Zusammenfassung Numerische Mathematik

Patrik Rohner :: <rohnerpa@student.ethz.ch> :: FS'09

1 Grundlagen der Numerik

1.1 Gleitpunktarithmetik

Gleitpunktzahl: $\hat{x} = \sigma M B^E$

σ: Vorzeichen B: Basis M: Mantisse E: Exponent

- Für die **Mantisse M** gilt:

$M = \sum_{j=1}^{l} m_{-j} B^{-j}$ mit $0 \le m_{-j} \le B-1$: Ziffern der Mantisse; l : Mantissenlänge (fest); $m_{-1} \neq 0$ für $\hat{x} \neq 0$: Normierung

- Für den **Exponenten E** gilt:

$E = \tau \sum_{j=1}^{k} e_j B^{k-j}$ mit τ : Vorzeichen; $0 \le e_j \le B-1$: Ziffern des Exponenten; k : definierter Exponentenbereich

Definition: $\mathbb{M}(B,l,k)$ = Menge der Maschinenzahlen

✏ $\mathbb{M}(2,5,3) \Rightarrow M_{max} = \frac{31}{32} \approx 1 \quad M_{min} = \frac{1}{2} \quad E_{max/min} = \pm 7$
$\hat{x}_{max} = M_{max} \cdot B^{E_{max}} = \frac{31}{32} \cdot 2^7 \quad \hat{x}_{min} = M_{min} \cdot B^{E_{min}} = \frac{1}{2} \cdot 2^{-7}$

✏ $\mathbb{M}(B,l,k) \Rightarrow M_{max} = \frac{B^l - 1}{B^l} \quad M_{min} = \frac{1}{B} \quad E_{max/min} = \pm B^k - 1$
$\hat{x}_{max} = \frac{B^l-1}{B^l} \cdot B^{B^k-1} \quad \hat{x}_{min} = \frac{1}{B} \cdot B^{-(B^k-1)}$

Definition Maschinengenauigkeit: eps ist die kleinste Zahl, die zu 1 addiert noch eine Veränderung bewirkt.

$$\left|\frac{x-\hat{x}}{x}\right| \le \frac{B^{E-L}}{2B^{E-1}} = \frac{B^{1-L}}{2} =: eps$$

Pseudo-Arithmetik: $(\hat{x}+\hat{y})\cdot\hat{z} = \hat{z}\cdot(\hat{y}+\hat{x}) \qquad (\hat{x}+\hat{y})\cdot\hat{z} \neq \hat{x}\cdot\hat{z} + \hat{y}\cdot\hat{z}$
$(\hat{x}+\hat{y})+\hat{z} \neq \hat{x}+(\hat{y}+\hat{z}) \Rightarrow$ von klein nach gross summieren

1.2 Fehlerfortpflanzung

Absoluter Fehler: $\Delta x = \hat{x} - x$
$\Delta(x \pm y) = (\hat{x} \pm \hat{y}) - (x \pm y) = \Delta x \pm \Delta y$
$\Delta(x * y) = \hat{x}\hat{y} - xy \cong x\Delta y + y\Delta x = xy(\delta x + \delta y)$
$\Delta(x/y) \cong x/y(\delta x - \delta y)$
Relativer Fehler: $\delta x = \frac{\Delta x}{x}$
$\delta(x \pm y) = \frac{\Delta(x\pm y)}{x\pm y} = \frac{x}{x\pm y}\cdot\frac{\Delta x}{x} \pm \frac{y}{x\pm y}\cdot\frac{\Delta y}{y} = \frac{x}{x\pm y}\delta x \pm \frac{y}{x\pm y}\delta y$
$\delta(x * y) = \frac{\Delta(x*y)}{xy} \cong \delta x + \delta y$
$\delta(x/y) = \frac{\Delta(x/y)}{x/y} \cong \delta x - \delta y$
Auslöschung: $|x \pm y| \ll \{|x|,|y|\} \Rightarrow |\delta(x\pm y)| \gg |\delta x| + |\delta y|$
Faustregel: wenn $\delta x = 10^{-k}$, dann sind $\approx k$ Ziffern richtig.
$\Rightarrow$ besser $(x^3-y^3)/(x^2+xy+y^2)$ als $x-y$ rechnen
$\Rightarrow$ besser $(x^6-y^6)/(x^4+x^2y^2+y^4)$ als x^2-y^2 rechnen

1.3 Kondition eines Problems

κ_H = Konditionszahl = Verstärkungsfaktor des relativen Fehlers von x

Konditionszahl κ_H: $\delta H = |\frac{xH'(x)}{H(x)}| * |\delta x| = \kappa_H * |\delta x|$

1.3.1 Kondition einer Matrix $\kappa(A) = \|A\| \cdot \|A^{-1}\|$:

- $A\cdot\hat{x} = \hat{b} \quad \Rightarrow \quad \|\delta x\| \le \kappa(A)\cdot\|\delta b\|$
- $\hat{A}\cdot\hat{x} = \hat{b} \quad \Rightarrow \quad \|\delta x\| \le \frac{\kappa(A)}{1-\kappa(A)\frac{\|\delta A\|}{\|A\|}}(\frac{\|\delta A\|}{\|A\|} + \frac{\|\delta b\|}{\|b\|})$

Faustregel: Bei d Dezimalstellen und $\kappa(A) \cong 10^k$ hat die Lösung im schlechtesten Fall noch $d-k$ richtige Stellen.

in der 2-Norm: $\|A\|_2 = \sqrt{\mu_{max}}$ (μ_{max} = max. EW von A^TA)

- A orthogonal: $A^T = A^{-1} \qquad \|A\|_2 = 1$
- A symmetrisch: $A^T = A \qquad \|A\|_2 = |\lambda_{max}| \qquad \|A^{-1}\|_2 = \frac{1}{|\lambda_{min}|}$
- A regulär: $\|A^{-1}\|_2 = \frac{1}{\sqrt{\mu_{min}}}$

$$\kappa(A) = \frac{\sqrt{\mu_{max}}}{\sqrt{\mu_{min}}} \quad \text{und falls } A^T = A \text{ gilt } \kappa(A) = \frac{|\lambda_{max}|}{|\lambda_{min}|}$$

$\kappa \approx 1$ gut konditioniertes Problem $\kappa \gg 1$ schlecht konditioniertes Problem
`>> k_A=cond(A)`

2 Lineare Gleichungssysteme

2.1 LRP-Zerlegung

LRP-Zerlegung von A aus $Ax = b$

i LRP-Zerlegung von A: $LR = PA$ `>> [L,R,P]=lu(A)`
ii Vorwärtseinsetzen: $Lc = Pb$ nach c `>> c=L\(P*b)`
iii Rückwärtseinsetzen: $Rx = c$ nach x `>> x=R\c`

Der Algorithmus braucht $\frac{2}{3}n^3 + 2n^2$ Operationen. $\mathbf{O(n^3)}$

2.1.1 Pivotstrategien (gegen Auslöschung)

- **Diagonalstrategie**: Diagonalelement = Pivot (schlecht)
- **Spaltenmaximumstrategie**: Das betragsmässig grösste Element wird gewählt.
- **relative Spaltenmaximumstrategie**: Das betragsmässig grösste Element, relativ zum zweitgrössten Element der Zeile, wird gewählt.

2.2 Iterative Methoden

Alle hier benötigen pro Iterationsschritt grob eine Matrix-Vektor-Multiplikation ($\mathbf{O(n^2)}$, wenn sparse: $\mathbf{O(n)}$) und konvergieren linear ($\mathbf{O(n)}$).

2.2.1 stationäre Iterationsverfahren

$$x_{k+1} = T\cdot x_k + c$$

Der absolute Fehler verändert sich mit $e_k = T^k \cdot e_0$, womit gilt:

Konvergenzbedingung: $\|T\| < 1$ `>> norm(T)`

Für den Spektralradius $\rho(T) := \max|\lambda_i|$ gilt $\rho(T) \le \|T\|$ und so

alternative Konvergenzbed.: $\rho(T) < 1$ `>> max(abs(eig(T)))`

Abschätzung (worst case) der Anzahl Iterationen für bestimmte Fehler:
$\|e\|_2 = \|T^k e^0\|_2 \quad \|e^k\|_2 \le \|T\|_2^k \|e^0\|_2 \quad \frac{\|e^k\|_2}{\|e^0\|_2} \le \|T\|_2^k \quad k \ge \frac{\ln(\text{rel.Fehler})}{\ln\|T\|_2}$
Aufspaltung für bestimmte Verfahren $A = L + D + R$
`>> D=diag(diag(A))  >> L=tril(A)-D  >> R=triu(A)-D`

Jacobi-Verfahren

$$T = -D^{-1}\cdot(L+R) \qquad c = D^{-1}b$$

- Konsistenzbedingung: A regulär, d.h $det(A) \neq 0$
- hinreichende Konvergenzbedingung: A strikt diagonal dominant (max |.| in der Diagonalen)
- notwendige Konvergenzbedingung (bei allen 3 Verfahren): $\|T\| < 1$

Gauss-Seidel-Verfahren

$$T = -(D+L)^{-1}R \qquad c = (D+L)^{-1}b$$

$(D+L)x_{k+1} = -Rx_k + b$ `>> x_(k+1)=(D+L)\(-Rx_k+b)`
Falls A strikt diagonal dominant oder A symmetrisch positiv definit ist, konvergiert das Verfahren, und zwar schneller als das Jacobi-Verfahren.

SOR-Verfahren (Successive Over-Relaxation)

$$x_{k+1} = \underbrace{(D+\omega L)^{-1}[-\omega R + (1-\omega)D]}_{=T(\omega)}x_k + \underbrace{(D+\omega L)^{-1}\omega b}_{=c(\omega)}$$

ω = Relaxationsparameter
$\omega_{optimal} = \frac{2}{1+\sqrt{1-[\rho(D^{-1}(L+R))]^2}}$ ρ: Spektralradius

2.2.2 Methode der konjugierten Gradienten (CG)

- Zu lösendes Gleichungssystem: $Ax + b = 0$
- **Voraussetzungen:** A symmetrisch und positiv definit.

$F(u) := 1/2u^T Au + u^T b \quad gradF = \begin{pmatrix} \frac{\partial F}{\partial u_1} \\ \dots \\ \frac{\partial F}{\partial u_n} \end{pmatrix} = Au + b = r(u)$ (Residuen)

Idee: Anstelle $Ax + b = 0$ zu lösen, wird das Minimum von $F(u)$ gesucht. Es gilt $F(x) = min_{u \in \mathbb{R}^n} F(u)$.

CG-Verfahren $Ax + b = 0$

- **Start:** $u_0, r_0 = Au_0 + b, TOL$
- **1. Schritt:** (Steilster Abstieg)
  - i $p_1 = -r_0$
  - ii $\rho_1 = \frac{(r_0, r_0)}{(p_1, Ap_1)}$
  - iii $u_1 = u_0 + \rho_1 p_1$
  - iv $r_1 = r_0 + \rho_1 Ap_1$
- **k. Schritt:** ($k \geq 2$, konjugierter Gradient)
  - i $e_{k-1} = \frac{(r_{k-1}, r_{k-1})}{(r_{k-2}, r_{k-2})}$
  - ii $p_k = -r_{k-1} + e_{k-1} p_{k-1}$
  - iii $\rho_k = \frac{(r_{k-1}, r_{k-1})}{(p_k, Ap_k)}$
  - iv $u_k = u_{k-1} + \rho_k p_k$
  - v $r_k = r_{k-1} + \rho_k Ap_k$
- **Abbruchkriterium:** $\|r_k\| \leq TOL \|r_0\|$

Aufwand: $\mathbf{O(n)}$ Iterationen (typisch aber: $\sqrt{n}$). Jeder Schritt $\mathbf{O(n^2)}$ Operationen, $\mathbf{O(n)}$ bei sparse-Matrix.

Abschätzung /obere Schranke: $\|u_k - x\| \leq 2\alpha^k \|u_0 - x\|$ mit $\alpha = \frac{\sqrt{\kappa_A}-1}{\sqrt{\kappa_A}+1}$

✏ Gesucht: obere Schranke der Anzahl Iterationen k, um den absoluten Anfangsfehler um den Faktor c zu reduzieren, mit κ_A gegeben.
$k = \frac{\ln(0.5 \cdot c)}{\ln(\alpha)}$

```
function[x,it,time]=my_cg(A,b,u0,maxit,tol)
tic;                % start measuring time
u=u0;               % start with initial guess
r0=A*u0+b;
p=-r0;
r=r0;
it=0;
while (it<maxit)
    it=it+1;
    rho=(r'*r)/(p'*(A*p));
    u=u+rho*p;
    r_old=r;
    r=r+rho*(A*p);
    if (norm(r)<tol*norm(r0))
        break;  % converged
    end
    e=(r'*r)/(r_old'*r_old);
    p=-r+e*p;
end
x=u;
time=toc;           % stop measuring time
if (it==maxit)
    error(['no convergence after ',int2str(it),'
        iterations']);
end
```

`>> [x,flag,relres,it]=pcg(A,b,tol,maxit)` Löst $Ax = b$!
Vorkonditionierung: Mit unvollständiger Cholesky-Zerlegung die Kondition von A verkleinern.

```
>> H=cholinc(A,1e-3)
>> [x,flag,relres,it]= pcg(A,b,tol,maxit,H',H)
```

3 Nichtlineare Gleichungssysteme

3.1 Eine Gleichung in einer Unbekannten

- **Gegeben:** Stetige Funktion $y = f(x)$.
- **Gesucht:** Lösung x^* der Gleichung $f(x) = 0$ (Nullstelle).

Für $|f'(x*)| \ll 1$ ist das Problem schlecht konditioniert:
$x* = f^{-1}(0) = H(0) \qquad \kappa_H = |\frac{H'(0)}{H(0)}| = |\frac{1}{x* f'(x*)}|$
Die Konvergenz der Fixpunkt-Iteration und des Newton-Verfahrens werden mit dem Banachschen Fixpunktsatz untersucht. Sie konvergieren nur wenn der Fixpunktsatz erfüllt ist.

