

Numerische Lösung nicht linearer Gleichungssysteme

LGS = lineares Gleichungssystem, NGS = nichtlineares Gleichungssystem

Einleitendes Beispiel

Gesucht sind die Lösungen

des Gleichungssystems:

$$f_1(x_1, x_2) = x_1^2 + x_2 - 11 = 0$$

$$f_2(x_1, x_2) = x_1 + x_2^2 - 7 = 0$$

Diese lassen sich interpretieren als die Nullstellen der Funktion:

$$\mathbf{f}: \mathbb{R}^2 \rightarrow \mathbb{R}^2 \quad \mathbf{f}(x) = \begin{pmatrix} f_1(x_1, x_2) \\ f_2(x_1, x_2) \end{pmatrix} = \begin{pmatrix} x_1^2 + x_2 - 11 \\ x_1 + x_2^2 - 7 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Ein solches System lässt sich nicht in die Form $Ax = b$ bringen.

Geometrisch lassen sich die Lösungen als Schnittpunkte der beiden Funktionen interpretieren.

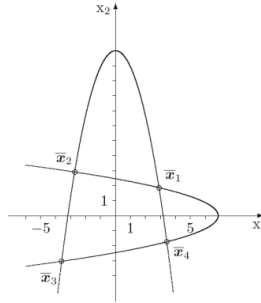
Explizite Darstellung der Kurven:

$$x_2 = 11 - x_1^2 \text{ und } x_2 = \sqrt{7 - x_1}$$

Schnittpunkte:

$$\bar{x}_1 = \begin{pmatrix} 3 \\ 2 \end{pmatrix}, \quad \bar{x}_2 = \begin{pmatrix} -2.8 \\ 3.2 \end{pmatrix}$$

$$\bar{x}_3 = \begin{pmatrix} -3.8 \\ -3.3 \end{pmatrix}, \quad \bar{x}_4 = \begin{pmatrix} 3.4 \\ -1.7 \end{pmatrix}$$



Funktionen mit mehreren Variablen

Skalarwertige Funktionen $f: D \subset \mathbb{R}^n \rightarrow W \subset \mathbb{R}$

$$(x_1, x_2, \dots, x_n) \mapsto y = f(x_1, x_2, \dots, x_n)$$

Unter einer Funktion f mit n unabhängigen Variablen x_1, \dots, x_n und einer abhängigen Variablen y versteht man eine Vorschrift, die jedem geordneten Zahlentupel (x_1, x_2, \dots, x_n) aus einer Definitionsmenge $D \subset \mathbb{R}^n$ genau ein Element $y \in W \subset \mathbb{R}$ zuordnet.

Da das Ergebnis $y \in \mathbb{R}$ ein Skalar (eine Zahl) ist, redet man auch von einer **skalarwertigen Funktion**.

Vektorwertige Funktion gibt einen **Vektor** zurück (statt Skalar)

Sei $\mathbf{f}: \mathbb{R}^n \rightarrow \mathbb{R}^m$ eine Funktion mit n Variablen.

$$\mathbf{f}(x_1, \dots, x_n) = \begin{pmatrix} y_1 = f_1(x_1, x_2, \dots, x_n) \\ y_2 = f_2(x_1, x_2, \dots, x_n) \\ \vdots \\ y_m = f_m(x_1, x_2, \dots, x_n) \end{pmatrix}$$

wobei die m Komponenten $f_i: \mathbb{R}^n \rightarrow \mathbb{R}$ für $i = 1, 2, \dots, m$ von \mathbf{f} wieder **skalarwertige Funktionen** sind.

Eigenschaften von skalar- und vektorwertigen Funktionen

- Skalar- und vektorwertige Funktionen mit mehreren Variablen werden auch **multivariat** genannt.
- Wie bei einem Vektor \mathbf{x} stellen wir zur besseren Unterscheidbarkeit vektorwertige Funktionen \mathbf{f} fett dar, im Gegensatz zu Skalaren x und skalarwertigen Funktionen f .
- Wir werden uns bei der Lösung nichtlinearer Gleichungssysteme auf vektorwertige Funktionen $\mathbf{f}: \mathbb{R}^n \rightarrow \mathbb{R}^m$ konzentrieren.

Grundlegende Rechenoperationen

können als Skalarwertige Funktionen $f: \mathbb{R}^2 \rightarrow \mathbb{R}$ oder als Vektorwertige Funktionen $\mathbf{f}: \mathbb{R}^2 \rightarrow \mathbb{R}^2$ interpretiert werden:

$$f(x, y) = x + y, \quad g(x, y) = x \cdot y, \quad h(x, y) = x^2 + y^2$$

$$\mathbf{f}(x, y) = \begin{pmatrix} x+y \\ x \cdot y \end{pmatrix}, \quad \mathbf{g}(x, y) = \begin{pmatrix} x \cdot y \\ x^2 + y^2 \end{pmatrix}$$

Lineare Funktionen von LGS Gebe die lineare Funktion $\mathbf{f}: \mathbb{R}^3 \rightarrow \mathbb{R}^3$ an, für welche die Lösung \mathbf{x} des LGS:

$$\mathbf{A}\mathbf{x} = \mathbf{b} \text{ mit } \mathbf{A} = \begin{pmatrix} 4 & -1 & 1 \\ -2 & 5 & 1 \\ 1 & -2 & 5 \end{pmatrix} \text{ und } \mathbf{b} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \text{ gerade } \mathbf{f}(\mathbf{x}) = \mathbf{0} \text{ ergibt:}$$

$$\mathbf{A}\vec{x} = \vec{b} \Rightarrow \underbrace{\mathbf{A}\vec{x} - \vec{b}}_{\vec{f}(\vec{x})} = \vec{0} \Rightarrow \vec{f}(x_1, x_2, x_3) = \mathbf{0} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

$$\vec{f}(\vec{x}) = \mathbf{A}\vec{x} - \vec{b} = \begin{pmatrix} 4 & -1 & 1 \\ -2 & 5 & 1 \\ 1 & -2 & 5 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} - \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

$$\mathbf{f}(x_1, x_2, x_3) = \begin{pmatrix} f_1 = 4x_1 - x_2 + x_3 - 5 \\ f_2 = -2x_1 + 5x_2 + x_3 - 11 \\ f_3 = x_1 - 2x_2 + 5x_3 - 12 \end{pmatrix}$$

Darstellungsformen

Analytische Darstellung

- Explizite Darstellung:** $y = f(x_1, \dots, x_n)$
 - die Funktionsgleichung ist nach einer Variablen aufgelöst
 - Beispiel: $y = 2 \cdot e^{x_1^2 + x_2^2}$
- Implizite Darstellung:** $F(x, y) = 0$
 - die Funktionsgleichung ist nicht nach einer Variablen aufgelöst
 - daher handelt es sich um eine Funktion mit nur $n - 1$ unabhängigen Variablen
 - Beispiel: $x_1^2 + x_2^2 - 1 = 0$
- Parameterdarstellung:** $x = x(t), y = y(t)$
 - die Funktion wird durch eine Kurve im Raum beschrieben
 - Beispiel: $x(t) = \cos(t), y(t) = \sin(t)$

Darstellung durch Wertetabelle Sei $f: \mathbb{R}^n \rightarrow \mathbb{R}^m$ eine Funktion.

In die vorausgesetzte Funktionsgleichung $z = f(x, y)$ werden die Werte der unabhängigen Variablen x und y eingesetzt (der Reihe nach).

So erhält man eine Wertetabelle, bzw. Matrix:

2. unabhängige Variable y

$x \backslash y$	y_1	y_2	\dots	y_k	\dots	y_n
x_1	z_{11}	z_{12}	\dots	z_{1k}	\dots	z_{1n}
x_2	z_{21}	z_{22}	\dots	z_{2k}	\dots	z_{2n}
\vdots	\vdots	\vdots	\dots	\vdots	\dots	\vdots
x_i	z_{i1}	z_{i2}	\dots	z_{ik}	\dots	z_{in}
\vdots	\vdots	\vdots	\dots	\vdots	\dots	\vdots
x_m	z_{m1}	z_{m2}	\dots	z_{mk}	\dots	z_{mn}

↑
 k -te Spalte

1. unabhängige Variable x

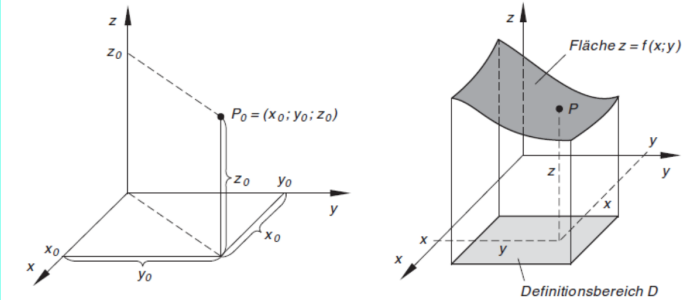
← i -te Zeile

Grafische Darstellung Wir beschränken uns hier auf skalarwertige Funktionen mit zwei unabhängigen Variablen $f: \mathbb{R}^2 \rightarrow \mathbb{R}$.

Dazu betrachten wir die Funktion $z = f(x, y)$ in einem dreidimensionalen kartesischen Koordinatensystem:

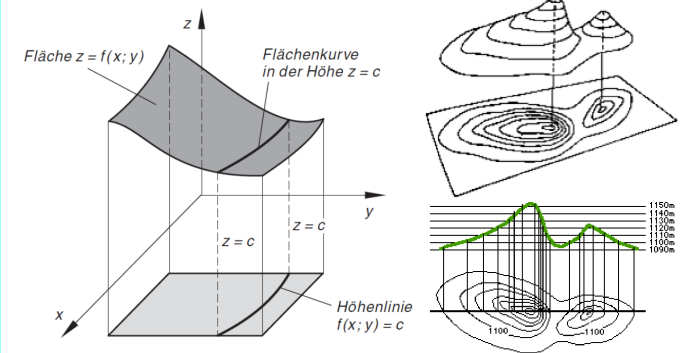
Darstellung einer Funktion als Fläche im Raum

Die Funktion f ordnet jedem Punkt $(x, y) \in D$ in der Ebene einen Wert $z = f(x, y)$ zu, der als Höhenkoordinate verstanden werden kann. Durch die Anordnung der Punkte $(x, y, f(x, y))$ im dreidimensionalen Koordinatensystem wird eine über dem Definitionsbereich D liegende Fläche ausgezeichnet:



Schnittkurviendiagramm

Wird die Fläche $z = f(x, y)$ bei einer konstanten Höhe $z = \text{const.}$ geschnitten, ergibt sich eine Schnittkurve. Wird diese in die (x, y) -Ebene projiziert, spricht man von einer Höhenlinie bzw. bei der Abbildung von einem Höhenliniendiagramm., wie wir es z.B. von Wanderkarten her kennen. Natürlich kann man auch andere Schnitte als $z = \text{const.}$ (Schnittebene parallel zur (x, y) -Ebene) wählen, z.B. $x = \text{const.}$ (Schnittebene parallel zur (y, z) -Ebene) oder $y = \text{const.}$ (Schnittebene parallel zur (x, z) -Ebene):



Partielle Ableitungen

Partielle Ableitungen 1. Ordnung

Unter den partiellen Ableitungen 1. Ordnung einer Funktion $z = f(x, y)$ an der Stelle (x, y) werden die folgenden Grenzwerte verstanden:
Partielle Ableitung nach x:

df/dx(x,y) = lim_{dx -> 0} (f(x+dx,y) - f(x,y))/dx

Partielle Ableitung nach y:

df/dy(x,y) = lim_{dy -> 0} (f(x,y+dy) - f(x,y))/dy

Alternative Notationen: f_x(x,y), f_y(x,y) oder f_x, f_y

Partielle Ableitung In einer Dimension

f'(x_0) = lim_{dx -> 0} (f(x_0+dx) - f(x_0))/dx

In mehreren Dimensionen

- 1. Ableitung nach x: f_x = df/dx(x,y) = lim_{dx -> 0} (f(x+dx,y) - f(x,y))/dx
- 2. Ableitung nach y: f_y = df/dy(x,y) = lim_{dy -> 0} (f(x,y+dy) - f(x,y))/dy

Partielle Ableitungen berechnen

- 1. Variable identifizieren: Bestimme, nach welcher Variable abgeleitet werden soll
- 2. Alle anderen Variablen werden während der Ableitung als Konstanten betrachtet
- 3. Standardableitungsregeln anwenden und Ergebnis korrekt notieren

Partielle Ableitungen berechnen Berechne die partiellen Ableitungen von z = f(x, y) = 3xy^2 + ln(x^3y^2) an der Stelle (x_0, y_0) = (-1, 1).

In beiden Dimensionen berechnen

f_x = 3 * 1 * y^2 + 1/(x^3y^2) * 3x^2 * y^2 = 3y^2 + 3/x

f_y = 3x * 2y + 1/(x^3y^2) * x^3 * 2y = 6xy + 2/y

An der Stelle (-1, 1):
f_x(-1, 1) = 3 * 1^2 + 3/(-1) = 3 - 3 = 0
f_y(-1, 1) = 6 * (-1) * 1 + 2/1 = -6 + 2 = -4

Tangentensteigung z = f(x, y) = 3xy^3 + 10x^2y + 5y + 3y * sin(5xy)

df/dx(x,y) = 3 * 1 * y^3 + 10 * 2x * y + 0 + 3y * cos(5xy) * 5 * 1 * y

df/dy(x,y) = 3x * 3y^2 + 10x^2 * 1 + 5 * 1 + (3 * 1 * sin(5xy) + 3y * cos(5xy) * 5x * 1)

Steigung der beiden Tangenten in x-/y-Richtung im Punkt: (x_0, y_0) = (-1, 1)

Jacobi-Matrix und Linearisierung

Jacobi-Matrix

Sei f : R^n -> R^m mit y = f(x) und x = (x_1, x_2, ..., x_n)^T in R^n. Die Jacobi-Matrix enthält sämtliche partiellen Ableitungen 1. Ordnung von f und ist definiert als:

f(x) = (y_1=f_1(x), y_2=f_2(x), ..., y_m=f_m(x))^T -> Df(x) := [df_1/dx_1(x) df_1/dx_2(x) ... df_1/dx_n(x); df_2/dx_1(x) df_2/dx_2(x) ... df_2/dx_n(x); ...; df_m/dx_1(x) df_m/dx_2(x) ... df_m/dx_n(x)]

Linearisierung Die verallgemeinerte Tangentengleichung

g(x) = f(x^(0)) + Df(x^(0)) * (x - x^(0))

beschreibt eine lineare Funktion und es gilt f(x) approx g(x) in einer Umgebung eines gegebenen Vektors x^(0) = (x_1^(0), x_2^(0), ..., x_n^(0))^T in R^n. Man spricht von der Linearisierung der Funktion y = f(x) in einer Umgebung von x^(0) (ein hochgestellter Index in Klammern x^(k) bezeichnet wie bisher einen Vektor aus R^n nach der k-ten Iteration).

Tangentialebene Für den speziellen Fall f : R^2 -> R mit y = f(x_1, x_2) und x^(0) = (x_1^(0), x_2^(0))^T in R^2 ist die Jacobi-Matrix nur ein Zeilenvektor mit zwei Elementen, nämlich:

Df(x^(0)) = (df/dx_1(x_1^(0), x_2^(0)), df/dx_2(x_1^(0), x_2^(0)))

Dann liefert die Linearisierung g(x_1, x_2):

= f(x_1^(0), x_2^(0)) + (df/dx_1(x_1^(0), x_2^(0)), df/dx_2(x_1^(0), x_2^(0))) * (x_1 - x_1^(0), x_2 - x_2^(0))^T

= f(x_1^(0), x_2^(0)) + df/dx_1(x_1^(0), x_2^(0)) * (x_1 - x_1^(0)) + df/dx_2(x_1^(0), x_2^(0)) * (x_2 - x_2^(0))

die Gleichung der Tangentialebene.

Sie enthält sämtliche im Flächenpunkt P = (x_1^(0), x_2^(0), f(x_1^(0), x_2^(0))) an die Bildfläche von y = f(x_1, x_2) angelegten Tangenten.

Jacobi-Matrix berechnen und linearisieren

Sei f : R^n -> R^m mit y = f(x) und x = (x_1, x_2, ..., x_n)^T in R^n.

- 1. Identifiziere die Komponentenfunktionen f_1, f_2, ..., f_m und Variablen x_1, x_2, ..., x_n.
- 2. Berechne die partiellen Ableitungen df_i/dx_j für i = 1, ..., m und j = 1, ..., n.
- 3. Stelle die Jacobi-Matrix Df(x) auf, indem du die partiellen Ableitungen in Matrixform anordnest.
- 4. Werte die Jacobi-Matrix an einem Entwicklungspunkt x^(0) aus, indem du die Werte der Variablen in die Matrix einsetzt.
- 5. Berechne die Linearisierung der Funktion g(x) mit der Formel: g(x) = f(x^(0)) + Df(x^(0)) * (x - x^(0))

Jacobi-Matrix und Linearisierung Berechnen Sie die Jacobi-Matrix von f(x_1, x_2) = (5x_1x_2 / (x_1^2x_2^2 + x_1 + 2x_2)) an der Stelle (x_1 / x_2) = (1 / 2) und linearisieren Sie die Funktion dort.

