Sei $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $x \in \mathbb{R}^n$.

Beispiele zu kQL und MNL von $Ax = b$

x^* kQL $\Leftrightarrow A^tAx^* = A^tb$

x^+ MNL $\Leftrightarrow A^tAx^+ = A^tb$, $x^+ \in Bild(A^t)$.

1) A = Nullmatrix. Sei $x^* \in \mathbb{R}^n$ beliebig
$\Rightarrow 0^t 0 x^* = 0^t b \Rightarrow x^*$ kQL. $x^+ \in Bild(A^t) = \{0\}$.
$\Rightarrow x^+ = 0$.

2) $Ax = b$ besitze eine Lösung x^*.
$\Rightarrow A^tAx^* = A^tb \Rightarrow x^*$ ist kQL.
Sei $A^tA\bar{x} = A^tb = A^tAx^*$
$\Rightarrow A^tA(\bar{x} - x^*) = 0 \Rightarrow A(\bar{x} - x^*) = 0 \Rightarrow A\bar{x} = b$
Also: Genau die Lösungen von $Ax = b$ sind kQL.

3) Falls A invertierbar, so ist $x^* = A^{-1}b$ einzige kQL und gleichzeitig MNL.

$Ax^* = A(A^{-1}b) = b \Rightarrow x^*$ ist kQL nach (2).
Alle weiteren kQL müssen ebenfalls $Ax = b$ erfüllen $\Rightarrow x^*$ ist einzige kQL, und damit auch MNL.

4) $\begin{matrix} L = l_1 \\ L = l_2 \\ \vdots \\ L = l_n \end{matrix} \Rightarrow \underbrace{\begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}}_{A} \cdot L = \underbrace{\begin{pmatrix} l_1 \\ \vdots \\ l_n \end{pmatrix}}_{b}$

$A^tA = (1 \dots 1)\begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = n$, $A^tb = (1 \dots 1)\begin{pmatrix} l_1 \\ \vdots \\ l_n \end{pmatrix} = l_1 + \dots + l_n$.

$A^tAL = A^tb \Leftrightarrow n \cdot L = l_1 + \dots + l_n \Rightarrow L = \frac{l_1 + \dots + l_n}{n}$.

5) Zu Zeitpunkten t_i werden Werte y_i gemessen.
Es wird ein linearer Zusammenhang vermutet.

t_i	-2	0	1	1
y_i	-2	-4	4	6

$y(t) = m \cdot t + b$

GLS: $\begin{matrix} -2 \cdot m + b = -2 \\ 0 \cdot m + b = -4 \\ 1 \cdot m + b = 4 \\ 1 \cdot m + b = 6 \end{matrix} \quad \Big| \quad \underbrace{\begin{pmatrix} -2 & 1 \\ 0 & 1 \\ 1 & 1 \\ 1 & 1 \end{pmatrix}}_{A} \begin{pmatrix} m \\ b \end{pmatrix} = \underbrace{\begin{pmatrix} -2 \\ -4 \\ 4 \\ 6 \end{pmatrix}}_{B}$

$A^tA = \begin{pmatrix} -2 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}\begin{pmatrix} -2 & 1 \\ 0 & 1 \\ 1 & 1 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 6 & 0 \\ 0 & 4 \end{pmatrix}$

$A^tB = \begin{pmatrix} -2 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}\begin{pmatrix} -2 \\ -4 \\ 4 \\ 6 \end{pmatrix} = \begin{pmatrix} 14 \\ 4 \end{pmatrix}$

$A^tA\begin{pmatrix} m \\ b \end{pmatrix} = A^tB \Leftrightarrow \begin{pmatrix} 6 & 0 \\ 0 & 4 \end{pmatrix}\begin{pmatrix} m \\ b \end{pmatrix} = \begin{pmatrix} 14 \\ 4 \end{pmatrix}$

$m = \frac{14}{6}$, $b = 1 \Rightarrow y(t) = \frac{7}{3}t + 1$.

6) Gleicher Fall wie oben, aber es gibt nur eine Messung, nämlich $y(t_1) = y_1$.
$m \cdot t_1 + b = y_1$ $\quad (y(t) = m \cdot t + b)$

$\Rightarrow \underbrace{(t_1 \; 1)}_{A}\begin{pmatrix} m \\ b \end{pmatrix} = \underbrace{y_1}_{B}$

Erstmal: Das GLS ist lösbar $(m = 0, b = y_1)$.
Also sind die kQL alle Lösungen von $A\begin{pmatrix} m \\ b \end{pmatrix} = B$.
Wir suchen ein $\begin{pmatrix} m^+ \\ b^+ \end{pmatrix}$ aus dem Bild von A^t.

$A^t = \begin{pmatrix} t_1 \\ 1 \end{pmatrix}$ $\quad A^tu = \begin{pmatrix} t_1 u \\ u \end{pmatrix} = u \cdot \begin{pmatrix} t_1 \\ 1 \end{pmatrix}$

$\begin{pmatrix} m^+ \\ b^+ \end{pmatrix} \in Bild(A^t) \Leftrightarrow \exists u : \begin{pmatrix} m^+ \\ b^+ \end{pmatrix} = u\begin{pmatrix} t_1 \\ 1 \end{pmatrix}$

$m^+ \cdot t_1 + b^+ = y_1 \Rightarrow u \cdot t_1^2 + u = y_1$

$\Rightarrow u = \frac{y_1}{1 + t_1^2} \Rightarrow \begin{pmatrix} m^+ \\ b^+ \end{pmatrix} = \frac{y_1}{1 + t_1^2}\begin{pmatrix} t_1 \\ 1 \end{pmatrix}$.

7) Anmerkung: $n > m$ hat nicht notwendig Lösungen.
$n < m$ ist nicht notwendig unlösbar.

Beispiel: $\begin{matrix} x + y + z = 1 \\ x + y + z = 2 \end{matrix}$ keine Lösung

$\begin{matrix} x + y = 1 \\ 2x + 2y = 2 \\ 4x + 4y = 4 \end{matrix}$ viele Lösungen.

Die Pseudo-Inverse

Sei $A \in \mathbb{R}^{m \times n}$. Dann heißt

$A^+ \in \mathbb{R}^{n \times m}$, $A^+ b := x^+$, x^+ MNL von $Ax = b$ $\forall b \in \mathbb{R}^m$.

Pseudo inverse von A.

1) Sei A invertierbar. Dann gilt
$A^+ = A^{-1}$.
Beweis: $A^+ b = x^+ = A^{-1} b$.

2) Sei A die Nullmatrix. Dann ist A^+ Nullmatrix.
$A^+ b = x^+ = 0 = 0 \cdot b$

3) Sei $m > n$ (mehr Gleichungen als Unbekannte).
$A : \mathbb{R}^n \to \mathbb{R}^m$.

Sei A injektiv, also $\ker(A) = \{0\}$.
$\Rightarrow \ker(A^+A) = \{0\}$
$\Rightarrow A^+A$ injektiv.
$A^+A : \mathbb{R}^n \to \mathbb{R}^n \Rightarrow A^+A$ invertierbar.
$A^+A x^* = A^+ b \Rightarrow x^* = (A^+A)^{-1} A^+ b$
ist einzige LOL, also auch MNL.
Es gilt $A^+ b = (A^+A)^{-1} A^+ b$
$\Rightarrow A^+ = (A^+A)^{-1} A^+$.

4) Sei $m < n$.
$A : \mathbb{R}^n \to \mathbb{R}^m$
Sei A surjektiv, also $\text{Bild}(A) = \mathbb{R}^m$.
oder $\text{rang}(A) = m$.
Dann ist $Ax = b$ immer lösbar
$\Rightarrow$ Die LOL sind die Lösungen von $Ax = b$.
x^+ löst also $Ax = b$ und $x^+ \in \text{Bild}(A^+) \Rightarrow x^+ = A^+ y$.
$A \cdot A^+ y = b$. $A^+ : \mathbb{R}^m \to \mathbb{R}^n$.
$\dim \text{Bild}(A^+) = \text{rang}(A^+) = \text{rang}(A) = m$.
$m = \dim \text{Bild}(A^+) + \dim \ker(A^+)$
$\Rightarrow \dim \ker(A^+) = 0 \Rightarrow A^+$ injektiv.
$\Rightarrow A \cdot A^+$ injektiv.
$A \cdot A^+ : \mathbb{R}^m \to \mathbb{R}^m \Rightarrow A \cdot A^+$ invertierbar.
$A \cdot A^+ y = b \Rightarrow y = (A \cdot A^+)^{-1} \cdot b$
$\Rightarrow x^+ = A^+ y = A^+ (A \cdot A^+)^{-1} \cdot b$
$\Rightarrow A^+ = A^+ (A \cdot A^+)^{-1}$

Bemerkung: $AA^+A = A$, $A^+AA^+ = A^+$

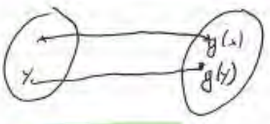
Iterative Lösg linearer GLS

Banachscher Fixpunktsatz

→ Bekannt aus der Analysis II, abstrakt
→ Hier: Anwendung auf den $\mathbb{R}^n$

$$g\left(\mathbb{R}^n, \|\cdot\|\right) \to \left(\mathbb{R}^m, \|\cdot\|\right)$$

Def.: g heißt **kontrahierend** (zusammenziehend).



$$\Rightarrow \exists\, q<1 : \|g(x)-g(y)\| \leq q\|x-y\| \quad \forall x,y \in \mathbb{R}^n$$

Def.: $\bar{x}$ heißt **Fixpunkt** einer Funktion g
$:\Leftrightarrow g(\bar{x}) = \bar{x}$.

Satz. **g kontrahierend $\Rightarrow$ g stetig.**

Sei $x_n \to x$. $\|g(x_n) - g(x)\| \leq q\|x_n - x\| \to 0$.
($x_n - x \to 0$)

$\Rightarrow g(x_n) \to g(x) \Rightarrow g$ ist stetig.

Satz Sei $(X, \|\cdot\|)$ vollständig (Banachraum).
Sei **$\emptyset \neq D \subset X$ abgeschlossen** (Jede CF in D ist Fkt in D).
Sei **$g: D \to D$ kontrahierend** mit
kontraktionskonstante $q<1$.
(4 Bedingungen!)
Dann hat g genau einen Fixpunkt.

Beweis: Sei $x^{(0)} \in D$ beliebig (geht, da $D \neq \emptyset$).
Definiere **$x^{(k+1)} := g(x^{(k)})$**. (Fixpunktiteration, geht wegen $g: D \to D$)
Behauptung: $x^{(k)}$ konvergiert gegen einen Fixpunkt $\bar{x}$.

I. $$\begin{aligned}\|x^{(k+1)} - x^{(k)}\| &= \|g(x^{(k)}) - g(x^{(k-1)})\| \\ &\leq q\|x^{(k)} - x^{(k-1)}\| \\ &\vdots \\ &\leq q^k \|x^{(1)} - x^{(0)}\|.\end{aligned}$$

II. Sei $l > k$.

$$\|x^{(l)} - x^{(k)}\| \leq \underbrace{\|x^{(l)} - x^{(l-1)}\|}_{\leq q^{l-1}\|x^{(1)}-x^{(0)}\|} + \underbrace{\|x^{(l-1)} - x^{(l-2)}\|}_{\leq q^{l-2}\|x^{(1)}-x^{(0)}\|} + \ldots + \|x^{(k+1)} - x^{(k)}\|$$

$$\leq \|x^{(1)} - x^{(0)}\| q^k \sum_{j=0}^{l-1-k} q^j$$

$$\leq \|x^{(1)} - x^{(0)}\| q^k \underbrace{\sum_{j=0}^{\infty} q^j}_{\frac{1}{1-q}} = \frac{q^k}{1-q}\|x^{(1)} - x^{(0)}\| \to 0$$

$\Rightarrow x^{(0)}$ ist CF.
$\Rightarrow x^{(k)} \to \bar{x}$ (X ist Banachraum)
$\Rightarrow \bar{x} \in D$ (D ist vollständig)
$x^{(k+1)} = g(x^{(k)}) \underset{k\to\infty}{\Rightarrow} \bar{x} = g(\bar{x})$, dann g ist stetig.
Sei $\tilde{x}$ ein Fixpunkt

$$\Rightarrow \|\tilde{x} - \bar{x}\| = \|g(\tilde{x}) - g(\bar{x})\| \leq q\|\tilde{x} - \bar{x}\| \quad (\Rightarrow q \geq 1, \text{ da } \|\tilde{x} - \bar{x}\| \neq 0)$$

$\Rightarrow \tilde{x} = \bar{x} \Rightarrow$ Fixpunkt eindeutig.

Also: Banach liefert nicht nur Existenz, sondern auch Algorithmus.

Fehlerabschätzung:

$$\|\bar{x} - x^{(k)}\| = \lim_{l\to\infty} \|x^{(l)} - x^{(k)}\| \leq \frac{q^k}{1-q}\|x^{(1)} - x^{(0)}\| \quad \text{a priori}$$

$y^{(0)} = x^{(k)} \Rightarrow y^{(1)} = g(y^{(0)}) = g(x^{(k)}) = x^{(k+1)}$,

$$\|\bar{x} - x^{(k+1)}\| = \|\bar{x} - y^{(1)}\| \leq \frac{q}{1-q}\|y^{(1)} - y^{(0)}\| = \frac{q}{1-q}\|x^{(k+1)} - x^{(k)}\|. \quad \text{a posteriori}$$

Beispiele Banachscher Fixpunktsatz

1) X Banachraum (vollständig)
2) $D \subset X$ abgeschlossen ($D = X$ zugelassen)
3) $g: D \to D$
4) g kontrahierend, $\exists q < 1 : \|g(x) - g(y)\| \leq q \|x - y\|$

$\Rightarrow$ $\exists$ eindeutigen Fixpunkt: $g(\bar{x}) = \bar{x}$

$x^{(0)} \in D$, $x^{(k+1)} := g(x^{(k)}) \Rightarrow x^{(k)} \to \bar{x}$

$\|x^{(k)} - \bar{x}\| \leq \frac{q^k}{1-q} \|x^{(1)} - x^{(0)}\|$, $\frac{q}{1-q} \|x^{(k)} - x^{(k-1)}\|$.

Idee: Approximiere $\bar{x}$: $g(\bar{x}) = \bar{x}$. Wähle $x^{(0)} \in D$, Fixpunktfolge, Abbruchkriterium.

Voraussetzung: Sei $x, y \in D$, g diffbar.

$g(x) - g(y) = g'(\xi) \cdot (x - y)$ [g': Jakobi-Matrix]

(MWS, ξ zwischen x und y)

$\Rightarrow \|g(x) - g(y)\| = \|g'(\xi)(x - y)\|$

$\leq \|g'(\xi)\| \|x - y\|$

↑ induzierte Matrixnorm

$q := \sup_{\xi \in D} \|g'(\xi)\|$

Lemma: D konvex, $q < 1 \Rightarrow g$ kontrahierend

$\exists_x \|g'(x)\| \geq 1 \Rightarrow g$ nicht kontrahierend.

1) $g(x) = Bx + c$, $g: \mathbb{R}^n \to \mathbb{R}^n$. ($\bar{x} = B\bar{x} + c$)

kontrahierend $\Leftrightarrow \|B\| < 1$.

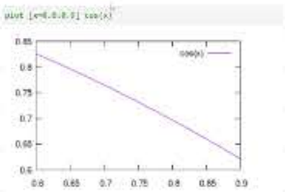
2) $g(x) := 0.9 \cos x$, $g: \mathbb{R} \to \mathbb{R}$.

3) $g(x) := \cos x$, $g: \mathbb{R} \to \mathbb{R}$.

4) $g(x) := \cos x$, $g: [-0.1, 0.1] \to \mathbb{R}$.

kontrahierend. $g: D \to D$?

5) $g(x) := \cos x$, $g: [0.6, 0.9] \to [0.6, 0.9]$



kontrahierend

monoton. $g(0.6) \sim 0.82 \in [0.6, 0.9]$

$g(0.9) \sim 0.62 \in [0.6, 0.9]$

$\Rightarrow g(x) \in [0.6, 0.9] \quad \forall x \in [0.6, 0.9]$

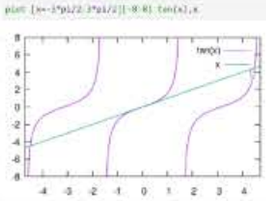
6) g kann in einer Norm kontrahierend sein, in einer anderen nicht.

$B = \frac{1}{10} \begin{pmatrix} 6 & 6 \\ 0 & 6 \end{pmatrix}$ $\|B\|_\infty = \frac{12}{10} = 1.2$ $\|B\|_2 = 0.97$

$\Rightarrow$ Wahl der Norm ist wichtig.

7) Formulierung ist wichtig.

Beispiel: $g(x) = \tan x \Rightarrow g'(x) = \frac{1}{\cos^2 x} \geq 1$.



$\bar{x} = \tan \bar{x}$

$\arctan \bar{x} + \pi = \bar{x}$

$D = \left[\frac{\pi}{2}, \frac{3\pi}{2}\right]$: $\frac{1}{1+x^2} \Rightarrow$ kontrah.

D enthält den gesuchten Fixpunkt. Wir führen die Fixpunktiteration durch mit dem Startwert $x_0 = \pi$ und erhalten

k	$x^{(k)}$	*a priori*	*a posteriori*	$\lvert\bar{x} - x^{(k)}\rvert$	$\lvert\bar{x} - x^{(k)}\rvert / \lvert\bar{x} - x^{(k-1)}\rvert$
1	3.1416			1.3518e + 000	
2	4.4042	1.4758e − 001	1.7744e + 000	8.9190e − 002	6.5978e − 002
3	4.4891	4.2563e − 002	1.1931e − 001	4.2900e − 003	4.8100e − 002
4	4.4932	1.2275e − 002	5.7439e − 003	2.0266e − 004	4.7239e − 002
5	4.4934	3.5401e − 003	2.7132e − 004	9.5871e − 006	4.7307e − 002
6	4.4934	1.0210e − 003	1.2804e − 005	4.7571e − 007	4.9619e − 002

Offensichtlich ist die a priori–Abschätzung viel zu pessimistisch, weil die Kontraktionskonstante zu groß abgeschätzt wurde.

Iterative Verfahren für Lineare Gleichungen

① Warum? Wir haben doch Gauss?
Wärmeleitung: Viele Nullen ⇒ spärlich, dünn besetzt.

$$\begin{pmatrix} 2 & -1 & & 0 \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ 0 & & -1 & 2 \end{pmatrix} \qquad \begin{pmatrix} 2 & -1 & \cdots & -1 \\ -1 & & & 0 \\ \vdots & & \ddots & \\ -1 & 0 & & 2 \end{pmatrix}$$

✓ ganz schlecht

Idee: Bx kann schnell berechnet werden, wenn B spärlich ist.

② Zu lösen sei $Ax = b$. Was können wir tun?

$x = g(x)$

① $x = A^{-1}b$, $g(x) := A^{-1}b$. (Blödsinn)

② $x = x + b - Ax = (I - A)x + b$
$=: g(x)$ (Neumannsche Reihe)

③ Verbinde ① + ②:
Sei $A = \begin{pmatrix} D & R \\ L & \ddots \end{pmatrix} = D + L + R$ (Aufteilung)

D invertierbar

$Ax = b \Leftrightarrow Dx = -(L+R)x + b$

$\Leftrightarrow x = \underbrace{-D^{-1}(L+R)}_{B} x + \underbrace{D^{-1}b}_{c} =: g(x)$

Gesamtschrittverfahren (GSV)
(Jacobi-Verfahren)

$Ax = b \Leftrightarrow (L+D)x = -Rx + b$

$\Leftrightarrow x = \underbrace{-(L+D)^{-1}R}_{B} x + \underbrace{(L+D)^{-1}b}_{c} =: g(x)$

Einzelschrittverfahren, Gauss-Seidel-Verfahren.

Bemerkung: Man berechnet natürlich nicht
$y = -(L+D)^{-1}Rx$, sondern löst $(D+L)y = -Rx$ durch Vorwärtseinsetzen.

Historische Idee von Gauss:

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1$$
$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2$$
$$a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3$$

① Wähle bel. Startvektor.

② Wähle x_1, so dass die 1. Gl. erfüllt ist.
Wähle x_2, so dass die 2. Gl. erfüllt ist.
x_3
→ x_1

Gauss: „Sie werden nie wieder eliminieren"

Bemerkung: Falls A dünn besetzt,
so ist g schnell berechenbar.

Triviales Beispiel:

$$\begin{pmatrix} 3 & 1 \\ 1 & 4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 4 \\ 5 \end{pmatrix} \qquad g(x) := Bx + c$$

$$D = \begin{pmatrix} 3 & 0 \\ 0 & 4 \end{pmatrix}, \; L = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}, \; R = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}.$$

① GSV: $B = -D^{-1}(L+R) = -\begin{pmatrix} \frac{1}{3} & 0 \\ 0 & \frac{1}{4} \end{pmatrix}\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$

$$= -\begin{pmatrix} 0 & \frac{1}{3} \\ \frac{1}{4} & 0 \end{pmatrix}, \quad c = D^{-1}b = \begin{pmatrix} \frac{1}{3} & 0 \\ 0 & \frac{1}{4} \end{pmatrix}\begin{pmatrix} 4 \\ 5 \end{pmatrix} = \begin{pmatrix} \frac{4}{3} \\ \frac{5}{4} \end{pmatrix}$$

$\|B\|_\infty = \frac{1}{3} =: q.$

$x^{(0)} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \; x^{(1)} = \begin{pmatrix} \frac{4}{3} \\ \frac{5}{4} \end{pmatrix} \quad \|x^{(1)} - x^{(0)}\|_\infty = \frac{4}{3} \Rightarrow \|x^{(4)} - \bar{x}\| \le \frac{\left(\frac{1}{3}\right)^4}{\frac{2}{3}} \cdot \frac{4}{3}$

$= 2 \cdot \left(\frac{1}{3}\right)^4$

$$x^{(2)} = Bx^{(1)} + c = \begin{pmatrix} -\frac{5}{12} \\ \frac{4}{12} \end{pmatrix} + \begin{pmatrix} \frac{4}{3} \\ \frac{5}{4} \end{pmatrix} = \begin{pmatrix} \frac{11}{12} \\ \frac{11}{12} \end{pmatrix} \approx \begin{pmatrix} 0.92 \\ 0.92 \end{pmatrix}$$

$\|x^{(2)} - \bar{x}\|_\infty \le \frac{q}{1-q} \quad \|x^{(2)} - x^{(1)}\|_\infty = \frac{5}{24}$

② ESV:

$B = -(D+L)^{-1}R, \quad c = (D+L)^{-1}b$

$x^{(0)} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}. \quad c = (D+L)^{-1}b$

$\Rightarrow (D+L)c = b \quad \begin{pmatrix} 3 & 0 \\ 1 & 4 \end{pmatrix}\begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} 4 \\ 5 \end{pmatrix}$

$\Rightarrow c_1 = \frac{4}{3}, \; c_2 = \frac{1}{4}\left(5 - \frac{4}{3}\right) = \frac{11}{12}$

$q = \frac{1}{3}. \quad x^{(2)} = \begin{pmatrix} 1.03 \\ 0.99 \end{pmatrix}$

Infimum der induzierten Matrixnormen

$g(x) := Bx + c$

Fixpunktfolge konvergent, wenn $q := \|B\| < 1$ irgendeiner induzierten Matrixnorm.

Frage: Wie klein kann man $\|B\|$ machen?

(1) Sei λ Eigenwert von B zum EV y

$$\Rightarrow \|B\| = \sup_{x \neq 0} \frac{\|Bx\|}{\|x\|} \geq \frac{\|By\|}{\|y\|} = |\lambda|$$

für jede induzierte Norm, insb. $\|B\| \geq \rho(B)$.

(2) $\forall \varepsilon, B$ gibt es eine induzierte Norm, so dass $\|B\| < \rho(B) + \varepsilon$.

Beweis: Technisch und Langwierig. Skript zur Num LA.

Statt dessen: Falls B pos. def., so gibt es eine induzierte Norm mit $\|B\| = \rho(B)$.

Beweis: $\|B\|_2^2 = \rho(B^t B) = \rho(B^2) = \rho(B)^2$

$$\Rightarrow \|B\|_2 = \rho(B).$$

Also: Falls alle Eigenwerte von B vom Betrag her < 1 sind, so gibt es eine Norm, in der $g(x) := Bx + c$ kontrahierend ist.

$\Rightarrow$ Das Fixpunktverfahren konvergiert für alle c und alle $x^{(0)}$ gegen einen Fixpunkt von g.

Falls ein EV y zum EW λ existiert mit $|\lambda| \geq 1$: Setze $c = 0$, $x^{(0)} = y$

$$\Rightarrow x^{(k)} = B^k y = \lambda^k y \not\to 0$$

Satz von Gerschgorin

(Grobe) Abschätzung für Eigenwerte von Matrizen

Sei $A \in \mathbb{C}^{n \times n}$.

Sei $r_i := \sum_{\substack{k=1 \\ k \neq i}}^{n} |A_{ik}|$ (Summe der Beträge der Außerdiagonalelemente in Zeile i)

Sei $K_i := \{z : |z - A_{ii}| \leq r_i\}$ (Kreise um A_{ii} mit Radius r_i in der komplexen Ebene)

1) Alle Eigenwerte liegen in der Vereinigung der Kreise K_i.

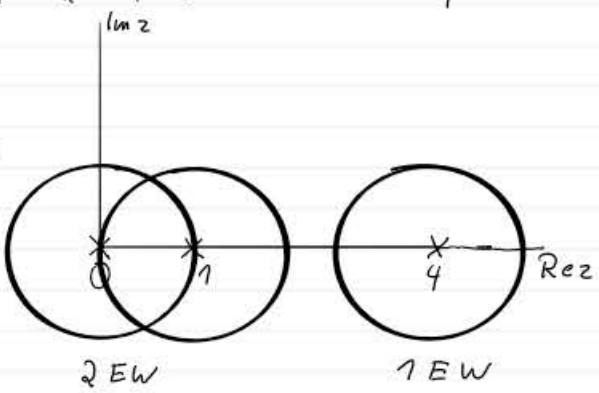
2) Falls die Vereinigung von m Kreisen disjunkt ist zum Rest, so befinden sich in dieser Vereinigung genau m Eigenwerte, der arithmetischen Vielfachheit nach gezählt.

Beispiel:

$$\begin{pmatrix} 2 & -1 & 0 \\ -1 & \ddots & -1 \\ 0 & -1 & 2 \end{pmatrix}$$

K_1 = Kreis um 2 mit Radius 1
$K_2 = K_{n-1}$ " " " " " 2
K_n " " " " " 1

$$\begin{pmatrix} 4 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \end{pmatrix} \Rightarrow$$



2 EW 1 EW

Beweis:

zu 1: Sei λ Eigenwert von A, $Ax = \lambda x$, $\|x\|_\infty = 1$, $|x_m| = 1$.

$$0 = \left((A - \lambda I)x\right)_m = \sum_{k=1}^{n} A_{mk} x_k - \lambda x_m$$

$$= (A_{mm} - \lambda) x_m + \sum_{\substack{k=1 \\ k \neq m}}^{n} A_{mk} x_k$$

$$\Rightarrow |A_{mm} - \lambda| = |(A_{mm} - \lambda) x_m| \leq \sum_{\substack{k=1 \\ k \neq m}}^{n} |A_{mk}| \underbrace{|x_k|}_{\leq \|x\|_\infty = 1} \leq r_m$$

$$\Rightarrow \lambda \in K_m.$$

zu 2: Beweis durch Film.

$$\begin{pmatrix} 1 & -4 \\ 3 & 2 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & -4t \\ 3t & 2 \end{pmatrix}$$

$t = 0$: EW 1, 2; $K_1 = \{1\}$, $K_2 = \{2\}$. alles korrekt.

$t = 1$: K_1 = Kreis um 1 mit Radius 9
K_2 = Kreis um 2 mit Radius 3

Idee:
- → Betrachte t von 0 bis 1.
- → Nullstellen eines Polynoms hängen stetig von den Koeffizienten ab.

Zeilensummenkriterien

Sei $A \in \mathbb{R}^{n\times n}$, $r_i := \sum_{k\neq i} |a_{ik}|$. (Gerschgorin). $\begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}$

$|a_{ii}| > r_i \Rightarrow |a_{ii} - 0| > r_i$ (stark diagonaldominant)

$\Rightarrow 0$ ist nicht in K_i

$\Rightarrow 0$ ist kein Eigenwert, A ist invertierbar.

GSV: $B = D^{-1}(L+R)$

$$= \begin{pmatrix} 0 & a_{ik} \\ a_{ik} & 0 \end{pmatrix} \begin{matrix} \uparrow \\ a_{nn} \\ \downarrow \end{matrix}$$

Gerschgorin: Kreise um 0 mit Radius $\frac{r_i}{|a_{ii}|} < 1$

$\Rightarrow$ EW von $B < 1$

$\Rightarrow$ GSV konvergiert.

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

Geht auch $\leq$? Nein:

$\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$

Geht auch $<$ / $\leq$? Nein:

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

Wir versuchen es trotzdem wie bei Gerschgorin, $B = D^{-1}(L+R)$.

Sei: $Bx = \lambda x$, $\|x\|_\infty = 1$, $|a_{mm}| > r_m$, $|a_{ii}| \geq r_i$.

1) Falls $|x_m| = 1$:

$$|\lambda| = |\lambda x_m| = |(Bx)_m| \leq \sum_{\substack{k=1 \\ k\neq m}}^{n} \frac{|a_{mk} x_k|}{|a_{m,m}|} \leq \frac{r_m}{|a_{m,m}|} < 1$$

2) Falls $|x_m| < 1$: Sei $|x_i| = 1$.

$$|\lambda| = |\lambda x_i| = |(Bx)_i| \leq \sum_{k\neq m} \frac{|a_{i,k}|\,|x_k|}{|a_{i,i}|} \uparrow \frac{r_i}{|a_{ii}|} \leq 1$$

$<$ wann?

$<$, falls $\exists i: |x_i| = 1, k: |x_k| < 1 : a_{ik} \neq 0$.

Def. $A \in \mathbb{R}^{n\times n}$ schwach diagonaldominant

$:\Leftrightarrow$ 1) $\forall i: |a_{ii}| \geq r_i$.

2) $\exists m: |a_{mm}| > r_m$

3) $\forall I \subset \{1..n\}, \emptyset \neq I \neq \{1..n\}$:
$\exists i \in I, k \notin I: a_{ik} \neq 0$.

Satz. A schwach diagonaldominant

$\Leftrightarrow g(x) := Bx + c$ ist kontrahierend für GSV

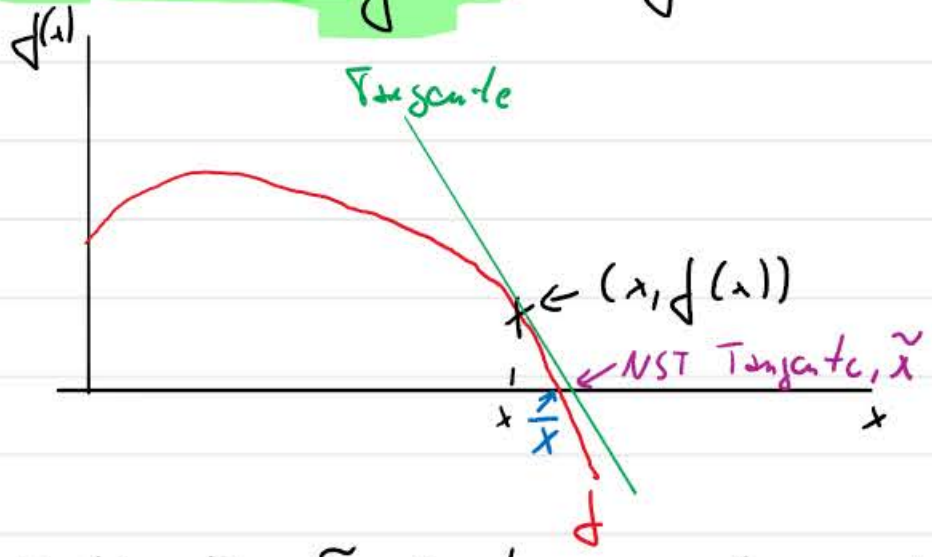
$\Leftrightarrow$ Das GSV konvergiert, A invertierbar, dann Fixpunkt ist eindeutig.

Bemerkung: Auch wichtig für ESV.