3.1.1 Fixpunkt-Iteration

- **Gegeben:** $y = f(x)$, $x \in [a, b]$
- **Gesucht:** Lösung der Fixpunktgleichung $x = F(x) \Rightarrow x_{k+1} = F(x_k)$
- **Abbruchkriterium:** $|x_{k+1} - x_k| \leq |x_{k+1}| \cdot RTOL + ATOL$

Die Fixpunktgleichung erhält man durch Umformung (Addition, usw.) der Funktionsgleichung $f(x) = 0$. Die Fixpunktgleichungen sind nicht eindeutig, es gibt meistens mehrere verschiedene Arten die Gleichung darzustellen.

3.1.2 Newton-Verfahren (1D)

- **Gegeben:** Stetige Funktion $y = f(x)$
- **Gesucht:** Lösung x^* der Gleichung $f(x) = 0$ (Nullstelle)
- **Idee:** Ersetze Graph von $f(x)$ durch Tangente in $P(x_0, f(x_0))$

Newton-Iteration in 1D

- **Start:** $x_0, f'(x), RTOL, ATOL$
- **Vorgehen:** $x_{k+1} = F(x_k) = x_k - \frac{f(x_k)}{f'(x_k)}$
- **Abbruchkriterium:** $|x_{k+1} - x_k| \leq |x_{k+1}| \cdot RTOL + ATOL$

⊞ schnell

⊟ lokal, teuer, $f'(x)$ muss bekannt sein

3.1.3 Banachscher Fixpunktsatz

Folgende Bedingungen müssen erfüllt sein:

- F bidet das Intervall $I = [a, b]$ in sich ab. Umkehrung möglich.
- F ist eine Kontraktion: $\exists \quad 0 < L < 1 \qquad L \geq |\frac{F(x^1) - F(x^2)}{x^1 - x^2}|$
  Falls F' stetig ist auf I genügt $|F'| \leq L < 1$
  ▸ Falls F'' in ganz I das gleiche Vorzeichen hat, nur $|F'(a)|, |F'(b)|$ betrachten, sonst auch Maxima @ $F'' = 0$

Daraus folgt dann:

- F hat genau einen Fixpunkt $\overline{x}$ in $[a, b]$
- Es gelten folgende Fehlerabschätzungen
  - $|\overline{x} - x_k| \leq \frac{L^k}{1-L} |x_1 - x_0|$ a priori
  - $|\overline{x} - x_k| \leq \frac{L}{1-L} |x_k - x_{k-1}|$ a posteriori

  Die a priori Abschätzung wird vor allem dazu benutzt nach nur einer Iteration bereits eine obere Schranke für die Zahl der Iterationen anzugeben. Die benötigt wird um $\overline{x}$ bis auf einen Fehler ε zu bestimmen.
  $|\overline{x} - x_k| \leq \frac{L^k}{1-L} |x_1 - x_0| \leq \varepsilon \qquad \Rightarrow k \geq \frac{\ln \frac{\varepsilon(1-L)}{|x_1 - x_0|}}{\ln L}$

Es gibt anziehende- und abstossende Fixpunkte

$|F'(x)| < 1$, so ist $\overline{x}$ anziehend
$|F'(x)| > 1$, so ist $\overline{x}$ abstossend
$|F'(x)| = 1$, so kann $\overline{x}$ anziehend, abstossend oder keines von beiden sein

Je kleiner $|F'(x)|$ ist, desto schneller konvergiert das Verfahren. Die selben Bedingungen gelten für die inverse Iteration für $(F^{-1})'$.

3.1.4 Konvergenzordnung p

$$|\overline{x} - x_{n+1}| \cong C |\overline{x} - x_n|^p$$

CG-Verfahren:	$p = 1$	Fixpunktiteration:	$p = 1$
Newton-Verfahren:	$p = 2$	Sekantenmethode:	$p = 1.6$
Bisektionsverfahren:	$p = 1$		

3.1.5 Effizienzindex E

$$E = p^{1/m}$$

wobei m die Zahl der Funktionsauswertungen in jedem Schritt bedeutet.

3.1.6 Die Sekantenmethode

Sekantenmethode

- **Start:** $x_0, x_1, RTOL, ATOL$
- **Vorgehen:** $x_{k+1} = \frac{x_{k-1} f(x_k) - x_k f(x_{k-1})}{f(x_k) - f(x_{k-1})}$
- **Abbruchkriterium:** $|x_{k+1} - x_k| \leq |x_{k+1}| \cdot RTOL + ATOL$

⊞ braucht keine Ableitung, billiger als Newton

⊟ lokal, langsamer als Newton

3.1.7 Das Bisektionsverfahren

- **Gegeben:** $f(x)$, stetig im Intervall $[a,b]$ und $f(a)\cdot f(b)<0$

BISEKTIONSVERFAHREN (DIVIDE AND CONQUER)

- **Start:** $a_0, b_0, x_0 = \frac{a_0+b_0}{2}, x_1, RTOL, ATOL$
- **Vorgehen:**
  - i **if** $f(x_k)=0$ **then** break
  - ii **if** $f(a_k)\cdot f(x_k)<0$ **then** $a_{k+1}=a_k,\ b_{k+1}=x_k$
    else $a_{k+1}=x_k,\ b_{k+1}=b_k$
    $x_{k+1}=\frac{a_{k+1}+b_{k+1}}{2}$
- **Abbruchkriterium:** $|x_{k+1}-x_k| \le |x_{k+1}|\cdot RTOL + ATOL$

⊞ global

⊟ langsame Konvergenz

3.2 Mehrere Gleichungen mit mehreren Unbekannten

Gegeben: $f(x)=0$ mit

$$x=\begin{pmatrix} x_1 \\ \cdot \\ \cdot \\ x_n \end{pmatrix} \quad f(x)=\begin{pmatrix} f_1(x1,\ldots,x_n) \\ \cdot \\ \cdot \\ f_n(x_1,\ldots,x_n) \end{pmatrix} = \begin{pmatrix} 0 \\ \cdot \\ \cdot \\ 0 \end{pmatrix}$$

Jacobi-Matrix: $J(x)=\frac{\partial f}{\partial x}=\begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ .. & & & .. \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{pmatrix}$

$\Rightarrow$ Linearisierung von f in x_0

3.2.1 Newton Verfahren (nD)

Das Newton-Verfahren ist immer lokal konvergent, falls $det(J(x^*))\neq 0$ Es konvergiert quadratisch. Es gilt der Banachsche Fixpunktsatz, ersetze $|.|$ durch $||.||$

NEWTONVERFAHREN IN $\mathbb{R}^n$

- **Start:** $x^0, RTOL, ATOL$
- **Vorgehen:**
  - i $J(x^k)\Delta^k = -f(x^k) \quad \Rightarrow \quad \Delta^k = J(x^k)\backslash -f(x^k)$
  - ii $x^{k+1} := x^k + \Delta^k$
- **Abbruchkriterium:** $||\Delta^k|| \le ||x^{k+1}||\cdot RTOL + ATOL$

```
function[x,it]=my_newton(x0,rtol,atol,maxit)
it=0; x=x0;
while (it<maxit)
   delta=-J(x)\f(x);
   x_old=x;
   x=x_old+delta;
   it=it+1;
   if (norm(delta)<=norm(x)*rtol+atol)
      break;            % converged
   end
end
if (it==maxit)
    error(['no convergence after ',int2str(it),'
        iterations']);
end;
end
function[f_]=f(x)        % f(x)=0
x1=x(1); x2=x(2);
f_=[x1+x2^3 ; x1^2+x2];
end
function[J_]=J(x)
x1=x(1); x2=x(2);
J_=[1   3*x2^2 ; 2*x1   1];
end
```

⊞ quadratische Konvergenz

⊟ Finden eines guten Startvektors x^0.

⊟ Berechnen der Jacobi-Matrix in jedem Schritt.

3.2.2 Quasi-Newton Verfahren (nD)

In jedem Schritt wird die gleiche Jacobi-Matrix $J(x^0)$ verwendet.

```
function[x,it]=my_quasinewton(x0,rtol,maxit)
it=0; x=x0;
J0=J(x0);              % <--
while (it<maxit)
   delta=-J0\f(x);     % <--
[...]                  % same as my_newton
```

⊞ J nur noch einmal berechnen.

⊟ nur noch lineare Konvergenz

3.2.3 gedämpftes Newton Verfahren (nD)

Wie Newton, aber mit Dämpfungsfaktor α_k.

$$x^{k+1} := x^k + \alpha_k \Delta^k \qquad \alpha_{k,start} \ll 1 \quad \alpha_{k,ende=1}$$

⊞ grösserer lokaler Konvergenzbereich durch weniger Überschiessen.

⊟ Durch Dämpfung aber langsamer.

4 Ausgleichsrechnung

Gesucht wird ein Vektor x, so dass die dazugehörigen Residuen r minimal sind.

4.1 Lineare Ausgleichsrechnung

4.1.1 mit Normalengleichungen nach Gauss

i **Fehlergleichungen:** $Ax-c=r$

ii **Normalengleichungen:** $\underbrace{A^TA}_{A^*}x = A^Tc$ `>> x=(A'A)\(A'c)`

⊟ A^* meist schlecht konditioniert $\Rightarrow$ QR-Zerlegung

4.1.2 mit QR-Zerlegung

QR-ZERLEGUNG VON A AUS $Ax-c=r$

i QR-Zerlegung von A: $A \rightsquigarrow R \quad c \rightsquigarrow d$

$$\left.\begin{aligned} A_{k+1} &= U_{pq}^T \cdot A_k \\ c_{k+1} &= U_{pq}^T \cdot c_k \end{aligned}\right\} \text{ bis } A \text{ quadratisch, dann } A=R \text{ und } c=d$$

$$U_{pq} = \begin{matrix} \\ p\rightarrow \\ \\ q\rightarrow \\ \\ \end{matrix} \begin{pmatrix} \cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi \end{pmatrix} \quad \begin{aligned} \sin\phi &= -\frac{A(q,p)}{\sqrt{A(p,p)^2+A(q,p)^2}} \\ \cos\phi &= \frac{A(p,p)}{\sqrt{A(p,p)^2+A(q,p)^2}} \end{aligned}$$

ii Rückwärtseinsetzen: $Rx=d$ nach x

`>> [Q,R]=qr(A)  >> d=Q'c  >> x=R0\d0` mit $R0$ quadratisch.

⊞ $\kappa_{Gauss}=\kappa(A^TA)=\frac{\mu_{max}}{\mu_{min}}$ $\quad \kappa_{QR}=\kappa(A)=\sqrt{\kappa(A^TA)}$

4.1.3 mit Singulärwertzerlegung $A=USV^T$

- **Gegeben:** Fehlergleichungen $Ax-c=r$

```
function[x,y]=my_svd(A,c)
[u,s,v]=svd(A);
k=rank(A);
[rows,cols]=size(A);
y=[0 0]';          % y to be a column vector
for i=1:k
    y(i)=1/s(i,i)*u(:,i)'*c;
end
for i=k+1:cols
    y(i)=0;        % arbitrary, chosen to be 0
end
x=v*y;
```

4.2 Nichtlineare Ausgleichsrechnung

Wie bei der linearen Ausgleichsrechnung ist die Kondition von A^TA zu gross, weshalb man besser mit den Fehlergleichungen rechnet.

4.2.1 Gauss-Newton-Methode

- Fehlergleichungen: $\underline{f}(x)-\underline{c}=\underline{r}$
- Linearisierte Fehlergln.: $\underbrace{A^0}_{J@x_0}\ \xi + \underbrace{\underline{f}^0-\underline{c}}_{\underline{d}^0} = \underline{\rho}^0 \quad \Rightarrow \quad \xi^1 = A_0\backslash(\underline{\rho}^0-\underline{d}^0)$
- $x^1=x^0+\xi^1$ und wieder von vorne...

⊟ Je nach x^0 keine Konvergenz → Minimierung.

4.2.2 Gauss-Newton-Methode mit Minimierung

Gleiches Vorgehen bis ξ^{k+1}, dann mit $\widetilde{x}=x^k+t\xi^{k+1}$ und $t=1,\frac{1}{2},\frac{1}{4},\ldots$ testen, ob $S(\widetilde{x})<S(x^k)$ erfüllt ist und bei $|S(\widetilde{x})-S(x^k)|<TOL$ den Vorschlag $\widetilde{x}$ akzeptieren: $x^{k+1}=\widetilde{x}$. $(S(x)=\underline{r}^T\underline{r})$

⊞ Konvergenz garantiert, dank $S(\widetilde{x})<S(x^k)$

⊟ Langsame Konvergenz am Anfang möglich, aber am Ende doch wieder fast quadratisch.

5 Interpolation und Approximation

5.1 Das Interpolationspolynom

- **Gegeben:** $n+1$ Stützstellen $x_0, x_1, \ldots, x_n$ und die Stützwerte $f_0 = f(x_0), f_1 = f(x_1), \ldots, f_n = f(x_n)$.
- **Gesucht:** Polynom $\leq n$-ten Grades $P_n(x) = a_0x^n + a_1x^{n-1} + \cdots + a_{n-1}x + a_n$ mit $P_n(x_0) = f_0, P_n(x_1) = f_1, \ldots, P_n(x_n) = f_n$

5.1.1 Darstellung nach Lagrange

$$P_n(x) = \sum_{i=0}^{n} f_i l_i(x) \qquad l_i(x) = \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j} \quad i = 0, 1, \ldots, n$$

✏ Gegeben: 4 Stützstellen und ihre Werte $\Rightarrow n = 3$
$P_n(x) = \sum_{i=0}^{n} f_i l_i(x)$ (Falls $f_i = 0 \Rightarrow l_i$ nicht berechnen)

$l_0(x) = \frac{(x-x_1)(x-x_2)(x-x_3)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)}$ $\quad l_1(x) = \frac{(x-x_0)(x-x_2)(x-x_3)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)}$

$l_2(x) = \frac{(x-x_0)(x-x_1)(x-x_3)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)}$ $\quad l_3(x) = \frac{(x-x_0)(x-x_1)(x-x_2)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)}$

5.1.2 Baryzentrsiche Darstellung

$$P_n(x) = \frac{\sum_{i=0}^{n} \mu_i f_i}{\sum_{i=0}^{n} \mu_i} \qquad \lambda_i := \frac{1}{\prod_{j=0, j\neq i}^{n}(x_i - x_j)} \qquad \mu_i := \frac{\lambda_i}{x - x_i}$$

mit Stützkoeffizienten λ_i (Nenner von $l_i(x)$) und $\quad i = 0, 1, \ldots, n$.

