

Numerik

Fabian Starc

Version: July 29, 2025

Fehler

absoluter Fehler:

$$|\Delta a| = |\tilde{a} - a|$$

relativer Fehler:

$$\delta_a = \frac{|\tilde{a} - a|}{|a|}$$

Vektornormen

p-Norm:

$$\|x\|_p = \left(\sum_{i=1}^n |x_i|^p \right)^{1/p}$$

Konditionszahl

$$\left| \frac{f(\tilde{x}) - f(x_0)}{f(x_0)} \right| \approx \kappa_{rel}(x_0) \cdot \left| \frac{(\tilde{x} - x_0)}{x_0} \right|$$

mit

$$\kappa_{rel}(x_0) = \left| f'(x_0) \cdot \frac{x_0}{f(x_0)} \right|$$

Die Zahl

$$\kappa_{rel}(x_0)$$

heisst **relative Konditionszahl** des Problems f an der stelle x_0 und beschreibt die maximale Verstärkung des relativen Eingabefehlers. Entsprechend bezeichnet man mit

$$\kappa_{abs}(x_0) = |f'(x_0)|$$

die **absolute Konditionsszahl** des Problems f .

Auch gilt:

$$|f(\tilde{x}) - f(x)| \leq \kappa_{abs} \cdot |x - \tilde{x}| \leq \delta$$

und

$$\left| \frac{f(\tilde{x}) - f(x)}{f(x)} \right| \leq \kappa_{rel}(x) \cdot \left| \frac{\tilde{x} - x}{x} \right| \leq \kappa_{rel}(x) \cdot \delta$$

δ sei der höchst erlaubte absolute Fehler von $f(\tilde{x})$

Kondition

Mit der relativen / absoluten Kondition eines durch f beschriebene Problems bezeichnet man das Verhältnis:

$$\kappa_{rel} = \frac{\delta_y}{\delta_x}$$

$$\kappa_{abs} = \frac{\|\Delta y\|_Y}{\|\Delta x\|_X}$$

Zahlendarstellung

Darstellung von dezimalen Zahlen als n-stellige Gleitpunktzahlen:

$$x = m \cdot 10^e$$

m : Mantisse

e : Exponent (Stellenverschiebung)

n : Anzahl der signifikanten Dezimalstellen in der Mantisse

1. normierte Gleitpunktdarstellung (wissenschaftliche Schreibweise)
2. Zählen der Signifikanten Ziffern (= n)

Numerisches Differenzieren

gegeben:

Punkte x_i mit $i = 0, 1, 2, 3, 4, \dots$

Funktionswerte $f(x_i)$

Konstanter Abstand zwischen Punkten: $h = x_i - x_{i-1}$

Näherungen für $f'(x_i)$:

Vorwärtsdifferenzenquotient:

$$f'(x_i) \approx \frac{f(x_i + h) - f(x_i)}{h} = \frac{f(x_{i+1}) - f(x_i)}{h}$$

(Steigung der Sekanten durch $(x_i; f(x_i))$ und $(x_{i+1}; f(x_{i+1}))$)

Rückwärtsdifferenzenquotient:

$$f'(x_i) \approx \frac{f(x_i) - f(x_i - h)}{h} = \frac{f(x_i) - f(x_{i-1})}{h}$$

(Steigung der Sekanten durch $(x_{i-1}; f(x_{i-1}))$ und $(x_i; f(x_i))$)

zentraler Differenzenquotient:

$$f'(x_i) \approx \frac{f(x_{i+1}) - f(x_{i-1}))}{2h} = \frac{f(x_i + h) - f(x_i - h)}{2h}$$

(Steigung der Sekanten durch $(x_{i-1}; f(x_{i-1}))$ und $(x_{i+1}; f(x_{i+1}))$)

einseitige Differenzenformel dritter Ordnung:

$$f'(x_i) \approx D_f(x_i, h) = \frac{2f(x_i + h) + 3f(x_i) - 6f(x_i - h) + f(x_i - 2h)}{6h}$$

lineare Gleichungen

Allgemein

Eingabe:

$$\vec{b} \in \mathbb{R}^n \xrightarrow{A \cdot \vec{x} = \vec{b}} \vec{x} \in \mathbb{R}^n$$

lineare Abbildung:

Die Kondition der aufgabe ist durch die Konditionsszahl von A gegeben.