Das Newton-Verfahren

Sei $f: \mathbb{R} \to \mathbb{R}$ differenzierbar (nicht linear).
Gesucht ist eine Nullstelle $\bar{x}$ von f. [hier: 1-dim]

Idee von Graus (und anderen).
Sei x eine Näherung für eine Nullstelle.
Statt einer Nullstelle von f bestimme die Nullstelle $\tilde{x}$ der Tangenten an f im Punkt $(x, f(x))$.



In diesem Fall wäre $\tilde{x}$ eine bessere Approximation an $\bar{x}$ als x.

Tangentengleichung:

$$y(t) = f'(x) \cdot (t - x) + f(x) \quad [\text{geht durch } (x, f(x)), \text{ Steigung } f'(x)]$$

Nullstelle bei $t = x - \underbrace{f'(x)^{-1}}_{\in \mathbb{R}^{n \times n}} \cdot f(x)$

Iteratives Verfahren (Newton-Verfahren)

Sei $x^{(0)} \in \mathbb{R}$.

$$x^{(k+1)} := x^{(k)} - f'(x^{(k)})^{-1} \cdot f(x^{(k)})$$

Was ist, wenn $x^{(0)}$ völlig falsch ist?
Wir brauchen einen Konvergenzsatz.
$\Rightarrow$ Fixpunktverfahren mit $g(x) = x - f'(x)^{-1} \cdot f(x)$.

Newton: Konvergenz

Sei $f: \mathbb{R} \to \mathbb{R} \in C^2$, $x^{(0)} \in \mathbb{R}$, $x^{(k+1)} := g(x^{(k)})$, $f(\bar{x}) = 0$.

$g: \mathbb{R} \to \mathbb{R}$, $g(x) = x - f'(x)^{-1} \cdot f(x) = x - \frac{f(x)}{f'(x)}$. $f(\bar{x}) = 0 \Leftrightarrow \bar{x}$ ist Fixpunkt von g.

Wir wollen den Fixpunktsatz von Banach anwenden.

Suche D: $g: D \to D$, $|g'(x)| \leq q < 1$.

$$g'(x) = 1 - \frac{f'(x)^2 - f(x) f''(x)}{f'(x)^2} = \frac{f(x) \cdot f''(x)}{f'(x)^2}$$

Falls $f'(x) \neq 0$. Sei $f'(\bar{x}) \neq 0$.

$\Rightarrow g'(\bar{x}) = 0$, denn $f(\bar{x}) = 0$.

f', g' sind stetig $\Rightarrow \exists \varepsilon: |g'(x)| \leq \frac{1}{2}$, $|f'(x)| \geq \frac{|f'(\bar{x})|}{2} =: \mu \quad \forall x \in [\bar{x} - \varepsilon, \bar{x} + \varepsilon]$

$\Rightarrow$ Auf $D = [\bar{x} - \varepsilon, \bar{x} + \varepsilon]$ ist g kontrahierend, $q = \frac{1}{2}$.

Sei $x \in D$, also $|x - \bar{x}| \leq \varepsilon$.

$\Rightarrow |g(x) - \bar{x}| = |g(x) - g(\bar{x})| \leq \frac{1}{2} |x - \bar{x}| \leq \frac{\varepsilon}{2}$

$\Rightarrow g(x) \in D$, also $g: D \to D$.

Also: $x^{(0)} \in D \Rightarrow x^{(k)} \to \bar{x}$ (einziger Fixpunkt in D).

Taylor: $0 = f(\bar{x}) = f(x) + f'(x)(\bar{x} - x) + f''(\xi) \cdot (\bar{x} - x)^2 \cdot \frac{1}{2}$

$$|\bar{x} - x^{(k+1)}| = \left| \bar{x} - x^{(k)} + \frac{f(x^{(k)})}{f'(x^{(k)})} \right|$$

$$= \left| \bar{x} - x^{(k)} + \frac{-f'(x^{(k)})(\bar{x} - x^{(k)}) - f''(\xi) \cdot (x^{(k)} - \bar{x})^2 \cdot \frac{1}{2}}{f'(x^{(k)})} \right|$$

$$\leq \frac{1}{\mu} \cdot \|f''\|_\infty \cdot \frac{1}{2} \cdot (x^{(k)} - \bar{x})^2$$

„quadratische Konvergenz"

10^{-3} im 4. Schritt $\to 10^{-6} \to 10^{-12} \to 10^{-24}$

sehr schnelle Konvergenz.

Falls $f'(\bar{x}) = 0$: $g(x) = x - \frac{f(x)}{f'(x)}$, $g'(x) = \frac{f(x) f''(x)}{f'(x)^2}$.

Sei $f'(x) \neq 0$ in einer kleinen Umgebung von $\bar{x}$, $f''(\bar{x}) \neq 0$, $f \in C^4$.

l'Hospital für g: $\frac{f(x)}{f'(x)} \to \frac{f'(x)}{f''(x)} \xrightarrow{x \to \bar{x}} 0$

$\Rightarrow g(\bar{x}) = \bar{x}$ ist stetige Fortsetzung von g.

für g': $\frac{f(x) f''(x)}{f'(x)^2} \to \frac{f(x) f'''(x) + f'(x) f''(x)}{2 f'(x) \cdot f''(x)}$

$$\to \frac{f(x) f'''(x) + f'(x) f'''(x) + f'(x) f'''(x) + f''(x) f''(x)}{2\left(f''(x)^2 + f'(x) f'''(x)\right)}$$

$x = \bar{x}$: Wegen $f(\bar{x}) = f'(\bar{x}) = 0$

$\to \frac{1}{2} < 1 \Rightarrow$ konvergent falls $|x^{(0)} - \bar{x}|$ klein genug.

keine quadratische Konvergenz.

Newton Beispiele

a) Sei $a > 0$. $f(x) = x^2 - a$, $f'(x) = 2x$, $\bar{x} = \sqrt{a}$

$$g(x) = x - \frac{f(x)}{f'(x)} = x - \frac{x^2 - a}{2x} = \frac{1}{2}\left(x + \frac{a}{x}\right)$$

„Verfahren von Heron"

Behauptung: Das Verfahren konvergiert für alle $x^{(0)} > 0$. $(g: \mathbb{R}^+ \to \mathbb{R}^+)$

Beweis: 1. Alle $x^{(n)}$ sind wohldefiniert positiv.

2. $\sqrt{a}$ ist Fixpunkt von g.

3. $g(x)$ hat Minimum bei $\sqrt{a}$:

$0 = g'(x) = \frac{1}{2}\left(1 - \frac{a}{x^2}\right) \Leftrightarrow x = \sqrt{a}$, $g''(\sqrt{a}) > 0 \Rightarrow g(x) \geq \sqrt{a}$.

Also ist $g: [\sqrt{a}, \infty) \to [\sqrt{a}, \infty)$.

4. $x \in [\sqrt{a}, \infty) \Rightarrow |g'(x)| = \frac{1}{2}\left|1 - \frac{a}{x^2}\right| \leq \frac{1}{2}$

$\Rightarrow g$ kontrahierend auf $[\sqrt{a}, \infty)$.

Sei $x^{(0)} \in (0, \sqrt{a}] \Rightarrow x^{(1)} \in [\sqrt{a}, \infty) \Rightarrow x^{(k)}$ konvergiert.

b) $f(x) = a - \frac{1}{x}$, $f'(x) = \frac{1}{x^2}$, $\bar{x} = \frac{1}{a}$.

$$g(x) = x - \frac{f(x)}{f'(x)} = x - \frac{a - \frac{1}{x}}{\frac{1}{x^2}} = x - ax^2 + x = 2x - ax^2 = x(2 - ax)$$

1) Sei $x \in \left(0, \frac{1}{a}\right) \Rightarrow 2 - ax > 1 \Rightarrow g(x) > x \Rightarrow g$ monoton steigend.

2) $0 = g'(x) = 2(1 - ax) \Leftrightarrow x = \frac{1}{a} \Rightarrow g(x) \leq \frac{1}{a}$.

$\Rightarrow x^{(k)}$ monoton steigend + beschränkt $\Rightarrow x^{(k)}$ ist kgt.

3) Sei y der Limes, $x^{(k+1)} = g(x^{(k)}) \Rightarrow y = g(y)$

$\Rightarrow x^{(k)} \to \bar{x} = \frac{1}{a}$.

Eigenwerte

May 15, 2020

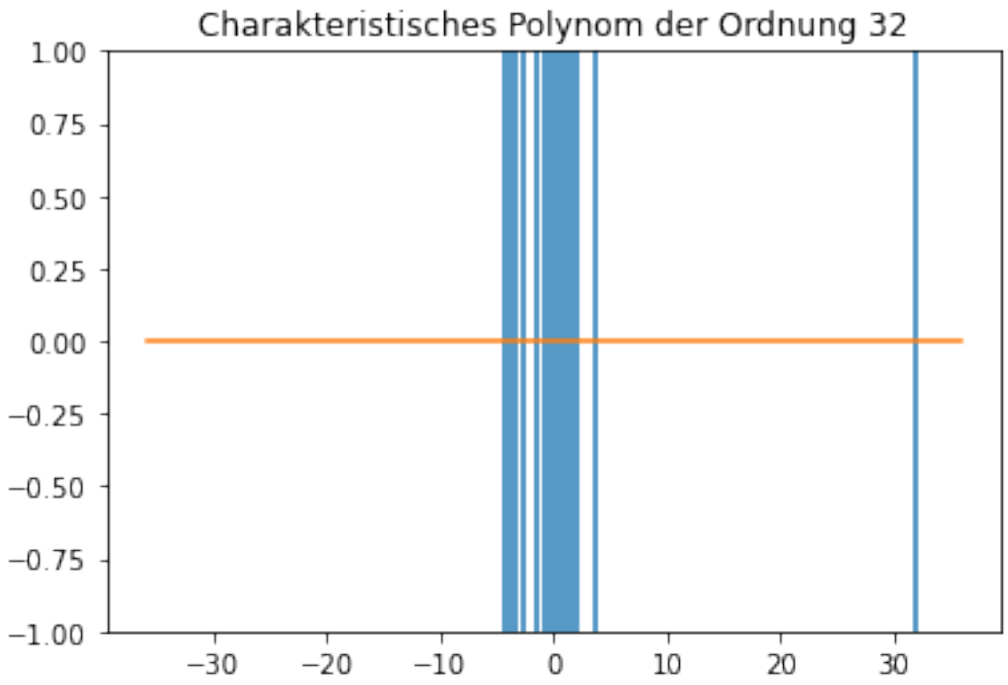
1 Eigenwerte

Wir wollen die Eigenwerte einer Matrix berechnen und schauen, ob man dies mit dem Newtonverfahren tun kann.

[1]:
```
import numpy as np
import math
import matplotlib.pyplot as plt
import sympy
```

[17]:
```
# lambda is a Python keyword.
def charpoly(A,lamda):
    return np.linalg.det(A-lamda*np.eye(N))
def dcharpoly(A,lamda):
    epsilon=1e-8
    return (charpoly(A,lamda+epsilon)-charpoly(A,lamda))/epsilon
def newton(A,lamda):
    return x-charpoly(A,lamda)/dcharpoly(A,lamda)
N=32
A=np.random.rand(N,N)
A=A+A.transpose()
L=np.linalg.norm(A,np.inf)
X=np.linspace(-L,L,256)
Y=[charpoly(A,lamda) for lamda in X]
plt.plot(X,Y,X,np.zeros(np.size(X)))
plt.ylim([-1,1])
plt.title('Charakteristisches Polynom der Ordnung '+str(N))
```

[17]:
```
Text(0.5, 1.0, 'Charakteristisches Polynom der Ordnung 32')
```



[18]:
```
x=32
for i in range(100):
    x=newton(A,x)
    print(x,charpoly(A,x))
```

```
32.03261665722028 5.226044929567333e+43
32.03161946649357 5.034139888136636e+40
32.03161850406691 4.882641209310934e+34
32.03161850406598 8.082805264090482e+31
32.03161850406598 8.082805264090482e+31
32.03161850406598 8.082805264090482e+31
32.03161850406598 8.082805264090482e+31
32.03161850406598 8.082805264090482e+31
32.03161850406598 8.082805264090482e+31
32.03161850406598 8.082805264090482e+31
32.03161850406598 8.082805264090482e+31
32.03161850406598 8.082805264090482e+31
32.03161850406598 8.082805264090482e+31
32.03161850406598 8.082805264090482e+31
32.03161850406598 8.082805264090482e+31
32.03161850406598 8.082805264090482e+31
32.03161850406598 8.082805264090482e+31
32.03161850406598 8.082805264090482e+31
32.03161850406598 8.082805264090482e+31
32.03161850406598 8.082805264090482e+31
32.03161850406598 8.082805264090482e+31
```

32.03161850406598 8.082805264090482e+31
32.03161850406598 8.082805264090482e+31
32.03161850406598 8.082805264090482e+31

```
32.03161850406598 8.082805264090482e+31
32.03161850406598 8.082805264090482e+31
32.03161850406598 8.082805264090482e+31
32.03161850406598 8.082805264090482e+31
32.03161850406598 8.082805264090482e+31
32.03161850406598 8.082805264090482e+31
32.03161850406598 8.082805264090482e+31
32.03161850406598 8.082805264090482e+31
32.03161850406598 8.082805264090482e+31
32.03161850406598 8.082805264090482e+31
32.03161850406598 8.082805264090482e+31
32.03161850406598 8.082805264090482e+31
32.03161850406598 8.082805264090482e+31
32.03161850406598 8.082805264090482e+31
32.03161850406598 8.082805264090482e+31
32.03161850406598 8.082805264090482e+31
32.03161850406598 8.082805264090482e+31
32.03161850406598 8.082805264090482e+31
32.03161850406598 8.082805264090482e+31
32.03161850406598 8.082805264090482e+31
32.03161850406598 8.082805264090482e+31
32.03161850406598 8.082805264090482e+31
32.03161850406598 8.082805264090482e+31
32.03161850406598 8.082805264090482e+31
32.03161850406598 8.082805264090482e+31
32.03161850406598 8.082805264090482e+31
32.03161850406598 8.082805264090482e+31
32.03161850406598 8.082805264090482e+31
32.03161850406598 8.082805264090482e+31
32.03161850406598 8.082805264090482e+31
32.03161850406598 8.082805264090482e+31
32.03161850406598 8.082805264090482e+31
```

[19]:
```
np.linalg.eigvals(A)
```

[19]:
```
array([ 3.20316185e+01, -4.46804258e+00, -4.03699430e+00,  3.80473375e+00,
        3.33004433e+00,  3.27645856e+00, -3.51194867e+00, -3.21824566e+00,
       -2.99484564e+00, -2.69196691e+00,  2.59025865e+00,  2.44135419e+00,
       -2.31118310e+00,  2.09821539e+00,  1.97897201e+00,  1.85728487e+00,
        1.56061124e+00, -2.14900963e+00, -1.81592500e+00, -1.74839285e+00,
       -1.41272876e+00,  1.05437690e+00,  7.67483106e-01,  6.73427681e-01,
        5.15948532e-01,  3.78739419e-01,  2.91298121e-01,  3.04994603e-02,
       -7.62990046e-01, -3.28757069e-01, -4.86653425e-01, -5.53510700e-01])
```

2 Satz von Perron/Frobenius

[20]:
```
x=np.random.rand(N)
for i in range(100):
    x=A.dot(x)
print(x)
x=x/np.linalg.norm(x)
print(x)
y=A.dot(x)
print(y)
print(y/x)
```

```
[2.00992190e+150 1.87230105e+150 2.19385105e+150 2.03811333e+150
 1.82434601e+150 1.57102371e+150 1.89032721e+150 2.00307980e+150
 1.92109845e+150 1.98914998e+150 1.97852036e+150 2.07477472e+150
 1.81709728e+150 1.76204644e+150 1.87781280e+150 1.91157389e+150
 2.06526032e+150 2.04718931e+150 1.97444355e+150 1.91657761e+150
 1.98496759e+150 2.02349679e+150 2.14273518e+150 1.93722909e+150
 2.05679230e+150 1.98291022e+150 2.06117307e+150 1.94107940e+150
 2.12401102e+150 1.95043759e+150 1.89650801e+150 1.84604785e+150]
[0.18104222 0.16864612 0.1976095  0.18358154 0.16432661 0.14150879
 0.17026981 0.18042592 0.17304151 0.1791712  0.17821375 0.18688379
 0.16367368 0.15871502 0.16914259 0.17218359 0.18602678 0.18439905
 0.17784653 0.1726343  0.17879448 0.18226497 0.19300527 0.17449447
 0.18526403 0.17860916 0.18565863 0.17484128 0.19131871 0.17568421
 0.17082655 0.16628139]
[5.79907525 5.40200826 6.32975208 5.88041387 5.26364721 4.5327556
 5.45401776 5.77933427 5.54279968 5.73914362 5.70847478 5.98619018
 5.242733   5.08389898 5.41791089 5.51531921 5.958739   5.9066001
 5.69671225 5.52975605 5.72707647 5.83824185 6.18227133 5.5893402
 5.93430686 5.72114051 5.94694636 5.60044922 6.12824793 5.62744967
 5.47185075 5.3262619 ]
[32.0316185 32.0316185 32.0316185 32.0316185 32.0316185 32.0316185
 32.0316185 32.0316185 32.0316185 32.0316185 32.0316185 32.0316185
 32.0316185 32.0316185 32.0316185 32.0316185 32.0316185 32.0316185
 32.0316185 32.0316185 32.0316185 32.0316185 32.0316185 32.0316185
 32.0316185 32.0316185 32.0316185 32.0316185 32.0316185 32.0316185
 32.0316185 32.0316185]
```

[ ]:

Potenzmethode

Sei $A \in \mathbb{R}^{n\times n}$ diagonalisierbar.
Sei $Ax_i = \lambda_i x_i$, x_i l.u., $i = 1..n$, $|\lambda_1| \geq |\lambda_2| \geq \dots \geq |\lambda_n|$, der mit ihrer algebraischen Vielfachheit nach gezählt.

$x^{(0)} \in \mathbb{R}^n$, $x^{(k+1)} = Ax^{(k)} \Rightarrow x^{(k)} = A^k x^{(0)}$.

$\exists \mu_i : x^{(0)} = \sum_{i=1}^{n} \mu_i x_i \Rightarrow x^{(k)} = \sum_{i=1}^{n} \mu_i A^k x_i = \sum_{i=1}^{n} \mu_i \lambda_i^k x_i$.

Fall 1: $\lambda_1 = \dots = \lambda_r$, $|\lambda_{r+1}| < |\lambda_r|$.

$$x^{(k)} = \sum_{i=1}^{r} \mu_i \lambda_1^k x_i + \sum_{i=r+1}^{n} \mu_i \lambda_i^k x_i$$

$$= \lambda_1^k \left(\underbrace{\sum_{i=1}^{r} \mu_i x_i}_{y_1 \text{ Eigenvektor}} + \underbrace{\sum_{i=r+1}^{n} \mu_i \frac{\lambda_i^k}{\lambda_1^k} x_i}_{\to 0,\ k\to\infty} \right)$$

$=: r^{(k)} \to y_1$.

$$\alpha^{(k)} = \frac{(x^{(k+1)}, d)}{(x^{(k)}, d)}$$

$$= \frac{\lambda_1^{k+1} (r^{(k+1)}, d)}{\lambda_1^k (r^{(k)}, d)} \xrightarrow[k\to\infty]{} \lambda_1, \text{ falls } (y, d) \neq 0$$

Geschw.: $\left(\frac{\lambda_2}{\lambda_1}\right)^k$ ⇒ wie Fixpunkt, q^k.

Bemerkung: Die Orthogonalraum ist Menge vom Maß 0, d.h. konvergenz für „fast alle" d.
Außer $y = 0 \Rightarrow \mu_i = 0 \Rightarrow$ Menge vom Maß 0
$\Rightarrow$ kgz für „fast alle" $x^{(0)}$ i.S.d. Analysis II.

Fall 2: $|\lambda_1| = |\lambda_2|$, $\lambda_1 \neq \lambda_2$. OE $|\lambda_3| < |\lambda_1|$.

$$x^{(k)} = \lambda_1^k \left(\underbrace{\mu_1 x_1}_{\text{EV}} + \underbrace{\left(\frac{\lambda_2}{\lambda_1}\right)^k \mu_2 x_2}_{\text{oszilliert!}} + \underbrace{\sum_{i=3}^{n} \mu_i \left(\frac{\lambda_i}{\lambda_1}\right)^k x_k}_{\to 0} \right)$$

$\Rightarrow \alpha^{(k)}$ konvergiert nicht (für fast alle ...)

Fall 3: A ist nicht diagonalisierbar.

Beispiel: $A = \begin{pmatrix} \lambda & 1 \\ 0 & \lambda \end{pmatrix} = \lambda I + \underbrace{\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}}_{N}$, $N^2 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$.

Einziger Eigenwert ist λ.

$$x^{(k)} = A^k x^{(0)} = (\lambda I + N)^k x^{(0)} = (\lambda^k I + k\lambda^{k-1} N) x^{(0)}$$

$$= \lambda^k \left(x^{(0)} + \frac{k}{\lambda} N x^{(0)} \right) = k \cdot \lambda^k \cdot \left(\underbrace{\frac{x^{(0)}}{k}}_{\to 0} + \underbrace{\frac{1}{\lambda} N x^{(0)}}_{=: r^{(k)} \to \frac{1}{\lambda} N x^{(0)}} \right)$$

$$\alpha^{(k)} = \frac{(x^{(k+1)}, d)}{(x^{(k)}, d)} = \underbrace{\frac{k+1}{k}}_{\to 1} \cdot \lambda \cdot \frac{(r^{(k+1)}, d)}{(r^{(k)}, d)} \to \lambda$$

Aber: $\frac{k+1}{k} = 1 + \frac{1}{k}$ konvergiert langsam.

Zusammenfassung: $\alpha^{(k)} = \frac{(x^{(k+1)}, d)}{(x^{(k)}, d)} \to \lambda_1$:

1. Falls $|\lambda_1| = |\lambda_2|$, $\lambda_1 \neq \lambda_2$: keine Konvergenz.
2. Falls $\lambda_1 = \dots = \lambda_r$, $|\lambda_1| > |\lambda_{r+1}|$: Konvergenz f.ü., schnell ($\left(\frac{\lambda_2}{\lambda_1}\right)^k$), $x_1 .. x_r$
3. Falls $\lambda_1 = \dots = \lambda_r$, $|\lambda_1| > |\lambda_{r+1}|$, nicht diagonalisierbar: Konvergenz f.ü., aber wie $\frac{1}{k}$ (langsam)

Eigenvektor: $x^{(k)}$ ist Näherung.

Fehlerabschätzung für Eigenwerte

Sei $A \in \mathbb{R}^{n \times n}$ symmetrisch.

Seien $\tilde{\lambda}, \tilde{x}$ Näherungen für Eigenwert / Eigenvektor von A.

Sei $d := A\tilde{x} - \tilde{\lambda}\tilde{x}$.

$\exists\, \lambda$: λ Eigenwert von A, $|\tilde{\lambda} - \lambda| \leq \frac{\|d\|_2}{\|\tilde{x}\|_2}$

Sei x_i ONB aus Eigenvektoren von A zu EW λ_i.

$\Rightarrow \tilde{x} = \sum_{i=1}^{n} \mu_i x_i$, $\|\tilde{x}\|_2^2 = \sum_{i=1}^{n} \mu_i^2$ (3.15, Teil 2).

$$d = A\tilde{x} - \tilde{\lambda}\tilde{x} = \sum_{i=1}^{n} \left(\mu_i A x_i - \tilde{\lambda}\mu_i x_i\right)$$

$$= \sum_{i=1}^{n} \mu_i (\lambda_i - \tilde{\lambda}) x_i$$

$$\Rightarrow \|d\|_2^2 = \sum_{i=1}^{n} \mu_i^2 (\lambda_i - \tilde{\lambda})^2 \geq \left(\sum_{i=1}^{n} \mu_i^2\right) \min_i (\lambda_i - \tilde{\lambda})^2 = \|\tilde{x}\|_2^2 \cdot \min_i (\lambda_i - \tilde{\lambda})^2$$

$$\Rightarrow \min_i (\lambda_i - \tilde{\lambda})^2 \leq \frac{\|d\|_2^2}{\|\tilde{x}\|_2^2}.$$

Korollar: Statt A sei nur $\tilde{A} = A + dA$ bekannt, $\tilde{A}, A, dA$ symmetrisch, $\tilde{x}$ sei Eigenvektor von $\tilde{A}$ zum EW $\tilde{\lambda}$.

$\Rightarrow \exists\, \lambda$: λ Eigenwert von A, $|\lambda - \tilde{\lambda}| \leq \frac{\|dA \cdot \tilde{x}\|_2}{\|\tilde{x}\|_2} \leq \|dA\|_2$

Beweis: $\|d\| = \|A\tilde{x} - \tilde{\lambda}\tilde{x}\| = \|(\tilde{A} - dA)\tilde{x} - \tilde{\lambda}\tilde{x}\|$

$= \|(dA) \cdot \tilde{x}\|$

$\Rightarrow$ Einsetzen.

Bemerkung: Geht so nur für symmetrische Matrizen.

Beispiele (+ Tricks)

$\begin{pmatrix} -3 & & \\ & 2 & \\ & & 1 \end{pmatrix}$ → schnelle Konvergenz gegen -3

$\begin{pmatrix} 3 & & \\ & 3 & \\ & & 1 \end{pmatrix}$ → " " " 3

$\begin{pmatrix} 3 & 1 & \\ & 3 & \\ & & 1 \end{pmatrix}$ → langsame " " 3

$\begin{pmatrix} 3 & & \\ & -3 & \\ & & 1 \end{pmatrix}$ → keine Konvergenz

Shift+I $\begin{pmatrix} 4 & & \\ & -2 & \\ & & 2 \end{pmatrix}$ → kgz gegen $4, -1 \to 3$.

→ **Statt A benutze $A+\mu I$**, ziehe μ ab.

Statt A benutze A^{-1} → betragskleinste EW werden zu betragsgrößten

$\begin{pmatrix} 3 & & \\ & 2 & \\ & & 4 \end{pmatrix} \to \begin{pmatrix} 1/3 & & \\ & 1/2 & \\ & & 1/4 \end{pmatrix} \to \frac{1}{2} \to$ Kehrwert 2.

Statt A benutze $(A-\mu I)^{-1}$: $\frac{1}{\lambda_k - \mu} \to$ Kehrwert $+\mu$ λ_k

$\begin{pmatrix} 4 & & \\ & 1 & \\ & & -3 \end{pmatrix}$ $\mu=2$ $\begin{pmatrix} \frac{1}{4-2} & & \\ & \frac{1}{1-2} & \\ & & \frac{1}{-3-2} \end{pmatrix}$ → $-1 \to \frac{1}{-1} + \mu = 1$

→ inverse Iteration nach Wieland

Interpolation

Gegeben $x_0 .. x_N \in \mathbb{R}$, paarweise verschieden. (Stützstellen)
$y_0 .. y_N \in \mathbb{R}$. (Stützwerte)

Suche Funktion p: $p(x_i) = y_i$.

Typisch, p ist Polynom,
$p \in \mathcal{P}_N = \{\text{Polynome } q : \text{Grad } q \le N\}$.

Ansatz: $p(x) = a_0 + a_1 x + .. + a_N x^N$.

$y_0 = p(x_0) = a_0 + a_1 x_0 + .. + a_N x_0^N$.

$$\Rightarrow \begin{pmatrix} 1 & x_0 & .. & x_0^N \\ \vdots & & & \\ 1 & x_N & .. & x_N^N \end{pmatrix} \begin{pmatrix} a_0 \\ \vdots \\ a_N \end{pmatrix} = \begin{pmatrix} y_0 \\ \vdots \\ y_N \end{pmatrix}$$

Vandermonde-Matrix, Quadratisch.

Satz: Vandermonde-Matrizen sind invertierbar.
Das Interpolationsproblem ist eindeutig lösbar.

Beweis: 1) Existenz: (Lagrange)

$\prod_{k \neq j} (x - x_k)$: Nullstellen x_k, $k \neq j$; $x = x_j$: $\prod_{k \neq j} (x_j - x_k)$
Polynomgrad N

$$w_j(x) := \frac{\prod_{k \neq j} (x - x_k)}{\prod_{k \neq j} (x_j - x_k)} \in \mathcal{P}_N, \quad w_j(x_k) = \delta_{k,j} = \begin{cases} 1, & k = j \\ 0, & k \neq j \end{cases}.$$

$$p := \sum_{j=0}^{N} y_j w_j(x) \Rightarrow p(x_k) = \sum_{j=0}^{N} y_j \delta_{k,j} = y_k$$

$p \in \mathcal{P}_N$ $\Rightarrow$ p ist Lösung der Interpolations-Aufgabe.

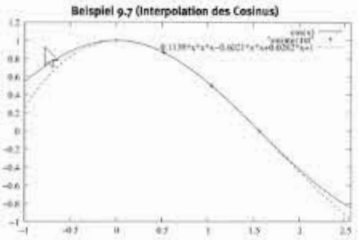
2) Eindeutig:
Seien $p_1, p_2 \in \mathcal{P}_n$ zwei Lösungen, $p = p_1 - p_2$.
$p(x_k) = p_1(x_k) - p_2(x_k) = y_k - y_k = 0$, $k = 0 .. N$
$\Rightarrow$ $p \in \mathcal{P}_N$ hat $N+1$ Nullstellen
$\Rightarrow$ $p = 0$ (Fundamentalsatz der Algebra)
$\Rightarrow$ $p_1 = p_2$.

Interpolationsfehler

Interpolationsaufgabe mit $y_i = f(x_i)$, $f \in C^{(N+1)}[a,b]$.

Sei $p \in \mathcal{P}_n$, $p(x_i) = y_i = f(x_i)$.

Wie groß wird der Fehler $f(x) - p(x)$?



Satz: $\forall \bar{x} \; \exists \xi \in [a,b]$: $f(\bar{x}) - p(\bar{x}) = \frac{f^{(N+1)}(\xi)}{(N+1)!} w(\bar{x})$,

$w(x) = \prod_{k=0}^{N} (x - x_k)$, $w \in \mathcal{P}_{N+1}$, $w(x_k) = 0$.

Korollar: $|f(\bar{x}) - p(\bar{x})| \leq \frac{\|f^{(N+1)}\|_\infty}{(N+1)!} |w(\bar{x})|$

$\|f - p\|_\infty \leq \frac{\|f^{(N+1)}\|_\infty}{(N+1)!} \|w\|_\infty$

Beweis: 1) $\bar{x} = x_k$: Trivial.

2) $\bar{x} \neq x_k \; \forall k \Rightarrow w(\bar{x}) \neq 0$.

$F(x) := \underbrace{f(x) - p(x)}_{=0,\; x = x_k} - K \underbrace{w(x)}_{=0,\; x = x_k}$

$\Rightarrow F(x_k) = 0, \; k = 0..N.$

$K = \frac{f(\bar{x}) - p(\bar{x})}{w(\bar{x})} \Rightarrow F(\bar{x}) = 0$ $\Rightarrow$ F hat (N+2) Nullstellen.

$F \in C^{(N+1)}$.

Satz von Rolle: Zwischen 2 NST von F liegt (mindestens) eine NST von F'.

F hat (N+2) NST $\Rightarrow$ F' hat (N+1) NST

$\ldots \Rightarrow F^{(N+1)}$ hat eine NST.

$\Rightarrow \exists \xi : F^{(N+1)}(\xi) = 0$

$f^{(N+1)}(\xi) - K w^{(N+1)}(\xi)$

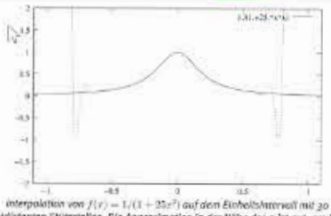
$w = \prod_{k=0}^{N} (x - x_k) = x^{N+1} + \ldots$

$\Rightarrow w^{(N+1)} = (N+1)!$

K einsetzen $\Rightarrow f(\bar{x}) - p(\bar{x}) = w(\bar{x}) \frac{f^{(N+1)}(\xi)}{(N+1)!}$

Was, wenn f nicht so schön ist...?

Beispiel von Runge: $f(x) = \frac{1}{1 + 25x^2}$ auf $[-1,1]$.



Interpolation von $f(x) = 1/(1 + 25x^2)$ auf dem Einheitsintervall mit 30 äquidistanten Stützstellen. Die Approximation in der Nähe der 0 ist gut, am Rand beliebig schlecht.

Folgerung: Es ist nicht mehr klar, dass für $N \to \infty$ die Interpolationen gegen die Funktion konvergieren.

Polynom vom Grad 30? $\Rightarrow$ Nächstes Kapitel.

Wie bekommen wir den Fehler klein?