⊞ Bei verschiedenen Polynomen weniger aufwendig als Lagrange.
$\lambda_i \to O(n^2)$ (1x); $P_n(x) \to O(n)$; bei Lagrange immer $O(n^2)$.

5.1.3 Fehler des Interpolationspolynoms

$$f(x) - P_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{i=0}^{n} (x - x_i) = O(h^{n+1})$$

$f(x) - P_n(x) \leq \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{i=0}^{n} (x - x_i)$ mit $\xi = x$ wo Fehler maximal

5.2 Splineinterpolation

Die Funktion $f(x)$ wird stückweise mit Polynomen 3.Grades $Q_i(t)$ interpoliert. $Q_i(t) := P_i(x_i + h_i t)$ mit $t = \frac{x - x_i}{h_i}$ und $h_i = x_{i+1} - x_i$.

- $Q_i(t) = \begin{array}{l} f_i(1 - 3t^2 + 2t^3) + f_{i+1}(3t^2 - 2t^3) + h_i f'_i(t - 2t^2 + t^3) \\ + h_i f'_{i+1}(-t^2 + t^3) \end{array}$
- $\dot{Q}_i(t) = f_i(6t + 6t^2) + f_{i+1}(6t - 6t^2) + h_i f'_i(1 - 4t + 2t^2) + h_i f'_{i+1}(-2t + 3t^2)$
- $\ddot{Q}_i(t) = f_i(-6 + 12t) + f_{i+1}(6 - 12t) + h_i f'_i(-4 + 6t) + h_i f'_{i+1}(-2 + 6t)$
- $\dddot{Q}_i(t) = 12f_i - 12f_{i+1} + 6h_i f'_i + 6h_i f'_{i+1}$
- $Q_i(0) = f_i \quad Q_i(1) = f_{i+1} \quad \dot{Q}_i(0) = h_i f'_i \quad \dot{Q}_i(1) = h_i f'_{i+1}$.
Sind die Ableitungen bekannt, handelt es sich um die **Hermite-Interpolation** und im Algorithmus beginnt man bei Schritt 2.
- Sind die Ableitungen nicht bekannt, fehlen diese Gleichungen, man stellt daher zusätzliche Forderungen:
  - **1.Forderung:** $P''_i(x_{i+1}) = P''_{i+1}(x_{i+1}) \quad \Rightarrow \quad \frac{\ddot{Q}_i(1)}{h_i^2} = \frac{\ddot{Q}_{i+1}(0)}{h_{i+1}^2}$
  - **2.Forderung "not a knot":**
  $$\left.\begin{array}{l} P_1(x) = P_2(x) \\ P_{n-2}(x) = P_{n-1}(x) \end{array}\right\} \Rightarrow \left\{\begin{array}{l} \frac{\dddot{Q}_1(1)}{h_1^3} = \frac{\dddot{Q}_2(0)}{h_2^3} \\ \ldots \\ \frac{\dddot{Q}_{n-2}(1)}{h_{n-2}^3} = \frac{\dddot{Q}_{n-1}(0)}{h_{n-1}^3} \end{array}\right.$$
  - **2.Forderung "natürliche Randbedingung":**
  $P''_1(x_1) = P''_{n-1}(x_n) = 0 \quad \Rightarrow \quad \frac{\ddot{Q}_1(0)}{h_1^2} = \frac{\ddot{Q}_{n-1}(1)}{h_{n-1}^2} = 0$
  - **2.Forderung "periodische Funktion":**
  $P''_1(x_1) = P''_{n-1}(x_n) \quad \Rightarrow \quad \frac{\ddot{Q}_1(0)}{h_1^2} = \frac{\ddot{Q}_{n-1}(1)}{h_{n-1}^2}$
  periodische Funktion $\quad \Rightarrow \quad \left\{\begin{array}{l} f_1 = f_n \\ f'_1 = f'_n \end{array}\right.$

Spline-Interpolationsfunktion $g(x)$

i Finde Lösung aus $Af' = d$

↘ not a knot:

$$\begin{pmatrix} b_1 & \frac{a_1}{2} & & & & & \\ b_1 & a_1 & b_2 & & & & \\ & b_2 & a_2 & b_3 & & & \\ & & \ldots & \ldots & \ldots & & \\ & & & b_{n-2} & a_{n-2} & b_{n-1} & \\ & & & & \frac{a_{n-2}}{2} & b_{n-1} & \end{pmatrix} \begin{pmatrix} f'_1 \\ f'_2 \\ f'_3 \\ \ldots \\ f'_{n-1} \\ f'_n \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ \ldots \\ d_{n-1} \\ d_n \end{pmatrix}$$

↘ natürliche RB:

$$\begin{pmatrix} 4 & 2 & & & & \\ b_1 & a_1 & b_2 & & & \\ \ldots & \ldots & \ldots & & & \\ & b_{n-2} & a_{n-2} & b_{n-1} & & \\ & & 2 & 4 & & \end{pmatrix} \begin{pmatrix} f'_1 \\ f'_2 \\ \ldots \\ f'_{n-1} \\ f'_n \end{pmatrix} = \begin{pmatrix} \frac{6(f_1 - f_2)}{h_1} \\ d_2 \\ \ldots \\ d_{n-1} \\ \frac{6(f_n - f_{n-1})}{h_{n-1}} \end{pmatrix}$$

↘ periodische RB, $h = const.$:

$$\begin{pmatrix} 8 & 2 & & & 2 \\ b & a & b & & \\ & \ldots & \ldots & \ldots & \\ & & b & a & b \\ b & & & b & a \end{pmatrix} \begin{pmatrix} f'_1 \\ f'_2 \\ \ldots \\ f'_{n-2} \\ f'_{n-1} \end{pmatrix} = \begin{pmatrix} \frac{6}{h}(f_2 - f_{n-1}) \\ d_2 \\ \ldots \\ d_{n-2} \\ d_{n-1} \end{pmatrix}$$

$h_i := x_{i+1} - x_i, \quad i = 1, \ldots, n-1$
$b_i := \frac{1}{h_i}, \quad i = 1, \ldots, n-1$
$a_i := 2(b_i + b_{i+1}), \quad i = 1, \ldots, n-2$
$c_i := \frac{f_{i+1} - f_i}{h_i^2}, \quad i = 1, \ldots, n-1$
$d_1 := 2c_1 + \frac{h_1}{h_1 + h_2}(c_1 + c_2)$ (nur bei "not a knot")
$d_i := 3(c_{i-1} + c_i), \quad i = 2, \ldots, n-1$
$d_n := 2c_{n-1} + \frac{h_{n-1}}{h_{n-1} + h_{n-2}}(c_{n-1} + c_{n-2})$ (nur bei "not a knot")

ii Bestimme den Index i, für den gilt $x_i \leq x \leq x_{i+1}$

iii Setze $t = \frac{x - x_i}{h_i}$

iv Berechne $Qi(t) = \alpha_0 + [\beta_0 + (\gamma_0 + \delta_0 t)(t-1)]t$, wobei

$\beta_{-1} = h_i f'_i$
$\alpha_0 = f_i \qquad \gamma_0 = \beta_0 - \beta_{-1}$
$\beta_0 = \alpha_1 - \alpha_0 \qquad \delta_0 = \gamma_1 - \gamma_0$
$\alpha_1 = f_{i+1} \qquad \gamma_1 = \beta_1 - \beta_0$
$\beta_1 = h_i f'_{i+1}$

v Setze $g(x) = Q_i(t)$.

Allgemeines Vorgehen beim Lösen eines Problems:

1 Entscheiden, welche 2.Forderung gewählt wird $\Rightarrow$ Gleichunssystem hinschreiben.

2 Tabelle erstellen, um den Vektor der rechten Seite d_i zu finden. Gleichungssystem mit `TR::rref()` lösen.

3 Aus f'_i per Hermiteschen Algorithmus (ii bis v) die gesuchte Funktion $g(x)$ finden, wobei $g(x) = Qi(t) = \alpha_0 + [\beta_0 + (\gamma_0 + \delta_0 t)(t-1)]t$ mit `TR::qi(`$\alpha_0, \beta_0, \gamma_0, \delta_0, t$`)`.

```
1  function[yy]=my_spline_notaknot(x,fx,xx)
2  h=diff(x);
3  b=1./h;
4  a=2*(b(1:end-1)+b(2:end));
5  c=diff(fx)./h.^2;
6  d=[2*c(1)+h(1)/(h(1)+h(2))*(c(1)+c(2)),...
7     3*(c(1:end-1)+c(2:end)),...
8     2*c(end)+h(end)/(h(end-1)+h(end))*...
9                        (c(end)+c(end-1))];
10 A=spdiags([[b(1:end-1)';a(end)/2;1]...
11             [b(1);a';b(end)]...
12             [1;a(1)/2;b(1:end-1)']],...
13             -1:1,length(x),length(x));
14 f_prime=A\d';
15 for k=1:length(xx)
16     i = min(find((xx(k)-x)<0))-1;
17     t=(xx(k)-x(i))/h(i);
18     alpha=[fx(i) fx(i+1)];
19     beta=[h(i)*f_prime(i) diff(alpha)...
20           h(i)*f_prime(i+1)];
21     gamma=diff(beta);
22     delta = diff(gamma);
23     yy(k)=alpha(1)+(beta(2)+...
24              (gamma(1)+delta(1)*t)*(t-1))*t;
25 end
```

`>> [yy]= spline(x,fx,xx)` ≡ my_spline_notaknot.m

5.2.1 Interpolationsfehler

Der Interpolationsfehler ist von der Ordnung $O(h^4)$. Verglichen mit dem Interpolationspolynom $O(h^{n+1})$ ist das viel besser, je grösser n ist. Beim Interpolationspolynom für n gross hat es zwischen den Stützstellen Extremalwerte, die gegen den Rand und mit zunehmendem n immer grösser werden. Dieses Verhalten zeigt die Splineinterpolation nicht.

5.3 Trigonometrische Interpolation

Diskrete Fourier-Transformation (DFT) und *Fast Fourier Transformation* (FFT) spielen eine wichtige Rolle in der Signal- und Bildverarbeitung, im besonderen auch bei der Daten-Kompression.

- **Problemstellung**: Es sind N Stützstellen x_k sowie deren Stützwerte f_k einer 2π-periodischen Funktion gegeben.
- **Gesucht**: ein trigonometrisches Polynom
$p(x) = \frac{a_0}{2} + \sum_{j=1}^{N/2-1}[a_j cos(jx) + b_j sin(jx)] + \frac{a_{N/2}}{2} cos\left(\frac{N}{2}x\right)$

5.3.1 DFT

Fourier-Reihe von f: $f(x) = \sum_{j=-\infty}^{\infty} c_j e^{ijx}$ $c_j = \frac{1}{2\pi}\int_0^{2\pi} f(x)e^{-ijx}dx$
Dieses Integral kann natürlich nicht gelöst werden, da die Funtkion $f(x)$ nur an N Stellen bekannt ist ⇒ Approximation des Integrals durch Trapeznäherung $\tilde{c}_j = \frac{1}{N}\sum_{l=0}^{N-1} f(x_l)e^{-ijx_l}$, mit $x_l = \frac{l2\pi}{N}$.

