Allgemein

x	0	$\frac{\pi}{6}$	$\frac{\pi}{4}$	$\frac{\pi}{3}$	$\frac{\pi}{2}$	π	$\frac{3\pi}{2}$	2π
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	$\sqrt{3}$	∞	0	$-\infty$	0

Grundbegriffe der Numerik

Gleitpunktzahlen und Maschinenzahlen

$\frac{m}{b^t} b^e = \pm 0.a_1 a_2 \cdots a_t \cdot b^e \quad mit \quad a_i \in \{0, \ldots, b-1\}$

Hierbei ist

- $s \in \{-1, 1\}$ das Vorzeichen
- $b \in \mathbb{N}$ die Basis
- $t \in \mathbb{N}$ die Genaugigkeit bzw. Anzahl signifikanter Stellen
- $e \in \mathbb{N}$ der Exponent
- $m \in \mathbb{N}$ die Mantisse
- $a_1 \neq 0$ („normalisiert“)

Bsp.: $-2 = -0.1000000 \cdot 2^2 (b = 2, t = 7)$ # Maschinenzahlen $= 2 \cdot (b-1) \cdot b^{t-1} \cdot a + 1$(hier: a=#Exponenten) (ohne $\pm\infty$ und NaN.)

Maschinengenauigkeit

$\epsilon_{b,t} = b^{-(t-1)}$ (d.h. ♯Maschinenzahl zwischen 1 und $1 + \epsilon_{b,t}$)
Bsp.: MATLAB: $\epsilon_{2,53} = 2^{-(53-1)} \approx 2 \cdot 10^{-16}$
Runden: Für $x = 2.387$ gilt $fl_{10,3} = 2.39 = +0.239 \cdot 10^1$
Wie viele Maschinenzahlen gibt es in $\mathbb{M}_{2,4,-3,3}$? Berechnen

Sie eps, x_{min} und x_{max}. Es gibt $2 \cdot 2^3 \cdot \overbrace{7}^{\#e} + 1 = 113$ Maschinenzahlen. $x_{min} = +\overbrace{1000}^{m}, \overbrace{-11}^{e} = \frac{1}{16}, x_{max} = +1111,11 = \frac{15}{16} \cdot 2^3 = \frac{15}{2}$, $eps(\hat{=}\epsilon_{2,4}) = 2^{-3} = \frac{1}{8}$

Kondition(Fehler bei Eingabe)

$\kappa_{abs}(x) = |f'(x)| \quad \kappa_{rel}(x) = \frac{\frac{|f'(x)|}{|f(x)|}}{x}$

- gut konditioniert für $\kappa_{rel} \leq 100$
- schlecht konditioniert für $\kappa_{rel} \geq 10^6$

Groß $\mathcal{O}$-Notation

$|f(x)| \leq c \cdot |g(x)| \quad \forall x > x_0 \wedge c > 0$

Stabilität(Rechenfehler)

$|\tilde{f}(x) - f(x)|$ = absoluter Fehler
$\frac{|\tilde{f}(x) - f(x)|}{|f(x)|}$ = relativer Fehler
Polynomauswertung bei f(a): $f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$

- naiv:
  function $y = peval(a_0, a_1, \ldots, a_n, a)$
  $y = a_0$;
  $for \quad k = 1 : n$
  $y = y + a_k \cdot a^k$;
  end
- Hornerschema function $y = horner(a_0, a_1, \ldots, a_n, a)$
  $y = a_n$;
  $for \quad k = n-1 : -1 : 0$
  $y = y \cdot a + a_k \cdot a^k$;
  end

LR-Zerlegung

$A \cdot x = b \Leftrightarrow x = A^{-1} \cdot b$ ist numerisch instabil→ LR-Zerlegung
Lösen mit $A = L \cdot R$:

1. Gauß auf A anwenden:
   $A = \begin{pmatrix} -1 & 2 & 3 \\ -2 & 7 & 4 \\ 1 & 4 & -2 \end{pmatrix} \quad II. - 2I. \quad III. + I.$
   $\Leftrightarrow \begin{pmatrix} -1 & 2 & 3 \\ +2(\hat{=}\frac{-2}{-1}) & 3 & -2 \\ -1 & 6 & 1 \end{pmatrix} \quad III. - 2II.$
   $\Leftrightarrow \begin{pmatrix} -1 & 2 & 3 \\ 2 & 3 & -2 \\ -1 & 2 & 5 \end{pmatrix}$
2. $\Rightarrow L = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & 2 & 1 \end{pmatrix}$ und $R = \begin{pmatrix} -1 & 2 & 3 \\ 0 & 3 & -2 \\ 0 & 0 & 5 \end{pmatrix}$
3. Löse
   - $L \cdot y = b$ durch Vorwärtseinsetzen $\Rightarrow y$, dann
   - $R \cdot x = y$ durch Rückwärtseinsetzen $\Rightarrow x$

Pivotisierung Tausche so, dass man durch die betragsmäßig größte Zahl teilen muss. Es gilt: $\boxed{P^{-1} = P}$

$Bsp.: \begin{pmatrix} 0 & 3 & 3 \\ -1 & 3 & 4 \\ -2 & 1 & 5 \end{pmatrix} \Rightarrow \begin{pmatrix} -1 & 3 & 4 \\ 0 & 3 & 3 \\ -2 & 1 & 5 \end{pmatrix}$ mit $P = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$
Löse $PAx = Pb$ bzw. $LR = Pb$

QR-Zerlegung mit $A^{m \times n}, m \geqslant n$

Ausgleichsproblem $Ax = b$ mit $|Ax - b| = min$
$\Rightarrow$ QR-Zerlegung mit $QRx = b \Leftrightarrow Q^T QRx = Q^T b$ mit fbox$Q^{-1} = Q^T$
Vorgehen:

1. $v_1 = a_1 + sgn(a_{11})|a|e_1$
2. $Q_1 = H_{v_1} = E_n - \frac{2}{v^T v} v v^T$
3. $\underbrace{Hv_1 A}_{R} = \begin{pmatrix} * & \cdots & * \\ \vdots & & \\ 0 & \tilde{A} & \end{pmatrix}$
4. Beginne von vorne mit Ã, wobei $a_2 = \begin{pmatrix} 0 \\ \tilde{a} \end{pmatrix} \Rightarrow$ erhalte alle Q
5. $Q_n Q_{n-1} \cdot \ldots \cdot Q_1 \cdot A = \underbrace{Q^T}_{hier\ bekommt\ man\ Q} \cdot A = R$
6. $\tilde{R}$=R ohne Nullzeilen
7. $Q_n Q_{n-1} \cdot \ldots \cdot Q_1 \cdot b = \begin{pmatrix} d_1 \\ d_2 \end{pmatrix} \Rightarrow d_1$
8. $\tilde{R}x = d_1 \Rightarrow x$

Bsp.: $A = \begin{pmatrix} 3 & 0 & 0 \\ 4 & 0 & 5 \\ 0 & 3 & -2 \\ 0 & 4 & 4 \end{pmatrix}$ und $b = \begin{pmatrix} 5 \\ 0 \\ 1 \\ -2 \end{pmatrix}$

1. $v_1 = \begin{pmatrix} 3 \\ 4 \\ 0 \\ 0 \end{pmatrix} + sgn(3)|a|e_1 = \begin{pmatrix} 3 \\ 4 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 5 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 8 \\ 4 \\ 0 \\ 0 \end{pmatrix}$
2. $Q_1 = E_m - \frac{2}{v^T v} v v^t = E_4 - \frac{1}{40} \begin{pmatrix} 64 & 32 & 0 & 0 \\ 32 & 16 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$
3. ...

Fixpunktiteration

z.B. $f(x) = \cos(x) - x \Leftrightarrow \Phi(x) = \cos(x)$
$Gegeben$: Φ ,Startwert $x_0 = s$
$\boxed{\Rightarrow x_{k+1} = \Phi(x_k)} \quad k = 0, 1, \ldots$
Φ stetige Abb. und Folge $(X_k)_k$ konvergiert für $x_0 \Rightarrow$ Grenzwert von (X_k) ist Fixpunkt von $\Phi(x) = x$
Beweis: $x = \lim_{k \to \infty} x_{k+1} = \lim_{k \to \infty} \Phi(x_k) = \Phi(\lim_{k \to \infty} x_k) = \Phi(x)$
Ein Iterationsverfahren $\Phi : X \to X$ mit FP x heißt