Numerisches Verfahren zur Lösung von LGS $\rightarrow$ DIREKTE Verfahren:

Eingabegrößen: $\vec{b}, A$

Ausgabe: $\vec{\tilde{x}} \approx \vec{x}$

Unendlich-Norm, maximale absolute Zeilensumme

$$\|B\|_{\infty} = \max \sum_{i=1}^n |b_{i,k}|$$

(**Achtung** Betrag des Eintrags der Matrix zur berechnung nehmen)

Bsp.:

$$B = \begin{pmatrix} 1 & 2 \\ 3 & -4 \\ 5 & 6 \end{pmatrix} \rightarrow \|B\|_{\infty} = \max\{1+2, 3+4, 5+6\} = 11$$

Bei Vektoren:

$$\|\vec{v}\|_{\infty} = \max\{|v_1|, |v_2|, |v_3|\}$$

1-Norm, maximale absolute Spaltensumme

$$\|B\|_1 = \max \sum_{i=1}^m |b_{i,k}|$$

$$\|B\|_1 = \max\{1+3+5, 2+4+6\} = 12$$

Bei Vektoren:

$$\|\vec{v}\|_1 = |v_1| + |v_2| + |v_3|$$

2-Norm, Spektralnorm

$$\|A\|_2 = \sqrt{\lambda_{max}(A^T A)}$$

Bsp.:

$$A = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

$$A^T \cdot A = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix} \text{ mit Eigenwerten } \lambda_{1,2} = 2$$

$$\|A\|_2 = \sqrt{2}$$

Bei Vektoren:

$$\|\vec{v}\|_2 = \sqrt{|v_1|^2 + |v_2|^2 + |v_3|^2}$$

Algorithmen zur Lösung von LGS

Rückwärtseinsetzen

Für die Gleichung:

$$R \cdot \vec{x} = \vec{b}$$

(R sei eine obere Dreiecksmatrix)

Zur Berechnung des Vektors $\vec{x}$

$$x_j = \frac{b_j - \sum_{k=j+1}^n r_{j,k} x_k}{r_{j,j}}$$

Gauss-Elimination

- gegeben: $A \in \mathbb{R}^{n \times n}$ $b \in \mathbb{R}^n$, wobei $\det(A) \neq 0$
- Für $j = 1, 2 \dots, n - 1$ und falls $a_{j,j}^{(j)} \neq 0$ ist
- für $i = j + 1, j + 2 \dots, n$
- subtrahiere (Ab) Zeile j mit Faktor $l_{i,j} = \frac{a_{i,j}}{a_{j,j}}$ von Zeile i

gegeben sei: $A \cdot \vec{x} = \vec{a}$

1. Eliminationsschritt (1):

Pivot-Element: a_{11}
(Betragsmässig grösstes Element der ersten Spalte in Matrix A)

Skalierungsfaktoren:

ℓ_{21} (für Zeile 2) = $\frac{a_{21}}{a_{11}}$

ℓ_{31} (für Zeile 3) = $\frac{a_{31}}{a_{11}}$

2. Zeile 2 $Z_2^{(1)}$ der neuen Matrix $A^{(1)}$

Elemente der Zeile 2 = Z_2

$Z_2^{(1)} = Z_2 - \ell_{21} \cdot Z_1$

3. Zeile 3 $Z_3^{(1)}$ der neuen Matrix $A^{(1)}$

$Z_3^{(1)} = Z_3 - \ell_{31} \cdot Z_1$

4. Durch Zeilentausch auf obere Dreiecksform bringen. $\Rightarrow A_{neu}^{(2)}$

5. neue Zeilen von $\vec{a}$:

$Z_{2\vec{a}}^{(1)} = Z_2(\text{von } \vec{a}) - \ell_{21} \cdot Z_1(\text{von } \vec{a})$