Analyse: $\tilde{c}^{(N)} = \frac{1}{N}Wf^{(N)}$	**Synthese:** $f^{(N)} = \overline{W}\tilde{c}^{(N)}$

mit $f^{(N)} = \begin{pmatrix} f_0 \\ \vdots \\ f_{N-1} \end{pmatrix}$ $\tilde{c}^{(N)} = \begin{pmatrix} \tilde{c}_0 \\ \vdots \\ \tilde{c}_{N-1} \end{pmatrix}$

und $W = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & w & w^2 & \cdots & w^{N-1} \\ 1 & w^2 & w^4 & \cdots & w^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & w^{N-1} & w^{2(N-1)} & \cdots & w^{(N-1)^2} \end{pmatrix}$, wobei $w = e^{-i\frac{2\pi}{N}}$

und $\overline{W}$ das konjugiert komplexe von W.

trigonometrische Interpolationspolynome

komplex: $p(x) := \sum_{j=-N/2+1}^{N/2+1} \tilde{c}_j e^{ijx} + \tilde{c}_{N/2}\cos(\frac{N}{2}x)$

2π-periodisch:
$$\begin{cases} p(x)_3 := \tilde{c}_0 + \tilde{c}_1 e^{ix} + \tilde{c}_2 e^{-ix} \\ p(x)_5 := \tilde{c}_0 + \tilde{c}_1 e^{ix} + \tilde{c}_2 e^{-ix} + \tilde{c}_3 e^{2ix} + \tilde{c}_4 e^{-2ix} \\ \dots \end{cases}$$

reell: $p(x) := \frac{a_0}{2} + \sum_{j=1}^{N/2+1}[a_j\cos(jx) + b_j\sin(jx)] + \frac{a_{N/2}}{2}\cos(\frac{N}{2}x)$

Allgemeines Vorgehen:

1 Diskrete Fourier-Analyse $\tilde{c}^{(N)} = \frac{1}{N}Wf^{(N)}$.

2 Auswertung des Polynoms $p(x)$.

✏ Bestimme das interpolierende trigonometrische Polynom von der 2π-periodischen Funktion mit

x_i	0	$\frac{2\pi}{3}$	$\frac{4\pi}{3}$
f_i	f_0	f_1	f_2

$$\tilde{c}^{(3)} = \frac{1}{3}Wf^{(3)} = \frac{1}{3}\begin{pmatrix} 1 & 1 & 1 \\ 1 & w & w^2 \\ 1 & w^2 & w^4 \end{pmatrix}\begin{pmatrix} f_0 \\ f_1 \\ f_2 \end{pmatrix} = \begin{pmatrix} \frac{f_0+f_1+f_2}{3} \\ \frac{f_0+wf_1+w^2f_2}{3} \\ \frac{f_0+w^2f_1+w^4f_2}{3} \end{pmatrix}$$

mit $\begin{cases} w = e^{-i\frac{2\pi}{N}} = e^{-i\frac{2\pi}{3}} = -\frac{1}{2} - \frac{\sqrt{3}}{2}i \\ w^2 = e^{-2i\frac{2\pi}{N}} = e^{-i\frac{4\pi}{3}} = -\frac{1}{2} + \frac{\sqrt{3}}{2}i \\ w^4 = w = e^{-i\frac{2\pi}{3}} = -\frac{1}{2} - \frac{\sqrt{3}}{2}i \end{cases}$

$$\tilde{c}^{(3)} = \begin{pmatrix} \frac{f_0+f_1+f_2}{3} \\ \frac{f_0+f_1(-\frac{1}{2}-\frac{\sqrt{3}}{2}i)+f_2(-\frac{1}{2}+\frac{\sqrt{3}}{2}i)}{3} \\ \frac{f_0+f_1(-\frac{1}{2}+\frac{\sqrt{3}}{2}i)+f_2(-\frac{1}{2}-\frac{\sqrt{3}}{2}i)}{3} \end{pmatrix}$$

$p(x) = \tilde{c}_0 + \tilde{c}_1 e^{ix} + \tilde{c}_2 e^{-ix} = \tilde{c}_0 + (\tilde{c}_1 + \tilde{c}_2)\cos(x) + i(\tilde{c}_1 - \tilde{c}_2)\sin(x)$

mit $\begin{cases} \tilde{c}_1 + \tilde{c}_2 = \frac{2f_0-f_1-f_2}{3} \\ \tilde{c}_1 - \tilde{c}_2 = \frac{-i\sqrt{3}f_1+i\sqrt{3}f_2}{3} \\ i(\tilde{c}_1 - \tilde{c}_2) = \frac{\sqrt{3}f_1-\sqrt{3}f_2}{3} \end{cases}$

$$p(x) = \frac{f_0+f_1+f_2}{3} + \frac{2f_0-f_1-f_2}{3}\cos(x) + \frac{\sqrt{3}f_1-\sqrt{3}f_2}{3}\sin(x)$$

5.3.2 FFT

Sei $\gamma^{(N)} = Wf^{(N)} = N\tilde{c}^{(N)}$ und $N = 2n$, womit $w^N = 1$ und $w^n = -1$.
gerade approximative Fourier-Koeffizienten: $\gamma_{2k} = \sum_{j=0}^{n-1}(w^2)^{kj}(f_j + f_{j+n})$
ungerade approx. Fourier-Koeffizienten: $\gamma_{2k+1} = \sum_{j=0}^{n-1}(w^2)^{kj}(f_j - f_{j+n})w^j$

Die Anzahl komplexer Multiplikationen beträgt $\mu = \frac{N}{2}\log_2(\frac{N}{2})$.

Wir betrachten den Fall $N = 2^m$, hier konkret: $m = 3$, $N = 8$, $n = 4$.
Es gilt: $w = e^{-i\frac{2\pi}{N}}$, $w^8 = 1$, $w^4 = -1$

$$W^{(N)} = \begin{pmatrix} \gamma_0 \\ \gamma_2 \\ \gamma_4 \\ \gamma_6 \\ \hline \gamma_1 \\ \gamma_3 \\ \gamma_5 \\ \gamma_7 \end{pmatrix} = \left(\begin{array}{cccc|cccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & w^2 & w^4 & w^6 & 1 & w^2 & w^4 & w^6 \\ 1 & w^4 & 1 & w^4 & 1 & w^4 & 1 & w^4 \\ 1 & w^6 & w^4 & w^2 & 1 & w^6 & w^4 & w^2 \\ \hline 1 & w & w^2 & w^3 & w^4 & w^5 & w^6 & w^7 \\ 1 & w^3 & w^6 & w & w^4 & w^7 & w^2 & w^5 \\ 1 & w^5 & w^2 & w^7 & w^4 & w & w^6 & w^3 \\ 1 & w^7 & w^6 & w^5 & w^4 & w^3 & w^2 & w \end{array}\right)$$

Das ganze faktorisiert ergibt: $\widehat{W}^{(N)} = \left(\begin{array}{c|c} W^{(n)} & 0 \\ \hline 0 & W^{(n)} \end{array}\right) D^{(N)}$

$$\widehat{W}^{(N)} = \left(\begin{array}{cccc|cccc} 1 & 1 & 1 & 1 & & & & \\ 1 & w^2 & w^4 & w^6 & & & & \\ 1 & w^4 & 1 & w^4 & & & & \\ 1 & w^6 & w^4 & w^2 & & & & \\ \hline & & & & 1 & 1 & 1 & 1 \\ & & & & 1 & w^2 & w^4 & w^6 \\ & & & & 1 & w^4 & 1 & w^4 \\ & & & & 1 & w^6 & w^4 & w^2 \end{array}\right)$$

$$\left(\begin{array}{cccc|cccc} 1 & & & & 1 & & & \\ & 1 & & & & 1 & & \\ & & 1 & & & & 1 & \\ & & & 1 & & & & 1 \\ \hline 1 & & & & w^4 & & & \\ & w & & & & w^5 & & \\ & & w^2 & & & & w^6 & \\ & & & w^3 & & & & w^7 \end{array}\right) \quad \text{mit} \quad \begin{cases} w^4 = -1 \\ w^5 = -w \\ w^6 = -w^2 \\ w^7 = -w^3 \end{cases}$$

Die Matrixmultiplikation $z = D^{(N)}f^{(N)}$ lässt sich einfach durchführen. Die übriggebliebene Rechnung $\widehat{\gamma}^{(N)} = \left(\begin{array}{c|c} W^{(n)} & 0 \\ \hline 0 & W^{(n)} \end{array}\right) z^{(N)}$ lässt sich wiederum auf zwei diskrete Fourier-Transformationen der Ordnung $n = 4$, im nächsten Schritt weitere Rückführung auf $n = 2$ und dann $n = 1$

```
1  function[]=my_fft
2  N=20; m=5; M=N*2^m;
3  % N big    --> more initial data
4  % m,M big --> more interpolation points
5  x_k=2*pi/N*[0:N-1]';      % initial: n points
6  f_k=sin(4*x_k).^2;        % given (example)
7  c=fft(f_k/N);             % analysis
8  c_star=[c(1:N/2);zeros(M-N,1);c(N/2+1:end)];
9  x_star=2*pi/M*[0:M-1]'; % new: m>n points
10 f_star=M*ifft(c_star);    % synthesis
```

⊞ Aufwand fft: $O(N\log_2 N)$
Aufwand dft: $O(N^2)$

6 Numerische Integration und Differentiation

6.1 Numerische Integration - Quadratur

Bei der numerischen Integration soll das Integral $I = \int_a^b f(x)dx$ mit $\widetilde{I}$ approximiert werden. Es werden nun einige dieser Verfahren vorgestellt.

6.1.1 Trapezmethode

- **Idee:** Unterteilung der Funktion in n Intervalle der Länge $h = \frac{b-a}{n}$ und approximiere linear mit $T(h) = \frac{h}{2}[f_a + f_b]$

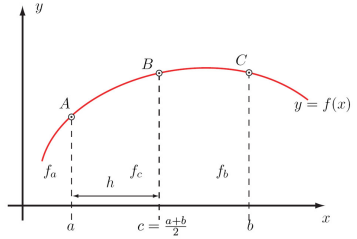
$$\widetilde{I} = T(h) = \frac{h}{2}[f_0 + f_1] + \frac{h}{2}[f_1 + f_2] + \cdots + \frac{h}{2}[f_{n-1} + f_n]$$
$$= h\left[\frac{1}{2}f_0 + f_1 + \cdots + f_{n-1} + \frac{1}{2}f_n\right]$$

- Das Trapezverfahren approximiert mind. Polynome des 1. Grades exakt.
- Abschätzung: $|I - T(h)| \le \frac{M}{12}(b-a)h^2$, $M \ge |f''(x)|$, $x \in [a,b]$
- Das Trapezverfahren hat Ordnung 2.

Trapezverfahren mit Schritthalbierung

- **Start:** $a,\ b,\ RTOL,\ ATOL$
- **1.Schritt:**
  - i $h_0 = b - a$
  - ii $s_0 = \frac{1}{2}(f(a) + f(b))$
  - iii $T_0 = s_0 h_0$
  - iv $N_0 = 1$
- **weitere Schritte:** $(n = 0, 1, 2, \dots)$
  - i $h_{n+1} = \frac{h_n}{2}$
  - ii $s_{n+1} = s_n + \sum_{j=1}^{N_n} f(a + (2j-1)h_{n+1})$
  - iii $T_{n+1} = h_{n+1}s_{n+1}$
  - iv $N_{n+1} = 2N_n$
- **Abbruchbedingung:** $|T_{n+1} - T_n| \le |T_{n+1}| \cdot RTOL + ATOL$

6.1.2 Simpson'sche Formel



- **Idee:** Unterteilung der Funktion in $2n$ Intervalle der Länge $h = \frac{b-a}{2n}$ und approximiere quadratisch mit $S(h) = \frac{h}{3}[f_a + 4f_c + f_b]$

$$\tilde{I} = S(h) = \frac{h}{3}[f_0 + 4f_1 + f_2] + \frac{h}{3}[f_2 + 4f_3 + f_4] + \cdots + \frac{h}{3}[f_{2n-2} + 4f_{2n-1} + f_{2n}]$$
$$= \frac{h}{3}[f_0 + 4f_1 + 2f_2 + 4f_3 + 2f_4 + \cdots + 2f_{2n-2} + 4f_{2n-1} + f_{2n}]$$

- Die Simpson-Methode approximiert mindestens Polynome des 3. Grades exakt.
- Abschätzung: $|I - S(h)| \leq \frac{M}{180}(b-a)h^4,\ M \geq |f^{(4)}(x)|,\ x \in [a,b]$
- Die Simpson-Methode hat Ordnung 4.

SIMPSON-METHODE MIT TRAPEZWERTEN

- **Start:** $a,\ b,\ RTOL,\ ATOL$
- **Berechnung der Trapezwerte:** $T_0,\ T_1,\ T_2, \ldots$
- **Berechnung der Simpsonwerte:** $S_1 = \frac{4T_1 - T_0}{3},\ S_2 = \frac{4T_2 - T_1}{3},\ S_3 = \frac{4T_3 - T_2}{3}, \ldots$
- **Abbruchbedingung:** $|S_{n+1} - S_n| \leq |S_{n+1}| \cdot RTOL + ATOL$

6.1.3 Romberg-Verfahren

- **Idee:** Durch Kombination von verschiedenen Trapezwerten kann man Fehlerterme tiefer Ordnung eliminieren und so die Ordnung des Verfahrens erhöhen → Simpsonwerte. Mit diesen verfährt man genauso...:

$R_{0,0} = T(h) = I + c_1 h^2 + c_2 h^4 + \ldots \qquad T(\frac{h}{2}) = I + \frac{1}{4}c_1 h^2 + \frac{1}{16}c_2 h^4 + \ldots$

$R_{1,1} = S_1 = S\left(\frac{h}{2}\right) = \frac{4T(\frac{h}{2}) - T(h)}{3} = \frac{T\left(\frac{h}{2}\right) - 4^{-1}T(h)}{1-4^{-1}} = I - \frac{1}{4}c_2 h^4 + O(h^6)$

$R_{2,2} = \frac{16S(\frac{h}{4}) - S(\frac{h}{2})}{15} = \frac{S\left(\frac{h}{4}\right) - 4^{-2}S\left(\frac{h}{2}\right)}{1-4^{-2}} = I + O(h^6)$

ROMBERG-VERFAHREN

Fehler: $O(h^2)$ $O(h^4)$ $O(h^6)$

h — $R_{0,0}$
$\frac{h}{2}$ — $R_{1,0}$ — $R_{1,1}$
$\frac{h}{4}$ — $R_{2,0}$ — $R_{2,1}$ — $R_{2,2}$
$\vdots$ $\vdots$ $\vdots$ $\ddots$

Start:
$h_0 = b - a,\ h_1 = \frac{h_0}{2},\ h_2 = \frac{h_1}{2}, \ldots$
$RTOL,\ ATOL$

Romberg-Verfahren:
$R_{j,0} = T_j$
$R_{j,k} = \frac{R_{j,k-1} - 4^{-k}R_{j-1,k-1}}{1-4^{-k}}$

Abbruchbedingung: $|R_{j,j} - R_{j,j-1}| \leq |R_{j,j}| \cdot RTOL + ATOL$

6.1.4 Guass'sche Quadraturformeln

Die Gauss'sche Quadraturformel gilt für das Integral $I = \int_{-1}^{1} f(x)dx$. Das allgemeine Integral $I = \int_a^b g(x)dx$ muss daher in diese Form transformiert werden.
Die n-Punkt Quadraturformel lautet:

$$\tilde{I} = Q_n = \sum_{j=1}^{n} w_j f_j$$

, wobei die Funtionswerte f_j und die Gewichte w_j

$$f(x) = \frac{b-a}{2} g\left(\frac{b-a}{2}x + \frac{a+b}{2}\right)$$
$$w_j = \int_{-1}^{1} \left[\prod_{k=1, k\neq j}^{n} \left(\frac{x - x_k}{x_j - x_k}\right)^2\right] dx > 0 \qquad j = 1, 2, \ldots, n$$

- ⊞ Eine n-Punkt-Quadraturformel hat Genauigkeitsgrad von (2n-1), wenn die Nullstellen der Legende-Polynome verwendet werden.
- ⊞ Es werden viel weniger Funktionsaufrufe benötigt.
- ⊟ Aber dafür müssen viele Funktionswerte bekannt sein.