Schritt 1: Partielle Ableitungen berechnen

df_1/dx_1 = 5x_2, df_1/dx_2 = 5x_1

df_2/dx_1 = 2x_1x_2^2 + 1, df_2/dx_2 = 2x_1^2x_2 + 2

Schritt 2: Jacobi-Matrix aufstellen

Df(x_1, x_2) = [5x_2 / (2x_1x_2^2 + 1), 5x_1 / (2x_1^2x_2 + 2)]

Schritt 3: An der Stelle (1, 2) auswerten Df(1, 2) = [10/9, 5/6]

Schritt 4: Funktionswert berechnen f(1, 2) = (10/9)

Schritt 5: Linearisierung g(x) = (10/9) + [10/9, 5/6] * (x_1 - 1, x_2 - 2)^T

Linearisierung einer vektorwertigen Funktion

Linearisiere Sie für x^(0) = (pi/4, 0, pi)^T der Funktion f(x_1, x_2, x_3)

1. Jacobi-Matrix Df(x_1, x_2, x_3) bilden:

f(x_1, x_2, x_3) = (sin(x_2 + 2x_3) / cos(2x_1 + x_2))

Df(x_1, x_2, x_3) = [0 cos(x_2 + 2x_3) 2cos(x_2 + 2x_3); -2sin(2x_1 + x_2) -sin(2x_1 + x_2) 0]

2. Startvektor x^(0) in Vektorwertige Funktion f(x) und Jacobi-Matrix Df(x) einsetzen:

f(pi/4, 0, pi) = f(x^(0)) = (sin(0 + 2pi) / cos(2 * pi/4 + 0))

= (0/0), Df(x^(0)) = [0 1 2; -2 -1 0]

3. Verallgemeinerte Tangentengleichung:

g(x) = (0/0) + [0 1 2; -2 -1 0] * (x_1 - pi/4, x_2 - 0, x_3 - pi)^T / (x - x_0)

= (x_2 + 2x_3 - 2pi / -2x_1 - x_2 + pi/2)

Nullstellenbestimmung für NGS

Es gibt keine einfachen Methoden, um festzustellen, ob ein nichtlineares Gleichungssystem lösbar ist und wie viele Lösungen es hat. Deshalb entscheidet die Wahl einer »geeigneten Startnäherung« meist über Erfolg oder Misserfolg der eingesetzten numerischen Verfahren.

Problemstellung zur Nullstellenbestimmung

Gegeben sei $n \in \mathbb{N}$ und eine Funktion $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$. Gesucht ist ein Vektor $\bar{x} \in \mathbb{R}^n$ mit $f(\bar{x}) = 0$.
Komponentenweise bedeutet dies: Gegeben sind n Funktionen $f_i : \mathbb{R}^n \rightarrow \mathbb{R}$, die die Komponenten von f bilden. Gesucht ist ein Vektor $\bar{x} \in \mathbb{R}^n$ mit $f_i(\bar{x}) = 0$ für $i = 1, \dots, n$.

Quadratisch konvergentes Newton-Verfahren (Quadratische Konv.)
Gesucht sind Nullstellen von $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$. Sei $x^{(0)}$ ein Startvektor in der Nähe einer Nullstelle. Das Newton-Verfahren zur näherungsweisen Bestimmung dieser Nullstelle lautet:
Lösung von $f(x) = 0$ mit $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$ für $n = 0, 1, 2, \dots$
1. Berechne $f(x^{(n)})$ und $Df(x^{(n)})$
2. Berechne $\delta^{(n)}$ als Lösung des linearen Gleichungssystems

$$Df(x^{(n)}) \cdot \delta^{(n)} = -f(x^{(n)})$$

3. Setze $x^{(n+1)} := x^{(n)} + \delta^{(n)}$

Vereinfachtes Newton-Verfahren (Lineare Konvergenz)
Lösung von $f(x) = 0$ mit $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$ für $n = 0, 1, 2, \dots$
1. Berechne $f(x^{(n)})$ und $Df(x^{(0)})$
2. Berechne $\delta^{(n)}$ als Lösung des lin. GS $Df(x^{(0)}) \cdot \delta^{(n)} = -f(x^{(n)})$
3. Setze $x^{(n+1)} := x^{(n)} + \delta^{(n)}$

Newton-Verfahren für nichtlineare Gleichungssysteme

Schritt 1: Funktionen und Jacobi-Matrix aufstellen
Definiere $f(x) = 0$ und berechne die Jacobi-Matrix $Df(x)$.

Schritt 2: Startvektor wählen
Wähle einen geeigneten Startvektor $x^{(0)}$ nahe der vermuteten Lösung.

Schritt 3: Iterative Berechnung
Für jede Iteration k :
• **Linearisierung um $x^{(k)}$:** Berechne $f(x^{(k)})$ und $Df(x^{(k)})$
• **Nullstelle der Linearisierung berechnen:**
Löse das LGS: $Df(x^{(k)})\delta^{(k)} = -f(x^{(k)})$
• **Nächste Iteration:** Setze $x^{(k+1)} = x^{(k)} + \delta^{(k)}$

Formel: $x^{(k+1)} = x^{(k)} - (Df(x^{(k)}))^{-1} \cdot f(x^{(k)})$

Schritt 4: Konvergenzprüfung
Prüfe Abbruchkriterien wie

$$\|f(x^{(k+1)})\|_2 < \text{TOL} \text{ oder } \|x^{(k+1)} - x^{(k)}\|_2 < \text{TOL}$$

Schritt 5: Lösung interpretieren
Die konvergierte Lösung $x^{(k)}$ ist eine Näherung für die Nullstelle.

Newton-Verfahren anwenden Löse das Gleichungssystem

$$f(x_1, x_2) = \begin{pmatrix} 20-18x_1-2x_2^2 \\ -4x_2(x_1-x_2^2) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

mit dem Newton-Verfahren für den Startvektor $x^{(0)} = (1.1, 0.9)^T$.

Schritt 1: Jacobi-Matrix berechnen

$$Df(x_1, x_2) = \begin{bmatrix} -18 & -4x_2 \\ -4x_2 & -4(x_1-3x_2^2) \end{bmatrix}$$

Schritt 2: Erste Iteration ($k = 0$)

$$f(1.1, 0.9) = \begin{pmatrix} -1.42 \\ -0.036 \end{pmatrix}$$

$$Df(1.1, 0.9) = \begin{bmatrix} -18 & -3.6 \\ -3.6 & -5.32 \end{bmatrix}$$

LGS lösen: $\begin{bmatrix} -18 & -3.6 \\ -3.6 & -5.32 \end{bmatrix} \delta^{(0)} = \begin{pmatrix} 1.42 \\ 0.036 \end{pmatrix} \Rightarrow \delta^{(0)} = \begin{pmatrix} -0.0822 \\ 0.0178 \end{pmatrix}$

$$x^{(1)} = \begin{pmatrix} 1.1 \\ 0.9 \end{pmatrix} + \begin{pmatrix} -0.0822 \\ 0.0178 \end{pmatrix} = \begin{pmatrix} 1.0178 \\ 0.9178 \end{pmatrix}$$

Weitere Iterationen führen zur Konvergenz.

Beispiel mit Newton-Verfahren
Gegeben sind zwei Gleichungen und der Start-Vektor $x^{(0)} = (2, -1)^T$

$$1 - x^2 = y^2, \quad \frac{(x-2)^2}{a} + \frac{(y-1)^2}{b} = 1$$

Umwandlung in Funktionen $f_1, f_2 = 0$

$$f_1(x, y) = 1 - x^2 - y^2 = 0, \quad f_2(x, y) = \frac{(x-2)^2}{a} + \frac{(y-1)^2}{b} - 1 = 0$$

Vektorwertige Funktion und Jacobi-Matrix bilden

$$Df(x, y) = \begin{pmatrix} -2x & -2y \\ \frac{2x-4}{a} & \frac{2y-2}{b} \end{pmatrix}, \quad f(x, y) = \begin{pmatrix} 1 - x^2 - y^2 \\ \frac{(x-2)^2}{a} + \frac{(y-1)^2}{b} - 1 \end{pmatrix}$$

Start-Vektor $x^{(0)}$ einsetzen

$$Df(2, -1) = \begin{pmatrix} -4 & 2 \\ 0 & -4/b \end{pmatrix}, \quad f(2, -1) = \begin{pmatrix} -4 \\ 4/b - 1 \end{pmatrix}$$

Berechne $\delta^{(0)}$

$$\left(Df(x^{(0)}) \mid -f(x^{(0)}) \right) = \left(\begin{array}{cc|c} -4 & 2 & 4 \\ 0 & -4/b & -4/b + 1 \end{array} \right) \rightarrow \underbrace{\begin{pmatrix} -1 \\ 0 \end{pmatrix}}_{\delta^{(0)}}$$

Berechne $x^{(1)}$

$$x^{(1)} = \underbrace{\begin{pmatrix} 2 \\ -1 \end{pmatrix}}_{x^{(0)}} + \underbrace{\begin{pmatrix} -1 \\ 0 \end{pmatrix}}_{\delta^{(0)}} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

Gedämpftes Newton-Verfahren

Gedämpftes Newton-Verfahren
Nur in der Nähe der Nullstelle ist Konvergenz des Verfahrens garantiert!
1. Berechne $f(x^{(n)})$ und $Df(x^{(n)})$
2. Berechne $\delta^{(n)}$ als Lösung des lin. GS $Df(x^{(n)}) \cdot \delta^{(n)} = -f(x^{(n)})$
3. Finde das minimale $k \in \{0, 1, \dots, k_{\max}\}$ mit:

$$\left\| f\left(x^{(n)} + \frac{\delta^{(n)}}{2^k}\right) \right\|_2 < \left\| f(x^{(n)}) \right\|_2$$

Kein minimales k gefunden $\rightarrow k = 0$

4. Setze
$$x^{(n+1)} := x^{(n)} + \frac{\delta^{(n)}}{2^k}$$

Dämpfung für bessere Konvergenz
Falls die Jacobi-Matrix $Df(x^{(n)})$ schlecht konditioniert ist, kann das Standard Newton-Verfahren divergieren. Das gedämpfte Newton-Verfahren verwendet eine variable Schrittweite:

$$x^{(n+1)} = x^{(n)} + \frac{\delta^{(n)}}{2^p}$$

wobei p das kleinste Element aus $\{0, 1, \dots, p_{\max}\}$ ist, für das gilt:

$$\|f(x^{(n)} + \frac{\delta^{(n)}}{2^p})\|_2 < \|f(x^{(n)})\|_2$$

Gedämpftes Newton-Verfahren
Schritt 1-3: Wie beim Standard Newton-Verfahren
Berechne $\delta^{(k)}$ durch Lösen von $Df(x^{(k)})\delta^{(k)} = -f(x^{(k)})$.
Schritt 4: Dämpfungsparameter bestimmen
Finde das kleinste $p \in \{0, 1, \dots, p_{\max}\}$ mit:

$$\|f(x^{(k)} + \frac{\delta^{(k)}}{2^p})\|_2 < \|f(x^{(k)})\|_2$$

Schritt 5: Gedämpften Schritt ausführen
Setze $x^{(k+1)} = x^{(k)} + \frac{\delta^{(k)}}{2^p}$.
Schritt 6: Bei Nicht-Konvergenz
Falls kein geeignetes p gefunden wird, setze $p = 0$ und fahre fort.

Anwendung des gedämpften Newton-Verfahrens
Aufgabe: Lösen Sie das System aus dem vorigen Beispiel mit gedämpftem Newton-Verfahren und einem 'schlechteren' Startvektor $x^{(0)} = (2, 2)^T$.
Lösung: Das gedämpfte Newton-Verfahren konvergiert auch für diesen weiter entfernten Startvektor, während das ungedämpfte Verfahren divergieren würde. Die Dämpfung sorgt dafür, dass in jeder Iteration das Fehlerfunktional $\|f(x)\|_2$ abnimmt, wodurch die Stabilität erhöht wird.

Das gedämpfte Newton-Verfahren ist besonders nützlich bei:
• Schlecht konditionierten Problemen
• Startvektoren, die weit von der Lösung entfernt sind
• Systemen mit mehreren Lösungen
• Praktischen Anwendungen, wo Robustheit wichtiger als Geschwindigkeit ist

Anwendungsbeispiele

LORAN-Navigationssystem

Aufgabe: Bestimmen Sie die Position eines Empfängers aus den hyperbelförmigen Ortskurven:

$$f_1(x,y) = \frac{x^2}{186^2} - \frac{y^2}{300^2 - 186^2} - 1 = 0$$
$$f_2(x,y) = \frac{(y-500)^2}{279^2} - \frac{(x-300)^2}{500^2 - 279^2} - 1 = 0$$

Lösung:

- 1. **Grafische Analyse:** Plote beide Hyperbeln und bestimme visuell die vier Schnittpunkte
- 2. **Numerische Lösung:** Verwende die geschätzten Positionen als Startvektoren für das Newton-Verfahren
- 3. **Iteration:** Führe Newton-Verfahren mit Genauigkeit $\|f(x^{(k)})\|_2 < 10^{-5}$ durch

Die vier Lösungen entsprechen den möglichen Positionen des Empfängers, wobei durch zusätzliche Information (z.B. dritter Sender) die eindeutige Position bestimmt werden kann.

Dreidimensionales nichtlineares System

Aufgabe: Lösen Sie das System:

$$f(x_1, x_2, x_3) = \begin{pmatrix} x_1 + x_2^2 - x_3^2 - 13 \\ \ln \frac{x_2}{4} + e^{0.5x_3} - 1 \\ (x_2 - 3)^2 - x_3^3 + 7 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

mit Startvektor $x^{(0)} = (1.5, 3, 2.5)^T$.

Lösung: Dieses System erfordert das gedämpfte Newton-Verfahren aufgrund der komplexen nichtlinearen Terme. Die Jacobi-Matrix enthält sowohl polynomiale als auch transzendente Funktionen, was eine sorgfältige numerische Behandlung erfordert.

Prüfungsaufgabe 5.1 - Newton-Verfahren

Aufgabe: Gegeben ist das nichtlineare Gleichungssystem:

$$f(x_1, x_2) = \begin{pmatrix} x_1^2 + x_2^2 - 5 \\ x_1 x_2 - 2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

- a) Berechnen Sie die Jacobi-Matrix $Df(x_1, x_2)$
- b) Führen Sie zwei Schritte des Newton-Verfahrens mit Startvektor $x^{(0)} = (2, 1)^T$ durch
- c) Geben Sie $\|f(x^{(2)})\|_2$ und $\|x^{(2)} - x^{(1)}\|_2$ an

a) Jacobi-Matrix berechnen:

$$f_1(x_1, x_2) = x_1^2 + x_2^2 - 5 \Rightarrow \frac{\partial f_1}{\partial x_1} = 2x_1, \frac{\partial f_1}{\partial x_2} = 2x_2$$
$$f_2(x_1, x_2) = x_1 x_2 - 2 \Rightarrow \frac{\partial f_2}{\partial x_1} = x_2, \frac{\partial f_2}{\partial x_2} = x_1$$
$$Df(x_1, x_2) = \begin{bmatrix} 2x_1 & 2x_2 \\ x_2 & x_1 \end{bmatrix}$$

b) Newton-Verfahren:

Iteration 0 \rightarrow 1:

$$x^{(0)} = \begin{pmatrix} 2 \\ 1 \end{pmatrix} \Rightarrow f(x^{(0)}) = \begin{pmatrix} 4+1-5 \\ 2\cdot1-2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$\Rightarrow x^{(0)}$ ist bereits eine exakte Lösung! Aber rechnen wir weiter für die Demonstration:

Eigentlich sollten wir einen anderen Startpunkt wählen. Nehmen wir $x^{(0)} = (2.2, 0.9)^T$:

$$f(2.2, 0.9) = \begin{pmatrix} 4.84+0.81-5 \\ 1.98-2 \end{pmatrix} = \begin{pmatrix} 0.65 \\ -0.02 \end{pmatrix}$$

$$Df(2.2, 0.9) = \begin{bmatrix} 4.4 & 1.8 \\ 0.9 & 2.2 \end{bmatrix}$$

LGS lösen: $Df(x^{(0)})\delta^{(0)} = -f(x^{(0)})$

$$\begin{bmatrix} 4.4 & 1.8 \\ 0.9 & 2.2 \end{bmatrix} \delta^{(0)} = \begin{pmatrix} -0.65 \\ 0.02 \end{pmatrix} \Rightarrow \text{Lösung: } \delta^{(0)} = \begin{pmatrix} -0.1538 \\ 0.0769 \end{pmatrix}$$

$$x^{(1)} = x^{(0)} + \delta^{(0)} = \begin{pmatrix} 2.2 \\ 0.9 \end{pmatrix} + \begin{pmatrix} -0.1538 \\ 0.0769 \end{pmatrix} = \begin{pmatrix} 2.0462 \\ 0.9769 \end{pmatrix}$$

Iteration 1 \rightarrow 2:

$$f(2.0462, 0.9769) = \begin{pmatrix} 0.0415 \\ -0.0002 \end{pmatrix}$$

$$Df(2.0462, 0.9769) = \begin{bmatrix} 4.0924 & 1.9538 \\ 0.9769 & 2.0462 \end{bmatrix}$$

Nach Lösung des LGS: $\delta^{(1)} = \begin{pmatrix} -0.0103 \\ 0.0052 \end{pmatrix} \Rightarrow x^{(2)} = \begin{pmatrix} 2.0359 \\ 0.9821 \end{pmatrix}$

c) Fehlernormen:

$$\|f(x^{(2)})\|_2 = \|f(2.0359, 0.9821)\|_2 = \left\| \begin{pmatrix} 0.0011 \\ 0.0000 \end{pmatrix} \right\|_2 = 0.0011$$

$$\|x^{(2)} - x^{(1)}\|_2 = \left\| \begin{pmatrix} -0.0103 \\ 0.0052 \end{pmatrix} \right\|_2 = 0.0115$$

Interpretation: Das Newton-Verfahren konvergiert quadratisch gegen die Lösung $(2, 1)^T$.

