Numerik – Zusammenfassung

Äquivalente Umformungen: Getreu dem Motto, Kleinvieh macht auch Mist!

$\sqrt{x} = x^{\frac{1}{2}}$
$x^{-2} = \frac{1}{x^2}$
$x^{-\frac{1}{2}} = \frac{1}{\sqrt{x}}$
$x^{-1} = \frac{1}{x}$

$\frac{2}{3\sqrt[4]{x^5}} = \frac{2}{3x^{\frac{5}{4}}} = \frac{2}{3}x^{-\frac{5}{4}}$
$10^{n+1} = 10^n + 10^1$
$-2^1 = -2,\ -2^2 = -4$
$(-2)^1 = -2, (-2)^2 = +4$
$\frac{x^{6n+2} * x^{3-n}}{(x^2)^n * (x^{n+3})^2} = \frac{x^{6n+2+3-2}}{x^{2n} * x^{2n+6}} =$

$\frac{x^{5n+5}}{x^{4n+6}} = x^{5n+5-4n-6} = x^{n-1}$
$3^{n+3} = 3^n * 3^3$
$4 = \frac{4}{1} = \frac{1}{1} * \frac{4}{1} = \frac{1}{1} : \frac{1}{4} = \frac{1}{\frac{1}{4}}$

Ableitung – Liste elementarer Regeln $f(x) = x^n \rightarrow f'(x) = n \cdot x^{n-1}$

Funktion		**Ableitung**	
$f\ (x)\ = u(x) * v(x)$		$f\ `(x)\ = u`(x) * v(x) + u(x)\ *\ v`(x)$	
$f(x) = \frac{u\ (x)}{v(x)}$		$f\ `(x) = \frac{u`\ (x) * v(x) - u\ (x) * v`\ (x)}{(v(x))^2}$	
$f(x) = \sin(x),$	$f(x) = \sinh(x)$	$f\ `(x) = \cos(x),$	$f'(x) = \cosh(x)$
$f(x) = \cos(x),$	$f(x) = \cosh(x)$	$f\ `(x) = -\sin(x),$	$f'(x) = \sinh(x)$
$f(x) = \tan(x)$		$f\ `(x) = 1 + \tan^2(x)$	
$f(x) = e^x$		$f\ `(x) = e^x$	
$f(x) = \ln(x)$		$f\ `(x) = \frac{1}{x}$	
$f(x) = a^x$		$f\ `(x) = \ln(a)\ *\ a^x$	
$f(x) = \arcsin(x)$		$f\ `(x) = \pm\frac{1}{\sqrt{1-x^2}}$	
$f(x) = \arccos(x)$		$f\ `(x) = \mp\frac{1}{\sqrt{1-x^2}}$	
$f(x) = arctan(x)$		$f\ `(x) = \frac{1}{1+x^2}$	
$\sin(\alpha+\beta)\ =\ \sin(\alpha)\ *\ \cos(\beta) +\ \cos(\alpha)\ *\ sin(\beta)$			

Ableitungsregeln

Faktorregel	$(\lambda\, f(x))'\ =\ \lambda\, f'(x), \lambda\ \in\ \mathbb{R}$
Summenregel	$(f(x) + g(x))'\ =\ f'(x) + g'(x)$
Produkteregel	$(f(x)\ *\ g(x))'\ =\ f'(x)\ *\ g(x)\ +\ f(x)\ *\ g'(x)$
Quotientenregel	$(\frac{f(x)}{g(x)})' = \frac{f'(x) * g(x) - f(x) * g'(x)}{g^2(x)}$
Inversenregel	$\frac{d}{dx}\ g\ *\ f(x)\ =\ \frac{dg}{dy}\ (f(x))\ *\ \frac{df}{dx}\ (x)$
Kettenregel	$f\ (x)\ =\ u(v(x))\ = f'\ (x) = u'\ (v(x))\ *\ v'\ (x)$

Beispiele für Ableitungen – Struktur erkannt, Gefahr gebannt!

$f(x)\ =\ 3(sin(x) + x)^7$	$f'(x)\ = 21(sin(x) + x)^6\ *\ (cos(x) + 1)$
$f(x)\ =\ (sin(2x))^3$	$f'(x)\ = 3(sin(2x))^2 * [sin(2x)]'$ $f'(x)\ = 3(sin(2x))^2\ *\ cos(2x)\ *\ 2$
$f(x)\ =\ cos(x^2 - 4x)$	$f'(x)\ = -sin(x^2\ -\ 4x)\ *\ (2x\ -\ 4)$
$f(x) = 20\ *\ \ln(3\ -\ x^4) + 304$	$f'(x) = 20\ *\ \frac{1}{3-x^4}\ *\ (-4x^3)\ =\ \frac{-80x^3}{3-x^4}$
$f(x) = \frac{\sqrt{3}x-\sqrt{2}}{3x^2-2} = \frac{\sqrt{3}x-\sqrt{2}}{(\sqrt{3}x-\sqrt{2})(\sqrt{3}x+\sqrt{2})}$	$f(x) = \frac{1}{\sqrt{3}x+\sqrt{2}} = -\frac{\sqrt{3}}{(\sqrt{3}x+\sqrt{2})^2}$ Quotientenregel
$f(x) = e^{2x}$ $\quad$ $f(x) = e^{x^2}$	$f(x) = 2 * e^{2x}$ $\quad$ $f(x) = 2x * e^{x^2}$
$f(x) = 2x + e^{-4x^3}$	$f(x) = 2 - 12x^2e^{-4x^3}$
$f(x) = (x^2 - x)e^{1-x^2}$	$f(x) = (2x-1) * e^{1-x^2} + (x^2-x) * -2xe^{1-x^2}$ $= e^{1-x^2}((2x-1) + (x^2-x) * (-2x))$ $= e^{1-x^2}(-2x^3 + 2x^2 + 2x - 1)$
$f(x) = 20 * sin(x^2 - 1)$	$f'(x) = 20 * cos(x^2 - 1) * 2x$

Verwendete Packages aus Python (import / import … as …. / from … import … as …):

- import math
- import numpy as np
- from matplotlib import pyplot as plt

SW1 – Python-Einführung / Grundlagen

- z = np.array([1], **dtype = np.float64**) #array immer mit np.float64 als Datentyp abspeichern!
- np.linspace(start, stop, num) #macht Vektor **inkl**. Start- und Stoppwerte, num steht für die Anzahl Elemente im Vektor, die Schrittweite wird somit automatisch angepasst. np.linspace(1, 4, 4) #[1., 2., 3., 4.]
- np.zeros((2, 3)) #ergibt hier eine 2x3 Matrix mit Nullen, wichtig ist die Übergabe als Tuple!
- np.eye(3) #ergibt hier eine 3x3 Einheitsmatrix = quadratisch mit Diagonalelemente = 1, Rest = 0
- np.ones((3, 3)) #ergibt hier eine 3x3 Einsermatrix (alle Elemente/Einträge der Matrix sind Einsen)
- np.diag([1, 2, 3]) #ergibt 3x3 Matrix, mit den Diagonalelementen [0,0] = 1, [1,1] = 2, [2,2] = 3, Rest = 0
- np.random.random((2, 3)) #ergibt hier eine 2x3 Zufallsmatrix

SW2 – Rechnerarithmetik

Eine Gleitkommazahl ist eine rationale Zahl mit weiteren einschränkenden Eigenschaften. Derartige Zahlen werden insbesondere von Computern für Berechnungen verwendet. Ob eine gegebene rationale Zahl eine Gleitkommazahl ist, hängt nicht von der Zahl selbst ab, sondern vom gewählten Darstellungsformat.

Problematik: Problem hängt mit Rundung der Rechenschritte zusammen → Problem ist uns aus PROG2 bereits bekannt. Damals mit **isclose()**

```
print(a := (1/3)**5)    # 0.004115226337448558
print(b := 1/243)       # 0.00411522633744856
print(a == b) # False
from math import isclose    \n    print(isclose(a, b, abs_tol = 0.001)) #True   # abs_tol = Genauigkeit
```

Eine Rechenmaschine führt eine Rechenoperation mit mehr stellen aus und rundet danach das Resultat auf die nächste Maschinenzahl. Nach jedem Rechenschritt wird gerundet. So entsteht die Diskrepanz!

Gleitkommazahl: $x = -30.0625$, diese Zahl kann als **Summe** von **10er Potenzen** geschrieben werden

$$x = -30.0625 = -(3 \cdot 10^1 + 0 \cdot 10^0 + 6 \cdot 10^{-2} + 2 \cdot 10^{-3} + 5 \cdot 10^{-4}) = -3006.25 \cdot 10^{-2}$$

Man sieht damit, dass die Darstellung der Gleitkommazahl **nicht eindeutig** ist!

Normierte Gleitpunktzahl: nur eine Ziffer vor dem Punkt, diese muss > 0 sein. → wiss. Schreibweise

Wert w einer Zahl ergibt sich nach der Formel $w := v \cdot m \cdot b^e = -3.00625 \cdot 10^1$

$v =$ Vorzeichen der Zahl, $m =$ die Mantisse der Zahl, $b =$ Basis des Darstellungsformats, $e =$ Exponent

Genauigkeit von numpy: kann eingestellt werden (aber nur für numpy-Befehle!) mit dem Befehl

```
np.set_printoptions(precision = 4)    \n    print(np.array([1.2345667e-5]))  #array([1.2346e-05])
```

Allgemeine Formatierung: print(np.format_float_scientific(1372.4828e-4, precision = 4)) #'1.3725e-01'

Die Rundung in der Darstellung der Mantisse erfolgt unabhängig vom Exponenten. Daraus folgt, dass der Abstand von Fliesskommazahlen mit dem Exponenten zunimmt. Umso näher beisammen zwei Zahlen sind, umso grösser wird der Rundungsfehler!

Sei x eine exakte Grösse, so ist $\hat{x}$ die Näherung von x. Daraus ergibt sich folgendes:

- **absoluter Fehler:** $e_{abs} = |\hat{x} - x|$
- **relativer Fehler:** $e_{rel} = \frac{|\hat{x}-x|}{|x|}$

$e_{rel} \cdot 100 \triangleq$ ***prozentuale Abweichung***, welche die Grössenordnung von x berücksichtigt.