6.2 Numerische Differentiation

Notwendig ist numerische Differentiation, wenn nur einige Werte der Funktion bekannt sind. Mit der Talyorentwicklung um z:
$f(z+h) = f(z) + \frac{h}{1!}f'(z) + \frac{h^2}{2!}f''(z) + \cdots + \frac{h^k}{k!}f^{(k)}(z)$
$f(z-h) = f(z) - \frac{h}{1!}f'(z) + \frac{h^2}{2!}f''(z) - \ldots$
...lassen sich Approximationen für $f'(z)$ herleiten:

Vorwärts-Differenzenquotient: $\frac{f(z+h)-f(z)}{h} = f'(z) + \frac{h}{2}f'' + O(h^2)$

Rückwärts-Differenzenquotient: $\frac{f(z)-f(z-h)}{h} = f'(z) - \frac{h}{2}f'' + O(h^2)$

Symmetrische Differenzquotienten der Ordnung 1 , 2 und n:

$$\Delta_h^1(z) = \frac{f(z+h) - f(z-h)}{2h} = f'(z) + \frac{h^2}{6}f'''(z) + O(h^3)$$
$$\Delta_h^2(z) = \frac{f(z+h) - 2f(z) + f(z-h)}{h^2} = f'' + O(h^2)$$
$$\Delta_h^n(z) = \frac{1}{(2h)^n}\sum_{j=0}^{n}(-1)^j a_j f(z + b_j h) \qquad a_j = \binom{n}{j} \qquad b_j = n - 2j$$

mit dem Binomialkoeffizienten a_j `TR::ncr(n,j)`.
Für grosse h ist das Ergebnis schlecht, doch auch zu kleine h resultieren in schlechten Ergebnissen aufgrund Auslöschung.
$h_{optimal} \cong \sqrt{10^{-d} \cdot 2\frac{|f(z)|}{|f''(z)|}}$, d-stellige Gleitpunktarithmetik

7 Gewöhnliche Differentialgleichungen

Gegeben: Die DGL $\dot{x} = f(t,x)$, sowie der Anfangswert $x(t_0) = x_0$.
Gesucht: Die Approximation an der Endstelle $\overline{x}(t_f)$.

7.1 Explizite Einschrittverfahren

7.1.1 Das explizite Eulerverfahren

Wir approximieren $x(t+h)$ durch das Taylorpolynom vom Grad 1:
$x(t) + \dot{x}(t)h$. Per Definition ist $\dot{x}(t) = f(t, x(t))$ und somit
$F(t,x,h) := x + hf(t,x)$.
Das Eulerverfahren ist wie folgt definiert:

$$\widetilde{x}_{j+1} = F(t_j, \widetilde{x}_j, h_j) = \widetilde{x}_j + h_j f(t_j, \widetilde{x}_j) \qquad \tilde{x}_0 = x0 \quad j = 0, 1, \ldots, n-1$$

Bei äquidistanten Schritten gilt: $h := \frac{t_f - t_0}{n} \quad t_j := t_0 + jh$.

```
function[x_tf]=my_eeuler_main
f1=@(x,t) x^2+t^2;   % function handle
h=0.1;               % ^-- on f=xprime
x0=0;
t0=0;    tf=1;
n=(tf-t0)/h;
x_tf=my_eeuler(f1,t0,x0,h,n);
```

```
function[x]=my_eeuler(f,t0,x0,h,n)
x=x0;
t=t0;
for it=1:n
    x=x+h*f(x,t)
    t=t+h;
end
```

Lokaler Fehler bei $j+1$: $l(t_j, \widetilde{x}_j, h) := \underbrace{F(t_j, \widetilde{x}_j, h)}_{\widetilde{x}_{j+1}} - \overline{x}(t_j + h)$

wobei $\overline{x}(t_j + h)$ die Lösung an der Stelle $t_j + h$ ist.

Globaler Fehler für $t = t_j$: $\widetilde{x}_j - x(t_j)$

Die **Fehlerordnung** p (Ordnung des globalen Fehlers) eines Verfahrens mit Verfahrensfunktion $F(t,x,h)$ ist gleich der Anzahl übereinstimmender Terme in den Taylorentwicklungen nach h von $x(t+h)$ und von $F(t, x(t), h)$.

- Der lokale Fehler des Verfahrens ist von der Ordnung $O(h^{p+1})$, beim expliziten Euler $O(h^2)$.
- Der globale Fehler des Verfahrens ist von der Ordnung $O(h^p)$, beim expliziten Euler $O(h)$. Ein k-fach kleinerer Schritt h ergibt einen k-fach kleineren globalen Fehler.

7.1.2 Taylorverfahren höherer Ordnung

Ordnung 2: Wir approximieren $x(t+h)$ durch:
$x(t) + \dot{x}(t)h + \ddot{x}(t)\frac{h^2}{2}$ (Taylorpolynom vom Grad 2)
$\ddot{x}(t) = f_t(t, x(t)) + f_x(t, x(t))\dot{x}(t) = f_t(t, x(t)) + f_x(t, x(t))f(t, x(t))$
$F(t,x,h) = x + hf(t,x) + \frac{h^2}{2}[f_t(t,x) + f_x(t,x)f(t,x)]$
Das Taylor-Verfahren der Ordnung 2 ist also wie folgt definiert:

$$\widetilde{x}_{j+1} = F(t_j, \widetilde{x}_j, h_j) = \widetilde{x}_j + h_j f(t_j, \widetilde{x}_j) + \frac{h_j^2}{2}[f_t(t_j, \widetilde{x}_j) + f_x(t_j, \widetilde{x}_j)f(t_j, \widetilde{x}_j)]$$

- ⊞ $p = 2$, lokaler Fehler somit $O(h^3)$ und globaler Fehler ist $O(h^2)$.
- ⊟ Die partiellen Ableitungen f_t und f_x müssen bekannt sein.

Grad p: Wir approximieren $x(t+h)$ durch:
$x(t) + \dot{x}(t)h + \cdots + x^{(p)}(t)\frac{h^2}{p!}$ (das Taylorpolynom vom Grad p)

- ⊞ $p = n$, lokaler Fehler somit $O(h^{n+1})$ und globaler Fehler ist $O(h^n)$.
- ⊟ Die partiellen Ableitungen von f bis zur Ordnung $n-1$ müssen bekannt sein.

7.1.3 Runge-Kutta-Verfahren (ableitungsfrei)

Idee: Simulation der partiellen Ableitungen durch Verschachtelung von Funktionsauswertungen.

BUTCHER-TABLEAU: EXPLIZITE VERFAHREN

$$\begin{array}{c|cccc} c_1=0 & & & & \\ c_2 & a_{21} & & & \\ c_3 & a_{31} & a_{32} & & \\ \vdots & \vdots & \vdots & \ddots & \\ c_s & a_{s1} & a_{s2} & \dots & a_{s,s-1} \\ \hline & b_1 & b_2 & \dots & b_{s-1} \quad b_s \end{array}$$

- $c_i = \sum_{j=1}^{i-1} a_{ij}$
- $k_i = f(t + c_i h, x + h\sum_{j=1}^{i-1} a_{ij} k_j), \qquad i = 1, 2, \dots, s$
- $\overline{x} = x + h\sum_{i=1}^{s} b_i k_i$

Das explizite Eulerverfahren:

$$\begin{array}{c|c} 0 & \\ \hline & 1 \end{array} \quad s=1, \; p=1 \qquad \begin{cases} k_1 &= f(t,x) \\ \overline{x} &= x + hk_1 \\ &= x + hf(t,x) \end{cases}$$

Das Verfahren von Heun:

$$\begin{array}{c|cc} 0 & & \\ 1 & 1 & \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array} \quad s=2, \; p=2 \qquad \begin{cases} k_1 &= f(t,x) \\ k_2 &= f(t+h, x+hk_1) \\ \overline{x} &= x + h[\frac{1}{2}k_1 + \frac{1}{2}k_2] \\ &= x + \frac{h}{2}[f(t,x) + f(t+h, x+hf(t,x))] \end{cases}$$

Das Runge-Kutta-Verfahren 3.Ordnung RK3:

$$\begin{array}{c|ccc} 0 & & & \\ \frac{1}{2} & \frac{1}{2} & & \\ 1 & -1 & 2 & \\ \hline & \frac{1}{6} & \frac{4}{6} & \frac{1}{6} \end{array} \quad s=3, \; p=3 \qquad \begin{cases} k_1 &= f(t,x) \\ k_2 &= f(t+\frac{h}{2}, x+\frac{h}{2}k_1) \\ k_3 &= f(t+h, x-hk_1+2hk_2) \\ \overline{x} &= x + h[\frac{1}{6}k_1 + \frac{4}{6}k_2 + \frac{1}{6}k_3] \end{cases}$$

Das klassische Runge-Kutta-Verfahren RK4:

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{2}{6} & \frac{2}{6} & \frac{1}{6} \end{array} \quad s=4, \; p=4 \qquad \begin{cases} k_1 &= f(t,x) \\ k_2 &= f(t+\frac{h}{2}, x+\frac{h}{2}k_1) \\ k_3 &= f(t+\frac{h}{2}, x+\frac{h}{2}k_2) \\ k_4 &= f(t+h, x+hk_3) \\ \overline{x} &= x + \frac{h}{6}[k_1 + 2k_2 + 2k_3 + k_4] \end{cases}$$

- Ein Runge-Kutta-Verfahren ist genau dann **konsistent**, wenn $\sum_{i=1}^{s} b_i = 1$.
- **Butcher Barrier:** Für $p = 5$ existiert kein explizites Runge-Kutta-Verfahren der Fehlerordnung p mit $s = p$ Stufen.

✏ Zeige, dass das explizite Mittelpunktsverfahren $F(t,x,h) = x + hf(t + \frac{h}{2}, x + \frac{h}{2}f(t,x))$ mindestens die Fehlerordnung 2 hat.

Somit muss die Taylorentwicklung von $F(t,x,h)$ bis zur 3. Ordnung identisch sein mit der von $x(t+h)$:

$x(t+h) = x(t) + hx'(t) + \frac{h^2}{2}x''(t) + \frac{h^3}{6}x'''(t) + \dots$

$x(t+h) = x(t) + hx'(t) + \frac{h^2}{2}x''(t) + O(h^3)$

$f(t+\delta, x+\Delta) = f(t,x) + f_t(t,x)\delta + f_x(t,x)\Delta + O(\delta^2) + O(\delta\Delta) + O(\Delta^2)$

$f(t+\frac{h}{2}, x+\frac{h}{2}f(t,x)) = f(t,x) + f_t(t,x)\frac{h}{2} + f_x(t,x)\frac{h}{2}f(t,x) + O(\frac{h}{2}^2) + O(\frac{h}{2}[\frac{h}{2}f(t,x)]) + O([\frac{h}{2}f(t,x)]^2)$

$f(t+\frac{h}{2}, x+\frac{h}{2}f(t,x)) = f(t,x) + f_t(t,x)\frac{h}{2} + f_x(t,x)\frac{h}{2}f(t,x) + O(h^2)$

$F(t,x,h) = x + hf(t,x) + \frac{h^2}{2}[f_t(t,x) + f_x(t,x)f(t,x)] + O(h^3)$

, was genau $x(t+h)$ entspricht. Also ist der lokale Fehler $O(h^3)$ und der globale $O(h^2)$ und die Fehlerordnung von ≥ 2.

```
function[x]=my_butcher_rk3(f,x0,h,n)
A=[0 0 0;0.5 0 0;-1 2 0];          % Butcher Table
c=[0 0.5 1]'; b=[1/6 2/3 1/6]';
dim=length(x0); s=length(c);
K=zeros(dim,s); x=zeros(dim,n+1); x(:,1)=x0;
for j=1:n                          % n Schritte
    x_old=x(:,j);
    for i=1:s                      % s Stufen
        K(:,i)=f(t+c(i)*h,x_old+h*K*A(i,:)');
    end
    x(:,j+1)=x_old+h*K*b;
    t=t+h;
end
```

7.1.4 Variable Schrittweiten

Kleine Schritte, wenn die Lösung stark variiert - grosse Schritte, wenn sie wenig variiert. Also werden die Schrittweiten h_j so gewählt, dass der lokale Fehler gleich einer vorgegeben Toleranz TOL ist:

$|l(t_j, \tilde{x}_j, h_j)| = TOL$ da l nicht bekannt, Schätzung $\bar{l}$:

1 Berechne $\tilde{x}_{j+1} = F(t_j, \tilde{x}_j, h)$ mit einem ersten Verfahren F. (p)
2 Berechne $\hat{\tilde{x}}_{j+1} = \hat{F}(t_j, \tilde{x}_j, h)$ mit einem Vergleichsverfahren $\hat{F}$. ($\hat{p} = p+1$)
3 Setze $\bar{l}(t_j, \tilde{x}_j, h) := F(t_j, \tilde{x}_j, h) - \hat{F}(t_j, \tilde{x}_j, h)$.