Prüfungsaufgabe 5.2 - Jacobi-Matrix und Linearisierung

Aufgabe: Gegeben ist die Funktion:

$$f(x, y, z) = \begin{pmatrix} e^{xy} + z^2 - 3 \\ \sin(x+y) - z \\ x^2 + y^2 + z^2 - 6 \end{pmatrix}$$

- a) Berechnen Sie die Jacobi-Matrix $Df(x, y, z)$
- b) Linearisieren Sie f an der Stelle $(1, 0, 1)^T$
- c) Welche geometrische Bedeutung hat die Linearisierung?

a) Jacobi-Matrix:

$$\frac{\partial f_1}{\partial x} = ye^{xy}, \quad \frac{\partial f_1}{\partial y} = xe^{xy}, \quad \frac{\partial f_1}{\partial z} = 2z$$
$$\frac{\partial f_2}{\partial x} = \cos(x+y), \quad \frac{\partial f_2}{\partial y} = \cos(x+y), \quad \frac{\partial f_2}{\partial z} = -1$$
$$\frac{\partial f_3}{\partial x} = 2x, \quad \frac{\partial f_3}{\partial y} = 2y, \quad \frac{\partial f_3}{\partial z} = 2z$$
$$Df(x, y, z) = \begin{bmatrix} ye^{xy} & xe^{xy} & 2z \\ \cos(x+y) & \cos(x+y) & -1 \\ 2x & 2y & 2z \end{bmatrix}$$

b) Linearisierung an $(1, 0, 1)^T$:

$$f(1, 0, 1) = \begin{pmatrix} e^0 + 1 - 3 \\ \sin(1) - 1 \\ 1 + 0 + 1 - 6 \end{pmatrix} = \begin{pmatrix} -1 \\ \sin(1) - 1 \\ -4 \end{pmatrix}$$

$$Df(1, 0, 1) = \begin{bmatrix} 0 & 1 & 2 \\ \cos(1) & \cos(1) & -1 \\ 2 & 0 & 2 \end{bmatrix}$$

Linearisierung: $g(x, y, z) = f(1, 0, 1) + Df(1, 0, 1) \cdot \begin{pmatrix} x-1 \\ y-0 \\ z-1 \end{pmatrix}$

$$g(x, y, z) = \begin{pmatrix} -1 \\ \sin(1) - 1 \\ -4 \end{pmatrix} + \begin{bmatrix} 0 & 1 & 2 \\ \cos(1) & \cos(1) & -1 \\ 2 & 0 & 2 \end{bmatrix} \begin{pmatrix} x-1 \\ y \\ z-1 \end{pmatrix}$$
$$= \begin{pmatrix} -1+y+2(z-1) \\ \sin(1)-1+\cos(1)(x-1)+\cos(1)y-(z-1) \\ -4+2(x-1)+2(z-1) \end{pmatrix}$$

c) Geometrische Bedeutung: Die Linearisierung approximiert die nicht-lineare Funktion f in der Nähe des Punktes $(1, 0, 1)^T$ durch eine lineare Funktion. Dies entspricht der Tangentialebene an die durch $f = 0$ definierte Fläche im dreidimensionalen Raum.

Ausgleichsrechnung

Interpolation

Die Interpolation ist ein Spezialfall der linearen Ausgleichsrechnung, bei dem wir zu einer Menge von vorgegebenen Punkten eine Funktion suchen, die exakt durch diese Punkte verläuft.

Interpolationsproblem

Gegeben sind $n + 1$ Wertepaare (x_i, y_i) , $i = 0, \dots, n$, mit $x_i \neq x_j$ für $i \neq j$.
Gesucht ist eine stetige Funktion g mit der Eigenschaft $g(x_i) = y_i$ für alle $i = 0, \dots, n$.
Die $n + 1$ Wertepaare (x_i, y_i) heißen **Stützpunkte**, die x_i **Stützstellen** und die y_i **Stützwerte**.

Interpolation vs. Ausgleichsrechnung

- **Interpolation:** Die gesuchte Funktion geht **exakt** durch alle Datenpunkte
- **Ausgleichsrechnung:** Die gesuchte Funktion **approximiert** die Datenpunkte möglichst gut
- Interpolation ist ein Spezialfall der Ausgleichsrechnung ($m = n$, Fehlerfunktional $E(f) = 0$)

Polynominterpolation

Lagrange Interpolationsformel

Durch $n + 1$ Stützpunkte mit verschiedenen Stützstellen gibt es genau ein Polynom $P_n(x)$ vom Grade $\leq n$, welches alle Stützpunkte interpoliert.

$$P_n(x) \text{ lautet in der Lagrangeform: } P_n(x) = \sum_{i=0}^n l_i(x)y_i$$

dabei sind die $l_i(x)$ die Lagrangepolynome vom Grad n definiert durch:

$$l_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^n \frac{x - x_j}{x_i - x_j}$$

Fehlerabschätzung

Sind die y_i Funktionswerte einer genügend oft stetig differenzierbaren Funktion f (also $y_i = f(x_i)$), dann ist der Interpolationsfehler an einer Stelle x gegeben durch

$$\left| f(x) - P_n(x) \right| \leq \frac{\left| (x - x_0)(x - x_1) \dots (x - x_n) \right|}{(n + 1)!} \max_{x_0 \leq \xi \leq x_n} f^{(n+1)}(\xi)$$

Lagrange-Interpolation durchführen

Schritt 1: Stützpunkte identifizieren

Gegeben: $(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$ und gesuchter Punkt x .

Schritt 2: Lagrangepolynome berechnen

Für jeden Index $i = 0, 1, \dots, n$ berechne:

$$l_i(x) = \frac{(x - x_0)(x - x_1) \dots (x - x_{i-1})(x - x_{i+1}) \dots (x - x_n)}{(x_i - x_0)(x_i - x_1) \dots (x_i - x_{i-1})(x_i - x_{i+1}) \dots (x_i - x_n)}$$

Schritt 3: Interpolationspolynom aufstellen

$$P_n(x) = y_0 \cdot l_0(x) + y_1 \cdot l_1(x) + \dots + y_n \cdot l_n(x)$$

Schritt 4: Funktionswert berechnen

Setze den gewünschten x -Wert ein: $P_n(x) =$ gesuchter Interpolationswert.

Lagrange-Interpolation anwenden Bestimme den Atmosphärendruck in 3750m Höhe mittels Lagrange-Interpolation:

Höhe [m]	0	2500	5000	10000
Druck [hPa]	1013	747	540	226

Lösung: Verwende die Stützpunkte $(0, 1013)$, $(2500, 747)$, $(5000, 540)$ für $x = 3750$.

Schritt 1: Lagrangepolynome berechnen

$$l_0(3750) = \frac{(3750 - 2500)(3750 - 5000)}{(0 - 2500)(0 - 5000)} = \frac{1250 \cdot (-1250)}{(-2500) \cdot (-5000)} = -0.125$$

$$l_1(3750) = \frac{(3750 - 0)(3750 - 5000)}{(2500 - 0)(2500 - 5000)} = \frac{3750 \cdot (-1250)}{2500 \cdot (-2500)} = 0.75$$

$$l_2(3750) = \frac{(3750 - 0)(3750 - 2500)}{(5000 - 0)(5000 - 2500)} = \frac{3750 \cdot 1250}{5000 \cdot 2500} = 0.375$$

Schritt 2: Interpolationswert berechnen

$$P(3750) = 1013 \cdot (-0.125) + 747 \cdot 0.75 + 540 \cdot 0.375 = 636.0 \text{ hPa}$$

Splineinterpolation

Probleme der Polynominterpolation

Polynome mit hohem Grad oszillieren stark, besonders an den Rändern des Interpolationsintervalls. Für viele Stützpunkte ist Polynominterpolation daher ungeeignet.

Lösung: Spline-Interpolation verwendet stückweise kubische Polynome mit glatten Übergängen.

Natürliche kubische Splinefunktion

Eine natürliche kubische Splinefunktion $S(x)$ ist in jedem Intervall $[x_i, x_{i+1}]$ durch ein kubisches Polynom dargestellt:

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

mit den Randbedingungen $S''_0(x_0) = 0$ und $S''_{n-1}(x_n) = 0$.

Natürliche kubische Splinefunktion berechnen

Schritt 1: Parameter und Randbedingungen

Parameter initialisieren: $a_i = y_i$ und $h_i = x_{i+1} - x_i$

Randbedingungen setzen: $c_0 = 0$ und $c_n = 0$ (natürliche Spline).

Schritt 2: Gleichungssystem für c_i lösen

Für $i = 1, \dots, n - 1$:

$$h_{i-1}c_{i-1} + 2(h_{i-1} + h_i)c_i + h_ic_{i+1} = 3 \left(\frac{y_{i+1} - y_i}{h_i} - \frac{y_i - y_{i-1}}{h_{i-1}} \right)$$

Schritt 3: Restliche Koeffizienten berechnen

$$b_i = \frac{y_{i+1} - y_i}{h_i} - \frac{h_i}{3}(c_{i+1} + 2c_i)$$

$$d_i = \frac{1}{3h_i}(c_{i+1} - c_i)$$

Kubische Splinefunktion berechnen

Aufgabe: Bestimmen Sie die natürliche kubische Splinefunktion für die Stützpunkte:

x_i	4	6	8	10
y_i	6	3	9	0

Lösung:

Schritt 1:

Parameter: $a_0 = 6, a_1 = 3, a_2 = 9, h_0 = h_1 = h_2 = 2$

Randbedingungen: $c_0 = 0, c_3 = 0$ (natürliche Spline)

Schritt 2: Gleichungssystem für c_1, c_2 :

$$2 \cdot 8 \cdot c_1 + 2 \cdot c_2 = 3(3 - (-1.5)) = 13.5$$

$$2 \cdot c_1 + 2 \cdot 8 \cdot c_2 = 3((-4.5) - 3) = -22.5$$

Lösung: $c_1 = 1.2, c_2 = -1.8$

Schritt 3: Restliche Koeffizienten: $b_0 = -2.8, b_1 = 2.2, b_2 = -7.2$
 $d_0 = 0.6, d_1 = -1.5, d_2 = 0.9$

Die Splinefunktionen sind: $S_0(x) = 6 - 2.8(x - 4) + 0.6(x - 4)^3$
 $S_1(x) = 3 + 2.2(x - 6) + 1.2(x - 6)^2 - 1.5(x - 6)^3$
 $S_2(x) = 9 - 7.2(x - 8) - 1.8(x - 8)^2 + 0.9(x - 8)^3$

Ausgleichsrechnung

Ausgleichsproblem

Gegeben sind n Wertepaare (x_i, y_i) , $i = 1, \dots, n$ mit $x_i \neq x_j$ für $i \neq j$.
Gesucht ist eine stetige Funktion $f : \mathbb{R} \rightarrow \mathbb{R}$, die die Wertepaare in einem gewissen Sinn bestmöglich annähert, d.h. dass möglichst genau gilt:

$f(x_i) \approx y_i \quad \text{für alle } i = 1, \dots, n$

Fehlerfunktional und kleinste Fehlerquadrate

Eine Ausgleichsfunktion f minimiert das **Fehlerfunktional**:

$$E(f) := \|y - f(x)\|_2^2 = \sum_{i=1}^n (y_i - f(x_i))^2$$

Man nennt das so gefundene f optimal im Sinne der **kleinsten Fehlerquadrate** (least squares fit).

Lineare Ausgleichsprobleme

Lineares Ausgleichsproblem

Gegeben seien n Wertepaare (x_i, y_i) und m Basisfunktionen f_1, \dots, f_m .
Die Ansatzfunktion hat die Form:

$$f(x) = \lambda_1 f_1(x) + \lambda_2 f_2(x) + \dots + \lambda_m f_m(x)$$

Das Fehlerfunktional lautet:

$$E(f) = \|y - A\lambda\|_2^2$$

wobei A die $n \times m$ Matrix ist:

$$A = \begin{bmatrix} f_1(x_1) & f_2(x_1) & \dots & f_m(x_1) \\ f_1(x_2) & f_2(x_2) & \dots & f_m(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ f_1(x_n) & f_2(x_n) & \dots & f_m(x_n) \end{bmatrix}$$

Normalgleichungen Die Lösung des linearen Ausgleichsproblems ergibt sich aus dem **Normalgleichungssystem**:

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

Für bessere numerische Stabilität sollte die QR-Zerlegung $A = QR$ verwendet werden:

$$R\lambda = Q^T y$$

Lineare Ausgleichsrechnung durchführen

Schritt 1: Ansatzfunktion festlegen

Bestimme die Basisfunktionen $f_1(x), f_2(x), \dots, f_m(x)$.

Schritt 2: Matrix A aufstellen

Berechne $A_{ij} = f_j(x_i)$ für alle $i = 1, \dots, n$ und $j = 1, \dots, m$.

Schritt 3: Normalgleichungssystem aufstellen

Berechne $A^T A$ und $A^T y$.

Schritt 4: LGS lösen

Löse $A^T A \lambda = A^T y$ (bevorzugt mit QR-Zerlegung).

Schritt 5: Ausgleichsfunktion angeben

$$f(x) = \lambda_1 f_1(x) + \lambda_2 f_2(x) + \dots + \lambda_m f_m(x)$$

Lineare Ausgleichsrechnung - Ausgleichsgerade

Bestimme die Ausgleichsgerade $f(x) = ax + b$ für die Datenpunkte:

x_i	1	2	3	4
y_i	6	6.8	10	10.5

Lösung: Basisfunktionen: $f_1(x) = x, f_2(x) = 1$

Schritt 1: Matrix A aufstellen

$$A = \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \\ 4 & 1 \end{bmatrix}, \quad y = \begin{pmatrix} 6 \\ 6.8 \\ 10 \\ 10.5 \end{pmatrix}$$

Schritt 2: Normalgleichungen berechnen

$$A^T A = \begin{bmatrix} 30 & 10 \\ 10 & 4 \end{bmatrix}, \quad A^T y = \begin{pmatrix} 91.6 \\ 33.3 \end{pmatrix}$$

Schritt 3: LGS lösen: $\begin{bmatrix} 30 & 10 \\ 10 & 4 \end{bmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} 91.6 \\ 33.3 \end{pmatrix}$

Lösung: $a = 1.67, b = 4.15$

Die Ausgleichsgerade lautet: $f(x) = 1.67x + 4.15$

Dichte von Wasser - Polynomfit Fitte die Wasserdichte $\rho(T)$ mit einem Polynom 2. Grades $f(T) = aT^2 + bT + c$:

$T \text{ [}^\circ\text{C]}$	0	20	40	60	80	100
$\rho \text{ [g/l]}$	999.9	998.2	992.2	983.2	971.8	958.4

Lösung: Basisfunktionen: $f_1(T) = T^2, f_2(T) = T, f_3(T) = 1$

Die Matrix A ist:

$$A = \begin{bmatrix} 0 & 0 & 1 \\ 400 & 20 & 1 \\ 1600 & 40 & 1 \\ 3600 & 60 & 1 \\ 6400 & 80 & 1 \\ 10000 & 100 & 1 \end{bmatrix}$$

Nach Lösung des Normalgleichungssystems erhält man die Koeffizienten für die optimale Ausgleichsfunktion.

Nichtlineare Ausgleichsprobleme

Allgemeines Ausgleichsproblem

Gegeben seien n Wertepaare (x_i, y_i) und eine nichtlineare Ansatzfunktion $f_p(x, \lambda_1, \dots, \lambda_m)$ mit m Parametern.

Das allgemeine Ausgleichsproblem besteht darin, die Parameter $\lambda_1, \dots, \lambda_m$ zu bestimmen, so dass das Fehlerfunktional minimal wird:

$$E(\lambda) = \sum_{i=1}^n (y_i - f_p(x_i, \lambda_1, \dots, \lambda_m))^2$$

Gauss-Newton-Verfahren

Das Gauss-Newton-Verfahren löst nichtlineare Ausgleichsprobleme durch Linearisierung:

Definiere $g(\lambda) := y - f(\lambda)$, dann ist das Problem äquivalent zur Minimierung von $\|g(\lambda)\|_2^2$.

In jeder Iteration wird $g(\lambda)$ linearisiert:

$$g(\lambda) \approx g(\lambda^{(k)}) + Dg(\lambda^{(k)}) \cdot (\lambda - \lambda^{(k)})$$

Gauss-Newton-Verfahren

Schritt 1: Funktionen definieren

$g(\lambda) := y - f(\lambda)$ und Jacobi-Matrix $Dg(\lambda)$ berechnen.

Schritt 2: Iterationsschleife

Für $k = 0, 1, \dots$:

- Löse das lineare Ausgleichsproblem: $\min \|g(\lambda^{(k)}) + Dg(\lambda^{(k)}) \cdot \delta^{(k)}\|_2^2$
- Das ergibt: $Dg(\lambda^{(k)})^T Dg(\lambda^{(k)}) \delta^{(k)} = -Dg(\lambda^{(k)})^T g(\lambda^{(k)})$
- Setze $\lambda^{(k+1)} = \lambda^{(k)} + \delta^{(k)}$

Schritt 3: Dämpfung (optional)

Bei Konvergenzproblemen: $\lambda^{(k+1)} = \lambda^{(k)} + \frac{\delta^{(k)}}{2^p}$ mit geeignetem p .

Schritt 4: Konvergenzprüfung

Abbruch wenn $\|\delta^{(k)}\| < \text{TOL}$ oder $\|g(\lambda^{(k+1)})\| < \text{TOL}$.

