Höhere Mathematik 2

Asha Schwegler

12. Juni 2022

Inhaltsverzeichnis

1 Funktionen mit mehreren Variablen **2**
- 1.1 Partielle Ableitungen . . . 2
- 1.2 Linearisierung von Funktionen . . . 2
- 1.3 Das Newton-Verfahren für Systeme . . . 2
  - 1.3.1 Vereinfachtes Newton-Verfahren . . . 3
  - 1.3.2 Gedämpftes Newton-Verfahren . . . 3

2 Ausgleichsrechnung **4**
- 2.1 Ausgleichsrechnung . . . 4
  - 2.1.1 Polynominterpolation . . . 4
  - 2.1.2 Spline-Interpolation . . . 5
  - 2.1.3 Lineare Ausgleichsprobleme . . . 7
  - 2.1.4 Nichtlineare Ausgleichprobleme . . . 8
  - 2.1.5 Das Gauss-Newton Verfahren . . . 8

3 Numerische Integration **10**
- 3.1 Rechteck- und Trapezregel . . . 10
  - 3.1.1 Summierte Rechkteckregle / summierte Trapezregel . . . 10
- 3.2 Simpsonregel . . . 10
  - 3.2.1 Summierte Simpsonregel . . . 10

4 Einführung in gewöhnliche Differentialgleichungen **11**

1 Funktionen mit mehreren Variablen

1.1 Partielle Ableitungen

$m = f'(x_0)$ im Punkt $(x_0, f(x_0)$

Tangentengleichung

Beispiel: P(1,3)

$$t_x = \underbrace{f(x_1, x_2)}_{\text{f(1,3)}} + \underbrace{\frac{\delta f}{\delta x_1}(x_1^{(0)}, x_2^{(0)})}_{\text{nach } x_1} * (x_1 - x_1^{(0)}) + \underbrace{\frac{\delta f}{\delta x_2}(x_1^{(0)}, x_2^{(0)})}_{\text{nach } x_2} * (x_2 - x_2^{(0)})$$

1.2 Linearisierung von Funktionen

Jacobi-Matrix Df(x)

Jacobi-Matrix enthält sämtliche partiellen Abl.1.Ord.von f:

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

Beispiel:

$$f(x) = \begin{bmatrix} f_1(x_1, x_2) \\ f_2(x_1, x_1) \end{bmatrix} = \begin{bmatrix} x_1^2 + x_2 - 11 \\ x_1 + x_2^2 - 7 \end{bmatrix}$$

$x^{(0)} = (1, 1)^T$

Partielle Ableitung:

$Df(x_1, x_2) = \begin{bmatrix} 2x_1 & 1 \\ 1 & 2x_2 \end{bmatrix}$

An der Stelle $x^{(0)}$:

$$Df(x_1^{(0)}, x_2^{(0)}) = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

Linearisierung:

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

$$g(x_1, x_2) = \begin{bmatrix} -9 \\ -5 \end{bmatrix} + \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} * \begin{bmatrix} x_1 - 1 \\ x_2 - 1 \end{bmatrix} = \begin{bmatrix} 2x_1 + x_2 - 12 \\ x_1 + 2x_2 - 8 \end{bmatrix}$$

Gleichung der Tangentialebene:

Alle angelegten Tangenten an die Bildfläche $y = f(x-1, x_2)$ im Flächenpunkt $P = f(x_1^{(0)}, x_2^{(0)})$

1.3 Das Newton-Verfahren für Systeme

- Konvergiert quadratisch wenn Df(x) regulär, und f 3-mal stetig differenzierbar ist.
- Vereinfachtes Newton Verfahren konvergiert linear.

Mögliche Abbruchkriterien: $\in > 0$

1. $n \geq n_{max}$ (bestimmte Anzahl Iterationen)
2. $\|x^{n+1} - x^n\| \leq \|x^{n+1}\| * \in$ (relativer Fehler)

3. $\|x^{n+1} - x^n\| \leq \in$ (absoluter Fehler)
4. $\|f(x^{n+1})\| \leq \in$ (max residual)

Algorithmus:

1. $Df(x^n)\delta^n = -f(x^{(n)})$
2. nach δ^n auflösen
3. $x^{n+1} = x^n + \delta^n$

1.3.1 Vereinfachtes Newton-Verfahren

Konvergiert nur noch linear!!
Natürlich deutlich langsamer!