- global konvergent, falls $\forall x_0 \in X : (X_k) \to x$
- lokal konvergent, falls $\forall x_0 \in U : (X_k) \to x$, wobei U Umgebung von x ist
- Kontraktion, falls $\exists L \in [0, 1)$ mit $||\Phi(x) - \Phi(y)|| \leq L||x - y||$, L=Lipschitzkonstante, für $L < 1$ gilt **strikte Kontraktion**

Konvergenzsätze:

- **Globaler Konvergenzsatz:** Gilt für $f : D \to \mathbb{R}^n$, wenn
  1. $D \subseteq \mathbb{R}^n$ **abgeschlossen**
  2. $f(D) \subseteq D$
  3. (strikte) Kontraktion mit $L < 1$

  $\Rightarrow \exists$ einen eindeutigen FP und es gilt:
  - $||x_k - x|| \leq \frac{\phi^k}{1-\phi} ||x_1 - x_0||$
    A-priori-Fehlerabschätzung
    # benötigter Iterationen mit Genauigkeit ϵ: $k \geq \frac{ln \frac{\epsilon(1-\phi)}{||x_1 - x_0||}}{ln \phi}$
  - $||x_k - x|| \leq \frac{\phi}{1-\phi} ||x_k - x_{k-1}||$
    A-posteriori-Fehlerabschätzung
- **Lokaler Konvergenzsatz:** $X \subseteq \mathbb{R}^d$ offen, stetig diffbar,x FP von Φ mit $||\underbrace{J_\phi(x)}_{Jacobimatrix}|| < 1 \rightarrow$ lokal konvergent
- **Lokaler,normunabhängiger Konvergenzsatz:** $X \subseteq \mathbb{R}^d$ offen, stetig diffbar,x FP von ϕ mit $\boldsymbol{\rho}(J_\phi(x)) < 1 \rightarrow$ lokal konvergent, wobei
  ρ(Matrix) ist betragsmäßig größter EW von Matrix (Spektralradius)

Iterative Verfahren für LGS

Gegeben: $Ax = b$
Formuliere das LGS als ein Fixpunktproblem:
$\boxed{A = M - N}$, mit A,M,N $\in \mathbb{R}^{n \times n}$ mit invertierbarem M, so ist das Lösen von $Ax = b$ gleichwertig zur Fixpunktbestimmung $\boxed{\phi(x) = x\phi(x) = M^{-1}b + M^{-1}Nx}$
wobei $x_0 = s, x_{k+1} = \phi(x_k)$ und für $\rho(M^{-1}N) < 1$ konvergiert.
$\Rightarrow$ Verschiedene Verfahren für verschiedene Wahlen von M und N:

Jacobi-Verfahren

$A = \overbrace{D}^{M} - \overbrace{(L + R)}^{N} = \begin{pmatrix} D & & & -R \\ & D & & \\ & & D & \\ -L & & & D \end{pmatrix}$ mit Startvektor x_0

$x_{k+1} = D^{-1}b + D^{-1}(L + R)x_k = \overbrace{D^{-1}b}^{c} + \overbrace{D^{-1}(D - A)x_k}^{T}$
Vorgehen: $\boxed{x_{k+1} = Tx_k + c}$
Konvergenz, falls A **strikt diagonal dominant.**
Bsp.: $A = \begin{pmatrix} 5 & -2 & 1 \\ 1 & 3 & 0 \\ -3 & -5 & 9 \end{pmatrix}$ $I. : |5| > |-2| + |1|, \forall$ Zeilen

Gauß-Seidl-Verfahren

$A = \overbrace{(D - L)}^{M} - \overbrace{R}^{N}$
Explizit: $x_{k+1} = (D - L)^{-1}b + (D - L)^{-1}Rx_k$
Konvergiert für

- strikt diagonal dominates A
- positiv definites A (alle EW>0 oder Hauptminoren positiv)

Vorgehen: $A = \begin{pmatrix} 15 & 2 \\ 1 & -4 \end{pmatrix}, b = \begin{pmatrix} -1 \\ -9 \end{pmatrix}, x_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$