$Z_{3\vec{a}}^{(1)} = Z_3(\text{von } \vec{a}) - \ell_{31} \cdot Z_1(\text{von } \vec{a})$

vom $\vec{a}_{neu}$ ebenfalls Zeilen vertauschen gemäss $A_{neu}^{(1)}$

6. Rückeinsetzen und nach x_1, x_2, x_3 auflösen

LR-Zerlegung

LR-Zerlegung ist die aufteilung einer Matrix A in eine untere Dreiecksmatrix L und obere Dreiecksmatrix R sodass $A = LR$ gilt. Dabei ist:

- L eine untere Dreiecksmatrix mit Einsen auf der Hauptdiagonalen.
- R eine obere Dreiecksmatrix.

man merke:

- Skalierung verbessert die Konditionszahl der Matrix.
- Pivotierung verbessert die Stabilität der Gauss-Elimination / LR-Zerlegung.

Ablauf:

1. Skalierung durch Diagonalmatrix D
2. Pivotmatrix P (Diagonalmatrix mit Einsen) zur notation der Zeilentaucher verwenden.

3. Diagonalmatrix D bestimmen. Diagonaleinträge d_{11}, d_{22}, d_{33} sind die Inverse der betragsmässigen Summen jeder Zeile.
4. $D \cdot A$ gibt uns die Skalierte Matrix A .
5. Zeilentausch sodass Betragsmässig grösstes Element der Spalte 1 zu oberst ist.
6. P -Matrix gemäss Zeilentausch notieren.
7. Ersten Eliminationsschritt für Zeilen 2 und 3 druchführen.
8. Zeilentausch sodass Betragsmässig grösstes Element der Spalte 2 zu oberst ist.
9. P -Matrix gemäss Zeilentausch aktualisieren.
10. Zweiten Eliminationsschritt für Zeilen 2 und 3 druchführen.
11. Resultat ist die R -Matrix als obere Dreiecksmatrix
12. L -Matrix mit Eliminierungsfaktoren in der jeweiligen Position.

Lineare Ausgleichsrechnung

Gesucht sei die Ausgleichsgerade durch n gegebene Punkte $P(x_i; y_i)$ mit Grundfunktion $y = x$:

Normalengleichungen:

$A^T A \lambda = A^T y$

$A^T A = \begin{pmatrix} \sum_{i=1}^n x_i^2 & \sum_{i=1}^n x_i \\ \sum_{i=1}^n x_i & n \end{pmatrix}$

$\lambda = (m, b)^T$

Steigung m und Y-Achsenabschnitt b

$A^T y = \begin{pmatrix} \sum_{i=1}^n x_i y_i \\ \sum_{i=1}^n y_i \end{pmatrix}$

$\Rightarrow$ Gleichungssystem lösen

minimale Fehlerquadratsumme:

$E = \sum_{i=1}^n ((mx_i + n) - y_i)^2$

Bei anderen Linearen Ansatzfunktionen bspw: $f(x) = a \cdot e^x + b$ gilt:

$A^T A = \begin{pmatrix} \sum_{i=1}^n (e^{x_i})^2 & \sum_{i=1}^n e^{x_i} \\ \sum_{i=1}^n e^{x_i} & n \end{pmatrix}$

und

$A^T y = \begin{pmatrix} \sum_{i=1}^n e^{x_i} y_i \\ \sum_{i=1}^n y_i \end{pmatrix}$

allg. Aufstellen des Gleichungssystems für lineare Ausgleichsrechnung

gegeben sind **Messpunkte** t_i sowie **Funktionswerte** f_i
gesucht sind unbekante Parameter: $\vec{\lambda} = (\lambda_1, \lambda_2, \dots)$

wir wissen:

$A \cdot \vec{\lambda} \approx f$

Durch einsetzen der Messpunkte t_i können wir A bestimmen.

Das klassische **Normalengleichungssystem** ist:

$A^T A \vec{\lambda} = A^T f$

Nichtlineare Ausgleichsrechnung

Lineare Gleichungen $f_i(t)$ können in der form:

$f(t) = \lambda_1 \cdot \phi_1(t) + \lambda_2 \cdot \phi_2(t) + \dots + \lambda_n \cdot \phi_n(t)$
geschrieben werden.