$\|f - p\|_\infty \leq \frac{\|f^{(N+1)}\|_\infty}{(N+1)!} \|w\|_\infty$

$w(x) = \prod_{k=0}^{N} (x - x_k)$. Wähle x_k geschickt

$\Rightarrow$ Tschebyscheff-Interpolation.

$T_N(x) = \cos(N \cdot \arccos x)$

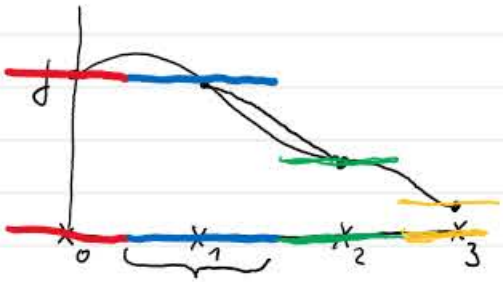
Satz. T_N ist Polynom mit Höchstkoeffizient 2^{N-1}.

$\|f - p\|_\infty \leq \frac{\|f^{(N+1)}\|_\infty}{(N+1)! \cdot 2^N}$, falls x_k als Nullstellen von T_{N+1} gewählt wurden.

Beweis: Übungen.

Splines

Motivation: Tabellen



Stützstellen $x_0 .. x_n$

Interpolation auf jedem Intervall,

1 Wert: $W=0$, Interpolation mit konstanter Funktion

Fehler: $\frac{\|f'\|_\infty}{1!} \underbrace{(x-x_1)}_{|\cdot| \le \frac{1}{2N}} \Rightarrow \|f-p\|_\infty \le \|f'\|_\infty \cdot \frac{1}{2N} \to 0,\ N\to\infty.$

konvergenz ▷

Idee: „Splines", „Straklatten"

Def: Sei $s_0 = a < s_1 < s_2 < .. < s_n = b$.
Eine Funktion $s: [a,b] \to \mathbb{R}$ heißt Spline der Ordnung k zu den Knoten $s_0 .. s_n$, falls:

1) $s|_{[s_i, s_{i+1}]} \in \mathcal{P}_{k-1}$

2) $s \in C^{k-2}$ $(k \ge 2)$

Die Interpolation mit dieser Funktionen heißt Spline-Interpolation.

Bemerkung: _ ¯ ¯ ist Spline der Ordnung 1.

Wie kommen die zu diesem Namen?

Anwendungen der Polynominterpolation

Es seien einige Auswertungen einer Funktion bekannt. Wie differenziert, integriert man?

Idee: Berechne Interpolationspolynom, differenziere/integriere dieses.

Problem: Hoher Polynomgrad.
⇒ Betrachte Teilintervalle.

① Differentiation

a) Gegeben $f(x), f(x+h)$. Approximiere $f'(x)$.

$$p(t) = f(x) + \frac{f(x+h)-f(x)}{h}(t-x) \Rightarrow p \in \mathcal{P}_1, p(x)=f(x), p(x+h)=f(x+h).$$

$$p'(x) = \frac{f(x+h)-f(x)}{h} = D_h^+(f)(x). \quad \text{(rechtsseitiger Differenzenquotient)}$$

b) $f(x-h), f(x) \Rightarrow f'(x)$:

$$p(t) = f(x) + \frac{f(x-h)-f(x)}{h}(x-t)$$

$$p'(x) = \frac{f(x)-f(x-h)}{h} =: D_h^-(f)(x) \quad \text{(linksseitig ...)}$$

c) $f(x-h), f(x+h) \Rightarrow f'(x)$:

$$p(t) = f(x-h)\cdot\frac{t-(x+h)}{(x-h)-(x+h)} + f(x+h)\frac{t-(x-h)}{(x+h)-(x-h)} \quad \text{(Lagrange)}$$

$$p'(x) = \frac{f(x+h)-f(x-h)}{2h} =: D_h^+(f)(x) \quad \text{(zentrale ...)}$$

d) $f(x-h), f(x), f(x+h) \Rightarrow f''(x)$: (Lagrange-Form)

$$p(t) = f(x-h)\cdot\frac{(t-x)(t-(x+h))}{(-h)\cdot(-2h)} + f(x)\frac{(t-(x-h))(t-(x+h))}{h\cdot(-h)} + f(x+h)\frac{(t-x)(t-(x-h))}{h\cdot(2h)}$$

$$\Rightarrow p''(x) = \frac{f(x+h)-2f(x)+f(x-h)}{h^2} =: D_h^2(f)(x)$$

(zentraler Quotient der 2. Ableitung)

Satz: 1) Sei $f \in C^2 \Rightarrow |D_h^+(f)(x)-f'(x)| = \mathcal{O}(h)$.
$|D_h^-(f)(x)-f'(x)| = \mathcal{O}(h)$.

2) Sei $f \in C^3 \Rightarrow |D_h(f)(x)-f'(x)| = \mathcal{O}(h^2)$.

3) Sei $f \in C^4 \Rightarrow |D_h^2(f)(x)-f''(x)| = \mathcal{O}(h^2)$.

Beweis: (von 2):

$$\exists \xi_1: f(x+h) = f(x) + hf'(x) + \frac{h^2}{2}f''(x) + \frac{h^3}{6}f'''(\xi_1)$$

$$\exists \xi_2: f(x-h) = f(x) - hf'(x) + \frac{h^2}{2}f''(x) - \frac{h^3}{6}f'''(\xi_2)$$

$$\frac{1}{2h}\left(f(x+h)-f(x-h)\right) = f'(x) + \frac{h^2}{12}\left(f'''(\xi_1)+f'''(\xi_2)\right)$$

$$\Rightarrow |D_h(f)(x)-f'(x)| \leq \frac{2}{12}\|f'''\|_\infty \cdot h^2 = \mathcal{O}(h^2)$$

② Numerische Integration

Zu bestimmen: $\int_a^b f(x)\,dx$

$x_i \in [a,b]$, $f(x_i)$ bekannt, $i=0\ldots N$

$P \in \mathbb{P}_n$ $\left(P(x_i) = f(x_i)\right)$, $i=0\ldots N$.

$$\Rightarrow \int_a^b f(x)\,dx \approx \int_a^b P(x)\,dx$$
$$= \int_{i=0}^{N} \int_a^b f(x_i) \prod_{\substack{j=0\\ j\neq i}}^{N} \frac{x-x_j}{x_i-x_j}\,dx \quad \text{(Lagrange-Form)}$$
$$= \sum_{i=0}^{N} \underbrace{\left(\int_a^b \prod_{\substack{j=0\\ j\neq i}}^{N} \frac{x-x_j}{x_i-x_j}\,dx \right)}_{=:A_i} f(x_i)$$
$$= \sum_{i=0}^{N} A_i\, f(x_i)$$

Newton-Cotes-Formeln: $x_i = a + ih$, $h = \frac{b-a}{N}$, $i=0\ldots N$.

N=1: Substitution $x = t + \frac{a+b}{2}$, $t = x - \frac{a+b}{2}$

$$A_0 = \int_a^b \frac{(x-a)}{(a-b)}\,dx = \int_{-\frac{b-a}{2}}^{\frac{b-a}{2}} \frac{t + \frac{b-a}{2}}{a-b}\,dt = \frac{1}{2}$$
$$A_1 = \int_a^b \frac{(x-b)}{(b-a)}\,dx = \int_{-\frac{b-a}{2}}^{\frac{b-a}{2}} \frac{t - \frac{b-a}{2}}{b-a}\,dt = \frac{1}{2}$$

Geometrisch:

$\Rightarrow$ Integral approximiert durch Flächen von Trapezen mit den Ecken $(a,0), (a,f(a)), (b,f(b)), (b,0)$

$$\Rightarrow \int_a^b f(x)\,dx \approx \frac{b-a}{2}\left(f(a)+f\left(\frac{a+b}{2}\right)+f(b)\right) =: T_1(f)$$

N=2: $x_0 = a$, $x_1 = \frac{a+b}{2}$, $x_2 = b$, $h = \frac{b-a}{2}$

Sei $f \in C^{4,\mathrm{var}}([a,b])$, $w(x) = \frac{(x-x_0)}{1}\frac{(x-x_1)}{2}\frac{(x-x_2)}{3}$

$$\Rightarrow \left| \int_a^b f(x)\,dx - T_2(f) \right| \leq C_0 \cdot h^4 \cdot \frac{f^{(4)}}{4!}\Big|_\infty$$

$$C_0 = \frac{1}{(4+1)!}\int_a^b |w(x)|\,dx$$
$$\leq \frac{h^{4+1}}{5!}\int_0^2 |(x-1)(x-2)|\,dx = \frac{h^{4+1}}{5!}\int_{-1}^{1} |(t+1)t(t-1)|\,dt = \frac{h^{4+1}}{5!}\cdot 2\int_0^1 t(1-t^2)\,dt = \frac{h^{4+1}}{5!}\cdot 2\left(\frac{1}{2}-\frac{1}{4}\right) = \frac{h^{4+1}}{4!}$$

$$\Rightarrow \left| \int_a^b f(x)\,dx - T_2(f) \right| \leq \frac{h^{4+1}}{4!}\cdot C_0 = \frac{h^{4+1}}{4!}\cdot\frac{1}{4!}\int_a^b |w(x)|\,dx \cdot \left| f^{(4)} \right|_\infty = \frac{h^5}{90}\cdot C \cdot \left| f^{(4)} \right|_\infty$$
$$= \mathcal{O}(h^5)$$

Beweis: Sei $p \in \mathbb{P}_3$, $p(x_i) = f(x_i)$.

$$\left| I_2(f) - \frac{b-a}{6}\left(f(a)+4f\left(\frac{a+b}{2}\right)+f(b)\right) \right| = \left| \int_a^b f(x)\,dx - \int_a^b p(x)\,dx \right|$$
$$= \left| \int_a^b \frac{f^{(4)}(\xi(x))}{(4+1)!} w(x)\,dx \right|$$
$$\leq \int_a^b \frac{|w(x)|}{(4+1)!}\,dx \cdot \left| f^{(4)} \right|_\infty \leq \frac{\sum_a^b |w(x)|}{(4+1)!} \cdot \left| f^{(4)} \right|_\infty = \frac{h^5}{90}\cdot C \cdot \left| f^{(4)} \right|_\infty$$

zu 2: $\int_0^2 w(x)\,dx = \int_0^2 x(x-1)(x-2)\,dx = 0$

$$\int_0^{|w(x)|} \left(\frac{w(x)}{2}\right)^{\max}(\xi(x))\,dx \leq \int_0^{|w(x)|} \left(\frac{w(x)}{2}\right)^{\max}(\xi(x))\,dx = \int_0^{|w(x)|} \left(\frac{w(x)}{2}\right)^{\max}(\mu(x))$$
$$\Rightarrow |T_2(f) - \frac{b-a}{6}\left(f(a)+4f\left(\frac{a+b}{2}\right)+f(b)\right)| \leq \left| f^{(4)} \right|_\infty \cdot \frac{h^5}{90} \cdot C_N$$

Genauer: Taylorreihe $f(x) = \sum_{i=0}^{3} \frac{h^i}{i!} f^{(i)}|_{x_0} + \frac{h^4}{4!} f^{(4)}|_\xi$

Simpson: $\left| \frac{h^4}{4!} \right| \cdot \left| f^{(4)} \right|_\infty$

Zusammengesetzte Formeln

Idee: Teile das Intervall $[a,b]$ auf in p Teilintervalle. Nutze dann Newton-Cotes auf Teilintervallen.

Beispiel: Simpson - 3 Teilintervalle.

$h = \frac{b-a}{3p}$

$$\int_a^b f(x)\,dx = \int_{x_0}^{x_2} f(x)\,dx + \int_{x_2}^{x_4} f(x)\,dx + \int_{x_4}^{x_6} f(x)\,dx + \ldots + \int_{x_{3p-2}}^{x_{3p}} f(x)\,dx$$
$$\sim \frac{h}{3}\left[f(x_0) + 4f(x_1) + f(x_2) \right] + \frac{h}{3}\left[f(x_2) + 4f(x_3) + f(x_4) \right] + \frac{h}{3}\left[f(x_4) + 4f(x_5) + f(x_6) \right] + \ldots + \frac{h}{3}\left[f(x_{3p-2}) + 4f(x_{3p-1}) + f(x_{3p}) \right]$$
$$\sim \frac{h}{3}\Big[f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + 2f(x_4) + 4f(x_5) + \ldots + f(x_{3p}) \Big]$$

Satz: Sei $f \in C^{4,\mathrm{var}}$. Das Integral $\int_a^b f(x)\,dx$ werde in p Teilintervalle (I_i, M_i) gemäß Intervallschachtelung der Newton-Cotes der Ordnung N unterteilt. $\tilde{T}_N(f)$.

Dann ist $\left| \int_a^b f(x)\,dx - \tilde{T}_N(f) \right| = \mathcal{O}(h^{4,\mathrm{var}})$, $h = \frac{b-a}{p\cdot N}$

N gerade $\Rightarrow$ $\left| \int_a^b f(x)\,dx - \tilde{T}_N(f) \right| = \mathcal{O}(h^{5,\mathrm{var}})$

Beweis: Auf jedem Teilintervall ist gilt $\left| \int_a^b f(x)\,dx - T_N(f) \right| \leq C \cdot h^{4,\mathrm{var}}$.

Also $\left| \int_a^b f(x)\,dx - \tilde{T}_N(f) \right| = \left| \sum_{i=1}^{p} \left(\int_a^b f(x)\,dx - T_N(f) \right) \right|$

$$\leq p \cdot C \cdot h^{4,\mathrm{var}} = \frac{p \cdot (b-a)}{N\cdot p} \cdot C \cdot h^{4,\mathrm{var}}$$
$$= \mathcal{O}(h^{4,\mathrm{var}})$$

Teil 2 entsprechend.

Folge: Manchmal ist es viel schneller eine ungerade Anzahl an Stützstellen zu nehmen.

(3) Richardson-Extrapolation

Zu berechnen sei $\lim_{h \to 0} F(h) =: F(0)$

Bekannt sei $F(h_k)$, $k = 0 .. N$.

$\Rightarrow p \in \mathcal{P}_N$, $p(h_k) = F(h_k)$. $F(0) \sim p(0)$.

Beispiel (Romberg-Verfahren):

Zu berechnen sei $\int_a^b f(x)\,dx$, $f \in C^\infty$.

zus. gesetzte Trapezregel, Schrittweite h: $F(h)$.
" $\frac{h}{2}$: $F(\frac{h}{2})$.

$$\int_a^b f(x)\,dx = \lim_{h \to 0} F(h).$$

$$\left.\begin{array}{l} p(h) = F(h) \\ p(\frac{h}{2}) = F(\frac{h}{2}) \end{array}\right\} \to p(x) = F(h) \cdot \frac{x - \frac{h}{2}}{h - \frac{h}{2}} + F(\frac{h}{2}) \frac{x - h}{\frac{h}{2} - h} \quad \text{(Lagrange)}$$

$$p(0) = -F(h) + 2F(\tfrac{h}{2})$$

↑ Richardson-Extrapolation

$$\left.\begin{array}{l} F(h) = F(0) + hF'(0) + \frac{h^2}{2} F''(\xi_1) \\ F(\frac{h}{2}) = F(0) + \frac{h}{2} F'(0) + \frac{h^2}{8} F''(\xi_2) \end{array}\right\} \to p(0) = F(0) + h^2 \left(\frac{F''(\xi_2)}{4} - \frac{F''(\xi_1)}{2} \right)$$

$$= F(0) + \mathcal{O}(h^2)$$

Ordnung h^2 ∇ 0

Gewöhnliche Differentialgleichungen

Aus der Vorbemerkungen:

→ Anfangswertproblem: Bestimme $y : [a,b] \to G \subset \mathbb{R}^n$:

$$y'(t) = f(t, y(t)), \quad y(a) = y_0$$

$f : [a,b] \times G \to \mathbb{R}^n$, $G \subset \mathbb{R}^n$ offen, zshgd,
$y_0 \in G$.

→ Im skalaren Beweisen: $n = 1$.

→ Differentialgleichungen mit höheren Ableitungen können in Systeme umgewandelt werden.

→ Äquivalente Form: $y(s) - y(a) = \int_a^s y'(t)\,dt = \int_a^s f(t, y(t))\,dt$

$$\Rightarrow y(s) = y_0 + \int_a^s f(t, y(t))\,dt$$

Integraldarstellung der Differentialgleichung

Beispiele:

1) Lineare DGL: $y'(t) = b\,y(t)$, $y(0) = y_0$:

$y(t) = y_0 e^{bt}$ ist Lösung

2) $y'(t) = b(t) \cdot y(t)$, $y(0) = y_0$:

$$b(t) = \frac{y'(t)}{y(t)} = \log(y(t))' \Rightarrow \int_a^s b(t)\,dt = \log\left(\frac{y(s)}{y_0}\right)$$

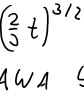
$$\Rightarrow y(s) = e^{\int_a^s b(t)\,dt} \cdot y_0$$

3) $y'(t) = b(t)\,y(t) + c(t)$, $y(0) = y_0$

→ Variation der Konstanten

$$y(s) = C(s)\, e^{\int_a^s b(t)\,dt}$$

4) $y'(t) = 1 + y(t)^2$: $y(t) = \tan t$ ist Lösung



keine globale Lösung!

5) $y'(t) = y(t)^{1/3}$, $y(0) = 0$:

$y = 0$ ist Lösung

$y(t) = \left(\frac{2}{3}t\right)^{3/2}$: $y'(t) = \left(\frac{2}{3}\right)^{3/2} \cdot \frac{3}{2} \cdot t^{1/2} = \left(\frac{2}{3}t\right)^{1/2} = y(t)^{1/3}$

→ Die AWA besitzt zwei Lösungen!

Satz von Picard-Lindelöf

Gegeben: $X \subset C^0([a,b], \mathbb{R}^n)$ mit $\|\cdot\|_\infty$

X ist vollständig (Aussage II).

$g: X \to X,\ (g(y))(s) := y_0 + \int_a^s f(t, y(t))\,dt$.

$\overline{y} \in X$ ist Lösung des AWPs $\Leftrightarrow \overline{y} = g(\overline{y})$

($\Leftrightarrow \overline{y}$ ist Fixpunkt von g).

Beweis: Integralcharakterisierung des AWPs.

Idee: Zeige die Voraussetzungen des Fixpunktsatzes von Banach 8

Def: $f: D \subset X$ heißt Lipschitzstetig mit Lipschitzkonstante L

$:\Leftrightarrow \|f(x) - f(y)\| \le L \|x - y\| \quad \forall x, y \in D$.

Bemerkung: 1) $C^1 \Rightarrow f$ Lokal Lipschitz.

2) f ist stetig.

3) D kompakt, f stetig diffbar

$\Rightarrow f$ Lipschitzstetig mit $L = \|f'\|_\infty$.

Beweis: ex-9-3: wie bei Taylor-Lemma.

4) $f: D \to \mathbb{R}^n,\ f(y) := Ay^2$ Lipschitzstetig

on D beschränkt.

Picard-Lindelöf: Sei f stetig und Lipschitzstetig in der 2. Variablen, also

$\|f(t, y_1) - f(t, y_2)\| \le L \|y_1 - y_2\| \quad \forall t \in [a,b],$
$y_1, y_2 \in G$.

Dann ex. $\varepsilon > 0$: Die AWA besitzt genau
einen stetig. Lösung auf dem Intervall $I_\varepsilon = [a, a+\varepsilon]$.

Beweis: Gegeben $y_0 \in G \Rightarrow$ Abgeschlossene Kugel B
um y_0 mit Radius δ, die ganz in G liegt.

$I \times B$ ist abgeschlossen + beschränkt, also kpt, f stetig

$\Rightarrow M := \sup_{\substack{t \in I \\ y \in B}} \|f(t,y)\| < \infty$

Sei $0 < \varepsilon \le a$ und $\varepsilon < \min\left(b-a, \frac{\delta}{M}\right)$.

Dann ist $I_\varepsilon \subset [a,b]$.

Sei nun g wie in der Vorbemerkung, also

$X := C^0(I_\varepsilon, \mathbb{R}^n)$ mit $\|\cdot\|_\infty$

$g: X \to X,\ (g(y))(s) := y_0 + \int_a^s f(t, y(t))\,dt,$
$s \in I_\varepsilon$.

1) X ist vollständig

2) $D := C^0(I_\varepsilon, B)$.

Sei y_n Folge in D, $y_n \to y \in X$.

$\|y_n(s) - y(s)\| \le \|y_n - y\|_\infty \to 0$

$\Rightarrow y_n(s) \to y(s) \Rightarrow y(s) \in B \Rightarrow y \in D$.

$\in B$, klar. $\Rightarrow$ Abgeschlossen

3) Sei $y \in D$, $s \in I_\varepsilon$.

$\|(g(y))(s) - y_0\| = \left\| \int_a^s f(t, y(t))\,dt \right\|$
$\le \int_a^s \|f(t, y(t))\|\,dt$
$\le M \cdot (s-a) < M \cdot \varepsilon \le \delta$

$\Rightarrow \big(g(y)\big)(s) \in B \Rightarrow g(y) \in D$

$\Rightarrow g: D \to D$.

4) Seien $u, v \in D$.

$\|g(u) - g(v)\|_\infty = \sup_{s \in I_\varepsilon} \|(g(u))(s) - (g(v))(s)\|$

$= \sup_{s \in I_\varepsilon} \left\| \int_a^s f(t, u(t))\,dt - \int_a^s f(t, v(t))\,dt \right\|$

$\le \sup_{s \in I_\varepsilon} \int_a^s \|f(t, u(t)) - f(t, v(t))\|\,dt$

$\le \int_a^{a+\varepsilon} L \|u(t) - v(t)\|\,dt$

$\le \int_a^{a+\varepsilon} L \cdot \|u - v\|_\infty\,dt$

$\le L \cdot \varepsilon \cdot \|u - v\|_\infty \le q \cdot \|u - v\|_\infty$

$\Rightarrow g$ ist kontrahierend. $\square$

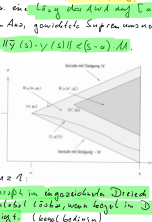
Globale Satz von Picard-Lindelöf:

Sei alles wie oben und $M \cdot (b-a) \le \delta$

Dann ex. ein ε-los. eine AWA auf $[a,b]$. ($\varepsilon \in \frac{\gamma}{L}$ vernünftig)

Beweis: Nun Anz. gewichtete Supremumsnorm.

Bemerkung: $\|g'(s) - y'(s)\| \le (s-a) \cdot M$.



$\Rightarrow$ Graph im angezeichneten Dreieck (keil)

$\Rightarrow$ Globale Lösung muss liegen im Definitionsgebiet von f
Lipschitz (Lange Bedingung)

$\Rightarrow$ AWA bei (a', y') auch lösbar.

$\Rightarrow$ Lösungen können in ganz Umgebung von y_0.

Ab jetzt: $[a,b]$ so gewählt, dass globale
Lösung existiert.

Lemma von Gronwall

Sei $I = [a,b]$.

Seien $\alpha, \beta, u : I \to \mathbb{R}$.

1) $u \in C^1$, $u'(t) \le \alpha(t) + \beta(t) u(t)$

$\Rightarrow u(t) \le u(a) e^{\int_a^t \beta(s)\,ds} + \int_a^t \alpha(s) e^{\int_s^t \beta(\zeta)\,d\zeta}\,ds \quad \forall t \in I.$

rechte Seite: Lösg der inhomogenen Lineare AWA
$u'(t) = \alpha(t) + \beta(t) u(t)$, $u(a) = y_0$. (Übung)

2) $\beta \ge 0$, $u(t) \le \alpha(t) + \int_a^t \beta(s) u(s)\,ds$

$\Rightarrow u(t) \le \alpha(t) + \int_a^t \alpha(s)\beta(s) e^{\int_s^t \beta(\zeta)\,d\zeta}\,ds \quad \forall t \in I.$

Beweis:

① Seien $\varphi, \psi \in C^1([a,b])$, $\varphi(a) \le \psi(a)$, $\varphi'(t) \le \psi'(t)$ $\forall t \in [a,b]$.
$(\varphi - \psi)'(t) \le 0 \Rightarrow \varphi - \psi$ monoton fallend
$(\varphi - \psi)(a) \le 0 \Rightarrow \varphi(t) \le \psi(t)$ $\forall t \in [a,b]$.

② Sei $\varphi \in C^1[a,b]$, $\psi \in C^0[a,b]$, $\varphi'(t) \le \psi(t)$ $\forall t \in [a,b]$.
$w(t) := \varphi(a) + \int_a^t \psi(s)\,ds \Rightarrow \varphi'(t) \le \psi(t) = w'(t)$, $\varphi(a) = w(a)$
$\Rightarrow \varphi(t) \le \varphi(a) + \int_a^t \psi(s)\,ds$

③ $v(t) = e^{-\int_a^t \beta(\zeta)\,d\zeta} \Rightarrow v'(t) = -\beta(t) v(t)$, $v(a) = 1$.

$\Rightarrow (uv)' = u'v + v'u$
$= u'v - \beta vu$
$\le \alpha v + \beta uv - \beta uv$ (v positiv)
$= \alpha v$

Einsetzen in ②

$\Rightarrow u(t) v(t) \le u(a)v(a) + \int_a^t \alpha(s) v(s)\,ds$

$\Rightarrow u(t) \le u(a) e^{\int_a^t \beta(s)\,ds} + \int_a^t \alpha(s) \underbrace{e^{\int_a^s \beta(s)\,ds} \cdot e^{\int_a^t \beta(\zeta)\,d\zeta}}_{e^{\int_s^t \beta(\zeta)\,d\zeta}}\,ds$

④ $v(t) = \int_a^t \beta(s) u(s)\,ds \Rightarrow u(t) \le \alpha(t) + v(t)$

$v'(t) = \beta(t) u(t) \le \beta(t)\alpha(t) + \beta(t) v(t)$ ($\beta \ge 0$)
$v(a) = 0$
$\Rightarrow v(t) \le \int_a^t \alpha(s)\beta(s) e^{\int_s^t \beta(\zeta)\,d\zeta}\,ds$
$\Rightarrow u(t) \le \alpha(t) + v(t) = \alpha(t) + \ldots$

β konstant:

$u(t) \le \alpha(t) + \beta \int_a^t u(s)\,ds$

$\Rightarrow u(t) \le \alpha(t) + \beta \int_a^t \alpha(s) e^{\beta(t-s)}\,ds$

Stabilität für AWA

AWA: $y'(t) = f(t, y(t)),\ y(a) = y_0$, f Lipschitzstetig im 2. Argument.

$f \to \tilde{f}$, $y_0 \to \tilde{y}_0$, $\|f - \tilde{f}\|_\infty \le \varepsilon$, $\|y_0 - \tilde{y}_0\| \le \tilde{\varepsilon}$.

$\widetilde{AWA}$: $\tilde{y}'(t) = \tilde{f}(t, \tilde{y}(t)),\ \tilde{y}(a) = \tilde{y}_0$.

$y, \tilde{y}$ existieren auf $[a,b]$.

$\Rightarrow \|\tilde{y}(t) - y(t)\| \le \left(\tilde{\varepsilon} + \varepsilon(t-a)\right) e^{L(t-a)} \quad \forall t \in [a,b]$.

Beweis: Gronwall.

$u(t) := \|\tilde{y}(t) - y(t)\|$.

$$u(t) = \|\tilde{y}_0 + \int_a^t \tilde{f}(s, \tilde{y}(s))\,ds - y_0 - \int_a^t f(s, y(s))\,ds\|$$

$$\le \|\tilde{y}_0 - y_0\| + \int_a^t \|\tilde{f}(s, \tilde{y}(s)) - f(s, \tilde{y}(s)) + f(s, \tilde{y}(s)) - f(s, y(s))\|\,ds$$

$$\le \tilde{\varepsilon} + \int_a^t \|\tilde{f}(s, \tilde{y}(s)) - f(s, \tilde{y}(s))\|\,ds + \int_a^t \|f(s, \tilde{y}(s)) - f(s, y(s))\|\,ds$$

$$\le \underbrace{\tilde{\varepsilon} + \varepsilon(t-a)}_{\alpha(t)} + \int_a^t \underbrace{L}_{\beta} \underbrace{\|\tilde{y}(s) - y(s)\|}_{u(s)}\,ds$$

Gronwall $\Rightarrow$

$$u(t) \le \alpha(t) + L \int_a^t \underbrace{\alpha(s)}_{\le \alpha(t)} e^{L(t-s)}\,ds$$

$$\le \alpha(t)\left(1 + \left(-e^{L(t-s)}\Big|_a^t\right)\right)$$

$$= \left(\tilde{\varepsilon} + \varepsilon(t-a)\right) e^{L(t-a)}$$

$$\Rightarrow \|y - \tilde{y}\|_\infty \le \tilde{\varepsilon} e^{L(b-a)} + \varepsilon(b-a) e^{L(b-a)}$$

Diskrete Lösung von Anfangswertaufgaben

Im Folgenden:

Gesucht sei die Lösung $y:[a,b] \to \mathbb{R}^n$

$y'(t) = f(t, y(t)),\ y(a) = y_0$

- f stetig
- Lipschitzbedingung bzgl. 2. Argument mit konst. L
- Kegelbedingung erfüllt
- Picard-Lindelöf: Die AWA ist lösbar auf $[a,b]$.
  $(t, y(t)) \in K_H((a, y_0)) \quad \forall t \in [a,b]$.
- Jede AWA mit $y(a) = y_0'$ mit $(a', y_0') \in K_H(a, y_0)$ ist lösbar auf $[a', b]$.

$\Rightarrow$ Um (wahrscheinlich) Lösbarkeit müssen wir uns nicht kümmern.

Idee des Verfahrens:

→ Berechne nicht die Funktion y sondern Approximationen $y_h(t) \approx y(t)$ für $t \in I_h$.
I_h ist eine endliche Teilmenge von $[a,b]$.

Formal: Sei $I_h = \{t_0 = a, t_1 \dots t_{m-1}, t_m = b\}$
mit $t_0 < t_1 < t_2 < \dots < t_m$.

I_h heißt Gitter, $h = \max(t_{i+1} - t_i)$
heißt Feinheit des Gitters.

$y_h : I_h \to \mathbb{R}^n$ (Man sagt dem Gitter definiert 8)

Wir setzen $y_h = y_h(t_i)$.

Für $n=1$ setzen wir auch gern $y_h = \begin{pmatrix} y_0 \\ \vdots \\ y_N \end{pmatrix}$.

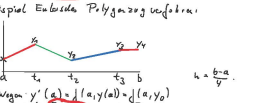
$a = t_0 \quad t_1 \quad \dots \quad t_{m-1} \quad t_m = b$
$y_0 \quad y_1 \quad \dots \quad y_{m-1} \quad y_m$

Typisch: t_i äquidistant, $t_i = \frac{b-a}{N}(i)$ $\quad t_{i+1} = a + i \cdot h$

Numerische Verfahren:

→ Berechnet zu einem Gitter I_h
eine Näherung y_h an die Lösung,
d.h. $y_h = y_h(t_i) \approx y(t_i)$.

Beispiel Eulersche Polygonzugverfahren:



$h = \frac{b-a}{N}$.

Wegen $y'(a) = f(a, y(a)) = f(a, y_0)$
ist die Lösungskurve an y im Punkt (a, y_0) bekannt. Folge der Tangente
bis zum Punkt (t_1, y_1), also $y_1 = y_0 + h \cdot f(a, y_0)$.
(Bemerke: Wegen $|f(a,y(a))| \le M$
liegt (t_1, y_1) im Kegel $K_H((a, y(a)))$.
y_1 ist Näherung an $y(t_1)$.

Für y_2: Folge der Tangente wieder unter der Annahme,
dass $y(t_1) = y_1$ bis zum Punkt (t_1, y_1), also $y_2 = y_1 + h \cdot f(t_1, y_1)$.
$f(t_1, y_1)$ ist wohl definiert → Kegelbedingung.

Allgemein: $y_{k+1} = y_k + h \cdot f(t_k, y_k)$.

Für y_0 entsteht das Problem dadurch, dass y keine Gerade sein muss.

Für y_2 entsteht ein Fehler dadurch, dass y keine Gerade sein muss, und dadurch, dass y_1 sowieso falsch war, $y_1 \neq y(t_1)$.

Die zentrale Idee in der Analysis von Verfahren wird sein, diese beiden Fehler zu trennen.