Beispiel 3-stellige Gleitpunktarithmetik: gesucht ist $x = x_1 - x_2$, gegeben ist $x_1 = 1.7851$ und $x_2 = 1.7849$

Rundung durchführen: $x_1 = 1.7851 \rightarrow \hat{x}_1 = 1.79$, $x_1 = 1.7849 \rightarrow \hat{x}_2 = 1.78$ $\quad \hat{x} = 1.79 - 1.78 = 0.01$

Exakt: $x = 1.7851 - 1.7849 = 0.0002$ $\quad e_{rel} = \frac{|\hat{x}-x|}{|x|} = \frac{|0.01-0.0002|}{|0.0002|} = 49$ $\quad \rightarrow e_{rel} \cdot 100 \triangleq 4900\%$**!!!**

Fazit: Viel schlimm! Der relativer Fehler wird sehr gross. Dieses Phänomen nennt man **Auslöschung,** welche auftritt, alsbald beinahe gleichgrosse Zahlen addiert / subtrahiert werden. Diese Fehler entstehen durch **Rundungen** oder aber **Verfahrensfehler/Abbruchfehler** (bspw. bei Approximationen vom Sinus)

Wann verschwindet der Fehler? Beispiel bei Taylor Approximationen die Stelle $x_0 =$ Entwicklungsstelle stimmt immer exakt überein!

Binärsystem: $x = -(11{,}11100001)_2 \cdot 2^{(100)_2} = v \cdot m \cdot b^e =$ Die indexierte 2 steht für die explizite Basis

$$x = -(1 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3} + 1 \cdot 2^{-8}) \cdot 2^{2^2} = -(30.0625)$$

Dezimalzahl $x = 2.0$ in Binär: $x = (10.0)_2 = (1.00)_2 \cdot 2^{(1)_2}$ → letzter Teil ist die normierte Zahl, $(1) =$ steht für die Verschiebung des Kommas um eine Stelle!

SW3 – Taylor-Approximation → Interaktive Online-Demo unter: https://www.geogebra.org/m/s9SkCsvC

Idee: Komplizierte Funktion durch eine einfachere Funktion möglichst gut zu approximieren.
Schrittweises Verbessern der lokalen Näherung durch Hinzunahme komplexerer Eigenschaften:

1. t_0 stimmt mit f am **Punkt** x_0 überein,
2. t_1 stimmt zusätzlich mit f am Punkt x_0 auch in der **Steigung** überein,
3. t_2 stimmt zusätzlich mit f am Punkt x_0 auch in der **Krümmung** überein, usw.

Diese Bedingungen lassen sich durch **Polynome immer höheren Grades** erfüllen:
1. t_0 ist eine konstante Funktion, 2. t_1 ist eine Gerade, 3. t_2 ist eine Parabel, usw.

Taylor-Reihe $t(x)$ einer oft genug differenzierbaren Funktion f am **Entwicklungspunkt** x_0:

$$t(x) = f(x_0) + \frac{f'(x_0)}{1!}(x-x_0) + \frac{f''(x_0)}{2!}(x-x_0)^2 + \frac{f'''(x_0)}{3!}(x-x_0)^3 + \cdots$$
$$= a_0 + a_1(x-x_0) + a_2(x-x_0)^2 + a_3(x-x_0)^3 + \cdots = \sum_{k=0}^{\infty} \frac{1}{k!} f^{(k)}(x_0)(x-x_0)^k = \sum_{k=0}^{\infty} a_k (x-x_0)^k$$

Beispiel: $f(x) = \frac{1}{1+x^2} = (1+x^2)^{-1}$, $x_0 = 0$ $\quad f'(x) = -1 \cdot (1+x^2)^{-2} \cdot 2x = -2x \cdot (1+x^2)^{-2}$

$t_0(x) = f(x_0) = f(0) = \frac{1}{1+0^2} = 1$ $\quad a_0 = f(x_0) = 1$

$t_1(x) = f(x_0) + \frac{1}{1!} \cdot f'(x_0)(x-x_0) = f(x_0) + \frac{1}{1!} \cdot 0 \cdot (x-x_0) = f(x_0) = 1$ $\quad a_1 = \frac{f'(x_0)}{1!} = \frac{0}{1!} = 0$

$f''(x) = -2 \cdot (1+x^2)^{-2} + -2x \cdot -2(1+x^2)^{-3} \cdot 2x = -2 \cdot (1+x^2)^{-2} + 8x^2 \cdot (1+x^2)^{-3}$

$f''(0) = -2 \cdot (1+0^2)^{-2} = -2 \cdot (1)^{-2} = -\frac{2}{1} = -2$ $\quad a_2 = \frac{f''(x_0)}{2!} = \frac{-2}{2!} = -1$

$t_2(x) = f(x_0) + \frac{1}{1!} \cdot f'(x_0)(x-x_0) + \frac{1}{2!} f''(x_0)(x-x_0)^2 = 1 + 0 + \frac{1}{2!} \cdot (-2) \cdot x^2 = 1 - x^2$

Beispiel: $f(x) = \sin(x)$, $\quad x_0 = 1$ $\quad f'(x) = \cos(x)$, $\quad f''(x) = -\sin(x)$, $\quad f'''(x) = -\cos(x)$

$t_0(x) = f(x_0) = f(1) = sin(1)$

$t_1(x) = f(x_0) + \frac{1}{1!} \cdot f'(x_0)(x-x_0) = f(x_0) + \frac{1}{1!} \cdot \cos(1) \cdot (x-1) = sin(1) + \cos(1) \cdot (x-1)$

$t_2(x) = f(x_0) + \frac{1}{1!} \cdot f'(x_0)(x-x_0) + \frac{1}{2!} f''(x_0)(x-x_0)^2 = sin(1) + \cos(1) \cdot (x-1) + \frac{1}{2} \cdot \big(-\sin(1)\big)(x-1)^2$

$t_3(x) = f(x_0) + \frac{1}{1!} \cdot f'(x_0)(x-x_0) + \frac{1}{2!} f''(x_0)(x-x_0)^2 + \frac{1}{3!} f'''(x_0)(x-x_0)^3$

$= sin(1) + \cos(1) \cdot (x-1) - \frac{\sin(1)}{2}(x-1)^2 + \frac{1}{6} \cdot (-\cos(1))(x-1)^3$

Konvergenzradius einer Funktionsapproximation durch Taylor-Polynome

Eine Taylor-Entwicklung als unendliche Reihe ist oft nur innerhalb eines eingeschränkten Konvergenzbereichs rund um den Entwicklungspunkt gültig! Konvergenzradius $r = \frac{1}{\limsup_{n\to\infty} (\sqrt[n]{|a_n|})}$ einer Reihe. Oft konvergiert Taylor-Reihe nur für $|x| < r < \infty$ $\quad \limsup$ = **Limes superior** bedeutet grösster Häufungspunkt

Beispiel: $f(x) = \frac{1}{1+x^2} = (1+x^2)^{-1}$, $x_0 = 0$ $\quad$ siehe oben $\quad a_0 = 1, a_1 = 0, a_2 = -1$

Nun werden die **Beträge** der a_n betrachtet. Hier haben wir die beiden Zahlen $0, 1$ $(|-1| = 1)$

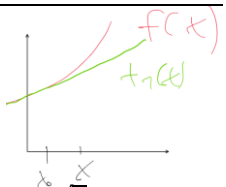
$r = \frac{1}{\limsup_{n\to\infty} (\sqrt[n]{|a_n|})} = \frac{1}{1} = 1$, $\quad$ d.h. konvergiert nur für $|x| < 1$

Fehlerschätzung per Restglied-Darstellung nach Lagrange mit Zwischenstelle z für exakten Fehler

$f(x) - t_k(x) = \frac{f^{(k+1)}(z)}{(k+1)!}(x-x_0)^{k+1}$ $\quad$ Dabei ist z aus dem Bereich zwischen x und x_0 aber unbekannt. Auch ohne Kenntnis des genauen Werts z kann man den Fehler abschätzen, falls man die entsprechende Ableitung begrenzen kann. Fehlerschranke: falls $\left|f^{(k+1)}\right| \le M$ im Intervall zwischen x und x_0, dann: $|f(x) - t_k(x)| \le \max_{z \in [x_0,\ x]} \left|\frac{f^{(k+1)}(z)}{(k+1)!}(x-x_0)^{k+1}\right| = \left|\frac{M}{(k+1)!}(|x-x_0|)^{k+1}\right|$

Linearisierung (als Einstieg für Newton-Verfahren)

$t_1(x) = f(x_0) + \frac{1}{1!} \cdot f'(x_0)(x-x_0)$ $\quad [x_0,\ x] = h$ $\quad x - x_0 = h,$



Beispiel: Approximative Berechnung von $\sqrt{5}$ $\quad \sqrt{5} = \sqrt{4+1} = \sqrt{x_0 + h}$

$f = \sqrt{x_0}$ = hier Wurzelfun. $f' = \frac{1}{2\sqrt{x_0}}$ =hier Ableitung von Wurzelfun. $\cong$ → Linearisierung

$x_0 = 4, h = 1, x - 4 = 1$ $\quad \sqrt{5} = \sqrt{4+1} \cong \sqrt{4} + \frac{1}{2\sqrt{4}}(5-4) = 2.25$ $\quad$ Erhalten 2.25 als Näherung für $\sqrt{5}$

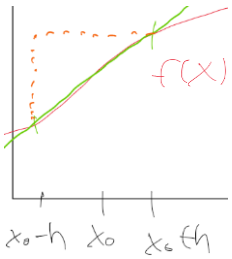
Beispiel: Approximative Berechnung von $\sqrt{5}$ $\quad \sqrt{5} = \sqrt{9-4} = \sqrt{x_0 + h}$

$x_0 = 9, h = -4, x - 9 = -5$ $\quad \sqrt{5} \cong \sqrt{9} + \frac{1}{2\sqrt{9}}(5-9) = 2.33$ $\quad$ Erhalten hier 2.33 als Näherung für $\sqrt{5}$

Beispiel: Approximative Berechnung von $\sqrt[3]{7}$ durch Linearisierung der dritten Wurzel in Punkt $x_0 = 8$

$\sqrt[3]{7} = \sqrt[3]{8-1} = \sqrt{x_0 + h}$ $\quad x_0 = 8, h = -1,$ $\quad f = \sqrt[3]{x_0},$ $\quad f' = \frac{1}{3 \cdot (x_0)^{\frac{2}{3}}}$ $\quad \sqrt[3]{7} \cong \sqrt[3]{8} + \frac{1}{3 \cdot (8)^{\frac{2}{3}}}(7-8) = 1.9167$

Numerisches Differentiation



$f'(x_0) \approx \frac{f(x_0+h)-f(x_0-h)}{2h} =: \Delta f(x_0)$ ist der zentrale Differenzenquotient an der Stelle x_0

Zentrale Frage: Wie gut ist Näherung $\Delta f(x_0)$ für $f'(x_0)$?

