4. Semester Formelsammlung

Maschinengenauigkeit:
Abstand von 1 zur nächst größeren Zahl
64 Bit:$eps \approx 2 * 10^{-16}$
32 Bit: $eps \approx 1 * 10^{-7}$

Kondition:
Absolut: $\|f(x) - \tilde{f}(x)\| \leq \kappa_{abs} * \|x - \tilde{x}\|$
Falls f Diffbar: $\kappa_{abs}(x) = |f'(x)|$

Relativ: $\frac{\|f(x)-f(x+\Delta)\|}{\|f(x)\|} \leq \kappa_{rel} * \frac{\|\Delta\|}{\|x\|}$
Falls f Diffbar: $\kappa_{rel}(x) = \frac{|f'(x)|}{|f(x)|} * x$

Lin. Gleichungssystem $Ax = b$
$\kappa_{\text{abs}} = \|A^{-1}\| \quad \kappa_{rel} = \|A^{-1}\| * \|A\|$

Matrixnormen: $\|A\|_\infty = Zeilensummennorm$
$\|A\|_2 = \sqrt{\lambda_{max}(A^T A)}$

Rückwärtsstabilität:
$\tilde{f}$ für Problem f heißt Rückwärts stabil, falls für jedes x ein $\tilde{x}$ existiert mit $\tilde{f}(x) = f(\tilde{x})$ und $\frac{\|\tilde{x}-x\|}{\|x\|} \leq eps$

LR Zerlegung:
$R = Gauß\ elimination\ von\ A$
$L_{ij} = \frac{a_{ij}}{a_{ii}}$ (aber nur aktuelle Spalte!)
Pivotisierung: Pivotelement möglichst groß wählen/Zeilen vertauschen
Permutationsmatrix: Zeile a und b vertauschen: Einheitsmatrix Spalte a und b vertauschen
$\Rightarrow PA = LR \quad Ax = b \rightarrow Ly = Pb \quad Rx = y$

QR Zerlegung:
$\tilde{A} = A$
1. $\tilde{A} = (k-1)te\ Untermatrix\ von\ A$
2. $a = 1.Spalte\ von\ \tilde{A}$
3. $v = a + sign(a_1) * \|a\| * e_1$
4. $H_v = I - 2\frac{vv^T}{\|v\|^2}$
5. $Q_k = \begin{pmatrix} I_{k-1} & 0 \\ 0 & H_v \end{pmatrix} \quad \tilde{A} = \tilde{A} * H_v$

$\tilde{A} = R \qquad Q_1 Q_2 \dots Q_k = Q$

$Ax = b \rightarrow Q^T b = y \quad Rx = y$

Lin Ausgleichsprobleme:
Finde x so dass $\|r\|^2 = \|b - Ax\|^2$ minimal
Polynomial Fitting: $A = \begin{pmatrix} 1 & t_1 & t_1^2 & \dots & t_1^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & t_m & t_m^2 & \dots & t_m^{n-1} \end{pmatrix}$
x löst Problem $\Leftrightarrow$ x löst Normalengleichung
$A^T Ax = A^T b$

Kondition:
$\vartheta = \sin^{-1}\left(\frac{\|r\|}{\|b\|}\right) \quad \kappa = \begin{cases} \kappa \leq \frac{\kappa(A)}{\cos(\vartheta)} & f: \mathbb{R}^m \rightarrow \mathbb{R}^n, b \rightarrow x \\ \kappa \leq \kappa(A) + \kappa(A)^2 \tan(\vartheta) & f: \mathbb{R}^{mxn} \rightarrow \mathbb{R}^n, A \rightarrow x \end{cases}$
Lösung: QR Zerlegung
$Q^T b = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \quad R = \begin{pmatrix} \tilde{R} \\ 0 \end{pmatrix}$
$x = \tilde{R}^{-1} b_1 \qquad \|b - Ax\| = \|b_2\|$

