

HM2 Summary (beliebig lang)

23. Februar, 2024; rev. 12. Juni 2024

Linda Riesen (rieselin)

1 Vorlesung 01

Numerik nichtlinearer Gleichungssysteme

Eine Funktion
 $f: \mathbb{R}^n \rightarrow \mathbb{R}, (x_1, x_2, \dots, x_n) \mapsto y = f(x_1, x_2, \dots, x_n)$
 heisst Skalarwertige Funktion in n Variablen x_1, x_2, \dots, x_n .

Abbildung 1: Definition Skalarwertige Funktion

Eine Funktion
 $f: \mathbb{R}^n \rightarrow \mathbb{R}^m, \underline{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \mapsto \underline{y} = f(\underline{x}) = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} f_1(x_1, x_2, \dots, x_n) \\ f_2(x_1, x_2, \dots, x_n) \\ \vdots \\ f_m(x_1, x_2, \dots, x_n) \end{bmatrix}$
 heisst vektorwertige Funktion in n Variablen x_1, x_2, \dots, x_n .

Abbildung 2: Definition Vektorwertige Funktion

2 Vorlesung 02

2.1 Ableitungen von Funktionen mit mehreren Variablen

Partielle Ableitung: Alle anderen Variablen werden als Konstanten betrachtet und dann nach x_i abgeleitet, Graphisch ist dies die Steigung des Graphen von f in Richtung x_i

Es sei
 $f: \mathbb{R}^n \rightarrow \mathbb{R}^m, \underline{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \mapsto \underline{y} = f(\underline{x}) = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} f_1(x_1, \dots, x_n) \\ \vdots \\ f_m(x_1, \dots, x_n) \end{bmatrix}$
 eine vektorwertige Funktion in n Variablen.
 Wenn die m Skalarwertigen Funktionen f_1 bis f_m nach den Variablen x_1 bis x_n differenzierbar sind, so gibt es $m \cdot n$ partielle Ableitungen
 $\frac{\partial f_i}{\partial x_j}(x_1, \dots, x_n)$ mit $i=1, \dots, m$ und $j=1, \dots, n$.
 Sie werden zu einer $m \times n$ -Matrix zusammengefasst

$$\begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \dots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \dots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \dots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

 Diese Matrix heisst Jacobi-Matrix von f und wird mit Df bezeichnet.

Abbildung 3: Jacobi Matrix

2.2 Linearisierung von Funktionen mit mehreren Variablen

Es seien
 $f: \mathbb{R}^n \rightarrow \mathbb{R}^m, \underline{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \mapsto \underline{y} = f(\underline{x}) = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} f_1(x_1, \dots, x_n) \\ \vdots \\ f_m(x_1, \dots, x_n) \end{bmatrix}$
 eine vektorwertige Funktion in n Variablen und
 $\underline{x}^{(0)} = \begin{bmatrix} x_1^{(0)} \\ \vdots \\ x_n^{(0)} \end{bmatrix}$
 ein fester Stelle.
 Dann heisst die Funktion
 $g: \mathbb{R}^n \rightarrow \mathbb{R}^m, \underline{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \mapsto \underline{y} = g(\underline{x}) = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} g_1(x_1, \dots, x_n) \\ \vdots \\ g_m(x_1, \dots, x_n) \end{bmatrix}$
 mit
 $g(\underline{x}) = f(\underline{x}^{(0)}) + Df(\underline{x}^{(0)}) \cdot (\underline{x} - \underline{x}^{(0)})$
 Linearisierung von f an der Stelle $\underline{x}^{(0)}$. Dabei ist

$$Df(\underline{x}^{(0)}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(\underline{x}^{(0)}) & \dots & \frac{\partial f_1}{\partial x_n}(\underline{x}^{(0)}) \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1}(\underline{x}^{(0)}) & \dots & \frac{\partial f_m}{\partial x_n}(\underline{x}^{(0)}) \end{bmatrix}$$

 die Jacobi-Matrix von f an der Stelle $\underline{x}^{(0)}$.

Abbildung 4: Definition Linearisierung mit skalarwertigen Funktionen

Graphisch: Graph g (Linearisierung von f an Stelle $\underline{x}^{(0)}$) ist n -dimensionale Tangentialebene an f

3 Vorlesung 03

3.1 Nichtlineare Gleichungssysteme als Nullstellenprobleme von Funktionen mit mehreren Variablen

Das Lösen eines Systems
 $f_1(x_1, \dots, x_n) = 0$
 $f_2(x_1, \dots, x_n) = 0$
 \vdots
 $f_n(x_1, \dots, x_n) = 0$
 aus n nichtlinearen Gleichungen mit n Unbekannten x_1, \dots, x_n
 entspricht der Nullstellenbestimmung der Vektorwertigen Funktion in n Variablen
 $f: \mathbb{R}^n \rightarrow \mathbb{R}^n, \underline{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \mapsto \underline{y} = f(\underline{x}) = \begin{bmatrix} f_1(x_1, \dots, x_n) \\ \vdots \\ f_n(x_1, \dots, x_n) \end{bmatrix}$

Abbildung 5: Nichtlineare Gleichungssysteme als Nullstellenprobleme v Funktionen mit mehreren Variablen

3.2 Newton-Verfahren zur Nullstellenbestimmung von Funktionen mit mehreren Variablen

Mehrdimensionales Newton-Verfahren (Version mit Matrixinversion)
 • Gesucht ist eine Nullstelle $\underline{x}^* \in \mathbb{R}^n$ einer Funktion
 $f: \mathbb{R}^n \rightarrow \mathbb{R}^n, \underline{x} \mapsto f(\underline{x})$
 • Wähle Startwert $\underline{x}^{(0)}$ nahe \underline{x}^*
 • Bilde für $i = 0, 1, 2, \dots$
 $\underline{x}^{(i+1)} = \underline{x}^{(i)} - (Df(\underline{x}^{(i)}))^{-1} \cdot f(\underline{x}^{(i)})$

