Wissenschaftliches Rechnen - Zusammenfassung

Patrick Pletscher

5. Oktober 2004

1. Endliche Arithmetik

1.1. Reelle Zahlen ↔ Maschinenzahlen

Jeder Computer ist ein *endlicher* Automat, d.h. er kann nur

- endlich viele Operationen durchführen
- endlich viele Zahlen speichern

Die reellen Zahlen $\mathbb{R}$ werden durch die endliche Menge der Maschinenzahlen $\mathbb{M}$ approximiert. Dabei wird jeweils ein ganzes Intervall (z.B. alle reellen Zahlen deren ersten 10 Ziffern gleich sind) auf eine Maschinenzahl abgebildet.

Darstellung einer Maschinenzahl

$$\tilde{a} \in \mathbb{M} \qquad \tilde{a} = \pm m \cdot B^{\pm e}$$

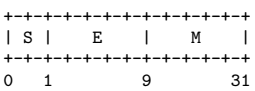
$m = D.D \cdots D$ Mantisse
$e = D \cdots D$: Exponent
B: Basis

Wobei D eine Ziffer $\in \{0, 1, \ldots, 9\}$

Heute ist B meistens 2 (nach dem IEEE Standard for floating Point operations), früher oftmals 10.

1.2. Gleitkommazahlen nach dem IEEE Standard

Single Precision (32 Bits)



S: Vorzeichen
E: Exponent 8 Bits
M: Mantisse 23 Bits

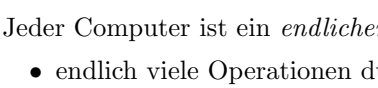
$$V = (-1)^S \cdot 1.M \cdot 2^{E-127} \quad \text{falls } 0 < E < 255$$

Ausnahmen:

$E = 255, M \neq 0$	$V = NaN$ (not a number)
$E = 255, M = 0, S = 0$	$V = Inf(= \infty)$
$E = 255, M = 0, S = 1$	$V = Inf(= -\infty)$
$E = 0, M \neq 0$	$V = (-1)^s \cdot 0.M \cdot 2^{-126}$ (nicht normalisierte Zahl)
$E = 0, M = 0, S = 1$	$V = -0$
$E = 0, M = 0, S = 0$	$V = 0$

Double Precision (64 Bits)

Double Precision ist heute der Normalfall.



Die Zahl ist:

$$V = (-1)^S \cdot 1.M \cdot 2^{E-1023} \quad \text{falls } 0 < E < 2047$$

anders geschrieben:

$$V = (-1)^S \cdot (1 + M/2^{52}) \cdot 2^{E-1023}$$

Ausnahmen:

$E = 2047, M \neq 0$	$V = NaN$ (not a number)
$E = 2047, M = 0, S = 0$	$V = Inf(= \infty)$
$E = 2047, M = 0, S = 1$	$V = Inf(= -\infty)$
$E = 0, M \neq 0$	$V = (-1)^s \cdot 0.M \cdot 2^{-1022}$ $V = (-1)^S \cdot M/2^{52} \cdot 2^{-102}$ (nicht normalisierte Zahl)
$E = 0, M = 0, S = 1$	$V = -0$
$E = 0, M = 0, S = 0$	$V = 0$

Es gibt eine grösste und eine kleinste Maschinenzahl. Der Rechenbereich ist $[-\texttt{realmax}, \texttt{realmax}]$, wobei $\texttt{realmax} = 2^{1024}$. Es gibt auch eine kleinste normalisierte Zahl $\texttt{realmin} = 2^{-1022}$. Die kleinste nicht normalisierte Zahl ist $0.0 \cdots 01 \cdot 2^{-1022}$.

Maschinengenauigkeit eps

Definition (Maschinengenauigkeit). Die Maschinengenauigkeit *eps* ist die kleinste pos. Maschinenzahl welche zu 1 addiert ein Resultat $\neq 1$ ergibt. Eine neuere Definition lautet: Der Abstand zwischen zwei aufeinanderfolgenden Floating Point Numbers zwischen 1 und 2.

In IEEE ist $eps = 2^{-52}$.

1.3. Rechnen mit Maschinenzahlen

Rundungsfehler

absoluter Rundungsfehler: $r_a = \tilde{c} - c$
relativer Rundungsfehler: $r = \frac{r_a}{c}$

Wie gross sind die Rundungsfehler? Bei heutigen Computern gilt:

$$a\tilde{\oplus}b = (a \oplus b)(1 + r)$$

Wobei r relativer Rundungsfehler mit $|r| \leq \varepsilon$ und $\oplus \in \{+, -, \times, /\}$.

Assoziativgesetz

Gesetze der Mathematik gelten nicht, so z.Bsp. Assoziativgesetz. Man muss also überlegen wie klammern. Für eine Summe ist es also von Vorteil mit dem kleinsten Element zu beginnen und das grösste Element erst als Letztes zu addieren.

Kahan's Summenalgorithmus

Um Summen auf dem Computer möglichst richtig zu berechnen, benutzt man den Algorithmus von Kahan.

$$s = \sum_{j=1}^{N} x_j$$

```
s = 0
c = 0
for j=1:N
  y = x(j) + c
  t = s + y
  c = (s-t)+y
  s = t
end
s = s + c
```

Monotonie

In IEEE wurden die Standardfunktionen sehr sorgfältig von W. Kahan implementiert. Aber im allgemeinen kann Monotonie nicht garantiert werden.

Vermeiden von Überlauf beim Radizieren

$$r = \sqrt{x^2 + y^2}$$

```
if abs(x) > abs(y)
  r = abs(x)*sqrt(1 + (y/x)^2)
elseif y == 0
  r = 0
else
  r = abs(y)*sqrt((x/y)^2 + 1)
```

Oder:

```
m = max(abs(x), abs(y))
if m == 0
  r = 0
else
  r = sqrt( (x/m)^2 + (y/m)^2 )*m
end
```

Test für Überlauf für x^2

Mit IEEE:

```
if x^2 == Inf
```

Ohne IEEE:

```
if (1/x)/x == 0
```

Noch besser ohne nichtnormalisierten Zahlen:

```
if (eps/x)/x == 0
```

numerische Auslöschung

Wenn 2 nahezu gleichgrosse Zahlen subtrahiert werden. Viele Bits der Mantisse sind dann unbekannt und werden mit 0 gefüllt.

1.4. Maschinenunabhängige Algorithmen

These: Gute Algorithmen funktionieren dank der Rundungsfehler!

Beispiel: Exponentialfunktion

```
function y= e2(x);
% stable computation of the exponential
% function using the series
if x<0, v=-1; x = abs(x); else v=1; end
sn = 1; term = 1; k=1;
while sn+term ~= sn
  s = sn; term = term*x/k;
  sn = s + term; k=k+1;
end
if v<0, y=1/sn; else y=sn; end
```

Beispiel: Quadratwurzel

```
xa = (1+a) / 2;
xn = (xa + a/xa) / 2;
while xn < xa
  xa = xn;
  xn = (xa+a/xa)/2;
end
```

1.5. Abbruchkriterium

Wann soll Iteration im Allgemeinen abgebrochen werden?

1. Wenn sich zwei 2 Näherungen nicht stark unterscheiden

   $$|x_{k+1} - x_k| < tol \quad \text{absolute Differenz}$$

   $$|x_{k+1} - x_k| < tol * |x_{k+1}| \quad \text{relativer "Fehler"}$$

2. Wenn Residuum $r = |x_k - x|$ klein ist, bzw. meist anderer Ausdruck, der die Abweichung vom zu erwartenden Resultat berechnet. Z.B. $r = |x_k^2 - a|$ für $x = \sqrt{a}$.

Diese beiden Bedingungen können aber nicht korrekte Resultate liefern! Darum sollte man immer Algorithmen benutzen, die keine Epsiliontik benutzen.

1.6. Kondition und Stabilität

Kondition: Definition

Ein Problem kann gut oder schlecht konditioniert sein. Gut konditioniert heisst: die Lösung eines "benachbarten" Problems (= Problem mit leicht geänderten Anfangsdaten)

unterscheidet sich nicht sehr von der ursprünglichen Lösung.

P mit Lösung x und $P(z)$ mit Lösung $x(\varepsilon)$.

Definition. P ist gut konditioniert wenn $|P - P(\varepsilon)| =$ klein $\Rightarrow |x - x(\varepsilon)| =$ klein.

Schlechte Kondition: Die Lösung ist sehr empfindlich auf Änderung der Anfangsdaten.

Gut/ Schlecht gestelltes Problem: Definition

Definition. Sei $A : X \to Y$ eine Abbildung eines Raumes X nach Y. Das Problem $Ax = y$ heisst gut gestellt (well posed) wenn

1. Zu jedem $y \in Y$ existiert eine Lösung $x \in X$.
2. Die Lösung ist eindeutig.
3. Die Lösung hängt stetig von den Daten ab.

Ist eine der drei Bedingungen verletzt dann ist das Problem *schlecht gestellt*. Ist 3 verletzt, so ist das Problem *schlecht konditioniert*.

Prinzip von Wilkinson

Rückwärtsfehleranalyse

$$\begin{aligned} a \cdot b &= a \cdot b \cdot (1 + r) \\ &= a \cdot (b + \underbrace{b \cdot r}_{\text{Störung von } b}) \end{aligned}$$

Das numerische Resultat $a \cdot b(1 + r)$ ist das exakte Resultat mit leicht gestörten Anfangsdaten $b + br$ statt b.

Kondition eines linearen Gleichungssystem

$$\mathbf{Ax} = \mathbf{b} \qquad \mathbf{A} \in \mathbb{R}^{n \times n} \text{ nicht singulär}$$

Gestörtes System:

$$(\mathbf{A} + \mathbf{E})\mathbf{y} = \mathbf{b}$$

Die Konditionszahl ist nun wie folgt definiert:

$$\frac{\| \mathbf{x} - \mathbf{y} \|}{\| \mathbf{y} \|} \leq \varepsilon \underbrace{\| \mathbf{A}^{-1} \| \| \mathbf{A} \|}_{\kappa(\mathbf{A})}$$

$\kappa(\mathbf{A})$ ist also die Norm von $\mathbf{A}$ multipliziert mit der Norm von $\mathbf{A}^{-1}$. Dabei wird $\kappa(A)$ als die Konditionszahl von $\mathbf{A}$ bezeichnet. Für die 2-Norm gilt

$$\kappa(\mathbf{A}) = \frac{\sigma_{\max}}{\sigma_{\min}}$$

$\sigma_{\max}$ bzw. $\sigma_{\min}$ ist der grösste bzw. kleinste Singulärwert: $\sigma_i = \sqrt{\lambda_i(\mathbf{A}^T\mathbf{A})}$.

Für spezielle Matrizen vereinfacht sich die Rechnung:

- Falls $\mathbf{A}$ eine symmetrische Matrix ist, so gilt

$$\kappa(\mathbf{A}) = \frac{\max_i |\lambda_i|}{\min_i |\lambda_i|}$$

  Wobei λ_i die Eigenwerte von $\mathbf{A}$ sind.