Eine Gleichung kann jedoch durch transformation Linear gemacht werden: (bspw. durch Quadrieren oder anwendung des Logarithmus)
der Parametervektor $\vec{\lambda}$ setzt sich aus allen unbekannten Parametern der Funktion f zusammen:

$\vec{\lambda} = \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \vdots \\ \lambda_n \end{pmatrix}$

Die Abbildungsmatrix A wird so aufgestellt, dass $A \cdot \vec{\lambda} = f$.

bsp:

$A = \begin{pmatrix} e^{t_1} & t_1 \\ e^{t_2} & t_2 \\ e^{t_3} & t_3 \\ \vdots & \vdots \\ e^{t_n} & t_n \end{pmatrix}$

Die Jacobi-Matrix $J_F(\vec{\lambda})$ mit der Residuenfunktion:
 $F_i = f(t_i; \vec{\lambda}) - y_i$

$J_F(\vec{\lambda}) = \begin{bmatrix} \frac{\partial F_1}{\partial \lambda_1} & \frac{\partial F_1}{\partial \lambda_2} & \frac{\partial F_1}{\partial \lambda_3} \\ \frac{\partial F_2}{\partial \lambda_1} & \frac{\partial F_2}{\partial \lambda_2} & \frac{\partial F_2}{\partial \lambda_3} \\ \vdots & \vdots & \vdots \\ \frac{\partial F_m}{\partial \lambda_m} & \frac{\partial F_m}{\partial \lambda_m} & \frac{\partial F_m}{\partial \lambda_m} \end{bmatrix}$

Nichtlineare Gleichungen

Zwischenwertsatz:
gegeben sei Funktion f sowie intervall $I = [a_0; b_0]$:

wenn

$f(a_0) \cdot f(b_0) < 0$

hat f in I mindestens eine Nullstelle.

Bisektionsverfahren:
Startwert:

$x_0 = \frac{a_0 + b_0}{2}$

prüfen:

$f(a_0) \cdot f(x_0) > 0 \Rightarrow [a_1; b_1] = [x_0; b_0]$

$f(a_0) \cdot f(x_0) < 0 \Rightarrow [a_1; b_1] = [a_0; x_0]$

$\Rightarrow x_1 = \frac{a_1 + b_1}{2}$

$f(a_0) \cdot f(x_0) = 0 \Rightarrow x_0 = \text{Nullstelle}$

Entwicklung des absoluten Fehlers:

$|x_k - x^*| \leq \frac{|b_0 - a_0|}{2^{k+1}}$

mit Fehlertoleranz f_t (bspw. als Dezimalzahl):

$$\frac{|b_0 - a_0|}{2^{k+1}} < f_t$$

nach k auflösen.

Newtonisches Tangentenverfahren:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Fixpunktiteration:

gesucht ist Schnittpunkt zwischen $h(x)$ und $g(x)$:

$$x^* = h^{-1}(g(x^*)) \qquad F(x^*) = x^*$$

⇒ äquivalente Umformung der Ausgangsfunktionen

Verfahren:

$$x_{k+1} = F(x_k)$$

gesicherte Konvergenz wenn:

$$F : I \rightarrow I \qquad \max |F'(x)| \leq k < 1 \quad x \in I$$

Banachscher Fixpunktsatz:

Nullstelle der Funktion $F(x)$ konvergiert zu s im Intervall $I = [a; b]$ mit Fehlertoleranz f_t und Startwert x_0

Konstante K bestimmen:

$$K = \max_{x \in I} |F'(x)|$$

a priori Fehlerabschätzung:

$$|x_n - \tilde{x}| \leq \frac{K^n}{1 - K} \cdot |x_1 - x_0|$$

Lösen der aufgabe:

$$\frac{K^n}{1 - K} \cdot |F(x_0) - x_0| < f_t$$

nach n auflösen für Mindestanzahl Iterationsschritte.