Alternative Herleitungen für Euler:

Taylor:
$y(t+h) \sim y(t) + y'(t) \cdot h$
$\sim y(t) + f(t, y(t)) \cdot h$
$\Rightarrow y_{k+1} = y_k(t_{k+1}) \sim y(t_k + h) \sim y(t_k) + h \cdot f(t_k, y(t_k))$
$\sim y_k + h \cdot f(t_k, y_k)$

Numerische Differentiation:

$y'(t_k) = f(t_k, y(t_k))$

$\rightsquigarrow \frac{y(t_{k+1}) - y(t_k)}{h} \sim f(t_k, y(t_k))$

→ siehe oben.

Numerische Integration:

$y(t_{k+1}) = y(t_k) + \int_{t_k}^{t_{k+1}} f(t, y(t))\, dt$

Stützpunkt $= t_k$.

$\sim h \cdot f(t_k, y(t_k))$
$\sim h \cdot f(t_k, y_k)$

konvergenz

Zu lösen sei $y'(t) = f(t, y(t)), y(a) = y_0$.
Die Lösung sei $\bar{y}(t)$.

Gegeben sei ein numerisches Verfahren, das für ein Gitter I_h eine Näherung y_h an $\bar{y}$ berechnet.

Def:

$$e_h = \| y_h - \bar{y}|_{I_h} \|_\infty = \max_{t \in I_h} |y_h(t) - \bar{y}(t)|$$

$$= \max_k |y_k - \bar{y}(t_k)|$$

heißt globaler Diskretisierungsfehler.

Achtung: $\|y_h - \bar{y}\|_\infty$ ist sinnlos.

Def: Sei I_{h_k} eine Folge von Gittern mit Feinheit h_k mit $h_k \to 0$.
Falls $e_{h_k} \to 0$, so heißt das Verfahren konvergent.

Achtung: $y_h \to \bar{y}$ ist sinnlos.

Def Ein Verfahren heißt konvergent von der Ordnung p, falls

$$e_h = \mathcal{O}(h^p).$$

konvergenz ist schwer zu beweisen → konsistenz.

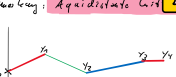
Numerisches Verfahren und Konsistenz

Euler: $y_{k+1} = y_k + h\, f(t_k, y_k)$

Allgemein: $y_{k+1} = y_k + h\cdot\varphi$

φ heißt Verfahrensfunktion, in φ kommen vor: $t_k, I_h, y_k, f, \dots$

Bemerkung: Äquidistante Gitter



konkret:

$y_{k+1} = y_k + h\cdot\varphi(t_k, y_k, h)$
→ Explizite Einschrittverfahren

$y_{k+1} = y_k + h\cdot\varphi(t_k, y_k, y_{k+1}, h)$
→ Implizite Einschrittverfahren

$y_{k+1} = y_k + h\cdot\varphi(t_k, t_{k-1}, y_k, y_{k-1}, h)$
→ Explizite Mehrschrittverfahren

$y_{k+1} = y_k + h\cdot\varphi(t_k, t_{k-1}, y_k, y_{k-1}, y_{k+1}, h)$
→ Implizite Mehrschrittverfahren

Wir beschränken uns zunächst auf die expliziten Einschrittverfahren.

$y_{k+1} = y_k + h\,\varphi(t_k, y_k, h)$

$y(t_{k+1})$ $y(t_k)$

$$\Rightarrow \frac{y(t_k+h) - y(t_k)}{h} \sim \varphi(t_k, y(t_k), h)$$

Differenz: Konsistenzfunktion.

Def: Sei y eine Lösung der Differentialgleichung $y \in K_M$.

$$\tau_h(t, y(t)) = \frac{y(t+h) - y(t)}{h} - \varphi(t, y(t), h)$$

heißt Konsistenzfehler des Verfahrens an der Stelle $(t, y(t))$.

Def. Ein numerisches Verfahren heißt konsistent, falls

$$\sup_{t,y} |\tau_h(t, y(t))| = \|\tau_h\|_\infty \to 0$$

Ein Verfahren heißt konsistent von der Ordnung p, falls

$\|\tau_h\|_\infty = \mathcal{O}(h^p)$.

Lemma: Sei f stetig differenzierbar.
Dann gibt es eine Konstante C, so dass

$\|y''\|_\infty \le C$ $\forall$ Lösungen y der Differentialgleichung, die in K_M liegen.

Beweis: $y'(t) = f(t, y(t))$

$$y''(t) = d_t f(t, y(t)) + d_y f(t, y(t))\, f(t, y(t))$$

$$\Rightarrow \|y''\|_\infty \le \|d_t f\|_\infty + \|d_y f\|_\infty \cdot \|f\|_\infty$$ (alles auf dem kompakten K_M)

Korollar: Sei f r-mal diffbar. Dann ex. C:

$\|y^{(r+1)}\|_\infty \le C$ $\forall$ Lösungen der DGL in K_M.

Beispiele für explizite Einschrittverfahren

① Explizites Polygonzugverfahren

$y_{k+1} = y_k + h \underbrace{f(t_k, y_k)}_{=: \varphi(t_k, y_k, h)}$

Sei $y(\cdot)$ eine Lösung des DGL in K_M.

$\tau_h(t, y(t)) = \frac{y(t+h) - y(t)}{h} - \varphi(t, y(t), h)$

$\overset{(Taylor)}{=} \frac{y(t) + h \cdot y'(t) + \frac{h^2}{2} y''(\xi) - y(t)}{h} - f(t, y(t))$

$= f(t, y(t)) + \frac{h}{2} y''(\xi) - f(t, y(t))$

$|\tau_h(t, y(t))| \le \frac{h}{2} |y''(\xi)| \le \frac{h}{2} \|y''\|_\infty \le \frac{c}{2} \cdot h = \mathcal{O}(h)$

$\Rightarrow$ Eulo ist konsistent von der Ordnung 1, falls $f \in C^1$.

Strategie: φ bestimmen, τ_h umschreiben, alles mit Taylor entwickeln.

Erinnerung:

$f(t+h_1, y+h_2) = f(t,y) + h_1 \partial_t f(t,y) + h_2 \partial_y f(t,y) + \mathcal{O}(h_1^2 + h_1 h_2 + h_2^2)$

② Verbessertes Eulo-Verfahren

Idee: $y(t_{k+1}) - y(t_k) \approx \int_{t_k}^{t_{k+1}} f(t, y(t))\, dt$

Stützstelle $t_k + \frac{h}{2}$:

$\sim h \cdot f\left(t_k + \frac{h}{2}, y\left(t_k + \frac{h}{2}\right)\right)$

$\sim y(t_k) + \frac{h}{2} f(t_k, y(t_k))$

$\Rightarrow y_{k+1} = y_k + h \underbrace{f\left(t_k + \frac{h}{2}, y_k + \frac{h}{2} f(t_k, y_k)\right)}_{=: \varphi(t_k, y_k, h)}$

Um y_{k+1} abzurechnen, müssen wir f 2x auswerten.

$\tau_h(t, y(t)) = \frac{y(t+h) - y(t)}{h} - \varphi(t, y(t), h)$

$= \frac{y(t) + h \cdot y'(t) + \frac{h^2}{2} y''(t) + \frac{h^3}{6} y'''(\xi) - y(t)}{h}$

$- f\left(t + \frac{h}{2}, y(t) + \frac{h}{2} f(t, y(t))\right)$

$= y'(t) + \frac{h}{2} y''(t) + \frac{h^2}{6} y'''(\xi)$

$- \Big(f(t, y(t)) + \frac{h}{2} \partial_t f(t, y(t)) + \frac{1}{2} \partial_y f(t, y(t)) \cdot \underbrace{y'(t)}_{f(t, y(t))} + \mathcal{O}(h^2) \Big)$

Nach dem Lemma: $y''(t) = \underbrace{\left(\partial_t + f \cdot \partial_y\right)}_{=: L} f(t, y(t))$

$= \mathcal{O}(h^2)$

$\Rightarrow$ Falls $f \in C^1$, so ist das verbesserte Eulo-Verfahren konsistent von der Ordnung 2.

③ Verfahren von Heun:

Idee: $y(t_{k+1}) - y(t_k) \approx \int_{t_k}^{t_k + h} f(t, y(t))\, dt$

Trapezregel: $\sim \frac{h}{2} \left(f(t_k, y(t_k)) + f(t_k + h, y(t_k + h)) \right)$

$\sim y(t_k) + h \cdot y'(t_k)$

$y_{k+1} = y_k + h \cdot \frac{1}{2} \Big(f(t_k, y_k) + f\big(t_k + h, \underbrace{y_k + h \cdot f(t_k, y_k)}_{y(t_k + h)}\big) \Big)$

$\varphi(t_k, y_k, h)$

$\tau_h(t, y(t)) = \frac{y(t+h) - y(t)}{h} - \varphi(t, y(t), h)$

$= \frac{y(t) + h \cdot y'(t) + \frac{h^2}{2} y''(t) + \frac{h^3}{6} y'''(\xi) \cdot y(t)}{h}$

$- \frac{1}{2} \Big(f(t, y(t)) + f\big(t + h, y(t) + h f(t, y(t))\big) \Big)$

$= f(t, y(t)) + \frac{h}{2} \left(\partial_t + f \partial_y \right)(t, y(t)) + \frac{h^2}{6} y'''(\xi)$

$- \frac{1}{2} \Big(f(t, y(t)) + f(t, y(t)) + h \partial_t f(t, y(t)) + h f(t, y(t)) \cdot \partial_y f(t, y(t)) + \mathcal{O}(h^2) \Big)$

$= \mathcal{O}(h^2)$

$f \in C^2 \Rightarrow$ Heun ist von zweiter Ordnung.

2 Auswertungen von f.

Aus Konsistenz folgt Konvergenz (expl. ESV) (äquidist. Gitter)

$y_{k+1} = y_k + h\,\varphi(t_k, y_k, h)$

konvergenz: $\bar{y}(t_k) - y_k$

konsistenz: $\frac{1}{h}\left(y(t_{k+1}) - \left(y(t_k) + h\,\varphi(t_k, y(t_k), h)\right)\right)$

$\bar{y}$ Lösung der AWA, y Lösung der DGl.

Diskretes Lemma von Gronwall:

$(a_k), (\beta_k), (e_k)$ seien ≥ 0.

$e_{k+1} \leq \alpha_k + (1+\beta_k) e_k$

$\Rightarrow e_k \leq \left(e_0 + \sum_{i=0}^{k-1} \alpha_i\right) e^{\sum_{i=0}^{k-1} \beta_i}$

Beweis: Übungen.

Es sei e_k der globale Fehler an der Stelle t_k, also $e_k = |\bar{y}(t_k) - y_k| \geq 0$. Verfahren konsistent.

$$
\begin{aligned}
e_{k+1} &= |\bar{y}(t_{k+1}) - y_{k+1}| \\
&= |\bar{y}(t_{k+1}) - (y_k + h\,\varphi(t_k, y_k, h))| \\
&= |\bar{y}(t_{k+1}) - (\bar{y}(t_k) + h\,\varphi(t_k, \bar{y}(t_k), h)) + (\bar{y}(t_k) - y_k) \\
&\quad + h\left(\varphi(t_k, \bar{y}(t_k), h) - \varphi(t_k, y_k, h)\right)| \\
&\leq h\,|\tau_k(t_k, y(t_k))| + e_k + hL' \underbrace{|\bar{y}(t_k) - y_k|}_{e_k}
\end{aligned}
$$

L' Lipschitz konstante von φ (α_k); β_k

Bemerkung: f Lipschitz $\Rightarrow \varphi$ Lipschitz

$$
\begin{aligned}
&= \alpha_k + (1+\beta_k) e_k \\
\Rightarrow e_k &\leq \left(e_0 + \sum_{i=0}^{k-1} h\,|\tau_h(t_i, y(t_i))|\right) e^{L' \sum_{i=0}^{k-1} h} \\
&\leq \left(e_0 + \|\tau_h\|_\infty \cdot k \cdot h\right) e^{L' \cdot k \cdot h} \\
&\leq \left(e_0 + \|\tau_h\|_\infty \cdot (b-a)\right) e^{L'(b-a)}
\end{aligned}
$$

$e_0 = 0, \|\tau_h\|_\infty \to 0 \Rightarrow e_k \to 0$ gleichmäßig:

$\|e\|_\infty \leq \|\tau_h\|_\infty \cdot (b-a)\, e^{L'(b-a)}$

Satz: Sei φ lipschitz, Verfahren konsistent von der Ordnung p. Dann ist es konvergent von der Ordnung p.

Implizite Einschrittverfahren

$$y_{k+1} = y_k + h\,\varphi(t_k, y_k, y_{k+1}, h)$$

Beispiel:

$$y(t_{k+1}) = y(t_k) + \int_{t_k}^{t_{k+1}} f(t, y(t))\,dt$$

$$\sim y(t_k) + h \cdot f(t_{k+1}, y(t_{k+1}))$$

$$\Rightarrow y_{k+1} = y_k + h \cdot f(t_{k+1}, y_{k+1}) \quad \text{[Implizite Euler]}$$

Geht das überhaupt???

Fixpunktsatz von Banach:

$g(z) := y_k + h \cdot \varphi(t_k, y_k, z, h)$, φ stetig, Lipschitz-stetig in y_{k+1}

$(t_k, y_k), (t_k, z) \in K_M$.

$\Rightarrow \|\varphi\|_\infty =: M' < \infty$.

Sei δ so klein, dass eine δ-Umgebung von y_k im Def.-Gebiet von φ liegt.

Sei $h \le \frac{\delta}{2M'} \Rightarrow |g(z) - y_k| \le \frac{1}{2} \cdot \delta$.

$g: [y_k - \delta, y_k + \delta] \to [y_k - \delta, y_k + \delta]$

Banachraum ✓. Abgeschlossen ✓.

$$|g(z) - g(z')| = |\varphi(t_k, y_k, z, h) - \varphi(t_k, y_k, z', h)|$$

$$\le h \cdot L' \cdot |z - z'|$$

$h \le \frac{1}{2L'} \Rightarrow g$ kontraktierend

$\Rightarrow g$ besitzt Fixpunkt

$\Rightarrow$ Verfahren durchführbar.

1) Falls nicht analytisch auflösbar:
   Fixpunktfolge.

2) $z = v(t_k, y_k)$.

   $y_{k+1} = y_k + h\,\varphi(t_k, y_k, v(t_k, y_k), h)$

   $\hat{=}$ explizites ESV $\triangle_0$

$\Rightarrow$ Aus Konsistenz folgt Konvergenz.

Beispiele für implizite Verfahren

① Implizite Euler

$$y_{k+1} = y_k + h f(t_{k+1}, y_{k+1})$$

Konsistenz: $\frac{y(t+h)-y(t)}{h} - \varphi(t, y(t), y(t+h), h)$

$= y'(t) + \mathcal{O}(h) - f(t+h, y(t+h))$

$\hookrightarrow f(t, y(t)) + \mathcal{O}(h)$

$= \mathcal{O}(h)$

$\Rightarrow$ konsistent von Ordnung 1

$\Rightarrow$ konvergent von Ordnung 1

Mit Fixpunktfolge:

$g(z) := y_k + h f(t_{k+1}, z)$, $z_0 = y_k$, $z_{k+1} = g(z_k)$.

$y_{k+1} = z_2 = g(z_1) = g(g(z_0))$

$= g(y_k + h f(t_{k+1}, y_k))$

$= y_k + h f(t_{k+1}, y_k + h f(t_{k+1}, y_k))$

2 Auswertungen, trotzdem Ordnung 1 !!

② Implizite Trapezregel:

$$y(t_{k+1}) = y(t_k) + \int_{t_k}^{t_{k+1}} f(t, y(t))\, dt$$

$$\sim h \cdot \frac{f(t_k, y(t_k)) + f(t_{k+1}, y(t_{k+2}))}{2}$$

$$\Rightarrow y_{k+1} = y_k + h \frac{f(t_k, y_k) + f(t_{k+1}, y_{k+1})}{2}$$

Ordnung: $\frac{y(t+h)-y(t)}{h} - \frac{1}{2}\left[f(t, y(t)) + f(t+h, y(t+h))\right]$

$f(t+h, y(t+h)) = f\left(t+h, y(t) + h \cdot y'(t) + \frac{h^2}{2} y''(\xi)\right)$

$= f(t, y(t)) + h \cdot f_t(t, y(t)) + h \cdot f(t, y(t)) \cdot f_y(t, y(t)) + \mathcal{O}(h^2)$

$\Rightarrow$ 2. Ordnung.

Implizite Verfahren

June 16, 2020

1 Implizite Einschrittverfahren

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import math
```

Definition der linearen Beispielgleichung (wie bei explizit)

[2]:
```
def f_linear(t,y):
    return y
a_linear=0
b_linear=1
y0_linear=1
def loesung_linear(t):
    return np.exp(t)
```

[3]:
```
f=f_linear
a=a_linear
b=b_linear
y0=y0_linear
loesungsfunc=loesung_linear
```

[4]:
```
N_plot=128
I_plot=np.linspace(0,1,N_plot)
```

Definition der Verahren: Impliziter Euler, Trapezregel

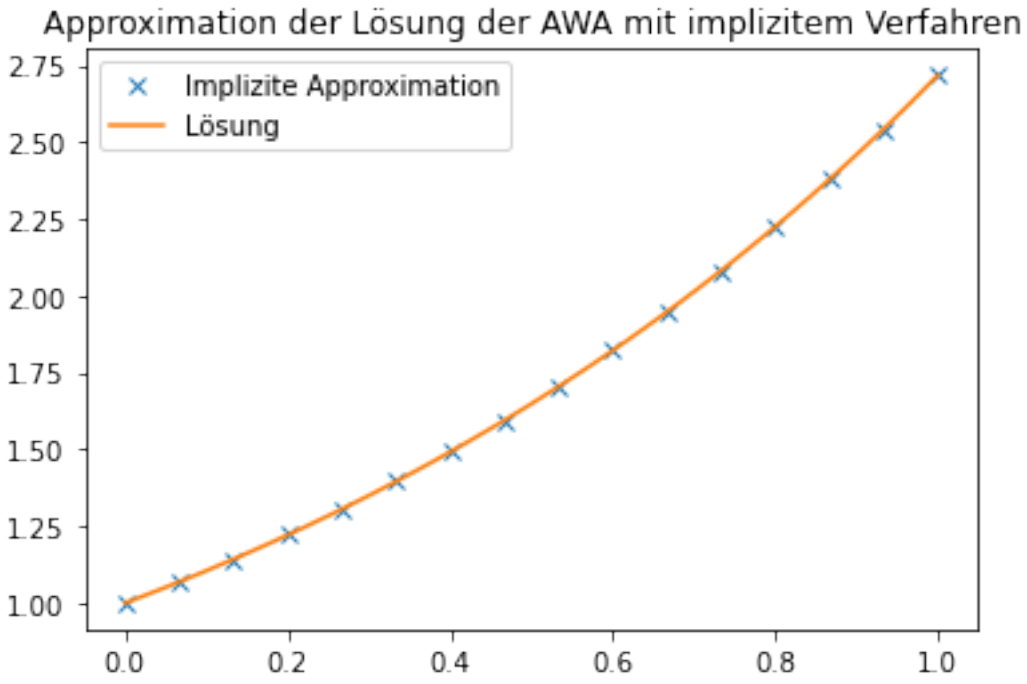
[5]:
```
def phi_impliziter_euler(f,t_k,y_k,y_k_1,h):
    return f(t_k+h,y_k_1)
def phi_implizite_trapezregel(f,t_k,y_k,y_k_1,h):
    return 1/2*(f(t_k,y_k)+f(t_k+h,y_k_1))
def fixpunktfolge(phi,f,t_k,y_k,h,niter):
    wert=y_k
    for i in range(0,niter):
        wert=y_k+h*phi(f,t_k,y_k,wert,h)
    return wert
```

Allgemeine Definition der impliziten Verfahren. Wir realisieren die Lösung der Gleichung, indem wir *niter* Elemente der Fixpunktfolge berechnen.

[6]:
```
def einschritt_implizit(I_h,f,y0,phi,niter):
    N=I_h.size
    y_h=np.zeros(N)
    y_h[0]=y0
    for i in range(0,N-1):
        h=I_h[i+1]-I_h[i]
        y_h[i+1]=fixpunktfolge(phi,f,I_h[i],y_h[i],h,niter)
    return y_h
```

Wir testen wieder.

[14]:
```
N=16
# phi=phi_impliziter_euler
phi=phi_implizite_trapezregel
I_h=np.linspace(a,b,N)
y_h=einschritt_implizit(I_h,f,y0,phi,2)
plt.plot(I_h,y_h,'x',I_plot,loesungsfunc(I_plot))
plt.legend(['Implizite Approximation','Lösung'])
plt.title('Approximation der Lösung der AWA mit implizitem Verfahren');
```



[15]:
```
N=64
I_h=np.linspace(a,b,N)
```

```
y_h_1=einschritt_implizit(I_h,f,y0,phi_impliziter_euler,1)
y_h_2=einschritt_implizit(I_h,f,y0,phi_implizite_trapezregel,2)
y_korrekt=loesungsfunc(I_h)
print(max(np.abs(y_h_1-y_korrekt)))
print(max(np.abs(y_h_2-y_korrekt)))
```

```
0.02126444668256866
0.00011279377868422813
```

[ ]:

Runge-Kutta-Verfahren

Erinnerung:

Heun: $k_1 = f(t_k, y_k)$

$k_2 = f(t_k + h, y_k + h \cdot k_1)$

$\varphi = \frac{1}{2} \cdot k_1 + \frac{1}{2} \cdot k_2$

verb. Euler: $k_1 = f(t_k, y_k)$

$k_2 = f(t_k + \frac{h}{2}, y_k + \frac{h}{2} \cdot k_1)$

$\varphi = 0 \cdot k_1 + 1 \cdot k_2$

Runge-Kutta-Verfahren:

explizite Form (m-stufiges Verfahren):

$k_1 = f(t_k + \alpha_1 h, y_k)$

$k_2 = f(t_k + \alpha_2 h, y_k + h \beta_{21} k_1)$

$k_3 = f(t_k + \alpha_3 h, y_k + h(\beta_{31} k_1 + \beta_{32} k_2))$

$\vdots$

$k_m = f(t_k + \alpha_m h, y_k + h \sum_{\ell=0}^{m-1} \beta_{m\ell} k_\ell)$

$\varphi = \gamma_1 k_1 + \gamma_2 k_2 + \dots + \gamma_m k_m$

implizite Form: $k_i = f(t_k + \alpha_i h, y_k + h(\beta_{i1} k_1 + \dots + \beta_{im} k_m))$

Satz 2. a) Sei $f \in C^1$, $\sum_{i=1}^{m} \gamma_i = 1$. $\leftarrow$ notwendig

Dann ist das Verfahren konsistent von Ordnung 1.

b) Sei $f \in C^2$, $\sum \gamma_i = 1$, $\sum_\ell \beta_{i\ell} = \alpha_i$, $\sum \alpha_i \gamma_i = \frac{1}{2}$.

Dann ist das Verfahren konsistent von Ordnung 2.

Beweis: Zur Übergruppierung der Konsistenz setzen wir für y_k ein $y(t)$ ein:

$$k_i = f(t + \alpha_i h, y(t) + h \sum_\ell \beta_{i\ell} k_\ell)$$

$$= f(t, y(t)) + \alpha_i h f_t(t, y(t)) + h \sum_\ell \beta_{i\ell} k_\ell \cdot f_y(t, y(t)) + \mathcal{O}(h^2)$$

$\uparrow$ $f(t, y(t)) + \mathcal{O}(h)$

$$= y'(t) + h\left[\alpha_i f_t + \sum_\ell \beta_{i\ell} k_\ell \cdot f_y\right](t, y(t)) + \mathcal{O}(h^2)$$

$\tau_h(t, y(t)) = \frac{y(t+h) - y(t)}{h} - \varphi(t, y(t))$

$$= y'(t) + \frac{h}{2} y''(t) + \mathcal{O}(h^2) - \sum_{i=1}^{m} \gamma_i \cdot k_i$$

$$= y'(t) + \frac{h}{2} y''(t) \cdot \mathcal{O}(h^2) - \sum_{i=1}^{m} \gamma_i y'(t)$$

$$- h \sum_{i=1}^{m} \gamma_i \left[\alpha_i f_t + \sum_\ell \beta_{i\ell} k_\ell \cdot f_y\right](t, y(t)) + \mathcal{O}(h^2)$$

zu a) konsistent mit Ordnung 1 ($\underbrace{\alpha_i}_{1}$): $y'(t) + f(t, y(t))$

b) $y''(t) + \left(f_t + f \cdot f_y\right)(t, y(t))$

$- h \sum_{i=1}^{m} \underbrace{\alpha_i \gamma_i}_{1/2} \cdot y''(t)$ $\Rightarrow$ Ordnung 2

Idee: Wähle die Koeffizienten so, dass die Ordnung möglichst groß ist.

Federbeispiel

June 21, 2020

1 Anwendungen / Energieerhaltung

[ ]:

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import sympy
```

2 Steinwurf

Ein Stein werde geworfen, wir bezeichnen die aktuelle Höhe mit $s(t)$. Dann gilt $s''(t) = -g$. Wir verwandeln diese Differentialgleichung in ein System in den Funktionen $s(t)$ und $v(t)$ und erhalten mit der Vertikalgeschwindigkeit $v(t)$

$$s'(t) = v(t) \tag{1}$$
$$v'(t) = -g. \tag{2}$$

Die Energie des Systems setzt sich zusammen aus der kinetischen Energie (Geschwindigkeit) und der potentiellen Energie, also (Masse=1)

$$E(t) = \frac{v(t)^2}{2} + s(t)g.$$

$E(t)$ ist konstant wegen

$$E'(t) = v(t)v'(t) + s'(t)g = -gv(t) + gv(t) = 0.$$

In diesem Beispiel ist

$$f\left(t, \begin{pmatrix} s \\ v \end{pmatrix}\right) = \begin{pmatrix} v \\ -g \end{pmatrix}.$$

Für das Eulerverfahren gilt also

$$\begin{pmatrix} s_{k+1} \\ v_{k+1} \end{pmatrix} = \begin{pmatrix} s_k \\ v_k \end{pmatrix} + hf\left(t_k, \begin{pmatrix} s_k \\ v_k \end{pmatrix}\right) = \begin{pmatrix} s_k + hv_k \\ v_k - hg \end{pmatrix}.$$

Die Energie sollte auch in der Diskretisierung konstant sein. Wir definieren mit E_k die Energie der Approximation zum Zeitpunkt t_k. Es gilt dann:

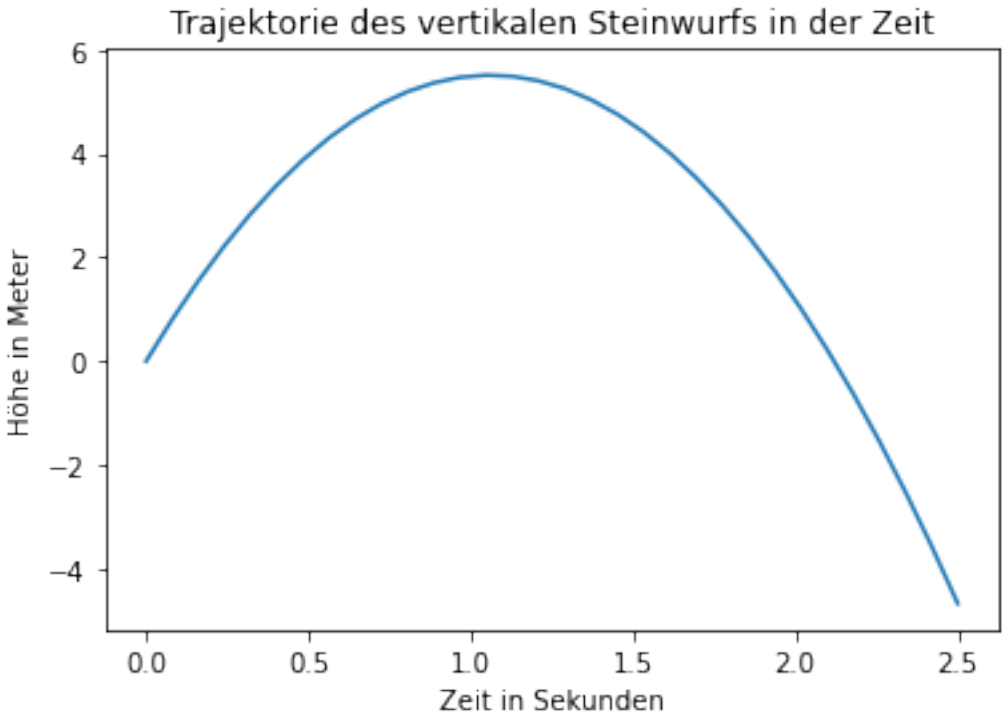
$$\begin{aligned} E_{k+1} - E_k &= \frac{v_{k+1}^2}{2} + s_{k+1}g - \frac{v_k^2}{2} - s_k g \\ &= \frac{(v_k - hg)^2}{2} + (s_k + hv_k)g - \frac{v_k^2}{2} - s_k g \\ &= \frac{h^2 g^2}{2} \end{aligned}$$

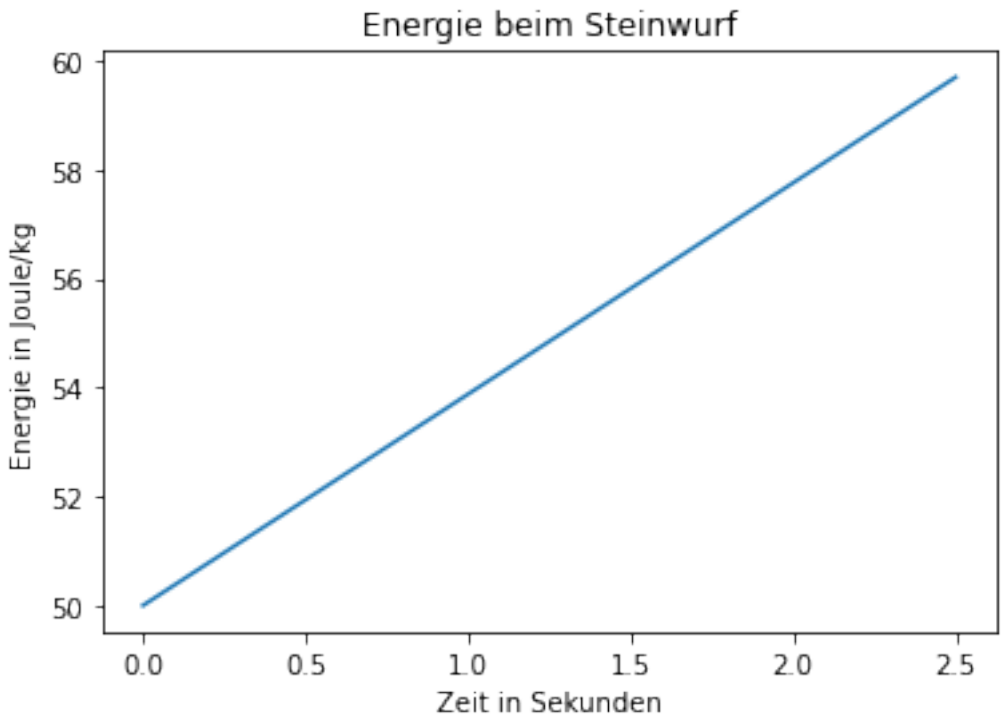
Die Energie nimmt also im Verlauf des Verfahrens zu. Dies sollten wir in der numerischen Simulation sehen. Der Stein wird in der Simulation nach oben geworfen und fällt dann zurück.

[19]:
```
# Explizite Einschrittverfahren für höhere Dimensionen.
# Nur eine minimale Änderung gegen skalare ist notwendig.
g=9.81
def f_steinwurf(t,y):
    return np.array([y[1],-g])
y0_steinwurf=np.array([0,10])
a=0
b=2.5
def E_steinwurf(y):
    return y[:,1]**2/2+y[:,0]*g

def phi_euler(f,t,y,h):
    return f(t,y)
def phi_verbesserter_euler(f,t,y,h):
    return f(t+h/2,y+h/2*f(t,y))
def phi_heun(f,t,y,h):
    f1=f(t,y)
    return 1/2*(f1+f(t+h,y+h*f1))
def einschritt_explizit(I_h,f,y0,phi):
    N=I_h.size
    y_h=np.zeros([N,y0.size])
    y_h[0]=y0
    for i in range(0,N-1):
        h=I_h[i+1]-I_h[i]
        y_h[i+1]=y_h[i]+h*phi(f,I_h[i],y_h[i],h)
    return y_h

N=32
I_h=np.linspace(a,b,N)
y_h=einschritt_explizit(I_h,f_steinwurf,y0_steinwurf,phi_euler)
plt.plot(I_h,y_h[:,0])
plt.title('Trajektorie des vertikalen Steinwurfs in der Zeit')
plt.xlabel('Zeit in Sekunden')
plt.ylabel('Höhe in Meter')
plt.figure()
E_h=E_steinwurf(y_h)
plt.plot(I_h,E_h)
```

```
plt.title('Energie beim Steinwurf')
plt.xlabel('Zeit in Sekunden')
plt.ylabel('Energie in Joule/kg')
h=(b-a)/(N-1)
print('Vorhergesagte Energiedifferenz:',(N-1)*h**2*g**2/2)
print('Tatsächliche Energiedifferenz: ',E_h[N-1]-E_h[0])
```

```
Vorhergesagte Energiedifferenz: 9.701219758064516
Tatsächliche Energiedifferenz:  9.701219758064568
```





Nun vergleichen wir dieses Ergebnis mit dem impliziten Eulerverfahren. Wir wollen hier die Funktion v aus dem Beweis über die Wohldefiniertheit der impliziten Verfahren direkt ausrechnen, und nicht über die Fixpunktfolge gehen, wie man es üblicherweise tut.

