dem *Krylov-Raum*

$$V_k = \text{span}\{Br^0, BABr^0, ..., (BA)^{k-1} Br^0\} = \{Q_k y : y \in \mathbb{R}^k\}, \quad Q_k = \left(v^1 | .... | v^k\right).$$

Es gilt $BAv^k = \sum_{j=1}^{k+1} h_{jk} v^j$, also $BAQ_k = Q_{k+1} H_k$ mit $H_k = (h_{jm}) \in \mathbb{R}^{k+1,k}$.

GMRES-Verfahren: Wähle $\langle v, w \rangle_V = v^T w$.
cg-Verfahren (A, B symmetrisch positiv definit): Wähle $\langle v, w \rangle_V = v^T Aw$.

4 Iterative Lösungsverfahren: GMRES-Verfahren

S0) Wähle $x^0 \in \mathbb{R}^N$, $\varepsilon > 0$.
Berechne $r^0 = b - Ax^0$, $z^1 = Br^0$, $h_{10} = |z^1|_2$ und $v^1 = \frac{1}{h_{10}} z^1$. Setze $k = 1$.

S1) Berechne

$$\begin{aligned} w^k &= BAv^k \\ z^{k+1} &= w^k - \sum_{j=1}^{k} h_{jk} v^j \text{ mit } h_{jk} = (v^j)^T w^k \\ v^{k+1} &= \frac{1}{h_{k+1,k}} z^{k+1} \text{ mit } h_{k+1,k} = |z^{k+1}|_2 \end{aligned}$$

S2) Berechne $y^k \in \mathbb{R}^k$ mit $\rho_k = |H_k y_k - h_{10} e^1|_2 = \min!$
Dabei ist $H_k = (h_{jm})_{j=1,\dots,k+1,\ m=1,\dots,k} \in \mathbb{R}^{k+1,k}$.

S3) Wenn $\rho_k < \varepsilon$, setze $x^k = x^0 + \sum_{j=1}^{k} y_j^k v^j$ STOP.

S4) Setze $k := k+1$ und gehe zu S1).

(4.4) Es gilt $\rho_k = \min_{z \in x^0 + V_k} |B(b - Az)|_2$.

4 Iterative Lösungsverfahren: cg-Verfahren

S0) Wähle $x^0 \in \mathbb{R}^N$, $\varepsilon > 0$.
Berechne $r^0 = b - Ax^0$, $w^0 = Br^0$, $\rho_0 = (w^0)^T r^0$ und $d^1 = w^0$. Setze $k = 0$.

S1) Falls $\rho_k \leq \varepsilon$ STOP

S2) Setze $k := k+1$ und berechne

$$\begin{aligned} u^k &= Ad^k \\ \alpha_k &= \frac{\rho_{k-1}}{(u^k)^T d^k} \\ x^k &= x^{k-1} + \alpha_k d^k \\ r^k &= r^{k-1} - \alpha_k u^k \\ w^k &= Br^k \\ \rho_k &= (w^k)^T r^k \\ d^{k+1} &= w^k + \frac{\rho_k}{\rho_{k-1}} d^k \end{aligned}$$

Gehe zu S1).

(4.4)
$$\begin{aligned} |x^k - x|_A &= \min_{z \in x^0 + V_k} |z - x|_A \leq \min_{P \in \mathbb{P}_k,\ P(0)=1} \max_{\lambda \in \sigma(BA)} |P(\lambda)| |x^0 - x|_A \\ &\leq 2\Big(\frac{\sqrt{\kappa(BA)} - 1}{\sqrt{\kappa(BA)} + 1}\Big)^k |x^0 - x|_A . \end{aligned}$$

5 Iterative Lösungsverfahren für nichtlineare Gleichungen

(5.1) Sei $D \subset \mathbb{R}^N$ offen und $F\colon D \longrightarrow \mathbb{R}^N$ ein stetig differenzierbares Vektorfeld.
Sei $x^* \in D$ eine Nullstelle von F, und sei $B \in \mathbb{R}^{N,N}$.
Wenn $\rho\big(I_N - BF'(x^*)\big) < 1$ gilt, dann existiert ein $\delta > 0$, so dass für alle $x^0 \in B(x^*, \delta)$ die Fixpunktiteration $x^{k+1} = \Phi(x^k)$ mit $\Phi(x) = x - BF(x)$ linear gegen x^* konvergiert.

(5.2) Seien $\alpha, \beta, \gamma > 0$ mit $2\alpha\gamma < \beta^2$ und $P(t) = \alpha - \beta t + \frac{\gamma}{2} t^2$.
Dann konvergiert für $t_0 = 0$ das Newton-Verfahren

$$t_{k+1} = t_k - P'(t_k)^{-1} P(t_k)$$

quadratisch gegen die kleinste Nullstelle t^* von P, und $\{t_k\}$ ist monoton steigend mit

$$t_0 = 0 < t_1 < \cdots < t_k < t_{k+1} = t_k + \frac{\gamma}{2} \frac{(t_k - t_{k-1})^2}{\beta - \gamma t_k} \leq t^* = \frac{2\alpha}{\beta + \sqrt{\beta^2 - 2\alpha\gamma}}.$$

5 Iterative Lösungsverfahren für nichtlineare Gleichungen

(5.3) Sei $D \subset \mathbb{R}^N$ offen und $F\colon D \longrightarrow \mathbb{R}^N$ ein stetig differenzierbares Vektorfeld.
Sei $x^0 \in D$ mit
a) $|F(x^0)| \leq \alpha$
b) $|F'(x^0)y| \geq \beta\,|y|$ für $y \in \mathbb{R}^N$
c) $B(x^0, 2\alpha/\beta) \subset D$
d) $\|F'(y) - F'(z)\| \leq \gamma |y-z|$ für $y, z \in B(x^0, 2\alpha/\beta)$
e) $2\alpha\gamma < \beta^2$.
Dann ist das Newton-Verfahren $x^{k+1} = x^k - F'(x^k)^{-1} F(x^k)$ wohldefiniert und konvergiert quadratisch gegen $x^* \in D$ mit

$$|x^* - x^0| \leq \frac{2\alpha}{\beta + \sqrt{\beta^2 - 2\alpha\gamma}}.$$

Gedämpftes Newton-Verfahren

S0) Wähle $x^0 \in D$, $\varepsilon > 0$, $\theta \in (0,1)$. Setze $k = 0$.
S1) Falls $|F(x^k)| \leq \varepsilon$ STOP
S2) Löse $F'(x^k)d^k = -F(x^k)$.
S3) Bestimme $t_k \in \{1, \theta, \theta^2, ..., \theta^r\}$ mit $|F(x^k + t_k d^k)|$ minimal.
S4) Setze $x^{k+1} = x^k + t_k d^k$, $k := k+1$ und gehe zu S1).