$$n > \ln \left(\frac{f_t(1 - K)}{|F(x_0) - x_0|} \right) \cdot \frac{1}{\ln(K)}$$

Nichtlineare Gleichungssysteme

Newtonisches Verfahren:

allgemein:

$$\vec{x}_{n+1} = \vec{x}_n - J_f(\vec{x}_n)^{-1} \cdot f(\vec{x}_n)$$

$J_f(\vec{x}_n)$ ist dabei die Jacobi-Matrix von f :

$$J_f = \begin{pmatrix} -\nabla f_1 - \\ -\nabla f_2 - \end{pmatrix} = \begin{pmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} \end{pmatrix}$$

das Gleichungssystem welches zu lösen gilt lautet:

$$J_f(\vec{x}_n) \cdot \vec{\delta}_n = -\vec{f}_n$$

Gauss-Newton-Verfahren:

gegeben: Startvektor $\vec{\lambda}^{(0)} \in \mathbb{R}^n$

Parametervektor $\vec{\lambda}$

Eingabe: Fehlerfunktion

$$\vec{F}(\vec{\lambda}) = (f(x_i; \vec{\lambda}) - y_i)$$

und Jacobimatrix

$$J_F(\vec{\lambda}) = \begin{pmatrix} -\nabla f_1 - \\ -\nabla f_2 - \end{pmatrix} = \begin{pmatrix} \frac{\partial f_1}{\partial \lambda_1} & \frac{\partial f_1}{\partial \lambda_2} \\ \frac{\partial f_2}{\partial \lambda_1} & \frac{\partial f_2}{\partial \lambda_2} \end{pmatrix}$$

berechne:

$$\vec{b}^{(k)} = \left(J_F(\vec{\lambda}^{(k)}) \right)^T \cdot \vec{F}(\vec{\lambda}^{(k)}) \qquad \mathbf{A}^{(k)} = \left(J_F(\vec{\lambda}^{(k)}) \right)^T \cdot J_F(\vec{\lambda}^{(k)})$$

das lineare Gleichungssystem lösen:

$$\mathbf{A}^{(k)} \cdot \vec{s}^{(k)} = \vec{b}^{(k)}.$$

Newton-Schritt:

$$\vec{\lambda}^{(k+1)} = \vec{\lambda}^{(k)} - \vec{s}^{(k)}.$$

Lagrange Interpolation

gegeben:

$(n + 1)$ Stützstellen x_i für $i = 0, 1, 2, 3 \dots$

Stützwerte $f(x_i)$ für $i = 0, 1, 2, 3 \dots$

Das Lagrange-Polynom definiert sich durch:

$$L(x) = \sum_{j=0}^n f(x_j) \cdot \ell_j(x)$$

$$\ell_j(x) = \prod_{m \neq j, 0 < m < n} \frac{x - x_m}{x_j - x_m}$$

Aufstellen von $\ell_j(x)$:

$$\ell_0(x) = \frac{(x - x_1)(x - x_2)(x - x_3)(x - x_4)}{(x_0 - x_1)(x_0 - x_2)(x_0 - x_3)(x_0 - x_4)} \dots$$

$$\ell_1(x) = \frac{(x - x_0)(x - x_2)(x - x_3)(x - x_4)}{(x_1 - x_0)(x_1 - x_2)(x_1 - x_3)(x_1 - x_4)} \dots$$

$$\ell_2(x) = \frac{(x - x_0)(x - x_1)(x - x_3)(x - x_4)}{(x_2 - x_0)(x_2 - x_1)(x_2 - x_3)(x_2 - x_4)} \dots$$

$$\vdots$$

an Stellen i wo $f_i = 0$ muss ℓ_i nicht berechnet werden

Numerisch Integrieren

gegeben:

$$I = \int_a^b f(x) \, dx$$

oder:

n gegebene Punkte x_i mit $i = 0, 1, 2, \dots$ sowie Funktionswerte f_i

einfache Trapezregel:

$$I \approx (b - a) \left(\frac{1}{2} f(a) + \frac{1}{2} f(b) \right)$$

zusammengesetzte Trapezregel:

gegeben sind n Punkte. Somit ist $h = \frac{b-a}{n-1}$:

$$I \approx h \cdot \left(\frac{f(x_0) + f(x_1)}{2} + \frac{f(x_1) + f(x_2)}{2} \dots \frac{f(x_{n-1}) + f(x_n)}{2} \right)$$

einfache Simpsonregel:

mit $h = \frac{b-a}{2}$ oder $h = \frac{b-a}{n-1}$

$$I \approx h \cdot \left(\frac{1}{3} f(a) + \frac{4}{3} f\left(\frac{a+b}{2}\right) + \frac{1}{3} f(b) \right)$$

zusammengesetzte Simpsonregel:

$[a; b]$ wird zerteilt in Teilintervalle mit länge $h = \frac{b-a}{n-1}$.

n muss gerade sein!!

$$I \approx \frac{h}{3} \left[f(x_0) + 4 \sum_{\substack{i=1 \\ \text{ungerade } i}}^{n-1} f(x_i) + 2 \sum_{\substack{i=2 \\ \text{gerade } i}}^{n-2} f(x_i) + f(x_n) \right]$$

Das Lagrange Polynom wird durch die Simpsonregel exakt integriert.

Spline Interpolation

gegeben:

x_i mit $i = 0, 1, 2, 3 \dots$

$f(x_i)$

Bei n gegebenen Punkten suchen wir $n - 1$ Splinefunktionen.

Je zwei benachbarte Punkte werden durch werden durch ein kubisches Polynom interpoliert.

- $S_i(x)$ und deren Ableitungen aufstellen mit Koeffizienten a_i bis d_i (beginnend bei 0)

$$S_i(x) = a_i x^3 + b_i x^2 + c_i x + d_i$$

$$S_i'(x) = 3a_i x^2 + 2b_i x + c_i$$

$$S_i''(x) = 6a_i x + 2b_i$$

- Interpolationsbedingungen:**

$$S_i(x_i) = y_i$$

$$S_i(x_{i+1}) = y_{i+1}$$

- Stetigkeitsbedingungen:**

$$S_i'(x_{i+1}) = S_{i+1}'(x_{i+1})$$

$$S_i''(x_{i+1}) = S_{i+1}''(x_{i+1})$$

- Randbedingungen:**

- Typ 1: keine Krümmung an Intervallenden: (natürliche Splinefunktionen)
 $S_0''(x_0) = 0$
 $S_n''(x_n) = 0$

- Typ 2: periodische Kubische splines:
 $S_1'(x_0) = S_n'(x_n)$ und $S_1''(x_0) = S_n''(x_n)$

- Typ 3: clamped Splines oder Hermite (vollständige) kubische Splines:
 $S_1'(x_0) = q_a$ und $S_n'(x_n) = q_b$
 $q_a = f'(x_0)$ und $q_b = f'(x_n)$

Bei n gegeben Messpunkten (x_0 bis x_n) gibt es $(n - 1)$ Splinefunktionen und damit $4 \cdot (n - 1)$ Gleichungen für die Unbekannten.

Lösen von AWP's

gegeben: DGL erster Ordnung:

y'(x) = dy/dx = f(x; y(x)) y(0) = y_0

gesucht:

y(x) für x ∈ I = [a; b] in k Schritten

explizites Euler-Verfahren

y_{k+1} = y_k + h \cdot f(x_k; y_k)

h = (b - a) / k

Vorgehen k-mal wiederholen bis Intervallende b mit x_k erreicht ist: Start mit x_0 = a

x_0 mit y_0
x_1 mit y_1 = y_0 + h \cdot f(x_0; y_0) \approx y(x_1)
x_2 mit y_2 = y_1 + h \cdot f(x_1; y_1) \approx y(x_2)

⋮

in jedem Schritt x_{k+1} = x_k + h

Stabilitätskriterium des Expliziten Euler Verfahrens

Durch die Eigenwerte λ_i der Jacobimatrix J_f des Gleichungssystems können wir folgende Stabilitätskriterien für das Explizite Euler Verfahren herleiten:

h_{krit} < 2 / max |λ_i| für λ_i ∈ ℝ
h_{krit} < min [2 \cdot |Re(λ_i)| / |λ_i|^2] für λ_i ∈ ℂ

implizites Euler-Verfahren

y_{k+1} = y_k + h \cdot f(x_{k+1}; y_{k+1}) ⇔ y_{k+1} = (y_k + h \cdot f(x_{k+1})) / (1 + 2h)

h = (b - a) / k

Vorgehen k-mal wiederholen.