- Für orthogonale Matrizen, also wo gilt $\mathbf{QQ}^T = \mathbf{I}$ gilt

$$\kappa(\mathbf{Q}) = 1$$

Bei einem lin. Gleichungssystem $\mathbf{Ax} = \mathbf{b}$ muss damit gerechnet werden, dass die numerisch berechnete Lösung um $\kappa(\mathbf{A}) \cdot \varepsilon \cdot \mathbf{x}$ falsch ist.

Stabile und instabile Algorithmen

Ein Algorithmus heisst *stabil* wenn der Einfluss der Rundungsfehler beschränkt bleibt. Bei einem instabilen Algorithmus werden die Rundungsfehler aufgeschaukelt.
Ein *instabiler* Algorithmus liegt vor, wenn er die Kondition eines Problems verschlechtert. Ein Beispiel eines instabilen Algorithmus ist der Gausseliminationsschritt ohne Pivotstrategie.
Heute verwenden die meisten Implementationen die Kolonnenmaximumstrategie, doch man kann zeigen, dass diese nicht stabil ist.

Satz 1.1. *Bei Elimination mit orthogonalen Matrizen (z.B. mit Givensrotationen) verschlechtert sich die Kondition nicht.*

1.7. Schlussbemerkungen

Mit sehr grossen Registern "exakte" Zwischenergebnisse zu liefern kann in einigen Fällen sinnvoll sein, so z.B. die *Nachiteration*.

Nachiteration

Problem: $\mathbf{Ax} = \mathbf{b}$, wo $\mathbf{A}$ eine schlechte Kondition hat. Dann führt man ein Korrektur $\mathbf{\Delta x}$ ein.

$$\mathbf{A}(\widetilde{\mathbf{x}} + \mathbf{\Delta x}) = \mathbf{b}$$

$$\mathbf{A}\widetilde{\mathbf{x}} + \mathbf{A\Delta x} = \mathbf{b}$$

$$\mathbf{A\Delta x} = \mathbf{b} - \mathbf{A}\widetilde{\mathbf{x}} = \mathbf{r}$$

Dieses Gleichungssystem löst man für $\mathbf{\Delta x}$ und setzt

$$\mathbf{x_{neu}} = \widetilde{\mathbf{x}} + \mathbf{\Delta x}$$

2. Quadratur

2.1. Computer Algebra und Numerische Approximation

Riemann Summe

Gesucht ist die numerische Approximation eines bestimmten Integrals mit Grenzen $[a, b]$. Man wählt $a = x_0 < x_1 < \ldots < x_n = b$ und $y_i = f(x_i)$.

$$\int_a^b f(x)dx \approx \sum w_i f(\xi_i)$$

Wobei $\xi_i \in [x_{i-1}, x_i]$ und w_i eine Gewichtung ist.

Probleme

Ein Programm, das numerisch integriert funktioniert nicht immer. Dies, da die Algorithmen über endlichen Intervallen arbeiten und nicht alle Funktionswerte dazwischen berücksichtigen können; falls es dort Pole usw. gibt, versagen die Algorithmen.

2.2. Newton-Cotes

Idee

Interpoliere $f(\xi_i)$ durch Polynom P und integriere das Polynom.

$$\int_a^b f(x)dx \approx \int_a^b P(x)dx$$

Allgemeiner Fall

Wir konstruieren das Interpolationspolynom $P_n(x)$ für die Daten

x	ξ_0	$\dots$	ξ_n
$f(x)$	y_0	$\dots$	y_n

und durch Integration erhalten wir die Quadratur Regel:

$$\int_a^b P_n(x)dx = \sum_{i=0}^{n} w_i y_i$$

Wir benutzen die Lagrange Form von P_n:

$$P_n(x) = \sum_{i=0}^{n} l_i(x) \cdot y_i$$

$$l_i(x) = \prod_{j=0}^{n} \frac{x-\xi_j}{\xi_i - \xi_j}$$

$$\int_a^b P_n(x)dx = \sum_{i=0}^{n} y_i \underbrace{\int_a^b l_i(x)dx}_{w_i}$$

Newton-Cotes Regeln

Gegeben: Stützstellen ξ_i
Berechne Gewichte $w_i = \int_a^b l_i(x)dx$

$$\int_a^b f(x)dx \approx \sum_{i=0}^{n} w_i f(\xi_i)$$

```
closedcotes := n -> factor(int(interp(
  [seq(i*h, i= 0..n)],
  [seq(f(i*h), i=0..n)], z),z=0..n*h));
```

Trapezregel

$$\int_a^b f(x)dx \approx \frac{b-a}{2}(f(a)+f(b))$$

Als zusammengesetzte Regel für das Intervall $[a,b]$ mit n gleich grossen Intervallen (x_i, x_{i+1}) der Länge $h = x_{i+1} - x_i = (b-a)/n$, wobei $y_i = f(x_i)$, ergibt sich:

$$T(h) = h\left(\frac{1}{2}y_0 + y_1 + \ldots + y_{n-1} + \frac{1}{2}y_n\right)$$

Für den Fehler der zusammengesetzten Regel gilt

$$\int_a^b f(x)dx - T(h) = -\frac{b-a}{12}h^2 f''(\xi) \qquad \xi \in [a,b]$$

Zur geschickten Programmierung, so dass die y_i neu "dazwischen" eingefügt werden und die alten Werte nicht nochmals berechnet werden müssen:

$$T(h) = h(\underbrace{\frac{1}{2}y_0 + y_1 + y_2 + y_3 + \frac{1}{2}y_4}_{s_{alt}})$$

$$s_{neu} = s_{alt} + y_{1/2} + y_{3/2} + y_{5/2} + y_{7/2}$$

$$T(h/2) = \frac{h}{2} \cdot s_{neu}$$

```
function T = trapez(f,a,b,tol);
h = b - a; s = (feval(f,a) + feval(f,b))/2;
tnew = h * s; zh = 1; told = 2*tnew;
while abs(told - tnew) > tol * abs(tnew),
  told = tnew; zh = 2 * zh;
  h = h / 2;
  s = s + sum(feval(f,a + [1:2:zh]*h));
  tnew = h * s;
end;
T = tnew;
```

Simpsonregel

Ein Polynom zweiten Grades

$$\int_a^b f(x)dx \approx \frac{b-a}{6}\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right)$$

Für die zusammengesetze Simpsonregel müssen wir das Integrationsintervall $[a,b]$ in $2n$ Unterintervalle aufteilen und es ergibt sich:

$$S(h) = \frac{h}{3}(y_0 + 4y_1 + 2y_2 + 4y_3 + \ldots + 2y_{2n-2} + 4y_{2n-1} + y_{2n})$$

Der Integrationsfehler ist

$$\left|\int_a^b f(x)dx - S(h)\right| = \frac{b-a}{180}h^4\left|f^{(4)}(\xi)\right| \qquad \xi \in [a,b]$$

Zur Programmierung (Einfügen von neuen ξ_i "dazwischen"):

Für h:

$$\begin{aligned} s_1 &= y_0 + y_4 \\ s_2 &= y_2 \\ s_4 &= y_1 + y_3 \end{aligned}$$

Dann für $\frac{h}{2}$:

$$\begin{aligned} s_1^{neu} &= s_1 \\ s_2^{neu} &= s_2 + s_4 \\ s_4^{neu} &= \text{neue Fkt. Werte} \end{aligned}$$

Dann ergibt sich für Simpson:

$$S(h/2) = \frac{\frac{h}{2}}{3}(s_1^{neu} + 2 \cdot s_2^{neu} + 4 \cdot s_4^{neu})$$

```
function S = simpson(f,a,b,tol);
h = (b-a)/2; s1 = feval(f,a) + feval(f,b); s2 = 0;
s4 = feval(f,a + h); snew = h *(s1 + 4 * s4)/3;
zh = 2; sold = 2*snew;
while abs(sold-snew)>tol*abs(snew),
  sold = snew; zh = 2*zh; h=h/2;
  s2 = s2 + s4;
  s4 = sum(feval(f,a + [1:2:zh]*h));
  snew = h*(s1 + 2*s2 + 4*s4)/3;
end
S = snew;
```

Überblick

$$\int_a^b P_n(x)dx = \frac{b-a}{ns}\sum_{i=0}^{n} w_i f_i$$

n	w_i					ns	Fehler	Name
1	1	1				2	$h^3 \frac{1}{12} f^{(2)}(\xi)$	Trapezregel
2	1	4	1			6	$h^5 \frac{1}{90} f^{(4)}(\xi)$	Simpsonregel
4	7	32	12	32	7	90	$h^7 \frac{8}{945} f^{(6)}(\xi)$	Milne-Regel

2.3. Fehler der Newton-Cotes Formeln

Resultat von Steffenson

$$\int_a^b P(x)dx - \int_a^b f(x)dx = h^{p+1} \cdot k \cdot f^{(p)}(\xi), \quad \xi \in (a,b)$$

Dabei hängen p und k nicht von f ab. Die Gleichung gilt natürlich nur für genügend oft differenzierbare f.

2.4. Euler-Mac Laurin Summationsformel

Suchen einer geschlossenen Funktion für

$$\sum_{i=\alpha}^{\beta} f(i)$$

Idee: Wir suchen eine Summenfunktion $s(x)$ mit der Eigenschaft

$$\Delta s(x) = s(x+1) - s(x) = f(x)$$

Dann gilt:

$$\sum_{i=\alpha}^{\beta} f(i) = s(\beta+1) - s(\alpha)$$

Wollen $s(x)$ als formale (= kümmern uns nicht um Konvergenzbereich) Potenzreihe berechnen. Wir betrachten die Taylorreihe für $f(x+h)$ für $h = 1$:

$$f(x+1) = \sum_{k=0}^{\infty} f^{(k)}(x)\frac{1}{k!}$$

$$\Delta f(x) = f(x+1) - f(x) = \sum_{k=1}^{\infty} f^{(k)}(x)\frac{1}{k!}$$

Für Ableitungen gilt:

$$\Delta f^{(i)}(x) = \sum_{k=1}^{\infty} f^{(k+i)}(x)\frac{1}{k!}$$

Und für das Integral:

$$\Delta F(x) = F(x+1) - F(x) = \sum_{k=1}^{\infty} f^{(k-1)}\frac{1}{k!}$$

In Matrixschreibweise:

$$\mathbf{f} = \begin{pmatrix} f(x) \\ f'(x) \\ \vdots \\ f^{(m)}(x) \\ \vdots \end{pmatrix} \quad \mathbf{\Delta f} = \begin{pmatrix} \Delta F(x) \\ \Delta f(x) \\ \vdots \\ \Delta f^{(m-1)}(x) \\ \vdots \end{pmatrix}$$

$$\mathbf{A} = \begin{pmatrix} \frac{1}{1!} & \frac{1}{2!} & \frac{1}{3!} & \frac{1}{4!} & \cdots \\ & \frac{1}{1!} & \frac{1}{2!} & \frac{1}{3!} & \cdots \\ & & \ddots & \ddots & \ddots \end{pmatrix}$$

$$\mathbf{\Delta f} = \mathbf{A f}$$

Nach $\mathbf{f}$ aufgelöst:

$$\mathbf{f} = \mathbf{A}^{-1}\mathbf{\Delta f}$$

Dabei ist $\mathbf{A}$ folgende Matrix:

$$\mathbf{A}^{-1} = \begin{pmatrix} b_1 & b_2 & b_3 & \cdots \\ & b_1 & b_2 & \cdots \\ & & \ddots & \ddots \end{pmatrix}$$