Abbildung 6: Mehrdimensionales Newton Verfahren

Leitbeispiel
 $f: \mathbb{R}^2 \rightarrow \mathbb{R}^2, \underline{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \mapsto f(\underline{x}) = \begin{bmatrix} f_1(x_1, x_2) \\ f_2(x_1, x_2) \end{bmatrix} = \begin{bmatrix} x_1^2 + x_2 - 11 \\ x_1 + x_2^2 - 7 \end{bmatrix}$
 a) Fülle mit Startwert $\underline{x}^{(0)} = \begin{bmatrix} 4 \\ 1 \end{bmatrix}$ einen Schritt des Newton-Verfahrens von Hand durch:
 b) Fülle mit Python vier weitere Schritte durch.
 c) Das Newton-Verfahren konvergiert gegen die Nullstelle
 $\underline{x}^* = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$. Bestimme mit Python $\|\underline{x}^{(i)} - \underline{x}^*\|$ für $i = 0, 1, 2, 3, 4, 5$.
 Wie schnell konvergiert das Newton-Verfahren?
 a) $f(\underline{x}^{(0)}) = \begin{bmatrix} 4^2 + 1 - 11 \\ 4 + 1^2 - 7 \end{bmatrix} = \begin{bmatrix} -5 \\ -6 \end{bmatrix}$
 $Df(\underline{x}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 2x_1 & 1 \\ 1 & 2x_2 \end{bmatrix}$
 $Df(\underline{x}^{(0)}) = \begin{bmatrix} 2 \cdot 4 & 1 \\ 1 & 2 \cdot 1 \end{bmatrix} = \begin{bmatrix} 8 & 1 \\ 1 & 2 \end{bmatrix}$
 $(Df(\underline{x}^{(0)}))^{-1} = \frac{1}{22 - 1 \cdot 1} \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix} = \frac{1}{21} \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}$
 $\underline{x}^{(1)} = \underline{x}^{(0)} - (Df(\underline{x}^{(0)}))^{-1} \cdot f(\underline{x}^{(0)}) = \begin{bmatrix} 4 \\ 1 \end{bmatrix} - \frac{1}{21} \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} -5 \\ -6 \end{bmatrix} = \begin{bmatrix} 46/21 \\ 4/3 \end{bmatrix}$
 $\begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} -5 \\ -6 \end{bmatrix} = \begin{bmatrix} -10 + 6 \\ 5 - 12 \end{bmatrix} = \begin{bmatrix} -4 \\ -7 \end{bmatrix}$
 $\frac{1}{21} \begin{bmatrix} -4 \\ -7 \end{bmatrix} = \begin{bmatrix} -4/21 \\ -1/3 \end{bmatrix}$

Abbildung 7: Leitbeispiel Newtonverfahren

Vereinfachtes Newton-Verfahren
 • Gesucht ist eine Nullstelle $\underline{x}^* \in \mathbb{R}^n$ einer Funktion
 $f: \mathbb{R}^n \rightarrow \mathbb{R}^n, \underline{x} \mapsto f(\underline{x})$
 • Wähle Startwert $\underline{x}^{(0)}$ nahe \underline{x}^* und berechne
 $D = Df(\underline{x}^{(0)})$
 • Berechne für $i = 0, 1, 2, \dots$ den Zuwachs
 $\underline{\Delta}^{(i)} = \underline{x}^{(i+1)} - \underline{x}^{(i)}$
 durch Lösen des linearen Gleichungssystems
 $D \cdot \underline{\Delta}^{(i)} = -f(\underline{x}^{(i)})$
 und setze
 $\underline{x}^{(i+1)} = \underline{x}^{(i)} + \underline{\Delta}^{(i)}$
 Leitbeispiel
 $f: \mathbb{R}^2 \rightarrow \mathbb{R}^2, \underline{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \mapsto f(\underline{x}) = \begin{bmatrix} f_1(x_1, x_2) \\ f_2(x_1, x_2) \end{bmatrix} = \begin{bmatrix} x_1^2 + x_2 - 11 \\ x_1 + x_2^2 - 7 \end{bmatrix}, \underline{x}^* = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$

Abbildung 8: Vereinfachtes Newtonverfahren

3.3 Newton Verfahren mit Dämpfung

Mehrdimensionales Newton-Verfahren mit Dämpfung

- Gesucht ist eine Nullstelle $\underline{x}^* \in \mathbb{R}^n$ einer Funktion $f: \mathbb{R}^n \rightarrow \mathbb{R}^m, \underline{x} \mapsto f(\underline{x})$.
- Wähle Startwert $\underline{x}^{(0)}$ nahe \underline{x}^*
- Berechne $\underline{\Delta}^{(i)}$ für $i=0,1,2,\dots$ durch Lösen von $Df(\underline{x}^{(i)}) \cdot \underline{\Delta}^{(i)} = -f(\underline{x}^{(i)})$
- Wähle positive ganze Zahl k_{\max} . Ermittle das kleinste k im Bereich $0,1,2,\dots,k_{\max}$ mit $\|f(\underline{x}^{(i)} + \frac{\underline{\Delta}^{(i)}}{2^k})\|_2 < \|f(\underline{x}^{(i)})\|_2$
- Wenn es kein solches k gibt, rechne mit $k=0$ weiter.
- Setze $\underline{x}^{(i+1)} = \underline{x}^{(i)} + \frac{\underline{\Delta}^{(i)}}{2^k}$

Abbildung 9: Mehrdimensionales Newtonverfahren Mit Dämpfung

4 Vorlesung 04

4.1 Ausgleichsrechnung Problemstellung

Gesucht: Funktion deren Graph die gegebenen Wertepaare (/Stützpunkte) möglichst gut approximiert
Mögliche Lösungen

- Interpolation (Graph geht durch alle Stützpunkte)
 - Stückweise lineare Funktion (welche durch alle Stützpunkte geht)
 - Polynomfunktion (welche durch alle Stützpunkte geht)
- Regression (Graph nähert alle Stützpunkte möglichst gut an)
 - Lineare Funktion (welche Stützpunkte möglichst gut approximiert)
 - Sinusfunktion (welche Stützpunkte möglichst gut approximiert)

4.2 Ausgleichsrechnung Polynominterpolation

Zu $n+1$ Wertepaaren $(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$ mit x_0, x_1, \dots, x_n alle verschieden gibt es genau ein Polynom vom Grad kleiner oder gleich n mit $p(x_0) = y_0, p(x_1) = y_1, \dots, p(x_n) = y_n$.
Dieses Polynom heißt Interpolationspolynom zu den Wertepaaren.