1. $T = \begin{pmatrix} 0 & \frac{-2}{15} \\ \frac{1}{4} & 0 \end{pmatrix}, c = \begin{pmatrix} \frac{-1}{15} \\ \frac{9}{4} \end{pmatrix}$
2. $x_1^1 = (0 \quad \frac{-2}{15}) \begin{pmatrix} x_0^1 \\ x_0^2 \end{pmatrix} + \frac{-1}{15} = \frac{-1}{15}$
   $x_1^2 = (\frac{1}{4} \quad 0) \begin{pmatrix} x_0^1 \\ x_0^2 \end{pmatrix} + \frac{9}{4} = \frac{67}{30}$
   $\Rightarrow x_1 = \begin{pmatrix} \frac{-1}{15} \\ \frac{67}{30} \end{pmatrix}$
3. $x_1^1 = (0 \quad \frac{-2}{15}) \begin{pmatrix} \frac{-1}{15} \\ \frac{67}{30} \end{pmatrix} + \frac{-1}{15} = \frac{-82}{225}$
   $x_2^2 = (\frac{1}{4} \quad 0) \begin{pmatrix} \frac{-82}{225} \\ \frac{67}{30} \end{pmatrix} + \frac{9}{4} = \frac{1943}{900} \quad \ldots$

SOR-Verfahren

$\boxed{x_{k+1} = (E_n - \omega D^{-1}L)^{-1}((1 - \omega)E_n + \omega D^{-1}R)x_k + \omega(E_n - \omega$
Konvergenz für:

- $0 < \omega < 2$
- A positiv definit

Optimales ω

- $\rho(M^{-1}N) < 1$ und
- alle EW $\lambda_1, \ldots, \lambda_n$ von $M^{-1}N$ reell und im Intervall $(-\infty, 1)$

$\Rightarrow \omega_{opt}$ für relaxiertes Verfahren
$\phi_\omega(x) = (\omega M^{-1}N + (1 - \omega)E_n)x + \omega M^{-1}b$ ist:

- $\boxed{\omega_{opt} = \frac{2}{2 - \lambda_1 - \lambda_n}}$ -für Jacobi
- $\boxed{\omega_{opt} = \frac{2}{1 + \sqrt{1 - \lambda_n^2}}}$ -für Gauß-Seidl
- $\omega = 1 \Rightarrow$ man erhält wieder Gauß-Seidl
- $1 < \omega < 2$ nennt man **Überrelaxierung**

Christian Obermaier und Tim Huber

Bemerkung:
Für $A \in \mathbb{R}^{\times\times\times}$ braucht jede Iteration der drei Verfahren $\mathcal{O}(n)$ Operationen

Nichtlineare Gleichungssysteme

Bisektionsverfahren

Sei $f : [a_0, b_0] \to \mathbb{R}$ stetig mit $f(a_0)f(b_0) < 0$, so erhalte die Nullstelle $x^\star$ folgendermaßen:

1. $x_k = \frac{1}{2}(a_k + b_k) und f(x_k)$
2. Setze
   - $a_{k+1} = a_k$ und $b_{k+1} = x_k$, falls $f(a_k)f(x_k) < 0$
   - $a_{k+1} = x_k$ und $b_{k+1} = b_k$, falls $f(x_k)f(b_k) < 0$
3. Abbruch der Iteration, falls $b_k - a_k < \epsilon$

Weiterhin gilt:
$$|x_k - x^\star| \leq b_k - a_k \leq \frac{1}{2^k}|b_0 - a_0|$$

- $\Rightarrow$ Globale Konvergenz(d.h. $|x_k - x^\star| \to 0$ mit $k \to \infty$ für beliebiges Anfangsintervall)
- $\Rightarrow$ maximal lineare Konvergenz, da der Fehler $|x_k - x^\star|$ maximal linear fällt.

Newton-Verfahren

Eindimensional

Ist $f : I \to \mathbb{R}$ stetig diffbar und $x^\star \in I$ mit
$f(x^\star) = 0$ und $f'(x^\star) \neq 0$,
so existiert eine Umgebung U von $x^\star$, so dass die Iteration
$$x_0 \in U \text{ und } x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, \quad k = 0, 1, \ldots$$
für jedes $x_0 \in U$ **quadratisch** gegen $x^\star$ konvergiert.
Ist f 2-mal stetig diffbar in U von $x^\star$, so gilt:
$$x_{k+1} - x^\star = \frac{1}{2}\frac{f''(\xi_k)}{f'(x_k)}(x_k - x^\star)^2 \text{ für } \xi_k \in U$$

Mehrdimensional

Ist $f : X \subseteq \mathbb{R}^\times, X$ offen und konvex, 2-mal stetig diffbar mit nichtsingulärer Jacobimatrix $J_f(x) \forall x \in X$, so konvergiert die Iteration

$$\boxed{x_{k+1} = x_k - (J_f(x_k))^{-1} f(x_k) =: \phi(x_k)}$$

lokal quadratisch gegen eine Nullstelle von f.
Bem.: Beim **vereinfachten** Newton-Verfahren wird J_f für mehrere Schritte verwendet.