6 Interpolation und Approximation

(6.1) *Lagrange Interpolation*
Zu Stützstellen $t_0 < t_1 < \cdots < t_N$ und Werten $f_0, f_1, \ldots, f_N \in \mathbb{R}$ bestimme $P \in \mathbb{P}_N$ mit $P(t_n) = f_n$.

(6.2) Die Interpolationsaufgabe (6.1) ist eindeutig lösbar.

(6.3) *Hermite Interpolation*
Zu $t_0 \le t_1 \le \cdots \le t_N$ definiere $d_n = \max\{n-k\colon t_k = \cdots = t_n\}$, $d = \max d_n$.
Zu $f \in C^d(\mathbb{R})$ bestimme $P \in \mathbb{P}_N$ mit

$$\Big(\frac{d}{dt}\Big)^{d_n} P(t_n) = \Big(\frac{d}{dt}\Big)^{d_n} f(t_n) \quad \text{für} \quad n = 0, \ldots, N.$$

(6.4) Die Interpolationsaufgabe (6.3) ist eindeutig lösbar.

(6.6) Zu $t_0 \le t_1 \le \cdots \le t_N$ definiere die Newton-Basis von $\mathbb{P}_N$ durch
$\omega_0 \equiv 1,\ \omega_1(t) = t - t_0,\ \omega_k(t) = (t - t_{k-1})\omega_{k-1}(t)$ für $k = 1, \ldots, N$.

(6.7) Der eindeutig bestimmte höchste Koeffizient $b_N \equiv \frac{1}{N!}\big(\frac{d}{dt}\big)^N P(t)$ der Lösung der Interpolationsaufgabe (6.3) wird mit $b_N = f[t_0, \ldots, t_N]$ bezeichnet.

(6.8) a) $P(t) = \sum_{k=0}^{N} f[t_0, \ldots, t_k]\omega_k(t)$ löst die Interpolationsaufgabe (6.3).

b) Für $f \in C^{N+1}(\mathbb{R})$ gilt $f(t) = P(t) + f[t_0, \ldots, t_N, t]\ \omega_{N+1}(t)$.

6 Interpolation und Approximation

(6.9) Für den Interpolationsfehler gilt $f(t) - P_N(t) = \dfrac{1}{(N+1)!}\left(\dfrac{d}{dt}\right)^{N+1} f(\tau)\ \omega_{N+1}(t)$
mit einer Zwischenstelle $\tau = \tau(t) \in I := [\min\{t, t_0\}\ \max\{t, t_N\}]$.

(6.10) a) Für $t_k = t_{k+1} = \cdots = t_n$ gilt $f[t_k, \ldots, t_n] = \dfrac{1}{(n-k)!}\left(\dfrac{d}{dt}\right)^{n-k} f(t_k)$.

b) Für $t_k < t_n$ gilt $f[t_k, \ldots, t_n] = \dfrac{f[t_{k+1}, \ldots, t_n] - f[t_k, \ldots, t_{n-1}]}{t_n - t_k}$.

c) Sei $P_0 \in \mathbb{P}_{n-k-1}$ Interpolation zu f an $t_k \le \cdots \le t_{n-1}$, und
sei $P_1 \in \mathbb{P}_{n-k-1}$ Interpolation zu f an $t_{k+1} \le \cdots \le t_n$.
Dann ist $P(t) = \frac{t-t_k}{t_n-t_k} P_1(t) + \frac{t_n-t}{t_n-t_k} P_0(t)$ Interpolation zu f an $t_k \le \cdots \le t_n$.

(6.11) Es gilt $f[t_0, \ldots, t_N] = \displaystyle\int_{\Sigma^N} \left(\frac{d}{dt}\right)^N f\Big(\sum_{n=0}^{N} s_n t_n\Big)\, ds$

mit $\quad \Sigma^N = \left\{ \begin{pmatrix} s_0 \\ \vdots \\ s_n \end{pmatrix} \in \mathbb{R}^{N+1} : \ \displaystyle\sum_{n=0}^{N} s_n = 1, s_n \ge 0 \right\}$ (Standard-Simplex)

(6.12) $f[t_0, \ldots, t_N] = \frac{1}{N!}\left(\frac{d}{dt}\right)^N f(\tau)$ mit $\tau \in [\min t_n, \max t_n]$.

6 Interpolation und Approximation

(6.13) a) Zu einer Zerlegung $\Delta\colon a = t_0 < t_1 < \cdots < t_N = b$ von $[a,b]$ definiere den Spline-Raum von Grad k

$$\mathscr{S}_k(\Delta) = \{S \in C^{k-1}[a,b]\colon S_n := S|_{[t_{n-1},t_n]} \in \mathbb{P}_k, \qquad n = 1,\dots,N\}.$$

b) $S \in \mathscr{S}_k(\Delta)$ heißt interpolierender Spline zu $f \in C^0[a,b]$ wenn $S(t_n) = f(t_n)$.

(6.14) Es gilt $\dim \mathscr{S}_k(\Delta) = k + N$.

Wir betrachten interpolierende Splines von Grad $k = 2r+1$ mit einer der Randbedingungen

(I) Natürliche Randbedingung

$$\left(\frac{d}{dt}\right)^m S(a) = 0 \quad \text{und} \quad \left(\frac{d}{dt}\right)^m S(b) = 0 \quad \text{für} \quad r+1 \le m \le 2r$$

(II) Hermite-Randbedingung zu $f \in C^r[a,b]$

$$\left(\frac{d}{dt}\right)^m S(a) = \left(\frac{d}{dt}\right)^m f(a) \quad \text{und} \quad \left(\frac{d}{dt}\right)^m S(b) = \left(\frac{d}{dt}\right)^m f(b) \ \text{ für } \ 1 \le m \le r$$

(III) periodische Randbedingungen

$$\left(\frac{d}{dt}\right)^m S(a) = \left(\frac{d}{dt}\right)^m S(b) \quad \text{für} \quad 1 \le m \le 2r$$

6 Interpolation und Approximation

Zu $g, h \in C^m[a,b]$ definiere

$$(g,h)_m = \int_a^b \left(\frac{d}{dt}\right)^m g(t) \left(\frac{d}{dt}\right)^m h(t)\, dt\,, \qquad |g|_m = \sqrt{(g,g)_m}\,.$$

(6.15) Sei $S \in \mathscr{S}_{2r+1}(\Delta)$ ein interpolierender Spline zu f und sei zusätzlich eine der Randbedingungen (I), (II) oder (III) erfüllt.
Sei $g \in C^{r+1}[a,b]$ mit $g(t_n) = f(t_n)$ für $n = 0,\ldots,N$, und zusätzlich gelte
im Fall (II)

$$\left(\frac{d}{dt}\right)^m g(a) = \left(\frac{d}{dt}\right)^m f(a) \quad \text{und} \quad \left(\frac{d}{dt}\right)^m g(b) = \left(\frac{d}{dt}\right)^m f(b) \ \text{ für } \ 1 \le m \le r\,,$$

im Fall (III)

$$\left(\frac{d}{dt}\right)^m g(a) = \left(\frac{d}{dt}\right)^m g(b) \quad \text{für} \quad 1 \le m \le r\,.$$

Dann gilt

$$(S-g,S)_{r+1} = 0 \quad \text{und} \quad |S|_{r+1} \le |g|_{r+1}\,.$$

(6.16) Die Spline-Interpolation $S \in S_{2r+1}(\Delta)$ zu f mit einer der Randbedingungen (I), (II), (III) ist eindeutig lösbar.

