Höhere Mathematik

Jil Zerndt, Lucien Perret
January 2025

Additional Examples

Rechnerarithmetik

Werteberechnung ausführlich Gegeben sei die Maschinenzahl zur Basis $B = 2$:

$$x = \underbrace{0.1101}_{\text{n=4}} \cdot \underbrace{2_2^{101}}_{\text{l=3}}$$

1. Normalisierung prüfen:
- $m_1 = 1 \neq 0 \rightarrow$ normalisiert

2. Exponent berechnen:

$$\begin{aligned}\hat{e} &= 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 \\ &= 4 + 0 + 1 = 5\end{aligned}$$

3. Wert berechnen:

$$\begin{aligned}\hat{\omega} &= 1 \cdot 2^{5-1} + 1 \cdot 2^{5-2} + 0 \cdot 2^{5-3} + 1 \cdot 2^{5-4} \\ &= 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 \\ &= 16 + 8 + 0 + 2 \\ &= 26\end{aligned}$$

Also ist $x = 26$

Weitere Beispiele
1. Basis 10: $0.3141 \cdot 10^2$
   - Normalisiert, da $m_1 = 3 \neq 0$
   - $\hat{e} = 2$
   - $\hat{\omega} = 3 \cdot 10^1 + 1 \cdot 10^0 + 4 \cdot 10^{-1} + 1 \cdot 10^{-2} = 31.41$
2. Basis 16 (hex): $0.A5F \cdot 16^3$
   - Normalisiert, da $m_1 = A = 10 \neq 0$
   - $\hat{e} = 3$
   - $\hat{\omega} = 10 \cdot 16^2 + 5 \cdot 16^1 + 15 \cdot 16^0 = 2655$

Werteberechnung Berechnung einer Zahl zur Basis B=2:

$$\underbrace{0.1011}_{\text{n=4}} \cdot \underbrace{2^3}_{\text{l=1}}$$

1. Exponent: $\hat{e} = 3$
2. Wert: $\hat{\omega} = 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-1}$
   $= 4 + 0 + 1 + 0.5 = 5.5$

Numerische Lösung von Nullstellenproblemen

Fixpunktiteration Nullstellen von $p(x) = x^3 - x + 0.3$
Fixpunktgleichung: $x_{n+1} = F(x_n) = x_n^3 + 0.3$
1. $F'(x) = 3x^2$ steigt monoton
2. Für $I = [0, 0.5]$: $F(0) = 0.3 > 0$, $F(0.5) = 0.425 < 0.5$
3. $\alpha = \max_{x \in [0,0.5]} |3x^2| = 0.75 < 1$
4. Konvergenz für Startwerte in $[0, 0.5]$ gesichert

Newton-Verfahren Berechnung von $\sqrt[3]{2}$ Nullstellenproblem: $f(x) = x^3 - 2$
Ableitung: $f'(x) = 3x^2$, Startwert $x_0 = 1$
1. $x_1 = 1 - \frac{1^3-2}{3 \cdot 1^2} = 1.333333$
2. $x_2 = 1.333333 - \frac{1.333333^3-2}{3 \cdot 1.333333^2} = 1.259921$
3. $x_3 = 1.259921 - \frac{1.259921^3-2}{3 \cdot 1.259921^2} = 1.259921$

Quadratische Konvergenz sichtbar durch schnelle Annäherung an $\sqrt[3]{2} \approx 1.259921$

Numerische Lösung von LGS

Pivotisierung in der Praxis Betrachten Sie das System:

$$\begin{pmatrix} 0.001 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

Ohne Pivotisierung:
Division durch 0.001 führt zu großen Rundungsfehlern:

$$x_1 \approx 1000 \cdot (1 - x_2)$$

Mit Pivotisierung:
Nach Zeilenvertauschung:

$$\begin{pmatrix} 1 & 1 \\ 0.001 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

Liefert stabile Lösung: $x_1 = 1$, $x_2 = 1$

LR-Zerlegung mit Pivotisierung Gegeben sei das System:

$$A = \begin{pmatrix} 1 & 2 & 1 \\ 3 & 8 & 1 \\ 0 & 4 & 1 \end{pmatrix}, \quad b = \begin{pmatrix} 2 \\ 3 \\ 5 \end{pmatrix}$$

1. Erste Spalte
Max Element in 1. Spalte: $|a_{21}| = 3$, tausche Z1 und Z2:

$$P_1 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad A^{(1)} = \begin{pmatrix} 3 & 8 & 1 \\ 1 & 2 & 1 \\ 0 & 4 & 1 \end{pmatrix}$$

Eliminationsfaktoren: $l_{21} = \frac{1}{3}$, $l_{31} = 0$
Nach Elimination:

$$A^{(2)} = \begin{pmatrix} 3 & 8 & 1 \\ 0 & -\frac{2}{3} & \frac{2}{3} \\ 0 & 4 & 1 \end{pmatrix}$$