Abbruch des Newtonverfahren

Abbruch, wenn

- $x^\star$ hinreichend genug approximiert ist oder
- die Iteration voraussichtlich nicht konvergiert

Lösbarkeit und Kondition des Nullstellenproblems

Lokale Eindeutigkeit
Ist $f : x \to \mathbb{R}^\times$ stetig diffbar mit regulärer Jacobimatrix $J_f(x^\star)$ für $x^\star \in X$ mit $f(x^\star) = 0$, so ist $x^\star$ lokal eindeutig, d.h.es gibt eine Umgebung U von $x^\star$, so dass $x^\star$ der einzige Punkt mit $f(x^\star) = 0$ in U ist.

Kondition
Ist $f : x \to \mathbb{R}^\times$ stetig diffbar mit regulärer Jacobimatrix $J_f(x^\star)$ für $x^\star \in X$ und zu $\delta f \in \mathbb{R}^\times$ ist $x^\star(\delta f)$ die Lösung von $f(x) + \delta(f) = 0$.

$$\Rightarrow \boxed{\kappa_{abs} = ||J_f^{-1}(x^\star)||}$$

Numerik gewöhnlicher DGLs

Einschrittverfahren

Näherungslösungen x_k für exakte Lösung $x(t_k)$ des AWP $\dot{x} = f(t, x)$ mit $x(t_0) = x_0$
an den Stellen $t_k = t_0 + k \cdot h$ mit $k = 0, 1, \ldots, n$ und $h = \frac{t - t_0}{n}$ für ein $n \in \mathbb{N}$.

explizites Euler-Verfahren:
$x_{k+1} = x_k + h \cdot f(t_k, x_k), \quad k = 0, 1, 2, \ldots$
Mittelpunktsregel:
$x_{k+1} = x_k + h \cdot f(\frac{t_k + t_{k+1}}{2}, \frac{x_k + x_{k+1}}{2}), \quad k = 0, 1, 2, \ldots$ **implizites Euler-Verfahren:**
$x_{k+1} = x_k + h \cdot f(t_{k+1}, x_{k+1}), \quad k = 0, 1, 2, \ldots$

Optimierung

Gradientenverfahren
Geg.: C^1-Funktion $f : \mathbb{R}^d \to \mathbb{R}$ und Startvektor $x_0 \in \mathbb{R}^d$

- Bestimme Abstiegsrichtung: $v_k = -\nabla f(x_k)$
- Schrittweite $f(x_k + h_k v_k) < f(x_k)$
  mit Armijo:$\gamma \in (0, 1); h_k \in \{1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \ldots\}$
  $f(x_k + h_k v_k) \leq f(x_k) + h_k \gamma \nabla f(x_k)^T v_k$
- $x_{k+1} = x_k + h_k v_k$

Konvergenz

- $f(x_{k+1}) < f(x_k) \forall k$
- alle Häufungspunkte von $(x_k)_k$ sind stationäre Punkte von f

Holomorphe Funktionen

Komplexe Funktionen
Gebiete

- zu jedem $z \in G \quad \exists \quad \epsilon > 0$ mit $B_\epsilon(a) \subseteq G$
- G zusammenhängend (Streckenzug zw. zwei Elementen von G existiert)