6 Interpolation und Approximation: Kubische Splines

(6.17) Es gilt für $m = 0, \dots, r$ und $h = \max_{i=n,\dots,N} h_i$, $h_n = t_n - t_{n-1}$

$$|S - f|_m \leq 2^{-(r+1-m)/2} \frac{(r+1)}{m!} h^{r+1-m} |f|_{r+1}$$

(6.18) Die *Momente* $M_n = S''(t_n)$ von $S \in \mathscr{S}_3(\Delta)$ zu f sind eindeutig durch

$$\mu_n M_{n-1} + M_n + \lambda_n M_{n+1} = 3f[t_{n-1}, t_n, t_{n+1}]$$

und $\mu_n = \dfrac{h_n}{2(h_n + h_{n+1})}$, $\quad \lambda_n = \dfrac{h_{n+1}}{2(h_n + h_{n+1})}$ bestimmt und es gilt

$$\begin{aligned} S_n(t) \quad = \quad & \frac{M_n (t - t_{n-1})^3 + M_{n-1}(t_n - t)^3}{6h_n} + \frac{f(t_n) + f(t_{n-1})}{2} - \frac{h_n^2}{12}(M_n + M_{n-1}) \\ & + \Big(\frac{f(t_n) - f(t_{n-1})}{h_n} - \frac{h_n}{6}(M_n - M_{n-1}) \Big) \Big(t - \frac{t_n + t_{n-1}}{2} \Big) . \end{aligned}$$

(6.19) Es gilt für $f \in C^2[a,b]$

$$\|S''\|_\infty \leq 3 \|f''\|_\infty .$$

(6.20) Es gilt für $f \in C^4[a,b]$

$$\|S - f\|_\infty \leq h^4 \Big\| \Big(\frac{d}{dt} \Big)^4 f \Big\|_\infty .$$

6 Interpolation und Approximation

Sei $W \in C(a,b),\ W > 0$ ein Gewicht und definiere

$$(u,v)_W = \int_a^b u(t)v(t)W(t)\,dt.$$

Aufgabe: Zu $f \in C(a,b)$ bestimme $P \in \mathbb{P}_N$ mit

$$\|P-f\|_W = \min_{Q\in\mathbb{P}_N} \|Q-f\|_W.$$

(6.21) Dann gilt für eine ONB $Q_0,\dots,Q_N \in \mathbb{P}_N$: $\quad P(t) = \sum_{k=0}^{N} (f,Q_k)_W\ Q_k(t).$

(6.22) Eine orthogonale Basis $R_n = \rho_n\ Q_n$, $n = 0,1,2,\dots$ berechnet sich mit $R_{-1} \equiv 0,\ R_0 \equiv \frac{\rho_0}{\|1\|_W}$ und

$$R_{n+1}(t) = (t-\beta_n)\ R_n(t) - \gamma_n\ R_{n-1}(t) \qquad n = 0,1,2,\dots$$

mit

$$\beta_n = \frac{(R_n, tR_n)_W}{\rho_n^2}, \qquad \gamma_n = \frac{\rho_n^2}{\rho_{n-1}^2}.$$

(6.23) Das Orthogonalpolynom R_n besitzt n verschiedene Nullstellen in (a,b).

Die Orthogonalpolynome $R_0, R_1, R_2, \dots$ bilden eine Sturmsche Kette.

7 Numerische Integration – Newton-Cotes-Formeln

(7.1) Eine Quadraturformel $I_\Xi : C[a,b] \longrightarrow \mathbb{R}$ mit $I_\Xi(f) = \sum_{\xi\in\Xi} \omega_\xi f(\xi)$ zu Stützstellen $\Xi \subset [a,b]$ und Gewichten ω_ξ ist eine Linearform zur Approximation des Integrals $I(f) = \int_a^b f(t)dt$. Die Quadraturformel I_Ξ heißt *exakt von der Ordnung p*, wenn der Fehler $I(P) = I_\Xi(P)$ für Polynome $P \in \mathbb{P}_p$.

(7.2) Sei $|\Xi| = N+1$. Wenn I_Ξ exakt von der Ordnung N ist, dann gilt

$$\omega_\xi = \int_a^b L_\xi(t)dt \text{ mit } L_\xi(t) = \textstyle\prod_{\eta\in\Xi\setminus\{\xi\}} \frac{t-\eta}{\xi-\eta} \in \mathbb{P}_N.$$

(7.3) Die Quadraturformeln von der Ordnung N zu äquidistanten Stützstellen $\xi_n = a+nh$, $h = \frac{b-a}{N}$ heißen *Newton-Cotes-Quadraturformeln*. Es gilt:

$$\omega_n = \omega_{N-n} = \frac{1}{N}\,\frac{(-1)^{N-n}}{n!(N-n)!}\int_0^N \prod_{j=0,\ j\neq n}^{N} (t-j)dt.$$

(7.4) Die Newton-Cotes-Formeln sind für gerade N exakt von der Ordnung $N+1$.

(7.5) Seien $g,h \in C[a,b]$ mit $g(t) \geq 0$ für $t \in [a,b]$. Dann gilt:

$$\int_a^b g(t)h(t)dt = h(\tau)\int_a^b g(t)dt \text{ für eine Zwischenstelle } \tau \in (a,b).$$

(7.6) Sei $f \in C^4[a,b]$. Dann gilt für die Simpsonregel:

$$I_2(f) - I(f) = \left(\frac{b-a}{2}\right)^5 \frac{1}{90} f''''(\tau) \text{ für ein } \tau \in (a,b).$$

7 Numerische Integration – Gauß-Quadatur

(7.7) Die maximale Ordnung einer Quadratur G_N mit N Stützstellen ist $2N-1$.