Taylor: $f(x_0+h) = f(x_0) + f'(x_0)h + \frac{1}{2!}f''(x_0)h^2 + \frac{1}{3!}f'''(z_1)h^3 \quad z_1 \in [x_0,\ x_0+h]$

$f(x_0-h) = f(x_0) - f'(x_0)h + \frac{1}{2!}f''(x_0)h^2 - \frac{1}{3!}f'''(z_2)h^3 \quad z_2 \in [x_0-h,\ x_0]$

$z_{1,2} =$ werden mit der Hilfe der Lagrangsche Restgliedschätzung dargestellt.

$\Delta f(x_0) = \frac{1}{2h}(2 \cdot f'(x_0)h + \frac{1}{3!}(f'''(z_1) + f'''(z_2))h^3 = f'(x_0) + \frac{1}{2\cdot 3!}(f'''(z_1) + f'''(z_2))h^2$

Somit (Verfahrens-)Fehler: $|\Delta f(x_0) - f'(x_0)| = \frac{1}{12} \cdot |f'''(z_1) + f'''(z_2)|h^2 \leq \frac{C}{12} \cdot h^2$ = Fehlerordnung

$C := \max_{\substack{z_1 \in [x_0, x_0+h] \\ z_2 \in [x_0-h, x_0]}} |f'''(z_1) + f'''(z_2)| \leq \frac{C}{12} \cdot h^2$ bedeutet hier, wenn wir Schrittweite h um den Faktor 10 verkleinern, dann wird der Fehler um den Faktor von ungefähr 100 reduziert.

SW4 – Nullstellen nichtlinearer Funktionen

Es gibt zwei Techniken zur Nullstellensuche. Eine beruht auf der Idee der binären Suche, die zweite verwendet die Linearisierung der Funktion. Allgemein kann f nicht nach $x^* = Nullstelle$ aufgelöst werden.

Zwischenwertsatz auf Nullstellensuche angewendet:

$\boldsymbol{f}: [\boldsymbol{a}, \boldsymbol{b}] \longrightarrow \mathbb{R}$ muss **stetig** sein. Man geht auf die Suche nach **zwei Funktionswerten** mit verschiedenen **Vorzeichen**. Hat man a, b gefunden mit $\boldsymbol{f(a) \cdot f(b) < 0}$ (d.h. $f(a)$ und $f(b)$ haben unterschiedliches Vorzeichen), so liegt dazwischen eine Nullstelle. Diese hat man mit den Schranken $[a, b]$ grob lokalisiert. Dazwischen können auch mehrere Nullstellen liegen!

Bisektionsverfahren mit Hilfe der binären Suche

Wenn $\boldsymbol{f\left(\frac{a+b}{2}\right) \cdot f(b) < 0}$ gilt: $x^* \in \left[\frac{a+b}{2}, \boldsymbol{b}\right]$ und wenn $\boldsymbol{f\left(\frac{a+b}{2}\right) \cdot f(b) > 0}$ gilt: $x^* \in \left[\boldsymbol{a}, \frac{a+b}{2}\right]$. $\frac{a+b}{2} = mid$

Wenn $\boldsymbol{f\left(\frac{a+b}{2}\right) \cdot f(b) = 0}$ herauskommt, ist eine Grenze bereits Nullstelle und wir sind glücklich! @stlk

Wenn $\boldsymbol{f(a) \cdot f(b) > 0}$ herauskommt, ist eine Bisektion unmöglich, dazwischen ist keine Nullstelle x^*

Mittelwert der Funktionsgrenzen. Wenn $\boldsymbol{f\left(\frac{a+b}{2}\right) < 0} \rightarrow \left[\frac{a+b}{2}, \boldsymbol{b}\right]$, wenn $\boldsymbol{f\left(\frac{a+b}{2}\right) > 0} \rightarrow \left[\boldsymbol{a}, \frac{a+b}{2}\right]$

Je nachdem ob Mittelwert Funktionsgrenzen kleiner/grösser als Null ist, werden die Grenzen angepasst.

Beispiel: $f(x) = x^2 - 3 = 0$ → Nullstelle intuitiv: $x^* = \sqrt{3}$ suchen nun Grenzen die je grösser/kleiner als Null sind. Diese sind hier bspw. $f(1) = 1^2 - 3 = -2 < 0$ und $f(2) = 2^2 - 3 = 1 > 0$ → $x^* \in [1,2]$

Mittelwert der Funktionsgrenzen: $m_1 = \frac{1+2}{2} = 1.5$ → Funktionswert: $f(m_1) = 1.5^2 - 3 = -\frac{3}{4} < 0$

Gemäss Gesetz oben gilt $f\left(\frac{a+b}{2}\right) < 0 \rightarrow \left[\frac{a+b}{2}, b\right]$, neue Grenze: $\left[\frac{a+b}{2}, b\right] = [1.5, 2] \rightarrow x^* \in [1.5, 2]$

Nun wieder gleiches Verfahren von vorne mit $[1.5, 2]$

Mittelwert der Funktionsgrenzen: $m_2 = \frac{1.5+2}{2} = 1.75$ → Funktionswert: $f(m_2) = 1.75^2 - 3 = 0.0625 > 0$

Gemäss Gesetz oben gilt $f\left(\frac{a+b}{2}\right) > 0 \rightarrow \left[a, \frac{a+b}{2}\right]$, neue Grenze: $\left[\frac{a+b}{2}, b\right] = [1.5, 1.75] \rightarrow x^* \in [1.5, 1.75]$

Dies nun so lange weiterführen, bis Intervall $[a, b]$ der gewünschten Genauigkeit entspricht: $\boldsymbol{|a - b| < tol}$

Pro/Contra des Bisektionsverfahrens: Konvergiert immer, ist dafür sehr langsam

Newton-Verfahren – Linearisierung

Grundidee: Linearisierung von Funktionen, dies bedeutet, dass die zu untersuchende Funktion durch eine lineare Funktion ersetzt wird (eine Gerade). Lösen der Ersatzfunktion ist einfacher als Ursprungsfunktion.

$f(x) \approx t_1(x) = f(x_0) + f'(x_0)(x - x_0) = 0$

$f(x)$ mit unserer Tangente zu approximieren und damit die Nullstelle der Tangente zu bestimmen.

$f(x_0)$ der Berührpunkt der Tangente, $f'(x_0)$ ist die Steigung der Tangente

Umstellen nach x resp. x_1: $x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$ wiederholen mit x_2: $x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$ usw. ergibt:

$\boldsymbol{x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, k = 0, 1, 2, \ldots}$ $x_0 \widehat{=}$ Startwert, kann beliebig gewählt werden. Am besten wählt man diesen Wert nahe an vermuteter Nullstelle.

Abbruchkriterium 1: Residuum $r = f(x_k) = f(x^*) + f'(z)(x_k - x^*)$ $z \in [x^*, x_k]$, $f'(z) =$ ist die Steigung der Sekante durch $f(x^*)$ und $f(x_k)$.

$|\boldsymbol{r}| = |\boldsymbol{f'(z)}| \cdot |\boldsymbol{x_k - x^*}|$, wobei $|x_k - x^*|$ = **Fehler der Näherung x^***, welches wir **klein halten** möchten.

Wenn $|f'(z)|$ sehr klein ist, ist auch das Residuum durch Multiplikation sehr klein. Der Fehler der Näherung der Nullstelle x^* kann jedoch trotzdem sehr gross sein.

I.A. garantiert ein kleines Residuum $|r|$ nicht, dass x_k auch nahe an der gesuchten Nullstelle x^* ist.

Abbruchkriterium 2: wenn $f(x_k + tol) \cdot f(x_k - tol) < 0$ → beschreibt wie bekannt Vorzeichenwechsel
Stellt sicher, dass Nullstelle x^* im folgenden Intervall liegt $x^* \in [x_k - tol, x_k + tol]$
Pro/Contra des Newton-Verfahrens: Konvergiert nicht immer, ist dafür bei Konvergenz sehr schnell.

SW5 – Nichtlineare Gleichungssysteme

Anwendung: Optimierung bzw. Minimierung einer Profitfunktion. Nullstellen von $g(x)$ mit der Hilfe vom Newton-Verfahren bestimmen: $x_{k+1} = x_k - \frac{g(x_k)}{g'(x_k)} = \frac{f'(x_k)}{f''(x_k)}$ Residuum: $r = |g(x_{k+1})| = |f'(x_{k+1})|$

Beispiel: Funktion: $f(x) = x^2 \cdot \sin(x)$, $f'(x) = 2x \cdot \sin(x) + x^2 \cdot \cos(x)$
$g'(x) = f''(x) = 2 \cdot \sin(x) + 2x \cdot \cos(x) + 2x \cdot \cos(x) + x^2 \cdot -\sin(x) = (2 - x^2) \cdot \sin(x) + 4x \cdot \cos(x)$
Lösen mit Python Programm. Je nach Startwert und Grenzen, kommen verschiedene Nullstellen heraus.