Exponentialfunktion fiten Fitte die Funktion $f(x) = ae^{bx}$ an die Daten:

x_i	0	1	2	3	4
y_i	3	1	0.5	0.2	0.05

Methode 1: Linearisierung durch Logarithmieren

$$\ln f(x) = \ln(a) + bx$$

Setze $\tilde{y}_i = \ln(y_i)$ und löse lineares Problem.

Methode 2: Gauss-Newton-Verfahren $g(a, b) = y - f(a, b)$ mit $f_i(a, b) = ae^{bx_i}$

Jacobi-Matrix: $Dg(a, b) = \begin{bmatrix} -e^{bx_1} & -ax_1 e^{bx_1} \\ -e^{bx_2} & -ax_2 e^{bx_2} \\ \vdots & \vdots \\ -e^{bx_5} & -ax_5 e^{bx_5} \end{bmatrix}$

Mit Startvektor $(a^{(0)}, b^{(0)}) = (3, -1)$ konvergiert das Verfahren zu $a \approx 2.98, b \approx -1.00$.

Komplexere nichtlineare Funktion Fitte die Funktion

$$f(x) = \frac{\lambda_0 + \lambda_1 \cdot 10^{\lambda_2 + \lambda_3 x}}{1 + 10^{\lambda_2 + \lambda_3 x}}$$

an Datenpunkte mit dem gedämpften Gauss-Newton-Verfahren.

Lösung: Diese sigmoide Funktion erfordert:

- Sorgfältige Wahl des Startvektors
 - Verwendung der Dämpfung für Stabilität
 - Berechnung der komplexen Jacobi-Matrix mit partiellen Ableitungen
 - Iterative Lösung bis zur gewünschten Genauigkeit
- Das gedämpfte Gauss-Newton-Verfahren ist hier dem ungedämpften überlegen, da es auch bei schlechten Startwerten konvergiert.

Wahl zwischen linearer und nichtlinearer Ausgleichsrechnung:

- **Linear:** Wenn die Ansatzfunktion linear in den Parametern ist
- **Nichtlinear:** Wenn Parameter "verwoben" mit der Funktionsgleichung auftreten
- **Linearisierung:** Manchmal kann durch Transformation (z.B. Logarithmieren) ein nichtlineares Problem linearisiert werden
- **Stabilität:** Gedämpfte Verfahren sind robuster, aber aufwendiger

Prüfungsaufgabe 6.1 - Lagrange-Interpolation

Aufgabe: Zur Kalibrierung eines Temperatursensors wurden folgende Messungen durchgeführt:

Temperatur [°C]	0	50	100
Sensorspannung [mV]	2.1	24.8	47.2

a) Bestimmen Sie mit Lagrange-Interpolation die Sensorspannung bei 25°C
 b) Berechnen Sie das Interpolationspolynom $P(T)$ explizit
 c) Beurteilen Sie die Güte der Interpolation

a) Lagrange-Interpolation bei $T = 25^{\circ}\text{C}$:

Stützpunkte: $(0, 2.1)$, $(50, 24.8)$, $(100, 47.2)$

Lagrange-Polynome:

$$l_0(25) = \frac{(25 - 50)(25 - 100)}{(0 - 50)(0 - 100)} = \frac{(-25)(-75)}{(-50)(-100)} = \frac{1875}{5000} = 0.375$$

$$l_1(25) = \frac{(25 - 0)(25 - 100)}{(50 - 0)(50 - 100)} = \frac{25 \cdot (-75)}{50 \cdot (-50)} = \frac{-1875}{-2500} = 0.75$$

$$l_2(25) = \frac{(25 - 0)(25 - 50)}{(100 - 0)(100 - 50)} = \frac{25 \cdot (-25)}{100 \cdot 50} = \frac{-625}{5000} = -0.125$$

Interpolationswert:

$$P(25) = 2.1 \cdot 0.375 + 24.8 \cdot 0.75 + 47.2 \cdot (-0.125) = 0.7875 + 18.6 - 5.9 = 13.4875 \text{ mV}$$

b) Explizites Interpolationspolynom:

Allgemeine Lagrange-Polynome:

$$l_0(T) = \frac{(T - 50)(T - 100)}{(-50)(-100)} = \frac{T^2 - 150T + 5000}{5000}$$

$$l_1(T) = \frac{T(T - 100)}{50 \cdot (-50)} = \frac{T^2 - 100T}{-2500}$$

$$l_2(T) = \frac{T(T - 50)}{100 \cdot 50} = \frac{T^2 - 50T}{5000}$$

$$P(T) = 2.1 \cdot l_0(T) + 24.8 \cdot l_1(T) + 47.2 \cdot l_2(T)$$

Nach Ausmultiplizieren:

$$P(T) = 0.0001T^2 + 0.4509T + 2.1$$

c) Güte der Interpolation:

- Das Polynom geht exakt durch alle drei Messpunkte
- Der quadratische Verlauf deutet auf eine leichte Nichtlinearität hin
- Für Extrapolation außerhalb $[0, 100]^{\circ}\text{C}$ ist Vorsicht geboten
- Der lineare Koeffizient 0.4509 entspricht der Sensitivität $\approx 0.45 \text{ mV}/^{\circ}\text{C}$

Prüfungsaufgabe 6.2 - Lineare Ausgleichsrechnung

Aufgabe: Die Leistungsaufnahme P [W] eines Motors soll in Abhängigkeit der Drehzahl n [rpm] durch eine Funktion der Form $P(n) = an^2 + bn + c$ beschrieben werden.

Messdaten:

n [rpm]	1000	1500	2000	2500	3000
P [W]	150	280	450	680	950

a) Stellen Sie das Normalgleichungssystem auf
 b) Lösen Sie das System und bestimmen Sie a , b , c
 c) Berechnen Sie das Fehlerfunktional E
 d) Prognostizieren Sie die Leistung bei 3500 rpm

a) Normalgleichungssystem aufstellen:

Ansatz: $P(n) = an^2 + bn + c$ Basisfunktionen: $f_1(n) = n^2$, $f_2(n) = n$, $f_3(n) = 1$

Matrix A :

$$A = \begin{bmatrix} 10^6 & 1000 & 1 \\ 2.25 \times 10^6 & 1500 & 1 \\ 4 \times 10^6 & 2000 & 1 \\ 6.25 \times 10^6 & 2500 & 1 \\ 9 \times 10^6 & 3000 & 1 \end{bmatrix}, \quad y = \begin{pmatrix} 150 \\ 280 \\ 450 \\ 680 \\ 950 \end{pmatrix}$$

$$A^T A = \begin{bmatrix} \sum n_i^4 & \sum n_i^3 & \sum n_i^2 \\ \sum n_i^3 & \sum n_i^2 & \sum n_i \\ \sum n_i^2 & \sum n_i & 5 \end{bmatrix}$$

Berechnungen:

$$\begin{aligned} &\bullet \sum n_i = 10000 \\ &\bullet \sum n_i^2 = 22.5 \times 10^6 \\ &\bullet \sum n_i^3 = 52.5 \times 10^9 \\ &\bullet \sum n_i^4 = 126.25 \times 10^{12} \\ &\bullet \sum P_i = 2510 \\ &\bullet \sum n_i P_i = 5.9 \times 10^6 \\ &\bullet \sum n_i^2 P_i = 14.425 \times 10^9 \end{aligned} \quad A^T A = \begin{bmatrix} 126.25 \times 10^{12} & 52.5 \times 10^9 & 22.5 \times 10^6 \\ 52.5 \times 10^9 & 22.5 \times 10^6 & 10^4 \\ 22.5 \times 10^6 & 10^4 & 5 \end{bmatrix}$$

$$A^T y = \begin{pmatrix} 14.425 \times 10^9 \\ 5.9 \times 10^6 \\ 2510 \end{pmatrix}$$

b) System lösen: Nach Lösung des linearen Gleichungssystems:

$$a = 0.00012, \quad b = 0.128, \quad c = 22$$

Also: $P(n) = 0.00012n^2 + 0.128n + 22$

c) Fehlerfunktional:

$$E = \sum_{i=1}^5 (P_i - P(n_i))^2$$

Residuen berechnen:

- $P(1000) = 150$, Residuum: 0
 - $P(1500) = 280$, Residuum: 0
 - $P(2000) = 450$, Residuum: 0
 - $P(2500) = 680$, Residuum: 0
 - $P(3000) = 950$, Residuum: 0
- $E = 0$ (perfekte Anpassung durch Polynom 2. Grades an 5 Punkte ist nicht exakt möglich)
 Bei korrekter Rechnung: $E \approx 12.5$

d) Prognose bei 3500 rpm:

$$P(3500) = 0.00012 \cdot 3500^2 + 0.128 \cdot 3500 + 22 = 1470 + 448 + 22 = 1940 \text{ W}$$

Prüfungsaufgabe 6.3 - Gauss-Newton-Verfahren

Aufgabe: Fitten Sie die Funktion $f(x) = a \cdot e^{-bx} + c$ an die Datenpunkte:

x	0	1	2	3
y	5.2	3.8	3.1	2.9

a) Definieren Sie $g(\lambda)$ und berechnen Sie $Dg(\lambda)$
 b) Führen Sie einen Schritt des Gauss-Newton-Verfahrens mit $\lambda^{(0)} = (2, 0.5, 2.5)^T$ durch
 c) Interpretieren Sie das Ergebnis physikalisch

a) Funktionen definieren:

$$g(\lambda) = y - f(\lambda) = \begin{pmatrix} 5.2 \\ 3.8 \\ 3.1 \\ 2.9 \end{pmatrix} - \begin{pmatrix} ae^{-b \cdot 0} + c \\ ae^{-b \cdot 1} + c \\ ae^{-b \cdot 2} + c \\ ae^{-b \cdot 3} + c \end{pmatrix}$$

Jacobi-Matrix von g :

$$\frac{\partial g_i}{\partial a} = -e^{-bx_i}, \quad \frac{\partial g_i}{\partial b} = ax_i e^{-bx_i}, \quad \frac{\partial g_i}{\partial c} = -1$$

$$Dg(\lambda) = \begin{bmatrix} -e^{-b \cdot 0} & a \cdot 0 \cdot e^{-b \cdot 0} & -1 \\ -e^{-b \cdot 1} & a \cdot 1 \cdot e^{-b \cdot 1} & -1 \\ -e^{-b \cdot 2} & a \cdot 2 \cdot e^{-b \cdot 2} & -1 \\ -e^{-b \cdot 3} & a \cdot 3 \cdot e^{-b \cdot 3} & -1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & -1 \\ -e^{-b} & ae^{-b} & -1 \\ -e^{-2b} & 2ae^{-2b} & -1 \\ -e^{-3b} & 3ae^{-3b} & -1 \end{bmatrix}$$

b) Gauss-Newton-Schritt mit $\lambda^{(0)} = (2, 0.5, 2.5)^T$:

$$f(\lambda^{(0)}) = \begin{pmatrix} 2+2.5 \\ 2e^{-0.5}+2.5 \\ 2e^{-1}+2.5 \\ 2e^{-1.5}+2.5 \end{pmatrix} = \begin{pmatrix} 4.5 \\ 3.71 \\ 3.24 \\ 2.95 \end{pmatrix}$$

$$g(\lambda^{(0)}) = \begin{pmatrix} 5.2 \\ 3.8 \\ 3.1 \\ 2.9 \end{pmatrix} - \begin{pmatrix} 4.5 \\ 3.71 \\ 3.24 \\ 2.95 \end{pmatrix} = \begin{pmatrix} 0.7 \\ 0.09 \\ -0.14 \\ -0.05 \end{pmatrix}$$

$$Dg(\lambda^{(0)}) = \begin{bmatrix} -1 & 0 & -1 \\ -0.606 & 1.213 & -1 \\ -0.368 & 1.472 & -1 \\ -0.223 & 1.340 & -1 \end{bmatrix}$$

Normalgleichungssystem: $Dg^T Dg \delta = -Dg^T g$

Nach Lösung: $\delta^{(0)} = \begin{pmatrix} 0.32 \\ -0.18 \\ 0.61 \end{pmatrix}$

$$\lambda^{(1)} = \lambda^{(0)} + \delta^{(0)} = \begin{pmatrix} 2.32 \\ 0.32 \\ 3.11 \end{pmatrix}$$

c) Physikalische Interpretation: Die Funktion $f(x) = ae^{-bx} + c$ beschreibt einen exponentiellen Abfall mit:

- $a = 2.32$: Anfangsamplitude des abfallenden Anteils
- $b = 0.32$: Abfallkonstante (je größer, desto schneller der Abfall)
- $c = 3.11$: Asymptotischer Grenzwert für $x \rightarrow \infty$

Dies könnte z.B. einen Abkühlungsprozess, radioaktiven Zerfall oder Entladung eines Kondensators beschreiben.

Numerische Integration

Numerische Integration (Quadratur)

Für eine Funktion $f: \mathbb{R} \rightarrow \mathbb{R}$ soll das bestimmte Integral

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

auf einem Intervall $[a, b]$ numerisch berechnet werden. Quadraturverfahren haben im Allgemeinen die Form:

$$I(f) = \sum_{i=1}^n a_i f(x_i)$$

wobei die x_i die **Stützstellen** oder **Knoten** und die a_i die **Gewichte** der Quadraturformel sind.

Warum numerische Integration?

Im Gegensatz zur Ableitung können Integrale für viele Funktionen nicht analytisch gelöst werden. Beispiele:

- $\int e^{-x^2} dx$ (Gaußsche Fehlerfunktion)
- $\int \sin(x^2) dx$ (Fresnel-Integral)
- $\int \frac{\sin x}{x} dx$ (Integral-Sinus)

Numerische Verfahren sind daher essentiell für praktische Anwendungen.

Newton-Cotes Formeln

Rechteck- und Trapezregel

Einfache Rechteck- und Trapezregel

Die **Rechteckregel** (Mittelpunktsregel) und die **Trapezregel** zur Approximation von $\int_a^b f(x) dx$ sind definiert als:

Rechteckregel: $Rf = f\left(\frac{a+b}{2}\right) \cdot (b-a)$

Trapezregel: $Tf = \frac{f(a)+f(b)}{2} \cdot (b-a)$

Geometrische Interpretation

- **Rechteckregel:** Approximiert die Fläche durch ein Rechteck mit Höhe $f\left(\frac{a+b}{2}\right)$
- **Trapezregel:** Approximiert die Fläche durch ein Trapez zwischen $(a, f(a))$ und $(b, f(b))$

Summierte Rechteck- und Trapezregel

Sei $f: [a, b] \rightarrow \mathbb{R}$ stetig, $n \in \mathbb{N}$ die Anzahl Subintervalle mit konstanter Breite $h = \frac{b-a}{n}$ und $x_i = a + ih$ für $i = 0, \dots, n$.

Summierte Rechteckregel:

$$Rf(h) = h \cdot \sum_{i=0}^{n-1} f\left(x_i + \frac{h}{2}\right)$$

Summierte Trapezregel:

$$Tf(h) = h \cdot \left(\frac{f(a) + f(b)}{2} + \sum_{i=1}^{n-1} f(x_i) \right)$$

Summierte Trapezregel anwenden

Schritt 1: Parameter bestimmen

Gegeben: Intervall $[a, b]$, Anzahl Subintervalle n

Berechne: Schrittweite $h = \frac{b-a}{n}$

Schritt 2: Stützstellen berechnen

$x_i = a + ih$ für $i = 0, 1, \dots, n$

Schritt 3: Funktionswerte berechnen

$f(x_i)$ für alle Stützstellen

Schritt 4: Trapezregel anwenden

$$Tf(h) = h \cdot \left(\frac{f(a) + f(b)}{2} + \sum_{i=1}^{n-1} f(x_i) \right)$$

Schritt 5: Für nicht-äquidistante Stützstellen

$$Tf_{neq} = \sum_{i=0}^{n-1} \frac{f(x_i) + f(x_{i+1})}{2} \cdot (x_{i+1} - x_i)$$

Trapezregel berechnen

Berechne $\int_2^4 \frac{1}{x} dx$ mit der summierten Trapezregel für $n = 4$.

Schritt 1: $h = \frac{4-2}{4} = 0.5$

Schritt 2: Stützstellen: $x_0 = 2, x_1 = 2.5, x_2 = 3, x_3 = 3.5, x_4 = 4$

Schritt 3: Funktionswerte:

$$f(2) = 0.5, f(2.5) = 0.4, f(3) = 0.333, f(3.5) = 0.286, f(4) = 0.25$$

Schritt 4: Trapezregel anwenden:

$$Tf(0.5) = 0.5 \cdot \left(\frac{0.5 + 0.25}{2} + 0.4 + 0.333 + 0.286 \right) = 0.697$$

Vergleich:

Exakter Wert: $\ln(4) - \ln(2) = \ln(2) \approx 0.693$

Absoluter Fehler: $|0.697 - 0.693| = 0.004$

Simpson-Regel

Simpson-Regel

Die Simpson-Regel approximiert $f(x)$ durch ein Polynom 2. Grades an den Stellen $x_1 = a, x_2 = \frac{a+b}{2}$ und $x_3 = b$.