(7.8) Seien $R_0 \equiv 1$, $R_1(t) = t - \beta_0$, $R_{n+1}(t) = (t-\beta_n)R_n(t) - \gamma_n R_{n-1}(t)$ Orthogonalpolynome bzgl. $(f,g) = \int_a^b f(t)g(t)dt$, und seien $\xi_1, \dots, \xi_N$ die Nullstellen von R_N. Dann ist die zugehörige *Gauß-Quadratur*

$$G_N(f) = \sum_{n=1}^{N} \omega_n f(\xi_n) \text{ exakt für } P \in \mathbb{P}_{2N-1}.$$

(7.9) Für die Gewichte gilt $\omega_n = \frac{(R_{N-1}, R_{N-1})}{R'_N(\xi_n) R_{N-1}(\xi_n)}$.

(7.10) $$\int_a^b f dt - G_N(f) = \frac{1}{(2N)!} \left(\frac{d}{dt}\right)^{2N} f(\tau)\ (Q_N, Q_N) \quad \text{mit} \quad Q_N(t) = \prod_{n=1}^{N} (t - \xi_n).$$

(7.11) Seien $\Xi \subset [0,1]$ Stützstellen und ω_ξ Gewichte zu einer Quadratur $\hat{I}$ in $[0,1]$, dann ist

$$I_h(f) = \sum_{m=1}^{M} h \sum_{\xi \in \Xi} \omega_\xi f\big(a + (m-1+\xi)h\big) \quad \text{mit} \quad h = \frac{b-a}{M}$$ die zugehörige summierte Quadratur.

(7.12) $\hat{I}$ sei exakt für Polynome vom Grad p, und sei $f \in C^{p+1}[a,b]$. Dann gilt für die summierte Quadratur $|I(f) - I_h(f)| \le C(b-a)h^{p+1} \left\| \left(\frac{d}{dt}\right)^p f \right\|_\infty$.

7 Numerische Integration – Extrapolation und Romberg-Quadatur

(7.13) Sei $I_h - I = Ch^p + O(h^q)$ mit $q > p$. Dann gilt für die Extrapolation

$$\hat{I}_h = I_h + \frac{1}{2^p - 1}\left(I_h - I_{2h}\right)$$

a) $|I - \hat{I}_h| = O(h^q)$

b) $|I - I_h| = \frac{1}{2^p-1}\left(I_h - I_{2h}\right) + O(h^q)$

(7.14) Sei $g \in C^{2m+2}[0,1]$. Es existieren Konstanten b_j und eine Zwischenstelle $\xi \in (0,1)$ mit

$$\int_0^1 g(t)\,dt = \frac{1}{2}g(0) + \frac{1}{2}g(1) + \sum_{j=1}^{m} b_j\left(\left(\frac{d}{dt}\right)^{2j-1} g(0) - \left(\frac{d}{dt}\right)^{2j-1} g(1)\right) +$$

$$b_{m+1}\left(\frac{d}{dt}\right)^{2m+2} g(\xi).$$

(7.15) Für die Bernoulli-Polynome $B_n \in \mathbb{P}_n$ mit $B_0 \equiv 1$ und $B_n' = B_{n-1}$, $\int_0^1 B_n(t)dt = 0$ für $n \geq 1$ gilt $B_n(0) = B_n(1)$ für $n \geq 2$, $B_{2m+1}(0) = 0$ für $n = 2m+1 \geq 3$ ungerade, und
$B_{2m}(t) - B_{2m}(0) \neq 0$ für $t \in (0, 1/2)$ und $n = 2m \geq 2$ gerade.

7 Numerische Integration – Extrapolation und Romberg-Quadatur

(7.16) Für $f \in C^{2m+2}[a,b]$ und $h = \dfrac{b-a}{n}$ gilt

$$T_h(f) - I(f) = \sum_{j=1}^{m} c_j h^{2j} + O(h^{2m+2})$$

mit $c_j = b_j\Big(\Big(\dfrac{d}{dt}\Big)^{2j-1} f(a) - \Big(\dfrac{d}{dt}\Big)^{2j-1} f(b)\Big)$.

(7.17) Zu $T_{k0} = T_{h_k}(f)$ mit $h_k = 2^{-k}h$ für $k = 0,1,\dots,m$ definiere rekursiv

$$T_{ki} = T_{k,i-1} + \frac{1}{4^i - 1}\,(T_{k,i-1} - T_{k-1,i-1}), \qquad i = 1,\dots,m \text{ und } k = i,\dots,m.$$

Dann gilt: $|T_{mm} - I| \le Ch^{2m+2}\Big\|\Big(\dfrac{d}{dt}\Big)^{2m+2} f\Big\|_\infty$ mit $C > 0$ unabhängig von f,
d.h. T_{mm} ist für Polynome $P \in \mathbb{P}_{2m+1}$ exakt.

(7.18) Es gilt

a) $|I - T_{kk}| = O(h^{2k+2})$

b) $|I - T_{k,k-1}| = |T_{kk} - T_{k,k-1}| + O(h^{2k+2})$.

Ein schlecht konditioniertes Gleichungssystem

Wir betrachten das lineare Gleichungssystem $Ax = b$ mit der Hilbertmatrix

$$A = \left(\frac{1}{m+n+1}\right)_{m,n=0,\dots,N} \in \mathbb{R}^{N+1,N+1}$$

und der rechten Seite $b = \left((-1)^n\left(\log(2) + \sum_{m=1}^{n}(-1)^m\right)\right)_{n=0,\dots,N} \in \mathbb{R}^{N+1}$.

Die exakte Lösung lautet:

N	x_0	x_1	x_2	x_3	x_4	x_5	x_6
1	0.93	−0.48					
2	0.99	−0.80	0.33				
3	1.00	−0.94	0.66	−0.22			
4	1.00	−0.98	0.86	−0.53	0.15		
5	1.00	−1.00	0.95	−0.77	0.42	−0.11	
6	1.00	−1.00	0.98	−0.90	0.67	−0.32	0.07

(Beispiel aus Kress: Numerical Analysis)

Ein schlecht konditioniertes Gleichungssystem

Stören wir nun aber die rechte Seite geringfügig, indem wir log(2) nur bis auf 5 Nachkommastellen auswerten, so erhalten wir folgende Lösung:

N	x_0	x_1	x_2	x_3	x_4	x_5	x_6
1	0.93	−0.48					
2	0.99	−0.81	0.33				
3	1.00	−0.96	0.70	−0.25			
4	1.01	−1.16	1.63	−1.70	0.72		
5	1.06	−2.74	12.68	−31.16	33.87	−13.26	
6	1.39	−16.58	151.10	−584.81	1071.96	−926.77	304.50

Also: Eine geringfügige Störung der Daten führt zu einer großï¿½en Störung des Ergebnisses. Der Grund dafür liegt in der schlechten Kondition der Hilbertmatrix. Diese ist in der Spektralnorm:

N	2	3	4	5	6
$\kappa_2(A)$	19.28	524.06	$1.55e+04$	$4.77e+05$	$1.495e+07$

(Beispiel aus Kress: Numerical Analysis)

Invertierung der Hilbert-Matrix (Matlab)

```
>> H = hilb(3); IH = inv(H); IH(1,1:3)

ans = 9.0000000000003 -36.0000000000001 30.0000000000001

>> H = hilb(5); IH = inv(H); IH(1,1:3)

ans = 24.9999999999919 -299.999999999882 1049.99999999958

>> H = hilb(7); IH = inv(H); IH(1,1:3)

ans = 49.0000000578711 -1176.00000232576 8820.00002248178

>> H = hilb(9); IH = inv(H); IH(1,1:3)

ans = 80.9999332549633 -3239.99529943927 41579.919225348

>> H = hilb(11); IH = inv(H); IH(1,1:3)

ans = 120.91751059331  -7251.2942628623  141342.563141014
```

Ein schlecht konditioniertes Gleichungssystem

Wir können die Kondition verbessern, indem wir die Tikhonov-Regularisierung auf die Hilbertmatrix anwenden, d.h.

$$x^\alpha = (A^T A + \alpha I_N)^{-1} A^T b.$$

Regularisieren wir mit einem Parameter $\alpha = 10^{-10}$, so erhalten wir

N	x_0	x_1	x_2	x_3	x_4	x_5	x_6
1	0.93	−0.48					
2	0.99	−0.81	0.33				
3	1.00	−0.95	0.69	−0.24			
4	0.99	−0.89	0.47	0.06	−0.14		
5	1.00	−0.91	0.52	0.02	−0.18	0.04	
6	1.00	−0.94	0.58	0.08	−0.25	−0.17	0.20

Auslöschung bei Nullstellenberechnung

Wir betrachten die Gleichung

$$x^2 - 2px + q = 0,$$

deren Nullstellen durch

$$x = p \pm \sqrt{p^2 - q}$$

gegeben sind. Diese Berechnungsvorschrift ist aber für $p \gg q$ nicht zu empfehlen, da dann Auslöschung bei der betragsmäßig kleineren Nullstelle auftritt. Wählt man beispielsweise $p = 10^8$ und $q = 1$, so berechnet Matlab

$$x_1 = 2 * 10^8, \qquad x_2 = 0.$$

Die Auslöschung bei x_2 kann man umgehen, indem man zuerst die größere Nullstelle durch

$$x_1 = p + \mathrm{sign}(p)\sqrt{p^2 - q}$$

berechnet und dann (mit dem Satz von Vieta) die zweite Nullstelle durch

$$x_2 = \frac{q}{x_1}$$

erhält. Mit dieser Vorschrift berechnet Matlab die bessere Lösung

$$x_1 = 2 * 10^8, \qquad x_2 = 0.5 * 10^{-9}.$$

Einschließung der Eigenwerte durch eine Sturmsche Kette

Mit Hilfe der Sturmschen Kette kann man Einschließungen der Eigenwerte einer symmetrischen, irreduziblen Tridiagonalmatrix berechnen, indem man den Bisektionsalgorithmus zur Berechnung von Nullstellen einer Funktion auf die charakteristischen Polynome anwendet und die Einschließungseigenschaft der Nullstellen dieser Polynome ausnutzt. Als Beispiel betrachten wir

$$A = \text{tridiag}(-1, 2, -1) \in \mathbb{R}^{N,N}.$$

Die Eigenwerte sind gegeben durch

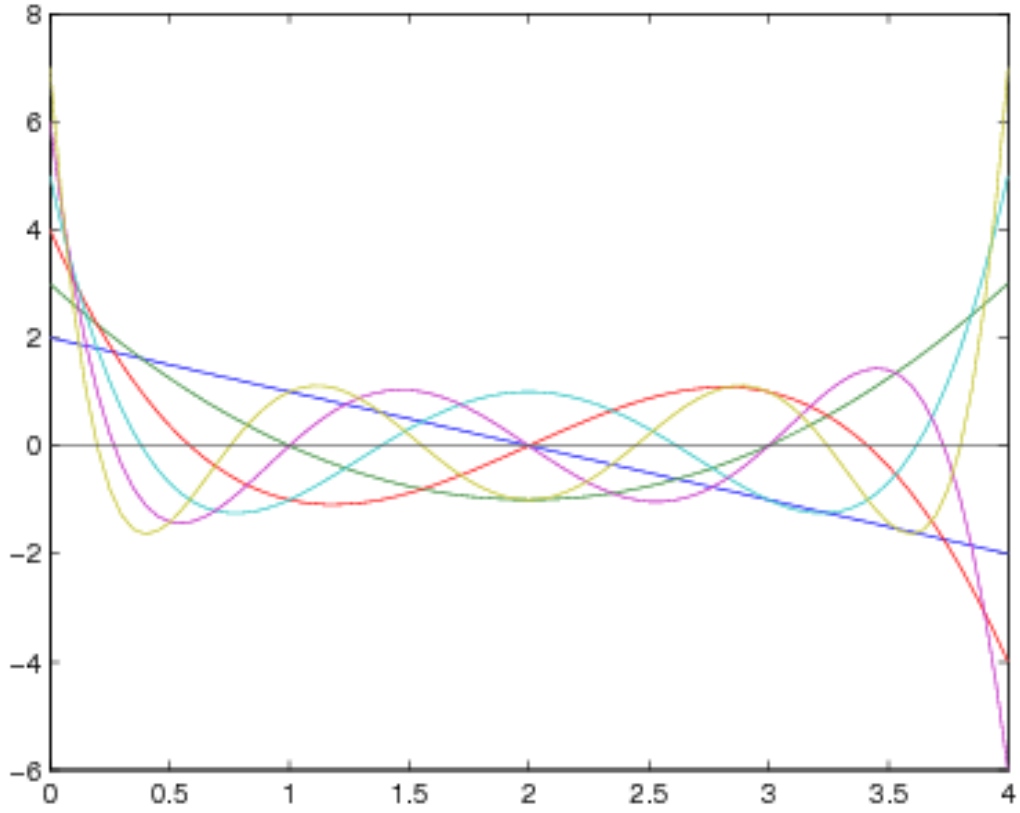
$$\lambda_n = 4\sin^2\left(\frac{n\pi}{2(N+1)}\right), \qquad n = 1, \ldots, N.$$

Bei einer vorgegeben Intervallbreite von 10^{-4} und $N = 6$ erhalten wir

n	linke Grenze	rechte Grenze	λ_n
1	0.19805908	0.19812012	0.19806226
2	0.75293642	0.75303431	0.75302040
3	1.55491605	1.55499216	1.55495813
4	2.44500844	2.44508306	2.44504187
5	3.24693201	3.24702692	3.24697960
6	3.80192175	3.80201367	3.80193774