Differentialrechnung von $f: \mathbb{R}^2 \to \mathbb{R}$

Partielle Ableitung gemäss AN3 anwenden. $f: \mathbb{R}^2 \to \mathbb{R}$, **Beispiel:** $f(x,y) = x^2 + y^2$, $\begin{pmatrix} x_0 \\ y_0 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$

$\frac{\partial f}{\partial x} = f_x = 2x$ $f_x(1,2) = 2 \cdot 1 = 2$, $\frac{\partial f}{\partial y} = f_y = 2y$ $f_y(1,2) = 2 \cdot 2 = 4$

Linearisierung (Gleichung der Tangentialebene $t_1(x,y)$) $f(x_0, y_0) = 1^2 + 2^2 = 5$

$$t_1(x,y) = f(x_0,y_0) + \frac{\partial f}{\partial x}(x_0,y_0)(x - x_0) + \frac{\partial f}{\partial y}(x_0,y_0)(y - y_0) = f(x_0,y_0) + \left(\frac{\partial f}{\partial x}(x_0,y_0), \frac{\partial f}{\partial y}(x_0,y_0)\right)\begin{pmatrix} x - x_0 \\ y - y_0 \end{pmatrix}$$

Weiterführung des vorherigen Beispiels: $t_1(x,y) = 5 + (2,4)\begin{pmatrix} x-1 \\ y-2 \end{pmatrix} = 5 + 2(x-1) + 4(y-2)$

Jacobi-Matrix: $\vec{f}(x,y) = \begin{pmatrix} f_1(x,y) \\ f_2(x,y) \end{pmatrix}$ $\vec{f}: \mathbb{R} \to \mathbb{R}^L$ linearisiere komponentenweise in (x_0, y_0)

$$\overrightarrow{f_1}(x,y) = \begin{pmatrix} f_1(x_0,y_0) + \frac{\partial f_1}{\partial x}(x_0,y_0)(x - x_0) + \frac{\partial f_1}{\partial y}(x_0,y_0)(y - y_0) \\ f_2(x_0,y_0) + \frac{\partial f_2}{\partial x}(x_0,y_0)(x - x_0) + \frac{\partial f_2}{\partial y}(x_0,y_0)(y - y_0) \end{pmatrix}, \quad \vec{x}(x,y) = (x_1, x_2)$$

$$\overrightarrow{f_1}(\vec{x}) = \vec{f}(\vec{x}^{(0)}) + \begin{pmatrix} \frac{\partial f_1}{\partial x}(\vec{x}^{(0)}) & \frac{\partial f_1}{\partial y}(\vec{x}^{(0)}) \\ \frac{\partial f_2}{\partial x}(\vec{x}^{(0)}) & \frac{\partial f_2}{\partial y}(\vec{x}^{(0)}) \end{pmatrix}(\vec{x} - \vec{x}^{(0)}), \quad \text{Jacobi-Matrix } J(\vec{x}^{(0)}) = J\left(x_1^{(0)}, x_2^{(0)}\right)$$

Linearisierung: $\overrightarrow{t_1}(\vec{x}) = \vec{f}(\vec{x}^{(0)}) + J(\vec{x}^{(0)})(\vec{x} - \vec{x}^{(0)})$

Beispiel: $\vec{f}(x_1,x_2) = \begin{pmatrix} f_1(x_1,x_2) \\ f_2(x_1,x_2) \end{pmatrix} = \begin{pmatrix} 2x_1 + 4x_2 \\ 4x_1 + 8(x_2)^3 \end{pmatrix}$, komponentenweise ableiten: $J(x_1,x_2) = \begin{pmatrix} 2 & 4 \\ 4 & 24(x_2)^2 \end{pmatrix}$

Linearisierung in $(2,1)$: $\vec{f}(x_1,x_2) = \vec{f}(2,1) = \begin{pmatrix} f_1(x_1,x_2) \\ f_2(x_1,x_2) \end{pmatrix} = \begin{pmatrix} 2x_1 + 4x_2 \\ 4x_1 + 8(x_2)^3 \end{pmatrix} = \begin{pmatrix} 2 \cdot 2 + 4 \cdot 1 \\ 4 \cdot 2 + 8(1)^3 \end{pmatrix} = \begin{pmatrix} 8 \\ 16 \end{pmatrix}$

$$\overrightarrow{f_1}(x_1,x_2) = \begin{pmatrix} 8 \\ 16 \end{pmatrix} + \begin{pmatrix} 2 & 4 \\ 4 & 24(x_2)^2 \end{pmatrix}\begin{pmatrix} x_1 - 2 \\ x_2 - 1 \end{pmatrix} = \begin{pmatrix} 8 \\ 16 \end{pmatrix} + \begin{pmatrix} 2 & 4 \\ 4 & 24(1)^2 \end{pmatrix}\begin{pmatrix} x_1 - 2 \\ x_2 - 1 \end{pmatrix} = \begin{pmatrix} 8 \\ 16 \end{pmatrix} + \begin{pmatrix} 2 & 4 \\ 4 & 24 \end{pmatrix}\begin{pmatrix} x_1 - 2 \\ x_2 - 1 \end{pmatrix}$$

2D-Newton-Verfahren:

$\vec{f}(\vec{x}) = \begin{pmatrix} f_1(x_1,x_2) \\ f_2(x_1,x_2) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} = \vec{0}$, linearisiere in $\vec{x}^{(k)} = \begin{pmatrix} x_1^{(k)} \\ x_2^{(k)} \end{pmatrix}$, k = Iterationsindex

Betrachten lineare Gleichungssystem $J(\vec{x}^{(k)})(\vec{x}^{(k+1)} - \vec{x}^{(k)}) = -\vec{f}(\vec{x}^{(k)}) \Leftrightarrow A \cdot \vec{d} = b$, $\mathbb{R}^{2\times 2}$, $\vec{d} \in \mathbb{R}^2$, $\mathbb{R}^2$
System liefert den Vektor $\vec{d}$. Dieser wird dann in folgende Gleichung eingesetzt: $\vec{x}^{(k+1)} = \vec{x}^{(k)} + \vec{d}$

Abbruchkriterium: $\vec{f}(\vec{x}^{(k)}) = \begin{pmatrix} f_1\left(x_1^{(k)}, x_2^{(k)}\right) \\ f_2\left(x_1^{(k)}, x_2^{(k)}\right) \end{pmatrix} =: \begin{pmatrix} f_1 \\ f_2 \end{pmatrix}$, $r = \sqrt{f_1^2 + f_2^2} = \|\vec{f}(\vec{x}^{(k)})\|$, $\|\dots\|$ = Normschreibw.

Beispiel: $\vec{f}(\vec{x}) = \begin{pmatrix} 2x_1 + 4x_2 \\ 4x_1 + 8x_2^3 \end{pmatrix}$, $J(\vec{x}) = \begin{pmatrix} 2 & 4 \\ 4 & 24x^2 \end{pmatrix}$, $\vec{x}^{(0)} = \begin{pmatrix} 3 \\ 2 \end{pmatrix}$, $J(\vec{x}^{(0)}) = J(3,2) = \begin{pmatrix} 2 & 4 \\ 4 & 24 \cdot 2^2 \end{pmatrix} = \begin{pmatrix} 2 & 4 \\ 4 & 96 \end{pmatrix}$

$$f(\vec{x}^{(0)}) = \vec{f}(3,2) = \begin{pmatrix} 2x_1 + 4x_2 \\ 4x_1 + 8x_2^3 \end{pmatrix} = \begin{pmatrix} 2 \cdot 3 + 4 \cdot 2 \\ 4 \cdot 3 + 8 \cdot (2)^3 \end{pmatrix} = \begin{pmatrix} 14 \\ 76 \end{pmatrix}$$

$$\begin{pmatrix} 2 & 4 \\ 4 & 96 \end{pmatrix}\vec{d} = -\begin{pmatrix} 14 \\ 76 \end{pmatrix} \Rightarrow \vec{d} = -\begin{pmatrix} 5.909 \\ 0.545 \end{pmatrix}$$

Vektor $\vec{d}$ einsetzen: $\vec{x}^{(k+1)} = \vec{x}^{(k)} + \vec{d} \Leftrightarrow \vec{x}^{(0+1)} = \vec{x}^{(0)} + \vec{d} \Leftrightarrow \vec{x}^{(1)} = \vec{x}^{(0)} + \vec{d} \Leftrightarrow \vec{x}^{(1)} = \vec{x}^{(0)} + \vec{d}$

$$\Leftrightarrow \begin{pmatrix} 3 \\ 2 \end{pmatrix} + -\begin{pmatrix} 5.909 \\ 0.545 \end{pmatrix} = \begin{pmatrix} -2.909 \\ 1.4545 \end{pmatrix}$$

SW6 – Probezwischenprüfung

Viel schlimm…

SW7 – Lineare Gleichungssysteme

Betrachtung des linearen Gleichungssystems $Ax = b$

LR/LU–Zerlegung

Aufteilung der quadratischen Matrix in eine untere L und obere R oder U Dreiecksmatrix. Beginn bei oberen Dreiecksmatrix, wobei der aus LA bekannt G(r)auss-Algorithmus angeworfen wird (jetzt wird's ganz bitter hier). Für die untere Dreiecksmatrix werden die Hauptdiagonalelemente mit 1 gefüllt. Die restlichen Elemente werden anhand der Zeilenoperationen der Matrix hinzugefügt. **Achtung Vorzeichenwechsel!**

Beispiel: Gegeben Matrix: $\boldsymbol{A} = \boldsymbol{L} \cdot \boldsymbol{U} = \begin{pmatrix} 2 & 4 & 3 & 5 \\ -4 & -7 & -5 & -8 \\ 6 & 8 & 2 & 9 \\ 4 & 9 & -2 & 14 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -2 & 1 & 0 & 0 \\ 3 & -4 & 1 & 0 \\ 2 & 1 & 3 & 1 \end{pmatrix} \cdot \begin{pmatrix} 2 & 4 & 3 & 5 \\ 0 & 1 & 1 & 2 \\ 0 & 0 & -3 & 2 \\ 0 & 0 & 0 & -4 \end{pmatrix}$

$$A = \begin{pmatrix} 2 & 4 & 3 & 5 \\ 0 & 1 & 1 & 2 \\ 0 & -4 & -7 & -6 \\ 0 & 1 & -8 & 4 \end{pmatrix} \begin{matrix} I = I \\ II + 2I \\ III - 3I \\ IV - 2I \end{matrix} = \begin{pmatrix} 2 & 4 & 3 & 5 \\ 0 & 1 & 1 & 2 \\ 0 & 0 & -3 & 2 \\ 0 & 0 & -9 & 2 \end{pmatrix} \begin{matrix} I = I \\ II = II \\ III + 4II \\ IV - 1II \end{matrix} = \begin{pmatrix} 2 & 4 & 3 & 5 \\ 0 & 1 & 1 & 2 \\ 0 & 0 & -3 & 2 \\ 0 & 0 & 0 & -4 \end{pmatrix} \begin{matrix} I = I \\ II = II \\ III = III \\ IV - 3III \end{matrix} \coloneqq U = R$$

Beispiel Rückwärtseinsetzen: lin. GS $Ax = b$ $\left| \begin{matrix} x_1 - 2x_2 - x_3 = 3 \\ 2x_1 - x_2 + x_3 = 0 \\ 3x_1 - 6x_2 - 5x_3 = 3 \end{matrix} \right| \begin{matrix} I \\ II \\ III \end{matrix} \rightarrow \left(\begin{matrix} 1 & -2 & -1 \\ 2 & -1 & 1 \\ 3 & -6 & -5 \end{matrix} \middle| \begin{matrix} 3 \\ 0 \\ 3 \end{matrix} \right)$

$$= \left(\begin{matrix} 1 & -2 & -1 \\ 0 & 3 & 3 \\ 0 & 0 & -2 \end{matrix} \middle| \begin{matrix} 3 \\ -6 \\ -6 \end{matrix} \right) \begin{matrix} I \\ -2I = \boldsymbol{U} \\ -3I \end{matrix}, \boldsymbol{y} = \begin{pmatrix} 3 \\ -6 \\ -6 \end{pmatrix}, \begin{matrix} x_3 = 3 \\ 3x_2 + 3 \cdot 3 = -6 \Leftrightarrow 3x_2 = -15 \Leftrightarrow x_2 = -5 \\ x_1 + (-2) \cdot (-5) + (-1) \cdot (3) = 3 \Leftrightarrow x_1 = -4 \end{matrix}, L = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 0 & 1 \end{pmatrix},$$