Im impliziten Eulerverfahren gilt

$$y_{k+1} = y_k + hf(t_{k+1}, y_{k+1}),$$

also

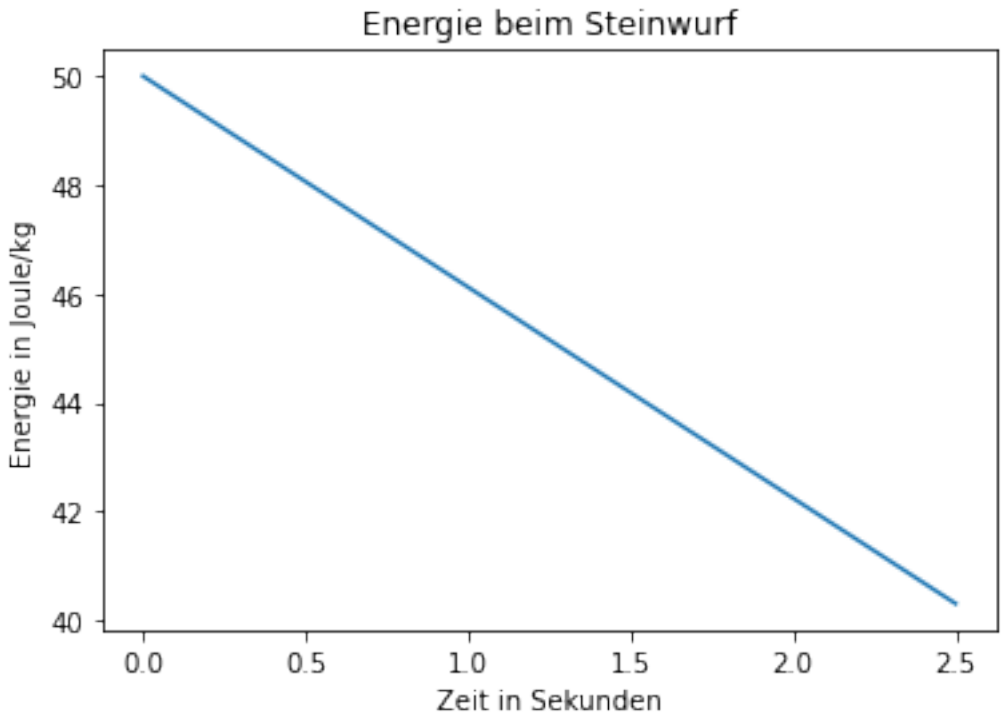
$$\begin{pmatrix} s_{k+1} \\ v_{k+1} \end{pmatrix} = \begin{pmatrix} s_k \\ v_k \end{pmatrix} + h \begin{pmatrix} v_{k+1} \\ -g \end{pmatrix} = \begin{pmatrix} s_k \\ v_k \end{pmatrix} + h \begin{pmatrix} v_k - hg \\ -g \end{pmatrix}.$$

Das φ, das wir dadurch erhalten, ist jetzt natürlich nicht mehr universell, sondern speziell für dieses f gewählt worden. Durch diesen Trick wird das implizite Verfahren zu einem expliziten, wie in der Vorlesung, und wir können unsere Formel für explizite Verfahren nutzen.

Wir schauen direkt auf die erzeugte Energie.

[3]:
```
def phi_steinwurf_euler_implizit(f,t,y,h):
    return np.array([y[1]-h*g,-g])
y_h=einschritt_explizit(I_h,f_steinwurf,y0_steinwurf,phi_steinwurf_euler_implizit)
E_h=E_steinwurf(y_h)
plt.plot(I_h,E_h)
plt.title('Energie beim Steinwurf')
plt.xlabel('Zeit in Sekunden')
plt.ylabel('Energie in Joule/kg')
```

[3]: Text(0, 0.5, 'Energie in Joule/kg')



Ohne das jetzt extra nochmal nachzurechnen: Man sieht schon, das ist auch nicht besser. Jetzt nimmt die Energie halt ab.

Vielleicht hilft die Mitte aus explizitem und implizitem Verfahren. Wir nutzen die implizite Trapezregel. Da gilt

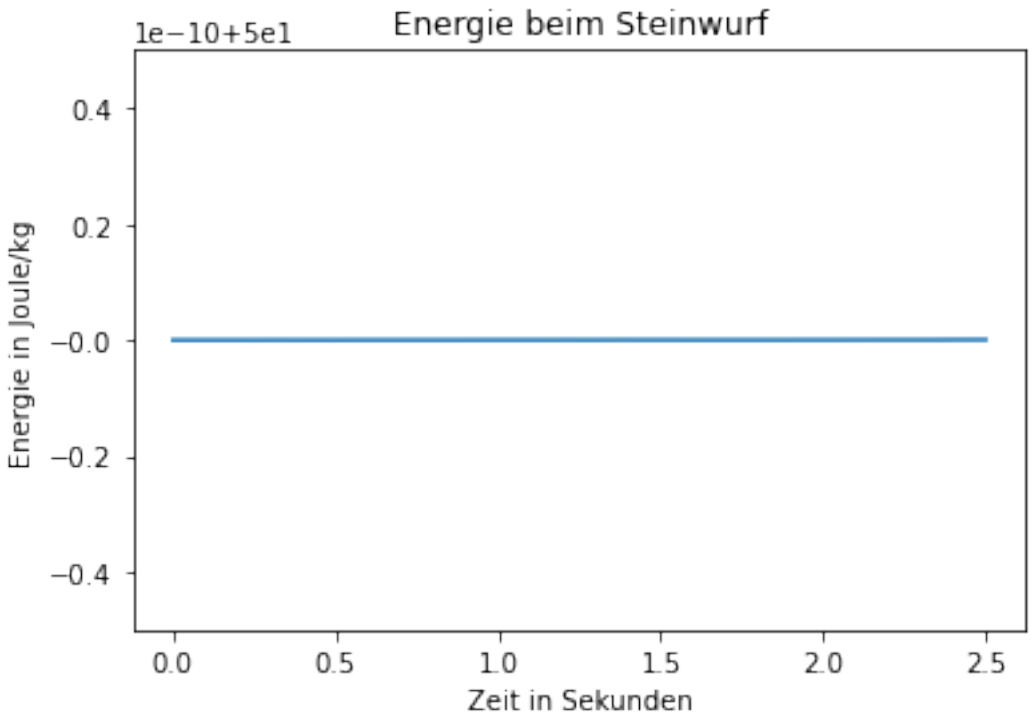
$$y_{k+1} = y_k + h\frac{1}{2}(f(t_k, y_k) + f(t_{k+1}, y_{k+1})).$$

Für unsere Gleichung liefert das

$$\begin{pmatrix} s_{k+1} \\ v_{k+1} \end{pmatrix} = \begin{pmatrix} s_k \\ v_k \end{pmatrix} + \frac{h}{2}\left(\begin{pmatrix} v_k \\ -g \end{pmatrix} + \begin{pmatrix} v_{k+1} \\ -g \end{pmatrix}\right) = \begin{pmatrix} s_k \\ v_k \end{pmatrix} + h\begin{pmatrix} v_k - \frac{hg}{2} \\ -g \end{pmatrix}$$

[4]:
```
def phi_steinwurf_trapez_implizit(f,t,y,h):
    return np.array([y[1]-h*g/2,-g])
y_h=einschritt_explizit(I_h,f_steinwurf,y0_steinwurf,phi_steinwurf_trapez_implizit)
E_h=E_steinwurf(y_h)
plt.plot(I_h,E_h)
plt.title('Energie beim Steinwurf')
plt.xlabel('Zeit in Sekunden')
plt.ylabel('Energie in Joule/kg')
```

[4]: `Text(0, 0.5, 'Energie in Joule/kg')`



...und jetzt ist tatsächlich die Energie konstant. Diese Formel spiegelt den wahren zugrundeliegenden physikalischen Sachverhalt wider, und ist damit eine sinnvolle Diskretisierung. Die anderen sind es nicht.

Sicherheitshalber rechnen wir das nochmal nach:

$$\begin{aligned} E_{k+1} - E_k &= \frac{v_{k+1}^2}{2} + s_{k+1}g - \frac{v_k^2}{2} - s_k g \\ &= \frac{(v_k - hg)^2}{2} + (s_k + hv_k - \frac{h^2 g}{2})g - \frac{v_k^2}{2} - s_k g \\ &= 0 \end{aligned}$$

Man könnte nun sagen - Ja und? Dann ist die Energie halt nicht konstant, wir erwarten ja sowieso nicht, dass das Ergebnis absolut korrekt ist. Und für $h \to 0$ geht auch, wie erwartet, die Energiedifferenz gegen 0.

Damit macht man es sich zu einfach. Wir schauen abschließend auf eins der Motivationsbeispiele der Vorlesung und klären eine damalige seltsame Beobachtung.

3 Federbeispiel revisited

Wir schauen nochmal auf das Federbeispiel aus dem Motivationskapitel der Vorlesung. Ein an einer Feder befestigtes Gewicht (1 kg) bewegte sich auf und ab. Wir erhielten damals die Differentialgleichung für die Höhe $s(t)$ und die Geschwindigkeit $v(t)$

$$s'(t) = v(t),\ v'(t) = -g - s(t)$$

mit Federkonstante 1. Wir haben hier keine Reibung oder Ähnliches berücksichtigt, d.h. wir erwarten eine Lösung, die mit konstanter Amplitude schwingt (das System lässt sich auch leicht analytisch lösen). Die Funktion

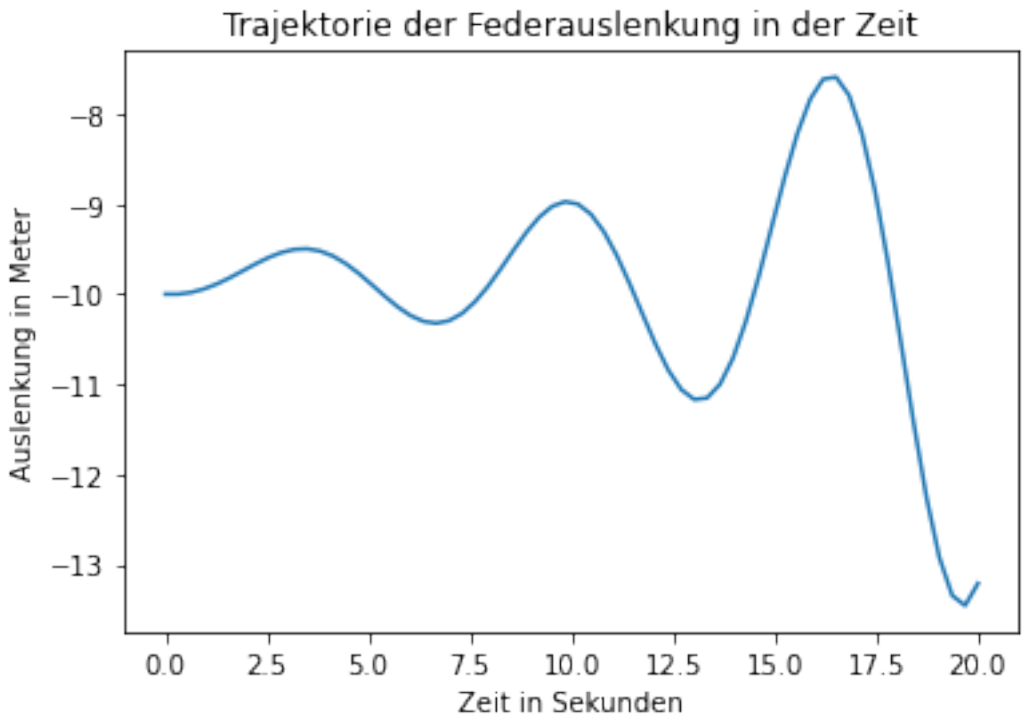
$$f\left(t, \begin{pmatrix} s(t) \\ v(t) \end{pmatrix}\right) = \begin{pmatrix} v(t) \\ -g - s(t) \end{pmatrix}$$

definiert die Differentialgleichung. Wir berechnen wieder eine Näherung mit Euler.

[22]:
```
g=9.81
def f_feder(t,y):
    return np.array([y[1],-g-y[0]])
y0_feder=np.array([-10,0])
a=0
b=20
```

[25]:
```
N=64
I_h=np.linspace(a,b,N)
y_h=einschritt_explizit(I_h,f_feder,y0_feder,phi_euler)
plt.plot(I_h,y_h[:,0])
plt.title('Trajektorie der Federauslenkung in der Zeit')
plt.xlabel('Zeit in Sekunden')
plt.ylabel('Auslenkung in Meter')
plt.figure()
```

[25]:
```
<Figure size 432x288 with 0 Axes>
```




```
<Figure size 432x288 with 0 Axes>
```

Das ist nicht, was wir erwarten.

Im Grunde wird die Periodizität der Lösung ja gut angenähert (tatsächlich stimmt die Frequenz auch), aber die Amplitude wird immer größer - Vermutung: Die Energie im System wird immer größer. Ich glaube, niemand würde dies als eine vernünftige Lösung akzeptieren.

Wir testen wieder die Energie. In diesem Fall kommt noch die in der Feder gespeicherte Energie hinzu. Wir definieren also

$$E(t) = \frac{v(t)^2}{2} + gs(t) + \frac{s(t)^2}{2}.$$

Für die Ableitung gilt

$$E'(t) = v(t)v'(t) + gs'(t) + s(t)s'(t) = v(t)(-g - s(t)) + (t) + gv(t) + s(t)v(t) = 0$$

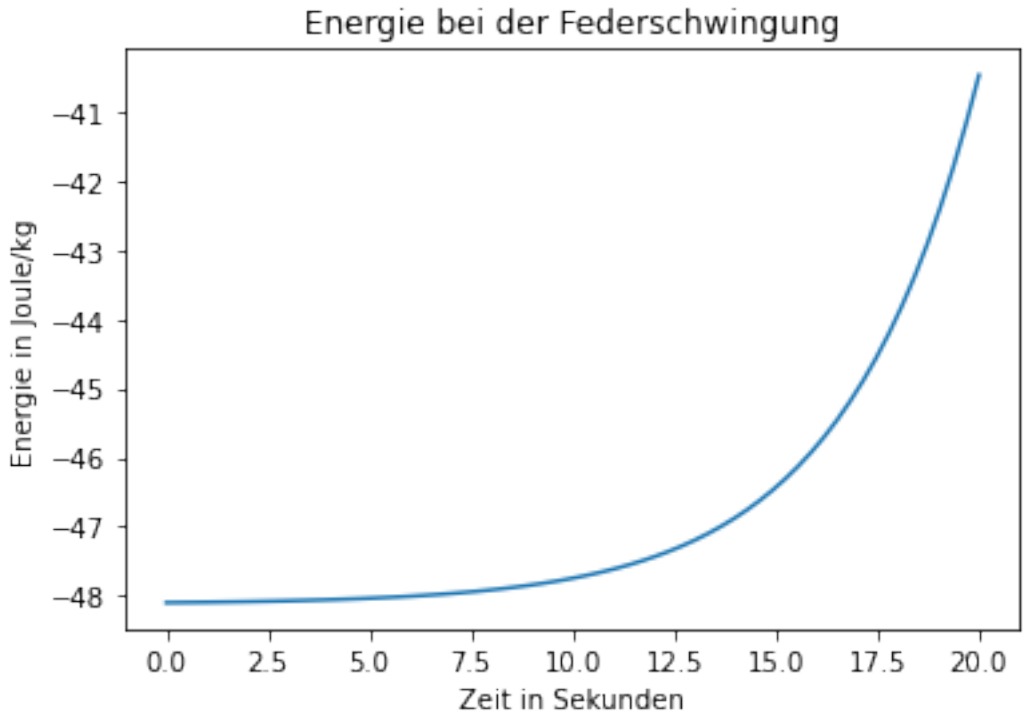
und damit ist die Energie wieder konstant.

Vermutung: In der Eulerdiskretisierung nimmt diese Energie zu.

[7]:
```
def E_feder(y):
    return y[:,1]**2/2+y[:,0]*g+y[:,0]**2/2
E_h=E_feder(y_h)
plt.plot(I_h,E_h)
plt.title('Energie bei der Federschwingung')
```

```
plt.xlabel('Zeit in Sekunden')
plt.ylabel('Energie in Joule/kg')
```

[7]:
```
Text(0, 0.5, 'Energie in Joule/kg')
```



...und die Energie nimmt tatsächlich zu (die Energie ist negativ, denn $s(t)$ ist nicht wirklich die Auslenkung, da müsste man eigentlich noch die Nullstellung abziehen).

Wir berechnen hier die Energiedifferenz diesmal mit sympy und Federkonstante c.

[8]:
```
import sympy
sk,vk,h,g_sym,c=sympy.symbols('s_k v_k h g c')
sk_1=sk+h*vk
vk_1=vk+h*(-g_sym-c*sk)
def E_feder_sym(s,v):
    return v*v/2+s*g_sym+c*s*s/2
display(sympy.simplify(E_feder_sym(sk_1,vk_1)-E_feder_sym(sk,vk)))
```

$$\frac{h^2\left(c^2 s_k^2 + 2cgs_k + cv_k^2 + g^2\right)}{2}$$

und wir erhalten exakt das gleiche Ergebnis wie oben: Expliziter Euler erhöht die Energie, und deshalb sehen wir keine perfekte Schwingung.

Zur Übung berechnen wir auch hier wieder das implizite Eulerverfahren. Wie oben gilt

$$\begin{pmatrix} s_{k+1} \\ v_{k+1} \end{pmatrix} = \begin{pmatrix} s_k \\ v_k \end{pmatrix} + h \begin{pmatrix} v_{k+1} \\ -g - s_{k+1} \end{pmatrix}. \tag{3}$$

Wir setzen s_{k+1} unten ein:

$$v_{k+1} = v_k - hg - h(s_k + hv_{k+1})$$

und damit

$$v_{k+1} = \frac{1}{1+h^2}(v_k - hg - hs_k).$$

Liefert für die Energie:

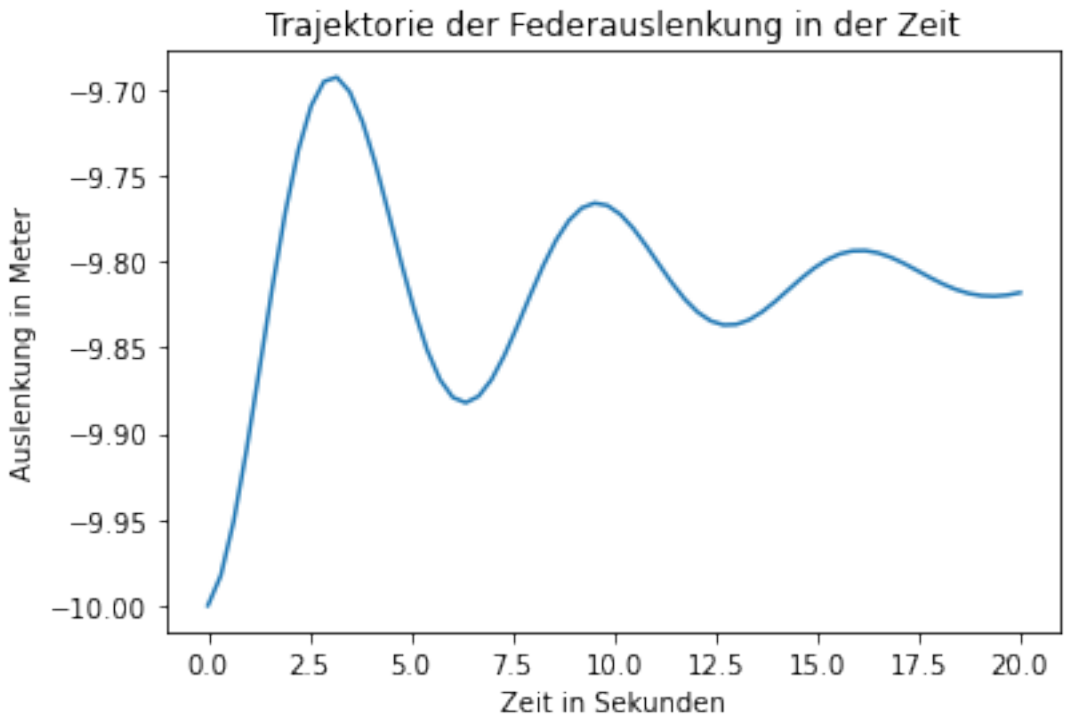
[9]:
```
vk_1=1/(1+c*h*h)*(vk+h*(-g_sym-c*sk))
sk_1=sk+h*vk_1
display(sympy.factor(sympy.simplify(sympy.
 ↪expand(E_feder_sym(sk_1,vk_1)-E_feder_sym(sk,vk)))))
```

$$-\frac{h^2\left(c^2s_k^2 + 2cgs_k + cv_k^2 + g^2\right)}{2\left(ch^2+1\right)}$$

Und wieder (fast) das Negative von Euler. Wir wissen schon, was wir bei der Implementation erwarten dürfen.

[10]:
```
def phi_feder_euler_implizit(f,t,y,h):
    yk_1=1/(1+h*h)*(y[1]-h*g-h*y[0])
    return np.array([yk_1,(yk_1-y[1])/h])
y_h=einschritt_explizit(I_h,f_feder,y0_feder,phi_feder_euler_implizit)
plt.plot(I_h,y_h[:,0])
plt.title('Trajektorie der Federauslenkung in der Zeit')
plt.xlabel('Zeit in Sekunden')
plt.ylabel('Auslenkung in Meter')
plt.figure()
```

[10]:
```
<Figure size 432x288 with 0 Axes>
```

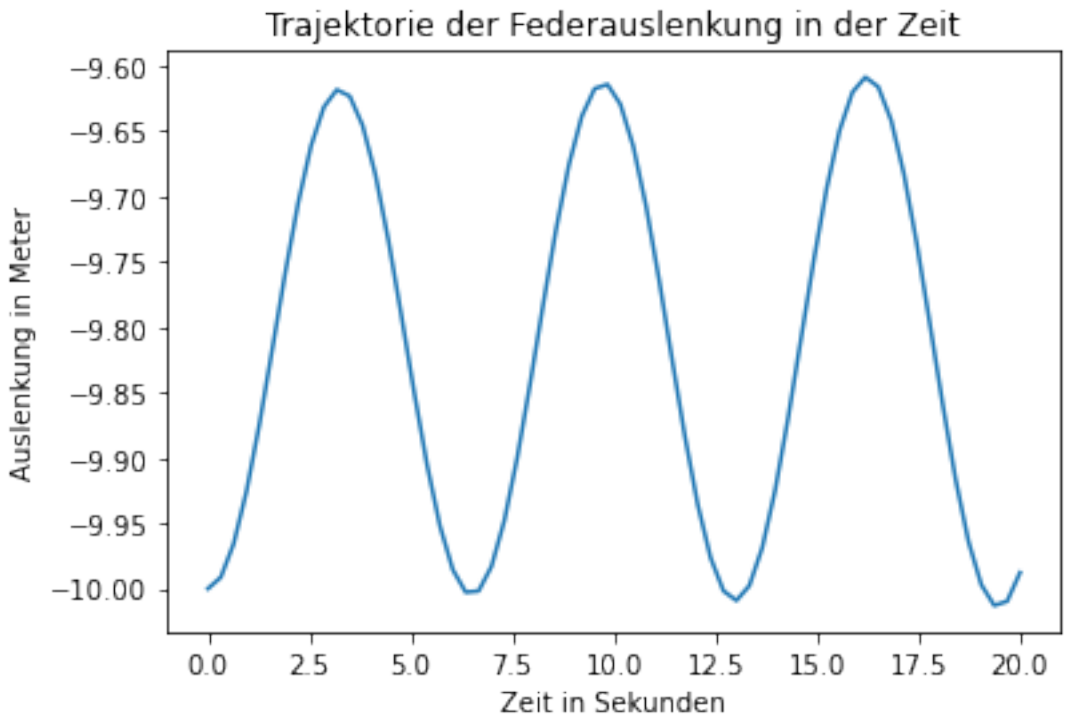


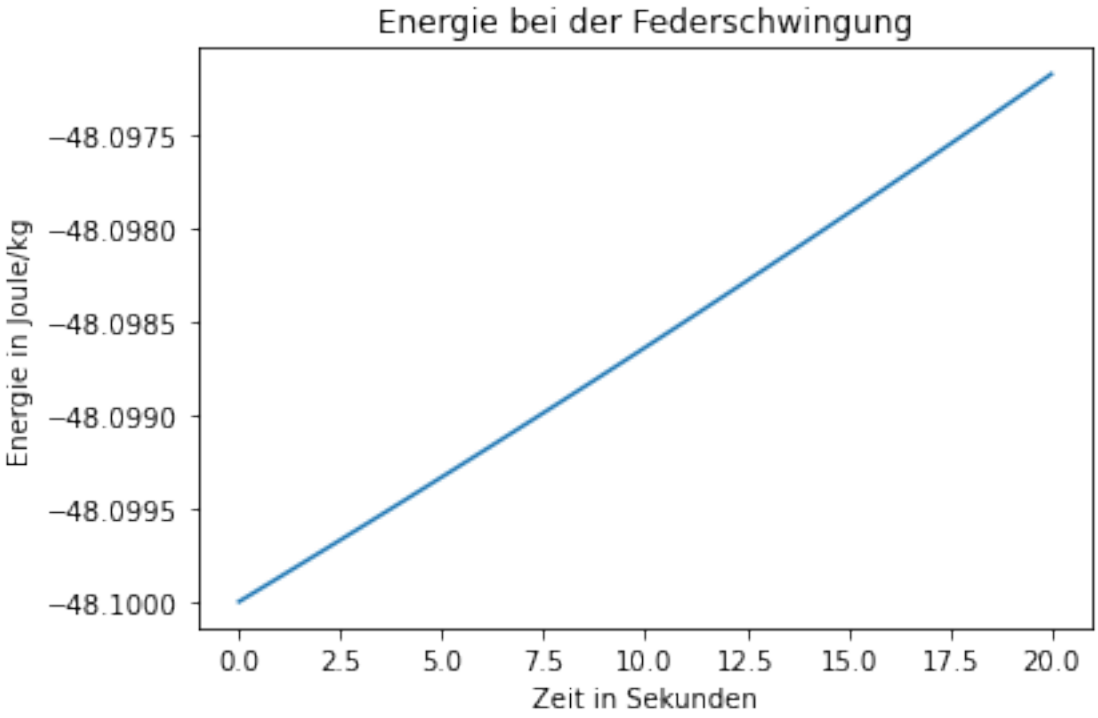

```
<Figure size 432x288 with 0 Axes>
```

Und wieder wie erwartet. Wir wissen, was wir zu tun haben.

[11]:
```
def phi_feder_trapez_implizit(f,t,y,h):
    return (phi_feder_euler_implizit(f,t,y,h)+phi_euler(f,t,y,h))/2
y_h=einschritt_explizit(I_h,f_feder,y0_feder,phi_feder_trapez_implizit)
plt.plot(I_h,y_h[:,0])
plt.title('Trajektorie der Federauslenkung in der Zeit')
plt.xlabel('Zeit in Sekunden')
plt.ylabel('Auslenkung in Meter')
plt.figure()
E_h=E_feder(y_h)
plt.plot(I_h,E_h)
plt.title('Energie bei der Federschwingung')
plt.xlabel('Zeit in Sekunden')
plt.ylabel('Energie in Joule/kg')
```

[11]:
```
Text(0, 0.5, 'Energie in Joule/kg')
```

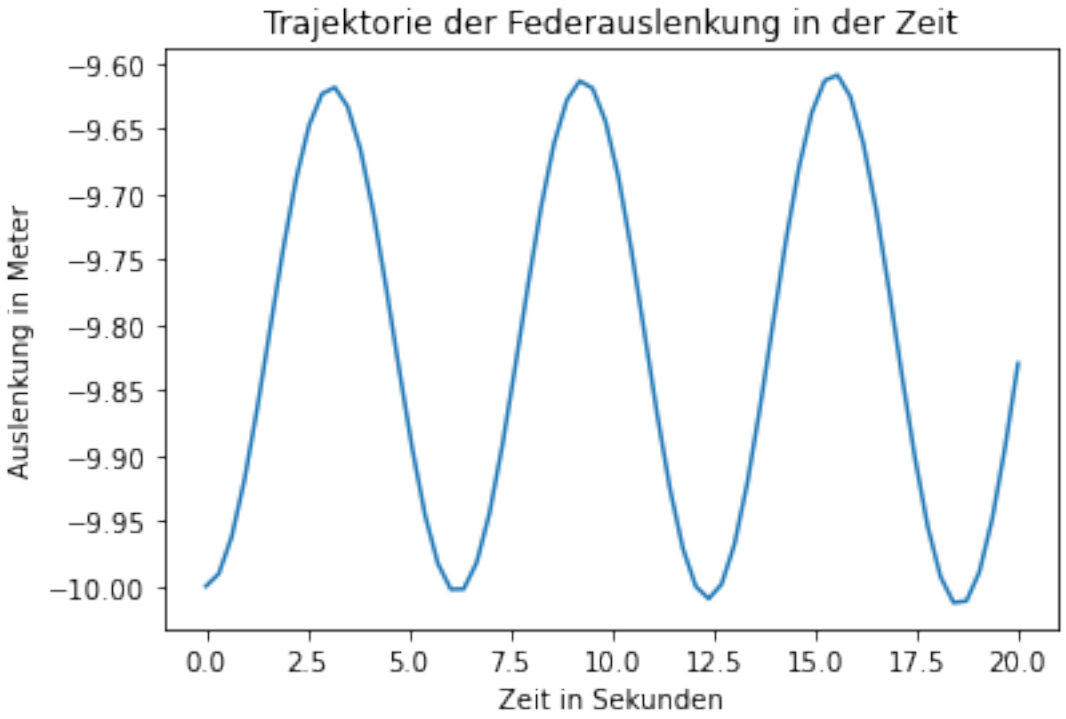

Trajektorie der Federauslenkung in der Zeit
Auslenkung in Meter
−9.60
−9.65
−9.70
−9.75
−9.80
−9.85
−9.90
−9.95
−10.00
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Zeit in Sekunden

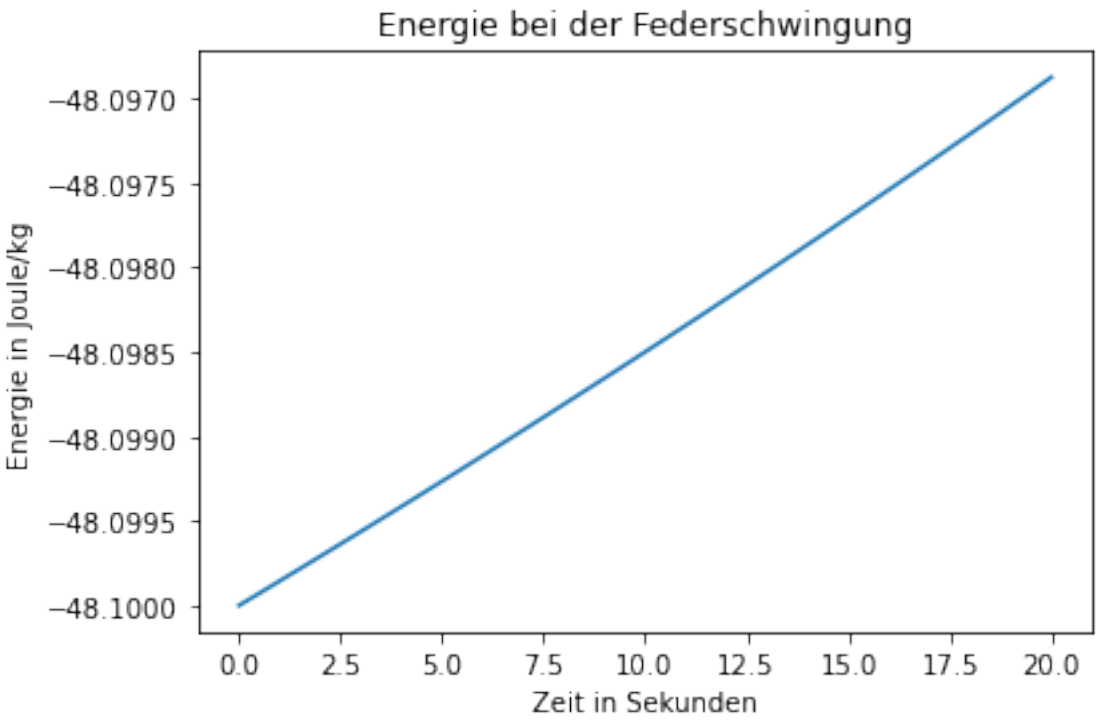

Energie bei der Federschwingung
Energie in Joule/kg
−48.0975
−48.0980
−48.0985
−48.0990
−48.0995
−48.1000
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Zeit in Sekunden

Die implizite Trapezregel hält tatsächlich die Energie konstant… Nein, nur fast. Wir trösten uns: Auch Heun und verbesserter Euler sind nicht exakt. Aufgabe: Finden Sie ein Verfahren, dass die Energie konstant lässt.