Fixpunktiteration:
Kontraktion falls $\sup|\varphi'(x)| < 1$ oder $\|\varphi(x) - \varphi(y)\| \leq \vartheta\|x - y\|$ mit $0 \leq \vartheta < 1$
Branarchscher Fixpunktsatz: falls $\varphi(x)$ eine Kontraktion, abgeschlossen und eine Selbstabbildung konvergiert die Iteration für jeden Startwert

Iterative Lösungsverfahren:
$\varphi(x) = Nb + Mx$
$\|M\| < 1$ ist hinreichend für Konvergenz
Jacobi Verfahren:
$A = \underbrace{D}_{B} - \underbrace{(L+R)}_{C}$
$x^{(k+1)} = B^{-1}\left(b + Cx^{(k)}\right) = D^{-1}\left(b + (L+R)x^{(k)}\right)$
Konvergenz falls A strikt Diagonaldominant
Gaus Seidel Verfahren:
$A = \underbrace{(D-L)}_{B} - \underbrace{R}_{C}$
$x^{(k+1)} = (D-L)^{-1}\left(b + Rx^{(k)}\right)$
Konvergenz falls A strikt Diagonaldominant oder s.p.d.
Dämpfung: $x^{(k+1)} = \omega\varphi\left(x^{(k)}\right) + (1-\omega)x^{(k)}$
$\omega_{opt} = \frac{2}{2-\lambda_1-\lambda_m}$ wobei $\lambda_1, \lambda_m < 1$ größter und kleinster Eigenwert von M sind

Nichtlineare Gleichungen:
Kondition: $\kappa = \|(f'(x))^{-1}\|$
Bisektionsverfahren: $f(a_0) * f(b_0) < 0$ Globale Konvergenz
1. $x_k = \frac{1}{2}(a_k + b_k)$
2. $\begin{cases} a_{k+1} = a_k & b_{k+1} = x_k \quad falls\ f(a_k) * f(b_k) < 0 \\ a_{k+1} = x_k & b_{k+1} = b_k \qquad\qquad sonst \end{cases}$
3. Abbruch falls $|b_k - a_k| < \varepsilon$

Newton Verfahren: $f(x)$ muss diffbar sein
$k_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$
Falls $f(\bar{x}) = 0, f'(\bar{x}) \neq 0$ lokale quadratische Konvergenz
Mehrdimensional: $x^{k+1} = x^{(k)} - J_f^{-1}\left(x^{(k)}\right) + f\left(x^{(k)}\right)$

Optimierung:

$\nabla f(x^*) = 0 \quad x^*$heißt stationärer Punkt
$\nabla f(x^*) = 0$ und $\nabla^2 f(x^*) =$ Pos. Definit (Alle EW > 0), dann ist x^* ein lokales Minimum
Newtonverfahren zur Bestimmung von x^*: $x_{k+1} = x_k - \nabla^2 f(x_k) + \nabla f(x_k)$
Gradientenverfahren:
1. Stopp falls $\nabla f(x^k) \cong 0$
2. Abstiegsrichtung bestimmen: $d^k = -\nabla f(x^k)$
3. Maximale Schrittweite:
   a. Exakt: $\min_{s>0} f(x^{(k)} + \sigma * d^{(k)})$
   b. Armijo setze $\sigma = 1,\ \gamma \in \left(0, \frac{1}{2}\right)$;
   Falls $f\left(x^k + \sigma_k d^k\right) - f(x_k) < \gamma\sigma_k + \nabla f\left(x^k\right)^T d^k$ halbiere σ.
4. $x^{k+1} = x^k + \sigma_k d^k$

Komplexe Differentiation:
$f: U \rightarrow \mathbb{C}$ heißt komplex diffbar falls $\lim_{z \rightarrow z_0} \frac{f(z)-f(z_0)}{z-z_0} =: f'(z_0)$
Falls f überall auf U diffbar, nennt man f analytisch
Rechenregeln wie bei normaler Differentation

$f(z) = f_1(z) + i\, f_2(z)$
Cauchy-Riemannsche Diffgl.: $\partial_1 f_1(z) = \partial_2 f_2(z)\,, \quad \partial_1 f_2(z) = -\partial_2 f_1(z)$
$f(z)$Ist in U diffbar falls CR-Difflg. Gillt.