$x = (-4, -5, 3)^T$ → Umsetzung in Python mit Befehl **x = backsubst(U, y)**, L könnte man sich sparen

---

Beispiel Vorwärtseinsetzen: lin. GS $A = \begin{pmatrix} 1 & -2 & -1 \\ 2 & -1 & 1 \\ 3 & -6 & -5 \end{pmatrix}, b = \begin{pmatrix} 3 \\ 0 \\ 3 \end{pmatrix}$, → $\begin{pmatrix} 1 & -2 & -1 \\ 0 & 3 & 3 \\ 0 & 0 & -2 \end{pmatrix} \begin{matrix} I \\ -2I = \boldsymbol{U} \\ -3I \end{matrix}$

$L = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 0 & 1 \end{pmatrix}$ → Umsetzung in Python mit Befehl **y = forwardsubst(L, b)**, danach Rückwärtseinsetzen um Lösung für x zu erhalten **x = backsubst(U, y)**, welche gleich ist wie oben.

Beispiel: Zeilenvertauschung, da keine Nullen in der Hauptdiagonalen (Pivot) sein sollen: $\boldsymbol{P} \cdot A = L \cdot U$

$$A = \begin{pmatrix} 0 & 2 & 2 \\ 1 & 2 & 1 \\ 2 & 1 & 0 \end{pmatrix} \begin{matrix} 1 \\ 2 \\ 3 \end{matrix} = \begin{pmatrix} 2 & 1 & 0 \\ 0 & 2 & 2 \\ 1 & 2 & 1 \end{pmatrix} \begin{matrix} 3 \\ 1 \\ 2 \end{matrix} = \begin{pmatrix} 2 & 1 & 0 \\ 0 & 2 & 2 \\ 1 & 2 & 1 \end{pmatrix} \begin{matrix} I \\ II \\ III \end{matrix} = \begin{pmatrix} 2 & 1 & 0 \\ 0 & 2 & 2 \\ 0 & \frac{3}{2} & 1 \end{pmatrix} \begin{matrix} I = I \\ II = II \\ III - \frac{1}{2}I \end{matrix} = \begin{pmatrix} 2 & 1 & 0 \\ 0 & 2 & 2 \\ 0 & \frac{3}{2} & 1 \end{pmatrix} \begin{matrix} I = I \\ II = II \\ III - \frac{3}{4}II \end{matrix} =$$

$$\begin{pmatrix} 2 & 1 & 0 \\ 0 & 2 & 2 \\ 0 & 0 & -\frac{1}{2} \end{pmatrix} \coloneqq U = R, \quad L = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \frac{1}{2} & \frac{3}{4} & 1 \end{pmatrix}, \quad P = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$ da Zeilentausch vorgenommen wurde!

$$P \cdot A = L \cdot U = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 & 2 & 2 \\ 1 & 2 & 1 \\ 2 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \frac{1}{2} & \frac{3}{4} & 1 \end{pmatrix} \begin{pmatrix} 2 & 1 & 0 \\ 0 & 2 & 2 \\ 0 & 0 & -\frac{1}{2} \end{pmatrix} = \begin{pmatrix} 2 & 1 & 0 \\ 0 & 2 & 2 \\ 1 & 2 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 1 & 0 \\ 0 & 2 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

Beurteilung einer Näherungslösung

Lin. GS $Ax = b$ mit theoretisch exakter Lösung $x = A^{-1}b$ sowie errechneter Näherungslösung $\tilde{x}$

Der Fehler $e \coloneqq x - \tilde{x}$ ist im allgemein unbekannt. Erinnerung an das Newton-Verfahren: Residuum.

Auch hier ist Residuum $r = b - A \cdot \tilde{x}$ messbar. Indirekte Fehler-Abschätzung: $\|r\| \leq \|A\| \cdot \|x - \tilde{x}\|$

Relativer Fehler einer Näherungslösung

Im 1D-Fall kann man aus dem relativen Fehler $e \coloneqq \frac{|x - \tilde{x}|}{|x|}$, sofern bekannt oder abgeschätzt, auf die Anzahl korrekter Dezimalstellen von $\tilde{x}$ schliessen. Abschätzung des relativen Fehlers: $\frac{\|e\|}{\|x\|} \leq \|A\| \cdot \|A^{-1}\| \cdot \frac{\|r\|}{\|b\|}$

Eine Matrix A mit grosser Konditionszahl $\kappa(A) = \|A\| \cdot \|A^{-1}\| \gg 1$ nennt man schlecht konditioniert.

Erkenntnis: in der numerischen Umsetzung des Gauss-Algorithmus ist nicht nur das exakte Null-Pivot störend, sondern auch allgemein kleine Pivots. → deshalb entsprechende Zeilenvertauschung vornehmen

---

SW8 – Ausgleichsrechnung (Regression)

Interpolation konstruiert Näherungswerte einer Funktion zwischen Punkten, an denen die Funktion exakt bekannt ist. **Regression** nimmt an, dass die **bekannten Funktionswerte Fehler enthalten**. Regression findet in einer gewählten Modellklasse (z.B. lineare Funktionen) dasjenige Modell, welches eine Fehlerfunktion (z.B. mittlere quadratische Abweichung) minimiert.

Abwägung bei Modellauswahl (sog. bias-variance tradeoff):

- komplexeres Modell: sensitiver gegenüber Datenänderungen, dafür besserer Fit der Daten
- einfacheres Modell: weniger guter Datenfit, dafür robust bei Datenänderungen und Voraussagen

Ausgleichspolynome aus Minimierung der Fehlerquadratsumme

Gegeben: Punktpaare $(x_1, y_1), \dots, (x_m, y_m)$ sowie vorgegebener Polynomgrad n, typisch: $m \gg n$.
Annahme: Messfehler in Ordinaten $y_1, \dots, y_m$. Gesucht: das beste Polynom p_n vom Höchstgrad n sodass **Fehlerquadratsumme** $F := \sum_{i=1}^{m}(p_n(x_i) - y_i)^2$ minimiert wird.
Matrixformulierung: finde Koeffizientenvektor $p \in \mathbb{R}^{n+1}$ für bestes Polynom p_n, sodass $F(p) = |A \cdot p - y|^2$ (das Quadrat der euklidischen Länge des Vektors) minimal ist, d.h. $|A \cdot p - y|^2 \leq |A \cdot q - y|^2$ für alle $q \in \mathbb{R}^{n+1}$. p minimiert also Länge des **Residuenvektors** $r = y - A \cdot p$

Designmatrix A: Allgemein $A = \begin{pmatrix} f_1(x_1) & f_2(x_1) & \cdots & f_m(x_1) \\ f_1(x_2) & f_2(x_2) & \cdots & f_m(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ f_1(x_n) & f_2(x_n) & \cdots & f_m(x_n) \end{pmatrix}$, wobei $f_1, \dots, f_m$ Basisfunktionen sind.

Normalengleichung: $(A^T \cdot A) \cdot p = (A^T \cdot y)$ → **p = np.linalg.solve(A.T@A, A.T@y)**

Beispiel: Ansatzpolynom vom Grad 2 → $p_2(x) = ax^2 + bx + c$ gesuchte Parameter sind (a, b, c).
Fehlerquadratsumme: $F(a,b,c) = \sum_{i=1}^{m}\big(r_i(a,b,c)\big)^2 = \sum_{i=1}^{m}(p_2(x_i,a,b,c) - y_i)^2 = \sum_{i=1}^{m}(ax_i^2 + bx_i + c - y_i)^2$

$$= \left| \begin{pmatrix} ax_1^2 + bx_1 + c \\ ax_2^2 + bx_2 + c \\ \vdots \\ ax_m^2 + bx_m + c \end{pmatrix} - \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix} \right|^2 = \left| \begin{pmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ & \vdots & \\ x_m^2 & x_m & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ \vdots \\ m \end{pmatrix} - \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix} \right|^2 = |A \cdot \vec{p} - \vec{y}|^2 = \vec{r} \to min\ ,\ \vec{r} = \text{Residuenvektor}$$

Gesucht ist $\vec{p}$, sodass die Fehlerquadratsumme $F(\vec{p}) = |A \cdot \vec{p} - \vec{y}|^2 \to min$ minimiert wird.

Beispiel: Gegeben Modell: $y = mx + b$ und Daten:

x_1	x_2	x_3
y_1	y_2	y_3

Gesucht $\vec{p}$ im Sinn der kleinsten Fehlerquadrate $\vec{r} = A \cdot \vec{p} - \vec{y}$

$$F(\vec{p}) = F(m,\ b) = \sum_{i=1}^{3}\big(mx + b - y_i\big)^2 = \left| \begin{pmatrix} mx_1 + b \\ mx_2 + b \\ mx_3 + b \end{pmatrix} - \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} \right|^2 = \left| \begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ x_3 & 1 \end{pmatrix} \begin{pmatrix} m \\ b \end{pmatrix} - \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} \right|^2 = |A \cdot \vec{p} - \vec{y}|^2$$

Minimale Fehlerquadratsumme wird gefunden durch 1. Ableitung = 0 setzen.