Wichtige Formeln

$e^{iz} = \cos(z) + i\sin(z)$
$e^z \cdot e^w = e^{z+w}$
$\cos(-z) = \cos(z)$
$\cos(z) = \frac{e^{iz}+e^{-iz}}{2}$
$\cos(z + w) = \cos(z)\cos(w) - \sin(z)\sin(w)$
$\cos(z) = 0 \Leftrightarrow z = (k + 0.5)\pi, k \in \mathbb{Z}$
$\cos(z + 2\pi) = \cos(z)$
$\cos(z) = \cos(x)\cosh(y) - i\sin(x)\sinh(y)$
$\cos^2(z) + \sin^2(z) = 1$
$\cosh(z) = \frac{1}{2}(e^z + e^{-z})$
$\cosh(it) = \cos(t)$
$\sin(-z) = -\sin(z)$
$\sin(z) = \frac{e^{iz}-e^{-iz}}{2i}$
$\sin(z + w) = \sin(z)\cos(w) + \sin(w)\cos(z)$
$\sin(z) = 0 \Leftrightarrow z = k\pi, k \in \mathbb{Z}$
$\sin(z + 2\pi) = \sin(z)$
$\sin(z) = \sin(x)\cosh(y) + i\cos(x)\sinh(y)$
$\sin(z + \frac{\pi}{2}) = \cos(z)$
$\sinh(z) = \frac{1}{2}(e^z - e^{-z})$
$\sinh(it) = i\sin(t)$

Kriterium für Holomorphie

Gegeben: Reelifizierte Funktion $f(x, y) = u(x, y) + i \cdot v(x, y)$
$f(x, y)$ holomorph, wenn $u(x, y)$ und $v(x, y)$ stetig part. diffbar und Cauchy-Riemann'sche DGLen erfüllt:

$$\boxed{u_x(x, y) = v_y(x, y) \text{ und } u_y(x, y) = -v_x(x, y)}$$

Harmonische Funktionen

geg.: $u(x, y)$; ges.: $v(x, y)$
Prüfe: $u_{xx} = -u_{yy}$
$v(x, y) = \int u_x dy$ mit Konstante $g(x)$
$v_x = -u_y \Rightarrow g_x(x)$ integriere $g_x(x)$

Wichtige Algorithmen

Bruch als Binärzahl
```
a = 1/11;
for i = 1 : 30;
if a ≥ 1
   a = a − 1;
   x(i) = 1;
else x(i) = 0;
end
a = 2 ∗ a;
end
```

Newton optimiert
```
function [x,xvec]=
newtonopt(f,Df,Hf,x,TOL)
weiter = 1;
xvec=[x];
while weiter
deltax= Hf(x)\Df(x);
x=x-deltax;
xvec=[xvec x];
weiter = norm(deltax) > TOL;
end
```

Householdermatrix bestimmen
```
function [Q,R,A] = householder(A)
[m, n] = size(A);
Dimensionspruefung der Matrix A
if n > m
disp('Matrix A muss mehr Zeilen als Spalten besitzen')
return
end
Ausgabe Initialisieren
Q = eye(size(A));
V = zeros(size(A));
R = zeros(n);
for k = 1:n
Spiegelungsvektor v konstruieren
x = A(k:m,k);
tmp = sign(x(1))*norm(x);
v = x;
v(1) = v(1) + tmp;
v = v/norm(v);
Spiegelung durchfuehren
A(k:end,k:end) = A(k:end,k:end) - 2*v*(v'*A(k:end,k:end));
Spiegelvektoren merken
V(k:end,k) = v;
end Die Matrix Q berechnen. for k = n:-1:1 v = V(k:end,k);
Q(k:end,:) = Q(k:end,:) - 2*v*(v'*Q(k:end,:)); end R ist ja gerade
der rechte obere Teil von A R = triu(A);
```

Jacobiverfahren
```
function [x,relerr,niter] = jacobi(A,b,x0,tol,maxiter)
relerr=inf;
niter=1;
M=diag(diag(A));
N=M-A;

while relerr >= tol & niter < maxiter
    x=M\(b+N*x0);
    relerr=norm(x-x0, inf)./norm(x,inf);
    x0=x;
    niter = niter+1;
end
```

SOR-Verfahren
```
function [x,relerr,niter] = SOR (A,b,x0,w,tol,maxiter)
relerr = inf;
niter = 1;
L=-tril(A,-1);
R=-triu(A,1);
D=diag(diag(A));
M = 1/w*D-L;
N = (1/w -1)*D+R;
while relerr qeq tol & niter < maxiter,
x=M  (b+N*x0);
relerr = norm( x-x0,inf )/norm( x,inf );
x0 = x;
niter = niter+1;
end
```