Immer wieder $Df(x^0) verwenden$

Algorithmus:

1. $Df(x^0)\delta^n = -f(x^{(n)})$
2. nach δ^n auflösen
3. $x^{n+1} = x^n + \delta^n$

1.3.2 Gedämpftes Newton-Verfahren

Nach dem n-ten Schritt wenn $Df(x^n)$ schlecht konditioniert ist (nicht oder fast nicht invertierbar), dann $x^n + \delta^n$ verwerfen!!
Funktioniert auch mit vereinfachtes Newton Verfahren.

1 ***Probieren:***

$x^n + \dfrac{\delta^n}{2}$

Mit der Bedingung:
$\|f(x^n) + \dfrac{\delta^n}{2})\|_2 < \|f(x^n\|_2$

Weil wir Iteration $\|f(x^n)\|_2$ *gegen 0 erreichen wollen*

Algorithmus:

1. $Df(x^n)\delta^n = -f(x^{(n)})$
2. nach δ^n auflösen
3. Finde minimale aus $k \in \|f(x^n) + \frac{\delta^n}{2})\|_2 < \|f(x^n\|_2$,$k_{max} = 4$
   sofern nichts Anderes als sinnvoll angegeben

```
        while k <= k_max:
                new_residual = np.linalg.norm
                (f_lambda(x_n + (delta.reshape(x0.shape[0], ) /
                (2 ** k))), 2)
#    f (x(n) +      (n) / 2^k)
                if new_residual < last_residual:
                                delta = delta / (2**k)
                                x_next = x_n +
                                delta.reshape(x0.shape[0], )
# x(n+1) = x(n) +       (n) / 2^k
                                k_actual = k
                        break
# Minimales k,  f  r  welches das Residuum besser ist wurde gefunden
-> abbrechen
                    else:
                                k=0
         k += 1
```

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

2 Ausgleichsrechnung

2.1 Ausgleichsrechnung

- Datenpunkte mit gewissen Streuung durch einfache Funktion annähern
- Mehr Gleichungen als unbekannte (Mehr Datenpunkte als Parameter)

2.1.1 Polynominterpolation

Gesucht: $P_n(x)$ welche n+1 Stützpunkte interpoliert Jeder Stützpunkt gibt lin.Gleichung für die Bestimmung der Koeffizienten.

Grad n so wählen , dass lin.Gleichungssystem gleich viele Gleichungen wie unbekannte Koeffizienten hat.

$P_n(x_0) = a_0 + a_1x_0 + a_2x_0^2 + ... + a_nx_0^n = y_0$

$P_n(x_1) = a_0 + a_1x_1 + a_2x_1^2 + ... + a_nx_1^n = y_1$

.
.
.

$P_n(x_n) = a_0 + a_1x_n + a_2x_n^2 + ... + a_nx_n^n = y_n$

$$= \begin{bmatrix} 1 & x_0 & ... & x_0^n \\ . & . & & \\ . & . & & \\ . & . & & \\ 1 & x_n & ... & n_n^n \end{bmatrix} * \begin{bmatrix} a_0 \\ . \\ . \\ . \\ a_n \end{bmatrix} = \begin{bmatrix} y_0 \\ . \\ . \\ . \\ y_n \end{bmatrix} \Bigg\} Vandermonde - Matrix = Schlechtkonditioniert.Abn \geq 20 numerischinstabil$$

Lagrange Interpolationsformel:
Lagrangeform von $P_n(x)$:

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

LagrangePolynome $= l_i(x)$:

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

Stückweise Interpolation:
Interpolationspolynom erster Ordnung: n = 1
Stützpunkte so wählen: $x_{j-1} und x_{j+1}$,somit 2 Werte: n = 1

Grösse des Fehlers an Stelle x wenn:
y_i Funktionswerte (genügend oft stetig differenzierbare Funktion)
$|f(x) - P_n(x)| \leq \frac{|(x - x_0)(x - x_1)...(x - x_n)|}{(n + 1!)}$
Max der (n+1)-ten Ableitung der f(x) Intervall $[x_0, x_n]$ kennen, Fehlerabschätzung nur dann möglich.