Abbildung 10: Polynominterpolation

Zu Wertepaaren $(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$ mit x_0, x_1, \dots, x_n alle verschieden sind die Koeffizienten $c_0, c_1, c_2, \dots, c_n$ des Interpolationspolynoms $p(x) = c_0 + c_1 x + c_2 x^2 + \dots + c_n x^n$ Lösung des linearen Gleichungssystems

$$\begin{matrix} c_0 + c_1 x_0 + c_2 x_0^2 + \dots + c_n x_0^n = y_0 \\ c_0 + c_1 x_1 + c_2 x_1^2 + \dots + c_n x_1^n = y_1 \\ \vdots \\ c_0 + c_1 x_n + c_2 x_n^2 + \dots + c_n x_n^n = y_n \end{matrix}$$

bzw.

$$\begin{bmatrix} 1 & x_0 & x_0^2 & \dots & x_0^n \\ 1 & x_1 & x_1^2 & \dots & x_1^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^n \end{bmatrix} \cdot \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}$$

V heißt auch Vandermonde-Matrix.

Abbildung 11: Gleichungssystem Methode, Vandermonde Matrix

Die Vandermonde Matrix ist i.d.Regel sehr schlecht konditioniert daher ist Berechnung von Interpolationspolynom durch lösen des linearen Gleichungssystems stark fehlerbehaftet.

Zu Wertepaaren $(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$ mit x_0, x_1, \dots, x_n alle verschieden heißt das Interpolationspolynom die Form $p(x) = y_0 l_0(x) + y_1 l_1(x) + \dots + y_n l_n(x)$ mit

$$l_0(x) = \frac{(x-x_1) \cdot (x-x_2) \cdot \dots \cdot (x-x_n)}{(x_0-x_1) \cdot (x_0-x_2) \cdot \dots \cdot (x_0-x_n)}$$

$$l_1(x) = \frac{(x-x_0) \cdot (x-x_2) \cdot \dots \cdot (x-x_n)}{(x_1-x_0) \cdot (x_1-x_2) \cdot \dots \cdot (x_1-x_n)}$$

$$\vdots$$

$$l_n(x) = \frac{(x-x_0) \cdot (x-x_1) \cdot \dots \cdot (x-x_{n-1})}{(x_n-x_0) \cdot (x_n-x_1) \cdot \dots \cdot (x_n-x_{n-1})}$$

$l_0(x), l_1(x), \dots, l_n(x)$ heißen Lagrange-Polynome.

Abbildung 12: Lagrange Formel, Lagrange Polynome

Es seien
 $f: \mathbb{R} \rightarrow \mathbb{R}$ eine n-mal stetig differenzierbare Funktion,
 $x_0 < x_1 < \dots < x_n$ irgendwelche Stellen,
 $y_0 = f(x_0), y_1 = f(x_1), \dots, y_n = f(x_n)$ die zugehörigen Funktionswerte.
 Weiter sei
 $p: \mathbb{R} \rightarrow \mathbb{R}$ das Interpolationspolynom zu den Stützpunkten
 $(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$.
 Dann gilt:
 $|f(x) - p(x)| \leq \frac{|(x-x_0)(x-x_1)\dots(x-x_n)|}{(n+1)!} \cdot \max_{x_0 \leq \xi \leq x_n} |f^{(n+1)}(\xi)|$
 für alle x mit $x_0 \leq x \leq x_n$.

Abbildung 13: Fehlerabschätzung Polynominterpolation

5 Vorlesung 05

5.1 Splineinterpolation

Stückweise Interpolation der Daten durch 4 Polynome von Grad 3 mit jeweils gleicher Steigung und gleicher Krümmung an Übergängen
 Natürliche Kubische Splineinterpolation

① $S_i(x_i) = y_i$ und $S_i(x_{i+1}) = y_{i+1}$
 für $i=0, 1, \dots, n-1$ Interpolation bei allen Stützpunkten

② $S_i'(x_{i+1}) = S_{i+1}'(x_{i+1})$
 $S_i''(x_{i+1}) = S_{i+1}''(x_{i+1})$
 für $i=0, 1, \dots, n-2$ Gleiche Steigung und gleiche Krümmung bei Übergängen von einem Polynom zum nächsten

③ $S_0''(x_0) = 0$ und $S_{n-1}''(x_n) = 0$
 natürliche Randbedingung keine Krümmung am ersten und letzten Stützpunkt

• Bemerkung
 n Polynome vom Grad 3 \rightarrow $4n$ unbekannte Koeffizienten
 Bedingung ① \rightarrow $2 \cdot n$ Gleichungen
 Bedingung ② \rightarrow $2 \cdot (n-1)$ Gleichungen
 Bedingung ③ \rightarrow 2 Gleichungen
 total $4n$ Gleichungen