$$\frac{\partial F}{\partial m} = 0 = \frac{\partial}{\partial m}((mx_1 + b - y_1)^2 + (mx_2 + b - y_2)^2 + (mx_3 + b - y_3)^2) =$$
$$0 = \cancel{2}(mx_1 + b - y_1)x_1 + \cancel{2}(mx_2 + b - y_2)x_2 + \cancel{2}(mx_3 + b - y_3)x_3$$
$$0 = mx_1^2 + bx_1 - y_1x_1 + mx_2^2 + bx_2 - y_2x_2 + mx_3^2 + bx_3 - y_3x_3$$
$$0 = m(x_1^2 + x_2^2 + x_3^2) + b(x_1 + x_2 + x_3) - y_1x_1 - y_2x_2 - y_3x_3$$
$$y_1x_1 + y_2x_2 + y_3x_3 = m(x_1^2 + x_2^2 + x_3^2) + b(x_1 + x_2 + x_3)$$
$$\frac{\partial F}{\partial b} = 0 = \frac{\partial}{\partial b}((mx_1 + b - y_1)^2 + (mx_2 + b - y_2)^2 + (mx_3 + b - y_3)^2) =$$
$$0 = \cancel{2}(mx_1 + b - y_1) \cdot 1 + \cancel{2}(mx_2 + b - y_2) \cdot 1 + \cancel{2}(mx_3 + b - y_3) \cdot 1$$
$$0 = m(x_1 + x_2 + x_3) + 3b - y_1 - y_2 - y_3 \Leftrightarrow y_1 + y_2 + y_3 = m(x_1 + x_2 + x_3) + 3b$$

Lineares Gleichungssystem: $\begin{pmatrix} x_1^2 + x_2^2 + x_3^2 & x_1 + x_2 + x_3 \\ x_1 + x_2 + x_3 & 3 \end{pmatrix} \begin{pmatrix} m \\ b \end{pmatrix} = \begin{pmatrix} y_1x_1 + y_2x_2 + y_3x_3 \\ y_1 + y_2 + y_3 \end{pmatrix}$

→ Normalengleichung Δ liefert die gesuchten Parameter. $\vec{p} = \begin{pmatrix} m \\ b \end{pmatrix}$

$F(\vec{p}) = (A \cdot \vec{p} - \vec{y})^T \cdot (A \cdot \vec{p} - \vec{y})$ Optimalitätsbedingung: $\nabla F = 0 \Rightarrow (A^T \cdot A) \cdot p = (A^T \cdot y)$

$(A^T \cdot A) \cdot \vec{p} = (A^T \cdot \vec{y}) \Leftrightarrow \begin{pmatrix} x_1 & x_2 & x_3 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ x_3 & 1 \end{pmatrix} \begin{pmatrix} m \\ b \end{pmatrix} = \begin{pmatrix} x_1 & x_2 & x_3 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}$ gibt wiederum LGS von oben!

Beispiel: Gegeben Modell: $y = mx + b$ und Daten:

x_i	1	2	3	4
y_i	6	6.8	10	10.5

Haben somit $f_1(x) = x$ und $f_2(x) = 1$, $n = 4$

$$A = \begin{pmatrix} f_1(x_1) & f_2(x_1) \\ f_1(x_2) & f_2(x_2) \\ f_1(x_3) & f_2(x_3) \\ f_1(x_4) & f_2(x_4) \end{pmatrix} = \begin{pmatrix} f_1(1) & f_2(1) \\ f_1(2) & f_2(2) \\ f_1(3) & f_2(3) \\ f_1(4) & f_2(4) \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \\ 4 & 1 \end{pmatrix}, \quad (A^T \cdot A) = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \\ 4 & 1 \end{pmatrix} = \begin{pmatrix} 30 & 10 \\ 10 & 4 \end{pmatrix}$$

$$(A^T \cdot A) \cdot \vec{p} = (A^T \cdot \vec{y}) \Leftrightarrow \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \\ 4 & 1 \end{pmatrix} \begin{pmatrix} m \\ b \end{pmatrix} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 6 \\ 6.8 \\ 10 \\ 10.5 \end{pmatrix} \Rightarrow \begin{pmatrix} 30 & 10 \\ 10 & 4 \end{pmatrix} \begin{pmatrix} m \\ b \end{pmatrix} = \begin{pmatrix} 91.6 \\ 33.3 \end{pmatrix}, \begin{matrix} m = 1.67 \\ b = 4.15 \end{matrix}$$

Verbesserte Berechnung von p mit Orthogonalität

Umformung der Normalengleichung $(A^T \cdot A) \cdot p = (A^T \cdot y)$ durch Distributivgesetz und mit Residuum r zu $0 = A^T(y - A \cdot p) = (A^T \cdot r)$.
Interpretation via Skalarprodukt: das minimale Residuum ist senkrecht zu den Spalten von A.
Orthogonalprojektion von y auf Spalten Designmatrix A äquivalent, aber numerisch vorteilhafter.

SW9 – Interpolation

Interpolationsaufgabe kann auf zwei verschiedene Arten gelöst werden: entweder anhand eines **linearen Gleichungssystems** oder andererseits konstruktiv nach **Lagrange**.

Polynom-Interpolation als lineares Gleichungssystem

Ein Polynom $p(x) = a_n \cdot x^n + a_{n-1} \cdot x^{n-1} + \cdots + a_1 \cdot x + a_0$ vom Grad n hat $(n+1)$ Koeffizienten.
Interpolationsbedingungen: Sämtliche x-Werte $x_0, x_1, \dots, x_n$ müssen im Polynom eingesetzt $p(x_0), p(x_1), \dots$ die entsprechenden y-Werte $y_0, y_1, \dots$ ergeben. Dies sind $(n+1)$ Bedingungen!
Dies lässt sich durch das Lösen des linearen Gleichungssystems $V\,x = y$ erreichen, wobei $V = V(x_1, x_2, \dots, x_n)$ die sogenannte **Vandermonde-Matrix** ist.

$$\begin{matrix} a_0 + a_1x_0 + a_2x_0^2 + \cdots + a_nx_0^n = y_0 \\ a_0 + a_1x_1 + a_2x_1^2 + \cdots + a_nx_1^n = y_1 \\ \vdots \\ a_0 + a_1x_n + a_2x_n^2 + \cdots + a_nx_n^n = y_2 \end{matrix} \quad V(x_1, x_2, \dots, x_n) = \begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ 1 & x_2 & x_2^2 & \cdots & x_2^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

Beispiel: Gegeben seien 5 Datenpunkte →
Brauchen Polynom Grade **4!** $p_4(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + +a_4x^4$

x_i	0	0.5	1	1.5	2
y_i	1	1	0	0	3

$$\begin{pmatrix} 1 & x_0 & x_0^2 & x_0^3 & x_0^4 \\ 1 & x_1 & x_1^2 & x_1^3 & x_1^4 \\ 1 & x_2 & x_2^2 & x_2^3 & x_2^4 \\ 1 & x_3 & x_3^2 & x_3^3 & x_3^4 \\ 1 & x_4 & x_4^2 & x_4^3 & x_4^4 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix} \Leftrightarrow \begin{pmatrix} 1 & 0 & 0^2 & 0^3 & 0^4 \\ 1 & 0.5 & 0.5^2 & 0.5^3 & 0.5^4 \\ 1 & 1 & 1^2 & 1^3 & 1^4 \\ 1 & 1.5 & 1.5^2 & 1.5^3 & 1.5^4 \\ 1 & 2 & 2^2 & 2^3 & 2^4 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 3 \end{pmatrix}$$ nun LGS lösen.

Nachteil des Verfahrens: Vandermonde-Matrix hat eine schlechte Kondition.

Polynom-Interpolation nach Lagrange

Formel für Grundpolynome nach Lagrange: $\boldsymbol{l_i(x) = \frac{x-x_0}{x_i-x_0} \cdot \dots \cdot \frac{x-x_{i-1}}{x_i-x_{i-1}} \cdot \frac{x-x_{i+1}}{x_i-x_{i+1}} \cdot \dots \cdot \frac{x-x_n}{x_i-x_n}}$

Beispiel: Daten mit 3 Punkten → Polynom vom Grade 2

$l_0(x) = \frac{x-x_1}{x_0-x_1} \cdot \frac{x-x_2}{x_0-x_2}$, $l_1(x) = \frac{x-x_0}{x_1-x_0} \cdot \frac{x-x_2}{x_1-x_2}$, $l_2(x) = \frac{x-x_0}{x_2-x_0} \cdot \frac{x-x_1}{x_2-x_1}$

$\boldsymbol{p_2(x) = y_0 \cdot l_0(x) + y_1 \cdot l_1(x) + y_2 \cdot l_2(x)}$

$l_0(x_0) = 1,\ l_0(x_1) = 0,\ l_0(x_2) = 0$ $\quad p_2(x_0) = y_0 \cdot 1 + y_1 \cdot 0 + y_2 \cdot 0 = y_0$
$l_1(x_0) = 0,\ l_1(x_1) = 1,\ l_1(x_2) = 0$ $\quad p_2(x_1) = y_0 \cdot 0 + y_1 \cdot 1 + y_2 \cdot 0 = y_1$
$l_2(x_0) = 0,\ l_2(x_1) = 0,\ l_2(x_2) = 1$ $\quad p_2(x_2) = y_0 \cdot 0 + y_1 \cdot 0 + y_2 \cdot 1 = y_2$

Dies führt uns zur Regel: $\boldsymbol{L_j(x_i) = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}}$

Beispiel: $f(x) = \frac{1}{x}$ Daten mit 4 Punkten → Polynom vom Grade 3 $\quad x_0 = 1, x_1 = 2, x_2 = 3, x_3 = 4$

$l_0(x) = \frac{x-x_1}{x_0-x_1} \cdot \frac{x-x_2}{x_0-x_2} \cdot \frac{x-x_3}{x_0-x_3}$ $l_1(x) = \frac{x-x_0}{x_1-x_0} \cdot \frac{x-x_2}{x_1-x_2} \cdot \frac{x-x_3}{x_1-x_3}$ $l_2(x) = \frac{x-x_0}{x_2-x_0} \cdot \frac{x-x_1}{x_2-x_1} \cdot \frac{x-x_3}{x_2-x_3}$ $l_3(x) = \frac{x-x_0}{x_3-x_0} \cdot \frac{x-x_1}{x_3-x_1} \cdot \frac{x-x_2}{x_3-x_2}$

$l_0(x) = \frac{x-2}{1-2} \cdot \frac{x-3}{1-3} \cdot \frac{x-4}{1-4}$, $l_1(x) = \frac{x-1}{2-1} \cdot \frac{x-3}{2-3} \cdot \frac{x-4}{2-4}$, $l_2(x) = \frac{x-1}{3-1} \cdot \frac{x-2}{3-2} \cdot \frac{x-4}{3-4}$, $l_3(x) = \frac{x-1}{4-1} \cdot \frac{x-2}{4-2} \cdot \frac{x-3}{4-3}$

$l_0(x) = -\frac{1}{6}(x^3 - 9x^2 + 26x - 24)$, $l_1(x) = \frac{1}{2}(x^3 - 8x^2 + 19x - 12)$, $l_2(x) = -\frac{1}{2}(x^3 - 7x^2 + 14x - 8)$, $l_3(x) = \frac{1}{6}(x^3 - 6x^2 + 11x - 6)$

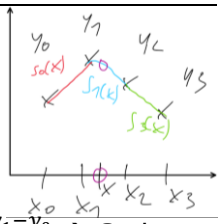
$p(x) = 1 \cdot l_0(x) + \frac{1}{2} \cdot l_1(x) + \frac{1}{3} \cdot l_2(x) + \frac{1}{4} \cdot l_3(x) = -\frac{1}{24}x^3 + \frac{5}{12}x^2 - \frac{35}{24}x + \frac{25}{12}$

Spline-Interpolation

Polynome können zwischen Stützstellen teilweise stark oszillieren.