Einschließung der Eigenwerte durch eine Sturmsche Kette

$W_6(0.0) = 0$,
also $0.0 \leq \lambda_1$
$W_6(4.0) = 6$,
also $\lambda_6 < 4.0$
$W_6(2.0) = 3$,
also $\lambda_3 < 2.0 \leq \lambda_4$
$W_6(1.0) = 2$,
also $\lambda_2 < 1.0 \leq \lambda_3$
$W_6(0.5) = 1$,
also $\lambda_1 < 0.5 \leq \lambda_2$
$W_6(3.0) = 4$,
also $\lambda_4 < 3.0 \leq \lambda_5$
$W_6(3.5) = 5$,
also $\lambda_5 < 3.5 \leq \lambda_6$



$$0.0 \leq \lambda_1 < 0.5 < \lambda_2 < 1.0 < \lambda_3 < 2.0 < \lambda_4 < 3.0 < \lambda_5 < 3.5 < \lambda_6 < 4.0$$

Symmetrischer impliziter QR-Algorithmus mit Wilkinson-Shift

Für symmetrische Matrizen (oBdA Hessenbergform) lässt sich der QR-Algorithmus so modifizieren, dass die QR-Zerlegung in jedem Iterationsschritt nur implizit durch $N-1$ Transformationen mit Givensrotationen durchgeführt werden muss. Durch den Wilkinson-Shift (siehe Golub, van Loan) erhalten wir außerdem kubische Konvergenz, im Gegensatz zur linearen Konvergenz des QR-Algorithmus' ohne Shift. Dabei können wir eine Toleranz für die verschwindenden Nebendiagonalelemente vorgeben.
Wir betrachten wieder

$$A = \text{tridiag}(-1, 2, -1) \in \mathbb{R}^{N,N}.$$

Für die Toleranz $\text{tol} = \varepsilon$ braucht der Algorithmus die folgende Anzahl von Iterationen:

N	Iterationen
100	281
200	532
500	1120
1000	2310

Das sind im Schnitt weniger als 3 Iterationen pro Eigenwert. Der maximale Fehler bei diesen Berechnungen beträgt zwischen $1e-13$ und $1e-14$.

Newton-Konvergenz mit Homotopie $F_t(x) = (1-t)F_0(x) + tF(x)$

```
epsilon = 1e-10;
S = 5;

x = [0;0]
for s=0:S;
  t = s / S;
  x = x - dF(x,t) \ F(x,t);
end
for k=1:10
  x = x - dF(x,1) \ F(x,1);
  if norm(F(x,1)) < epsilon
    break
  end
end
```

$\lvert F_0(x^k)\rvert_2$	0
$\lvert F_{0.2}(x^k)\rvert_2$	0.20000000000000
	0.00099256195800 2
$\lvert F_{0.4}(x^k)\rvert_2$	0.20099256195800 2
	0.00398434416087 8
$\lvert F_{0.6}(x^k)\rvert_2$	0.20398434416087 8
	0.00726936400093 0
$\lvert F_{0.8}(x^k)\rvert_2$	0.20726936400093 0
	0.01108655783936
$\lvert F_1(x^k)\rvert_2$	0.21108655783936
	0.01579686439489 1
	0.00012134175649 6
	0.00000000736032 3
	0.00000000000000 2

Newton-Verfahren Konvergenz/Divergenz 1

Wir betrachten drei Beispiele, die stellvertretend für die drei Fälle sind, die bei der Anwendung des Newton-Verfahrens auftreten können. Das Verfahren kann konvergieren, beschränkt divergieren und unbeschränkt divergieren. Für den ersten und letzten Fall sei für $x \in \mathbb{R}$

$$f(x) := \arctan(10x).$$

Die einzige Nullstelle ist $\tilde{x} = 0$. Führen wir das Newton-Verfahren zum Startwert $x_0 = 0.1$ aus, so erhalten wir folgende Werte:

k	x_k
0	0.100000000000000
1	−0.057079632679489 7
2	0.011685990399891 3
3	−0.000106102211704 5
4	0.000000000796310
5	0

Wir sehen, dass sich die Anzahl der richtigen Stellen mit jedem Schritt ungefähr verdoppelt, das deutet auf quadratische Konvergenz hin.

Newton-Verfahren Konvergenz/Divergenz 2

Nun wählen wir als Startwert $x_0 = 0.3$ und erhalten

k	x_k
0	0.3000000000000000
1	−0.9490457723982544
2	12.3999511178884152
3	−2390.5940294920856104

Im darauffolgenden Schritt ist die Ableitung im wesentlichen schon Null, die Folge divergiert unbeschränkt.

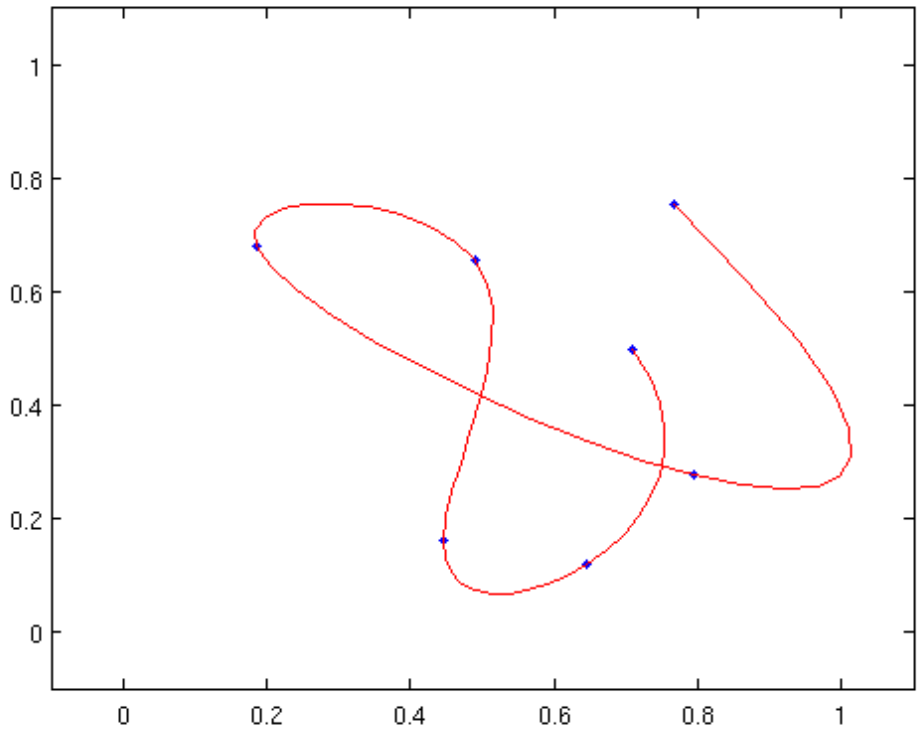
Für den Fall der beschränkten Divergenz betrachten wir für $x \in \mathbb{R}$ die Funktion

$$g(x) := x^3 - 2x + 2.$$

Starten wir mit $x_0 = 0$ so folgt $x_{2n+1} = 1$ und $x_{2n} = 0$, die Folge oszilliert.