Konstanten, Polynome, Potenzreihen (sin, cos, exp) und Rationale Fkt sind analytisch

Komplexe Integration
Rechenregeln wie für reelle Integrale
Kurvenintegral: $\gamma: [a,b] \rightarrow \mathbb{C}$
$\int_\gamma g(s)\, ds = \int_a^b g\big(\gamma(t)\big)\|\dot{\gamma}(t)\|\, dt$
Cauchy-Integrationsformel: $f: U \rightarrow \mathbb{C}$ analytisch, γ geschlossen mit Innerem in ganz U Dann gilt für jeden Punkt z im Inneren von U:
$f(z) = \frac{1}{2\pi i}\int_\gamma \frac{f(\zeta)}{\zeta - z}\, d\zeta$
Für $f: U \rightarrow \mathbb{C}$ analytisch, $\{|z-a| \leq r\} \subset U, a \in U, r > 0$
$f(z) = \sum_{n=0}^{\infty} c_n (z-a)^n$ mit $c_n = \frac{1}{2\pi i}\int_{|\zeta-a|=r} \frac{f(\zeta)}{(\zeta-z)^{n+1}}\, d\zeta$

Auch wichtig: Schrödingers Katze

4. Semester Formelsammlung-Anhang

Aufwand:

Laplace Determ.	$n * n! - 1$
Skalarprodukt	$2n - 1$
Vorwärts/Rückwärtssubstitution	n^2
Cholesky Zerlegung	$\frac{1}{3}n^3 + \frac{1}{2}n^2 + \frac{1}{6}n$
LR- Zerlegung	$\frac{2}{3}n^3 + \frac{1}{2}n^2 - \frac{1}{6}n$
QR- Zerlegung	$2n^2\left(m - \frac{1}{3}n\right)$
Jacobi/Gauß-Seidel Verfahren	$2n^2$ im k-ten Schritt, falls A Vollbesetzt

Cholesky-Zerlegung:

A: s.p.d. $A = GG^T$

$$g_{ik} = \begin{cases} 0 & \text{für } i < k \\ \sqrt{a_{kk} - \sum_{j=1}^{k-1} g_{kj}^2} & \text{für } i = k \\ \frac{1}{g_{kk}}\left(a_{ik} - \sum_{j=1}^{k-1} g_{ij}g_{kj}\right) & \text{für } i > k \end{cases}$$

Praktischerer Ansatz: GG^T Aufstellen, (mit $g_{11}, g_{21}, \ldots$) und mit A gleichsetzen

Sinus mit Hornerschema

```
function y=sinhorner(x,n)
y=1;
for i=n:-1:1
        y=1-x.^2/(2*n*(2*n+1))*y;
end;
y=x*y;
```

Inverse Berechnen mit LR

```
function Ainv = inverseberechnen(A)
n    = size(A,1);
[L,R,P]  = lu(A);
y    = zeros(n,1);
Ainv    = zeros(n);
for j=1:n
  for i=1:n
    y(i) = P(i,j) - L(i,1:i-1)*y(1:i-1);
  end
  for i=n:-1:1
    Ainv(i,j) = ( y(i) - R(i,i+1:n)*Ainv(i+1:n,j) )/R(i,i);
  end
  y    = 0*y;
end
```

Cholesky Zerlegung:

```
function R = cholesky(A)
[m,n]=size(A);
if m~=n error('A muss quadratisch sein.');
end
R=zeros(n);
R(1,1)=sqrt(A(1,1));
if n==1 return; end
R(1,2:n)=A(1,2:n)/R(1,1);
R(2:n,2:n)=cholesky( A(2:n,2:n) -
R(1,2:n)'*R(1,2:n) );
end
```