2.1.2 Spline-Interpolation

Bedingungen für S_i:

1. $S_i(x_i) = y_i, S_{i+1}(x+1) = y_{i+1}, ... Interpolation$
2. $S_i(x_{i+1}) = S_{i+1}(x_{i+1}), S_{i+1}(x_{i+2}) = S_{i+2}(x_{i+2}), ...$Stetiger Übergang
3. $S_i'(x_{i+1}) = S_{i+1}'(x_{i+1}), S_{i+1}'(x_{i+2}) = S_{i+2}', ... KeineKnicke$
4. $S_i''(x_{i+1}) = S_{i+1}''(x_{i+1}), S_{i+1}''(x_{i+2}) = S_{i+2}'', ...$ Gleiche Krümmung
5. Mindestens den Grad 3 ... Kubische Splines

3 Intervalle, 4 Stützpunkte:
$[x_0, x_1], [x_1, x_2], [x_2, x_3]$
Ansatz:
$S_0 = a_0 + b_0(x - x_0) + c_0(x - x_0)^2 + d_0(x - x_0)^3, x \in [x_0, x_1]$
$S_1 = a_1 + b_1(x - x_1) + c_1(x - x_1)^2 + d_1(x - x_1)^3, x \in [x_1, x_2]$
$S_2 = a_2 + b_2(x - x_2) + c_2(x - x_2)^2 + d_2(x - x_2)^3, x \in [x_2, x_3]$

$3 * 4 = 12$ Koeffizienten $\Longrightarrow$ 12 Bedingungen
1.Interpolation der Stützpunkte:
1.$S_0(x_0) = y_0$
2.$S_1(x_1) = y_1$
3.$S_2(x_2) = y_2$
4.$S_3(x_3) = y_3$

2. Stetiger Übergang an Stellen x_1 und x_2:
5.$S_0(x_1) = S_1(x_1)$
6.$S_1(x_2) = S_2(x_2)$

3. Erste Ableitung an Übergangstellen übereinstimmen:
7.$S_0'(x_1) = S_1'(x_1)$
8.$S_1'(x_2) = S_2'(x_2)$

4. Zweite Ableitung an Übergangstellen übereinstimmen:
9.$S_0''(x_1) = S_1''(x_1)$
10.$S_1''(x_2) = S_2''(x_2)$

$= 10 Bedingungen$

Die weiteren 2 Bedingungen können frei gewählt"werden.
Diese beziehen sich häufig auf Randstellen x_0 und x_3.
Beispiele:
Natürliche kubische Splinefunktion:
Mit moeglichen Wendepunkt im Anfangs und Endpunkt.

$$\left.\begin{array}{l} S_0'''(x_0) = 0 \\ S_2'''(x_3) = 0 \end{array}\right\}$$

Periodische kubische Splinefunktion:
Wenn man Periode $p = x_3 - x - 0$ hat und damit y_0 bzw. $S_0(x_0) = S_2(x_3)$ gilt

$$\left.\begin{array}{l} S_0'(x_0) = S_2'(x_3) \\ S_0''(x_0) = S_2''(x_3) \end{array}\right\}$$

mit not-a-knot Bedingung kubische Splinefunktion:
s.d Auch dritte Ableitung in $x - 1, x - 2$ noch stetig ist. (x_1, x_2 keine echten Knoten)

$$\left.\begin{array}{l} S_0'''(x_1) = S'''x_1(x_1) \\ S_1'''(x_2) = S_2'''(x_2) \end{array}\right\}$$

Algorithmus: natürliche kubische Splinefunktion
$S_i = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$
Koeffizienten a_i, b_i, c_i, d_i berechnen:

1. $a_i = y_i$
2. $h_i = x_{i+1} = -x_i$
3. $c_0 = 0, c_n = 0$!!! (für periodisch($s_1 = s_0$),für not a knot($d_0 = d_1 d_{n-2} = d_{n-1}$)
4. Berechnung der Koeffizienten $c_1, c_2, c_3, ..., c_{n-1}$
   - i = 1:
     - $2(h_0 + h_1)\; c_1 + h_1\; c_2 = 3\dfrac{y_2 - y_1}{h_1} - 3\dfrac{y_1 - y_0}{h_0}$
   - falls $n \geq 4$ dann für i = 2,...,n-2
     - $h_{i-1}\; c_{i-1} + 2(h_{i-1} + h_i)\; c_i + h_i\; c_{i+1} = 3\dfrac{y_{i+1} - y_i}{h_i} - 3\dfrac{y_i - y_{i-1}}{h_{i-1}}$
   - i = n - 1:

$$-\ 2(h_{n-2}+h_i)\ c_{n-2}\ +\ 2(h_{n-2}+h_{n-1})\ c_{n-1} = 3\frac{y_n - y_{n-1}}{h_{n-1}} - 3\frac{y_{n-1}-y_{n-2}}{h_{n-2}}$$

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

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

Ergibt das Gleichungssystem A c = z

- A ist immer invertierbar
- Numerische Lsg.durch Gauss-Algo
- System ist gut konditioniert
- Pivotsuche nicht erforderlich

i = 0,...,n-1

$A = \begin{bmatrix} \\ \\ \\ \end{bmatrix}\ c = \begin{bmatrix} \\ \\ \end{bmatrix} = z = \begin{bmatrix} \\ \\ \end{bmatrix}$

2.1.3 Lineare Ausgleichsprobleme

Ausgleichsgerade: $f(x) = ax + b$: gesucht: $F = af_1 + bf_2 \mid a, b \in \mathbb{R}$ **Basisfunktion:** $f_1(x) = x$ und $f_2(x) = 1$

Form Fehlerfunktional:

$E(f)(a,b) := \sum_{i=1}^{n}(y_i - f(x_i))^2 = \sum_{i=1}^{n}(y_i - (ax_i + b))^2)$

minimal = patiellen Ableitung nach a,b müssen verschwinden

$$\begin{bmatrix} \sum_{i=1}^{n}(x_i)^2 & \sum_{i=1}^{n}(x_i) \\ \\ \sum_{i=1}^{n}(x_i) & n \end{bmatrix} * \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n}(x_i y_i) \\ \\ \sum_{i=1}^{n}(y_i) \end{bmatrix}$$

Nach a,b auflösen

Allgemeine Definition:

Gegeben:

n-Wertepaare: $(x_i, y_i), i = 1, ..., n$

Basisfunktionen: $(f_1, ..., f_m)[a, b]$

Ansatzfunktionen: $f := \lambda_1 f_1 + ... + \lambda_m f_m$

Lineares Ausgleichproblem mit Fehlerfunktional:

$E(f) = \|y - f(x)\|_2^2 = \sum_{i=1}^{n}(y_i - \sum_{i=1}^{m}\lambda_j * f_j * (x_i))^2 = \|y - A\lambda\|_2^2$

Fehlergleichungssystem: $A\lambda = y$

$$\text{A}=\begin{bmatrix} f_1(x_1) & f_2(x)(x_1) & ... & f_m(x_1) \\ f_1(x_2) & f_2(x_2)(x_1) & ... & f_m(x_2) \\ . & & & \\ . & & & \\ . & & & \\ f_1(x_n) & f_2(x_n)(x_n) & ... & f_m(x_n) \\ & & & \end{bmatrix} * y = \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ . \\ y_n \end{bmatrix} = \lambda \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ . \\ . \\ . \\ \lambda_n \end{bmatrix}$$

Normalgleichungen: $\frac{\delta E(f)(\lambda_1, ..., \lambda_m)}{\delta \lambda_j}$