Dabei sind die Koeffizienten b_i aus den *Bernoullizahlen* B_k zu berechnen durch:

$$b_k = \frac{B_{k-1}}{(k-1)!}$$

Die ersten Bernoullizahlen:

$$B_0 = 1, \quad B_1 = -\frac{1}{2}, \quad B_2 = \frac{1}{6}, \quad B_3 = 0$$

Durch Umformen ergibt sich die Euler-Mac Laurin Summenformel

$$\begin{aligned} \sum_{i=\alpha}^{\beta} f(i) &= \int_\alpha^\beta f(x)dx + \frac{f(\alpha)+f(\beta)}{2} \\ &\quad + \sum_{j=1}^{\infty} \frac{B_{2j}}{(2j)!}(f^{(2j-1)}(\beta) - f^{(2j-1)}(\alpha)) \end{aligned}$$

Folgerung

Für Summen von Polynomen $P_m(x)$ werden alle Ableitungen grösser m gleich 0 und somit ist es hinreichend die rechte Summe bei der Euler-Mac Laurin Formel nur bis zum Term $m/2$ zu bilden.

Spezialisierung für Trapezsumme

$$T(h) = \underbrace{\int_a^b g(t)dt}_{\text{exaktes Integral}} + \underbrace{\sum_{j=1}^{\infty} \frac{B_{2j}}{(2j)!}(g^{(2j-1)}(b) - g^{(2j-1)}(a))h^{2j}}_{\text{Fehler der Trapezregel}}$$

Wir haben die asymptotische Entwicklung der Trapezregel erhalten

$$T(h) = \int_a^b g(t)dt + c_1h^2 + c_2h^4 + \ldots + c_mh^{2m} + R_m$$

Wobei für das Restglied R_m gilt:

$$R_m = \frac{B_{2m+2}}{(2m+2)!}h^{2m+2}(b-a)^{(2m+2)}(\xi)$$

2.5. Romberg Integration

$$T(h) = \underbrace{\int_a^b g(t)dt}_{=c_0} + c_1h^2 + c_2h^4 + c_3h^6 + \ldots$$

Somit

$$T(h) \approx P_m(h^2)$$

Wir wollen $c_0 \approx P_m(0)$ berechnen.

Vorgehen:

- interpolieren durch Polynom vom Grade i
- evaluieren für $x = 0$ ergibt Näherung für Integral.

Für die Interpolation könnte ein Lagrange-Polynom verwendet werden, besser ist es aber das *Aitken-Neville Schema* zu benützen.

Aitken Neville Schema

x	y					
x_0	T_{00}					
x_1	T_{10}	T_{11}				
x_2	T_{20}	T_{21}	T_{22}			
$\ldots$	$\ldots$			$\ddots$		
x_i	T_{i0}	T_{i1}	T_{i2}	$\ldots$	T_{ii}	
$\ldots$	$\ldots$	$\ldots$	$\ldots$		$\ldots$	$\ddots$

Wobei gilt

$$\left.\begin{array}{lcl} T_{i0} & = & y_i \\ T_{ij} & = & \frac{(x_i - z)T_{i-1,j-1} + (z - x_{i-j})T_{i,j-1}}{x_i - x_{i-j}} \\ j & = & 1, 2, \ldots, i \end{array}\right\} i = 1, 2, 3, \ldots$$

z ist dabei eine feste Stelle, an dem die Polynome im ANS ausgewertet werden.

Spezialisierung des ANS für Romberg Integration

$$z = 0, \quad x_i = h_i^2, \quad h_i = \frac{h_{i-1}}{2} = \frac{h_0}{2^i}$$

Somit ergibt sich das Romberg Schema:

$$T_{ij} = \frac{4^{-j}T_{i-1,j-1} - T_{i,j-1}}{4^{-j} - 1}$$

Satz 2.1. *In der 2. Kolonne des Rombergschemas stehen die Simpson Werte. In der 3. Kolonne die Milne's Regel.*

Integrale über periodische Funktionen für eine Periode am Besten mit Trapez. Romberg bringt hier keine Verbesserung.

```
function [I, T] = romberg(f,a,b,tol);
h = b - a; intv =1;
s = (feval(f,a)  + feval(f,b)) / 2;
T(1,1) = s * h;
for i = 2:15
  intv = 2*intv; h = h/2;
  s = s + sum(feval(f,a+[1:2:intv]*h));
  T(i,1) = s * h;
  vhj = 1;
  for j = 2:i
    vhj = vhj/4;
    T(i,j) = (vhj*T(i-1,j-1) - T(i,j-1)) / (vhj -1);
  end;
  if abs(T(i,i)-T(i-1,i-1)) < tol * abs(T(i,i)),
    I = T(i,i); return
  end
end
warning(['limit of extrapolation steps reached. ', ...
         'Required tolerance may not be met.']);
I = T(i,i);
```

2.6. Gauss Quadratur

Wollen Stützstellen optimieren.

Transformation von "normalem" Integral zu $\int_{-1}^{1}$:

$$\int_a^b f(x)dx = \frac{b-a}{2}\int_{-1}^{1} f\left(\frac{b-a}{2}t + \frac{a+b}{2}\right)dt$$

Nun versucht man die Gewichte w_k und Stützstellen ξ_k optimal zu wählen:

$$\int_{-1}^{1} f(x)dx \approx \sum_{k=1}^{n} w_k f(\xi_k) \tag{1}$$

Satz 2.2. *Die Ordnung der Gauss Formel 1 ist* $\leq 2n - 1$

Die w_i und ξ_i sind symmetrisch. Ungerade Monome werden automatisch richtig integriert, Gleichungen nur für gerade Monome aufstellen.

Für $n \leq 6$ können wir die Lösung noch analytisch bestimmen.

```
gauss := proc(n)
 local w, xi, i, j, firsteq, eqns, sols, m;
 global res;

 firsteq := 2 * sum(w[i], i=1..m);
 if irem(n,2)=1 then firsteq := firsteq+w[0] fi;
```

```
  eqns := {2 = firsteq};
  for j from 2 by 2 to 2*(n-1) do
    eqns := eqns union{
    int(x^j, x=-1..1) = 2*sum(w[i]*xi[i]^j, i=1..m)};
  od;
  if irem(n,2)=1 then sols := {w[0]}
    else sols := {}
  fi;
  for j from 1 to m do
    sols := union {w[j], xi[j]};
  od;
  res := solve(eqns, sols);
  evalf(res);
end;
```

Charakterisieren der ξ_i, w_i

$$\int_{-1}^{1} P_{2n-1}(x)dx = \sum_{i=1}^{n} P_{2n-1}(\xi_i)w_i \tag{2}$$

Annahme: $P_{2n-1}(x) = H_{n-1}(x)\cdot Q_n(x) + R_{n-1}(x)$

Somit ergibt sich für die LHS von 2:

$$\int_{-1}^{1} P_{2n-1}(x)dx = \int_{-1}^{1} H_{n-1}(x)Q_n(x)dx + \int_{-1}^{1} R_{n-1}(x)dx$$

1. Wir wählen für Q_n das orthogonale Polynom zum Intervall $[-1,1]$ mit dem Skalarprodukt $(f,g) = \int_{-1}^{1} f(x)g(x)dx$
   Mit dieser Wahl ist $\int_{-1}^{1} H_{n-1}(x)Q_n(x)dx = 0$
2. Wählen ξ_i = Nullstellen des orthogonalen Polynoms $Q_n \Rightarrow \sum w_i H_{n-1}(\xi_i)Q_n(\xi_i) = 0$
3. Die ξ_i sind fest also kann man die Gewichte nach Idee von Newton-Cotes bestimmen
   $\Rightarrow w_i = \int_{-1}^{1} l_i(x)dx$ (l_i= Lagrange Polynome)
   $\Rightarrow \int_{-1}^{1} R_{n-1}(x)dx = \sum_{i=1}^{n} w_i R_{n-1}(\xi_i)$

Satz 2.3. *Die Gauss-Quadraturregel 2 hat die Ordnung $2n-1$. Die ξ_i sind Nullstellen der orthogonalen Q_n und die w_i sind die Integrale der Lagrangepolynome.*

```
n := 20;
X := sort([fsolve(orthopoly[P](n,x)=0,x)]);
Q := int(interp(X, [seq(y[i], i=1..n)],z),z=-1..1);
```

Orthogonale Polynome

Polynome P_k (vom Grade k) sind orthogonal auf $[-1,1]$ wenn

$$(P_i, P_k) = \int_{-1}^{1} P_i(x)P_k(x)dx = \begin{cases} 0 & k \neq i \\ 1 & k = i \end{cases} \tag{3}$$

Satz 2.4. *Die orthogonalen Polynome $p_0(x), p_1(x), \ldots$ genügen der 3-gliedrigen Rekursionsformel*

$$x\cdot p_{k-1}(x) = \beta_{k-1}p_{k-2}(x) + \alpha_k p_{k-1}(x) + \beta_k p_k, \qquad k = 1,2,\ldots$$

Mit $\alpha_k = (xp_{k-1}, p_{k-1}), \beta_k = (xp_{k-1}, p_k)$ und mit der Initialisierung $\beta_0 := 0, p_{-1}(x) := 0$ und $p_0(x) := 1/\sqrt{2}$.

```
restart;
lanczos := proc(p, alpha, beta, n)
  local k, q, x; p := array(0..n);
  alpha := array (1..n); beta := array (1..n-1);
  p[0] := 1/sqrt(2);
  alpha[1] := int(x*p[0]*p[0], x=-1..1);
  q := (x - alpha[1]) * p[0];
  for k from 2 to n do
    beta[k-1] := sqrt(int(q*q, x=-1..1));
    p[k-1] := expand(q / beta[k-1]);
    alpha[k] := int(x*p[k-1]*p[k-1], x= -1..1);
    q := (x - alpha[k])*p[k-1] - beta[k-1]*p[k-2];
  od;
  RETURN (NULL);
end;
```

Bezüglich dem Skalarprodukt 3 ergeben sich für die α_k und β_k:

$$\alpha_k = 0, \qquad \beta_k = \frac{k}{\sqrt{4k^2-1}}$$

β_{k-1} lässt sich relativ bequem durch das Skalarprodukt $(\beta_k p_k, \beta_k p_k)$ berechnen, wobei $\beta_k p_k = (x-\alpha_k)p_{k-1} - \beta_{k-1}p_{k-2}$, dies berechnet man am Besten immer am Anfang der neuen Iteration für die vorhergehende, also eigentlich β_{k-1}.