[17]:
```
y_h=einschritt_explizit(I_h,f_feder,y0_feder,phi_heun)
plt.plot(I_h,y_h[:,0])
plt.title('Trajektorie der Federauslenkung in der Zeit')
plt.xlabel('Zeit in Sekunden')
plt.ylabel('Auslenkung in Meter')
plt.figure()
E_h=E_feder(y_h)
plt.plot(I_h,E_h)
plt.title('Energie bei der Federschwingung')
plt.xlabel('Zeit in Sekunden')
plt.ylabel('Energie in Joule/kg')
```

[17]:
```
Text(0, 0.5, 'Energie in Joule/kg')
```





[ ]:

4 Zusammenfassung

Bei der Betrachtung von numerischen Algorithmen für Anfangswertaufgaben kommt es nicht nur auf die Konsistenzordnung an.

Im betrachteten Fall (Feder) war die Gleichung so, dass man schon auf einem groben Gitter einen Teil der Lösung (die Schwingfrequenz) sehr gut annähern konnte, aber durch den Energieverlust sehr hoch auflösen musste, um die Amplitude korrekt zu bekommen. Differentialgleichungen mit dieser (dann korrekt mathematisch definierten) Eigenschaft heißen steif. In diesem Fall gibt es geeignete und weniger geeignete Verfahren, wie wir gesehen haben.

Anmerkung: Natürlich sind alle hier vorgestellten Verfahren am Ende konvergent. Sie unterscheiden sich darin, wie fein man das Gitter tatsächlich wählen muss.

[ ]:

Fehlerabschätzung + Schrittweitensteuerung

Frage: Wie können wir den Fehler einer Abschätzung angeben?

→ Globale Diskretisierungsfehler

Diskretes Lemma von Gronwall:

$$\|e_h\|_\infty \leq \sum_j h \, |\tau_h(t_j, y(t_j))| \cdot e^{(t_j - a) \cdot L}$$

Einfluss des lok. Fehlers auf den Gesamtfehler,

$\tau_j' = h \cdot \tau_j \cdot e^{L(t_j - a)}$, $\|e_h\|_\infty \sim \sum \tau_j'$.

Idee: Wähle h_j so, dass $\tau_j' \sim \tau_k'$, abhängig von j.

Benötigt: Abschätzung für lokalen Diskretisierungsfehler.

Idee: 1. In jedem Schritt, benutze 2 Verfahren mit unterschiedlicher Konsistenzordnung.

2. Schätze den Fehler als Unterschied der Verfahren.

3. Wähle h optimal.

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
```

Fehlerschätzung

In [2]:

```
def phi_euler(f,t,y,h):
    return f(t,y)
def phi_verbesserter_euler(f,t,y,h):
    return f(t+h/2,y+h/2*f(t,y))
def phi_heun(f,t,y,h):
    f1=f(t,y)
    return 1/2*(f1+f(t+h,y+h*f1))
def einschritt_explizit(I_h,f,y0,phi):
    N=I_h.size
    y_h=np.zeros([N,y0.size])
    y_h[0]=y0
    for i in range(0,N-1):
        h=I_h[i+1]-I_h[i]
        y_h[i+1]=y_h[i]+h*phi(f,I_h[i],y_h[i],h)
    return y_h
```

In [3]:

```
def f(t,y):
    return 1+y*y
def yquer(t):
    return np.tan(t)
y0=0
a=0
b=1.5
#def f(t,y):
#    return y
#def yquer(t):
#    return np.exp(t)
#y0=1

h=1e-3
# Konsistenzfehler an der Stelle t
def konsistenzfehler(yquer,f,phi,t,h):
    fehler=(yquer(t+h)-yquer(t))/h-phi(f,t,yquer(t),h)
    return fehler
def schrittfunk(yquer,f,phi,t,h):
    return yquer(t)+h*phi(f,t,yquer(t),h)
t=0.7
euler=schrittfunk(yquer,f,phi_euler,t,h)
heun=schrittfunk(yquer,f,phi_heun,t,h)
true_value=yquer(t+h)
print('Fehler Euler:',abs(true_value-euler))
print('Fehler Heun:',abs(true_value-heun))
print('Differenz Euler - Heun',abs(heun-euler))
```

```
Fehler Euler: 1.4416342015710626e-06
Fehler Heun: 3.2345970346625563e-10
Differenz Euler - Heun 1.4413107418675963e-06
```

Verfahren von Dormand-Prince

Erinnerung Runge-Kutta:

$$\varphi(t_k, y_k, h_k) = \gamma_1 f_1 + \gamma_2 f_2 + \ldots \gamma_m f_m$$

$$\begin{aligned}
f_1 &= f(t_k + \alpha_1 h_k, y_k + h_k \sum_{l=1}^{m} \beta_{1,l} f_l) \\
f_2 &= f(t_k + \alpha_2 h_k, y_k + h_k \sum_{l=1}^{m} \beta_{2,l} f_l) \\
&\vdots \\
f_m &= f(t_k + \alpha_m h_k, y_k + h_k \sum_{l=1}^{m} \beta_{m,l} f_l).
\end{aligned}$$

Idee: Benutze unterschiedliche γ_k.

In [5]:

```
import scipy
import scipy.integrate
y0=[0]
lsg=scipy.integrate.solve_ivp(f, [a,b],y0,rtol=1e-8,method='RK45')
```

In [6]:

```
print(lsg)
```

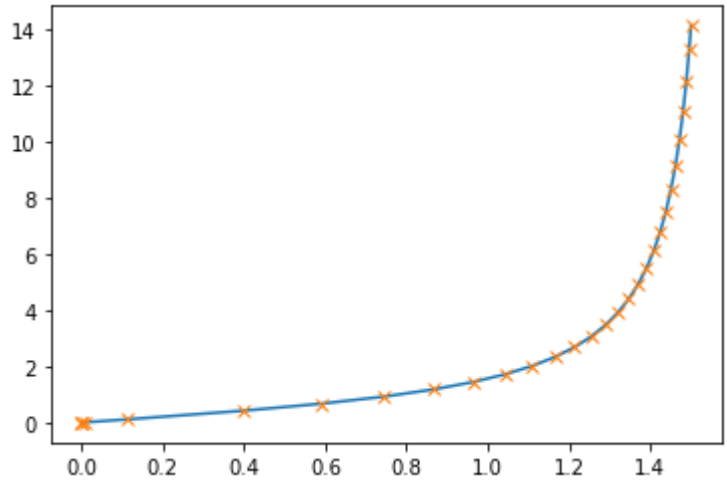
```
  message: 'The solver successfully reached the end of the integration int
erval.'
     nfev: 320
     njev: 0
      nlu: 0
      sol: None
   status: 0
  success: True
        t: array([0.00000000e+00, 1.00000000e-04, 1.10000000e-03, 1.110000
00e-02,
       1.11100000e-01, 3.97422868e-01, 5.90724370e-01, 7.45766653e-01,
       8.65720790e-01, 9.62858933e-01, 1.04267093e+00, 1.10924888e+00,
       1.16539894e+00, 1.21317227e+00, 1.25411083e+00, 1.28940408e+00,
       1.31998810e+00, 1.34661152e+00, 1.36988094e+00, 1.39029316e+00,
       1.40825875e+00, 1.42411965e+00, 1.43816248e+00, 1.45062896e+00,
       1.46172390e+00, 1.47162173e+00, 1.48047154e+00, 1.48840131e+00,
       1.49552124e+00, 1.50000000e+00])
 t_events: None
        y: array([[0.00000000e+00, 1.00000000e-04, 1.10000044e-03, 1.11004
559e-02,
        1.11559378e-01, 4.19758705e-01, 6.70605216e-01, 9.23719993e-01,
        1.17508502e+00, 1.43708453e+00, 1.71408454e+00, 2.01054438e+00,
        2.33007791e+00, 2.67599428e+00, 3.05143093e+00, 3.45945973e+00,
        3.90315186e+00, 4.38562246e+00, 4.91006280e+00, 5.47976505e+00,
        6.09814232e+00, 6.76874595e+00, 7.49528093e+00, 8.28162042e+00,
        9.13181970e+00, 1.00501300e+01, 1.10410127e+01, 1.21091533e+01,
        1.32594766e+01, 1.41013862e+01]])
 y_events: None
```

In [7]:

```
I_h=lsg.t
y_h=lsg.y[0,:]
plt.plot(I_h,y_h,I_h,y_h,'x')
```

Out[7]:

```
[<matplotlib.lines.Line2D at 0x7f8eaa019d10>,
 <matplotlib.lines.Line2D at 0x7f8eaa019f10>]
```



In [9]:

```
print(y_h-np.tan(I_h))
```

```
[ 0.00000000e+00  0.00000000e+00 -2.16840434e-19 -3.46944695e-18
 -2.57605187e-11 -8.28939845e-09 -4.69073531e-08 -1.32489828e-07
 -2.37264706e-07 -3.67336794e-07 -5.26518478e-07 -7.22703915e-07
 -9.65105530e-07 -1.26464426e-06 -1.63416005e-06 -2.08875598e-06
 -2.64618738e-06 -3.32730041e-06 -4.15652465e-06 -5.16242760e-06
 -6.37833961e-06 -7.84305896e-06 -9.60164757e-06 -1.17063289e-05
 -1.42175012e-05 -1.72048796e-05 -2.07487851e-05 -2.49415960e-05
 -2.98893841e-05 -3.37840690e-05]
```

In [ ]:

Lineare Mehrschrittverfahren: Einführung, Definitionen, Herleitg, Konsistenz.

ESV:
- Konsistenz (lokale Fehler geht gegen Null)
- Konvergenz (Gesamtfehler geht gegen Null)
- Stabilität (Aus Konsistenz folgt Konvergenz, Diskretes Lemma von Gronwall)

Beispiel für Mehrschrittverfahren:

ESV: $y_{k+m} = y_k + h\,\varphi(t_k, y_k, h)$

MSV: $y_{k+m} = y_k + h\,\varphi(t_k, y_{k+m-1}, y_k, h)$ (I) (implizit: $\ldots y_{k+m}$).

$y_{k+m} = y_{k+m-1} + h\,\varphi(t_{k+m}, y_{k+m-1}, y_k, h)$ (II)

$y_k \sim \bar{y}(t_k)$ usw.

(I): Adams-Bashforth (explizit)
Adams-Moulton (implizit)

(II) Nyström (expl.)
Milne-Simpson (impl.)

$$y(t_{k+m}) = y(t_k) + \int_{t_k}^{t_{k+m}} f(s, y(s))\,ds$$

explizit: $y_k \sim y(t_k), y_{k+m} \sim y(t_{k+m})$. $f_k = f(t_k, y_k) \sim f(t_k, y(t_k))$.

Interpolationspolynom $p(s) = f_{k+m} + \frac{s - t_{k+m}}{t_k - t_{k+m}} \cdot (f_k - f_{k+m})$

$$= \frac{s - t_{k+m}}{h} f_k + \frac{t_k - s}{h} \cdot f_{k+m} \quad \text{(Lagrange)}$$

St. F.: $\frac{(s - t_{k+m})^2}{2h} f_k - \frac{(t_k - s)^2}{2h} \cdot f_{k+m}$

$$\int_{t_k}^{t_{k+m}} p(s)\,ds = 2h f_k - \frac{h}{2} f_{k+m} - \frac{h}{2} f_k$$

$$= h\left(\frac{3}{2} f_k - \frac{1}{2} f_{k+m}\right)$$

$$\Rightarrow y_{k+m} = y_k + h\left(\frac{3}{2} f_k - \frac{1}{2} f_{k+m}\right) \qquad \left[\varphi = \frac{3}{2} f_k - \frac{1}{2} f_{k+m}\right]$$

$$\int_{t_{k+m}}^{t_{k+m}} p(s)\,ds = 2h f_k - \frac{h}{2} f_{k+m} + \frac{h}{2} f_{k+m}$$

$$= 2h f_k$$

$y_{k+m} = y_{k+m} + 2h f_k$

Mehrschrittverfahren, allgemeine Form:

$$\sum_{i=0}^{m} \alpha_i y_{k+i} = h \sum_{i=0}^{m} \beta_i f(t_{k+i}, y_{k+i}). \quad \beta_m \neq 0 \text{ explizit.}$$

Achtung: Erst definiert y_m.
Äquidistante Gitter.
Indexverschiebung.

Konsistenz: Wie bei Einschrittverfahren.
Setze $y_k = y_k(t_k) = y(t_k)$ in eine Lösung ein.

$$\tau_k(t, y(t)) = \frac{1}{h}\sum_{i=0}^{m} \alpha_i y(t_{k+i}h) - \sum_{i=0}^{m} \beta_i f(t_{k+i}h, y(t_{k+i}h))$$

konsistenzfehler, Lokale Diskretisierungsfehler.

Satz: Ein Mehrschrittverfahren sei durch Integration entstanden wie oben.
Dann ist ein m-stufiges Verfahren (mindestens) von der Konsistenzordnung m (explizit) bzw. m+1 (implizit).

Beweis: $y_{k+m} = y_{k+m-1} + h\,\varphi(t_k, y_{k+1}, \ldots, y_{k+m-1}, h)$

$$\varphi(t_k, y_k \ldots y_{k+m-1}) = \int_{t_{k+m-1}}^{t_{k+m}} p(s)\,ds \cdot \frac{1}{h}, \; p(t_{k+i}) = f_{k+i}$$

$$\tau_k(t, y(t)) = \frac{1}{h}\left(y(t_{k+m}h) - y(t_{(k+m-1)}h)\right) - \varphi(t, y(t), \ldots y(t_{(k+m-1)}h))$$

$$= \frac{1}{h}\left(\int_{t_{(k+m-1)}h}^{t_{k+m}h} y'(s)\,ds - \int_{t_{(k+m-1)}h}^{t_{k+m}h} p(s)\,ds\right)$$

($y'(s) = f(s, y(s))$; $p(s)$: Interpolationspolynom $= f(t_i, y(t_i))$)

Stützstellen $t_k \ldots t_{k+m-1} \Rightarrow$ Genauigkeit $\mathcal{O}(h^{m+1})$ (Satz 10.3)

$\Rightarrow \mathcal{O}(h^m)$.

Mehrschrittverfahren

Herleitung durch Integration (Polynominterpolation) ¶

Motivation (Adams-Bashforth): Sei y eine Lösung der Differentialgleichung. Dann gilt

$$y(t_{k+m}) - y(t_{k+m-1}) = \int_{t_{k+m-1}}^{t_{k+m}} f(s, y(s))\, ds$$

Diskretisiert:

$$y_{k+m} - y_{k+m-1} = \int p(s)\, ds$$

...und $p(s)$ ist das Interpolationspolynom, das an den Stellen t_k bis t_{k+m-1} die Werte $f_k = f(t_k, y_k)$ bis $f_{k+m-1} = f(t_{k+m-1}, y_{k+m-1})$ annimmt.

In [80]:

```
import numpy as np
import sympy
yk=sympy.symbols('y_k y_{k+1} y_{k+2} y_{k+3}')
fk=sympy.symbols('f_k f_{k+1} f_{k+2} f_{k+3}')
tk,h=sympy.symbols('t_k h')
t=[]
for i in range(len(fk)+2):
    t.append(tk+i*h)
x=sympy.symbols('x')
```

1. Wir konstruieren das Polynom p, das an den Stellen $t_k + jh$ den Wert f_j annimmt mit Lagrange, $j = 0 \ldots m-1$.
2. Wir integrieren p zwischen t_{k+m-1} und t_k und erhalten die Formeln von Adams-Bashforth.
3. Wir integrieren p zwischen t_{k+m-2} und t_k und erhalten die Formeln von Nyström.

$m = 2$ liefert jeweils das Beispiel der Vorlesung. Wir vergleichen jeweils mit den Ergebnissen aus dem Buch von Hairer.

In [88]:

```
m=3
# Erst: Lagrange
p=0
for i in range(m):
    q=1
    for k in range(m):
        if (i!=k):
            q=q*(x-t[k])/(t[i]-t[k])
    p=p+fk[i]*q
#p=p.expand()
#p=p.collect('x')
print('Interpolierendes Polynom')
display(p)
#for i in range(m):
#    display(p.subs(x,t[i]).simplify())
print('Stammfunktion')
P=p.integrate(x)
display(P)
print('Adams-Bashforth')
Q=P.subs(x,t[m])-P.subs(x,t[m-1])
Q=Q.simplify()
display(Q)
print('Nyström')
Q=P.subs(x,t[m])-P.subs(x,t[m-2])
Q=Q.simplify()
display(Q)
```

Interpolierendes Polynom

$$\frac{f_k\left(-2h-t_k+x\right)\left(-h-t_k+x\right)}{2h^2}-\frac{f_{k+1}\left(-t_k+x\right)\left(-2h-t_k+x\right)}{h^2}+\frac{f_{k+2}\left(-t_k\right.}{}$$

Stammfunktion

$$\frac{x^3\left(f_k-2f_{k+1}+f_{k+2}\right)}{6h^2}+\frac{x^2\left(-3f_kh-2f_kt_k+4f_{k+1}h+4f_{k+1}t_k-f_{k+2}h-2f_{k+}\right.}{4h^2}$$
$$+\frac{x\left(2f_kh^2+3f_kht_k+f_kt_k^2-4f_{k+1}ht_k-2f_{k+1}t_k^2+f_{k+2}ht_k+f_{k+2}t_k^2\right)}{2h^2}$$

Adams-Bashforth

$$\frac{h\left(5f_k-16f_{k+1}+23f_{k+2}\right)}{12}$$

Nyström

$$\frac{h\left(f_k-2f_{k+1}+7f_{k+2}\right)}{3}$$

Special cases of (1.5). For $k = 1, 2, 3, 4$, after expressing the backward differences in terms of f_{n-j}, one obtains the formulas

$$
\begin{aligned}
k = 1: &\qquad y_{n+1} = y_n + h f_n && \text{(explicit Euler method)} \\
k = 2: &\qquad y_{n+1} = y_n + h\Big(\frac{3}{2} f_n - \frac{1}{2} f_{n-1}\Big) \\
k = 3: &\qquad y_{n+1} = y_n + h\Big(\frac{23}{12} f_n - \frac{16}{12} f_{n-1} + \frac{5}{12} f_{n-2}\Big) \\
k = 4: &\qquad y_{n+1} = y_n + h\Big(\frac{55}{24} f_n - \frac{59}{24} f_{n-1} + \frac{37}{24} f_{n-2} - \frac{9}{24} f_{n-3}\Big).
\end{aligned}
\tag{1.5'}
$$

Implizite Verfahren gehen genauso, nur nehmen wir diesmal auch den Punkt t_{k+m} in die Polynominterpolation mit auf.

In [83]:

```
m=2
p=0
for i in range(m+1):
    q=1
    for k in range(m+1):
        if (i!=k):
            q=q*(x-t[k])/(t[i]-t[k])
    p=p+fk[i]*q
#p=p.expand()
#p=p.collect('x')
print('Interpolierendes Polynom')
display(p)
#for i in range(m+1):
#    display(p.subs(x,t[i]).simplify())
print('Stammfunktion')
P=p.integrate(x)
display(P)
print('Adams-Moulton')
Q=P.subs(x,t[m])-P.subs(x,t[m-1])
Q=Q.simplify()
display(Q)
print('Milne-Simpson')
Q=P.subs(x,t[m])-P.subs(x,t[m-2])
Q=Q.simplify()
display(Q)
```

Interpolierendes Polynom

$$\frac{f_k\left(-2h-t_k+x\right)\left(-h-t_k+x\right)}{2h^2}-\frac{f_{k+1}\left(-t_k+x\right)\left(-2h-t_k+x\right)}{h^2}+\frac{f_{k+2}\left(-t_k\right.}{}$$

Stammfunktion

$$\frac{x^3\left(f_k-2f_{k+1}+f_{k+2}\right)}{6h^2}+\frac{x^2\left(-3f_kh-2f_kt_k+4f_{k+1}h+4f_{k+1}t_k-f_{k+2}h-2f_{k+}\right.}{4h^2}$$
$$+\frac{x\left(2f_kh^2+3f_kht_k+f_kt_k^2-4f_{k+1}ht_k-2f_{k+1}t_k^2+f_{k+2}ht_k+f_{k+2}t_k^2\right)}{2h^2}$$

Adams-Moulton

$$\frac{h\left(-f_k+8f_{k+1}+5f_{k+2}\right)}{12}$$

Milne-Simpson

$$\frac{h\left(f_k+4f_{k+1}+f_{k+2}\right)}{3}$$

$$
\begin{aligned}
k=0: &\qquad y_{n+1} = y_n + h f_{n+1} = y_n + h f(x_{n+1}, y_{n+1}) \\
k=1: &\qquad y_{n+1} = y_n + h\left(\frac{1}{2} f_{n+1} + \frac{1}{2} f_n\right) \\
k=2: &\qquad y_{n+1} = y_n + h\left(\frac{5}{12} f_{n+1} + \frac{8}{12} f_n - \frac{1}{12} f_{n-1}\right) \\
k=3: &\qquad y_{n+1} = y_n + h\left(\frac{9}{24} f_{n+1} + \frac{19}{24} f_n - \frac{5}{24} f_{n-1} + \frac{1}{24} f_{n-2}\right).
\end{aligned}
$$

In [ ]:

Beispiel: Konsistenz von Mehrschrittverfahren

$$\tau_h(t, y(t)) = \frac{1}{h}\sum_{j=0}^{m} \alpha_j y(t+jh) - \sum_{j=0}^{m} \beta_j \underbrace{f(t+jh, y(t+jh))}_{= y'(t+jh)}$$

$$y_{k+2} - (1+\alpha) y_{k+1} + \alpha y_k = h\left(\frac{3-\alpha}{2} f_{k+1} - \frac{(1+\alpha)}{2} f_k\right)$$

$$\tau_h(t, y(t)) = \frac{1}{h}\left(y(t+2h) - (1+\alpha) y(t+h) + \alpha y(t)\right)$$
$$- \left(\frac{3-\alpha}{2} y'(t+h) - \frac{(1+\alpha)}{2} y'(t)\right)$$

$$= \frac{1}{h}\Big[y(t) + 2h y'(t) + \frac{4h^2}{2} y''(t) + \frac{8h^3}{6} y'''(t) + O(h^4)$$
$$- (1+\alpha)\left[y(t) + h y'(t) + \frac{h^2}{2} y''(t) + \frac{h^3}{6} y'''(t) + O(h^4)\right]$$
$$+ \alpha y(t)\Big]$$
$$- \left[\frac{3-\alpha}{2}\left[y'(t) + h y''(t) + \frac{h^2}{2} y'''(t) + O(h^3)\right]\right.$$
$$\left. - \frac{(1+\alpha)}{2} y'(t)\right]$$

$$= y'(t)\left[2 - (1+\alpha) - \frac{3-\alpha}{2} + \frac{1+\alpha}{2}\right]$$
$\hookrightarrow 0$

$$+ y''(t)\cdot h\left[2 - \frac{(1+\alpha)}{2} - \frac{3-\alpha}{2}\right]$$
$\hookrightarrow 0$

$$+ y'''(t)\cdot h^2\left[\underbrace{\frac{4}{3} - \frac{(1+\alpha)}{6} - \frac{3-\alpha}{4}}_{\frac{5}{12} + \frac{\alpha}{12}}\right] + O(h^3)$$

Ordnung 2, Ordnung 3 für $\alpha = -5$.

$$y_{k+2} - 2y_{k+1} + y_k = 0$$

$$\tau_h(t, y(t)) = \left[y(t+2h) - 2y(t+h) + y(t)\right]\cdot\frac{1}{h}$$
$$= \left[y(t) + 2h y'(t) - 2y(t) - 2y'(t)\cdot h + y(t) + O(h^2)\right]\cdot\frac{1}{h}$$
$$= O(h)$$

konsistent von Ordnung 1.

hängt nicht von den Daten ab

Mehrschrittverfahren Implementation

June 24, 2020

1 Mehrschrittverfahren Implementation

Bei den Mehrschrittverfahren nutzt man die schon berechneten Werte

$$y_k, \ldots, y_{k+m-1}$$

und die Auswertungen

$$f_k = f(t_k, y_k), \ldots, f(t_{k+m-1}, y_{k+m-1}),$$

um y_{k+m} auszurechnen.

Am Anfang stehen diese natürlich gar nicht zur Verfügung, d.h. wir beginnen erstmal mit einem Einschrittverfahren, bis wir $m-1$ Werte zusammenhaben. Dann beginnt das Mehrschrittverfahren.

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import math
```

[3]:
```
# Beispielaufgaben der Vorlesung
def f_linear(t,y):
    return y
a_linear=0
b_linear=1
y0_linear=1
def loesung_linear(t):
    return np.exp(t)

def f_tangens(t,y):
    return 1+y*y
a_tangens=0
b_tangens=1
y0_tangens=0
def loesung_tangens(t):
    return np.tan(t)

def f_aufgabe(t,y):
    return t+t*y
a_aufgabe=0
b_aufgabe=1
```

```
y0_aufgabe=1
def loesung_aufgabe(t):
    return -1+2*np.exp(t*t/2)

f=f_linear
a=a_linear
b=b_linear
y0=y0_linear
loesungsfunc=loesung_linear
```

[4]:
```
# Verfahrensfunktionen für Einschrittverfahren
def phi_euler(f,t,y,h):
    return f(t,y)
def phi_verbesserter_euler(f,t,y,h):
    return f(t+h/2,y+h/2*f(t,y))
def phi_heun(f,t,y,h):
    f1=f(t,y)
    return 1/2*(f1+f(t+h,y+h*f1))
def phi_runge_kutta(f,t,y,h):
    f1=f(t,y)
    f2=f(t+h/2,y+h/2*f1)
    f3=f(t+h/2,y+h/2*f2)
    f4=f(t+h,y+h*f3)
    return 1/6*(f1+2*f2+2*f3+f4)
```

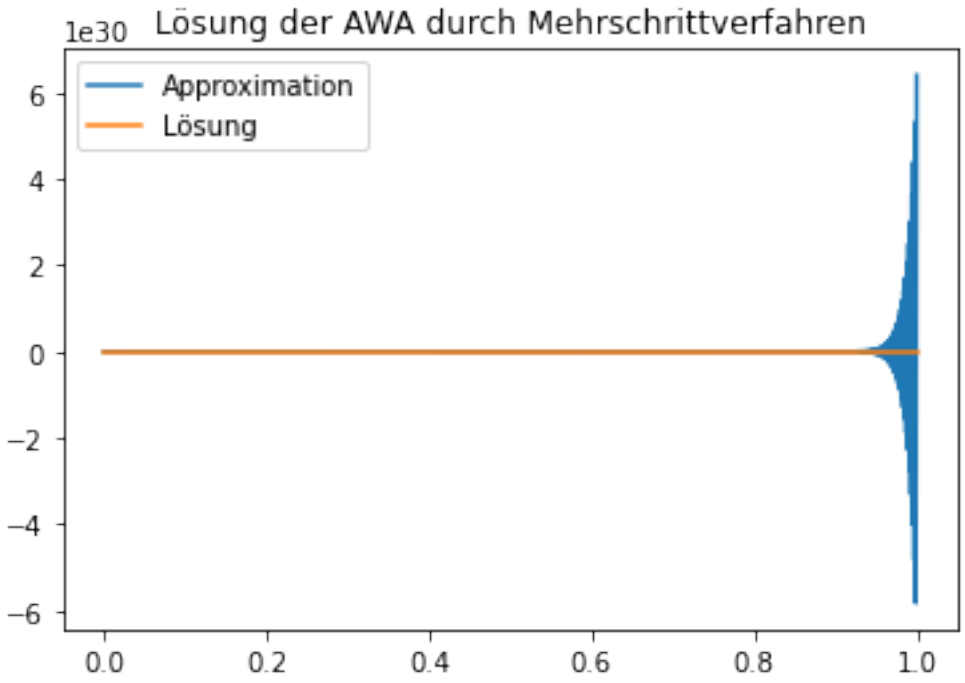
[135]:
```
# Koeffizienten für Mehrschrittverfahren
# Adams-Bashforth nach Vorlesung
alpha_ab=[0,-1,1]
beta_ab=[-1/2,3/2,0]

beta_ab3=[5/12,-16/12,23/12]
alpha_ab3=[0,0,-1,1]

alpha=-1.1
alpha_instab=[alpha,-(1+alpha),1]
beta_instab=[-(1+alpha)/2,(3-alpha)/2,0]

alpha=alpha_instab
beta=beta_instab
```

[136]:
```
def mehrschritt(f,I_h,y0,phi_einschritt,alpha,beta):
    N=len(I_h)
    y_h=np.zeros(N)
    f_h=np.zeros(N)
    h=I_h[1]-I_h[0]
    m=len(alpha)-1
    print('Stufe: ',m)
```

```
    y_h[0]=y0
    f_h[0]=f(I_h[0],y0)
    # Die ersten Werte mit Einschrittverfahren
    for i in range(m-1):
        y_h[i+1]=y_h[i]+h*phi_einschritt(f,I_h[i],y_h[i],h)
        f_h[i+1]=f(I_h[i+1],y_h[i+1])
    # Jetzt mit Mehrschrittverfahren (explizit)
    for i in range(0,N-m):
        # print(np.sum(beta[0:m]*f_h[i:i+m]))
        y_h[i+m]=(h*np.sum(beta[0:m]*f_h[i:i+m])-np.sum(alpha[0:m]*y_h[i:i+m]))/
 ↪alpha[m]
        # print(y_h[i+m])
        f_h[i+m]=f(I_h[i+m],y_h[i+m])
    return y_h

N=1000
I_h=np.linspace(a,b,N)
print('alpha: ',alpha)
print('beta: ',beta)
y_h=mehrschritt(f,I_h,y0,phi_runge_kutta,alpha,beta)
plt.plot(I_h,y_h,I_h,loesungsfunc(I_h))
plt.legend(['Approximation','Lösung'])
print('Globaler Diskretisierungsfehler: ',np.linalg.
 ↪norm(y_h-loesungsfunc(I_h),math.inf))
plt.title('Lösung der AWA durch Mehrschrittverfahren')
```

```
alpha:  [-1.1, 0.10000000000000009, 1]
beta:  [0.050000000000000044, 2.05, 0]
Stufe:  2
Globaler Diskretisierungsfehler:  6.396588507524683e+30
```

[136]: Text(0.5, 1.0, 'Lösung der AWA durch Mehrschrittverfahren')


Lösung der AWA durch Mehrschrittverfahren
1e30
Approximation
Lösung
6
4
2
0
−2
−4
−6
0.0
0.2
0.4
0.6
0.8
1.0

Stabilität von Mehrschrittverfahren für Anfangswertprobleme

① $y'(t) = f(t, y(t)),\ y(a) = y(0)$ auf $[a,b]$, $y: [a,b] \to \mathbb{C}$.

② Äquidistante Gitter $I_h = \{t_0 = a, \dots t_N = b\}$,
$h = \frac{(b-a)}{N}$

③ Approximationen $y_k = y_h(t_k)$ von $y(t_k)$,
$y_h : I_h \to \mathbb{C}$.