Bisektionsverfahren
```
function [tm,n]=bisection (f,t0,t1,TOL)
f0=feval(f,t0);
f1=feval(f,t1);
n=0;
if (f0==0), tm=t0; return; end
if (f1==0), tm=t1; return; end
if (f0*f1>0)
error('f hat bei t0 und t1 das gleiche Vorzeichen!');
end
while (1)
tm=t0+0.5*(t1-t0);
if (t1-t0<TOL), break;end
fm=feval(f,tm);
if (fm==0), break;end
if (f0*fm>0), t0=tm;f0=fm;else t1=tm;f1=fm;end
n=n+1;
end
end
```

Newtonverfahren
```
function [ x,xvec,deltax ] = newtonverf( f,Df,x,TOL)
weiter = 1;
xvec=[];
while weiter
deltax= Df(x)\f(x);
lS= Df(x)\f(x-deltax);
lS: linke Seite des natuerlichen Monotonietests
x=x-deltax;
xvec=[xvec x];
weiter = norm(deltax) > TOL & norm(lS) < norm (deltax);
end
```

Ein-/Mehrschrittverfahren(Runge-Kutta)
```
function aufgabe
h = 0.1;
x0 = 0;
y0 = 0.5;
x1 = 1.8;
Schrittweite
Start an Stelle x0 = 0
Anfangswert y0 an Stelle x0
Stop an Stelle x1 = 1.8
u_ euler = y0;
u_rk = y0;
for x=linspace(x0, x1-h, (x1-x0)/h)
u_ euler = u_ euler + h*f(x, u_ euler); Explizites Euler-Verfahren
Klass. Runge-Kutta Verfahren
k1 = f(x, u_ rk);
k2 = f(x+h/2, u_ rk+h/2*k1);
k3 = f(x+h/2, u_ rk+h/2*k2);
k4 = f(x+h, u_ rk+h*k3);
u_ rk = u_ rk + h*(k1/6+k2/3+k3/3+k4/6);
end
format long
u_ exakt = x1 + 1/(2-x1)
Ausgabe der Ergebnisse
u_ euler
fehler_ euler = abs(u_ euler-u_ exakt)
u_ rk
fehler_ rk = abs(u_ rk-u_ exakt)
end
function f = f(x, y)
f = 1 + (y-x)^2;
end
```

Gradientenverfahren
```
function [x,xvec]=gradientenverf(f,Df,x,gamma,TOL)
weiter = 1;
xvec=[x];
while weiter
fx=f(x);
Dfx=Df(x);
h=1;
while (f(x-h*Dfx) > fx-gamma*h*Dfx'*Dfx)
h=h/2;
end
x=x-h*Dfx;
xvec=[xvec x];
weiter = norm(h*Dfx) > TOL;
end
```

```
function [ x ] = fixpunktiteration( phi, s, n )
x=s;
for k=1:n
    x=phi(x)
end
```

```
function [ x,relerr,niter ] = gausseidel( A,b,x0,tol,maxiter )
relerr=inf;
niter=1;
M=tril(A);
N=M-A;

while relerr >= tol & niter < maxiter
    x=M\(b+N*x0);
    relerr=norm(x-x0, inf)./norm(x,inf);
    x0=x;
    niter = niter+1;
end
```

Die LR-Zerlegung ohne Pivotisierung
```
function [L,R] = LR(A)
  [n,n] = size(A);
  for j =1:n-1
  I = j+1:n;
  A(I,j) = A(I,j)/A(j,j);
  A(I,I) = A(I,I)-A(I,j)*A(j,I);
  end
  R = triu(A);
  L = eye(n,n) + tril(A,-1);
```

Die LR-Zerlegung mit Pivotisierung
```
function [A,p] = LR_pivot(A)
  [n,n] = size(A);
  p = 1:n;
  for j =1:n-1
  [mx,k] = max(abs(A(p(j:n),j)));
  k = k + (j-1);
  p([j k]) = p([k j]);
  I = j+1:n;
  A(p(I),j) = A(p(I),j)/A(p(j),j);
  A(p(I),I) = A(p(I),I)-A(p(I),j)*A(p(j),I);
  end
  R = triu(A);
  L = eye(n,n) + tril(A,-1);
```

Christian Obermaier und Tim Huber