Golub-Welsh Algorithmus

Die 3-gliedrige Rekursionsformel von 2.4 in Vektor-Matrixschreibweise

$$x\cdot \underbrace{\begin{pmatrix} p_0(x) \\ p_1(x) \\ \vdots \\ p_{n-1}(x) \end{pmatrix}}_{\mathbf{p}(x)} = \underbrace{\begin{bmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \alpha_{n-1} & \beta_{n-1} \\ & & & \beta_{n-1} & \alpha_n \end{bmatrix}}_{\mathbf{T_n}} \begin{pmatrix} p_0(x) \\ p_1(x) \\ \vdots \\ p_{n-1}(x) \end{pmatrix} + \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \beta_n p_n(x) \end{pmatrix}$$

$$x\cdot \mathbf{p}(x) = \mathbf{T_n}\cdot \mathbf{p}(x) + \mathbf{e_n}\cdot \beta_n \cdot p_n(x)$$

Die ξ_i sind die Nullstellen von $p_n(x)$, d.h.

$$\xi_i \mathbf{p}(\xi_i) = \mathbf{T_n}\cdot \mathbf{p}(\xi_i)$$

Die ξ_i sind also die Eigenwerte von $\mathbf{T_n}$, da $\mathbf{T_n}$ symmetrische Matrix ist, sind die EW sehr gut konditioniert und reell.

Da die Gaussquadraturregel sagt, dass Polynome bis Grad $2n-1$ korrekt integriert werden, im Speziellen auch die orthogonalen Polynome $p_i(x)$ für $i = 0,1,\ldots,n-1$ und alle Integrale über den Polynomen Null ergeben mit Ausnahme von $p_0(x)$, welches $\sqrt{2}$ ergibt, gilt für die Gewichte w_i:

$$\mathbf{Pw} = \sqrt{2}\cdot \mathbf{e_1}$$

Wobei $\mathbf{P}$ die Eigenvektormatrix von $\mathbf{T}$ ist, dabei ist zu beachten, dass die Vektoren so normiert sind, dass die ersten Komponenten gleich $\frac{1}{\sqrt{2}}$ sind, da $p_0(x) = \frac{1}{\sqrt{2}}$. Dies erreicht man am besten in dem man von der EW-Zerlegung $\mathbf{T_n} = \mathbf{QDQ}^T$

1. Jede Kolonne von $\mathbf{Q}$ durch das erste Element dividieren.
2. Multiplizieren jede Kolonne mit $1/\sqrt{2}$.

dies kann auch geschrieben werden als

$$\mathbf{P} = \mathbf{QV}, \qquad \mathbf{V} = \text{diag}\left(\frac{1}{\sqrt{2}q_{1,1}}, \dots \frac{1}{\sqrt{2}q_{1,n}}\right)$$

$\mathbf{Pw} = \sqrt{2}\mathbf{e_1}$ kann nun gelöst werden durch

$$\begin{aligned} \mathbf{QVw} &= \sqrt{2}\mathbf{e_1} \\ \mathbf{w} &= \sqrt{2}\mathbf{V}^{-1}\mathbf{Q}^T\mathbf{e_1} = 2\begin{pmatrix} q_{11}^2 \\ q_{12}^2 \\ \vdots \\ q_{1n}^2 \end{pmatrix} \end{aligned}$$

Dadurch ergibt sich der Gauss-Legendre Algorithmus

```
function [xi, w] = gausslegendre(n)
  beta = 0.5 ./ sqrt(1-(2*(1:n-1).^(-2)));
  [Q, D] = eig(diag(beta,1) + diag(beta,-1));
  [x, i] = sort(diag(D));
  w = 2 * Q(1,i).^2;
```

2.7. Adaptive Quadratur

Definition (adaptiv). Schrittweite h dem Verlauf der Funktion anpassen, dass der Fehler für die einzelnen Intervalle ungefähr gleich ist.

Divide & Conquer

Idee: Zwei Integralnäherungen für Intervall bestimmen, z.B. mit Simpson für h und $h/2$, falls diese nicht genügend nahe beieinander liegen wird das Intervall halbiert und der selbe Algorithmus rekursiv aufgerufen. Bei den Abbruchkriterien, muss man aber aufpassen.

```
function Q = int(f, a, b)
  i1 = ...
  i2 = ...
  is = ...

  if is + (i1-i2) == is | m <= a | b <= m
    Q = i1;
  else
    Q = int(f,a,(a+b)/2) + int(f,(a+b)/2,b);
  end
```

Wobei $m = (a+b)/2$ und `is` ist eine *grobe Schätzung* für $|\int_a^b f(x)dx|$, diese macht man meist mit der Monte-Carlo Schätzung. Dabei werden zufällige Funktionswerte $f(\xi_i)$ generiert und dann das Integral durch auswerten an diesen Stellen approximiert

$$(b-a)\frac{1}{m}\sum_{i=1}^{m} f(\xi_i) \approx \int_a^b f(x)dx$$

Man könnte z.B. is so bestimmen:

$$is = \frac{b-a}{8}(f(a) + f(m) + f(b) + \sum_{i=1}^{5} f(\xi_i))$$

Falls man das Integral nicht auf Maschinengenauigkeit eps integrieren möchte, sondern nur bis zu einer Toleranz tol, kann man folgende Abbruchbedingung benutzen:

$$\frac{tol}{eps}\cdot is + (i1 - i2) == \frac{tol}{eps}\cdot is$$

Um die Neu-Berechnung von Funktionswerten zu vermeiden, kann man f_a, f_m, f_b als Parameter der rekursiv aufgerufenen Funktion übergeben.

Als weitere Verbesserung kann man das Romberg-Schema auf die beiden mit der Simpsonmethode gerechneten $i1, i2$ anwenden

$$i1_{neu} = \frac{16\cdot i2 - i1}{15}$$

```
function Q = simadpt(f,a,b,tol,trace,varargin)
 global warn1, warn2

 if (nargin < 4), tol = []; end;
 if (nargin < 5), trace = []; end;
 if (isempty(tol)), tol = eps; end;
 if (isempty(trace)), trace = 0; end;
 if tol <= eps, tol = 10*eps; end
 warn1 = 0; warn2 = 0;
 x = [a (a+b)/2 b];
 y = feval(f, x, varargin{:});
 for p=1:length(y)
   if isinf(y(p)) | isnan(y(p)),
     y(p) = 0; warn1=1;
   end
 end
 fa = y(1); fm = y(2); fb = y(3);
 yy = feval(f, a + ...
   [.9501 .2311 .6068 .4860 .8913]*(b-a), vargin{:});
 for p=1:length(yy)
   if isinf(yy(p)) | isnan(yy(p)),
     yy(p) = 0; warn1=1;
   end
 end
 is = (b-a)/8*(sum(y)+sum(yy));
 if is==0, is = b-a; end;
 is = is * tol/eps;
 Q = simadaptstp(f,a,b,fa,fm,fb,is,trace,varargin{:});
 if warn1==1,
   warning(['Infinite or Not-a-Number function', ...
     'value encountered. Singularity likely.', ...
     'Required tolerance may not be met.']);
 end
 if warn2==1
   warning(['Interval contains no more machine', ...
     number. Singularity likely. Required', ...
     'tolerance may not be met.']);
 end

function Q = simadaptstp (f,a,b,fa,fm,fb,is, ...
  trace,varargin)

 global warn1 warn2
 m = (a+b)/2; h = (b-a)/4;
 x = [a + h, b - h];
 y = feval(f, x, varargin{:});
 for p=1:length(y)
   if isinf(y(p)) | isnan(y(p)),
```

```
    y(p) = 0; warn1 = 1;
  end
end
fml = y(1); fmr = y(2);
i1 = h/1.5 * (fa + 4*fm + fb);
i2 = h/3 * (fa + 4*(fml + fmr) + 2*fm + fb);
i1 = (16*i2 - i1)/15;
if (is + (i1-i2) == is) | (m <= a) | (b<=m),
  if ((m <= a) | (b <= m)), warn2=1; end;
  Q = i1;
  if (trace), disp([a b-a Q]), end;
else
  Q = simadaptstp(f,a,m,fa,fml,fm,is,trace, ...
      varargin{:}) + simadaptstp(f,m,b,fm, ...
      fmr,fb,is,trace,varargin{:});
end;
```

3. Gewöhnliche Differentialgleichungen

3.1. Notation, Definitionen

Eine DGL *erster* Ordnung hat die Gestalt

$$y' = f(x, y) \tag{4}$$

Wobei f gegeben und $y(x)$ gesucht wird. $y(x)$ ist eine Lösung von 4 wenn

$$y'(x) = f(x, y(x))$$

für alle x gilt. Lösung wird eindeutig, wenn zusätzlich eine Anfangsbedingung (=AB)

$$y(x_0) = y_0$$

vorgegeben wird.

Eine DGL 2. Ordnung hat die Gestalt

$$y'' = f(x, y, y')$$

Nun kann man daraus ein System von DGL's erster Ordnung bilden, im allg. lautet ein System von n DGL 1. Ordnung

$$\begin{aligned} y_1' &= f_1(x, y_1, \ldots, y_n) \\ &\vdots \\ y_n' &= f_n(x, y_1, \ldots, y_n) \end{aligned}$$

oder in Vektorschreibweise

$$\mathbf{y}' = \mathbf{f}(x, \mathbf{y}) \tag{5}$$

Die allg. Lösung von 5 wird i.a. n frei wählbare Integrationskonstanten enthalten. Durch die Anfangsbedingung $\mathbf{y(x_0)} = \mathbf{y_0}$ wird aus der Lösungsschar eine *spezielle Lösung* ausgewählt. Falls die Bedingungen für spez. Lösung alle an *einer* Stelle x_0 gegeben, so spricht man von einem *Anfangswertproblem*; falls die Bedingungen aber über *mehrere* Stellen verteilt sind, so spricht man von einem *Randwertproblem*.

3.2. Analytische Lösung, Existenz

Satz 3.1. *Sei $f(x, y)$ definiert und stetig auf dem Streifen $-\infty < a \le x \le b < \infty$, $-\infty < y < \infty$.*
f erfülle die Lipschitzbedingung

$$|f(x, y) - f(x, y^*)| \le L \cdot |y - y^*| \tag{6}$$

für alle $x \in [a, b]$ und y, y^*, L ist eine Konstante.
Sei ferner η gegeben, dann gibt es genau eine Funktion $y(x)$ mit

1. *y ist stetig differenzierbar für $x \in [a, b]$*
2. *$y'(x) = f(x, y(x))$ für alle $x \in [a, b]$*
3. *$y(a) = \eta$*

3.3. Zurückführen auf ein System 1. Ordnung

Alle num. Verfahren verlangen die Standardform $y' = f(x, y)$. Lineare DGL höherer Ordnung werden durch Einführung von neuen Variablen auf ein System 1. Ordnung zurückgeführt.