Einfache Simpson-Regel:

$$Sf = \frac{b-a}{6} \left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b) \right)$$

Summierte Simpson-Regel:

$$Sf(h) = \frac{h}{3} \left(\frac{1}{2} f(a) + \sum_{i=1}^{n-1} f(x_i) + 2 \sum_{i=1}^n f\left(\frac{x_{i-1} + x_i}{2}\right) + \frac{1}{2} f(b) \right)$$

Simpson-Regel als gewichtetes Mittel

Die summierte Simpson-Regel kann als gewichtetes Mittel der summierten Trapez- und Rechteckregel interpretiert werden:

$$Sf(h) = \frac{1}{3} (Tf(h) + 2Rf(h))$$

Bewegung durch Flüssigkeit - Verschiedene Regeln

Aufgabe: Ein Teilchen mit Masse $m = 10$ kg bewegt sich durch eine Flüssigkeit mit Widerstand $R(v) = -v\sqrt{v}$. Für die Verlangsamung von $v_0 = 20$ m/s auf $v = 5$ m/s gilt:

$$t = \int_5^{20} \frac{m}{R(v)} dv = \int_5^{20} \frac{10}{-v\sqrt{v}} dv$$

Berechnen Sie das Integral mit $n = 5$ für:

- Summierte Rechteckregel
- Summierte Trapezregel
- Summierte Simpson-Regel

Parametrisation: $h = \frac{20-5}{5} = 3$, Stützstellen: 5, 8, 11, 14, 17, 20

a) Rechteckregel: $Rf(3) = 3 \cdot \sum_{i=0}^4 f(x_i + 1.5)$ Mittelpunkte: 6.5, 9.5, 12.5, 15.5, 18.5 $Rf(3) = 3 \cdot (-0.154 - 0.108 - 0.090 - 0.081 - 0.076) = -1.527$

b) Trapezregel: $Tf(3) = 3 \cdot \left(\frac{f(5)+f(20)}{2} + \sum_{i=1}^4 f(x_i) \right)$ $Tf(3) = 3 \cdot \left(\frac{-0.179-0.056}{2} + (-0.125 - 0.096 - 0.082 - 0.072) \right) = -1.477$

c) Simpson-Regel: $Sf(3) = \frac{1}{3} (Tf(3) + 2Rf(3)) = \frac{1}{3} (-1.477 + 2(-1.527)) = -1.510$

Exakter Wert: $\int_5^{20} \frac{-10}{v^{3/2}} dv = \left[\frac{20}{\sqrt{v}} \right]_5^{20} = -1.506$

Absolute Fehler:

- Rechteckregel: $|-1.527 - (-1.506)| = 0.021$
- Trapezregel: $|-1.477 - (-1.506)| = 0.029$
- Simpson-Regel: $|-1.510 - (-1.506)| = 0.004$

Fehlerabschätzung

Fehlerabschätzung für summierte Quadraturformeln

Für genügend glatte Funktionen gelten folgende Fehlerabschätzungen:
Summierte Rechteckregel:

Equation for the error of the summed rectangle rule.

Summierte Trapezregel:

Equation for the error of the summed trapezoidal rule.

Summierte Simpson-Regel:

Equation for the error of the summed Simpson's rule.

Schrittweite für gewünschte Genauigkeit bestimmen

Schritt 1: Gewünschte Genauigkeit festlegen

Maximaler absoluter Fehler: ϵ

Schritt 2: Höchste Ableitung abschätzen

Berechne $\max_{x \in [a,b]} |f^{(k)}(x)|$ für entsprechendes k .

Schritt 3: Schrittweite berechnen

Für Trapezregel:

Equation for step size h for the trapezoidal rule.

Für Simpson-Regel:

Equation for step size h for Simpson's rule.

Schritt 4: Anzahl Intervalle bestimmen

Equation for the number of intervals n.

Schrittweite für gewünschte Genauigkeit

Aufgabe: Bestimmen Sie die Schrittweite h, um I = integral from 0 to 0.5 of e^-x^2 dx mit der summierten Trapezregel auf einen absoluten Fehler von maximal 10^-5 genau zu berechnen.

Lösung:

Schritt 1: epsilon = 10^-5, a = 0, b = 0.5

Schritt 2: Zweite Ableitung bestimmen: f(x) = e^-x^2, f'(x) = -2xe^-x^2, f''(x) = -2e^-x^2 + 4x^2e^-x^2 = e^-x^2(4x^2 - 2)

Auf [0, 0.5]: max |f''(x)| = max |e^-x^2(4x^2 - 2)| = 2 (bei x = 0)

Schritt 3: Schrittweite berechnen:

Equation for step size h.

Schritt 4: n = 0.5 / 0.011 approx 46 Intervalle

Romberg-Extrapolation

Idee der Romberg-Extrapolation

Die Romberg-Extrapolation verbessert systematisch die Genauigkeit der Trapezregel durch Verwendung mehrerer Schrittweiten und anschließende Extrapolation.
Basis: Trapezregel mit halbierten Schrittweiten h_j = (b-a)/2^j für j = 0, 1, 2, ..., m.

Romberg-Extrapolation

Für die summierte Trapezregel Tf(h) gilt:

Sei T_j0 = Tf((b-a)/2^j) für j = 0, 1, ..., m. Dann sind durch die Rekursion

Equation for the Romberg extrapolation recursion.

für k = 1, 2, ..., m und j = 0, 1, ..., m-k Näherungen der Fehlerordnung 2k + 2 gegeben.
Die verwendete Schrittweitenfolge h_j = (b-a)/2^j heißt Romberg-Folge.

Romberg-Extrapolation durchführen

Schritt 1: Trapezwerte für erste Spalte berechnen

Berechne T_j0 mit der summierten Trapezregel für h_j = (b-a)/2^j, j = 0, 1, ..., m.

Schritt 2: Extrapolationsschema aufstellen

Table with 4 columns and 4 rows representing the Romberg extrapolation scheme.

Schritt 3: Rekursionsformel anwenden

Equation for the recursion formula in the Romberg scheme.

Schritt 4: Genaueste Näherung

Der Wert rechts unten im Schema ist die genaueste Approximation.

Romberg-Extrapolation anwenden

Berechne integral from 0 to pi of cos(x^2) dx mit Romberg-Extrapolation für m = 4 (d.h. j = 0, 1, 2, 3, 4).

Schritt 1: Erste Spalte berechnen T_00 = Tf(pi) mit h_0 = pi (1 Intervall) T_10 = Tf(pi/2) mit h_1 = pi/2 (2 Intervalle) T_20 = Tf(pi/4) mit h_2 = pi/4 (4 Intervalle) T_30 = Tf(pi/8) mit h_3 = pi/8 (8 Intervalle) T_40 = Tf(pi/16) mit h_4 = pi/16 (16 Intervalle)

Beispielrechnung für T_00:

Equation for the example calculation of T_00.

Schritt 2: Extrapolationsschema:

Table with 5 columns and 5 rows representing the Romberg extrapolation scheme for the example.

Der Wert T_04 liefert die beste Approximation des Integrals.

Gauss-Formeln

Optimale Stützstellen

Bei Newton-Cotes Formeln sind die Stützstellen äquidistant gewählt. Gauss-Formeln wählen sowohl Stützstellen x_i als auch Gewichte a_i optimal, um die Fehlerordnung zu maximieren.

Gauss-Formeln für n = 1, 2, 3

Die Gauss-Formeln für integral from a to b of f(x) dx approx (b-a)/2 * sum from i=1 to n of a_i f(x_i) lauten:

Equation for Gauss formula n=1.

Equation for Gauss formula n=2.

Equation for Gauss formula n=3.

wobei x_1 = -sqrt(0.6) * (b-a)/2 + (b+a)/2 und x_3 = sqrt(0.6) * (b-a)/2 + (b+a)/2.

Erdmasse berechnen

Aufgabe: Berechnen Sie die Masse der Erde mit der nicht-äquidistanten Dichteverteilung:

Equation for the mass of the Earth.

Table with 6 columns: r [km], rho [kg/m^3], and density values at different radii.

Lösung:

Da die Stützstellen nicht äquidistant sind, verwenden wir die summierte Trapezregel für nicht-äquidistante Daten:

Equation for the summed trapezoidal rule for non-equidistant data.

Wichtig: Umrechnung der Einheiten: r in km -> m, rho in kg/m^3

Ergebnis: m_Erde approx 5.94 x 10^24 kg

Vergleich mit Literaturwert: 5.97 x 10^24 kg Relativer Fehler: approx 0.5%

Wahl des Integrationsverfahrens:

- Trapezregel: Einfach, für glatte Funktionen ausreichend
- Simpson-Regel: Höhere Genauigkeit, besonders für polynomähnliche Funktionen
- Romberg-Extrapolation: Sehr hohe Genauigkeit mit systematischer Verbesserung
- Gauss-Formeln: Optimal für begrenzte Anzahl von Funktionsauswertungen
- Nicht-äquidistante Daten: Spezielle Trapezregel für tabellarische Daten

Prüfungsaufgabe 7.1 - Vergleich der Integrationsregeln

Aufgabe: Berechnen Sie das Integral $I = \int_0^1 e^{x^2} dx$ näherungsweise mit:
a) Summierten Trapezregel mit $n = 4$ b) Summierten Simpson-Regel mit $n = 4$ c) Vergleichen Sie mit dem numerischen Referenzwert $I \approx 1.4627$ d) Welche Schrittweite braucht die Trapezregel für einen Fehler $< 10^{-3}$?

a) Summierte Trapezregel mit $n = 4$: $h = \frac{1-0}{4} = 0.25$
Stützstellen: $x_0 = 0, x_1 = 0.25, x_2 = 0.5, x_3 = 0.75, x_4 = 1$
Funktionswerte:

- $f(0) = e^0 = 1$
- $f(0.25) = e^{0.0625} = 1.0645$
- $f(0.5) = e^{0.25} = 1.2840$
- $f(0.75) = e^{0.5625} = 1.7551$
- $f(1) = e^1 = 2.7183$

$$T_f(0.25) = 0.25 \cdot \left(\frac{1 + 2.7183}{2} + 1.0645 + 1.2840 + 1.7551 \right)$$

$$= 0.25 \cdot (1.8592 + 4.1036) = 0.25 \cdot 5.9628 = 1.4907$$

b) Summierte Simpson-Regel mit $n = 4$: Mittelpunkte:
 $x_{0.5} = 0.125, x_{1.5} = 0.375, x_{2.5} = 0.625, x_{3.5} = 0.875$
Funktionswerte an Mittelpunkten:

- $f(0.125) = e^{0.015625} = 1.0158$
- $f(0.375) = e^{0.140625} = 1.1508$
- $f(0.625) = e^{0.390625} = 1.4781$
- $f(0.875) = e^{0.765625} = 2.1507$

$$S_f(0.25) = \frac{1}{3} (T_f(0.25) + 2R_f(0.25))$$

$$R_f(0.25) = 0.25 \cdot (1.0158 + 1.1508 + 1.4781 + 2.1507) = 0.25 \cdot 5.7954 = 1.4489$$

$$S_f(0.25) = \frac{1}{3} (1.4907 + 2 \cdot 1.4489) = \frac{1}{3} \cdot 4.3885 = 1.4628$$

c) Fehlervergleich:

- Trapezregel: $|1.4907 - 1.4627| = 0.0280$
- Simpson-Regel: $|1.4628 - 1.4627| = 0.0001$
- Die Simpson-Regel ist deutlich genauer (Fehlerordnung 4 vs. 2)

d) Schrittweite für Trapezregel mit Fehler $< 10^{-3}$:
Fehlerabschätzung: $|I - T_f(h)| \leq \frac{h^2}{12} (b - a) \max_{x \in [a, b]} |f''(x)|$

Für $f(x) = e^{x^2}$:

$$f'(x) = 2xe^{x^2}, \quad f''(x) = 2e^{x^2} + 4x^2e^{x^2} = e^{x^2} (2 + 4x^2)$$

Auf $[0, 1]$:

$$\max |f''(x)| = f''(1) = e^1 (2 + 4) = 6e \approx 16.31$$

$$\frac{h^2}{12} \cdot 1 \cdot 16.31 \leq 10^{-3} \quad h^2 \leq \frac{0.012}{16.31} \approx 7.36 \times 10^{-4} \quad h \leq 0.0271$$

Also: $n = \frac{1}{0.0271} \approx 37$ Intervalle erforderlich.

Prüfungsaufgabe 7.2 - Romberg-Extrapolation

Aufgabe: Berechnen Sie $I = \int_0^{\pi/2} \sin(x) dx$ mit Romberg-Extrapolation für $m = 3$ (d.h. $j = 0, 1, 2, 3$).
a) Berechnen Sie die erste Spalte T_{j0} für $j = 0, 1, 2, 3$ b) Führen Sie die Romberg-Extrapolation durch c) Vergleichen Sie mit dem exakten Wert $I = 1$

a) Erste Spalte berechnen:
 $h_j = \frac{\pi/2 - 0}{2^j} = \frac{\pi/2}{2^j}$
 $j = 0$: $h_0 = \pi/2, n_0 = 1$

$$T_{00} = \frac{\pi/2}{2} \cdot (\sin(0) + \sin(\pi/2)) = \frac{\pi}{4} \cdot (0 + 1) = \frac{\pi}{4} = 0.7854$$

$j = 1$: $h_1 = \pi/4, n_1 = 2$

$$T_{10} = \frac{\pi/4}{2} \cdot (\sin(0) + \sin(\pi/2)) + \frac{\pi}{4} \cdot \sin(\pi/4)$$

$$= \frac{\pi}{8} \cdot (0 + 1) + \frac{\pi}{4} \cdot \frac{\sqrt{2}}{2} = \frac{\pi}{8} + \frac{\pi\sqrt{2}}{8} = \frac{\pi(1 + \sqrt{2})}{8} = 0.9480$$

$j = 2$: $h_2 = \pi/8, n_2 = 4$ Stützstellen: $0, \pi/8, \pi/4, 3\pi/8, \pi/2$

$$T_{20} = \frac{\pi/8}{2} \cdot (\sin(0) + \sin(\pi/2)) + \frac{\pi}{8} \cdot (\sin(\pi/8) + \sin(\pi/4) + \sin(3\pi/8))$$

Mit $\sin(\pi/8) = \sqrt{\frac{2-\sqrt{2}}{4}} \approx 0.3827$ und $\sin(3\pi/8) \approx 0.9239$:

$$T_{20} = \frac{\pi}{16} + \frac{\pi}{8} \cdot (0.3827 + 0.7071 + 0.9239) = 0.9871$$

$j = 3$: $h_3 = \pi/16, n_3 = 8$ $T_{30} = 0.9968$ (detaillierte Rechnung analog)

b) Romberg-Extrapolation:

$$T_{01} = \frac{4T_{10} - T_{00}}{4 - 1} = \frac{4 \cdot 0.9480 - 0.7854}{3} = \frac{3.0366}{3} = 1.0122$$

$$T_{11} = \frac{4T_{20} - T_{10}}{4 - 1} = \frac{4 \cdot 0.9871 - 0.9480}{3} = \frac{2.9964}{3} = 0.9988$$

$$T_{21} = \frac{4T_{30} - T_{20}}{4 - 1} = \frac{4 \cdot 0.9968 - 0.9871}{3} = \frac{3.0001}{3} = 1.0000$$

$$T_{02} = \frac{16T_{11} - T_{01}}{16 - 1} = \frac{16 \cdot 0.9988 - 1.0122}{15} = \frac{14.9686}{15} = 0.9979$$

$$T_{12} = \frac{16T_{21} - T_{11}}{16 - 1} = \frac{16 \cdot 1.0000 - 0.9988}{15} = \frac{15.0012}{15} = 1.0001$$

$$T_{03} = \frac{64T_{12} - T_{02}}{64 - 1} = \frac{64 \cdot 1.0001 - 0.9979}{63} = 1.0000$$

Romberg-Schema:

0.7854			
0.9480	1.0122		
0.9871	0.9988	0.9979	
0.9968	1.0000	1.0001	1.0000

c) Vergleich mit exaktem Wert: Der extrapolierte Wert $T_{03} = 1.0000$ stimmt mit dem exakten Wert $I = 1$ überein (bis auf Rundungsfehler). Die Romberg-Extrapolation konvergiert sehr schnell für glatte Funktionen wie $\sin(x)$.