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

Lösung:λ beinhaltet gesuchte Prameter des linearen Ausgleichproblems
Verfahren am Besten mit QR-Zerlegung

2.1.4 Nichtlineare Ausgleichprobleme

Allgemeines Ausgleichproblem:

Fehlerfunktional:
f_p = Ansatzfunktion

$$E(f)(a,b) := \sum_{i=1}^{n}(y_i - f_p(\lambda_1,\lambda_2,...,\lambda_m,x_i))^2 = \left\| \begin{bmatrix} y_1 - f_p(\lambda_1,\lambda_2,...,\lambda_m,x_1) \\ y_2 - f_p(\lambda_1,\lambda_2,...,\lambda_m,x_2) \\ . \\ . \\ . \\ y_n - f_p(\lambda_1,\lambda_2,...,\lambda_m,x_n) \end{bmatrix} \right\|_2^2 =$$

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

$$f(\lambda) := f(\lambda_1,\lambda_2,...,\lambda_m) := \begin{bmatrix} f_p(\lambda_1,\lambda_2,...,\lambda_m) \\ f_p(\lambda_1,\lambda_2,...,\lambda_m) \\ . \\ . \\ . \\ f_p(\lambda_1,\lambda_2,...,\lambda_m) \end{bmatrix} := \begin{bmatrix} f_p(\lambda_1,\lambda_2,...,\lambda_m,x_1) \\ f_p(\lambda_1,\lambda_2,...,\lambda_m,x_2) \\ . \\ . \\ . \\ f_p(\lambda_1,\lambda_2,...,\lambda_m,x_n) \end{bmatrix},$$

$$y = \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ . \\ y_n \end{bmatrix}, \lambda = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ . \\ . \\ . \\ \lambda_n \end{bmatrix}$$

Falls Ansatzfunktionen linear in den Parametern $\Longrightarrow$
Spezialfall des Ausgleichsproblems: $f(\lambda) = A\lambda \Longrightarrow$ Allg.Ausgleichsproblem = Minimums einer Funktion

Schritte:

1. Normalgleichungen aufstellen (part.Abl.von f)
2. $\lambda_i = 0$
3. Nichtlineare Gleichungssystem lösen

2.1.5 Das Gauss-Newton Verfahren

Quadratmittelproblem: Problem einen Vektor x zu finden für den E(x) minimal wird.
Nichtlineare Ausgleichsprobleme: $g(\lambda) := y - f(\lambda)$
Fehlerfunktional: $E(\lambda) := \|g(\lambda)\|_2^2 = \|y - f(\lambda)\|_2^2$
Gauss-Newton-Verfahren: Lin.Ausgl.RG + Newton-Verfahren = $g(\lambda) = g(\lambda^{(0)}) + Dg(\lambda^0) * (\lambda - \lambda^0)$
= Verallg.Tangentgleichung!!

$$
Dg(\lambda^0) = \begin{bmatrix}
\frac{\delta g_1}{\lambda_1}(\delta\lambda^0) & \frac{\delta g_1}{\lambda_2}(\delta\lambda^0) & \dots & \frac{\delta g_1}{\lambda_m}(\delta\lambda^{(0)}) \\
\frac{\delta g_2}{\lambda_1}(\delta\lambda^0) & \frac{\delta g_2}{\lambda_2}(\delta\lambda^0) & \dots & \frac{\delta g_2}{\lambda_m}(\lambda^{(0)}) \\
. & & & \\
. & & & \\
. & & & \\
\frac{\delta g_n}{\lambda_1}(\delta\lambda^0) & \frac{\delta g_n}{\lambda_2}(\delta\lambda^0) & \dots & \frac{\delta g_n}{\lambda_m}(\delta\lambda^{(0)})
\end{bmatrix}
$$

Minimum Fehlerfunktionals: $E(\lambda) = \underbrace{\|g(\lambda^{(0)})\|}_{\text{y}} + \underbrace{Dg(\lambda^0)}_{\text{-A}} * \underbrace{\lambda - \lambda^0)}_{\delta}$