Vorgehen für System 1. Ordnung

1. DGL nach höchster Vorkommender Ableitung auflösen

   $$y^{(n)} = f(x, y, y', \ldots, y^{(n-1)})$$

2. Neue Variablen einführen

   $$\begin{aligned} z_1(x) &= y(x) \\ z_2(x) &= y'(x) \\ &\vdots \\ z_n(x) &= y^{(n-1)}(x) \end{aligned}$$

   (bis höchste Ableitung - 1)

3. Neue Variablen ableiten und System aufschreiben:

   $$\begin{aligned} z_1' &= y' &&= z_2 \\ z_2' &= y'' &&= z_3 \\ &\vdots \\ z_n' &= y^{(n)} &&= f(x, y, y', \ldots, y^{(n-1)}) \end{aligned}$$

   Da DGL linear ist Matrix-Vektorschreibweise möglich

   $$\mathbf{z}' = \mathbf{A}\mathbf{z} + \begin{pmatrix} 0 \\ \vdots \\ 0 \\ c \end{pmatrix}, \quad \mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & \ldots & 0 \\ & & \ddots & & \\ & & & \ddots & \\ & & & & 1 \\ a_1 & a_2 & a_3 & \ldots & a_n \end{pmatrix}$$

   a_i bezeichnen hier die Koeffizienten der Ableitungen in $f(x, y, y', \ldots, y^{n-1})$

3.4. Richtungsfeld, Verfahren von Heun und Euler

$y' = f(x, y)$ definiert ein Richtungsfeld. Lösung $y(x)$ ist *Feldlinie* des Richtungsfeldes. Die AB legt eine Feldlinie fest.

Methode von Euler

Ausgehend von AB einen (kleinen) Schritt in Richtung Tangente gehen.

$$y_1 = y_0 + h \cdot f(x_0, y_0)$$

wobei h die Integrationsschrittweite. Die Methode von Euler ist aber ziemlich unbrauchbar, da man sehr viele Schritte gehen muss.

Methode von Heun

Vorausschauen wie die Steigung bei x_1 aussieht, danach Eulerschritt mit *gemittelter* Steigung.

$$\begin{aligned} y^* &= y_0 + h \cdot f(x_0, y_0) \\ y_1 &= y_0 + \frac{h}{2}(f(x_0, y_0) + f(x_1, y^*)) \end{aligned}$$

3.5. Die Fehlerordnung eines Verfahrens

$$y' = f(x, y), \quad y(x_0) = y_0, \quad \text{exakte Lösung: } y(x)$$

Integrationsschrittweite: h

$$y_1 \approx y(x_0 + h) = y(x_1)$$

Definition. $d_1(h) = y_1 - y(x_1)$ heisst lokaler Diskretationsfehler.

Wegen $d_1(0) = 0$ (AB!) gilt

$$d_1(h) = c_1 \cdot h^{p+1} + c_2 \cdot h^{p+2} + \ldots \qquad p \geq 0$$

Definition. p heisst *Fehlerordnung* des Verfahrens.

Für den *globalen* Fehler (= Fehler bei $x = x_0 + nh$) gilt:

$$d_n(h) = y(x) - y_n = e_p(x) \cdot h^p + o(h^{p+1})$$

Bei einem Verfahren der Fehlerordnung p nimmt bei Halbierung der Schrittweite der Fehler 2^p ab.

3.6. Konstruktion von Einschrittverfahren vom Typus Runge-Kutta

Definition (s-stufiges, explizites Runge-Kutta Verfahren).

$$\begin{aligned} k_1 &= f(x_0, y_0) \\ k_2 &= f(x_0 + c_2 h, y_0 + h a_{21} k_1) \\ k_3 &= f(x_0 + c_3 h, y_0 + h(a_{31} k_1 + a_{32} k_2)) \\ &\vdots \\ k_s &= f(x_0 + c_s h, y_0 + h \sum_{i=1}^{s-1} a_{s,i} k_i) \end{aligned}$$

$$y_1 = y_0 + h(b_1 k_1 + b_2 k_2 + \ldots + b_s k_s)$$

Die Parameter c_i, a_{ij} und b_i werden üblicherweise in eine Δ-Matrix geschrieben

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

Die Parameter sollen so gewählt werden, dass das Verfahren eine *möglichst hohe Fehlerordnung* hat. *Vereinfachung*: üblicherweise wird verlangt, dass die DGL $y' = 1, y(0) = 0$ mit exakter Lösung $y(x) = x$ alle Zwischenwerte $y_0 + h \sum_{i=1}^{j-1} a_{ji} k_i$ und der Endwert y_i exakt sind. Es gelten dann folgende Gleichungen:

$$\begin{aligned} c_j &= \sum_{i=1}^{j-1} a_{ji}, \quad j = 2, \ldots, s \\ \sum_{i=1}^{s} b_i &= 1 \end{aligned}$$

Klassisches Runge-Kutta-Verfahren

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline 1 & \frac{1}{6} & \frac{2}{6} & \frac{2}{6} & \frac{1}{6} \end{array}$$

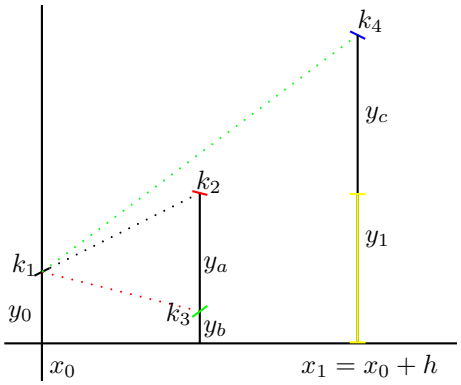


Abbildung 1: *Geometrische Interpretation des klassischen Runge-Kutta-Verfahrens*

$$\begin{aligned} k_1 &= f(x_0, y_0) & y_a &= y_0 + \tfrac{h}{2} k_1 \\ k_2 &= f(x_0 + \tfrac{h}{2}, y_a) & y_b &= y_0 + \tfrac{h}{2} k_2 \\ k_3 &= f(x_0 + \tfrac{h}{2}, y_b) & y_c &= y_0 + h k_3 \\ k_4 &= f(x_0 + h, y_c) \end{aligned}$$

$$y_1 = y_0 + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

Bemerkungen

Man kann zeigen, dass m Stufen $p^*(m)$ = maximale Fehlerordnung

$$\left.\begin{array}{rcl} m & = & 10+n \\ p^* & = & 8+n \end{array}\right\} \text{unmöglich für } n \geq 0$$

3.7. Eingebettete Runge-Kutta-Verfahren, automatische Schrittweitensteuerung

$$\begin{array}{c|cccc} 0 & & & & \\ c_2 & a_{21} & & & \\ \vdots & \vdots & & & \\ c_s & a_{s1} & \dots & a_{s,s-1} & \\ \hline & b_1 & \dots & b_{s-1} & b_s \\ \hline & \hat{b}_1 & \dots & \hat{b}_{s-1} & \hat{b}_s \end{array}$$

$$\begin{aligned} y_1 &= y_0 + h\sum_{i=1}^{s} b_i k_i \quad \text{Fehlerordnung } p \\ \hat{y}_1 &= y_0 + h\sum_{i=1}^{s} \hat{b}_i k_i \quad \text{Fehlerordnung } q \end{aligned}$$

meistens ist $q = p+1$

$$y_1 - \hat{y}_1 = \underbrace{h\sum_{i=1}^{s}(b_i - \hat{b}_i)}_{\text{Fehlerabschätzung}}\ k_i$$

Schrittweitensteuerung für $p/p+1$- Verfahren

$$\begin{aligned} y_k - y(x_k) &\sim c_1 h^{p+1} \\ \hat{y}_k - y(x_k) &\sim c_2 h^{p+2} \end{aligned}$$

Für kleines h gilt:

$$y_k - \hat{y}_k \sim c_1 h^{p+1} \quad \Rightarrow c_1 \approx \frac{y_k - \hat{y}_k}{h^{p+1}}$$

Analog gilt:

$$y_{k+1} - y(x_{k+1}) \sim c_3 h_{neu}^{p+1}$$

Wir wollen h_{neu} so wählen, dass

$$|y_{k+1} - y(x_{k+1})| < tol|\hat{y}(x_{k+1})|$$

wobei *tol* die gewünschte Genauigkeit ist.

Man trifft folgende Annahmen

1. $c_3 \approx c_1$
2. $y(x_{k+1}) \approx y(x_k) \approx y_k$

So ergibt sich

$$|c_3| h_{neu}^{p+1} = \frac{|y_k - \hat{y}_k|}{h^{p+1}} h_{neu}^{p+1} < tol|\hat{y_k}|$$

Durch Umformen erhält man

$$h_{neu} < h\ \sqrt[p+1]{\frac{tol|\hat{y}_k|}{|y_k - \hat{y}_k|}}$$

Durch Erfahrung wählt man den Faktor 0.8 statt $<$ und die Vektornorm anstatt $|\ |$

3.8. Bemerkungen

schlecht konditionierte Probleme

Es gibt Differentialgleichungen, welche schlecht konditioniert sind, ein Beispiel dafür ist z.Bsp.

$$y' = \lambda(y - F(x)) + F'(x)$$

Dabei bewirkt eine kleine Änderung der Anfangsdaten grosse Änderungen der Lösungen. Nach dem Prinzip von Wilkinson verhält sich die numerische Berechnung wie die exakte Rechnung mit leicht gestörten Anfangsdaten, somit ist es unmöglich mit numerischen Verfahren eine genaue Lösung zu erhalten.

Instabilität des Verfahrens

$$\mathbf{y}' = \mathbf{A}\mathbf{y}, \quad \mathbf{y}(0) = \mathbf{y}_0$$

Exakte Lösung

$$\mathbf{y}(x_k) = e^{\mathbf{A}\mathbf{h}}\mathbf{y}(x_{k-1})$$

Durch die Eigenwertzerlegung und Umformungen erhält man für die exakte Lösung:

$$\begin{aligned} \mathbf{z}(x_k) &= \mathbf{Q}^{-1}\mathbf{y}(x_k) \\ \mathbf{y}(x_k) &= \sum_{i=1}^{n} \mathbf{c}\mathbf{Q}_i e^{\lambda_i h} z_i(x_{k-1}) \end{aligned}$$

wobei $\mathbf{Q} = [\mathbf{c}\mathbf{Q}_1, \dots, \mathbf{c}\mathbf{Q}_n]$. Für RK ergibt sich aber

$$\begin{aligned} \mathbf{y}_k &= \sum_{i=1}^{n} \mathbf{c}\mathbf{Q}_i P_4(\lambda_i h) z_i^{(k)} \\ z_i^{(k)} &= P_4(\lambda_i h) z_i^{(k-1)} \end{aligned}$$

Falls $\Re(\lambda_i) < 0$ dann konvergiert $z_i(x_k)$ für $k \to \infty$ nach 0. Aber die RK-Lösung $z_i^{(k)} \to 0$ für $k \to \infty$ *nur falls*

$$|P_4(\lambda_i h)| < 1$$

Definition. Sei $y' = \lambda y, \quad y(0) = 1$. Sei $y_1 = F(h\lambda)y_0$ ein durch ein numerisches Verfahren berechneter Wert. Die Menge

$$B = \{\mu \in \mathbb{C} \text{ mit } |F(\mu)| < 1\}$$

heisst Gebiet der *absoluten Stabilität.*

Für R-K ist z.B. $F(\mu) = 1 + \frac{\mu}{1!} + \frac{\mu^2}{2!} + \frac{\mu^3}{3!} + \frac{\mu^4}{4!}$.