Abbildung 14: Bedingungen Spline Interpolation

Gegeben sind $n+1$ Stützpunkte
 $(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$ mit $x_0 < x_1 < \dots < x_n$.
 Gesucht ist die natürliche kubische Splinefunktion $S(x)$ zu den Stützpunkten. Es ist
 $S(x) = \begin{cases} S_0(x) = a_0 + b_1(x-x_0) + c_1(x-x_0)^2 + d_1(x-x_0)^3 & \text{für } x_0 \leq x \leq x_1 \\ S_1(x) = a_1 + b_2(x-x_1) + c_2(x-x_1)^2 + d_2(x-x_1)^3 & \text{für } x_1 \leq x \leq x_2 \\ \vdots \\ S_{n-1}(x) = a_{n-1} + b_n(x-x_{n-1}) + c_n(x-x_{n-1})^2 + d_n(x-x_{n-1})^3 & \text{für } x_{n-1} \leq x \leq x_n \end{cases}$
 Denn können die Koeffizienten a_i, b_i, c_i, d_i der Polynome $S_i(x)$ für $i=0, 1, \dots, n-1$ wie folgt berechnet werden:
 ① Setze $h_i = x_{i+1} - x_i$
 ② Setze $a_i = y_i$
 ③ Setze $c_0 = 0$
 Berechne c_1, c_2, \dots, c_{n-1} aus dem linearen Gleichungssystem
 $A \cdot c = z$
 mit der tridiagonalen Matrix
 $A = \begin{bmatrix} 2(h_0+h_1) & h_1 & & & \\ h_1 & 2(h_1+h_2) & h_2 & & \\ & h_2 & 2(h_2+h_3) & h_3 & \\ & & & \ddots & \ddots \\ & & & & h_{n-2} & 2(h_{n-2}+h_{n-1}) \end{bmatrix}$
 sowie den Vektoren
 $c = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_{n-1} \end{bmatrix}$ und $z = \begin{bmatrix} \frac{3(y_2-y_0)}{h_1} - \frac{3(y_1-y_0)}{h_0} \\ \frac{3(y_2-y_1)}{h_2} - \frac{3(y_1-y_0)}{h_1} \\ \vdots \\ \frac{3(y_n-y_{n-2})}{h_{n-1}} - \frac{3(y_{n-1}-y_{n-2})}{h_{n-2}} \end{bmatrix}$
 für zudem zusätzlich $c_n = 0$ ein, um nachfolgende Berechnungsformeln zu vereinfachen
 ④ $b_i = \frac{y_{i+1} - y_i}{h_i} - \frac{(c_{i+1} + 2c_i) \cdot h_i}{3}$
 ⑤ $d_i = \frac{c_{i+1} - c_i}{3h_i}$

Abbildung 15: Lösung Spline Interpolation

5.2 Grundlagen der Regression

5.2.1 Problemstellung:

Gegeben: Wertepaare/Datenpunkte, Menge/Klasse von Funktionen (f)
 Gesucht: Funktion welche Wertepaare bestmöglich approximiert sodass:
 $E(f) = (y_1 - f(x_1))^2 + \dots + (y_n - f(x_n))^2$

5.2.2 Begriffe

- E = Fehlerfunktional

- Zugelassene Funktionen (f): Ansatzfunktionen
- Funktion f für welche E am kleinsten: Ausgleichs- / Regressionsfunktion, besitzt das kleinste Fehlerquadrat

5.2.3 Ausgleichsfunktion

Normalerweise geht Ausgleichsfunktion nicht durch alle Datenpunkte (Fehlerfunktional ist positiv)

Im Spezialfall geht Ausgleichsfunktion durch alle Datenpunkte: Interpolationsfunktion (Fehlerfunktional = 0)

lineare / nicht lineare Ausgleichsprobleme linear (ist Linearkombination): Ansatzfunktion von der Form: $f(x) = \lambda_1 * f_1(x) + \dots + \lambda_m * f_m(x)$ mit festen Basisfunktionen $f_1(x) \dots f_m(x)$ aus R mit variablem Parametern λ
 nicht linear (ist keine Linearkombination): keine Linearkombination von Basisfunktionen

7 Vorlesung 08

Bei einem **nichtlinearen** Ausgleichsproblem mit Wertepaaren $(x_1, y_1), \dots, (x_n, y_n)$ hat die Ansatzfunktion $f(x; \lambda_1, \dots, \lambda_m)$ das kleinste Fehlerfunktional $E(\lambda_1, \dots, \lambda_m) = (y_1 - f(x_1; \lambda_1, \dots, \lambda_m))^2 + \dots + (y_n - f(x_n; \lambda_1, \dots, \lambda_m))^2$ wenn die Parameter $\lambda_1, \dots, \lambda_m$ Lösung des **wichtigeren** Gleichungssystems $\frac{\partial E}{\partial \lambda_1}(\lambda_1, \dots, \lambda_m) = 0$
 \vdots
 $\frac{\partial E}{\partial \lambda_m}(\lambda_1, \dots, \lambda_m) = 0$ sind.

Abbildung 17: Nichtlineares Ausgleichsproblem

6 Vorlesung 06

Bei einem **linearen** Ausgleichsproblem mit Datenpaaren $(x_1, y_1), \dots, (x_n, y_n)$ hat die Ansatzfunktion $f(x; \lambda_1, \dots, \lambda_m) = \lambda_1 * f_1(x) + \dots + \lambda_m * f_m(x)$ das kleinste Fehlerfunktional $E(\lambda_1, \dots, \lambda_m) = (y_1 - f(x_1; \lambda_1, \dots, \lambda_m))^2 + \dots + (y_n - f(x_n; \lambda_1, \dots, \lambda_m))^2$ wenn die Parameter $\lambda_1, \dots, \lambda_m$ Lösung des Gleichungssystems $\frac{\partial E}{\partial \lambda_1}(\lambda_1, \dots, \lambda_m) = 0$
 \vdots
 $\frac{\partial E}{\partial \lambda_m}(\lambda_1, \dots, \lambda_m) = 0$ sind. Dieses Gleichungssystem ist **linear** und lautet in Matrix-Vektor-Form $A^T \cdot \lambda = A^T \cdot y$ mit $A = \begin{bmatrix} f_1(x_1) & \dots & f_m(x_1) \\ \vdots & & \vdots \\ f_1(x_n) & \dots & f_m(x_n) \end{bmatrix}$, $\lambda = \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_m \end{bmatrix}$, $y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$. Es heißt **Normalgleichungssystem**.