Gauß-Seidel Verfahren:

```
function [x,N] = gs(A,b,delta)
x=b;
N=0;
for i=1:200
  x=tril(A)\(b-triu(A,1)*x);
  N=N+1;
if norm(b-A*x)<delta
    return;
  end
end
error('Keine Konvergenz nach 200 Iterationen.');
end
```

Newton Verfahren:

```
function [x,k] = newton(f,Df,x,maxit,TOL)
for k=1:maxit
  x_alt = x;
  x = x - f(x)/Df(x);
  fprintf('%1.15e\n',x)
  if norm(x-x_alt) < TOL
    break
  end
end
```

LR-Zerlegung

```
function A = LR(A)
[n,~] = size(A);
if ~all(size(A) == n)
  error('A muss quadratisch sein.');
end
for k = 1:n
  I = k+1:n;
  A(I,k) = A(I,k)/A(k,k);
  A(I,I) = A(I,I) - A(I,k)*A(k,I);
end
```

LR-Zerlegung mit Pivotisierung:

```
function [A,p] = LR_pivot(A)
[n,~] = size(A);
if ~all(size(A) == n)
  error('A muss quadratisch sein.');
end
p = 1:n;
for k = 1:n
  [~,j] = max(abs(A(p(k:n),k)));
  j = j + (k-1);
  p([k,j]) = p([j,k]);
  I = k+1:n;
  A(p(I),k) = A(p(I),k)/A(p(k),k);
  A(p(I),I) = A(p(I),I) - A(p(I),k)*A(p(k),I);
end
```

QR-Zerlegung:

```
function A = QR_householder(A)
[m,n] = size(A);
for k = 1:min(m-1,n)
  v = A(k:end,k);
  na = norm(v);
if v(1) >= 0
    s = 1;
  else
    s = -1;
  end
  v(1) = v(1) + s*na;
  v = [1; v(2:end)/v(1)];
  A(k:end,k+1:end) = A(k:end,k+1:end) -
(2/(v'*v)*v)*(v'*A(k:end,k+1:end));
  A(k,k) = -s*na;
A(k+1:end,k) = v(2:end);
end
```

Bisektionsverfahren:

```
function [x,n]=bisektion (f,a,b,TOL)
fa=feval(f,a);
fb=feval(f,b);
n=0;
if fa==0; x=a; return; end
if fb==0; x=b; return; end
if fa*fb>0; error('f(a) und f(b) haben gleiches Vorzeichen.');
end
if a>b; error('a>b, leeres Intervall'); end
while b-a>TOL
x=a+0.5*(b-a);
fm=feval(f,x);
if fm==0; break; end
if fa*fm>0
  a=x;
  fa=fm;
else
  b=x;
  fb=fm;
end
n=n+1;
end
en
```

Gradientenverfahren:

```
function [x,n] = gradverf(f,gradf,x0,TOL,maxsteps)
x=x0;
gfx=feval(gradf,x);
n=0;
while n<maxsteps
        d = -gfx;
        s = 1;
        while feval(f,x+s*d)-feval(f,x) > -s*norm(gfx)^2/2
                s=s/2;
        end
        x = x + s*d;
        gfx = feval(gradf,x);
        n=n+1;
        if norm(gfx)<TOL return; end
end
error('Toleranz nicht erreicht');
```

Stationären Punkt suchen:

```
function [v,type] = statpointsincos(x0,y0)
v=[x0;y0];
if x0==y0 error('x0=y0 => Df singulaer'); end
f = @(v) [%Funktion f in v(1) und v(2)];
Df= @(v) [%Gradient f in v(1) und v(2)];
[v,n]=newton_system(f,Df,v,1000,10^(-10));
eigenvals=eig(Df(v));   %eig liefert Eigenwerte
if eigenvals(1)*eigenvals(2)<0
  type='sattel';
else
  if eigenvals(1)>0
    type='min';
  else
    type='max';
  end
end
end
```