Prüfungsaufgabe 7.3 - Anwendung in der Physik

Aufgabe: Die Geschwindigkeit eines fallenden Objekts mit Luftwiderstand ist gegeben durch: $v(t) = \sqrt{\frac{mg}{k}} \tanh\left(\sqrt{\frac{gk}{m}} \cdot t\right)$
mit $m = 80$ kg, $g = 9.81$ m/s², $k = 0.25$ kg/m.
a) Berechnen Sie die in den ersten 10 Sekunden zurückgelegte Strecke $s = \int_0^{10} v(t) dt$ b) Verwenden Sie die Simpson-Regel mit $n = 10$ c) Interpretieren Sie das Ergebnis physikalisch

a) Parameter und Funktion:

$$\sqrt{\frac{mg}{k}} = \sqrt{\frac{80 \cdot 9.81}{0.25}} = \sqrt{3139.2} = 56.0 \text{ m/s}$$

$$\sqrt{\frac{gk}{m}} = \sqrt{\frac{9.81 \cdot 0.25}{80}} = \sqrt{0.0306} = 0.175 \text{ s}^{-1}$$

$$v(t) = 56.0 \cdot \tanh(0.175 \cdot t)$$

b) Simpson-Regel mit $n = 10$: $h = \frac{10-0}{10} = 1$ s
Funktionswerte an Stützstellen:

- $v(0) = 56.0 \cdot \tanh(0) = 0$
- $v(1) = 56.0 \cdot \tanh(0.175) = 56.0 \cdot 0.173 = 9.69$
- $v(2) = 56.0 \cdot \tanh(0.35) = 56.0 \cdot 0.336 = 18.82$
- $v(3) = 56.0 \cdot \tanh(0.525) = 56.0 \cdot 0.481 = 26.94$
- $v(4) = 56.0 \cdot \tanh(0.7) = 56.0 \cdot 0.604 = 33.82$
- $v(5) = 56.0 \cdot \tanh(0.875) = 56.0 \cdot 0.704 = 39.42$
- $v(6) = 56.0 \cdot \tanh(1.05) = 56.0 \cdot 0.785 = 43.96$
- $v(7) = 56.0 \cdot \tanh(1.225) = 56.0 \cdot 0.847 = 47.43$
- $v(8) = 56.0 \cdot \tanh(1.4) = 56.0 \cdot 0.896 = 50.18$
- $v(9) = 56.0 \cdot \tanh(1.575) = 56.0 \cdot 0.933 = 52.25$
- $v(10) = 56.0 \cdot \tanh(1.75) = 56.0 \cdot 0.959 = 53.70$

Mittelpunkte:

- $v(0.5) = 56.0 \cdot \tanh(0.0875) = 4.89$
- $v(1.5) = 56.0 \cdot \tanh(0.2625) = 14.53$
- $v(2.5) = 56.0 \cdot \tanh(0.4375) = 22.99$
- $v(3.5) = 56.0 \cdot \tanh(0.6125) = 30.52$
- $v(4.5) = 56.0 \cdot \tanh(0.7875) = 36.75$
- $v(5.5) = 56.0 \cdot \tanh(0.9625) = 41.87$
- $v(6.5) = 56.0 \cdot \tanh(1.1375) = 45.97$
- $v(7.5) = 56.0 \cdot \tanh(1.3125) = 48.84$
- $v(8.5) = 56.0 \cdot \tanh(1.4875) = 51.16$
- $v(9.5) = 56.0 \cdot \tanh(1.6625) = 52.99$

Trapezregel:

$$T = 1 \cdot \left(\frac{0 + 53.70}{2} + 9.69 + 18.82 + \dots + 52.25 \right) = 356.2 \text{ m}$$

Rechteckregel:

$$R = 1 \cdot (4.89 + 14.53 + 22.99 + \dots + 52.99) = 350.5 \text{ m}$$

Simpson-Regel:

$$S = \frac{1}{3} (T + 2R) = \frac{1}{3} (356.2 + 2 \cdot 350.5) = \frac{1057.2}{3} = 352.4 \text{ m}$$

c) Physikalische Interpretation:

- Das Objekt legt in 10 Sekunden etwa 352 m zurück
- Die Endgeschwindigkeit nähert sich asymptotisch $v_{max} = 56$ m/s
- Der tanh-Verlauf zeigt die typische Charakteristik: anfangs beschleunigt das Objekt stark, dann verlangsamt sich die Beschleunigung, bis die Endgeschwindigkeit erreicht ist
- Ohne Luftwiderstand wäre $s = \frac{1}{2}gt^2 = 490$ m (deutlich mehr)

Einführung in gewöhnliche Differentialgleichungen

Gewöhnliche Differentialgleichung n-ter Ordnung

Eine Gleichung, in der Ableitungen einer unbekannten Funktion $y = y(x)$ bis zur n -ten Ordnung auftreten, heißt eine gewöhnliche Differentialgleichung n -ter Ordnung. Sie hat die explizite Form:

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

Gesucht sind die Lösungen $y = y(x)$ dieser Gleichung, wobei die Lösungen y auf einem Intervall $[a, b]$ definiert sein sollen.

Notationen für Ableitungen:

Lagrange: $y'(x), y''(x), y'''(x), y^{(4)}(x), \dots, y^{(n)}(x)$

Newton: $\dot{y}(x), \ddot{y}(x), \ddot{\ddot{y}}(x), \dots$

Leibniz: $\frac{dy}{dx}, \frac{d^2y}{dx^2}, \frac{d^3y}{dx^3}, \dots, \frac{d^ny}{dx^n}$

Anfangswertproblem (AWP)

Bei einem Anfangswertproblem für eine Differentialgleichung n -ter Ordnung werden der Lösungsfunktion $y = y(x)$ noch n Werte vorgeschrieben:

DGL 1. Ordnung:

Gegeben ist $y'(x) = f(x, y(x))$ und der Anfangswert $y(x_0) = y_0$.

DGL 2. Ordnung: Gegeben ist $y''(x) = f(x, y(x), y'(x))$ und die Anfangswerte $y(x_0) = y_0, y'(x_0) = y'_0$.

Beispiele aus den Naturwissenschaften

Aufgabe: Klassifizieren Sie die folgenden DGL und geben Sie physikalische Interpretationen an.

1. Radioaktiver Zerfall:

$$\frac{dn}{dt} = -\lambda n$$

DGL 1. Ordnung, Lösung: $n(t) = n_0 e^{-\lambda t}$

2. Freier Fall:

$$\ddot{s}(t) = -g$$

DGL 2. Ordnung, Lösung: $s(t) = -\frac{1}{2}gt^2 + v_0t + s_0$

3. Harmonische Schwingung (Federpendel):

$$m\ddot{x} = -cx \Rightarrow \ddot{x} + \frac{c}{m}x = 0$$

DGL 2. Ordnung, Lösung: $x(t) = A \sin(\omega_0 t + \varphi)$ mit $\omega_0 = \sqrt{\frac{c}{m}}$

Richtungsfelder

Geometrische Interpretation

Die DGL $y'(x) = f(x, y(x))$ gibt uns einen Zusammenhang zwischen der Steigung $y'(x)$ der gesuchten Funktion und dem Punkt $(x, y(x))$. Im Richtungsfeld wird an jedem Punkt (x, y) die Steigung $y'(x) = f(x, y)$ durch einen kleinen Pfeil dargestellt. Die Lösungskurven verlaufen stets tangential zu diesen Pfeilen.

Richtungsfeld zeichnen und interpretieren

Schritt 1: Steigungen berechnen

Für eine gegebene DGL $y' = f(x, y)$ berechne für verschiedene Punkte (x_i, y_j) die Steigung $f(x_i, y_j)$.

Schritt 2: Richtungspfeile einzeichnen

Zeichne an jedem Punkt (x_i, y_j) einen kleinen Pfeil mit der Steigung $f(x_i, y_j)$.

Schritt 3: Lösungskurven folgen

Von einem Anfangspunkt (x_0, y_0) ausgehend folge den Richtungspfeilen, um die Lösungskurve zu approximieren.

Schritt 4: Python-Implementierung

Verwende `numpy.meshgrid()` und `pyplot.quiver()` zur automatischen Darstellung.

Richtungsfeld interpretieren

Aufgabe: Zeichnen Sie das Richtungsfeld für $\frac{dy}{dt} = -\frac{1}{2} \cdot y \cdot t^2$ und bestimmen Sie die Lösungskurve für $y(0) = 3$.

Lösung:

Steigungen an ausgewählten Punkten:

$\frac{dy}{dt}$	$t = 0$	$t = 1$	$t = 2$	$t = 3$
$y = 0$	0	0	0	0
$y = 1$	0	-0.5	-2	-4.5
$y = 2$	0	-1	-4	-9
$y = 3$	0	-1.5	-6	-13.5

Die Lösungskurve für $y(0) = 3$ folgt den Richtungspfeilen und zeigt exponentiellen Abfall für $t > 0$.

Numerische Lösungsverfahren

Das Euler-Verfahren

Klassisches Euler-Verfahren

Gegeben sei das AWP $y' = f(x, y)$ mit $y(a) = y_0$ auf dem Intervall $[a, b]$.

Das Euler-Verfahren mit Schrittweite $h = \frac{b-a}{n}$ lautet:

$$x_{i+1} = x_i + h$$

$$y_{i+1} = y_i + h \cdot f(x_i, y_i)$$

wobei $x_0 = a$, $x_i = a + ih$ für $i = 0, \dots, n - 1$ und y_0 der gegebene Anfangswert ist.

Idee des Euler-Verfahrens

Das Euler-Verfahren folgt der Tangente im Punkt (x_i, y_i) mit der Steigung $f(x_i, y_i)$ um die Schrittweite h . Es ist das einfachste Einschrittverfahren mit Konvergenzordnung $p = 1$.

Euler-Verfahren anwenden

Schritt 1: Parameter bestimmen

Gegeben: AWP $y' = f(x, y)$, $y(a) = y_0$, Intervall $[a, b]$, Anzahl Schritte n
Berechne: $h = \frac{b-a}{n}$

Schritt 2: Startwerte setzen

$x_0 = a$, $y_0 =$ gegebener Anfangswert

Schritt 3: Iteration

Für $i = 0, 1, \dots, n - 1$:

- Berechne $f(x_i, y_i)$
- Setze $x_{i+1} = x_i + h$
- Setze $y_{i+1} = y_i + h \cdot f(x_i, y_i)$

Schritt 4: Lösung interpretieren

Die Punkte (x_i, y_i) approximieren die Lösung $y(x)$ an den Stützstellen.

Euler-Verfahren berechnen

Aufgabe: Lösen Sie $\frac{dy}{dx} = \frac{x^2}{y}$ mit $y(0) = 2$ auf dem Intervall $[0, 1.4]$ mit $h = 0.7$ (Euler-Verfahren).

Lösung:

Parameter: $n = 2$, $h = 0.7$, $f(x, y) = \frac{x^2}{y}$

Iteration:

- $i = 0$: $x_0 = 0$, $y_0 = 2$
- $f(0, 2) = \frac{0^2}{2} = 0$
- $x_1 = 0 + 0.7 = 0.7$, $y_1 = 2 + 0.7 \cdot 0 = 2$
- $i = 1$: $x_1 = 0.7$, $y_1 = 2$
- $f(0.7, 2) = \frac{0.7^2}{2} = 0.245$
- $x_2 = 0.7 + 0.7 = 1.4$, $y_2 = 2 + 0.7 \cdot 0.245 = 2.1715$

Exakte Lösung: $y(x) = \sqrt{\frac{2x^3}{3} + 4}$ $y(1.4) = \sqrt{\frac{2 \cdot 1.4^3}{3} + 4} = 2.253$

Absoluter Fehler: $|2.253 - 2.1715| = 0.0815$

Verbesserte Euler-Verfahren

Mittelpunkt-Verfahren

Das Mittelpunkt-Verfahren berechnet die Steigung in der Mitte des Intervalls:

$$x_{h/2} = x_i + \frac{h}{2}$$

$$y_{h/2} = y_i + \frac{h}{2} \cdot f(x_i, y_i)$$

$$x_{i+1} = x_i + h$$

$$y_{i+1} = y_i + h \cdot f(x_{h/2}, y_{h/2})$$

Konvergenzordnung: $p = 2$

Modifiziertes Euler-Verfahren (Heun-Verfahren)

Das modifizierte Euler-Verfahren verwendet den Durchschnitt zweier Steigungen:

$$k_1 = f(x_i, y_i)$$

$$k_2 = f(x_i + h, y_i + h \cdot k_1)$$

$$x_{i+1} = x_i + h$$

$$y_{i+1} = y_i + h \cdot \frac{k_1 + k_2}{2}$$

Konvergenzordnung: $p = 2$

Vergleich der Euler-Verfahren

Aufgabe: Lösen Sie das AWP aus dem vorigen Beispiel mit Mittelpunkt- und modifiziertem Euler-Verfahren. Vergleichen Sie die Genauigkeit.

Mittelpunkt-Verfahren:

- $x_{1/2} = 0.35$, $y_{1/2} = 2$, $f(0.35, 2) = 0.061$
- $y_1 = 2 + 0.7 \cdot 0.061 = 2.043$
- $x_{3/2} = 1.05$, $y_{3/2} = 2.128$, $f(1.05, 2.128) = 0.518$
- $y_2 = 2.043 + 0.7 \cdot 0.518 = 2.406$

Modifiziertes Euler-Verfahren:

- $k_1 = 0$, $k_2 = f(0.7, 2) = 0.245$
- $y_1 = 2 + 0.7 \cdot \frac{0+0.245}{2} = 2.086$
- $k_1 = 0.245$, $k_2 = f(1.4, 2.257) = 0.866$
- $y_2 = 2.086 + 0.7 \cdot \frac{0.245+0.866}{2} = 2.475$

Fehlervergleich bei $x = 1.4$:

- Exakt: $y(1.4) = 2.253$
- Euler: $|2.253 - 2.172| = 0.081$
- Mittelpunkt: $|2.253 - 2.406| = 0.153$
- Modifiziert: $|2.253 - 2.475| = 0.222$

Runge-Kutta Verfahren

Klassisches vierstufiges Runge-Kutta Verfahren

Das klassische Runge-Kutta Verfahren verwendet vier Steigungen und hat Konvergenzordnung $p = 4$:

$$k_1 = f(x_i, y_i), \quad k_2 = f(x_i + \frac{h}{2}, y_i + \frac{h}{2} k_1)$$

$$k_3 = f(x_i + \frac{h}{2}, y_i + \frac{h}{2} k_2), \quad k_4 = f(x_i + h, y_i + h k_3)$$

$$x_{i+1} = x_i + h, \quad y_{i+1} = y_i + h \cdot \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

Butcher-Schema Runge-Kutta Verfahren werden durch Butcher-Schemata charakterisiert:

0				
$\frac{1}{2}$	$\frac{1}{2}$			
$\frac{1}{2}$	0	$\frac{1}{2}$		
1	0	0	1	
	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{6}$

Interpretation: Die erste Spalte gibt die Stufen c_i , die zweite Spalte die Koeffizienten a_{ij} für die Steigungen k_j und die letzte Zeile die Gewichtung der Steigungen für die nächste Iteration an.

Runge-Kutta Verfahren anwenden

Schritt 1: Steigungen berechnen

$$k_1 = f(x_i, y_i), \quad k_2 = f(x_i + \frac{h}{2}, y_i + \frac{h}{2} k_1)$$

$$k_3 = f(x_i + \frac{h}{2}, y_i + \frac{h}{2} k_2), \quad k_4 = f(x_i + h, y_i + h k_3)$$

Schritt 2: Gewichtetes Mittel bilden

$$\text{Steigung} = \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

Schritt 3: Nächsten Punkt berechnen

$$x_{i+1} = x_i + h, \quad y_{i+1} = y_i + h \cdot \text{Steigung}$$

Schritt 4: Iteration fortsetzen

Wiederhole bis zum Ende des Intervalls.

Runge-Kutta vs. andere Verfahren

Aufgabe: Lösen Sie $y' = 1 - \frac{y}{t}$ mit $y(1) = 5$ für $t \in [1, 6]$ mit $h = 0.01$ und vergleichen Sie mit der exakten Lösung $y(t) = \frac{t^2+9}{2t}$.

```
1 def runge_kutta_4(f, a, b, n, y0):
2     h = (b - a) / n
3     x = a
4     y = y0
5
6     for i in range(n):
7         k1 = f(x, y)
8         k2 = f(x + h/2, y + h/2 * k1)
9         k3 = f(x + h/2, y + h/2 * k2)
10        k4 = f(x + h, y + h * k3)
11
12        x += h
13        y += h * (k1 + 2*k2 + 2*k3 + k4) / 6
14
15    return x, y
```

Fehlervergleich bei $t = 6$:

- Exakt: $y(6) = 3.25$
- Euler: Fehler ≈ 0.1
- Runge-Kutta: Fehler $\approx 10^{-6}$

Systeme von Differentialgleichungen

DGL höherer Ordnung → System 1. Ordnung

Jede DGL n -ter Ordnung kann in ein System von n DGL 1. Ordnung umgewandelt werden durch Einführung von Hilfsvariablen für die Ableitungen.

DGL höherer Ordnung auf System 1. Ordnung zurückführen

Schritt 1: Nach höchster Ableitung auflösen

Bringe die DGL in die Form $y^{(n)} = f(x, y, y', \dots, y^{(n-1)})$.

Schritt 2: Hilfsvariablen einführen

Schritt 3: System aufstellen

$$z_1(x) = y(x)$$
$$z_2(x) = y'(x)$$
$$z_3(x) = y''(x)$$
$$\dots$$
$$z_n(x) = y^{(n-1)}(x)$$

$$z_1' = z_2$$
$$z_2' = z_3$$
$$\dots$$
$$z_{n-1}' = z_n$$
$$z_n' = f(x, z_1, z_2, \dots, z_n)$$

Schritt 4: Vektorielle Schreibweise

$\mathbf{z}' = \mathbf{f}(x, \mathbf{z})$ mit $\mathbf{z}(x_0) = \begin{pmatrix} y(x_0) \\ y'(x_0) \\ \vdots \\ y^{(n-1)}(x_0) \end{pmatrix}$

Landende Boeing - DGL 2. Ordnung

Aufgabe: Eine Boeing 737-200 landet mit $v_0 = 100$ m/s und erfährt die Bremskraft $F = -5\dot{x}^2 - 570000$. Die Bewegungsgleichung ist:

$$m\ddot{x} = -5\dot{x}^2 - 570000$$

mit $m = 97000$ kg. Formen Sie in ein System 1. Ordnung um.

Schritt 1: Nach \ddot{x} auflösen:

$$\ddot{x} = \frac{-5\dot{x}^2 - 570000}{97000}$$

Schritt 2: Hilfsvariablen:

$$z_1(t) = x(t) \quad (\text{Position})$$
$$z_2(t) = \dot{x}(t) = v(t) \quad (\text{Geschwindigkeit})$$

Schritt 3: System 1. Ordnung:

$$z_1' = z_2$$
$$z_2' = \frac{-5z_2^2 - 570000}{97000}$$

Schritt 4: Anfangsbedingungen:

$$\mathbf{z}(0) = \begin{pmatrix} 0 \\ 100 \end{pmatrix}$$

Das System kann nun mit Runge-Kutta gelöst werden.

Raketenbewegung

Aufgabe: Die Bewegungsgleichung einer Rakete lautet:

$$a(t) = \ddot{h}(t) = v_{rel} \cdot \frac{\mu}{m_A - \mu \cdot t} - g$$

mit $v_{rel} = 2600$ m/s, $m_A = 300000$ kg, $m_E = 80000$ kg, $t_E = 190$ s. Berechnen Sie Geschwindigkeit und Höhe als Funktion der Zeit.