Damit sich die Komponenten der Lösung nicht aufschaukeln muss die Schrittweite h so gewählt werden, dass für alle λ_i von $\mathbf{A}$ gilt $\lambda_i h \in B$.

Definition. Ein DGLsystem $\mathbf{y}' = \mathbf{A}\mathbf{y} + \mathbf{b}(x)$ heisst steif (stiff equations) falls die EW von $\mathbf{A}$ sehr verschiedene negative Realteile haben.

3.9. Beispiel für die Benutzung `ode45`

$$\frac{dh}{dt} = \alpha(t) - \beta\sqrt{h}$$

Wobei

$$\alpha(t) = 10 + 4\sin(t) \qquad \beta = 2 \qquad h_0 = 1$$

`tankfill.m` erstellen:

```
function dhdt = tankfill(t,h)
% RHS function for tank-fill problem

A = 10 + 4*sin(t);   % alpha(t)
H = 2 * sqrt(h);     % beta*sqrt(h)

dhdt = A - H;
```

Die Lösung kann dann mit `ode45` berechnet werden.

```
>> tspan = [0 30]
>> h0 = 1;
>> [t,h] = ode45('tankfill',tspan,h0);
```

`tspan` ist dabei das Integrationsintervall.

4. Randwertprobleme

RWP der Form

$$pu'' + qu = f \qquad u(0) = 0,\ u'(\pi) = 0$$

4.1. Vorgehen bei analytischer Lösung

$$Lu = f$$

wo L ein linearer Operator der definiert ist für 2 stetig differenzierbare u welche die RB erfüllen. Man möchte Umkehrabbildung L^{-1} bestimmen, also

$$u = L^{-1}f$$

berechnen.
Entwicklung nach Eigenfunktionen. Man betrachtet das zugehörige EW Problem

$$Lu = \lambda u$$

Wenn wir die Eigenwerte λ_n und die zugehörigen Eigenfunktionen $u_n(x)$ (welche normiert sein müssen bzgl. irgendeiner Norm) kennen, dann können wir das RW Problem wie folgt lösen. Wir stellen $f(x)$ als Linearkombination der Eigenfunktionen dar.

$$f(x) = \sum_{n=1}^{\infty} a_n u_n(x)$$

sind die Koeffizienten a_n (=Fourierkoeffizienten) bekannt, dann gilt

$$u(x) = \sum_{n=1}^{\infty} \frac{a_n}{\lambda_n} u_n(x)$$

Bemerkung: Die kleinsten $|\lambda_i|$ sind die wichtigsten.

4.2. Numerische Lösung durch finite Differenzen

Wir betrachten ein etwas allg. Problem

$$-\frac{d}{dx}\left(p(x)\frac{du(x)}{dx}\right) + q(x)u(x) = f(x)$$

Wir diskretisieren und ersetzen die Ableitungen durch zentrale Differenzen:

$$h = \frac{\pi}{n} \qquad u_i \widetilde{=} u(i \cdot h)$$

$$u'(x) \approx \frac{u(x + \frac{h}{2}) - u(x - \frac{h}{2})}{h}$$

$$(p(x)u'(x))' \approx \frac{p(x + \frac{h}{2})\frac{u(x+h)-u(x)}{h} - p(x - \frac{h}{2})\frac{u(x)-u(x-h)}{h}}{h}$$

mittels $u_i = u(i \cdot h)$ erhalten wir für $x = x_i = i \cdot h$ die Gleichung

$$\begin{aligned} f(x_i) &= -\frac{1}{h^2}[p(x_i + \frac{h}{2})(u_{i+1} - u_i) - \\ &\quad p(x_i - \frac{h}{2})(u_i - u_{i-1})] + q(x_i)u_i \end{aligned}$$

Wir können diese Gl. für alle inneren Punkte aufschreiben, dies gibt uns $n-1$ Gleichungen (für u_0 nehmen wir die Randbedingung an). Für den rechten Rand haben wir zwei Varianten.

1. $u'(\pi) = \frac{u_n - u_{n-1}}{h} = 0$ d.h. $u_n = u_{n-1}$

2. $u'(\pi) = \frac{u_{n+1} - u_{n-1}}{2h} = 0$ wobei u_{n+1} eine zusätzliche Unbekannte, aber DGL für x_n aufstellbar.

Die zweite Methode ist in jedem Fall vorzuziehen, da der Diskretitionsfehler in der Ordnung h^2 ist, während er bei der ersten Methode in h ist.

Es folgt: Darauf achten, dass überall mit der gleichen Ordnung diskretisiert wird.

Die Gleichung des Systems beinhaltet ein lineares Gleichungssystem mit einer schwach besetzten Matrix, das Lösen dieses Gl.system ergibt eine Tabelle der unbekannten Funktion.

A. Trigonometrie

A.1. Funktionswerte für einige Winkel

α	0	$\frac{\pi}{6}$	$\frac{\pi}{4}$	$\frac{\pi}{3}$	$\frac{\pi}{2}$
$\sin\alpha$	0	$\frac{1}{2}$	$\frac{\sqrt{2}}{2}$	$\frac{\sqrt{3}}{2}$	1
$\cos\alpha$	1	$\frac{\sqrt{3}}{2}$	$\frac{\sqrt{2}}{2}$	$\frac{1}{2}$	0
$\tan\alpha$	0	$\frac{\sqrt{3}}{3}$	1	$\sqrt{3}$	–

A.2. Trigonometrische Kurven

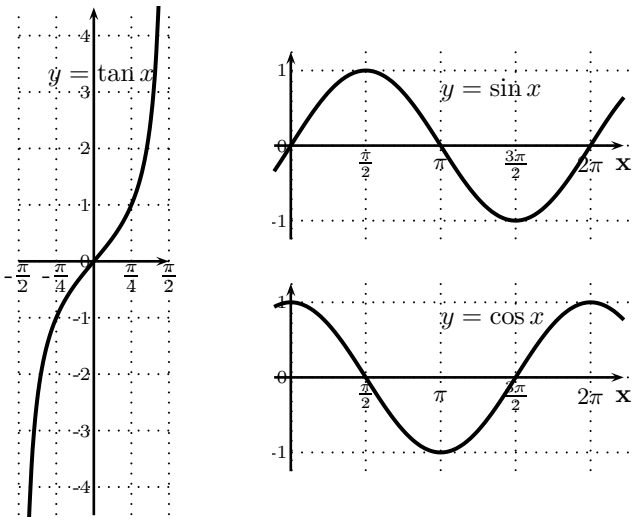


Abbildung 2: *Trigonometrische Kurven*

B. Tschebyscheff-Polynom

$$\xi_k = \cos\left(\frac{\pi(k-1/2)}{n}\right) \qquad \text{für } k = 1,\ldots,n$$

C. Umwandlung von uneigentlichen Integralen

Gute Ansätze:

$$t = \frac{1}{x} \quad t = \frac{1}{x+1} \quad t = e^{-x}$$

Beispiel

$$t = \frac{1}{x+1}, x = \frac{1}{t} - 1 \Rightarrow dx = -\frac{1}{t^2}dt$$

D. Differentialrechung

D.1. Wichtige Ableitungen

$f(x) = c$	$f'(x) = 0$
$f(x) = cx$	$f'(x) = c$
$f(x) = x^n$	$f'(x) = nx^{n-1}$
$f(x) = \sqrt{x}$	$f'(x) = \frac{1}{2\sqrt{x}}$
$f(x) = e^{cx}$	$f'(x) = ce^{cx}$
$f(x) = \ln\lvert x\rvert$	$f'(x) = \frac{1}{x}$
$f(x) = \log_a\lvert x\rvert$	$f'(x) = (\log_a e)\frac{1}{x} = \frac{1}{x\ln x}$
$f(x) = a^x$	$f'(x) = a^x \cdot \ln(a)$
$f(x) = a^{cx}$	$f'(x) = a^{cx} \cdot (c\ln a)$
$f(x) = x^x$	$f'(x) = (1+\ln x)x^x$
$f(x) = \sin(x)$	$f'(x) = \cos(x)$
$f(x) = \cos(x)$	$f'(x) = -\sin(x)$
$f(x) = \tan x$	$f'(x) = \frac{1}{\cos^2(x)} = 1 + \tan^2(x)$
$f(x) = \cot x$	$f'(x) = -\frac{1}{\sin^2(x)} = -(1+\cot^2(x))$
$f(x) = \arcsin(x)$	$f'(x) = \frac{1}{\sqrt{1-x^2}}$
$f(x) = \arccos(x)$	$f'(x) = -\frac{1}{\sqrt{1-x^2}}$
$f(x) = \arctan(x)$	$f'(x) = \frac{1}{1+x^2}$
$f(x) = arccot(x)$	$f'(x) = -\frac{1}{1+x^2}$
$f(x) = \sinh(x)$	$f'(x) = \cosh(x)$
$f(x) = \cosh(x)$	$f'(x) = \sinh(x)$
$f(x) = \tanh(x)$	$f'(x) = \frac{1}{\cosh^2(x)} = 1 - \tanh^2(x)$
$f(x) = coth(x)$	$f'(x) = -\frac{1}{\sinh^2(x)} = 1 - coth^2(x)$
$f(x) = arcsinh(x)$	$f'(x) = \frac{1}{\sqrt{x^2+1}}$
$f(x) = arccosh(x)$	$f'(x) = \frac{1}{\sqrt{x^2-1}}$
$f(x) = artanh(x)$	$f'(x) = \frac{1}{\sqrt{1-x^2}}$
$f(x) = arcoth(x)$	$f'(x) = \frac{1}{\sqrt{1-x^2}}$

D.2. Taylor Approximationspolynom

Sei $f : [a,b] \to \mathbb{R}$ genügend oft differenzierbar. Das n-te Taylorpolynom von f um x_0:

$$j^n_{x_0} f(x) = \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k$$

D.3. Repetition: Taylorentwicklung

$$\begin{aligned} f(x+a, y+b) \quad &\approx \quad f(x,y) + f_x a + f_y b + f_{xx}\frac{a^2}{2} + f_{xy}ab \\ &\quad + f_{yy}\frac{b^2}{2} + o(h^3) \end{aligned}$$

$$\begin{aligned} y' &= f(x, y(x)) \\ y'' &= f_x + f_y y' \\ y''' &= \ldots \end{aligned}$$