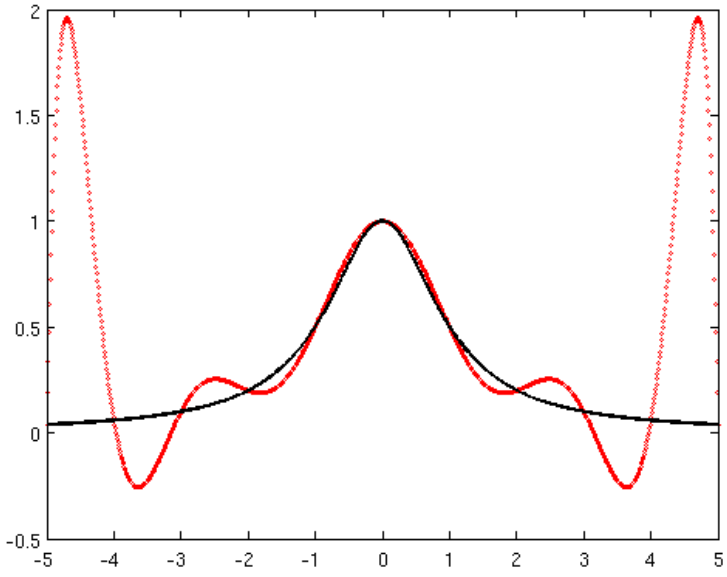
Spline-Interpolation mit Matlab

```
n = 7;
x = rand(n,1);
y = rand(n,1);
plot(x,y,'.')
axis([-0.1 1.1 -0.1 1.1]
t = 1:n;
ts = 1:1/10:n;
xs = spline(t,x,ts);
ys = spline(t,y,ts);
hold on;
plot(xs,ys,'r');
hold off;
```



Runge-Phänomen bei Interpolation

Wir betrachten das Interpolationsproblem für äquidistant verteilte Stützstellen. Wir würden erwarten, dass für eine genügend glatte Funktion die Folge der Interpolationspolynome, gehörig zu immer kleinerem Stützstellenabstand, in der Maximumsnorm gegen die Funktion konvergiert.
Dass dem nicht so ist zeigt zum Beispiel das sogenannte Runge-Phänomen für die Funktion $f\colon [-5,5] \to \mathbb{R}$,



$$f(x) := \frac{1}{1+x^2} \,.$$

Der maximale Fehler wird hier sogar immer größer, je höher der Grad des Interpolationspolynoms ist. Für den Grad 10 ist das zugehörige Interpolationspolynom eingezeichnet. Der Fehler wird offenbar durch die starke Oszillation verursacht, dies verschlimmert sich noch mit steigendem Polynomgrad.

Quadratur: Summierte Trapez- und Simpson-Regel

Approximiere $I(f) = \int_0^1 \frac{dt}{1+t^2}$ durch die summierte Trapez-Regel $I_{1,M}$ $(h = 1/M)$.

M	$I_{2,M}(f)$	$I(f) - I_{2,M}(f)$	$I_{2,M}(f) - I_{2,M/2}(f)$	$\frac{I_{2,M}(f) - I_{2,M/2}(f)}{I_{2,2M}(f) - I_{2,M}(f)}$
2	0.77500000	0.010398		
			0.007794	
4	0.78279411	0.002604		3.9908314
			0.001953	
8	0.78474712	0.000651		3.9997714
			0.000488	
16	0.78523540	0.000162		3.9999856
			0.000122	
32	0.78535747	0.000040		3.9999991
			0.000030	
64	0.78538799	0.000010		

Approximiere $I(f) = \int_0^1 \frac{dt}{1+t}$ durch die summierte Simpson-Regel $I_{2,M}$ $(h = 1/M)$.

M	$I_{2,M}(f)$	$I(f) - I_{2,M}(f)$	$I_{2,M}(f) - I_{2,M/2}(f)$	$\frac{I_{2,M}(f) - I_{2,M/2}(f)}{I_{2,2M}(f) - I_{2,M}(f)}$
2	0.6932539682	0.00010678769		
			0.0000994375	
4	0.6931545306	0.00000735009		14.45
			0.0000068778	
8	0.6931476528	0.00000047225		15.54
			0.0000004425	
16	0.6931472102	0.00000002972		15.87
			0.0000000278	
32	0.6931471824	0.00000000186		15.96
			0.0000000017	
64	0.6931471806	0.00000000011		

LR-Zerlegung

```
N = size(A,1);

for n=1:N-1
   A(n+1:N,n) = A(n+1:N,n)/A(n,n);
   A(n+1:N,n+1:N) = A(n+1:N,n+1:N) - A(n+1:N,n) * A(n,n+1:N);
end;

x = b;
for n=2:N
   x(n) = x(n) - A(n,1:n-1) * x(1:n-1);
end;

for n=N:-1:1
    x(n) = (x(n) - A(n,n+1:N)*x(n+1:N))/A(n,n);
end;
```

LR-Zerlegung (ohne Vektorisierung)

```
N = size(A,1);

for n=1:N-1
   % Berechnung der n-ten Spalte von L
   for m=n+1:N
     A(m,n) = A(m,n)/A(n,n);
   end;
   % keine Berechnung der n-ten Zeile von R erforderlich

   % Berechnung der Restmatrix
   for m=n+1:N
     for k=n+1:N
       A(m,k) = A(m,k) - A(m,n) * A(n,k);
     end;
   end;
end;
```

LR-Zerlegung mit Pivotsuche

```
N = size(A,1);    p = (1:N)';
for n = 1:N-1
   [r,m] = max(abs(A(n:N,n)));
   m = m+n-1;
   if abs(A(m,n))<eps
      error('*** ERROR *** LR-Zerlegung existiert nicht');
   end;
   if (m ~= n)
      A([n m],:) = A([m n],:);    p([n m]) = p([m n]);
   end;
   A(n+1:N,n) = A(n+1:N,n)/A(n,n);
   A(n+1:N,n+1:N) = A(n+1:N,n+1:N) - A(n+1:N,n)*A(n,n+1:N);
end;
x = b(p);
for n=2:N
   x(n) = x(n) - A(n,1:n-1)*x(1:n-1);
end;
for n=N:-1:1
    x(n) = (x(n) - A(n,n+1:N)*x(n+1:N))/A(n,n);
end;
```