SCHRITTWEITENSTEUERUNG

- Berechne $\bar{l}(t_j, \hat{x}_j, h_j) = \tilde{x}_{j+1} - \hat{\tilde{x}}_{j+1}$
  Richardson: $\bar{l}(t_j, \hat{x}_j, h_j) = \tilde{x}_{j+1} - \hat{\tilde{x}}_{j+1} \cdot \frac{1}{2^{p-1}}$
- Falls $|\bar{l}| > TOL$:
  i Verwerfe $\tilde{x}_{j+1}$
  ii Wähle $h^* = h_j \left(\frac{TOL}{|\bar{l}|}\right)^{\frac{1}{p+1}} \cdot Fak$ (z.B. $Fak = 0.8$)
  iii Setze $h_j := h^*$ und gehe zum Anfang.
- Falls $|\bar{l}| \leq TOL$:
  i Akzeptiere $\tilde{x}_{j+1}$ als Approximation für $x(t_{j+1})$
  ii Vorschlag: $h_{j+1} = h_j \left(\frac{TOL}{|\bar{l}|}\right)^{\frac{1}{p+1}} \cdot Fak$

```
>> [t,x]=ode23(@xprime,[t0,tf],y0);     Bogacki - Shampine
>> [t,x]=ode45(@xprime,[t0,tf],y0);     Dormand - Prince
```

Richardson: Gleiches Verfahren, als Vergleich 2 halbe Schritte:

```
function[t,y]=my_rhrdsn_eeuler(f,t0,tf,h0,tol)
t=t0; y=y0; h=h0; j=1; fak=0.8; p=1;
while t(j)<tf
    y(:,j+1)=y(:,j)+h*f(t(j),y(:,j));   % 1 x h
    y_tmp=y(:,j)+h/2*f(t(j),y(:,j));
    y_hat=y_tmp+h/2*f(t(j)+h/2,y_tmp); % 2 x h/2
    l=(y(:,j+1)-y_hat)/(2^p-1);
    h=h*(tol/(norm(l)))^(1/(p+1))*fak;
    if norm(l)<=tol
        t(j+1)=t(j)+h; j=j+1;
    end
end
err=norm(y_exact(t)-y);    % Fehler in der 2-Norm
hwidth=diff(t);            % Schrittweitenvektor
```

7.2 Implizite Einschrittverfahren

Bei den Impliziten Verfahren muss eine nichtlineare Gleichung in jedem Schritt gelöst werden z.B. mit dem Newton-Verfahren oder mit einer Fixpunktiteration.

BUTCHER-TABLEAU: IMPLIZITE VERFAHREN

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \dots & a_{1s} \\ c_2 & a_{21} & a_{22} & \dots & a_{2s} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ c_s & a_{s1} & a_{s2} & \dots & a_{ss} \\ \hline & b_1 & b_2 & \dots & b_s \end{array}$$

- $c_i = \sum_{j=1}^{s} a_{ij}$
- $k_i = f(t + c_i h, x + h\sum_{j=1}^{s} a_{ij} k_j), \qquad i = 1, 2, \dots, s$
- $\overline{x} = x + h\sum_{i=1}^{s} b_i k_i$

Das Theta-Verfahren:

- $\vartheta = 0$: Explizites Eulerverfahren ($p = 1, s = 1$)
- $\vartheta = \frac{1}{2}$: Implizite Mittelpunktsregel ($p = 2, s = 1$)
- $\vartheta = 1$: Implizites Eulerverfahren ($p = 1, s = 1$)

$$\begin{array}{c|cc} \vartheta & \vartheta & \\ \hline & 1 & \end{array} \quad s=1, \; p=1 \qquad \begin{cases} k_1 &= f(t+\vartheta h, x+\vartheta h k_1) \\ \overline{x} &= x + hk_1 \\ &= x + hf(t+\vartheta h, x+\vartheta h k_1) \end{cases}$$

Trapezmethode:

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & \frac{1}{2} & \frac{1}{2} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array} \quad s=2, \; p=2 \qquad \begin{cases} k_1 &= f(t,x) \\ k_2 &= f(t+h, x+\frac{h}{2}k_1 + \frac{h}{2}k_2) \\ \overline{x} &= x + \frac{h}{2}k_1 + \frac{h}{2}k_2 \\ &= x + \frac{h}{2}[f(t,x) + f(t+h, \overline{x})] \end{cases}$$

⊞ Das Stabilitätsgebiet ist grösser.
⊟ Der Aufwand ist auch grösser.

```
function[y]=my_impr(y0,t0,tf,h,maxit,tol)
N=(tf-t0)/h;
y=zeros(length(y0),N+1); y(:,1)=y0;
for k=1:N                    % n Schritte
  y(:,k+1)=...
  impr_step(@yprime,(k-1)*h,y(:,k),h,maxit,tol);
end
end
function[y]=impr_step(fct,t,y,h,maxit,tol)
y_old=y+h*fct(t,y);          % Start mit eEuler
it=0;
while it<maxit               % Fixpunktiteration
    it=it+1;
    y=y+h*fct(t+h/2,(y+y_old)/2);
    if norm(y-y_old)<=norm(y)*tol+tol
        break;
    end
    y_old=y;
    if (it==maxit)
    error(['no conv after ',int2str(it),' it']);
    end
end
end
function[f_]=yprime(t,y)
f_=[f_1(t,y);f_2(t,y);f_3(t,y);f_4(t,y)];
end
```

7.3 Systeme gewöhnlicher DGL

Wir betrachten das Anfangswertproblem der Dimension n.

$$\begin{cases} \dot{\underline{x}} = f(t,\underline{x}) \\ \underline{x}(t_0) = \underline{x}^0 \end{cases}, \quad \underline{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}, \quad \underline{f}(t,\underline{x}) = \begin{pmatrix} f_1(t,x_1,\ldots,x_n) \\ \vdots \\ f_n(t,x_1,\ldots,x_n) \end{pmatrix}$$

Für die numerische Integration n-dimensionaler Differentialgleichungssysteme gelten alle bisherigen Formeln, einfach neu mit $\underline{x}_i$ und $\underline{k}_i$
Auch die Schrittweitensteuerung funktioniert für Systeme wie im skalaren Fall. Der Betrag muss einfach durch die Norm ersetzt werden.

7.3.1 Differentialgleichungen höherer Ordnung

Differentialgleichungen der Ordnung k kann man als k-dimensionale Systeme 1. Ordnung schreiben:

$$\dddot{x} = g(t,x,\dot{x},\ddot{x}) \qquad \begin{cases} x_1 = x \\ x_2 = \dot{x} \\ x_3 = \ddot{x} \end{cases} \Rightarrow \begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = x_3 \\ \dot{x}_3 = g(t,x,\dot{x},\ddot{x}) \end{cases}$$

✏ Approximieren sie $x(h)$ von $\ddot{x} + 0.5\dot{x} + x = 0$ mit x_0 und $\dot{x}_0$.

$$\underline{f}(t,\underline{x}) = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} x_2 \\ -0.5x_2 - x_1 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -1 & -0.5 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

mit $\begin{pmatrix} x_1(0) \\ x_2(0) \end{pmatrix} = \begin{pmatrix} x_0 \\ \dot{x}_0 \end{pmatrix}$ $\Rightarrow$ ins entsprechende Vefahren einsetzen.

7.4 Mehrschrittverfahren

Versuche höhere Fehlerordnung eines Verfahrens anstatt durch Funktionsschachtelung durch Einbeziehen von Information an mehreren Stellen zu erreichen.

7.4.1 Adams-Bashforth-Verfahren

In der k-Schritt-Adams-Bashforth-Formel berechnet man ,mit einem Einschrittverfahren der gleichen Fehlerordnung oder mit einem der Fehlerordnung 1 und kleinen Schritten, zusätzlich zu x_0 $k-1$ weitere Startwerte $x_{-1},\ldots$, aus denen man dann dass Interpolationspolynom (Lagrange) und daraus x_{j+1} berechnet.

- Das Verfahren ist explizit, linear und hat die Fehlerordnung k.
- $k=1$ $\Rightarrow$ expliziter Euler

2-Schritt-Adams-Bashforth-Formel:

$$\tilde{x}_{j+1} = \tilde{x}_j + \frac{h}{2}[3f(t_j,\tilde{x}_j) - 1f(t_{j-1},\tilde{x}_{j-1})]$$

3-Schritt-Adams-Bashforth-Formel:

$$\tilde{x}_{j+1} = \tilde{x}_j + \frac{h}{12}[23f(t_j,\tilde{x}_j) - 16f(t_{j-1},\tilde{x}_{j-1}) + 5f(t_{j-2},\tilde{x}_{j-2})]$$

4-Schritt-Adams-Bashforth-Formel:

$$\tilde{x}_{j+1} = \tilde{x}_j + \frac{h}{24}[55f(t_j,\tilde{x}_j) - 59f(t_{j-1},\tilde{x}_{j-1}) + 37f(t_{j-2},\tilde{x}_{j-2}) - 9f(t_{j-3},\tilde{x}_{j-3})]$$

⊞ billiges Verfahren - nur 1 Funktionsauswertung pro Schritt
⊟ Startverfahren benötigt, Schrittweitensteuerung schwierig

7.4.2 Adams-Moulton-Verfahren

Wie AB, aber mit x_{j+1} im Interpolationspolynom. Das k-Schritt-Adams-Moulton-Verfahren

- ist implizit, linear und hat die Fehlerordnung $k+1$.
- $k=0$ $\Rightarrow$ impliziter Euler
- $k=1$ $\Rightarrow$ Trapezverfahren

2-Schritt-Adams-Moulton-Formel:

$$\tilde{x}_{j+1} = \tilde{x}_j + \frac{h}{12}[5f(t_{j+1},\tilde{x}_{j+1}) + 8f(t_j,\tilde{x}_j) - 1f(t_{j-1},\tilde{x}_{j-1})]$$

3-Schritt-Adams-Moulton-Formel:

$$\tilde{x}_{j+1} = \tilde{x}_j + \frac{h}{24}[9f(t_{j+1},\tilde{x}_{j+1}) + 19f(t_j,\tilde{x}_j) - 5f(t_{j-1},\tilde{x}_{j-1}) + f(t_{j-2},\tilde{x}_{j-2})]$$

4-Schritt-Adams-Moulton-Formel:

$$\tilde{x}_{j+1} = \tilde{x}_j + \frac{h}{720}[251f(t_{j+1},\tilde{x}_{j+1}) + 646f(t_j,\tilde{x}_j) - 264f(t_{j-1},\tilde{x}_{j-1}) + 106f(t_{j-2},\tilde{x}_{j-2}) - 19f(t_{j-3},\tilde{x}_{j-3})]$$

Die impliziten Gleichungen kann gut durch Fixpunktiteration mit Startwert $\tilde{x}^0_{j+1} = \tilde{x}_j$ gelöst werden.

⊞ höhere Fehlerordnung als AB
⊞ stabiler, da implizit
⊟ teurer als AB - 1 Fixpunktiteration pro Schritt

- Wenn rechtes $\tilde{x}_{j+1}$ mit AB berechnet, explizit, **Prädiktor-Korrektor-Methode**.

7.5 Stabilität von Einschrittverfahren

Das Stabilitätsverhalten eines Einschrittverfahrens wird untersucht, indem man es auf ein lineares System $\dot{y} = Ay$ anwendet, entkoppelt die Form $\dot{x} = \lambda x$ (λ Eigenwert von A) hat.

Stabilitätsintervall: $B = \{z \in \mathbb{R} \mid |R(z)| < 1\} \subset \mathbb{R}$

Stabilitätsgebiet: $A = \{z \in \mathbb{C} \mid |R(z)^2| < 1\} \subset \mathbb{C}$

Berechnen des Stabilitätsgebiets

i Schreibe $\widetilde{x}_{j+1}$ um, mit $f(x_j) = \lambda x_j$
ii Ersetze λh mit $z = x + iy$ oder $z = -1 + re^{i\varphi}$
iii Berechne die Stabilitätsfunktion $R(z) = \frac{\widetilde{x}_{j+1}}{\widetilde{x}_j}$
iv Stabilitätsgebiet $A = \{z \in \mathbb{C} \mid |R(z)|^2 < 1\}$ mit $|R(z)|^2 = R \cdot \overline{R}$

- **A-stabil:** Ein Einschrittverfahren dessen Stabilitätsgebiet die linke Halbebene von enthält, nennt man absolut stabil.

✏ expliziter Euler: $\widetilde{x}_{j+1} = h\lambda\widetilde{x}_j \Rightarrow \begin{cases} R(z) = 1+z \\ A = \{(1+x)^2 + y^2 < 1\} \end{cases}$

✏ Heun-Verfahren: $\widetilde{x}_{j+1} = \widetilde{x}_j + h\lambda\widetilde{x}_j + \frac{(h\lambda)^2}{2}\widetilde{x}_j$

$$\Rightarrow \begin{cases} R(z) = 1 + z + \frac{z^2}{2} = \frac{1}{2}(1 + r^2 e^{i2\varphi}) \\ A = \{r = \sqrt{-\cos(2\varphi) + \sqrt{\cos^2(2\varphi) + 3}}\} \end{cases}$$

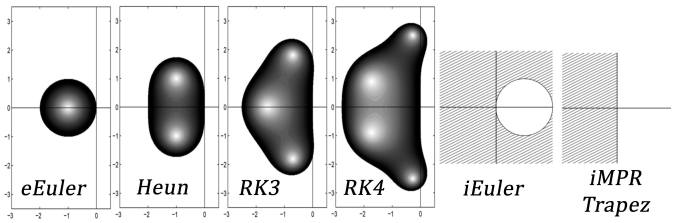
✏ RK3: $\widetilde{x}_{j+1} = \widetilde{x}_j[1 + h\lambda + \frac{(h\lambda)^2}{2} + \frac{(h\lambda)^3}{6}] \Rightarrow R(z) = 1 + z + \frac{z^2}{2} + \frac{z^3}{6}$

✏ RK4: $R(z) = 1 + z + \frac{z^2}{2} + \frac{z^3}{6} + \frac{z^4}{24}$

✏ impliziter Euler: $\widetilde{x}_{j+1} = \widetilde{x}_j + h\lambda\widetilde{x}_{j+1} \Rightarrow \begin{cases} R(z) = \frac{1}{1-z} \\ A = \{(1+x)^2 + y^2 > 1\} \end{cases}$

✏ Trapez-Methode: $\widetilde{x}_{j+1} = \widetilde{x}_j + \frac{h\lambda}{2}\widetilde{x}_j + \frac{h\lambda}{2}\widetilde{x}_{j+1}$

$$\Rightarrow \begin{cases} R(z) = \frac{1+\frac{z}{2}}{1-\frac{z}{2}} = \frac{2+z}{2-z} \\ A = \{\frac{(2+x)^2+y^2}{(2-x)^2+y^2} < 1\} = \{(2+x)^2 < (2-x)^2\} = \{x < 0\} \end{cases}$$



7.5.1 Steife Differentialgleichungen

Ein lineares (inhomogenes) Differentialgleichungssystem $\dot{\underline{x}} = A\underline{x} + \underline{b}$ heisst steif, falls ein Eigenwert von A mit negativem Realteil existiert, dessen Betrag sehr viel grösser ist als die der anderen Eigenwerte von A.
Wenn man nicht ausschliessen kann, dass ein Problem steif ist, sollte man zur numerischen Approximation der Lösung ein implizites Verfahren wählen, z.B. ein A-stabiles Verfahren.