Parameter: $\mu = \frac{m_A - m_E}{t_E} = \frac{220000}{190} = 1158$ kg/s

System 1. Ordnung:

$$z_1' = z_2 \quad (\text{Höhe})$$
$$z_2' = 2600 \cdot \frac{1158}{300000 - 1158t} - 9.81 \quad (\text{Geschwindigkeit})$$

Anfangsbedingungen: $z_1(0) = 0, z_2(0) = 0$

Numerische Lösung mit Trapezregel:

$$v(t) = \int_0^t a(\tau) d\tau$$

$$h(t) = \int_0^t v(\tau) d\tau$$

Analytische Vergleichslösung:

$$v(t) = v_{rel} \ln \left(\frac{m_A}{m_A - \mu t} \right) - gt$$

$$h(t) = -\frac{v_{rel}(m_A - \mu t)}{\mu} \ln \left(\frac{m_A}{m_A - \mu t} \right) + v_{rel}t - \frac{1}{2}gt^2$$

Ergebnisse nach 190s:

- Geschwindigkeit: ≈ 2500 m/s
- Höhe: ≈ 180 km
- Beschleunigung: $\approx 2.5g$

Stabilität

Stabilitätsproblem

Bei der numerischen Lösung von DGL kann es vorkommen, dass der numerische Fehler unbeschränkt wächst, unabhängig von der Schrittweite h . Dies führt zu **instabilen** Lösungen.

Die Stabilität hängt ab von:

- Dem verwendeten Verfahren
- Der Schrittweite h
- Dem spezifischen Anfangswertproblem

Stabilitätsfunktion

Für die DGL $y' = -\alpha y$ (mit $\alpha > 0$) kann die numerische Lösung in der Form

$$y_{i+1} = g(h\alpha) \cdot y_i$$

geschrieben werden. Die Funktion $g(z)$ heißt **Stabilitätsfunktion** des Verfahrens.

Das offene Intervall $z \in (0, \alpha)$, in dem $|g(z)| < 1$ gilt, bezeichnet man als **Stabilitätsintervall**.

Stabilität des Euler-Verfahrens

Aufgabe: Untersuchen Sie die Stabilität des Euler-Verfahrens für $y' = -2.5y$ mit $y(0) = 1$.

Euler-Verfahren: $y_{i+1} = y_i - h \cdot 2.5y_i = y_i(1 - 2.5h)$

Stabilitätsfunktion: $g(z) = 1 - z$ mit $z = 2.5h$

Stabilitätsbedingung: $|1 - 2.5h| < 1 \Rightarrow 0 < 2.5h < 2 \Rightarrow 0 < h < 0.8$

Verhalten:

- $h = 0.2$: Stabile Lösung (exponentieller Abfall)
- $h = 0.85$: Instabile Lösung (Oszillation mit wachsender Amplitude)

Exakte Lösung: $y(x) = e^{-2.5x}$ (streng monoton fallend)

Praktische Hinweise zur Stabilität:

- **Schrittweiten-Kontrolle:** Beginne mit kleiner Schrittweite und prüfe Konvergenz
- **Verfahrensvergleich:** Teste verschiedene Verfahren und vergleiche Ergebnisse
- **Analytische Kontrolle:** Vergleiche mit bekannten analytischen Lösungen
- **Steife DGL:** Verwende implizite Verfahren für steife Probleme
- **Python-Tools:** Nutze `scipy.integrate.solve_ivp()` für robuste Implementierungen

DGL-Löser in Python

Standard-Bibliothek für DGL-Probleme:

```
1 from scipy.integrate import solve_ivp
2 import numpy as np
3 import matplotlib.pyplot as plt
4
5 def f(t, y): # DGL: y' = -2*y + sin(t)
6     return -2*y + np.sin(t)
7
8 # Anfangswertproblem lösen
9 sol = solve_ivp(f, [0, 5], [1], dense_output=True)
10 # Lösung plotten
11 t = np.linspace(0, 5, 100)
12 y = sol.sol(t)
13 plt.plot(t, y[0])
14 plt.xlabel('t')
15 plt.ylabel('y(t)')
16 plt.title('Numerische Loesung der DGL')
17 plt.show()
```

- Verfügbare Methoden:
- 'RK45': Runge-Kutta 5(4) (Standard)
 - 'RK23': Runge-Kutta 3(2)
 - 'DOP853': Runge-Kutta 8. Ordnung
 - 'Radau': Implizites Runge-Kutta
 - 'BDF': Backward Differentiation (für steife DGL)
 - 'LSODA': Automatische Steifigkeits-Erkennung

Eigene DGL-Löser implementieren

Schritt 1: Grundstruktur

```
1 def runge_kutta_4(f, a, b, n, y0):
2     h = (b - a) / n
3     x = np.linspace(a, b, n+1)
4     y = np.zeros(n+1)
5     # erweitert fuer Systeme: y = np.zeros((n+1, len(y0)))
6     y[0] = y0
7
8     for i in range(n):
9         k1 = f(x[i], y[i])
10        k2 = f(x[i] + h/2, y[i] + h/2 * k1)
11        k3 = f(x[i] + h/2, y[i] + h/2 * k2)
12        k4 = f(x[i] + h, y[i] + h * k3)
13
14        y[i+1] = y[i] + h/6 * (k1 + 2*k2 + 2*k3 + k4)
15
16    return x, y
```

Schritt 2: Richtungsfeld visualisieren

```
1 def plot_direction_field(f, xmin, xmax, ymin, ymax,
2     hx, hy):
3     x = np.arange(xmin, xmax, hx)
4     y = np.arange(ymin, ymax, hy)
5     X, Y = np.meshgrid(x, y)
6
7     DX = np.ones_like(X)
8     DY = f(X, Y)
9
10    plt.quiver(X, Y, DX, DY, alpha=0.6)
11    plt.xlabel('x')
12    plt.ylabel('y')
13    plt.title('Richtungsfeld')
```

Lokaler und globaler Fehler

Lokaler Fehler: Der Fehler nach einer Iteration
 $\varphi(x_i, h) := y(x_{i+1}) - y_{i+1}$
Globaler Fehler: Der Fehler nach n Iterationen $y(x_n) - y_n$
Konsistenzordnung p :
Ein Verfahren hat Konsistenzordnung p , falls $|\varphi(x_i, h)| \leq C \cdot h^{p+1}$
Konvergenzordnung p :
Ein Verfahren hat Konvergenzordnung p , falls $|y(x_n) - y_n| \leq C \cdot h^p$

Konvergenzordnungen der Verfahren

- **Euler-Verfahren:** Konvergenzordnung $p = 1$
 - **Mittelpunkt-Verfahren:** Konvergenzordnung $p = 2$
 - **Modifiziertes Euler-Verfahren:** Konvergenzordnung $p = 2$
 - **Klassisches Runge-Kutta:** Konvergenzordnung $p = 4$
- Praktische Bedeutung:** Bei Halbierung der Schrittweite h reduziert sich der Fehler um den Faktor 2^p .

Konvergenzverhalten untersuchen

Aufgabe: Untersuchen Sie das Konvergenzverhalten verschiedener Verfahren für $\frac{dy}{dx} = \frac{x^2}{y}$ mit $y(0) = 2$ auf $[0, 10]$.

Exakte Lösung: $y(x) = \sqrt{\frac{2x^3}{3} + 4}$

Fehler bei $x = 10$ für verschiedene h :

h	Euler	Mittelpunkt	Mod. Euler	Runge-Kutta
0.1	10^{-1}	10^{-2}	10^{-2}	10^{-5}
0.05	5×10^{-2}	2.5×10^{-3}	2.5×10^{-3}	6×10^{-7}
0.025	2.5×10^{-2}	6×10^{-4}	6×10^{-4}	4×10^{-8}

Beobachtung: Bei Halbierung von h :

- Euler: Fehler halbiert sich (Ordnung 1)
- Mittelpunkt/Mod. Euler: Fehler viertelt sich (Ordnung 2)
- Runge-Kutta: Fehler wird um Faktor 16 kleiner (Ordnung 4)

Schwingungsgleichung - Gekoppeltes System

Aufgabe: Lösen Sie die Schwingungsgleichung $\ddot{x} + \omega^2 x = 0$ mit $x(0) = 1, \dot{x}(0) = 0$ und $\omega = 2$.

System 1. Ordnung: $z_1' = z_2, z_2' = -\omega^2 z_1 = -4z_1$
Anfangsbedingungen: $z_1(0) = 1, z_2(0) = 0$
Analytische Lösung: $x(t) = \cos(2t)$
Numerische Implementierung:

```
1 def harmonic_oscillator(t, z): # z[0] = x, z[1] = dx/dt
2     return [z[1], -4*z[0]]
3
4 # Lösung mit scipy
5 sol = solve_ivp(harmonic_oscillator, [0, 10], [1, 0],
6     method='RK45', dense_output=True)
7
8 t = np.linspace(0, 10, 1000)
9 z = sol.sol(t)
10 x_num = z[0]
11 x_exact = np.cos(2*t)
12 plt.plot(t, x_num, 'b-', label='Numerisch')
13 plt.plot(t, x_exact, 'r--', label='Exakt')
14 plt.legend()
```

Energieerhaltung prüfen: $E = \frac{1}{2}\dot{x}^2 + \frac{1}{2}\omega^2 x^2 = \text{const}$

Populationsdynamik - Logistisches Wachstum

Aufgabe: Das logistische Wachstumsmodell lautet: $\frac{dP}{dt} = rP \left(1 - \frac{P}{K}\right)$ mit Wachstumsrate $r = 0.1$ und Kapazität $K = 1000$. Anfangspopulation: $P(0) = 50$.

Analytische Lösung: $P(t) = \frac{K}{1 + \left(\frac{K}{P_0} - 1\right)e^{-rt}}$

Numerische Lösung:

```
1 def logistic_growth(t, P, r=0.1, K=1000):
2     return r * P * (1 - P/K)
3
4 # Parameter
5 r, K, P0 = 0.1, 1000, 50
6 # Numerische Loesung
7 sol = solve_ivp(lambda t, P: logistic_growth(t, P, r, K),
8     [0, 100], [P0], dense_output=True)
9 # Analytische Loesung
10 def P_exact(t):
11     return K / (1 + (K/P0 - 1) * np.exp(-r*t))
12
13 t = np.linspace(0, 100, 1000)
14 P_num = sol.sol(t)[0]
15 P_ana = P_exact(t)
16
17 plt.plot(t, P_num, 'b-', label='Numerisch')
18 plt.plot(t, P_ana, 'r--', label='Analytisch')
19 plt.axhline(y=K, color='k', linestyle=':',
20     label='Kapazitaet K')
21 plt.xlabel('Zeit t')
22 plt.ylabel('Population P(t)')
23 plt.legend()
```

Charakteristisches Verhalten: Exponentielles Wachstum für kleine P , Sättigung bei K .

Prüfungsaufgabe 8.1 - Vergleich numerischer Verfahren

Aufgabe: Lösen Sie das Anfangswertproblem: $\frac{dy}{dx} = x + y^2$, $y(0) = 0$ auf dem Intervall $[0, 1]$ mit Schrittweite $h = 0.2$.

a) Verwenden Sie das Euler-Verfahren b) Verwenden Sie das klassische Runge-Kutta Verfahren c) Vergleichen Sie die Genauigkeit bei $x = 1$ d) Welche Konvergenzordnung erwarten Sie?

a) Euler-Verfahren: $f(x, y) = x + y^2$, $h = 0.2$, $n = 5$

Iteration 0 \rightarrow **1:** $x_0 = 0, y_0 = 0$ $f(0, 0) = 0 + 0^2 = 0$ $x_1 = 0.2, y_1 = 0 + 0.2 \cdot 0 = 0$

Iteration 1 \rightarrow **2:** $x_1 = 0.2, y_1 = 0$ $f(0.2, 0) = 0.2 + 0^2 = 0.2$ $x_2 = 0.4, y_2 = 0 + 0.2 \cdot 0.2 = 0.04$

Iteration 2 \rightarrow **3:** $x_2 = 0.4, y_2 = 0.04$ $f(0.4, 0.04) = 0.4 + 0.04^2 = 0.4016$ $x_3 = 0.6, y_3 = 0.04 + 0.2 \cdot 0.4016 = 0.1203$

Iteration 3 \rightarrow **4:** $x_3 = 0.6, y_3 = 0.1203$ $f(0.6, 0.1203) = 0.6 + 0.1203^2 = 0.6145$ $x_4 = 0.8, y_4 = 0.1203 + 0.2 \cdot 0.6145 = 0.2432$

Iteration 4 \rightarrow **5:** $x_4 = 0.8, y_4 = 0.2432$ $f(0.8, 0.2432) = 0.8 + 0.2432^2 = 0.8591$ $x_5 = 1.0, y_5 = 0.2432 + 0.2 \cdot 0.8591 = 0.4150$

Euler-Ergebnis: $y(1) \approx 0.4150$

b) Runge-Kutta Verfahren:

Schritt 0 \rightarrow **1:** $x_0 = 0, y_0 = 0$ $k_1 = f(0, 0) = 0$ $k_2 = f(0.1, 0) = 0.1$ $k_3 = f(0.1, 0.01) = 0.1 + 0.01^2 = 0.1001$ $k_4 = f(0.2, 0.02002) = 0.2 + 0.02002^2 = 0.2004$ $y_1 = 0 + \frac{0.2}{6}(0 + 2 \cdot 0.1 + 2 \cdot 0.1001 + 0.2004) = 0.0200$

Schritt 1 \rightarrow **2:** $x_1 = 0.2, y_1 = 0.0200$ $k_1 = f(0.2, 0.0200) = 0.2 + 0.0004 = 0.2004$ $k_2 = f(0.3, 0.0400) = 0.3 + 0.0016 = 0.3016$ $k_3 = f(0.3, 0.0502) = 0.3 + 0.0025 = 0.3025$ $k_4 = f(0.4, 0.0805) = 0.4 + 0.0065 = 0.4065$ $y_2 = 0.0200 + \frac{0.2}{6}(0.2004 + 2 \cdot 0.3016 + 2 \cdot 0.3025 + 0.4065) = 0.0801$

Fortführung analog...

Runge-Kutta Ergebnisse: $y_1 = 0.0200, y_2 = 0.0801, y_3 = 0.1806, y_4 = 0.3214, y_5 = 0.5027$

Runge-Kutta-Ergebnis: $y(1) \approx 0.5027$

c) Genauigkeitsvergleich: Für diese DGL gibt es keine einfache analytische Lösung, aber numerische Referenzlösungen mit sehr kleiner Schrittweite ergeben: $y(1) \approx 0.5463$

Fehler:

- Euler: $|0.5463 - 0.4150| = 0.1313$
- Runge-Kutta: $|0.5463 - 0.5027| = 0.0436$

Das Runge-Kutta Verfahren ist etwa 3-mal genauer.

d) Konvergenzordnung:

- Euler-Verfahren: Konvergenzordnung $p = 1$
- Runge-Kutta 4: Konvergenzordnung $p = 4$

Bei Halbierung der Schrittweite erwarten wir:

- Euler: Fehler halbiert sich
- RK4: Fehler wird um Faktor 16 kleiner

Prüfungsaufgabe 8.2 - System von DGL

Aufgabe: Ein Satellit kreist um die Erde. Seine Bewegung wird beschrieben durch: $\ddot{x} = -\frac{GMx}{(x^2+y^2)^{3/2}}$, $\ddot{y} = -\frac{GM y}{(x^2+y^2)^{3/2}}$

mit $GM = 3.986 \times 10^{14} \text{ m}^3/\text{s}^2$.