Abbildung 16: Lösungsverfahren Lineare Ausgleichsprobleme

• Gegeben sind Wertepaare $(x_1, y_1), \dots, (x_n, y_n)$, Ansatzfunktionen $f(x; \lambda_1, \dots, \lambda_m)$.
 • Gesucht sind $\lambda_1, \dots, \lambda_m$, so dass $E(\lambda_1, \dots, \lambda_m) = (y_1 - f(x_1; \lambda_1, \dots, \lambda_m))^2 + \dots + (y_n - f(x_n; \lambda_1, \dots, \lambda_m))^2$ minimal ist.
 • Es seien $\lambda = \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_m \end{bmatrix}$, $\underline{g}(\lambda) = \begin{bmatrix} y_1 - f(x_1; \lambda_1, \dots, \lambda_m) \\ \vdots \\ y_n - f(x_n; \lambda_1, \dots, \lambda_m) \end{bmatrix}$, $D_{\underline{g}}(\lambda) = \text{Jacobi-Matrix von } \underline{g}(\lambda)$.
 Dann ist $E(\lambda) = \|\underline{g}(\lambda)\|_2^2$.
 • Ausgehend von einem Startvektor $\lambda^{(0)}$ mit $E(\lambda^{(0)})$ in der Nähe vom minimalen Wert von E mache für $k = 0, 1, 2, \dots$ Folgendes:
 - Berechne $\underline{\lambda}^{(k)}$ als Lösung des linearen Gleichungssystems $D_{\underline{g}}(\lambda^{(k)})^T \cdot D_{\underline{g}}(\lambda^{(k)}) \cdot \underline{\lambda}^{(k)} = -D_{\underline{g}}(\lambda^{(k)})^T \cdot \underline{g}(\lambda^{(k)})$
 - Setze $\lambda^{(k+1)} = \lambda^{(k)} + \underline{\lambda}^{(k)}$

Abbildung 18: Gauss Newton Verfahren ohne Dämpfung

• Gesucht sind Wertepaar $(x_j, y_j), \dots, (x_n, y_n)$, Ansatzfunktionen $f(x; \lambda_1, \dots, \lambda_m)$.
 • Gesucht sind $\lambda_1, \dots, \lambda_m$, sodass $E(\lambda_1, \dots, \lambda_m) = (y_1 - f(x_1; \lambda_1, \dots, \lambda_m))^2 + \dots + (y_n - f(x_n; \lambda_1, \dots, \lambda_m))^2$ minimal ist.
 • Es seien $\lambda = \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_m \end{bmatrix}$, $\underline{g}(\lambda) = \begin{bmatrix} y_1 - f(x_1; \lambda_1, \dots, \lambda_m) \\ \vdots \\ y_n - f(x_n; \lambda_1, \dots, \lambda_m) \end{bmatrix}$, $\underline{D}\underline{g}(\lambda) =$ Jacobi-Matrix von $\underline{g}(\lambda)$.
 Dann ist $E(\lambda) = \|\underline{g}(\lambda)\|_2^2$.
 • Ausgehend von einem Startvektor $\lambda^{(0)}$ mit $E(\lambda^{(0)})$ in der Nähe vom minimalen Wert von E mache für $k = 0, 1, 2, \dots$ folgendes:
 - Bilde die QR-Zerlegung $\underline{D}\underline{g}(\lambda^{(k)}) = \underline{Q}^{(k)} \underline{R}^{(k)}$.
 - Berechne $\underline{d}^{(k)}$ als Lösung des linearen Gleichungssystems $\underline{R}^{(k)} \underline{d}^{(k)} = -(\underline{Q}^{(k)})^T \underline{g}(\lambda^{(k)})$.
 - Bestimme das kleinste k aus $0, 1, 2, \dots, k_{\max}$ mit $\|\underline{g}(\lambda^{(k)} + \frac{\underline{d}^{(k)}}{2^k})\|_2^2 < \|\underline{g}(\lambda^{(k)})\|_2^2$.
 mögliches $\lambda^{(k+1)}$ $E(\lambda^{(k)})$
 $E(\lambda^{(k+1)})$
 Falls es kein solches k gibt, setze $k=0$.
 - Setze $\lambda^{(k+1)} = \lambda^{(k)} + \frac{\underline{d}^{(k)}}{2^k}$.

Abbildung 19: Gauss Newton Verfahren mit Dämpfung

8 Vorlesung 09

Ziel: Möglichst genauer Näherungswert des bestimmten Integrals (oft nicht algebraisch berechenbar) (einer mindest. stetigen Funktion)

- **Rechteckregel:** Approximation von $f(x)$ durch **konstante** Funktion
- **Trapezregel:** Approximation von $f(x)$ durch **lineare** Funktion
- **Simpsonregel:** Approximation von $f(x)$ durch **quadratische** Funktion

8.1 Rechteckregel und Trapezregel

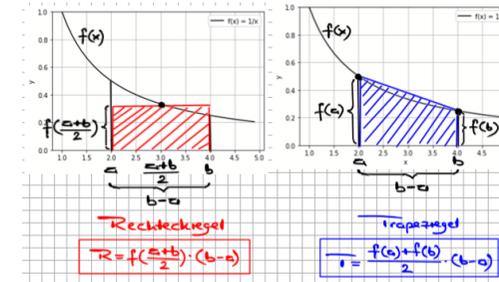


Abbildung 20: Rechteck- und Trapezregel

8.1.1 Summierte Rechteck- und Trapezregel

Summierte Rechteckregel: $R(h) = h * \sum_{i=0}^{n-1} f(x_i + \frac{h}{2})$ mit $[h = \frac{b-a}{n}, x_i = a + i * h]$