Runge Verfahren

Explizit und konvergent von Ordnung 2.
Ordnung: O(h^2)

y_{k+1} = y_k + h \cdot r_2

r_1 = f(x_k; y_k)

r_2 = f(x_k + 1/2 h; y_k + 1/2 h \cdot r_1)

Stufe s = 2 0 | 0 0
 1/2 | 1/2 0
 0 | 0 1

Klassisches Runge-Kutta-Verfahren

y_{k+1} = y_k + h/6 (r_1 + 2r_2 + 2r_3 + r_4)

r_1 = f(x_k; y_k)

r_2 = f(x_k + 1/2 h; y_k + 1/2 h r_1)

r_3 = f(x_k + 1/2 h; y_k + 1/2 h r_2)

r_4 = f(x_k + 1h; y_k + 1h r_3)

y_{k+1} = y_k + h (1/6 r_1 + 1/3 r_2 + 1/3 r_3 + 1/6 r_4)

Stufe s = 4 0 | 0 0
 1/2 | 1/2 0
 1 | 0 0 0
 1/6 | 1/6 1/3 1/3 1/6

Butcher-Tableau

r_1 = f(x_k + c_1 h, y_k + h(a_{11}r_1 + a_{12}r_2 + ... + a_{1s}r_s))
r_2 = f(x_k + c_2 h, y_k + h(a_{21}r_1 + a_{22}r_2 + ... + a_{2s}r_s))
⋮
r_s = f(x_k + c_s h, y_k + h(a_{s1}r_1 + a_{s2}r_2 + ... + a_{ss}r_s))
y_{k+1} = y_k + h(b_1r_1 + b_2r_2 + ... + b_sr_s)

(c | A / b^T) respektive (c_1 | a_{11} a_{12} ... a_{1s} / c_s a_{s1} a_{s2} ... a_{ss} / b_1 b_2 ... b_s)

DGL-Systeme

gegeben sei DGL

ẍ + 10ẋ^2 + 10x = cos(3t - 0.3)

mit Anfangswerten:

x(0) = 1 ẋ(0) = 0

Transformation in System Differenzialgleichungen erster Ordnung:

d/dt z(t) = f̃(t; z)

- 1. Wir definieren x (ohne Ableitung) als z_1
- 2. und ẋ (erste Ableitung) als z_2
- 3. somit ist ẍ = ż_2 und ż_1 = ẋ = z_2
- 4. zusammengefasst:

x = z_1
ẋ = z_2
ż_1 = ẋ = z_2
ẍ = ż_2

- 5. einsetzen für d/dt z(t):

(ż_1 / ż_2) = (-10z_2^2 - 10z_1 + cos(3t - 0, 3)) / z_2

O-Notation

Die O -Notation beschreibt das asymptotische Verhalten eines Terms (meist eines Fehlers) für kleine Werte einer Variablen typischerweise h → 0 (z.B. Schrittweite) oder n → ∞ (z.B. Anzahl Schritte).

- Die kleinste Potenz von h sagt aus, wie schnell der Fehler bei feinerer Schrittweite verschwindet.
- Wenn im Fall von O(h^2) das h halbiert wird, viertelt sich der Fehler.
- Höhere Ordnung ⇒ schneller genau ⇒ effizientere Methode.
- Aber: Höhere Ordnung kann mehr Rechenaufwand oder Instabilität bedeuten.

Typische Beispiele:

Fehlerordnung:

Vorwärtsdifferenz (Ableitung)	O(h)
Zentrale Differenz	O(h^2)
Trapezregel	O(h^2)
Simpsonregel	O(h^4)
Runge-Kutta (4. Ordnung)	O(h^4)