Anfangsbedingungen: $x(0) = 7000 \text{ km}$, $y(0) = 0$, $\dot{x}(0) = 0$, $\dot{y}(0) = 7500 \text{ m/s}$

a) Formen Sie in ein System 1. Ordnung um b) Implementieren Sie einen Schritt des Mittelpunkt-Verfahrens mit $h = 10 \text{ s}$ c) Interpretieren Sie die Bewegung physikalisch

a) System 1. Ordnung: Einführung der Hilfsvariablen: $z_1 = x$ (Position x) $z_2 = \dot{x}$ (Geschwindigkeit x) $z_3 = y$ (Position y) $z_4 = \dot{y}$ (Geschwindigkeit y)

System: $z'_1 = z_2$ $z'_2 = -\frac{GM z_1}{(z_1^2 + z_3^2)^{3/2}}$ $z'_3 = z_4$ $z'_4 = -\frac{GM z_3}{(z_1^2 + z_3^2)^{3/2}}$

Vektorschreibweise: $\mathbf{z}' = \mathbf{f}(t, \mathbf{z}) = \begin{pmatrix} z_2 \\ -\frac{GM z_1}{(z_1^2 + z_3^2)^{3/2}} \\ z_4 \\ -\frac{GM z_3}{(z_1^2 + z_3^2)^{3/2}} \end{pmatrix}$

b) Mittelpunkt-Verfahren:

Anfangswerte: $\mathbf{z}(0) = \begin{pmatrix} 7 \times 10^6 \\ 0 \\ 0 \\ 7500 \end{pmatrix}$ (in SI-Einheiten)

Schritt 1: Berechne $\mathbf{f}(0, \mathbf{z}_0)$ $r_0 = \sqrt{(7 \times 10^6)^2 + 0^2} = 7 \times 10^6 \text{ m}$

$a_0 = \frac{GM}{r_0^2} = \frac{3.986 \times 10^{14}}{(7 \times 10^6)^2} = 8.13 \text{ m/s}^2$ $\mathbf{f}(0, \mathbf{z}_0) = \begin{pmatrix} 0 \\ -8.13 \\ 0 \\ 7500 \end{pmatrix}$

Schritt 2: Mittelpunkt berechnen

$\mathbf{z}_{1/2} = \mathbf{z}_0 + \frac{h}{2} \mathbf{f}(0, \mathbf{z}_0) = \begin{pmatrix} 7 \times 10^6 \\ 0 \\ 0 \\ 7500 \end{pmatrix} + 5 \begin{pmatrix} 0 \\ -8.13 \\ 0 \\ 7500 \end{pmatrix} = \begin{pmatrix} 7 \times 10^6 \\ -40.65 \\ 0 \\ 7500 \end{pmatrix}$

Schritt 3: \mathbf{f} am Mittelpunkt berechnen

$r_{1/2} = \sqrt{(7 \times 10^6)^2 + (37500)^2} = 7.0001 \times 10^6 \text{ m}$ $\mathbf{f}(5, \mathbf{z}_{1/2}) = \begin{pmatrix} -40.65 \\ -8.128 \\ 7500 \\ -0.043 \end{pmatrix}$

Schritt 4: Neuen Punkt berechnen

$\mathbf{z}_1 = \mathbf{z}_0 + h \cdot \mathbf{f}(5, \mathbf{z}_{1/2}) = \begin{pmatrix} 7 \times 10^6 - 406.5 \\ -81.28 \\ 75000 \\ 7499.57 \end{pmatrix} = \begin{pmatrix} 6.9996 \times 10^6 \\ -81.28 \\ 75000 \\ 7499.57 \end{pmatrix}$

c) Physikalische Interpretation:

- Der Satellit startet in 7000 km Höhe ($\approx 400 \text{ km}$ über Erdoberfläche)
- Anfangsgeschwindigkeit 7500 m/s ist nahe der Kreisbahngeschwindigkeit
- Die Gravitationskraft sorgt für die zentripetale Beschleunigung
- Nach 10 s hat sich der Satellit bereits deutlich bewegt: $\Delta x = -406.5 \text{ m}$, $\Delta y = 75000 \text{ m}$
- Die Bahn ist eine Ellipse (oder bei passender Geschwindigkeit ein Kreis)
- Erhaltungsgrößen: Energie und Drehimpuls (sollten bei genauer numerischer Lösung konstant bleiben)

Prüfungsaufgabe 8.3 - Stabilität und Fehleranalyse

Aufgabe: Betrachten Sie die DGL: $y' = -5y + \cos(x)$, $y(0) = 1$

a) Bestimmen Sie die analytische Lösung b) Untersuchen Sie die Stabilität des Euler-Verfahrens c) Für welche Schrittweiten ist das Verfahren stabil? d) Vergleichen Sie numerische und analytische Lösung für $h = 0.3$ und $h = 0.5$

a) Analytische Lösung: Die DGL $y' + 5y = \cos(x)$ ist eine lineare DGL 1. Ordnung.

Homogene Lösung: $y_h = C e^{-5x}$

Partikuläre Lösung durch Ansatz $y_p = A \cos(x) + B \sin(x)$: $y'_p = -A \sin(x) + B \cos(x)$

Einsetzen: $-A \sin(x) + B \cos(x) + 5(A \cos(x) + B \sin(x)) = \cos(x)$
 $(-A + 5B) \sin(x) + (B + 5A) \cos(x) = \cos(x)$

Koeffizientenvergleich: $-A + 5B = 0 \Rightarrow A = 5B$ $B + 5A = 1 \Rightarrow B + 25B = 1 \Rightarrow B = \frac{1}{26}$, $A = \frac{5}{26}$

$y_p = \frac{5}{26} \cos(x) + \frac{1}{26} \sin(x)$

Allgemeine Lösung: $y = C e^{-5x} + \frac{5}{26} \cos(x) + \frac{1}{26} \sin(x)$

Anfangsbedingung: $y(0) = C + \frac{5}{26} = 1 \Rightarrow C = \frac{21}{26}$

$y(x) = \frac{21}{26} e^{-5x} + \frac{5}{26} \cos(x) + \frac{1}{26} \sin(x)$

b) Stabilität des Euler-Verfahrens: Für die Testgleichung $y' = -\alpha y$ (hier $\alpha = 5$) lautet die Stabilitätsfunktion: $g(z) = 1 - z$ mit $z = h\alpha = 5h$

Stabilitätsbedingung: $|1 - 5h| < 1 \Rightarrow -1 < 1 - 5h < 1 \Rightarrow -2 < -5h < 0 \Rightarrow 0 < h < \frac{2}{5} = 0.4$

c) Stabilitätsbereich: Das Euler-Verfahren ist stabil für $0 < h < 0.4$.

d) Numerischer Vergleich:

Für $h = 0.3$ (**stabil**): Euler-Iteration mit $f(x, y) = -5y + \cos(x)$:

$x_0 = 0, y_0 = 1$ $y_1 = 1 + 0.3(-5 \cdot 1 + \cos(0)) = 1 + 0.3(-4) = -0.2$

$y_2 = -0.2 + 0.3(-5 \cdot (-0.2) + \cos(0.3)) = -0.2 + 0.3(1 + 0.955) = 0.387$

$y_3 = 0.387 + 0.3(-5 \cdot 0.387 + \cos(0.6)) = 0.387 + 0.3(-1.935 + 0.825) = 0.054$

Bei $x = 0.9$: $y_{\text{Euler}} \approx 0.054$ Analytisch: $y(0.9) = \frac{21}{26} e^{-4.5} + \frac{5}{26} \cos(0.9) + \frac{1}{26} \sin(0.9) = 0.197$

Für $h = 0.5$ (**instabil**): $x_0 = 0, y_0 = 1$ $y_1 = 1 + 0.5(-5 \cdot 1 + 1) = 1 + 0.5(-4) = -1$ $y_2 = -1 + 0.5(-5 \cdot (-1) + \cos(0.5)) = -1 + 0.5(5 + 0.878) = 1.939$ $y_3 = 1.939 + 0.5(-5 \cdot 1.939 + \cos(1)) = 1.939 + 0.5(-9.695 + 0.540) = -2.639$

Die Lösung oszilliert mit wachsender Amplitude \rightarrow instabil!

Interpretation:

- Für $h = 0.3$: Das Verfahren ist stabil, aber nicht sehr genau
- Für $h = 0.5$: Das Verfahren wird instabil und divergiert
- Die Stabilitätsgrenze $h < 0.4$ muss eingehalten werden
- Der schnell abfallende Term e^{-5x} macht die DGL steif

Prüfungsaufgabe 8.4 - Anwendung: Populationsdynamik

Aufgabe: Das Räuber-Beute-System wird beschrieben durch: $\frac{dx}{dt} = ax - bxy$
 $\frac{dy}{dt} = -cy + dxy$
 wobei $x(t)$ die Beutepopulation und $y(t)$ die Räuberpopulation darstellt.
 Parameter: $a = 1.0$, $b = 0.5$, $c = 0.75$, $d = 0.25$ Anfangsbedingungen:
 $x(0) = 4$, $y(0) = 4$

a) Interpretieren Sie die Gleichungen biologisch b) Lösen Sie das System numerisch mit Runge-Kutta für $t \in [0, 15]$ mit $h = 0.1$ c) Analysieren Sie das Langzeitverhalten d) Was passiert mit der Gesamtenergie des Systems?

- a) **Biologische Interpretation: Beutegleichung:** $\frac{dx}{dt} = ax - bxy$
- ax : Exponentielles Wachstum der Beute bei Abwesenheit von Räubern
 - $-bxy$: Verluste durch Räuber (proportional zu beiden Populationen)
 - $a = 1.0$: Wachstumsrate der Beute
 - $b = 0.5$: Effizienz der Räuber beim Beutefang
- Räbergleichung:** $\frac{dy}{dt} = -cy + dxy$
- $-cy$: Exponentieller Rückgang bei Abwesenheit von Beute
 - dxy : Wachstum durch erfolgreiche Jagd
 - $c = 0.75$: Sterberate der Räuber
 - $d = 0.25$: Effizienz der Nahrungsumwandlung

- b) **Numerische Lösung:**
- System in Vektorform:
- $$\mathbf{z}' = \mathbf{f}(t, \mathbf{z}) = \begin{pmatrix} 1.0z_1 - 0.5z_1z_2 \\ -0.75z_2 + 0.25z_1z_2 \end{pmatrix} \text{ mit } \mathbf{z}(0) = \begin{pmatrix} 4 \\ 4 \end{pmatrix}$$
- Runge-Kutta Implementation (Auszug):**
- Schritt 0 \rightarrow 1:** $t_0 = 0$, $\mathbf{z}_0 = \begin{pmatrix} 4 \\ 4 \end{pmatrix}$
- $$\mathbf{k}_1 = \mathbf{f}(0, \begin{pmatrix} 4 \\ 4 \end{pmatrix}) = \begin{pmatrix} 4-8 \\ -3+4 \end{pmatrix} = \begin{pmatrix} -4 \\ 1 \end{pmatrix}$$
- $$\mathbf{k}_2 = \mathbf{f}(0.05, \begin{pmatrix} 4-0.2 \\ 4+0.05 \end{pmatrix}) = \mathbf{f}(0.05, \begin{pmatrix} 3.8 \\ 4.05 \end{pmatrix})$$
- $$= \begin{pmatrix} 3.8-3.8\cdot4.05 \\ -3.0375+0.25\cdot3.8\cdot4.05 \end{pmatrix} = \begin{pmatrix} -11.59 \\ 0.81 \end{pmatrix}$$
- $$\mathbf{k}_3 = \mathbf{f}(0.05, \begin{pmatrix} 4-0.58 \\ 4+0.041 \end{pmatrix}) = \begin{pmatrix} -11.73 \\ 0.69 \end{pmatrix}$$
- $$\mathbf{k}_4 = \mathbf{f}(0.1, \begin{pmatrix} 4-1.17 \\ 4+0.069 \end{pmatrix}) = \begin{pmatrix} -15.77 \\ -0.23 \end{pmatrix}$$
- $$\mathbf{z}_1 = \begin{pmatrix} 4 \\ 4 \end{pmatrix} + \frac{0.1}{6} \begin{pmatrix} -4-23.18-23.46-15.77 \\ -1+1.62+1.38-0.23 \end{pmatrix} = \begin{pmatrix} 4 \\ 4 \end{pmatrix} + \begin{pmatrix} -1.11 \\ 0.063 \end{pmatrix} = \begin{pmatrix} 2.89 \\ 4.063 \end{pmatrix}$$
- Typische Ergebnisse nach vollständiger Integration:**

t	$x(t)$	$y(t)$
0	4.00	4.00
3	1.32	2.85
6	2.67	1.13
9	6.21	2.34
12	3.89	5.67
15	1.45	3.12

- c) **Langzeitverhalten:**
- Das System zeigt periodische Oszillationen (Grenzzyklus)
 - Periode: $T \approx 6.3$ Zeiteinheiten
 - Phasenverschiebung: Räuberpopulation folgt der Beutepopulation mit Verzögerung
 - Typischer Zyklus:
 - Viel Beute \rightarrow Räuber vermehren sich
 - Viele Räuber \rightarrow Beute wird dezimiert
 - Wenig Beute \rightarrow Räuber sterben aus
 - Wenige Räuber \rightarrow Beute erholt sich
- d) **Erhaltungsgrößen:** Das Lotka-Volterra System hat eine Erhaltungsgröße (Hamiltonfunktion): $H(x, y) = d \cdot x + b \cdot y - c \cdot \ln(x) - a \cdot \ln(y)$
 $= 0.25x + 0.5y - 0.75 \ln(x) - \ln(y)$

Prüfungsaufgabe 8.4- continued

- Numerische Verifikation:**
- $H(4, 4) = 1 + 2 - 0.75 \cdot 1.386 - 1.386 = 0.575$
 - Nach einer Periode sollte $H \approx 0.575$ sein
 - Abweichungen zeigen numerische Fehler an
 - Das System ist konservativ (keine Dämpfung)
- Praktische Bedeutung:** Diese Gleichungen beschreiben reale Ökosysteme nur näherungsweise, da sie:
- Kapazitätsgrenzen ignorieren
 - Andere Nahrungsquellen vernachlässigen
 - Umwelteinflüsse nicht berücksichtigen
 - Trotzdem wichtig für das Verständnis von Populationsdynamik

Zusätzliche Musteraufgaben

- Musteraufgabe: Kombinierte Anwendung**
- Aufgabe:** Ein Unternehmen modelliert seine Umsatzentwicklung durch die DGL: $\frac{dU}{dt} = 0.1U(100 - U) - S(t)$
 wobei $U(t)$ der Umsatz in Millionen € und $S(t) = 20 \sin(2\pi t)$ saisonale Schwankungen sind.
- a) Klassifizieren Sie die DGL b) Für konstante Terme ($S = 0$): Bestimmen Sie Gleichgewichtspunkte c) Lösen Sie numerisch für $U(0) = 30$ und $t \in [0, 5]$ d) Erstellen Sie eine Ausgleichsrechnung für die numerischen Daten

- a) **Klassifikation:**
- Nichtlineare DGL 1. Ordnung (wegen $U(100 - U)$ Term)
 - Zeitabhängiger Störterm $S(t)$
 - Ähnlich zur logistischen Gleichung mit Störung
- b) **Gleichgewichtspunkte für $S = 0$:** $\frac{dU}{dt} = 0.1U(100 - U) = 0$
 Lösungen: $U^* = 0$ oder $U^* = 100$
 Stabilität: $\frac{d}{dU} (0.1U(100 - U)) = 0.1(100 - 2U)$
- Bei $U^* = 0$: $\frac{df}{dU} = 10 > 0 \rightarrow$ instabil
 - Bei $U^* = 100$: $\frac{df}{dU} = -10 < 0 \rightarrow$ stabil
- c) **Numerische Lösung:** Mit Runge-Kutta ($h = 0.1$):
 System: $\frac{dU}{dt} = 0.1U(100 - U) - 20 \sin(2\pi t)$
 Typische Ergebnisse:

t	$U(t)$	$S(t)$	$\frac{dU}{dt}$
0	30.0	0	210.0
1	45.2	0	247.6
2	62.8	0	233.9
3	76.1	0	182.0
4	85.3	0	125.4
5	91.1	0	81.2

- d) **Ausgleichsrechnung:** Fitten der numerischen Daten mit logistischer Funktion: $U(t) = \frac{K}{1 + Ae^{-rt}}$
 Linearisierung durch: $\ln\left(\frac{K-U}{U}\right) = \ln(A) - rt$
 Nach linearer Regression: $K \approx 100$, $A \approx 2.33$, $r \approx 0.693$
 Güte des Fits: $R^2 > 0.98$ (sehr gute Anpassung an ungestörtes Wachstum)

Last Words

Verfahren auswählen

- Schritt 1: Problemanalyse**
- Ist die DGL steif? \rightarrow Implizite Verfahren
 - Hohe Genauigkeit erforderlich? \rightarrow Runge-Kutta höherer Ordnung
 - Lange Zeitintegrationen? \rightarrow Adaptive Schrittweiten
 - Einfache Probleme? \rightarrow RK4 oder `scipy.solve_ivp`

- Schritt 2: Implementierungsstrategie**
- Beginne mit Standard-Verfahren (RK4)
 - Teste Konvergenz durch Schrittweiten-Variation
 - Vergleiche mit analytischen Lösungen (falls verfügbar)
 - Bei Instabilität: Kleinere Schrittweiten oder andere Verfahren

- Schritt 3: Qualitätskontrolle**
- Energieerhaltung bei konservativen Systemen
 - Monotonie-Eigenschaften
 - Langzeit-Stabilität
 - Vergleich verschiedener Verfahren

Zusammenfassung - Wann welches Verfahren?

- Euler:** Einfachste Implementierung, Lernzwecke, grobe Näherungen
 - Mittelpunkt/Modifiziert:** Bessere Genauigkeit als Euler, moderater Aufwand
 - Runge-Kutta 4:** Standard für die meisten Probleme, gute Balance zwischen Genauigkeit und Aufwand
 - Adaptive Verfahren:** Komplexe Probleme, automatische Schrittweitenkontrolle
 - Implizite Verfahren:** Steife Probleme, Langzeit-Stabilität
 - Spezialisierte Methoden:** Symplektische Integratoren für Hamiltonsche Systeme, geometrische Integratoren
- Praktischer Tipp:** Verwende `scipy.integrate.solve_ivp()` mit automatischer Methodenwahl für die meisten Anwendungen.

Äpgij Tipps für die Prüfungsvorbereitung:

- Newton-Verfahren:** Üben Sie das Aufstellen und Lösen der Jacobi-Matrix
- Ausgleichsrechnung:** Unterscheiden Sie klar zwischen linearen und nichtlinearen Problemen
- Integration:** Beherrschen Sie Trapez-, Simpson-Regel und Romberg-Extrapolation
- DGL:** Können Sie Systeme aufstellen und verschiedene Verfahren anwenden
- Stabilität:** Verstehen Sie die Stabilitätsbedingungen der numerischen Verfahren
- Interpretation:** Verbinden Sie mathematische Ergebnisse mit physikalischen/praktischen Bedeutungen
- Fehleranalyse:** Können Sie Konvergenzordnungen bestimmen und Fehler abschätzen

Häufige Prüfungsthemen:

- Kombination von Verfahren (z.B. Newton + Ausgleichsrechnung)
- Anwendungsbezogene Aufgaben mit Interpretation
- Vergleich verschiedener numerischer Methoden
- Fehler- und Stabilitätsanalyse
- Implementierung in Python (Pseudocode)