Vorgehen beim Approximieren einer steifen DGL:

- In der boundary-layer-Region (wo der grosse Eigenwert wirkt) muss mit sehr kleinen Schritten integriert werden.
- Für die Wahl der Schrittweite ausserhalb der boundary-layer-Region wählt man h so, dass $R(\lambda h)$ innerhalb des Stabilitätsgebietes liegt:
  $x(t) = 0.01e^{-t} + 0.01e^{-100t}$
  $x(t+nh) = 0.01e^{-t}(e^{-h})^n + 0.01e^{-100t}(e^{-100h})^n$
  $\tilde{x}(t+nh) = 0.01e^{-t}(R(-h))^n + 0.01e^{-100t}(R(-100h))^n$
  also muss $R(-h) < e^{-h}$ und $R(-100h) < e^{-100h}$ erfüllt sein.

```
>> options=odeset('reltol',1e-2,'abstol',1e-2);
>> [t,y]=ode23s(@yprime,[t0 tf],y0,options);
```

8 Partielle Differentialgleichungen

Allgemeine lineare partielle DGL zweiter Ordnung:

$$Au_{xx} + 2Bu_{xy} + Cu_{yy} + Du_x + Eu_y + Fu = f(x,y)$$

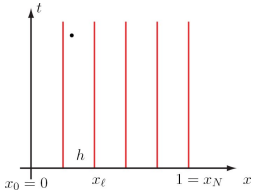
Klassifikation: $\Delta := AC - B^2 \quad \Rightarrow \quad \begin{array}{l} \Delta > 0: \text{ Elliptisch} \\ \Delta = 0: \text{ Parabolisch} \\ \Delta < 0: \text{ Hyperbolisch} \end{array}$

8.1 Parabolische Probleme

Eindimensionale Wärmeleitungsgleichung: $u_t = u_{xx}$
AB: $u(0,x) = f(x)$ RB: $u(t,0) = u(t,1) = 0$

8.1.1 Method of Lines

Diskretisierung der Ortskoordinate x, dann Anwenden der Methoden der numerischen Integration von gewöhnlichen DGL.



$u_{xx} = \frac{u(t,x+h)-2u(t,x)+u(t,x-h)}{h^2} + O(h^2)$
$h = \frac{1}{N} \qquad x_l = lh$
$l = 0,1,\ldots,N$

Die Approximation u_l mit $l = 1,\ldots,N-1$:
DGL: $\dot{u}_l(t) = \frac{u_{l+1}(t)-2u_l(t)+u_{l-1}(t)}{h^2}$;
AB: $u_l(0) = f(x_l)$
RB: $u_0(t) = 0 \quad u_N(t) = 0$

führt mit $A = \frac{1}{h^2}\hat{A}$ auf das Gleichungssystem

$$\underbrace{\begin{pmatrix} \dot{u}_1 \\ \dot{u}_2 \\ \dot{u}_3 \\ \vdots \\ \dot{u}_{N-1} \end{pmatrix}}_{\dot{\underline{u}}} = \underbrace{\frac{1}{h^2}\underbrace{\begin{pmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{pmatrix}}_{\hat{A}}}_{A} \underbrace{\begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{N-1} \end{pmatrix}}_{\underline{u}}$$

mit den Anfangswerten $\underline{u}(0) = \underline{f} = \begin{pmatrix} f(x_1) \\ \vdots \\ f(x_{N-1}) \end{pmatrix}$.

Richardson-Methode (mit explizitem Eulerverfahren):

$$\underline{\tilde{u}}_{j+1} = \underline{\tilde{u}}_j + \bar{h}A\underline{\tilde{u}}_j \qquad \text{mit} \quad j = 0,1,\ldots$$

- ⊞ pro Schritt eine Matrix-Vektor-Multiplikation
- ⊟ Einschränkung Stabilität: $\frac{\bar{h}}{h^2} \le \frac{1}{2}$
- ⊟ globaler Fehler $O(h^2) + O(\bar{h})$

Crank-Nicolson (mit Trapezverfahren):

$$(2\cdot\mathbb{1} - \bar{h}A)\underline{\tilde{u}}_{j+1} = (2\cdot\mathbb{1} + \bar{h}A)\underline{\tilde{u}}_j \qquad \text{mit} \quad j = 0,1,\ldots$$

- ⊞ globaler Fehler $O(h^2) + O(\bar{h}^2)$
- ⊟ pro Schritt ein lineares Gleichungssystem lösen

- Parabolische Probleme führen mit dieser Methode immer auf steife ODE's.

✏ **Gegeben** sei die folgende Wärmeleitungsgleichung und $N = 5$:

$$\begin{aligned} u_t &= u_{xx} - cos(2\pi x), && x \in (0,1) \\ u(0,x) &= 0, && x \in (0,1) \\ u(t,0) &= u(t,1) = 0, && t > 0 \end{aligned}$$

Gesucht ist $\underline{\tilde{u}}$ nach dem impliziten Eulerverfahren.

Die Ortsdiskretisierung mit der Method of Lines führt auf das folgende Gleichungssystem:

$$\underbrace{\begin{pmatrix} \dot{u}_1 \\ \dot{u}_2 \\ \dot{u}_3 \\ \dot{u}_4 \end{pmatrix}}_{\dot{\underline{u}}} = \underbrace{\frac{1}{h^2}\begin{pmatrix} -2 & 1 & & \\ 1 & -2 & 1 & \\ & 1 & -2 & 1 \\ & & 1 & -2 \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix}}_{\underline{u}} - \underbrace{\begin{pmatrix} \cos(\frac{2\pi}{5}) \\ \cos(\frac{4\pi}{5}) \\ \cos(\frac{6\pi}{5}) \\ \cos(\frac{8\pi}{5}) \end{pmatrix}}_{\cos(2\pi\underline{x})}$$

mit den Anfangsbedingungen $\underline{u}(0) = \underline{0}$
Der allgemeine Schritt des impliziten Eulerverfahrens lautet

$$\underline{\tilde{u}}_{j+1} = \underline{\tilde{u}}_j + \bar{h}(A\underline{\tilde{u}}_{j+1} - \cos(2\pi\underline{x})) \qquad \text{mit} \quad j = 0,1,\ldots$$

$$(\mathbb{1} - \bar{h}A)\underline{\tilde{u}}_{j+1} = \underline{\tilde{u}}_j - \bar{h}\cos(2\pi\underline{x})$$

```
>> n=N-1;  >> v=ones(n,1);
>> A=(1/h^2)*spdiags([v -2*v v],-1:1,n,n);
>> [u_j+1]=(eye(4)-h_bar*A)\(u_j-h_bar*cos(2*pi*x));
```

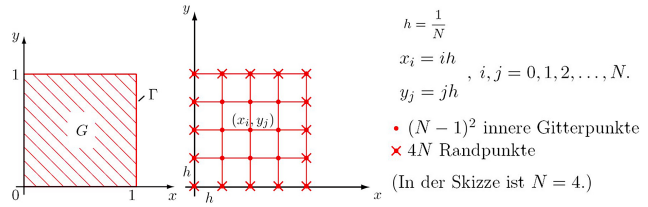
8.2 Elliptische Probleme

Poissongleichung: $u_{xx} + u_{yy} = f(x,y)$
Gegeben ist ein Einheitsquadrat G in der x-y-Ebene mit

- $f(x,y)$ in G
- $\varphi(x,y)$ auf dem Rand Γ

Gesucht ist $u(x,y)$, so dass:

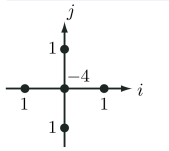
- $\Delta u = u_{xx} + u_{yy} = f(x,y)$ für $(x,y) \in G$
- $u(x,y) = \varphi(x,y)$ für $(x,y) \in \Gamma$



$$u_{xx} = \frac{u(x+h,y) - 2u(x,y) + u(x-h,y)}{h^2} + O(h^2)$$
$$u_{yy} = \frac{u(x,y+h) - 2u(x,y) + u(x,y-h)}{h^2} + O(h^2)$$

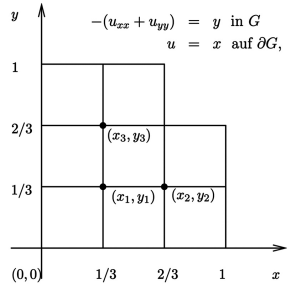
8.2.1 5-Punkte-Differenzenverfahren: quadratisches Gitter

$$f(x_i,y_j) = \frac{1}{h^2}(\tilde{u}_{i+1,j} + \tilde{u}_{i-1,j} + \tilde{u}_{i,j+1} + \tilde{u}_{i,j-1} - 4\tilde{u}_{i,j})$$



- Lokaler Diskretisationsfehler ≠ Lokaler Fehler in ODE's
  $l_{5Pdiff} = O(h^4) \neq ldis_{5Pdiff} = O(h^2)$
  $l_{Eeuler} = O(h^2) \neq ldis_{Eeuler} = O(h)$
- Pro innerem Punkt 1 Gleichung $\Rightarrow$ $(N-1)^2$ Gln

✏ Bestimme das lineare Gleichungssystem $A\underline{\tilde{u}} = \underline{b}$ des gezeigten Gebietes:



Aus den 3 inneren Punkten folgen 3 Gleichungen mit dem 5-Punkte-Differenzenverfahren und $N = 3$:

$$\tilde{u}_{i+1,j} + \tilde{u}_{i-1,j} + \tilde{u}_{i,j+1} + \tilde{u}_{i,j-1} - 4\tilde{u}_{i,j}$$
$$= h^2 \cdot f = \frac{f}{N^2} = \frac{-y}{N^2} = \frac{-y}{9}$$

mit $f = -y$ und $\varphi = x$

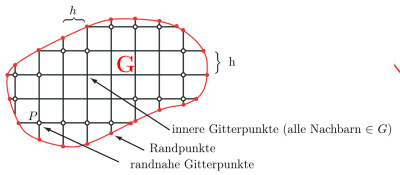
Das lineare Gleichungssystem $A\underline{\tilde{u}} = \underline{b}$ sieht wie folgt aus:

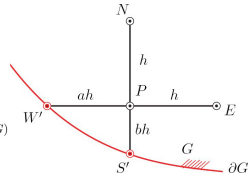
$$\begin{pmatrix} -4 & 1 & 1 \\ 1 & -4 & 0 \\ 1 & 0 & -4 \end{pmatrix}\begin{pmatrix} \tilde{u}_1 \\ \tilde{u}_2 \\ \tilde{u}_3 \end{pmatrix} = \begin{pmatrix} -\frac{y_1}{9} - \varphi(0,\frac{1}{3}) - \varphi(\frac{1}{3},0) \\ -\frac{y_2}{9} - \varphi(\frac{2}{3},0) - \varphi(\frac{2}{3},\frac{2}{3}) - \varphi(1,\frac{1}{3}) \\ -\frac{y_3}{9} - \varphi(0,\frac{2}{3}) - \varphi(\frac{1}{3},1) - \varphi(\frac{2}{3},\frac{2}{3}) \end{pmatrix}$$
$$= \begin{pmatrix} -\frac{1}{3}\frac{1}{9} - \frac{1}{3} \\ -\frac{2}{3}\frac{1}{9} - \frac{2}{3} \\ -\frac{2}{3}\frac{1}{9} - \frac{2}{3} - \frac{2}{3} - 1 \end{pmatrix} = \begin{pmatrix} -\frac{10}{27} \\ -\frac{64}{27} \\ -\frac{29}{27} \end{pmatrix}$$

✏ Diskretisiere und löse das folgende Problem mit dem CG-Verfahren:
$\Delta u(x,y) = f = -1 \quad G = (-1,1)\times(-1,1) \qquad u = 0 \quad \partial G$

```
1  function[u_vec]=my_poisson
2  N=100; h=2/(N-1);                  % L=2
3  A=delsq(numgrid('S',N));           % gives -A...
4  [rows,cols]=size(A);
5  b=-(-h^2*ones(rows,1));            % ... --> -()
6  u_vec=pcg(A,b,1e-6,1000);          % solves Ax=b
7  u=zeros(N,N);
8  u(2:N-1,2:N-1)=reshape(u_vec,N-2,N-2);
9  surf(-1:h:1,-1:h:1,u);
```

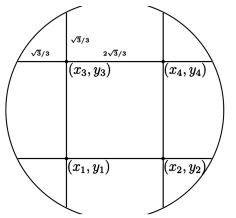
8.2.2 5-Punkte-Differenzenverfahren: allgemeines Gebiet