E. Integralrechnung

E.1. Wichtige Integrale

$\int x^n dx$	=	$\frac{x^{n+1}}{n+1} + C \;\; n \neq -1$
$\int \frac{1}{x} dx$	=	$\ln \lvert x \rvert + C$
$\int e^x dx$	=	$e^x + C$
$\int \ln \lvert x \rvert dx$	=	$x \ln \lvert x \rvert - x + C$
$\int \frac{f'(x)}{f(x)} dx$	=	$\ln \lvert f(x) \rvert$
$\int \sin x dx$	=	$-\cos x + C$
$\int \cos x dx$	=	$\sin x + C$
$\int \frac{1}{\cos^2 x} dx$	=	$\tan x + C$
$\int \tan x dx$	=	$-\ln \lvert \cos x \rvert + C$
$\int \frac{1}{\sqrt{1-x^2}} dx$	=	$\arcsin x + C$
$\int \frac{-1}{\sqrt{1-x^2}} dx$	=	$\arccos x + C$
$\int \frac{1}{1+x^2} dx$	=	$\arctan x + C$
$\int \sinh x dx$	=	$\cosh x + C$
$\int \cosh x dx$	=	$\sinh x + C$
$\int \frac{1}{\cosh^2 x} dx$	=	$\tanh x + C$
$\int \tanh x dx$	=	$\ln \cosh x + C$
$\int \frac{1}{\sqrt{1+x^2}} dx$	=	$arsinhx + C$
$\int \frac{1}{\sqrt{x^2-1}} dx$	=	$arcoshx + C$
$\int \frac{1}{1-x^2} dx$	=	$artanhx + C$
$\int \sin^2 x dx$	=	$\frac{1}{2}(x - \sin x \cos x) + C$
$\int \cos^2 x dx$	=	$\frac{1}{2}(x + \sin x \cos x) + C$
$\int \tan^2 x dx$	=	$\tan x - x + C$
$\int \cot^2 x dx$	=	$-\cot x - x + C$
$\int (ax+b)^n dx$	=	$\frac{(ax+b)^{n+1}}{a(n+1)}$
$\int \frac{1}{ax+b} dx$	=	$\frac{1}{a} \ln \lvert ax+b \rvert + C$
$\int (ax^p+b)^s x^{p-1} dx$	=	$\frac{(ax^p+b)^{s+1}}{ap(s+1)} + C$
$\int (ax^p+b)^{-1} x^{p-1} dx$	=	$\frac{1}{ap} \ln \lvert ax^p+b \rvert + C$
$\int \log_a \lvert x \rvert dx$	=	$x(\log_a \lvert x \rvert - \log_a e) + C$
$\int x^{-1} \ln x dx$	=	$\frac{1}{2}(\ln^x)^2 + C$
$\int \cot x dx$	=	$\ln \lvert \sin x \rvert + C$
$\int \sin(ax+b) dx$	=	$-\frac{1}{a} \cos(ax+b) + C$
$\int \cos(ax+b) dx$	=	$\frac{1}{a} \sin(ax+b) + C$
$\int \frac{1}{\sin x} dx$	=	$\ln \lvert \tan \frac{x}{2} \rvert + C$
$\int \frac{1}{\cos x} dx$	=	$\ln \lvert \tan(\frac{x}{2} + 4\pi) \rvert + C$
$\int \sin^n x dx$	=	$-\frac{1}{n} \sin^{n-1} x \cos x + \frac{n-1}{n} \int \sin^{n-2} + C$
$\int \cos^n x dx$	=	$\frac{1}{n} \sin x \cos^{n-1} x + \frac{n-1}{n} \int \cos^{n-2} + C$

E.2. Partielle Integration

$$\int u(t)v'(t)\,dt = u(t)v(t) - \int u'(t)v(t)\,dt$$

E.3. Substitutionsmethode

$F(g(x)) + c = \int f(g(x)) \cdot g'(x) dx$
Beispiel:
$\int \tan(x) dx = \int \frac{\sin(x)}{\cos(x)} dx$
$t = \cos(x)$
$dt = -\sin(x) dx \rightarrow dx = -\frac{dt}{\sin x}$
$\Rightarrow -\int \frac{1}{t} dt = -ln|\cos(x)| + c$

Oftmals auch umgekehrt:
$\int \sqrt{1-t^2} dt$
$t = \sin(x)$
$dt = \cos(x) dx$
$\Rightarrow \int \cos^2(x) dx$

F. Differentialgleichungen analytisch

F.1. Differentialgleichungen I

homogene DGL mit konstanten Koeffizienten

Gesucht $y(x)$ so dass

$$y^{(n)} + a_{n-1} y^{(n-1)} + \ldots + a_1 y' + a_0 y = 0 \qquad (7)$$

Satz

- Sind y_1, y_2 Lösungen von (7), so auch $y_1 + y_2$ und $\alpha \cdot y_1$ $(\alpha \in \mathbb{R})$
- Es gibt genau n linear unabhänigige Lösungen $y_0, \ldots, y_{n-1}$ und die allg. Lösung ist $y = c_0 y_0 + c_1 y_1 + \ldots + c_{n-1} y_{n-1}$

Ansatz für (7): $y(x) = e^{\lambda \cdot x}$
$y'(x) = \lambda e^{\lambda x}, \ldots, y^{(n)}(x) = \lambda^n e^{\lambda x}$
(7)$\rightarrow \lambda^n e^{\lambda x} + \ldots + a_1 \lambda e^{\lambda x} + a_0 e^{\lambda x} = 0$
$\rightarrow e^{\lambda x} \underbrace{[\lambda^n + \ldots + a_1 \lambda + a_0]}_{} = 0$
$chp(\lambda)$: charakteristisches Polynom

Falls λ_i eine Nullstelle von $chp(\lambda)$ ist und gilt:

- λ_i *ist einfache Nullstelle*
  so ist $e^{\lambda_i x}$ eine linear unabhängige Lösung von (7).
- λ_i *ist m-fache Nullstelle*
  so sind die Funktionen $e^{\lambda_i t}, t \cdot e^{\lambda_i t}, t^2 \cdot e^{\lambda_i t}, \ldots, t^{m-1} \cdot e^{\lambda_i t}$ linear unabhängige Lösungen von (7).
- λ_i *ist komplexe Nullstelle*
  wenn also $\lambda_i = \mu_i + i\nu_i, \nu_i \neq 0$ bzw. die Lösung $z(x) = e^{\lambda_i x}$ so bilden $Re(z)$ und $Im(z)$ zwei reelle unabhängige Lösungen von (7).
  $Re(z) = e^{\mu_i x} \cos(\nu_i x)$
  $Im(z) = e^{\mu_i x} \sin(\nu_i x)$

Inhomogene DGL mit konstanten Koeffizienten

$$y^{(n)} + a_{n-1} y^{(n-1)} + \ldots + a_1 y' + a_0 y = K(t) \qquad (8)$$

Satz Sei $y_{part}(t)$ eine spezielle Lösung von (8) und sei $y(t) = c_1 y_1 + \ldots + c_n y_n$ die allg. Lösung der homogenen Gleichung. Dann ist

$$y = y_{part} + c_1 y_1 + \ldots + c_n y_n$$

die allg. Lösung der inhomogenen Gleichung (8).

Suche von partikulären Lösungen

Wenn $K(t) := t^r e^{\lambda t}$ bzw. allgemeiner $K(t) := q(t) e^{\lambda_0 t}$ mit einem Polynom $q(t)$ vom Grad r und λ_0 m-facher Nullstelle von $chp(\lambda)$, so erhält man die partikuläre Lösung durch den Ansatz:

$$y_{part}(x) := (A_0 + A_1 t + \ldots + A_r t^r) t^m e^{\lambda_0 t}$$

mit unbestimmten Koeffizienten A_k. Danach alle in (8) vorkommenden Ableitungen ausrechnen und in (8) einsetzen

und A_k durch Koeffizientenvergleich bestimmen.

mögliche Ansätze

$K(t)$	Bedingung	Ansatz für $y_{part}(t)$
t^r	$0 \notin L$	$A_0 + \ldots + A_r t^r$
t^r	$0 \in L$, m-fach	$A_0 t^m + \ldots + A_r t^{m+r}$
$b_0 + \ldots + b_r t^r$	$0 \notin L$	$A_0 + A_1 t + \ldots + A_r t^r$
$e^{\lambda_0 t}$	$\lambda \notin L$	$Ae^{\lambda_0 t}$
$e^{\lambda_0 t}$	$\lambda \in L$, m-fach	$at^m e^{\lambda_0 t}$
$\cos(\omega t), \sin(\omega t)$	$\pm i\omega \notin L$	$A\cos(\omega t) + B\sin(\omega t)$
$\cos(\omega t), \sin(\omega t)$	$\pm i\omega \in L$,1-fach	$t(A\cos(\omega t) + B\sin(\omega t))$
$t^2 e^{-t}$	$-1 \notin L$	$(A_0 + A_1 t + A_2 t^2)e^{-t}$

Beispiel Bestimme allgem. Lösung der DGL

$$y''(x) + 3y'(x) + 2y(x) = \cos x$$

1. Homogene Lösung: $y(x) = c_1 e^{-x} + c_2 e^{-2x}$
2. Inhomogene Lösung:
   Ansatz (nach Tabelle):

$$\begin{aligned} y &= a\cos x + b\sin x \\ y' &= -a\sin x + b\cos x \\ y'' &= -a\cos x - b\sin x \end{aligned}$$

   $y'' + 3y' + 2y = (a+3b)\cos x + (b-3a)\sin x = \cos x$
   $\Rightarrow a = \frac{1}{10}, b = \frac{3}{10}$
   Allgem. Lös der DGL:
   $y(x) = c_1 e^{-x} + c_2 e^{-2x} + \frac{1}{10}\cos x + \frac{3}{10}\sin x$

Eulersche Differentialgleichungen

Gesucht $y = y(r)$ so dass

$$y^{(n)} + \frac{b_{n-1}}{r} y^{(n-1)} + \ldots + \frac{b_0}{r^n} = 0 \qquad (9)$$

Koeffizienten sind hier also nicht konstant. Auch hier gibt es einen Ansatz:

$$y(r) = r^\alpha$$
$$y'(r) = \alpha r^{\alpha-1},\ y^{(k)} = \alpha(\alpha-1)\cdot\ldots\cdot(\alpha-k+1)r^{\alpha-k}$$

Das Indexpolynom lautet dann:

$$inp(\alpha) := \alpha\ldots(\alpha-n+1) + \ldots + \alpha b_1 + b_0 = 0$$

Falls α_i eine Nullstelle von $inp(\alpha)$ ist und gilt:

- *α_i ist einfache Nullstelle*
  so ist r^{α_i} eine linear unabhängige Lösung von (9).
- *α_i ist m-fache reelle Nullstelle*
  so sind die Funktionen $r^{\alpha_i}, r^{\alpha_i}\cdot\ln r, \ldots, r^{\alpha_i}\cdot(\ln r)^{m-1}$ linear unabhängige Lösungen von (9).
- *α_i ist einfach komplex konjugierte Nullstelle*
  wenn also $\alpha_i = \mu_i + i\nu_i, \nu_i \neq 0$ so ersetze $r^{\alpha_i}, r^{\overline{\alpha_i}}$ durch $r^{\mu_i}\cos(\nu_i \ln r)$ und $r^{\mu_i}\sin(\nu_i \ln r)$ welche zwei reelle unabhängige Lösungen von (9) sind.
- *mehrfache komplex konjugierte Nullstelle*
  so sind die rellen Lösugen der komplex konjugierten Nullstelle multipliziert mit $(\ln r)^k$ ebenfalls unabhängige Lösugen von (9).