Stückweise lineare Interpolation $\quad s(x) := \begin{cases} s_0(x) & falls\ x \in I_0 \\ s_1(x) & falls\ x \in I_1 \\ s_2(x) & falls\ x \in I_2 \end{cases}$

$I_0 = [x_0, x_1]$, $I_1 = [x_1, x_2]$, $I_2 = [x_2, x_3]$ führt zu Bedingungen: ↗
$s_0(x) = a_0 + b_0(x - x_0)$, $\quad s_1(x) = a_1 + b_1(x - x_1)$, $\quad s_2(x) = a_2 + b_2(x - x_2)$



Damit sind gesucht: $a_0,\ b_0,\ a_1,\ b_1,\ a_2,\ b_2$ → $3 \cdot 2 = 6$ Bedingungen
Interpolationsbedingungen: $y_0 = s_0(x_0) = a_0$, $\quad s_0(x_1) = a_0 + b_0(x_1 - x_0) = y_1$, $\quad b_0 = \frac{y_1 - y_0}{x_1 - x_0}$ → Steigung

$s_0(x_0) = y_0 + \frac{y_1 - y_0}{x_1 - x_0}(x - x_0)$, $\quad s_1(x_1) = y_1 + \frac{y_2 - y_1}{x_2 - x_1}(x - x_1)$, $\quad s_2(x_2) = y_2 + \frac{y_3 - y_2}{x_3 - x_2}(x - x_2)$

Problem: Auswertung kann nur innerhalb des Ausgangsintervall stattfinden (keine Extrapolation!)
Nachteil der stückweise linearen Interpolation: Funktion hat «Spitzen».

Stückweise kubische Splines – Kubische Polynome

Idee: Verwendung einer kubischen Funktion (x^3) auf jedem Teilintervall.
n Teilintervalle, also $4n$ Koeffizienten (davon $2n$ bestimmt durch Interpolationsbedingungen wie lin. Fall)
Zwei Freiheitsgrade pro Teilabschnitt mehr als stückweise lineare Interpolation; benutzt, um Funktion «runder» zu machen. Stetige erste und zweite Ableitung, an jedem von $(n-1)$ Übergängen:

$s_{i-1}^{(j)}(x_i) = s_i^{(j)}(x_i),\ \ i = 1, \dots, n-1,\ \ j = 1,2$

Beispiel von oben mit drei Intervallen:

$x \in I_0$: $s_0(x) = a_0 + b_0(x-x_0) + c_0(x-x_0)^2 + d_0(x-x_0)^3$
$x \in I_1$: $s_1(x) = a_1 + b_1(x-x_1) + c_1(x-x_1)^2 + d_1(x-x_1)^3$
$x \in I_2$: $s_2(x) = a_2 + b_2(x-x_2) + c_2(x-x_2)^2 + d_2(x-x_2)^3$

Total: $3 \cdot Anzahl\ Teilintervalle = 4 \cdot 3 = 12$ Koeffizienten a_k, b_k, c_k, d_k

Interpolationsbedingungen (Stetigkeit):

$s_0(x_0) = y_0,\ \ s_0(x_1) = y_1 \ \mid\ s_1(x_1) = y_1,\ \ s_1(x_2) = y_2 \ \mid\ s_2(x_2) = y_2,\ \ s_2(x_3) = y_3$ → 6 Bedingungen

Glattheitsbedingungen (Stetigkeit der 1. & 2. Ableitungen):

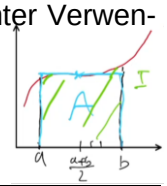
$s_0'(x_1) = s_1'(x_1),\ \ s_1'(x_2) = s_2'(x_2) \ \mid\ s_0''(x_1) = s_1''(x_1),\ \ s_1''(x_2) = s_2''(x_2)$ → 4 Bedingungen

Zusatzbedingungen:

Natürliche Splines: $s_0''(x_0) = 0,\ \ s_2''(x_3) = 0$ → 2 Bedingungen

SW10 – Numerische Integration (Quadratur)

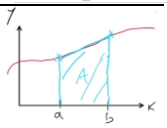
Numerische Integration approximiert ein bestimmtes Integral, ohne dass die Stammfunktion bekannt sein muss. Die Idee verallgemeinert die aus der Analysis bekannte Untersumme/Obersumme unter Verwendung von Interpolationstechniken.



Mittelpunktregel (Rechteckregel)

$I = \int_a^b f(x)dx$ Annäherung mit: $\boldsymbol{A \approx f\left(\frac{a+b}{2}\right) \cdot (b-a)}$

→ Polynome vom **Grad 1** können mit dieser Methode **exakt** dargestellt werden.



Trapezregel:

$I = \int_a^b f(x)dx$ Annäherung mit: $\boldsymbol{A \approx \frac{f(a)+f(b)}{2} \cdot (b-a)}$

→ Polynome vom **Grad 1** können mit dieser Methode **exakt** dargestellt werden.

Beispiel: Berechnen $A_1 = \int_{-1}^{1} 2x\,dx$ und $A_2 = \int_{-1}^{1} 3x^2\,dx$

Exakte Werte:	$A_1 = \int_{-1}^{1} 2x\,dx = x^2\|_{-1}^{1} = 0$	und	$A_2 = \int_{-1}^{1} 3x^2\,dx = x^3\|_{-1}^{1} = 2$
Mittelpunktregel:	$A_1 \approx f\left(\frac{-1+1}{2}\right) \cdot (1--1) = 0$	und	$A_2 \approx f\left(\frac{-1+1}{2}\right) \cdot (1--1) = 0$ ↯↯↯
Trapezregel:	$A_1 \approx \frac{-2+2}{2} \cdot (1--1) = 0$	und	$A_2 \approx \frac{3+3}{2} \cdot (1--1) = 6$ ↯↯↯

Konstruktion mit Hilfe der Interpolation

Idee: anstatt der Originalfunktion f integriere Interpolationspolynom (welches f gut approximiert).

Äquidistante Unterteilung des Intervalls $[a,\ b]$ mit $h = \frac{b-a}{n}$: $x_0 = a,\ x_1 = a+h,\ x_2 = a+2h, \dots,\ x_n = b$

Für gegebene $x-y$ Paare $\big(x_0, y_0 = f(x_0)\big), \dots, \big(x_n, y_n = f(x_n)\big)$ sind die Grundpolynome $l_i, 0 \le i \le n$ nach Lagrange vom Grad n, $\boldsymbol{l_i(x) = \frac{x-x_0}{x_i-x_0} \cdot \dots \cdot \frac{x-x_{i-1}}{x_i-x_{i-1}} \cdot \frac{x-x_{i+1}}{x_i-x_{i+1}} \cdot \dots \cdot \frac{x-x_n}{x_i-x_n}}$ und das dazugehörige Interpolationspolynom: $p(x) = y_0 \cdot l_0(x) + y_1 \cdot l_1(x) + \dots + y_n \cdot l_n(x)$

Benützen die Approximation: $\int_a^b f(x)\,dx \approx \int_a^b p(x)\,dx = \sum_{j=0}^{n} y_j \int_a^b l_j(x)\,dx$, wobei $\int_a^b l_j(x)\,dx := w_j$

Mit $n+1$ Knoten werden alle Polynome vom Grad n exakt integriert!

Beispiel: Gegeben sei auf Intervall $[-1,\ 1]$ die Quadraturformel: $Q_2 = w_0 \cdot f\left(-\frac{3}{4}\right) + w_1 \cdot f\left(\frac{1}{4}\right) + w_2 \cdot f\left(\frac{3}{4}\right)$

Bestimmen Sie Gewichte w_0, w_1, w_2, sodass alle Polynome vom Höchstgrad 2 exakt integriert werden:
Die zu den Knoten gehörenden Lagrange-Interpolationspolynome lauten

$l_0(x) = \frac{\left(x-\frac{1}{4}\right)}{\left(-\frac{3}{4}-\frac{1}{4}\right)}\frac{\left(x-\frac{3}{4}\right)}{\left(-\frac{3}{4}-\frac{3}{4}\right)} = \left(-x+\frac{1}{4}\right) \cdot -\frac{2}{3}\left(x-\frac{3}{4}\right) = \left(-x+\frac{1}{4}\right) \cdot \left(-\frac{2}{3}x+\frac{2}{4}\right) = \frac{2}{3}x^2 - \frac{2}{12}x - \frac{1}{2}x + \frac{1}{8} = \frac{2}{3}x^2 - \frac{2}{3}x + \frac{1}{8}$

$l_1(x) = \frac{\left(x--\frac{3}{4}\right)}{\left(\frac{1}{4}--\frac{3}{4}\right)}\frac{\left(x-\frac{3}{4}\right)}{\left(\frac{1}{4}-\frac{3}{4}\right)} = \left(x+\frac{3}{4}\right) \cdot -\frac{4}{2}\left(x-\frac{3}{4}\right) = \left(x+\frac{3}{4}\right) \cdot \left(-2x+\frac{6}{4}\right) = -2x^2 + \frac{3}{2}x - \frac{6}{4}x + \frac{9}{8} = -2x^2 + \frac{9}{8}$

$l_2(x) = \frac{\left(x--\frac{3}{4}\right)}{\left(\frac{3}{4}--\frac{3}{4}\right)}\frac{\left(x-\frac{1}{4}\right)}{\left(\frac{3}{4}-\frac{1}{4}\right)} = \frac{2}{3}\left(x+\frac{3}{4}\right) \cdot 2\left(x-\frac{1}{4}\right) = \left(\frac{2}{3}x+\frac{2}{4}\right) \cdot \left(2x-\frac{1}{2}\right) = \frac{4}{3}x^2 - \frac{1}{3}x + x - \frac{1}{4} = \frac{4}{3}x^2 + \frac{2}{3}x - \frac{1}{4}$ → $\int_a^b l_j(x)\,dx$