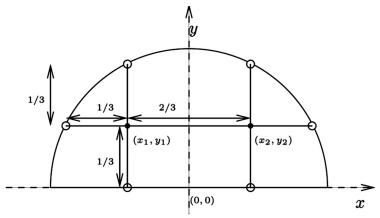


Die 5-Punkt-Formel gilt ganz allgemein (Dirichlet: $f = 0$):

$$\frac{2}{h^2}\left[\frac{\widetilde{u}(E)}{1+a}+\frac{\widetilde{u}(N)}{1+b}+\frac{\widetilde{u}(W')}{a(1+a)}+\frac{\widetilde{u}(S')}{b(1+b)}-(\frac{1}{a}+\frac{1}{b})\widetilde{u}(P)\right]=f(x,y)$$



(1) $\begin{array}{rcl} -\Delta u(x,y) &=& x+y^2 \text{ in } G \\ u &=& 2 \text{ auf } \partial G, \end{array}$



(2) $\begin{array}{rcl} -(u_{xx}+u_{yy}) &=& x+1 \text{ in } G \\ u &=& 1 \text{ auf } \partial G, \end{array}$

✏ Berechnen sie eine Approximation der Lösung von *(1)*:

Mit $h = \frac{2\sqrt{3}}{3} \Rightarrow \frac{2}{h^2} = \frac{3}{2}$ und $a = b = \frac{1}{2}$ folgt für den Punkt (x_1, y_1):

$\frac{3}{2}\left[\frac{\widetilde{u}_2}{\frac{3}{2}}+\frac{\widetilde{u}_3}{\frac{3}{2}}+\frac{2}{\frac{3}{4}}+\frac{2}{\frac{3}{4}}-4\widetilde{u}_1\right] = f(x,y) = -(x_1+y_1^2)$

$[\widetilde{u}_2+\widetilde{u}_3-6\widetilde{u}_1] = -(x_1+y_1^2)-8$ da $\frac{3}{2}(\frac{2}{\frac{3}{4}}+\frac{2}{\frac{3}{4}}) = 8$

mit analogen Gleichungen für die anderen Punkte folgt

$$\begin{pmatrix} -6 & 1 & 1 & 0 \\ 1 & -6 & 0 & 1 \\ 1 & 0 & -6 & 1 \\ 0 & 1 & 1 & -6 \end{pmatrix}\begin{pmatrix} \widetilde{u}_1 \\ \widetilde{u}_2 \\ \widetilde{u}_3 \\ \widetilde{u}_4 \end{pmatrix} = \begin{pmatrix} -x_1-y_1^2-8 \\ -x_2-y_2^2-8 \\ -x_3-y_3^2-8 \\ -x_4-y_4^2-8 \end{pmatrix}$$

✏ Berechnen sie eine Approximation der Lösung von *(2)*:

Mit $h = \frac{2}{3} \Rightarrow \frac{2}{h^2} = \frac{9}{2}$ und $a = b = \frac{1}{3}$ folgt für den Punkt (x_1, y_1):

$\frac{9}{2}\left[\frac{\widetilde{u}_2}{\frac{4}{3}}+\frac{1}{\frac{1}{3}\frac{4}{3}}+\frac{1}{\frac{1}{3}\frac{4}{3}}+\frac{1}{\frac{1}{3}\frac{4}{3}}-6\widetilde{u}_1\right] = f(x,y) = -(x_1+1)$

$[\frac{27}{8}\widetilde{u}_2-27\widetilde{u}_1] = -(x_1+1)-\frac{243}{8}$ da $\frac{9}{2}(\frac{3}{\frac{1}{3}\frac{4}{3}}) = \frac{243}{8}$ [...]

8.3 Hyperbolische Probleme

Eindimensionale Wellengleichung: $u_{tt} = c^2 u_{xx}$

- $u(0,x) = \varphi(x) \Rightarrow$ Anfangsamplitude
- $u_t(0,x) = \psi(x) \Rightarrow$ Anfangsgeschwindigkeit

$\Rightarrow$ Allgemeine Lösung: $u(t,x) = \underbrace{P(x+ct)}_{\text{Linkswelle}} + \underbrace{Q(x-ct)}_{\text{Rechtswelle}}$

- Die Geraden (Strahlen) $x \pm ct = const.$ heissen Charakteristiken der eindimensionalen Wellengleichung.

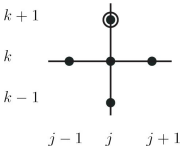
Setzt man die Anfangsbedingungen ein, erhält man:

$$u(t,x) = \frac{1}{2}\left[\varphi(x+ct)+\varphi(x-ct)\right]+\frac{1}{2c}\int_{x-ct}^{x+ct}\psi ds$$

8.3.1 Numerische Approximation

= Diskretisation mit symm. Diefferenzquotienten.

$$\frac{1}{\Delta t^2}\left(\widetilde{u}_j^{k+1}-2\widetilde{u}_j^k+\widetilde{u}_j^{k-1}\right) = \frac{c^2}{\Delta x^2}\left(\widetilde{u}_{j+1}^k-2\widetilde{u}_j^k+\widetilde{u}_j^k\right)$$



- mit $\lambda = c\frac{\Delta t}{\Delta x}$ folgt
  $\widetilde{u}_j^{k+1} = 2\widetilde{u}_j^k + \lambda^2(\widetilde{u}_{j+1}^k - 2\widetilde{u}_j^k + \widetilde{u}_{j-1}^k) - \widetilde{u}_j^{k-1}$
  $\widetilde{u}_j^0 = \varphi(x_j) \qquad j \in \mathbb{Z}$
  $\widetilde{u}_j^1 = \widetilde{u}_j^0 + \Delta t \cdot \psi(x_j) \qquad j \in \mathbb{Z}$
- Lokaler Diskretisationsfehler $O(\Delta t^2) + O(\Delta x^2)$

- **Kovergenz: CFL-Bedingung** (Courant-Friedrichs-Levy)

$$\frac{\Delta t}{\Delta x} \le \frac{1}{c} \quad \Leftrightarrow \quad \lambda \le 1$$

✏ 1-dimensionale Wellengleichung

DGL: $u_{tt} - u_{xx} = 0 \qquad 0 < x < L, \quad t > 0$

AB: $u(0,x) = \varphi(x) = \sin(\pi x)$

$u_t(0,x) = \psi(x) = 0 \qquad 0 \le x \le L$

RB: $u(t,0) = u(t,1) = 0 \qquad 0 < t$

Das Differenzenverfahren lautet (mit $c = 1 \Rightarrow \lambda = \frac{\Delta t}{\Delta x}$):

$\widetilde{u}_j^{k+1} = 2\widetilde{u}_j^k + (\frac{\Delta t}{\Delta x})^2(\widetilde{u}_{j+1}^k - 2\widetilde{u}_j^k + \widetilde{u}_{j-1}^k) - \widetilde{u}_j^{k-1}$

mit $\Delta t = \Delta x = 0.25$ folgt $\lambda = 1$ und so

$$\widetilde{u}_j^{k+1} = \widetilde{u}_{j+1}^k + \widetilde{u}_{j-1}^k - \widetilde{u}_j^{k-1} \quad \Rightarrow \quad \begin{pmatrix} \widetilde{u}_1^2 \\ \widetilde{u}_2^2 \\ \widetilde{u}_3^2 \end{pmatrix} = \begin{pmatrix} \widetilde{u}_2^1 + \widetilde{u}_0^1 - \widetilde{u}_1^0 \\ \widetilde{u}_3^1 + \widetilde{u}_1^1 - \widetilde{u}_2^0 \\ \widetilde{u}_4^1 + \widetilde{u}_2^1 - \widetilde{u}_3^0 \end{pmatrix}$$

da $\psi(x) = 0$ gilt $\widetilde{u}_j^1 = \widetilde{u}_j^0 = \varphi(x) = \sin(\pi x)$

$$\text{mit } \begin{pmatrix} \widetilde{u}_0^1 = \widetilde{u}_0^0 \\ \widetilde{u}_1^1 = \widetilde{u}_1^0 \\ \widetilde{u}_2^1 = \widetilde{u}_2^0 \\ \widetilde{u}_3^1 = \widetilde{u}_3^0 \\ \widetilde{u}_4^1 = \widetilde{u}_4^0 \end{pmatrix} = \begin{pmatrix} \sin(0) \\ \sin(\frac{\pi}{4}) \\ \sin(\frac{\pi}{4}) \\ \sin(\frac{3\pi}{4}) \\ \sin(\pi) \end{pmatrix} \quad \Rightarrow \quad \underline{\underline{\begin{pmatrix} \widetilde{u}_1^2 \\ \widetilde{u}_2^2 \\ \widetilde{u}_3^2 \end{pmatrix} = \begin{pmatrix} 1-\frac{\sqrt{2}}{2} \\ \sqrt{2}-1 \\ 1-\frac{\sqrt{2}}{2} \end{pmatrix}}}$$

✏ Angeschlagene Klaviersaite

DGL: $u_{tt} - u_{xx} = 0 \qquad 0 < x < L, \quad t > 0$

AB: $u(0,x) = \varphi(x) = 0$

$u_t(0,x) = \psi(x) = e^{-16(x-1.5)^2} \qquad 0 \le x \le L$

RB: $u(t,0) = u(t,L) = 0 \qquad 0 < t$

Diskretisierung mit $L = 3$, $N = 25$, $n = 100$, $\Delta t = \Delta x$. Approximation:

```
1  function[u]=my_wave
2  c=1;  L=3;  N=25;  n=100;
3  dx=L/N;  dt=dx;  tf=n*dt;  lbd=c*dt/dx;
4  x=(0:dx:L)';  t=(0:dt:tf)';  v=ones(N-1,1);
5  A=spdiags([v -2*v v],-1:1,N-1,N-1);
6  u=zeros(N+1,n+1);  u(:,1)=0;  u(1,:)=0;
7  u(2:N,2)=u(2:N,1)+dt*exp(-16*(x(2:N)-L/2).^2);
8  for k=2:n
9      u(2:N,k+1)=...
10       -u(2:N,k-1)+2*u(2:N,k)+lbd^2*A*u(2:N,k);
11 end
12 surfc(x,t,u');
```

9 Numerik des Eigenwertproblems

Das Berechnen der Eigenwerte mit dem Computer ist fehlerbehaftet. Der relative Fehler der Polynomkoeffizienten ϵ ist 10^{-d}, bei d-stelliger Arithmetik. Liegen 2 Eigenwerte nahe zusammen, so kann sich dieser Fehler noch verstärken.
Deshalb benützt man andere Methoden wie die *Vektoriteration* (um den betragsmässig grössten/kleinsten Eigenwert zu berechnen) oder das *Jacobi-Verfahren* (um alle Eigenwerte einer symmetrischen Matrix zu berechnen).

9.1 Vektoriteration

RÜCKWÄRTS-VEKTORITERATION $\rightarrow |\lambda|_{min}$

- **Start:** $y^0 \neq 0,\ TOL$
  - i Normieren: $N = \|y^0\| \qquad \hat{x}^0 = sign(y_1^0)\frac{1}{N}y^0$
  - ii Zerlegen: $A = LR$
- **k. Schritt:** $(k = 1, 2, \ldots)$
  - i Vorwärtseinsetzen: $Lc = \hat{x}^{k-1} \quad \Rightarrow \quad c = L\backslash(\hat{x}^{k-1})$
  - ii Rückwärtseinsetzen: $Ry^k = c \quad \Rightarrow \quad y^k = R\backslash c$
  - iii Normieren: $N = \|y^k\| \qquad \hat{x}^k = sign(y_1^k)\frac{1}{N}y^k$
- **Abbruchkriterium:** $\Delta := \|\hat{x}^k\| - \|\hat{x}^{k-1}\| \le TOL$
  - i Eigenvektor: $\hat{x}^{(1)} = \hat{x}^k$
  - ii betr. kleinster Eigenwert: $|\hat{\lambda}_{min}| = sign(y_1^k)\frac{1}{N}$

```
>> [L,R,P]=lu(A);  >> [L,R]=lu(A);    >> [N]=norm(x);
TR::LU A,L,R,P    TR::norm(x)
>> [T,D]=eig(A);  >> [val,pos]=min(abs(diag(D)));
>> x_min=T(:,pos);  >> lambda_min=val;
```

VORWÄRTS-VEKTORITERATION $\rightarrow |\lambda|_{max}$

- **Start:** $y^0 \neq 0,\ TOL$
  - i Normieren: $N = \|y^0\| \qquad \hat{x}^0 = sign(y_1^0)\frac{1}{N}y^0$
- **k. Schritt:** $(k = 1, 2, \ldots)$
  - i Vorwärtsiteration: $\hat{x}^k = A\hat{x}^{k-1}$
  - ii Normieren: $N = \|y^k\| \qquad \hat{x}^k = sign(y_1^k)\frac{1}{N}y^k$
- **Abbruchkriterium:** $\Delta := \|\hat{x}^k\| - \|\hat{x}^{k-1}\| \le TOL$
  - i Eigenvektor: $\hat{x}^{(n)} = \hat{x}^k$
  - ii betr. grösster Eigenwert: $|\hat{\lambda}_{max}| = sign(y_1^k)N$

9.2 Jacobi-Verfahren

JACOBI-VERFAHREN

Bilde $A_{k+1} = U_{pq}^T \cdot A_k \cdot U_{pq}$ bis A approx. diagonal.

mit $U_{pq} = \begin{matrix} \\ p \rightarrow \\ \\ q \rightarrow \\ \\ \end{matrix}\begin{pmatrix} \cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi \end{pmatrix}$ $\sin\phi = -\frac{A(q,p)}{\sqrt{A(p,p)^2+A(q,p)^2}}$ $\cos\phi = \frac{A(p,p)}{\sqrt{A(p,p)^2+A(q,p)^2}}$