Falls das Indexpolynom m-fache NS hat, dann gibt es folgende Lösungen:
$r^{\alpha_1}, (\ln r)\cdot r^{\alpha_1}, \ldots, (\ln r)^{m-1} r^{\alpha_1}$

Falls α komplex (r^{a+ib}):
Ersetze $r^\alpha, r^{\overline{\alpha}}$ durch $r^a\cos(b\ln r)$ und $r^a\sin(b\ln r)$

inhomogene DGL mit variablen Koeffizienten

$$y'' + p_0(t)y' + p_1(t)y = q(t) \qquad (10)$$

wobei $q(t)$ beliebig ist.

Annahme: die allgemeine Lösung der homogenen Gleichung sei bekannt, z.B. p_0, p_1 konst. oder Eulersche DGL.
Ziel: daraus eine spezielle Lösung der inhomogenen Gleichung finden.
Ansatz:

$$y_0(t) = c_1(t)y_1(t) + c_2(t)y_2(t)$$

mit y_1, y_2 lin. unabh. Lösungen der homogenen Gleichung von (10)

$$y_0 = c_1 y_1 + c_2 y_2$$
$$y_0' = c_1 y_1' + c_2 y_2' + \underbrace{c_1' y_1 + c_2' y2}_{\text{1. Bed} = 0}$$
$$y_0'' = c_1 y_1'' + c_2 y_2'' + c_1' y_1' + c_2' y_2'$$

Diese Gleichung kann man wieder in die DGL einsetzen. Durch ausklammern von c_1, c_2 erhält man dann die 2.Bedingung.

$$c_1' y_1' + c_2' y_2' = q(t) \quad \text{2. Bedingung}$$

Mit diesen beiden Bedingungnen hat man ein Gleichungssystem aus dem man die unbekannten Funktionen bestimmen kann.

F.2. Anwendungen der $\int$-Rechnung auf Diff'Gl

Variation der Konstante

$$y' = p(x)\cdot y + q(x) \qquad (11)$$

1. homogene Lösung von (11) finden
   $y_{hom}(x) = Ce^{P(x)}$ wobei $P(x)$ die Stammfunktion von $p(x)$
2. inhomogene Lösung von (11) finden
   Dafür benutzt man den Ansatz:

$$y_{inhom}(x) = c(x)\cdot y_{hom}(x)$$

   Man setzt also die homogene Lösung als "Konstante" einer unbekannten Funktion.
   So erhält man schliesslich nach einsetzen in (11)

$$c'(x)\cdot y_{hom}(x) = q(x)$$

3. Die Allgemeine Lösung lautet dann

$$y(x) = y_{hom} + y_{inhom}$$

Separtion (Trennung der Variablen)

Eine DGL 1. Ordnung der Form $\frac{dy}{dx} = g(x)\cdot k(y)$ lässt sich folgendermassen lösen:

1. Trennung der beiden Variablen

$$\frac{dy}{dx} = g(x)\cdot k(y) \quad \Rightarrow \quad \frac{dy}{k(y)} = dx\cdot g(x)$$

2. Integration auf beiden Seiten der Gleichung

$$\int \frac{dy}{k(y)} = \int dx\cdot g(x)$$

3. nach y auflösen

Substitution

DGL 1.Ordnung können teilweise durch geschickte Substitution auf seperierbare zurückgeführt, danach kann man die neu erhaltene DGL für u lösen und rücksubstituieren:

- $y' = \Phi(\frac{y}{x})$
  $u = \frac{y}{x} \quad \Rightarrow \quad u'(x) = \frac{\Phi(u)-u}{x}$
- $y' = \Psi(x+y+c)$
  $u = x+y \quad \Rightarrow \quad y = u-x \Rightarrow y' = u'-1 = \Psi(u)$
  Danach löse $\frac{du}{dx} = u' = \Psi(u)+1 \Rightarrow \frac{du}{\Psi(u)+1} = dx$

Beispiel Löse $\frac{dy}{dx} = \frac{2x^3+y^3}{3xy^2}$
Setze $u := \frac{y}{x}$. $\Rightarrow y(x) = u(x)\cdot x \Rightarrow y' = u'x+u$

$$\begin{aligned} y' &= \frac{2}{3}\frac{x^2}{y^2} + \frac{1}{3}\frac{y}{x} \\ u'x+u &= \frac{2}{3}v^{-} + \frac{1}{3}\frac{y}{x} \\ u'x &= \frac{2}{3}(v^{-2}-v) \end{aligned}$$

So ist $v = 1$ und damit $y = x$ eine konstante Lösung und für nicht konstantes v rechnet man mit Seperation der Variablen weiter:

$$\frac{3v'}{v^{-2}-v} = 2\frac{1}{x}$$
$$\ldots$$

G. Beispiele

G.1. Differentialgleichungssystem

$$\begin{aligned} y_1' &= y_1 - y_2 \\ y_2' &= y_1 + y_2 \end{aligned}$$

Anfangsbedingungen: $y_1(0) = 1,\ y_2(0) = 0$. Ein Integrationsschritt mit Heun mit $h = 0.1$, berechne $y_1(0.1)$ und $y_2(0.1)$.

Es ergibt sich also in Matrixschreibweise

$$\mathbf{y}' = \underbrace{\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}}_{\mathbf{A}} \mathbf{y}$$

Heunschritt:

$$\begin{aligned} \mathbf{y}^* &= \mathbf{y_0} + h\cdot \mathbf{A y_0} \\ \mathbf{y}^* &= \begin{pmatrix} 1 \\ 0 \end{pmatrix} + 0.1\cdot \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \begin{pmatrix} 0.1 \\ 0.1 \end{pmatrix} \\ &= \begin{pmatrix} 1.1 \\ 0.1 \end{pmatrix} \end{aligned}$$

$$\begin{aligned} \mathbf{y} &= \mathbf{y_0} + \frac{h}{2}(\mathbf{A y_0} + \mathbf{A y}^*) \\ \mathbf{y} &= \begin{pmatrix} 1 \\ 0 \end{pmatrix} + 0.05\cdot \left(\left(\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} \mathbf{y}^* \right) \right) \\ &= \begin{pmatrix} 1 \\ 0 \end{pmatrix} + 0.05\cdot \left(\left(\begin{pmatrix} 1 \\ 1 \end{pmatrix} + \begin{pmatrix} 1 \\ 1.2 \end{pmatrix} \right) \right) \\ &= \begin{pmatrix} 1 \\ 0 \end{pmatrix} + 0.05\cdot \begin{pmatrix} 2 \\ 2.2 \end{pmatrix} \\ &= \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \begin{pmatrix} 0.1 \\ 0.11 \end{pmatrix} \\ &= \begin{pmatrix} 1.1 \\ 0.11 \end{pmatrix} \end{aligned}$$

G.2. Differentialgleichungssystem

Vektorwertige Funktion $\mathbf{z}(t)$ bestehend aus den beiden unbekannten Funktionen der Zeit $\mathbf{z} = [x(t), y(t)]^T$ (g, F seien gegeben):

$$\frac{d^2\mathbf{z}}{dt^2} = -g\frac{\mathbf{z}}{\| \mathbf{z} \|^3} - F\frac{\mathbf{z}-\mathbf{u}}{\| \mathbf{z}-\mathbf{u} \|}$$

Wir wollen ein System erster Ordnung, d.h der Form $\frac{dw}{dt} = f(t,w)$.

$$\begin{aligned} \mathbf{w_1}(t) &= \mathbf{z}(t) \\ \mathbf{w_2}(t) &= \mathbf{z}'(t) \end{aligned}$$

$$\begin{aligned} \mathbf{w_1}' &= \mathbf{z}' &&= \mathbf{w_2} \\ \mathbf{w_2}' &= \mathbf{z}'' &&= -g\frac{\mathbf{z}}{\|\mathbf{z}\|^3} - F\frac{\mathbf{z}-\mathbf{u}}{\|\mathbf{z}-\mathbf{u}\|} \\ & &&= -g\frac{\mathbf{w_1}}{\|\mathbf{w_1}\|^3} - F\frac{\mathbf{w_1}-\mathbf{u}}{\|\mathbf{w_1}-\mathbf{u}\|} \end{aligned}$$

H. Bestimmen von Maschinengrössen

```
% epsilon
e = 1.0
while ((1.0+e) > 1.0)
  e = e/2.0;
end
e = e*2.0;
```

```
% kleinste pos. norm.
a = 1.0;
while ((a+a*e) > a)
  a = a/2.0;
end
a = a*2.0;

% kleinste pos. nicht-norm.
b = 1.0;
while ((b/2.0) > 0.0)
  b = b/2.0;
end

 % gr"osste
c = 2-e;
while ((c*2.0) ~= Inf)
  c = c*2.0;
end
```

I. Additionstheoreme

$\sin^2\varphi + \cos^2\varphi = 1$
$\cos\varphi = \sin(\varphi + \pi/2)$
$\sin(-\varphi) = -\sin\varphi$
$\cos(-\varphi) = \cos\varphi$
$\sin(\alpha \pm \beta) = \sin\alpha\cos\beta \pm \cos\alpha\sin\beta$
$\cos(\alpha \pm \beta) = \cos\alpha\cos\beta \mp \sin\alpha\sin\beta$
$\sin(2\varphi) = 2\sin\varphi\cos\varphi$
$\cos(2\varphi) = \cos^2\varphi - \sin^2\varphi = 2\cos^2\varphi - 1 = 1 - 2\sin^2\varphi$
$\sin(3\varphi) = 3\sin\varphi - 4\sin^3\varphi$
$\cos(3\varphi) = 4\cos^3\varphi - 3\cos\varphi$
$\sin^2(\frac{\varphi}{2}) = \frac{1-\cos\varphi}{2}$
$\cos^2(\frac{\varphi}{2}) = \frac{1+\cos\varphi}{2}$
$\tan(\alpha \pm \beta) = \frac{\tan\alpha \pm \tan\beta}{1 \mp \tan\alpha\tan\beta}$
$\tan(2\varphi) = \frac{2\tan\varphi}{1-\tan^2\varphi}$
$\tan(3\varphi) = \frac{3\tan\varphi - \tan^3\varphi}{1-3\tan^2\varphi}$
$\tan^2(\frac{\varphi}{2}) = \frac{1-\cos\varphi}{1+\cos\varphi}$
$\tan\frac{\varphi}{2} = \frac{1-\cos\varphi}{\sin\varphi} = \frac{\sin\varphi}{1+\cos\varphi}$