2. Zweite Spalte
Max Element: $|a_{32}| = 4$, tausche Z2 und Z3:

$$P_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Eliminationsfaktor: $l_{32} = -\frac{1}{6}$
Nach Elimination:

$$R = \begin{pmatrix} 3 & 8 & 1 \\ 0 & 4 & 1 \\ 0 & 0 & \frac{5}{6} \end{pmatrix}$$

Endergebnis

$$P = P_2 P_1 = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}, \quad L = \begin{pmatrix} 1 & 0 & 0 \\ \frac{1}{3} & 1 & 0 \\ 0 & -\frac{1}{6} & 1 \end{pmatrix}$$

Lösung des Systems
1. $Pb = \begin{pmatrix} 3 \\ 5 \\ 2 \end{pmatrix}$
2. $Ly = Pb$: $y = \begin{pmatrix} 3 \\ 4 \\ 1 \end{pmatrix}$
3. $Rx = y$: $x = \begin{pmatrix} 1 \\ 0 \\ \frac{6}{5} \end{pmatrix}$

gauss_elimination Gauss-Elimination mit Pivotisierung und freien Variablen

```python
def gauss_elimination(A, b, tol=1e-10):
    """Gauss-Elimination mit Behandlung freier
        Variablen

    Returns: Dictionary mit
        'solution': Basisloesung
        'free_vars': Liste der freien Variablen
        'general_solution': Lambda-Funktion fuer allg.
            Loesung
        'rank': Rang der Matrix
        'consistent': System loesbar
    """
    n = len(A)
    # Erweiterte Matrix erstellen
    M = copy_matrix(A)
    b = b.copy()

    # Speichere Zeilenoperationen fuer Rueckwaertsphase
    row_ops = []
    # Markiere freie Variablen
    free_vars = []
    rank = 0

    # Vorwaertselimination mit Pivotisierung
    for i in range(n):
        # Finde maximales Element in Spalte i
        pivot_row = i
        for j in range(i+1, n):
            if abs(M[j][i]) > abs(M[pivot_row][i]):
                pivot_row = j

        # Zeilen tauschen falls noetig
        if pivot_row != i:
            M[i], M[pivot_row] = M[pivot_row], M[i]
            b[i], b[pivot_row] = b[pivot_row], b[i]

        # Pruefe auf freie Variable
        if abs(M[i][i]) < tol:
            free_vars.append(i)
            continue

        rank += 1
        # Speichere Operationen
        row_ops.append((i, [(j, -M[j][i]/M[i][i])
                            for j in range(i+1, n)]))

        # Elimination durchfuehren
        for j in range(i+1, n):
            factor = M[j][i] / M[i][i]
            for k in range(i, n):
                M[j][k] -= factor * M[i][k]
            b[j] -= factor * b[i]

    # Pruefe Loesbarkeit
    consistent = True
    for i in range(rank, n):
        if abs(b[i]) > tol:
            consistent = False
            break

    if not consistent:
        return {
            'solution': None,
            'free_vars': [],
            'general_solution': None,
            'rank': rank,
            'consistent': False
        }

    # Berechne Basisloesung
    x = [0] * n
    # Rueckwaertssubstitution
    for i in range(n-1, -1, -1):
        if i not in free_vars:
            sum_val = sum(M[i][j] * x[j]
                          for j in range(i+1, n))
            x[i] = (b[i] - sum_val) / M[i][i]

    # Erstelle Funktion fuer allgemeine Loesung
    def general_solution(*params):
        if len(params) != len(free_vars):
            raise ValueError(
                f"Benoetigt {len(free_vars)}
                    Parameter")
        solution = x.copy()
        for var, param in zip(free_vars, params):
            solution[var] = param
            # Update abhaengige Variablen
            for op in row_ops:
                row, factors = op
                if row < var:
                    continue
                solution[row] -= param * sum(
                    f * solution[j]
                    for j, f in factors if j > var)
        return solution

    return {
        'solution': x,
        'free_vars': free_vars,
        'general_solution': general_solution,
        'rank': rank,
        'consistent': True,
        'matrix': M
    }

# Beispielnutzung:
"""
A = [
    [1, 2, 1],
    [2, 4, 2],
    [3, 6, 3]
]
b = [2, 4, 6]

result = gauss_elimination(A, b)
if result['consistent']:
    print(f"Basisloesung: {result['solution']}")
    print(f"Freie Variablen: {result['free_vars']}")
    if result['free_vars']:
        # Erzeuge spezielle Loesung
        print("Spezielle Loesung mit t=1:")
        print(result['general_solution'](1))
else:
    print("System nicht loesbar")
"""
```

Umgang mit Systemen mit freien Variablen

1. Vorgehensweise
   - Matrix auf Stufenform bringen
   - Freie Variablen identifizieren (Nullspalten)
   - Basislösung berechnen
   - Allgemeine Lösung parametrisch aufstellen
2. Interpretation
   - Rang der Matrix bestimmen
   - Lösbarkeit prüfen
   - Dimension des Lösungsraums bestimmen
   - Spezielle Lösungen generieren
3. Sonderfälle beachten
   - Unlösbare Systeme erkennen
   - Abhängige Gleichungen identifizieren
   - Numerische Genauigkeit berücksichtigen