Summierte Trapezregel: $T(h) = h * (\frac{f(x_0)+f(x_n)}{2} + \sum_{i=0}^{n-1} f(x_i))$ mit $[h = \frac{b-a}{n}, x_i = a + i * h]$

8.2 Simpsonregel

Simpsonregel: $S = \frac{b-a}{3} * (\frac{f(a)}{2} + 2 * f(\frac{a+b}{2}) + \frac{f(b)}{2})$

8.2.1 Summierte Simpsonregel

Summierte Simpsonregel: $S(h) = \frac{h}{3} (\frac{f(x_0)+f(x_n)}{2} + \sum_{i=0}^{n-1} f(x_i) + 2 * \sum_{i=0}^{n-1} f(x_i + \frac{h}{2}))$ mit $[h = \frac{b-a}{n}, x_i = a + i * h]$

8.3 Fehlerabschätzung

Es seien
 $f: [a, b] \rightarrow \mathbb{R}$ viermal stetig differenzierbar,
 $I = \int_a^b f(x) dx$,
 $R(h)$ die Approximation von I mit Summierter Rechteckregel,
 $T(h)$ die Approximation von I mit Summierter Trapezregel,
 $S(h)$ die Approximation von I mit Summierter Simpson-Regel.
 Dann gilt:
 $|R(h) - I| \leq \frac{h^2}{24} \cdot (b-a) \cdot \max_{a \leq x \leq b} |f''(x)|$
 $|T(h) - I| \leq \frac{h^2}{12} \cdot (b-a) \cdot \max_{a \leq x \leq b} |f''(x)|$
 $|S(h) - I| \leq \frac{h^4}{2880} \cdot (b-a) \cdot \max_{a \leq x \leq b} |f^{(4)}(x)|$

Abbildung 21: Fehlerabschätzung

9 Vorlesung 10

9.1 Gauss-Formel

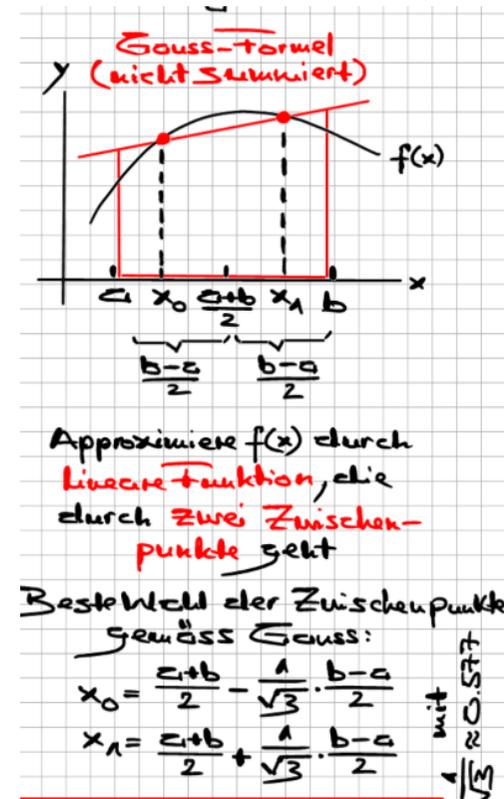


Abbildung 22: Gauss Formel

$$G = \frac{f\left(\frac{a+b}{2} - \frac{1}{\sqrt{3}} \cdot \frac{b-a}{2}\right) + f\left(\frac{a+b}{2} + \frac{1}{\sqrt{3}} \cdot \frac{b-a}{2}\right)}{2} \cdot (b-a)$$

```
# Implementation
def sum_gauss(f, a, b, h):
    n = int((b - a)/h)
    G = 0
    alpha = 1/3**0.5
    for i in range(0, n):
        xm = a + i*h + h/2 # mittlerer x-Wert des i-ten Teilintervalls
        G = G + f(xm - alpha*h/2) + f(xm + alpha*h/2)
    G = G/2*h
    return G
```

Abbildung 23: Summierte Gauss-Formel

```
# Fehler fuer h = 1, 0.1, 0.01,
kmax = 5
for k in range(0, kmax+1):
    h = 10.**(-k)
    G = sum_gauss(f, a, b, h)
    print(h, G, np.abs(G - I))
```

Abbildung 24: Fehler Gauss-Formel

9.2 Romberg-Extrapolation

Es seien
 $I = \int_a^b f(x) dx,$
 $T(h)$ die Approximation von I mit Summenformel Trapezregel.
 Dann heißt

$$E(h) = \frac{4 \cdot T(h/2) - T(h)}{3}$$
 einstufige Romberg-Extrapolation.

Abbildung 25: Romberg-Extrapolation

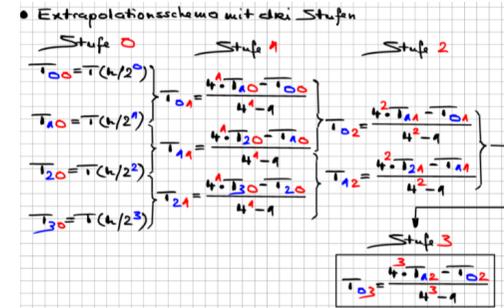


Abbildung 26: Mehrstufige Romberg-Extrapolation

10 Vorlesung 11

10.1 Definitionen DGL

• Eine gewöhnliche Differentialgleichung (DGL) ist eine Gleichung für eine gesuchte Funktion $y(x)$ mit $x \in [a, b]$ und $y \in \mathbb{R}$, wobei in der Gleichung $y(x)$, mindestens eine ihrer Ableitungen $y'(x), y''(x), y'''(x), \dots$ und allenfalls x auftreten:

$$F(x, y(x), y'(x), y''(x), y'''(x), \dots) = 0$$

• Eine DGL hat Ordnung n mit $n \in \mathbb{N}$, wenn die höchste auftretende Ableitung $y^{(n)}(x)$ ist:

$$F(x, y(x), y'(x), y''(x), \dots, y^{(n)}(x)) = 0$$

• Eine DGL n -ter Ordnung heißt explizit, wenn die Gleichung nach $y^{(n)}(x)$ aufgelöst ist:

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

 Eine nicht explizite DGL heißt implizit.