$\int_{-1}^{1} l_0(x)\,dx = \int_{-1}^{1} \left(\frac{2}{3}x^2 - \frac{2}{3}x + \frac{1}{8}\right) dx = \left(\frac{2}{3\cdot 3}x^3 - \frac{2}{2\cdot 3}x^2 + \frac{1}{8}x\right)|_{-1}^{1} = \frac{25}{36}$,

$\int_{-1}^{1} l_1(x)\,dx = \int_{-1}^{1} \left(-2x^2 + \frac{9}{8}\right) dx = \left(-\frac{2}{3}x^3 + \frac{9}{8}x\right)|_{-1}^{1} = \frac{11}{12}$, $\int_{-1}^{1} l_2(x)\,dx = \int_{-1}^{1} \left(\frac{4}{3}x^2 + \frac{2}{3}x - \frac{1}{4}\right) dx = \frac{7}{18}$

$w_0 = \frac{25}{36}, w_1 = \frac{11}{12}, w_2 = \frac{7}{18}$

Simpsonregel:

Beispiel Simpsonregel ($n = 2$): $S_2 := \frac{b-a}{6} \cdot f(a) + 4 \cdot \frac{b-a}{6} \cdot f\left(\frac{a+b}{2}\right) + \frac{b-a}{2} \cdot f(b)$

Allgemeine Gewichte auf $[a,\ b]$: $w_k : \frac{b-a}{2} \cdot \tilde{w}_k$

→ Polynome bis und mit **Grad 3** können mit dieser Methode **exakt** dargestellt werden.

Drei-Achtel Regel

Polynome **Grad** ≤ 3 auf Intervall $[a,\ b]$: $Q_3 := \frac{b-a}{8} \cdot f(a) + 3 \cdot \frac{b-a}{8} \cdot f\left(\frac{2a+b}{3}\right) + 3 \cdot \frac{b-a}{8} \cdot f\left(\frac{a+2b}{3}\right) + \frac{b-a}{8} \cdot f(b)$

Zusammengesetzte Newton-Cotes Formeln

Idee: Je kleiner Teilintervalle, desto besser stimmen Interpolationspolynome und Originalfunktion überein.

Unterteilung Intervall $[a,\ b]$ in n Teilintervalle der Grösse $h = \frac{b-a}{n}$, Anwendung Originalformel auf jedes Teilintervall.

Zusammengesetzte Mittelpunktregel: $M_0(h) = h \cdot \left[f\left(\frac{a+(a+h)}{2}\right) + f\left(\frac{(a+h)+(a+2h)}{2}\right) + \cdots + f\left(\frac{(a+(n-1)h)+b}{2}\right)\right]$

Zusammengesetzte Trapezregel: $T_1(h) = \frac{h}{2} \cdot [f(a) + 2f(a+h) + \cdots + 2f(a+(n-1)h) + f(b)]$

Zusammengesetzte Simpsonregel: $S_2(h) = \frac{h}{6} \cdot \left[f(a) + 4f\left(\frac{a+(a+h)}{2}\right) + 2f(a+h) + \cdots + f(b)\right]$

SW11 – Gewöhnliche Differentialgleichungen

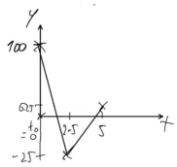
Numerische Lösung einer Differentialgleichung

Euler-Verfahren:

Idee zur numerischen Lösung: Differenzenquotient mit Zeitschrittweite h zur expliziten Berechnung von $y(t+h)$ aus $y(t)$. **Formel:** $y_{k+1} = y_k + h \cdot f(t_k, y_k)$

Beispiel: $y' = -\frac{1}{2}y$ → $f(t,y) = -\frac{1}{2}y, \quad y(0) = 100$ → $t_0 = 0, y_0 = 100$

2. Schritte mit Euler-Verfahren mit $h = 2.5$



1. $t_1 = t_0 + h = 0 + 2.5 = 2.5, \quad y_1 = y_0 + h \cdot f(t_0, y_0) = 100 + 2.5\left(-\frac{1}{2} \cdot 100\right) = -25$

2. $t_2 = t_1 + h = 2.5 + 2.5 = 5, \quad y_2 = y_1 + h \cdot f(t_1, y_1) = -25 + 2.5\left(-\frac{1}{2} \cdot (-25)\right) = 6.25$

Beispiel: Berechne zwei numerische Näherungswerte für AWP: $y' = 3y - e^{2x}$ mit $y(0.1) = 1$ indem zwei Schritte mit dem Euler-Verfahren im Intervall $0.1 \leq x \leq 0.3$ durchgeführt werden.

1. Schrittweite bestimmen: $h = \frac{b-a}{n} = \frac{0.3-0.1}{2} = 0.1$ $\quad P_0(x_0, y_0) = (0.1, 1)$

2. Koordinaten von $P_1(x_1, y_1)$ bestimmen: $x_1 = x_0 + h = 0.1 + 0.1 = 0.2$

$y_1 = y_0 + h \cdot f(x_0, y_0) = 1 + 0.1 \cdot (3 \cdot 1 - e^{2 \cdot 0.1}) = 1.1778$ $\quad P_1(x_1, y_1) = (0.2, 1.1778)$

3. Koordinaten von $P_2(x_2, y_2)$ bestimmen: $x_2 = x_1 + h = x_0 + 2h = 0.1 + 2 \cdot 0.1 = 0.3$

$y_2 = y_1 + h \cdot f(x_1, y_1) = 1.1778 + 0.1 \cdot (3 \cdot 1.1778 - e^{2 \cdot 0.2}) = 1.3820$ $\quad P_2(x_2, y_2) = (0.3, 1.3820)$

Allgemein AWP: $\vec{z}' = \vec{f}(t, \vec{z}), \quad \vec{z}(t_0) = \vec{z}_0$ AWP: $y_1' = -0.5y_1, \quad y_1(0) = 100$; $y_2' = 0.5y_1 - 0.2y_2, \quad y_2(0) = 100$

$\vec{z} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}, \quad \vec{z}' = \begin{pmatrix} y_1' \\ y_2' \end{pmatrix} = \begin{pmatrix} -0.5 & 0 \\ 0.5 & -0.2 \end{pmatrix} \vec{z} = \vec{f}(t, \vec{z}), \quad \vec{z}_0 = \begin{pmatrix} 100 \\ 100 \end{pmatrix}$

Euler-Verfahren für Systeme: $\vec{z}_{k+1} = \vec{z}_k + h \cdot \vec{f}(t_k, \vec{z}_k), \quad t_{k+1} = t_k + h$

Beispiel: $\vec{z}' = \begin{pmatrix} -0.5 & 0 \\ 0.5 & -0.2 \end{pmatrix} \vec{z}, \quad \vec{z}(0) = \begin{pmatrix} 100 \\ 100 \end{pmatrix} = \vec{z}_0, \quad t_0 = 0 \quad h = \frac{1}{2}$

1. $\vec{z}_1 = \vec{z}_0 + h \cdot \vec{f}(t_0, \vec{z}_0) = \begin{pmatrix} 100 \\ 100 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} -0.5 & 0 \\ 0.5 & -0.2 \end{pmatrix}\begin{pmatrix} 100 \\ 100 \end{pmatrix} = \begin{pmatrix} 100 \\ 100 \end{pmatrix} + \begin{pmatrix} -25 \\ 25 - 10 \end{pmatrix} = \begin{pmatrix} 75 \\ 115 \end{pmatrix}$

$t_1 = t_0 + h = \frac{1}{2}$

Transformation einer DGL höherer Ordnung in ein System 1. Ordnung

Beispiel: $y'' + ay' + by = 0$

Auflösen nach höchster Ordnung/Ableitung: $y'' = -by - ay' = f(t, y, y')$

Trick: setze $z_1 = y,\ z_2 = y'$ $\quad \vec{z} = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix}, \quad \vec{z}' = \begin{pmatrix} z_1' \\ z_2' \end{pmatrix} = \begin{pmatrix} y' \\ y'' \end{pmatrix} = \begin{pmatrix} y' \\ -by - ay' \end{pmatrix} = \begin{pmatrix} z_2 \\ -bz_1 - az_2 \end{pmatrix} =$

$= \begin{pmatrix} 0 & 1 \\ -b & -a \end{pmatrix} \vec{z} = \vec{f}(t, \vec{z})$

SW12 – Probezwischenprüfung 2.0

$u'' = u + 2 \cdot v' + \frac{-u}{(u^2+v^2)^{3/2}}, \quad v'' = v - 2 \cdot u' + \frac{-v}{(u^2+v^2)^{3/2}}$ $\quad u(0) = 1, v(0) = 2, u'(0) = 3, v'(0) = 4$

Viel schlimm 2.0…. Gesucht ist für das AWP ein äquivalentes System erster Ordnung:

$z_1 = u,\ z_2 = v,\ z_3 = u',\ z_4 = v' \rightarrow z_1' = z_3, z_2' = z_4,\ z_3' = z_1 + 2z_4 + \frac{-z_1}{(z_1^2+z_2^2)^{3/2}}, z_4' = z_2 - 2z_3 + \frac{-z_2}{(z_1^2+z_2^2)^{3/2}}$

Stand der Arbeit

SW	Vorlesung	AB/Aufgabe	Skript	ZF
1	☒	☒	☒	☒
2	☒	☒	☒	☒
3	☒	☒	☒	☒
4	☒	☐	☒	☒
5	☒	☐	☒	☒
6	☒	☒ ZEP	☒	☒
7	☒	☒	☒	☒
8	☒	☒	☒	☒
9	☒	☒	☒	☒
10	☒	☒	☒	☒
11	☒	☒	☒	☒
12	☐	☐ ZEP	☒	☐
13	☐	☐ Probe SEP	☒	☐
14	☒	☒	☒	☒

Modulauflagen

Kursnote

Die Semesterendprüfung mit Dauer 90 Minuten findet während der Prüfungswochen statt und zählt 100%. Die Teilnahme an der Semesterendprüfung ist obligatorisch. Unbegründete Absenz gemäß der Rahmenprüfungsordnung führt zu Nichtbestehen. Anspruch auf eine Nachprüfung im gleichen Prüfungszeitraum besteht nicht.

Zugelassene Hilfsmittel an Tests und Semesterendprüfung

Formelsammlung, selbst geschriebene Zusammenfassung (Test: 5 Blätter, SEP: 10 Blätter), Python, alle selbst erstellten Python-Programme