QR-Zerlegung Gegeben sei die Matrix:

$$A = \begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}$$

1. Erste Spalte

$v_1 = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$, $\|v_1\| = \sqrt{2}$

Householder-Vektor: $w_1 = v_1 + \sqrt{2}\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1+\sqrt{2} \\ 1 \\ 0 \end{pmatrix}$

Normierung: $u_1 = \frac{1}{\sqrt{4+2\sqrt{2}}}\begin{pmatrix} 1+\sqrt{2} \\ 1 \\ 0 \end{pmatrix}$

Erste Householder-Matrix:

$$H_1 = I - 2u_1u_1^T = \begin{pmatrix} -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

2. Zweite Spalte

Nach Anwendung von H_1:

$$H_1A = \begin{pmatrix} -\sqrt{2} & -\frac{1}{\sqrt{2}} \\ 0 & \frac{1}{\sqrt{2}} \\ 0 & 1 \end{pmatrix}$$

Untervektor für zweite Transformation: $v_2 = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ 1 \end{pmatrix}$

Analog zur ersten Transformation erhält man:

$$H_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -\frac{1}{\sqrt{5}} & -\frac{2}{\sqrt{5}} \\ 0 & -\frac{2}{\sqrt{5}} & \frac{1}{\sqrt{5}} \end{pmatrix}$$

Endergebnis

$$Q = H_1^T H_2^T = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$R = H_2H_1A = \begin{pmatrix} \sqrt{2} & 1 \\ 0 & \sqrt{2} \\ 0 & 0 \end{pmatrix}$$

Verifikation

- $Q^TQ = QQ^T = I$ (Orthogonalität)
- $QR = A$ (bis auf Rundungsfehler)
- R ist obere Dreiecksmatrix

Iterative Verfahren Vergleich Jacobi und Gauss-Seidel System:

$$\begin{pmatrix} 4 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 4 \end{pmatrix} x = \begin{pmatrix} 1 \\ 5 \\ 0 \end{pmatrix}$$

k	Jacobi		Gauss-Seidel	
0	$(0,0,0)^T$		$(0,0,0)^T$	
1	$(0.25,1.25,0)^T$	1.25	$(0.25,1.31,0.08)^T$	1.31
2	$(0.31,1.31,0.31)^T$	0.31	$(0.33,1.33,0.33)^T$	0.02
3	$(0.33,1.33,0.33)^T$	0.02	$(0.33,1.33,0.33)^T$	0.00

Eigenvektoren und Eigenwerte

Darstellungsformen Gegeben: $z = 3 - 11i$ in Normalform

$$r = \sqrt{3^2 + 11^2} = \sqrt{130}, \quad \varphi = \arcsin(\frac{11}{\sqrt{130}}) = 1.3\text{rad} = 74.74^\circ$$

Trigonometrische Form: $z = \sqrt{130}(\cos(1.3) + i\sin(1.3))$

Exponentialform: $z = \sqrt{130}e^{i\cdot 1.3}$

Eigenwertberechnung $A = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 3 & 0 \\ 0 & 1 & 2 \end{pmatrix}$

1. Da A eine Dreiecksmatrix ist, sind die Diagonalelemente die Eigenwerte: $\lambda_1 = 1, \lambda_2 = 3, \lambda_3 = 2$
2. $\det(A) = \lambda_1 \cdot \lambda_2 \cdot \lambda_3 = 6$
3. $\text{tr}(A) = \lambda_1 + \lambda_2 + \lambda_3 = 6$
4. Spektrum: $\sigma(A) = \{1, 2, 3\}$

Von-Mises-Iteration Berechne größten Eigenwert der Matrix:

$A = \begin{pmatrix} 4 & -1 & 1 \\ -1 & 3 & -2 \\ 1 & -2 & 3 \end{pmatrix}$, Startvektor: $v^{(0)} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$

k	$v^{(k)}$	$\lambda^{(k)}$
0	$(1,0,0)^T$	-
1	$(0.970,-0.213,0.119)^T$	4.000
2	$(0.957,-0.239,0.164)^T$	4.827
3	$(0.953,-0.244,0.178)^T$	4.953
4	$(0.952,-0.245,0.182)^T$	4.989

Konvergenz gegen $\lambda_1 \approx 5$
Eigenvektor $v \approx (0.952,-0.245,0.182)^T$

QR-Verfahren Matrix:

$$A = \begin{pmatrix} 2 & -1 & 1 \\ -1 & 3 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

QR-Iteration:

1. $A_0 = A$
2. Nach erster Iteration:

$$A_1 = \begin{pmatrix} 3.21 & -0.83 & 0.62 \\ -0.83 & 2.13 & 0.41 \\ 0.62 & 0.41 & 0.66 \end{pmatrix}$$

3. Nach 5 Iterationen:

$$A_5 \approx \begin{pmatrix} 4 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Die Diagonalelemente von A_5 sind die Eigenwerte: $\lambda_1 = 4, \lambda_2 = 1, \lambda_3 = 1$