

0.1 Partielle Ableitungen

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

Tangentengleichung

Beispiel: $P(1,3)$

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

0.2 Linearisierung von Funktionen

Jacobi-Matrix $Df(x)$

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

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

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 $P = f(x_1^{(0)}, x_2^{(0)})$

0.3 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: $\epsilon > 0$

1. $n \geq n_{max}$ (bestimmte Anzahl Iterationen)
2. $\|x^{n+1} - x^n\| \leq \|x^{n+1}\| * \epsilon$ (relativer Fehler)
3. $\|x^{n+1} - x^n\| \leq \epsilon$ (absoluter Fehler)
4. $\|f(x^{n+1})\| \leq \epsilon$ (max residual)

Algorithmus:

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

0.3.1 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. $x^{n+1} = x^n + \delta^n$

0.3.2 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:

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

Mit der Bedingung:

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

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