Cholesky-Zerlegung

```
N = size(A,1);

for n=1:N
  A(n:N,n) = A(n:N,n) - A(n:N,1:n-1) * A(n,1:n-1)';
  A(n:N,n) = A(n:N,n) / sqrt(A(n,n));
end;

x = b;
for n=1:N
  x(n) = (x(n) - A(n,1:n-1) * x(1:n-1))/ A(n,n);
end;

for n=N:-1:1
  x(n) = (x(n) - A(n+1:N,n)' * x(n+1:N))/ A(n,n);
end;
```

LDL-Zerlegung (Cholesky-Variante ohne Wurzeln)

```
N = size(A,1);
for n=1:N
  v = A(n,1:n-1)';
  for m=1:n-1
    v(m) = v(m) * A(m,m);
  end;
  A(n,n) = A(n,n) - A(n,1:n-1) * v;
  A(n+1:N,n) = (A(n+1:N,n) - A(n+1:N,1:n-1) * v) / A(n,n);
end;

x = b;
for n=1:N
  x(n) = x(n) - A(n,1:n-1) * x(1:n-1);
end;
for n=1:N
  x(n) = x(n) / A(n,n);
end;
for n=N:-1:1
  x(n) = x(n) - A(n+1:N,n)' * x(n+1:N);
end;
```

Cholesky-Zerlegung (ohne Vektorisierung)

```
N = size(A,1);

for n = 1:N
  for k = 1:n-1
    A(n,n) = A(n,n) - A(n,k)^2;
  end
  A(n,n) = sqrt(A(n,n));
  for m = n+1:N
    for k = 1:n-1
      A(m,n) = A(m,n) - A(m,k) * A(n,k);
    end
    A(m,n) = A(m,n) / A(n,n);
  end;
end;
```

Berechnung der Householder-Vektoren

```
function [v,beta] = householder(y)
N = length(y);
s = y(2:N)' * y(2:N);
if N == 1
  s = 0;
end;
v = [1;y(2:N)];
if s == 0
  beta = 0;
else
  mu = sqrt(y(1)^2 + s);
  if y(1) <= 0
    v(1) = y(1) - mu;
  else
    v(1) = -s/(y(1) + mu);
  end;
  beta = 2*v(1)^2/(s + v(1)^2);
  v = v / v(1);
end;
return;
```

QR-Zerlegung

```
[M,N] = size(A);
for m = 1:min(N,M-1)
  [v,beta] = householder(A(m:M,m));
  if beta ~= 0
    w = beta * v' * A(m:M,m:N);
    A(m:M,m:N) = A(m:M,m:N) - v * w;
    A(m+1:M,m) = v(2:M-m+1);
  end;
end;

for m = 1:min(N,M-1)
  v = [1;A(m+1:M,m)];
  beta = 2 / (v' * v);
  if beta ~= 2
     b(m:M) = b(m:M) - beta*(v'*b(m:M)) * v;
  end;
end;
for n=min(N,M):-1:1
  x(n) = (b(n) - A(n,n+1:N) * x(n+1:N)) / A(n,n);
end;
```

Hessenberg-Transformation

```
N = size(A,1);
for n = 1:N-2
  [v,beta] = householder(A(n+1:N,n));
  if beta ~= 0
    w = beta * A(:,n+1:N) * v;
    A(:,n+1:N) = A(:,n+1:N) - w * v';
    w = beta * v' * A(n+1:N,n:N);
    A(n+1:N,n:N) = A(n+1:N,n:N) - v * w;
    A(n+2:N,n) = v(2:N-n);
  end
end
Q = eye(N);
for n = 1:N-2
  v = [1;A(n+2:N,n)];
  beta = 2 / (v' * v);
  if beta ~= 2
    w = beta * v' * Q(n+1:N,:);
    Q(n+1:N,:) = Q(n+1:N,:) - v * w;
  end
end
```

Berechnung der Givens-Parameter

```
function [c,s] = givens_param (a,b)

if b == 0.0
    c = 1.0;
    s = 0.0;
else
    if abs(b) > abs(a)
        t = -a/b;
        s = 1/sqrt(1+t^2);
        c = s*t;
    else
        t = -b/a;
        c = 1/sqrt(1+t^2);
        s = c*t;
    end
end

return;
```

Berechnung der Transformation mit einer Givensrotation

```
function A = givens_trafo (A,G,k)

N = size(A,1);

if k>1 && k<N-1 && N>=3
    ind = k-1:k+2;
elseif k==N-1 && N>=3
    ind = N-2:N;
elseif k==1 && N>=3
    ind = 1:3;
elseif N == 2
    ind = 1:2;
end

A([k k+1],ind) = G'*A([k k+1],ind);
A(ind,[k k+1]) = A(ind,[k k+1])*G;

return;
```

Symmetrischer impliziter QR-Algorithmus mit Shift

```
N = size(A,1); H = hessenberg(A);
while 1
    h00 = H(N-1,N-1);    h10 = H(N,N-1);    h11 = H(N,N);
    % Reduktions- und Abbruchkriterium
    if (abs(h10) < tol*abs(h11+h00)), N = N-1; end
    if N < 2, break; end
    % Bestimmung des Wilkinson-Shifts
    d = (h00 - h11)/2;
    if d ~= 0, mu = h11 + d - sign(d)*sqrt(d^2+h10^2);
    else, mu = h11 - h10; end
    a = H(1,1) - mu; b = H(2,1);
    % Impliziter QR-Schritt
    for k = 1:N-1
        [c,s] = givens_param(a,b);
        H = givens_trafo(H,[c,s;-s,c],k);
        if k<N-1, a = H(k+1,k); b = H(k+2,k); end
    end
end
lambda = diag(H);
```

Das Neville-Schema – Berechnung der Koeffizienten

```
function f = coeff(t,f)
N = length(t);
for k=1:N-1
  for n=N:(-1):k+1
    if t(n) ~= t(n-k)
      f(n) = (f(n) - f(n-1)) / (t(n) - t(n-k));
    end
  end
end
return

function y = eval_newton(t,b,x)
N = length(t);
y = b(1);
p = 1;
for n=2:N
  p = p * (x - t(n-1));
  y = y + b(n) * p;
end
return
```

Das Neville-Schema – Auswertung des Interpolationspolynoms

```
function y = eval_neville(t,f,x)
N = length(t);
for k=1:N-1
  for n=N:(-1):k+1
    if t(n) ~= t(n-k)
      f(n) = ((x-t(n-k))*f(n)+(t(n)-x)*f(n-1)) / (t(n)-t(n-k))
    end
  end
end
y = f(N);
return
```