Abbildung 27: Definiton Begriffe DGL

10.1.1 AWP (Anfangswertproblem)

AWP = DGL n -ter Ordnung und Vorgabe Funktionswert von $y(x)$ sowie Werte der Ableitungen an Stelle x_0

Ein AWP
 $y^{(n)}(x) = f(x, y(x), y'(x), y''(x), \dots, y^{(n-1)}(x))$
 $y(x_0) = y_0, y'(x_0) = y_0', y''(x_0) = y_0'', \dots, y^{(n-1)}(x_0) = y_0^{(n-1)}$
 besitzt unter gewissen, in der Praxis meist erfüllten Bedingungen an die rechte Seite $f(\dots)$ der DGL genau eine Lösung
 $y(x)$ mit $x \in [a, b]$ und $y \in \mathbb{R}$,
 wobei x_0 in $[a, b]$ enthalten ist.

Abbildung 28: Existenz und Eindeutigkeit der Lösung eines AWP

Existenz u Eindeutigkeit

10.2 Anwendungen

$\ddot{s}(t) = -\frac{k}{m} * s(t)$ [mit k = Federkonstante, Ort $s(t)$]

10.3 Richtungsfeld von expliziter DGL 1. Ordnung

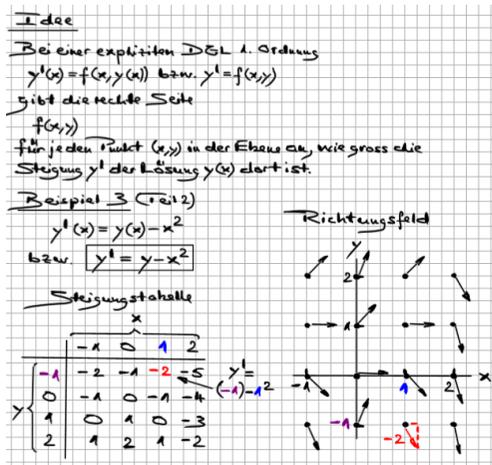


Abbildung 29: Richtungsfeld

10.4 Euler-Verfahren

Gegeben sei für $x \in [a, b]$ das AWP
 $y'(x) = f(x, y(x)), y(a) = y_0$.
 Dann liefert das Euler-Verfahren die numerische Näherungslösung
 (x_i, y_i) für $i = 0, \dots, n$
 mit
 $x_{i+1} = x_i + h,$
 $y_{i+1} = y_i + h \cdot f(x_i, y_i),$
 wobei
 $x_0 = a, h = \frac{b-a}{n}$ und $i = 0, 1, \dots, n-1$.

Abbildung 30: Euler-Verfahren

Gegeben sei für $x \in [a, b]$ das AWP
 $y'(x) = f(x, y(x)), y(a) = y_0$.
 Dann liefert das modifizierte Euler-Verfahren die Näherungslösung
 (x_i, y_i) für $i = 0, \dots, n$
 mit
 $x_{i+1} = x_i + h,$
 $k_1 = f(x_i, y_i)$
 $y_E = y_i + h \cdot k_1$
 $k_2 = f(x_{i+1}, y_E)$
 $k = \frac{k_1 + k_2}{2}$
 $y_{i+1} = y_i + h \cdot k$
 wobei
 $x_0 = a, h = \frac{b-a}{n}$ und $i = 0, 1, \dots, n-1$.

Abbildung 31: Modifiziertes Euler-Verfahren

11 Vorlesung 12

11.1 Mittelpunktverfahren

Gegeben sei für $x \in [a, b]$ das AWP
 $y'(x) = f(x, y(x)), y(a) = y_0$
 Dann liefert das Mittelpunkt-Verfahren die Näherungslösung
 (x_i, y_i) für $i = 0, \dots, n$
 durch
 $x_{i+1} = x_i + \frac{h}{2}$
 $k_1 = f(x_i, y_i)$
 $y_{i+1} = y_i + \frac{h}{2} \cdot k_1$
 $x_{i+1} = x_i + h$
 $k = f(x_{i+1}, y_{i+1})$
 $y_{i+1} = y_i + h \cdot k$
 wobei
 $x_0 = a, h = \frac{b-a}{n}$ und $i = 0, 1, \dots, n-1$.

Abbildung 32: Mittelpunkt Verfahren Algorithmus

11.2 Rundungsfehler vs Diskretisierungsfehler

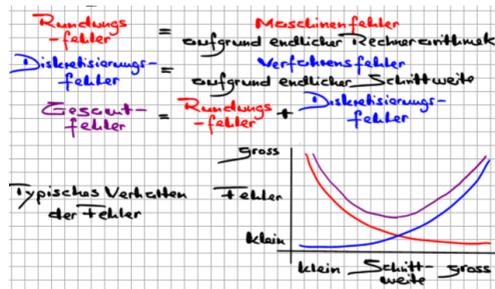


Abbildung 33: Rundungs vs Diskretisierungsfehler

11.3 Lokaler + Globaler Fehler, Konvergenzordnung

Verfahren zum numerischen Lösen einer gewöhnlichen DGL
 $\frac{\text{Lokaler Fehler}}{\text{Fehler}} = \frac{\text{Absoluter Fehler}}{\text{pro einzelner Iterationsschritt}}$
 $\frac{\text{Globaler Fehler}}{\text{Fehler}} = \frac{\text{Absoluter Fehler}}{\text{über alle Iterationsschritte}} = |y_n - y(x_n)|$
 Letzter Iterationswert \uparrow exakter Wert
 Konvergenzordnung: $|y_n - y(x_n)| \leq C \cdot h^p$
 $p \geq 1$ \uparrow Konstante \uparrow Schrittweite