Normagleichungssystem: $A^T A\delta = A^T y = Dg(\lambda^0)^T Dg(\lambda^0)\delta = Dg(\lambda^0)^T * g(\lambda^0) \implies$
QR-Zerlegung: $Dg(\lambda^0)$=QR$\geq R\delta = -Q^T g(\lambda^0)$
$\implies$ Lösung: $\Lambda = (\lambda^0) + \delta$

Schritte:

1. Berechne Funktion
   - $g(\lambda) := y - f(\lambda)$
   - $Dg(\lambda)$
2. k = 0,1...:
   δ^k als Lösung: $min\|g(\lambda^k) + Dg(\lambda^k) * \delta^k\|_2^2$
   $\implies Dg(\lambda^k)^T * Dg(\lambda^k)\delta^k = Dg(\lambda^k)^T * g(\lambda^k)$
   nach nach δ^k auflösen durch QR-Zerlegung.
   - $Dg(\lambda^k) = Q^k R^k$
   - $R^k \delta^k = -(Q^k)^T g(\lambda^k)$
3. Setze: $\lambda^{k+1} = \lambda^k + \delta^k$

Korrektur δ^k nur akzeptiert wenn Fehlerfunktional zur Abnahme führt: **Fehlerfunktional** $E(\lambda^{k+1}) = \|g(\lambda^{k+1})\|_2^2 < \|g(\lambda^k)\|_2^2 = E(\lambda^k)$

Gedämpftes Gauss-Newton-Verfahren

1. Berechne Funktion:
   - $g(\lambda) := y - f(\lambda)$
   - $Dg(\lambda)$
2. k = 0,1...:
   δ^k als Lösung: $min\|g(\lambda^k) + Dg(\lambda^k) * \delta^k\|_2^2$
   $\implies Dg(\lambda^k)^T * Dg(\lambda^k)\delta^k = Dg(\lambda^k)^T * g(\lambda^k)$
   nach nach δ^k auflösen durch QR-Zerlegung.
   - $Dg(\lambda^k) = Q^k R^k$
   - $R^k \delta^k = -(Q^k)^T g(\lambda^k)$

3. Finde das minimale p $\in$ 0,1,...,p_{max}
$\underbrace{\|\lambda^{(k)} + \frac{\delta^k}{2^p}\|_2^2}_{\lambda^{(k+1)}} < \|g(\lambda^{(k)})\|_2^2$

4. Falls kein min. p gefunden:
p=0 und weiterfahren

5. Setze: $\lambda^{(k)} + \frac{\delta^{(k)}}{2^p}$

Abbruchkriterium: $\|\frac{\delta^{(k)}}{2^p}\| < TOL$
Keine Grantie, dass die Näherung max. Abstand von TOL zum ges. Min. ist.

3 Numerische Integration

- Summierte Rechteckregel ist genauer als die summierte Trapezregel
- Summierte Simpsonregel ist am genaustren(verglichen mit sum.Recht.und Trap.)
- Faktor Fehlerabschätzung summ.Rechteckregel < summ.Trap.Regel = 2

3.1 Rechteck- und Trapezregel

Annäherung bestimmtes Integral.
Rechtecksregel / Mittelpunktsregel:
$Rf = f(\frac{a+b}{2}) * (b-a)$
$Tf = \frac{f(a)+f(b)}{2} * (b-a)$

3.1.1 Summierte Rechkteckregle / summierte Trapezregel

h=$\frac{b-a}{n}$; $x_i = a + i*$ h ; $x_n = b$; (i=0,...,n-1)
$Rf(h) = h * \sum_{i=0}^{n-1} f(x_i + \frac{h}{2})$
$Tf(h) = h * (\frac{f(a)+f(b)}{2} + \sum_{i=1}^{n-1} f(x_i))$
n = Anzahl Subintervalle $[x_i, x_{i+1}]$

3.2 Simpsonregel

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

3.2.1 Summierte Simpsonregel

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

4 Einführung in gewöhnliche Differentialgleichungen