Abbildung 34: Lokaler Fehler, Globaler Fehler, Konvergenzordnung

11.4 Runge-Kutta-Verfahren (Runge Kutta Verfahren)

11.4.1 Klassisches vierstufiges Runge-Kutta-Verfahren

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

Abbildung 35: Klassisches Vierstufiges Runge Kutta Verfahren

11.4.2 Allgemeines vierstufiges Runge-Kutta-Verfahren

Konstruktion von (x_{i+1}, y_{i+1}) aus (x_i, y_i)
 $k_1 = f(x_i + c_1 \cdot h, y_i)$
 $k_2 = f(x_i + c_2 \cdot h, y_i + h \cdot e_{21} \cdot k_1)$
 $k_3 = f(x_i + c_3 \cdot h, y_i + h \cdot (e_{31} \cdot k_1 + e_{32} \cdot k_2))$
 $k_4 = f(x_i + c_4 \cdot h, y_i + h \cdot (e_{41} \cdot k_1 + e_{42} \cdot k_2 + e_{43} \cdot k_3))$
 $k = b_1 \cdot k_1 + b_2 \cdot k_2 + b_3 \cdot k_3 + b_4 \cdot k_4$ mit $b_1 + b_2 + b_3 + b_4 = 1$
 $x_{i+1} = x_i + h$
 $y_{i+1} = y_i + h \cdot k$

Abbildung 36: Allgemeines Vierstufiges Runge-Kutta

11.4.3 Zweistufige u Dreistufige Runge-Kutta-Verfahren

$$\left. \begin{aligned} k_1 &= f(x_i + c_1 \cdot h, y_i) \\ k_2 &= f(x_i + c_2 \cdot h, y_i + h \cdot a_{21} \cdot k_1) \\ k &= b_1 \cdot k_1 + b_2 \cdot k_2 \text{ mit } b_1 + b_2 = 1 \\ x_{i+1} &= x_i + h \\ y_{i+1} &= y_i + h \cdot k \end{aligned} \right\} \begin{array}{c|c} c_1 & a_{21} \\ \hline c_2 & b_1 \quad b_2 \end{array}$$

Abbildung 37: Zweistufiges Runge-Kutta 2

$$\left. \begin{aligned} k_1 &= f(x_i + c_1 \cdot h, y_i) \\ k_2 &= f(x_i + c_2 \cdot h, y_i + h \cdot a_{21} \cdot k_1) \\ k_3 &= f(x_i + c_3 \cdot h, y_i + h \cdot (a_{31} \cdot k_1 + a_{32} \cdot k_2)) \\ k &= b_1 \cdot k_1 + b_2 \cdot k_2 + b_3 \cdot k_3 \text{ mit } b_1 + b_2 + b_3 = 1 \\ x_{i+1} &= x_i + h \\ y_{i+1} &= y_i + h \cdot k \end{aligned} \right\} \begin{array}{c|cc} c_1 & a_{21} & \\ \hline c_2 & a_{31} & a_{32} \\ \hline c_3 & b_1 & b_2 \quad b_3 \end{array}$$

Abbildung 38: Dreistufiges Runge-Kutta 3

- $S'(t) = -a * S(t) * I(t)$
- $I'(t) = a * S(t) * I(t) - b * I(t)$

12.2 Einzelne explizite Differentialgleichungen höherer Ordnung

Umwandlung DGL höherer Ordnung in System von mehreren DGL erster Ordnung

Eine einzelne explizite DGL m-ter Ordnung
 $y^{(m)}(x) = f(x, y(x), y'(x), y''(x), \dots, y^{(m-1)}(x))$
 kann durch Einführen der Funktionen
 $z_1(x) = y(x), z_2(x) = y'(x), z_3(x) = y''(x), \dots, z_m(x) = y^{(m-1)}(x)$
 in folgendes System von m explizite DGL 1. Ordnung
 umgeschrieben werden:

$$\begin{aligned} z_1'(x) &= z_2(x) \\ z_2'(x) &= z_3(x) \\ &\vdots \\ z_{m-1}'(x) &= z_m(x) \\ z_m'(x) &= f(x, z_1(x), z_2(x), z_3(x), \dots, z_m(x)) \end{aligned}$$

Abbildung 40: Explizite DGL m-ter Ordnung

12 Vorlesung 13

Numerisches Lösen von gewöhnlichen Differentialgleichungen

12.1 Systeme von expliziten Differentialgleichungen 1. Ordnung

Gesucht m Funktionen $y_1(x), y_2(x) \dots y_m(x)$ welche alle m DGL und alle m Anfangswerte erfüllen (bzw. numerische Approximation davon)

$$\vec{y}(x) = \begin{bmatrix} y_1(x) \\ \vdots \\ y_m(x) \end{bmatrix}, \vec{y}'(x) = \begin{bmatrix} y_1'(x) \\ \vdots \\ y_m'(x) \end{bmatrix}, f(x, \vec{y}(x)) = \begin{bmatrix} f_1(x, \vec{y}(x)) \\ \vdots \\ f_m(x, \vec{y}(x)) \end{bmatrix}, \vec{y}'(x) = \begin{bmatrix} y_1'(x) \\ \vdots \\ y_m'(x) \end{bmatrix}$$

$$\vec{y}'(x) = f(x, \vec{y}(x)), \vec{y}(x_0) = \vec{y}^{(0)}$$

Abbildung 39: Vektorisierung

Vorher genannte Verfahren können durch Vektorisierung angewendet werden.

12.1.1 Beispiel - Anwendung

- Bilanzgleichung: $S(t) + I(t) + R(t) = N$ [S = susceptible (noch nicht erkrankt), I = infected, R = removed (genesen/verstorben)]