④ Globale Diskretisierungsfehler:
$e_h := \| y|_{I_h} - y_h \|, \quad \|e_h\|_\infty$

⑤ Konvergenz:
$\|e_h\|_\infty \to 0$ (von der Ordnung p)
$h \to 0$

⑥ Konsistenz:
Lokal entstehende Fehler — 2k Iterationen

Mehrschrittverfahren:

- Berechnet y_{k+m}
- $\sum_{j=0}^{m} \alpha_j y_{k+j} = h \sum_{j=0}^{m} \beta_j f(t_{k+j}, y_{k+j})$ / $y_{k+2} - y_k$ = 2 Iterationen
  Stufe 2
  $\alpha_m \neq 0$, $\beta_m = 0$: explizit
  → Anfangswerte $y_0 \dots y_{m-1}$.
- Konsistenzfehler:
  $\tau_h(t, y(t)) = \frac{1}{h} \sum_{j=0}^{m} \alpha_j y(t+jh) - \sum_{j=0}^{m} \beta_j y'(t+jh)$

Entscheidend: Wie stark wirkt sich der Fehler, der bei der Berechnung von y_k macht, auf das Ergebnis aus?

Ohne Beweis: Es reicht sich auf die Auswirkung der Fehler in den Anfangswerte bei der AWA $y'(t) = 0, y(a) = 0$ zu beschränken

Ansatz: Betrachte Folge von Gitter I_h
- Fehler der „Anfangswerte" $(\gamma_k)_{k=0,\dots,m}$ gegen 0
- Gilt der globale Diskretisierungsfehler gegen Null?

Motivation / Beweisübersicht

$\sum_{j=0}^{m} \alpha_j y_{k+j} = 0 \Rightarrow y_{k+m} = -\frac{1}{\alpha_m} \sum_{j=0}^{m-1} y_{k+j}$

→ Abhängigkeit von h (Bedeutung ist anders), festgelegt durch Anfangswerte

Fibonacci-Folge: (Beispiel, nicht konsistent)
$y_k - y_{k+1} + y_{k+2} = 0 \quad (\alpha_m = 1)$
- Festgelegt durch y_0, y_1.
- Sei $y^{(0)}$ die Folge mit Anfang $(1, 0)$
  $y^{(1)}$ " " " $(0, 1)$
- $y_k = ((y_k)_0, (y_k)_1, \dots)$
  Anfangswerte

$y_k = (y_k)_0 \cdot y^{(0)} + (y_k)_1 \cdot y^{(1)}$

Angenommen $(y_k)_0, (y_k)_1 \to_{k \to \infty} 0$,
$y_k \to$ Nullfolge genau dann, wenn $y^{(0)}, y^{(1)}$ beschränkt sind.

Ergebnis:
Der globale Diskretisierungsfehler geht genau dann gegen Null, wenn die Lösung der Differenzengleichung
$\sum_{j=0}^{m} \alpha_j y_{k+j} = 0$
beschränkt sind für alle Anfangswerte.

Differenzengleichung

→ Untersuche die Folge y, die der Gleichung $\sum_{i=0}^{l+m} a_i y_{n+i} = 0 \quad \forall k \geq 0$ genügt, $a_m = 1$.

→ Sei $y^{(k)}$ die Folge, die beginnt mit $(0, \ldots, 0, \underset{k\text{-Position}}{1}, 0, \ldots)$, $k = 0 \ldots m-1$.

→ $\sum_{k=0}^{m-1} \gamma_k y^{(k)}$ stimmt in den Positionen $0 \ldots m-1$ mit y überein.

→ $\Rightarrow$ ist Lösung der Differenzengleichung

→ $\Rightarrow \dim = m$

→ Die Lösung y der Differenzengleichung bildet einen m-dimensionalen Unterraum der Menge aller Folgen.

(Verallgemeinerung der Fibonacci-Basis-Idee)

Idee: Berechne eine Basis dieses Unterraums nicht-rekursiv.

Ansatz $y_i = x^i$, $x \neq 0$. Idee: $y_i = x^i$.

$\Rightarrow \sum_{i=0}^{l+m} a_i x^i = 0$

$\Rightarrow$ „charakteristisches Polynom"

Sei $p(x) = 0$, $y_i = x^i$ ($x \neq 0$).

$p(x) = 0 \Rightarrow x^k p(x) = 0 \Rightarrow \sum_{i=0}^{l+m} a_i x^{k+i} = 0 \Rightarrow \sum_{i=0}^{l+m} a_i y_{k+i} = 0 \Rightarrow y$ erfüllt Differenzengl.

$\Rightarrow$ x-fache Nullstelle von p liefert x linear unabhängige Lösungen der Differenzengl., i.a. noch Fundamentalsystem.

Sei x doppelte Nullstelle von p.

$\Rightarrow p(x) = 0, p'(x) = 0 \Rightarrow \left(x^k p(x)\right)' = 0$

$0 = \sum_{i=0}^{l+m} a_i (k+i) x^{k+i-1} = \sum_{i=0}^{l+m} a_i x^{k+i} + \sum_{i=0}^{l+m} a_i i x^{k+i-1}$

$= \sum_{i=0}^{l+m} a_i (\widetilde{y})_{k+i} + \widetilde{\gamma} x \cdot x^k$, $\quad \widetilde{\gamma} x = 0$

$\widetilde{y}$ ist Lösung der Differenzengleichung (und l.u.)

$\Rightarrow$ zu einer doppelten Nullstelle bekommt man zwei Basislösungen der Differenzengl.: $x^k, k \cdot x^k$

$\Rightarrow$ zu einer r-fachen Nullstelle bekommt man $x^k, k \cdot x^k, \ldots, k^{r-1} \cdot x^k$

Satz: Seien x_j die Nullstellen des charakteristischen Polynoms $p(x) = \sum_{i=0}^{l+m} a_i x^i$ der Differenzengleichung

$$\sum_{i=0}^{l+m} a_i y_{n+i} = 0$$

mit zugehöriger Vielfachheit σ_j.

Dann ist eine Basis des Vektorraums der Lösungen der Differenzengleichung als Unterraum des Vektorraums aller komplexen Folgen gegeben durch

$$\left(\left(y^{(k,j)}\right)_n = j^n \cdot n^k \;\middle|\; r = 0 \ldots \sigma_j - 1\right).$$

Beweis: $\sum \sigma_j = m$.

Beispiel: Fibonacci

$-y_k - y_{k+1} + y_{k+2} = 0$

$p(x) = x^2 - x - 1$

$x_{1/2} = +\frac{1}{2} \pm \sqrt{\frac{1}{4} + 1} = \frac{1 \pm \sqrt{5}}{2}$, $\quad \gamma = \frac{1+\sqrt{5}}{2}$ (= Goldener Schnitt).

„Standard-Fibonacci": $(0, 1, \ldots) \to y$

$y = \lambda_1 (x_1)^k + \lambda_2 (x_2)^k$

$0 \cdot 0 = \lambda_1 + \lambda_2$

$1 = y_1 = \lambda_1 \frac{1+\sqrt{5}}{2} - \lambda_1 \frac{1-\sqrt{5}}{2} = \lambda_1 \sqrt{5}$.

$\Rightarrow y_k = \frac{1}{\sqrt{5}} \left(\left(\frac{1+\sqrt{5}}{2}\right)^k - \left(\frac{1-\sqrt{5}}{2}\right)^k \right)$

nicht-rekursive Darstellung der Fibonacci-Zahlen.

Beispiel II:

$y_{k+2} - 2y_{k+1} + y_k = 0$

$p(x) = x^2 - 2x + 1$

$x = 1$ einfache (doppelte) Nullstelle.

Lösungen: $y_k = 1^k$

$y_k = k \cdot 1^k = k$

konvolver: Die Basislösungen der Differenzengleichung sind linear unabhängig, wenn

1) $p(x) = 0$ in x ist

2) $p(x) = 0$, $\ln x = 0$ Vielfachheit von x als Nullstelle von p $(x \neq 0)$.

Eine Differenzengleichung bei der Stabilität wegen ihrer Basislösungen beobachtet sind.

→ Wurzelbedingung von Dahlquist

1. Fibonacci ist instabil: $\frac{1+\sqrt{5}}{2} > 1$.

2. Beispiel II ist instabil: $x = 1$ hat Vielf. 2.

Falls für ein Modellproblem gilt, dass die zugehörige Differenzengleichung instabil ist, so gilt:

Aus konsistente folgt keine gute

Stabilität von Mehrschrittverfahren: Beispiele

1) Einschrittverfahren:

$$y_{k+1} - y_k = h\varphi$$

char. Polynom ρ: $\rho(x) = x - 1$, NST $x = 1$, einfach

Dahlquist $\Rightarrow$ stabil, da kon. $\Rightarrow$ konv.

2) Aus Integration gewonnene Mehrschrittverfahren:

$$y_{k+m} - y_{k+m-1} = \ldots$$

$$\rho(x) = x^m - x^{m-1} \qquad \Rightarrow \text{stabil mit Dahlquist}$$

$$= x^{m-r}\,(x^r - 1)$$

x^{m-r}: $=0$ — $(x^r - 1)$: Einheitswurzeln, $\sigma = 1$

3) $y_{k+2} - 2y_{k+1} + y_k = 0$:

$\rho(\lambda) = x^2 - 2x + 1 \Rightarrow$ instabil nach Dahlquist

4) Beispiel mit α:

$$y_{k+2} - (1+\alpha)\,y_{k+1} + \alpha\, y_k = 0$$

$$\rho(x) = x^2 - (1+\alpha)x + \alpha$$

NST $1, \alpha$

1) $|\alpha| \le 1$

2) $\alpha = 1$: Vfh 2 $\rightarrow$ ↯.

$\alpha = -1$: ✓

Mehrschrittverfahren Implementation

June 24, 2020

1 Mehrschrittverfahren Implementation

Bei den Mehrschrittverfahren nutzt man die schon berechneten Werte

$$y_k, \ldots, y_{k+m-1}$$

und die Auswertungen

$$f_k = f(t_k, y_k), \ldots, f(t_{k+m-1}, y_{k+m-1}),$$

um y_{k+m} auszurechnen.

Am Anfang stehen diese natürlich gar nicht zur Verfügung, d.h. wir beginnen erstmal mit einem Einschrittverfahren, bis wir $m-1$ Werte zusammenhaben. Dann beginnt das Mehrschrittverfahren.

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import math
```

[3]:
```
# Beispielaufgaben der Vorlesung
def f_linear(t,y):
    return y
a_linear=0
b_linear=1
y0_linear=1
def loesung_linear(t):
    return np.exp(t)

def f_tangens(t,y):
    return 1+y*y
a_tangens=0
b_tangens=1
y0_tangens=0
def loesung_tangens(t):
    return np.tan(t)

def f_aufgabe(t,y):
    return t+t*y
a_aufgabe=0
b_aufgabe=1
```

```
y0_aufgabe=1
def loesung_aufgabe(t):
    return -1+2*np.exp(t*t/2)

f=f_linear
a=a_linear
b=b_linear
y0=y0_linear
loesungsfunc=loesung_linear
```

[4]:
```
# Verfahrensfunktionen für Einschrittverfahren
def phi_euler(f,t,y,h):
    return f(t,y)
def phi_verbesserter_euler(f,t,y,h):
    return f(t+h/2,y+h/2*f(t,y))
def phi_heun(f,t,y,h):
    f1=f(t,y)
    return 1/2*(f1+f(t+h,y+h*f1))
def phi_runge_kutta(f,t,y,h):
    f1=f(t,y)
    f2=f(t+h/2,y+h/2*f1)
    f3=f(t+h/2,y+h/2*f2)
    f4=f(t+h,y+h*f3)
    return 1/6*(f1+2*f2+2*f3+f4)
```

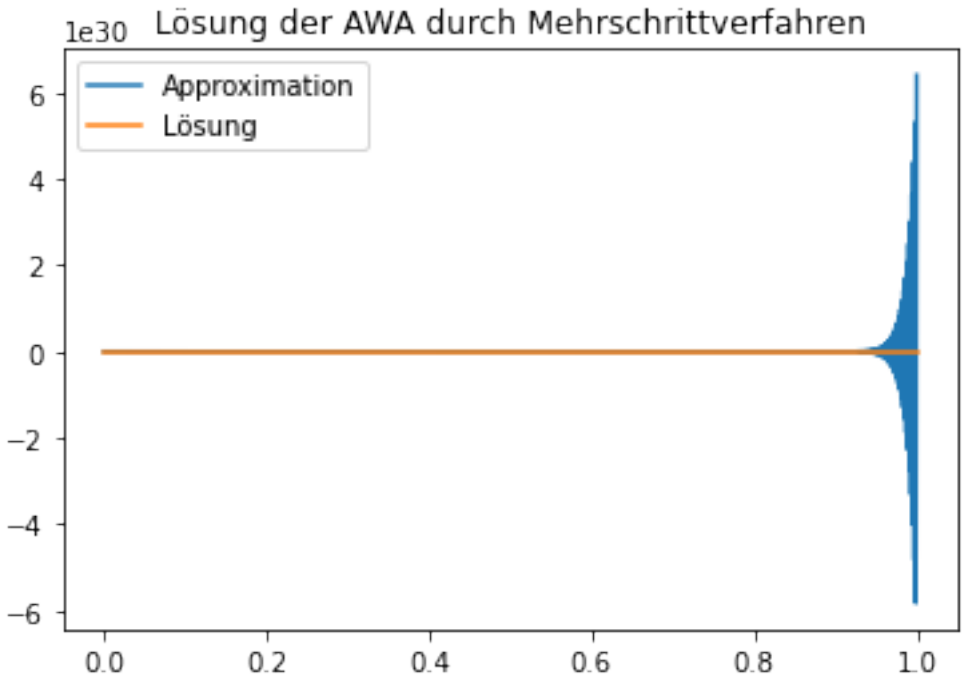
[135]:
```
# Koeffizienten für Mehrschrittverfahren
# Adams-Bashforth nach Vorlesung
alpha_ab=[0,-1,1]
beta_ab=[-1/2,3/2,0]

beta_ab3=[5/12,-16/12,23/12]
alpha_ab3=[0,0,-1,1]

alpha=-1.1
alpha_instab=[alpha,-(1+alpha),1]
beta_instab=[-(1+alpha)/2,(3-alpha)/2,0]

alpha=alpha_instab
beta=beta_instab
```

[136]:
```
def mehrschritt(f,I_h,y0,phi_einschritt,alpha,beta):
    N=len(I_h)
    y_h=np.zeros(N)
    f_h=np.zeros(N)
    h=I_h[1]-I_h[0]
    m=len(alpha)-1
    print('Stufe: ',m)
```

```
    y_h[0]=y0
    f_h[0]=f(I_h[0],y0)
    # Die ersten Werte mit Einschrittverfahren
    for i in range(m-1):
        y_h[i+1]=y_h[i]+h*phi_einschritt(f,I_h[i],y_h[i],h)
        f_h[i+1]=f(I_h[i+1],y_h[i+1])
    # Jetzt mit Mehrschrittverfahren (explizit)
    for i in range(0,N-m):
        # print(np.sum(beta[0:m]*f_h[i:i+m]))
        y_h[i+m]=(h*np.sum(beta[0:m]*f_h[i:i+m])-np.sum(alpha[0:m]*y_h[i:i+m]))/
 ↪alpha[m]
        # print(y_h[i+m])
        f_h[i+m]=f(I_h[i+m],y_h[i+m])
    return y_h

N=1000
I_h=np.linspace(a,b,N)
print('alpha: ',alpha)
print('beta: ',beta)
y_h=mehrschritt(f,I_h,y0,phi_runge_kutta,alpha,beta)
plt.plot(I_h,y_h,I_h,loesungsfunc(I_h))
plt.legend(['Approximation','Lösung'])
print('Globaler Diskretisierungsfehler: ',np.linalg.
 ↪norm(y_h-loesungsfunc(I_h),math.inf))
plt.title('Lösung der AWA durch Mehrschrittverfahren')
```

```
alpha:  [-1.1, 0.10000000000000009, 1]
beta:  [0.050000000000000044, 2.05, 0]
Stufe:  2
Globaler Diskretisierungsfehler:  6.396588507524683e+30
```

[136]: Text(0.5, 1.0, 'Lösung der AWA durch Mehrschrittverfahren')


Lösung der AWA durch Mehrschrittverfahren
1e30
Approximation
Lösung
6
4
2
0
−2
−4
−6
0.0
0.2
0.4
0.6
0.8
1.0

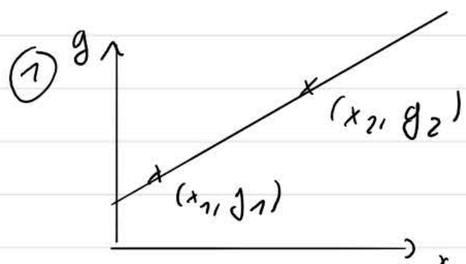
Lösung über- und unterbestimmter Gleichungssysteme

Aufgabe: Löse $Ax = b$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $x \in \mathbb{R}^n$

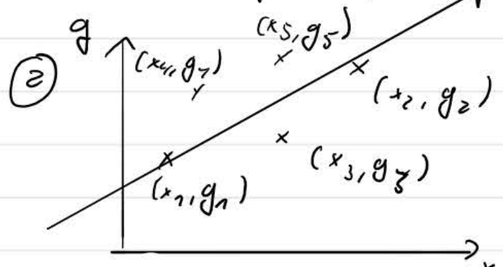
Motivation: Es sei bekannt, dass für die Position g einer Feder bei Belastung x gilt

$$g(x) = s x + b$$

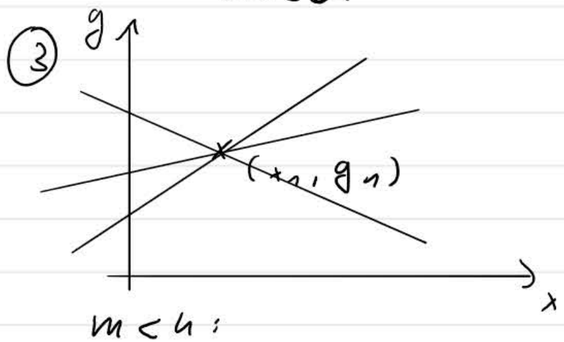
s, b sollen bestimmt werden. m = # Messpunkte, $n = 2$.



$m = n$: $s \cdot x_1 + b = g_1$
$s \cdot x_2 + b = g_2$
$x_1 \neq x_2 \Rightarrow$ eindeutig lösbar.



$m > n$: $s \cdot x_1 + b = g_1$
$s \cdot x_2 + b = g_2$
$\vdots$
$s \cdot x_5 + b = g_5$
$\Rightarrow$ i.a. nicht lösbar „überbestimmt",



$m < n$:
$s \cdot x_1 + b = g_1$
i.a. ∞ viele Lösungen (hier: immer) „unterbestimmt"

Bemerkung: Dies geht auch für Polynome vom Grad n.

Gauss-Elimination

$$\begin{array}{rrrr|l} 3x_1 & +2x_2 & +x_3 & = 8 & \cdot \\ 6x_1 & +5x_2 & -4x_3 & = 12 & l_{21} = -\frac{6}{3} + l_{21} \cdot \text{I} \\ -3x_1 & +x_2 & -2x_3 & = -3 & l_{31} = +\frac{3}{3} + l_{31} \cdot \text{I} \end{array} \qquad A^{(1)} x = b^{(1)}$$

$$\begin{array}{rrrr|l} 3x_1 & +2x_2 & +x_3 & = 8 & \cdot \\ & x_2 & -6x_3 & = -4 & \cdot \\ & 3x_2 & -x_3 & = 5 & l_{32} = -\frac{3}{1} + l_{32} \cdot \text{II} \end{array} \qquad A^{(2)} x = b^{(2)}$$

$$\begin{array}{rrrr|l} 3x_1 & +2x_2 & +x_3 & = 8 & \cdot \\ & x_2 & -6x_3 & = -4 & \cdot \\ & & 17x_3 & = 17 & \cdot \end{array} \qquad A^{(3)} x = b^{(3)}$$

Rückwärtseinsetzen:

$x_3 = \frac{17}{17} = 1$, $x_2 = -4 + 6x_3 = 2$, $x_1 = \frac{1}{3}(8 - 2x_2 - x_3) = 1$.

Zur Lösg von $Ax = b$:

$A^{(1)} = A$, $b^{(1)} = b$.

Für $i = 1 .. n-1$:

// konstruiere $A^{(i+1)}, b^{(i+1)}$ gegeben $A^{(i)}, b^{(i)}$.

Übernehme die ersten i Zeilen von $A^{(i)}, b^{(i)}$ in $A^{(i+1)}, b^{(i+1)}$

// konstruiere die Zeilen $i+1 .. n$

Für $j = i+1 .. n$

$l_{ji} = (A^{(i)})_{j,i} / (A^{(i)})_{i,i}$ // Faktor bei Subtrahieren

$(A^{(i+1)})_{j,*} = (A^{(i)})_{j,*} + l_{ji} (A^{(i)})_{i,*}$ // i Nullen am Anfang

$(b^{(i+1)})_j = (b^{(i)})_j + l_{ji} (b^{(i)})_i$

Für $i = n .. 1$:

$$x_i = \left[(b^{(n)})_i - \sum_{j=i+1}^{n} (A^{(n)})_{i,j} \cdot x_j \right] / (A^{(n)})_{ii}$$

konstruktionsaufwand:

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \underbrace{\left(1 + (n-i) + 1 \right)}_{\substack{\uparrow \\ j_i}} = \frac{1}{6} n (n-1)(2n+5) \quad + n \text{ Divisionen}$$

Zeile $b_j^{(i+1)}$

Rechenoperationen $\hat{=}$ MP+Add:

$\frac{n^3}{3} + \frac{n^2}{2} - \frac{5}{6} n$ für große n: $\sim \frac{n^3}{3} + \mathcal{O}(n^2)$

Landau: $f(n) = \mathcal{O}(n^q) \quad \Leftrightarrow |f(n)| \leq C \cdot n^q$, für $n \to \infty$.

$f(h) = \mathcal{O}(h^q) \quad \Leftrightarrow |f(h)| \leq C \cdot h^q$, für $h \to 0$.

Beispiel: $\frac{n^2}{2} - \frac{5}{6} n = \mathcal{O}(n^2)$

$\sin(n) = \mathcal{O}(1)$

$h^2 + \frac{h^3}{3} = \mathcal{O}(h^2)$

Bemerkungen:

① Cramersche Regel: $n!$ Rechenoperationen mit dem Entwicklungssatz

② Durchführbar, falls $(A^{(i)})_{i,i} \neq 0$.

③ Falls $(A^{(i)})_{i,i} = 0$, $(A^{(i)})_{i,k} \neq 0$, $k > i$: vertausche Gleichung i und k.

④ Falls $(A^{(i)})_{i,k} = 0 \ \forall k \geq i$:
$\Rightarrow \det A^{(i)} = \det A = 0$
x_i ist schon eliminiert ▷

⑤ $x_1 = \frac{1}{(A^{(1)})_{1,1}} \left(b_1^{(1)} - \underbrace{\sum_{k=2}^{n} (A^{(1)})_{1,k} \cdot x_k}_{= (A^{(1)})_{1,1} \cdot x_1} \right)$ Auslöschung ▷

Wähle $|(A^{(i)})_{i,i}|$ möglichst groß durch Vertauschen der Gleichungen.

Beispiel $p = 2$:

$$\begin{array}{rr} 10^{-4} x_1 + x_2 = 1 + 10^{-4} \\ x_1 + x_2 = 2 \end{array} \Rightarrow \begin{array}{rr} 10^{-4} x_1 + x_2 = 1 \\ -10^{-4} x_2 = -10^{-4} \end{array}$$

$\Rightarrow x_2 = 1, x_1 = 0$

$$\begin{array}{rr} x_1 + x_2 = 2 \\ 10^{-4} x_1 + x_2 = 1 + 10^{-4} \end{array} \Rightarrow \begin{array}{rr} x_1 + x_2 = 2 \\ x_2 = 1 \end{array}$$

$\Rightarrow x_2 = 1, x_1 = 1$

$\Rightarrow$ Anordnung der Zeilen ist wichtig ▷

Satz: Für alle Matrizen $A \in \mathbb{R}^{n \times n}$ gibt es
eine normierte (links) untere Dreiecksmatrix L,
eine rechte obere Dreiecksmatrix R,
und eine Permutationsmatrix P so dass

$$PA = LR \quad (P, L, R \in \mathbb{R}^{n \times n}).$$

① $Ax = b \Rightarrow PAx = Pb$

$\Rightarrow L(\underbrace{Rx}_{y}) = Pb$

$\Rightarrow Ly = Pb, \; Rx = y$

② kann man auf P verzichten?

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

③ Auch singuläre Matrizen haben LR-Zerlegung.

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

Aufwand zur Herstellung: $\frac{n^3}{3} + \mathcal{O}(n^2)$ Rechenoperationen

④ Sei A invertierbar,

$$A = L_1 R_1 = L_2 R_2.$$

$$\Rightarrow (L_2)^{-1} L_1 = R_2 (R_1)^{-1} =: D$$

Lemma: Dreiecksmatrizen bilden einen Ring,
Produkte von normierten Matrizen
sind normiert.

$\Rightarrow D$ ist l.u. Δ-Matrix,
r.o. Δ-Matrix, $\Rightarrow D = I$
normiert.

$\Rightarrow L_1 = L_2, \; R_1 = R_2.$

$$y'(t) = y(t) \qquad y(t) = e^t$$

$$y(t) = C \cdot e^t$$

$$y(a) = y_0$$

$$y'(t) = f(t, y(t)), \; y(a) = y_0$$

$$\left.\begin{aligned} H'(t) &= f_1(t, H(t), F(t)) & H(a) &= y_0 \\ F'(t) &= f_2(t, H(t), F(t)) & F(a) &= y_1 \end{aligned}\right\} \text{System}$$

$$y''(t) = f(t, y(t)) \qquad z = y'$$

$$z'(t) = f(t, y(t), z(t))$$

$$y'(t) = z(t)$$

$$y'(t) = y(t) \qquad y(t) = e^t$$

$$y(t) = C \cdot e^t$$

$$y(a) = y_0$$

$$y'(t) = f(t, y(t)), \; y(a) = y_0$$

$$\left.\begin{aligned} H'(t) &= f_1(t, H(t), F(t)) & H(a) &= y_0 \\ F'(t) &= f_2(t, H(t), F(t)) & F(a) &= y_1 \end{aligned}\right\} \text{System}$$

$$y''(t) = f(t, y(t)) \qquad z = y'$$

$$z'(t) = f(t, y(t), z(t))$$

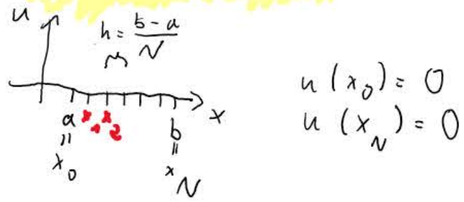
$$y'(t) = z(t)$$

Diskrete Lösg d. Wärmeleitgs gleichg

→ Suche $u : [a,b] \to \mathbb{R}$

$-u''(x) = s(x)$

$u(a) = u(b) = 0$



$u(x_0) = 0$

$u(x_N) = 0$

Bereits gezeigt:

$$\lim_{h \to 0} \frac{u(x+h) - 2u(x) + u(x-h)}{h^2} \to u''(x)$$

(Taylor, l'Hospital)

$x_0 = a$, $x_1 = a+h$, $x_2 = a+2h$. Suche $u_k : u(x_k) \sim u_k$

$s(x_1) = -u''(x_1) \sim \frac{1}{h^2}(-u(x_2) + 2u(x_1) - \underbrace{u(x_0)}_{u_0})$

$\sim \frac{1}{h^2}(-u_2 + 2u_1) \Rightarrow s(x_1) \sim \frac{1}{h^2}(-u_2 + 2u_1)$ ($\uparrow n = 1$)

$\Rightarrow$ lineare Gleichg in u_1, u_2 ▷ 0

$u''(x_2)$: lineare Gleichg in u_1, u_2, u_3 ▷ 0

$u''(x_{N-1})$: " in u_{N-2}, u_{N-1} ▷ 0

$$s(x_k) = \frac{1}{h^2}(-u_{k+1} + 2u_k - u_{k-1}), \quad k = 1, \dots, N-1.$$

LGS in $u_1, \dots, u_{N-1}$

$N-1$ Gleichungen, $N-1$ Unbekannte

$$\begin{pmatrix} 2 & -1 & & 0 \\ -1 & 2 & -1 & \\ & \ddots & \ddots & \ddots \\ 0 & & -1 & 2 \end{pmatrix} \begin{pmatrix} u_1 \\ \vdots \\ u_{N-1} \end{pmatrix} = \begin{pmatrix} s(x_1) \\ \vdots \\ s(x_{N-1}) \end{pmatrix} \cdot h^2$$

$$A u = s$$

$x \in \mathbb{R}^n$, $\tilde{x} = x + dx$, $\tilde{x}, dx \in \mathbb{R}^n$. $\Rightarrow$ Fehler in Eingangs-Daten

$y = f(x)$, $\tilde{y} = f(\tilde{x})$ $\Rightarrow$ Fehler im Ergebnis

Sei $\|\cdot\|$ eine Norm.

$\|dx\| \rightarrow$ absoluter Fehler

$\frac{\|dx\|}{\|x\|} \rightarrow$ relativer Fehler

$\frac{\|dx\|}{\|\tilde{x}\|} \sim \frac{\|dx\|}{\|x\|}$, falls $\|dx\|$ klein.

induzierte 2-Norm einer Matrix

Seien $x, y \in \mathbb{C}^n$, $A \in \mathbb{C}^{n \times n}$, $(\cdot,\cdot)$ Standard-Skalarprodukt.

$\|x\|_2^2 = (x,x) = x^t \overline{x}$, $(x,y) = x^t \overline{y}$.

$(x, Ay) = x^t \overline{(Ay)} = x^t \overline{A}\,\overline{y} = (\overline{A}^t x)^t \cdot \overline{y} = (\underbrace{\overline{A}^t}_{=:A^*} x, y)$. Ansatz: $A^* = A^t$

A^*A ist positiv semidefinit

$\Rightarrow \exists$ ONB aus Eigenvektoren v_k zu Eigenwerten $\lambda_k \geq 0$, $|\lambda_1| \geq |\lambda_2| \geq \dots \geq \lambda_n|$

$\{v_k\}$ ist Basis $\Rightarrow \exists \{\alpha_k\} : x = \sum_k \alpha_k v_k$.

$\exists \{\beta_j\} : y = \sum_j \beta_j v_j$

$\Rightarrow (x,y) = \left(\sum_k \alpha_k v_k, \sum_j \beta_j v_j \right)$

$= \sum_{k,j} \alpha_k \overline{\beta_j} \underbrace{(v_k, v_j)}_{\delta_{k,j}} = \sum_k \alpha_k \overline{\beta_k}$

Insb.: $\|x\|_2^2 = (x,x) = \sum_k |\alpha_k|^2$

$\|Ax\|_2^2 = (Ax, Ax)$

$= (A^*Ax, x)$

$= \left(A^*A \left(\sum_k \alpha_k v_k\right), \sum_k \alpha_k v_k\right)$

$= \left(\sum_k \lambda_k \alpha_k v_k, \sum_k \alpha_k v_k \right)$

$= \sum_k \lambda_k |\alpha_k|^2$

Def.: Sei $B \in \mathbb{C}^{m \times m}$.

$\rho(B) := \max \{ |\lambda| : \lambda \text{ Eigenwert von } B \}$

heißt Spektralradius von B.

Behauptung: $\|A\|_2 = \rho(A^*A)^{1/2}$.

Beweis: Seien $A \in \mathbb{C}^{n \times m}$, $x \in \mathbb{C}^m$.

Sei $\{v_k\}$ eine ONB aus Eigenvektoren von A^*A zu Eigenwerten λ_k.

Sei $x = \sum_k \alpha_k v_k$ (also alles wie oben).

$\|Ax\|_2^2 = \sum_k \lambda_k |\alpha_k|^2 \leq \lambda_1 \sum_k |\alpha_k|^2$ (λ_1 größter EW)

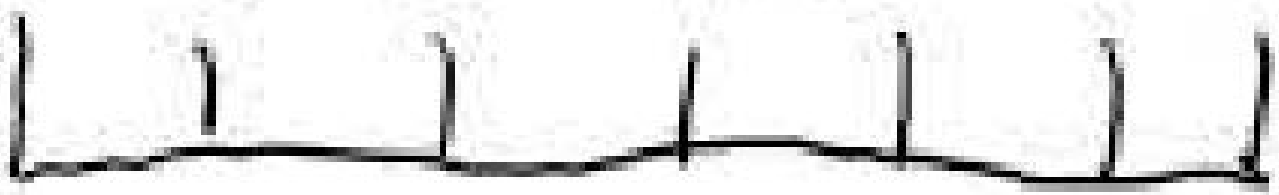
$= \rho(A^*A) \cdot \|x\|_2^2$

$\|Av_1\|_2^2 = (A^*Av_1, v_1) = \lambda_1 (v_1, v_1) = \rho(A^*A) \|v_1\|_2^2$

Also: $\|Ax\|_2 \leq \rho(A^*A)^{1/2} \|x\|$

$\|Av_1\|_2 = \rho(A^*A)^{1/2} \|v_1\|$ $\Rightarrow \|A\|_2 = \rho(A^*A)$.

$\Delta t = \frac{b - a}{n}$



t_0 t_1 t_2 t_n

Lösung von Systemen von GDGL
mit dem Eulerverfahren

Gesucht: $y: [a,b] \to \mathbb{R}^n$,
$y(t) = (y^{(1)}(t), \dots, y^{(n)}(t))$.
$y'(t) = f(t, y(t))$. $f: [a,b] \times \mathbb{R}^n \to \mathbb{R}^n$, $f(t, y(t)) = (f_1(t, y(t)), \dots)$
$y'(t) = (y^{(1)'}(t), \dots, y^{(n)'}(t))$, $y': (a,b) \to \mathbb{R}^n$.

AWA: $y(a) = y_0$, $y_0 \in \mathbb{R}^n$.

Idee des Eulerverfahrens:

Sei $y(t)$ bekannt. Approximiere $y(t+dt)$.

Taylor: $y^{(i)}(t+dt) = y^{(i)}(t) + y^{(i)'}(t) \cdot dt + \dots$

$= f_i(t, y(t))$ — Restglieder

$\to$ berücksichtigen wir nicht

$\sim y^{(i)}(t) + f^{(i)}(t, y(t)) \cdot dt$

$\Rightarrow y(t_1) = y(t_0 + dt)$

$\sim y(t_0) + dt \cdot f(t_0, y(t_0)) =: y_1$

(mit $y(t_0) \sim y_0$, $t_0 = a$, $y(t_0) = a$)

Also: $y(t_1) \sim y_1$.

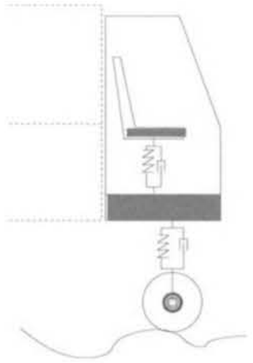
Dann: $y(t_2) = y(t_1 + dt) \sim y(t_1) + dt \cdot f(t_1, y(t_1))$

$\sim y_1 + dt \cdot f(t_1, y_1) =: y_2$

Insgesamt:

$y_{k+1} = y_k + dt \cdot f(t_k, y_k) \sim y(t_{k+1})$

Abbildung 2.1: Fahrerkabine



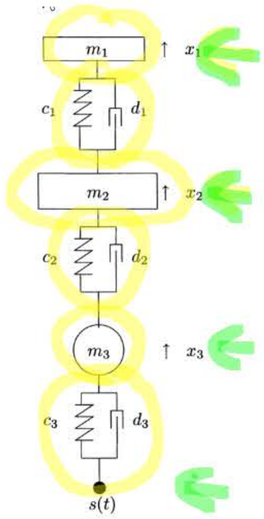
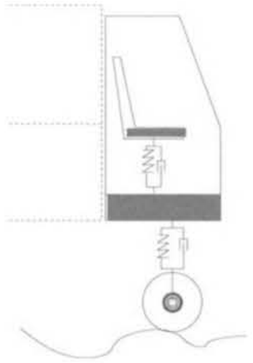
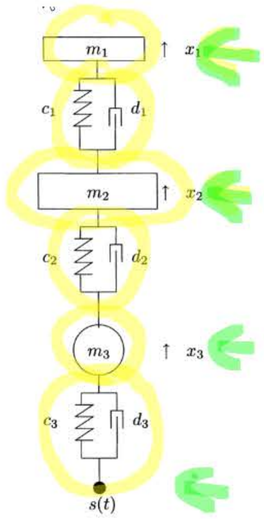


Abbildung 2.1: Fahrerkabine





$\tilde{x} = x + dx$. $f: \mathbb{R} \to \mathbb{R}$

$$\frac{|f(\tilde{x}) - f(x)|}{|f(x)|}$$

$$\frac{f(x+dx) - f(x)}{dx} = f'(\xi), \quad \xi \in [x, \tilde{x}]$$

$$\Rightarrow \left| \frac{f(x+dx) - f(x)}{f(x)} \right| = \left| \frac{f'(\xi)}{f(x)} \frac{dx}{x} \cdot x \right|$$

$$= |f'(\xi)| \left| \frac{x}{f(x)} \right| \cdot \underbrace{\frac{|dx|}{|x|}}_{\text{rel. Fehler Eingang}}$$

$dx \to 0 : \xi \to x$

$$\tilde{M} \sim M = \left| f'(x) \cdot \frac{x}{f(x)} \right|$$

„Verstärkungsfaktor"

~ 1: gut gestellt

$\gg 1$: schlecht gestellt

$f: \mathbb{R}^2 \to \mathbb{R}$. $f(x,y)$

$$M_x = \left| \frac{\partial f}{\partial x}(x,y) \cdot \frac{x}{f(x,y)} \right|, \quad M_y = \left| \frac{\partial f}{\partial y}(x,y) \cdot \frac{y}{f(x,y)} \right|$$

① $f(x,y) = x \cdot y : M_x = y \cdot \frac{x}{xy} = 1 = M_y$

② $f(x,y) = x + y : M_x = 1 \frac{x}{x+y} = \frac{1}{1 + \frac{y}{x}}$

groß, falls $\frac{y}{x} \sim -1$,

d. h. x, y unterschiedliches Vorzeichen, gleiche Größenordnung.

Auslöschung

Beispiel: $x = 1{,}01, y = -1$

$\tilde{x} = 1{,}00$ $\tilde{y} = -1{,}00$

relative Fehler: $\sim 1\%$ Eingang

$f(x,y) = 0.01$

$f(\tilde{x}, \tilde{y}) = 0$ → Fehler 100 %

Google Page Rank

Idee: Bewerte die Wichtigkeit von Seiten.

Eine Seite ist wichtig, wenn sie von wichtigen Seiten verlinkt wird.

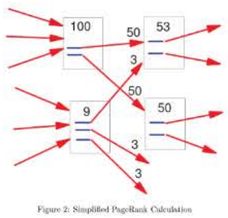
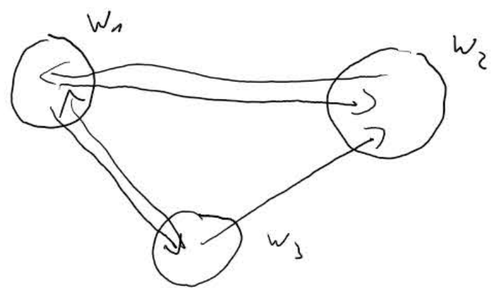


Figure 2: Simplified PageRank Calculation



$$w_1 = w_2 + \frac{w_3}{2} \qquad w = (w_1, w_2, w_3)$$

$$w_2 = \frac{w_1}{2} + \frac{w_3}{2}$$

$$w_3 = \frac{w_1}{2}$$

$$\Rightarrow w = \underbrace{\begin{pmatrix} 0 & 1 & 1/2 \\ 1/2 & 0 & 1/2 \\ 1/2 & 0 & 0 \end{pmatrix}}_{A} \cdot w$$

$\Rightarrow$ w ist Eigenvektor von A zum Eigenwert $1 > 0$

Spaltensumme ist $1 \Rightarrow 1$ ist EW,
1 ist maximaler Eigenwert.

Wir suchen einen Eigenvektor der Matrix zum betragsmaximalen Eigenwert.

Bemerkung: Jede Seite gibt einen Teil der Wichtigkeit an sich selbst ab,

$$w_1 = (1-d)\, w_1 + d\,(w_2 + w_3/2)$$

Wie berechnet man w?

Google 2009: $3 \cdot 10^{13} = 30$ Billionen

Google Page Rank

Idee: Bewerte die Wichtigkeit von Seiten.

Eine Seite ist wichtig, wenn sie von wichtigen Seiten verlinkt wird.

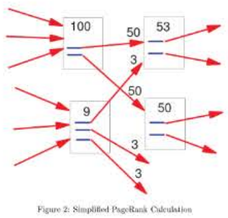
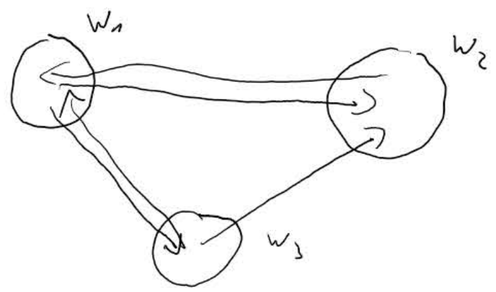


Figure 2: Simplified PageRank Calculation



$$w_1 = w_2 + \frac{w_3}{2} \qquad w = (w_1, w_2, w_3)$$

$$w_2 = \frac{w_1}{2} + \frac{w_3}{2}$$

$$w_3 = \frac{w_1}{2}$$

$$\Rightarrow w = \underbrace{\begin{pmatrix} 0 & 1 & 1/2 \\ 1/2 & 0 & 1/2 \\ 1/2 & 0 & 0 \end{pmatrix}}_{A} \cdot w$$

$\Rightarrow$ w ist Eigenvektor von A zum Eigenwert $1 > 0$

Spaltensumme ist $1 \Rightarrow 1$ ist EW,
1 ist maximaler Eigenwert.

Wir suchen einen Eigenvektor der Matrix zum betragsmaximalen Eigenwert.

Bemerkung: Jede Seite gibt einen Teil der Wichtigkeit an sich selbst ab,

$$w_1 = (1-d)\, w_1 + d\,(w_2 + w_3/2)$$

Wie berechnet man w?

Google 2009: $3 \cdot 10^{13} = 30$ Billionen

Einfache numerische Lösung von Anfangswertaufgaben

Gegeben sei die Anfangswertaufgabe

$$y'(t) = f(t, y(t))$$

mit dem Anfangswert $y(a) = y_0$. Wir suchen die Lösung dieser Gleichung auf dem Intervall $[a, b]$.

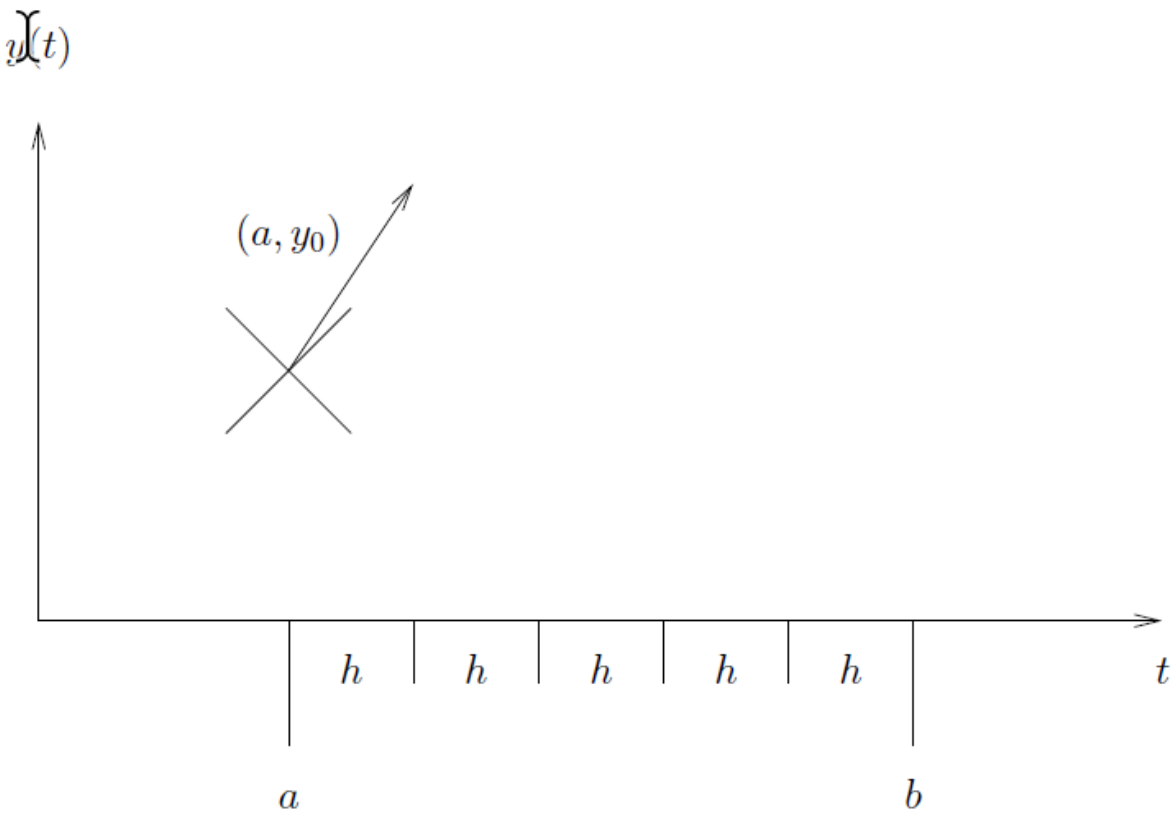
In [4]:

```
import numpy as np
import math
import matplotlib.pyplot as plt
```

Definition der Differentialgleichung. Wir setzen $f(t, y) = y$, $a = 0$, $b = 1$, $y_0 = 1$, also lautet die DGL $y'(t) = y(t)$, $y(0) = 1$, und die Lösung der AWA ist $y(t) = e^t$.

In [9]:

```
def f(t,y):
    return y
def loesung(t):
    return exp(t)
a=0
b=1.56
y0=1
```



Die Tangente hat im Punkt (a, y_0) die Geradengleichung

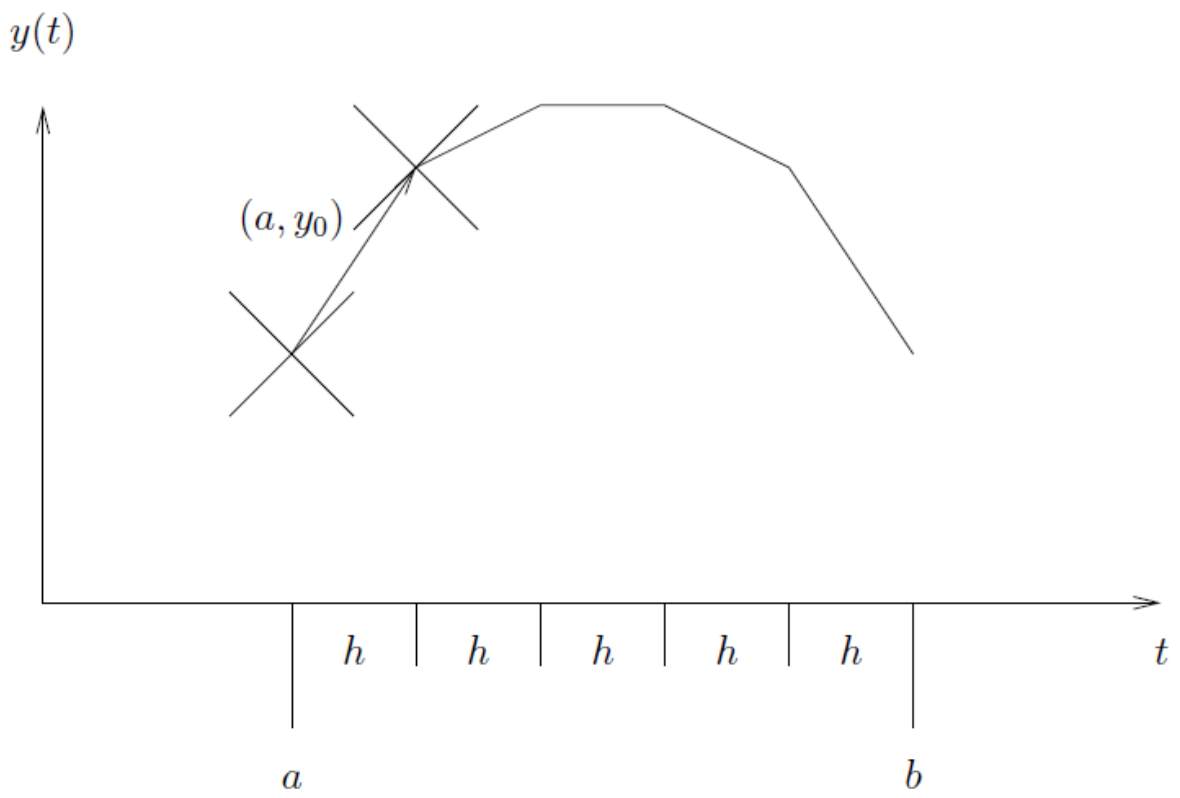
$$z(t) = y_0 + f(a, y_0) \cdot (t - a).$$

Als Näherung für den Wert von y an der Stelle $t = a + h$ erhalten wir also

$$y_1 = z(a + h) = y_0 + h \cdot f(a, y_0).$$

Das machen wir immer so weiter und erhalten dann die Näherung für $y(t_{k+1})$

$$y_{k+1} = y_k + hf(t_k, y_k).$$



In [10]:

```
n=1000
t=np.linspace(a,b,n)
h=(b-a)/(n-1)
```

In [11]:

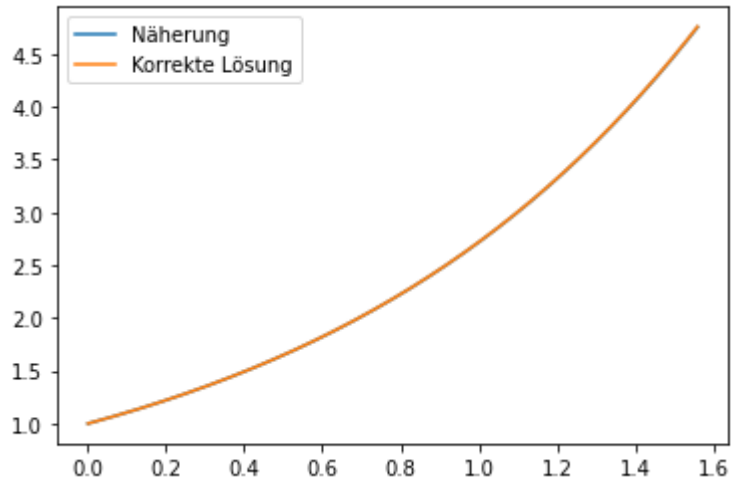
```
y=np.zeros(n)
y[0]=y0
for i in range(0,n-1):
    abl=f(t[i],y[i])
    y[i+1]=y[i]+h*abl
```

In [12]:

```
plt.plot(t,y,t,np.exp(t))
plt.legend(['Näherung','Korrekte Lösung'])
```

Out[12]:

```
<matplotlib.legend.Legend at 0x7f11a81d58d0>
```



Dieses Verfahren heißt Euler-Verfahren.

In [ ]:

```

```

Lösung einer Anfangswertaufgabe für ein System von Differentialgleichungen mit dem Eulerverfahren

Wir suchen eine Funktion $y : [a,b] \mapsto \mathbb{R}^n$, $y(t) = (y^{(1)}(t), \ldots, y^{(n)}(t))$, mit

$$y'(t) = f(t, y(t)),\ y(a) = y_0$$

wobei $y'(t) = ({y^{(1)}}'(t), \ldots, {y^{(n)}}'(t))$ und $f : [a,b] \times \mathbb{R}^n \mapsto \mathbb{R}^n$, sowie $y_0 \in \mathbb{R}^n$.

Als Beispiel betrachten wir das Problem einer schwingenden Feder, zunächst ohne Dämpfung. Sei $x(t)$ die Auslenkung der Feder, m die Masse eines anhängenden Gewichts, g die Erdbeschleunigung, dann gilt

$$x''(t) = -g - \frac{c}{m}x(t).$$

Zusammen mit der Nullstellung $x_0 = -\frac{mg}{c}$ ist dann

$$x(t) = \cos(\sqrt{\frac{c}{m}}t) + x_0$$

die Lösung der Gleichung mit der Anfangsbedingung $x(0) = 1$, $x'(0) = 0$ (d.h. wir schieben die Feder nach oben und lassen los).

All dies haben wir im Modellierungsteil nachgerechnet.

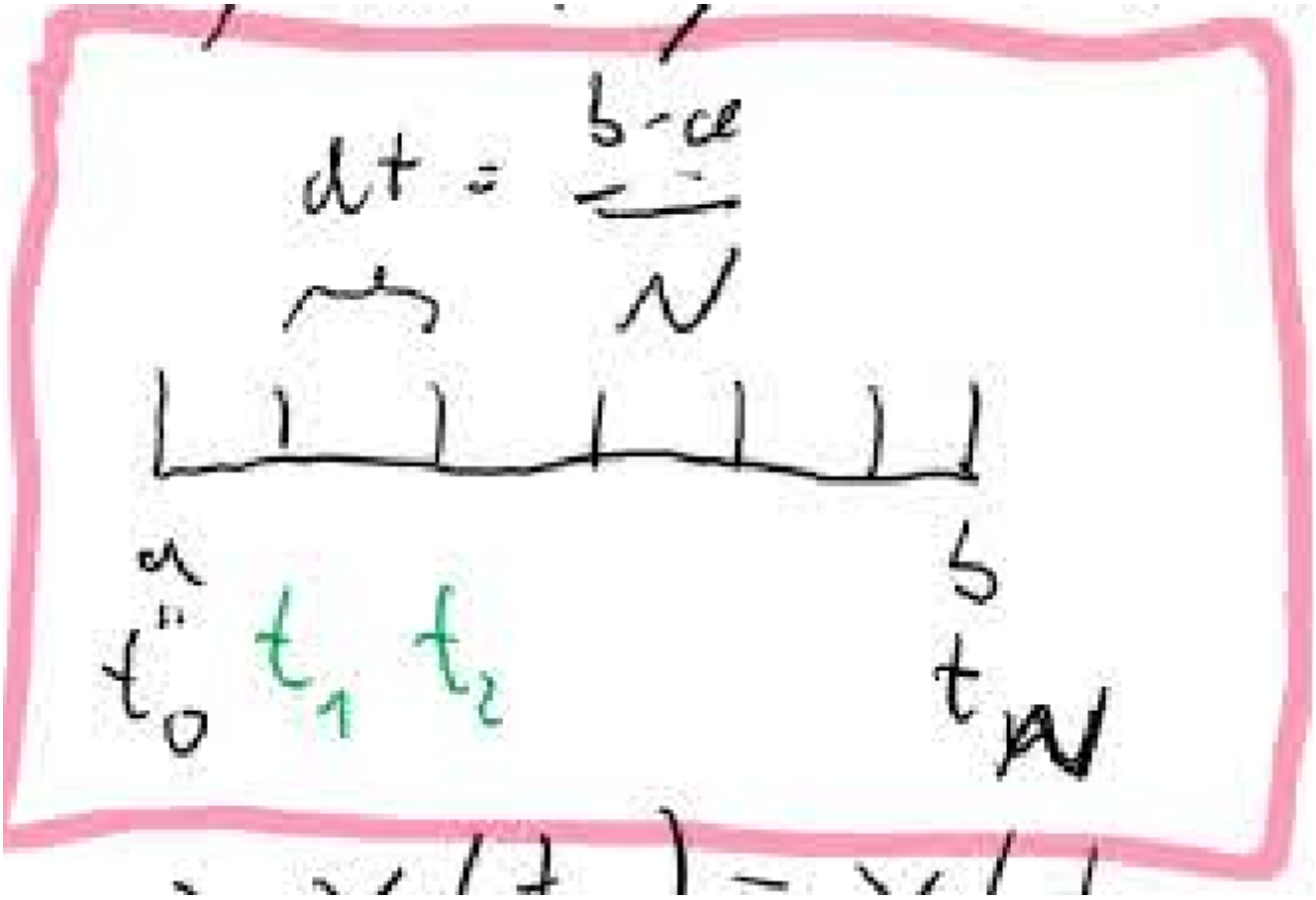
Um dies numerisch zu tun, verwandeln wir die DGL zunächst in Normalform und führen die Geschwindigkeit v ein. Wir erhalten

$$\begin{aligned} x'(t) &= v(t) \\ v'(t) &= -g - \frac{c}{m}x(t) \end{aligned}$$

Wir setzen nun $y(t) = (x(t), v(t)$. Dann gilt

$$y'(t) := (x'(t), y'(t)) = (v(t), -g - \frac{c}{m}x(t)) = (y_2(t), -g - \frac{c}{m}y_1(t))$$

Wir wählen die Federkonstante $c = 4$ und die Masse $m = 1$, dann ist die Lösung $\cos(2t) - \frac{g}{4}$. Wir versuchen, die Gleichung auf dem Intervall $[0, 2\pi]$ zu lösen. Zunächst definieren wir das Eulerverfahren. Hierzu zunächst das Bild aus der Diskretisierung:



Wir haben also die Diskretisierungspunkte t_k im Intervall $[a, b]$, wir wollen berechnen Näherungen y_k für die $y(t_k)$, und tun dies mit der Euler-Näherung

$$y_{k+1} = y_k + dt \cdot f(t_k, y_k).$$

In [1]:

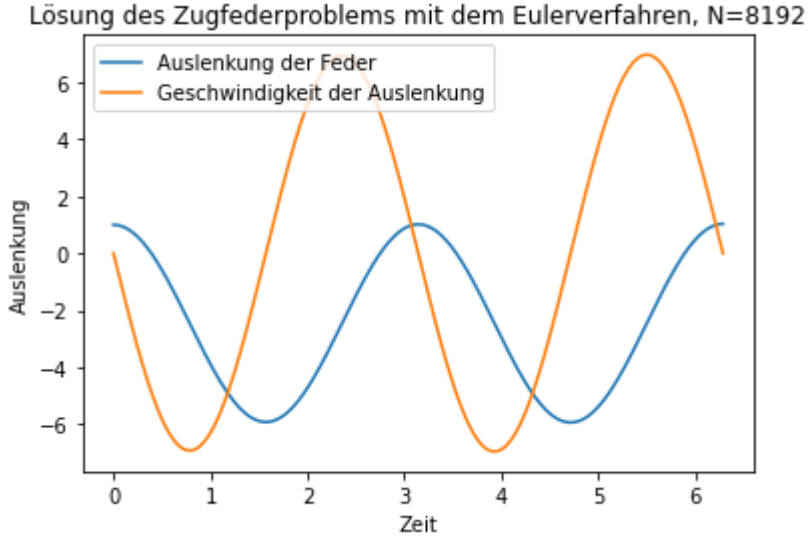
```
import numpy as np
import math
import matplotlib.pyplot as plt
```

In [2]:

```
# Löse eine Differentialgleichung y'=f(t,y), y(0)=y_0 approximativ.
def euler(f,y0,a,b,N):
    n=y0.size
    t=np.linspace(a,b,N+1)
    dt=(b-a)/N
    # Platz für die Werte von y
    y=np.zeros([N+1,n])
    # Anfangswert
    y[0,:]=y0
    # Euler-Schritt
    for k in range(0,N):
        y[k+1,:]=y[k,:]+dt*f(t[k],y[k,:])
    return (t,y)

# Zeichne die numerische Approximation
def plotit(t,y):
    N=t.size-1
    plt.title('Lösung des Zugfederproblems mit dem Eulerverfahren, N='+str(N))
    plt.plot(t,y)
    plt.xlabel('Zeit')
    plt.ylabel('Auslenkung')
    plt.legend(['Auslenkung der Feder','Geschwindigkeit der Auslenkung'])
```

In [3]:

```
# Wir definieren f wie oben angegeben, nur das Python leider bei 0 anfängt zu nummerier
en.
def f(t,y):
    x1=y[1]
    x2=-g-c/m*y[0]
    return np.array((x1,x2))
a=0
b=2*math.pi
g=9.81
c=4
m=1
N=8192
y0=np.array((1,0))
(t,y)=euler(f,y0,a,b,N)
plotit(t,y)
```

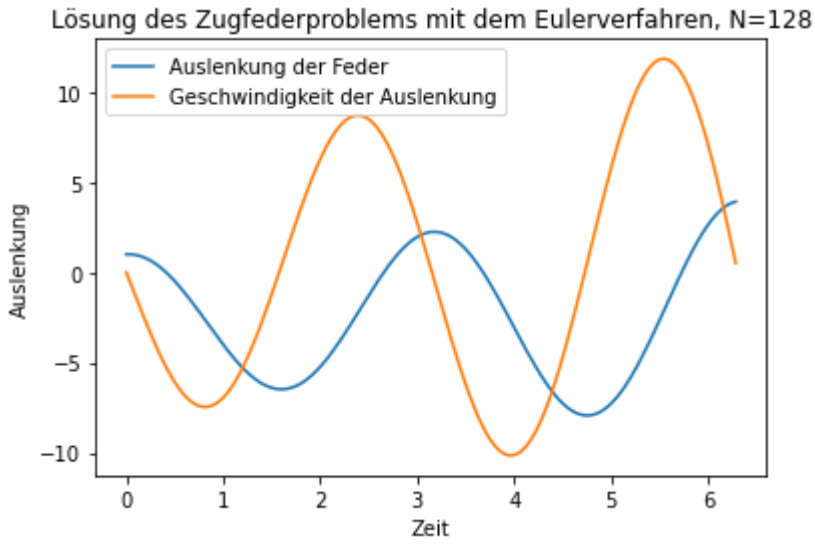


Natürlich erhalten wir zwei Lösungen, y_1 ist die Auslenkung, y_2 ihre Geschwindigkeit, wir haben beide gezeichnet. Die blaue Linie ist genau der Cosinus, den wir vorhin als Lösung angegeben haben. Das sieht also gut aus.

Wir spielen etwas herum mit dieser Lösung. Wählen wir z.B. mal N kleiner.

In [4]:

```
N=128
(t,y)=euler(f,y0,a,b,N)
plotit(t,y)
```



Offensichtlich ist das N zu klein, denn die Approximation ist recht schlecht, der Wert der Funktion y wird immer größer. Dies ist sehr ärgerlich, denn die Form der Funktion ist eigentlich korrekt. Etwas scheint also schief zu gehen.

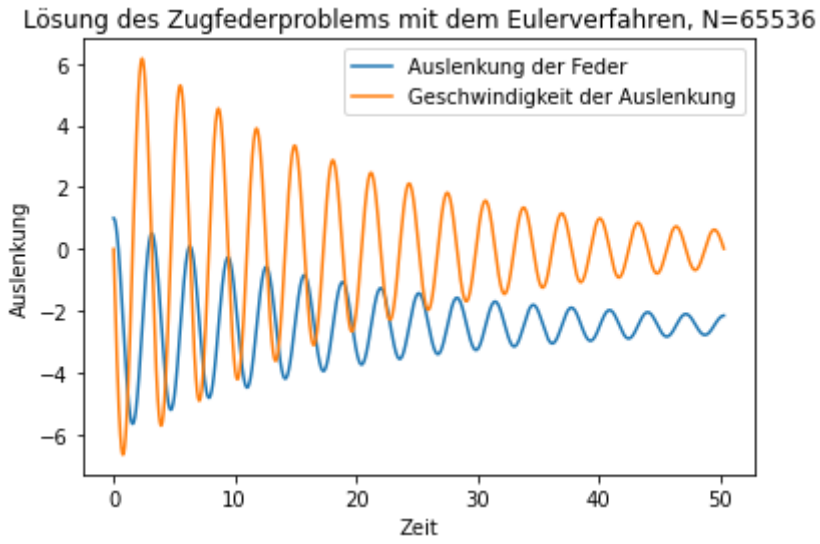
Dies kann man erklären: Die Auslenkung der Feder wird im Bild mit der Zeit immer größer, das ist komplett unphysikalisch. Größere Auslenkung würde größere Energie bedeuten, die Energie (Summe aus kinetischer und potentieller Energie) sollte aber immer konstant sein. Und genau dies ist das Problem: Unser (zu) einfaches Eulerverfahren vergrößert tatsächlich in jedem Schritt die Energie, tut also etwas unphysikalisches, deshalb ist vermutlich das Eulerverfahren hier eher ungeeignet - auch wenn es für sehr große N die korrekte Lösung liefert.

Wir ahnen also bereits: Das Eulerverfahren ist vermutlich nicht das effizienteste. Wir bleiben trotzdem erstmal dabei und wählen N halt ausreichend groß.

Abschließend wollen wir uns noch den Effekt der Dämpfung anschauen. Gemäß der Modellierung müssen wir dazu eine Dämpfungsfunktion einführen, die von der Geschwindigkeit abhängt, und die Differentialgleichung (das f) entsprechend ändern.

In [5]:

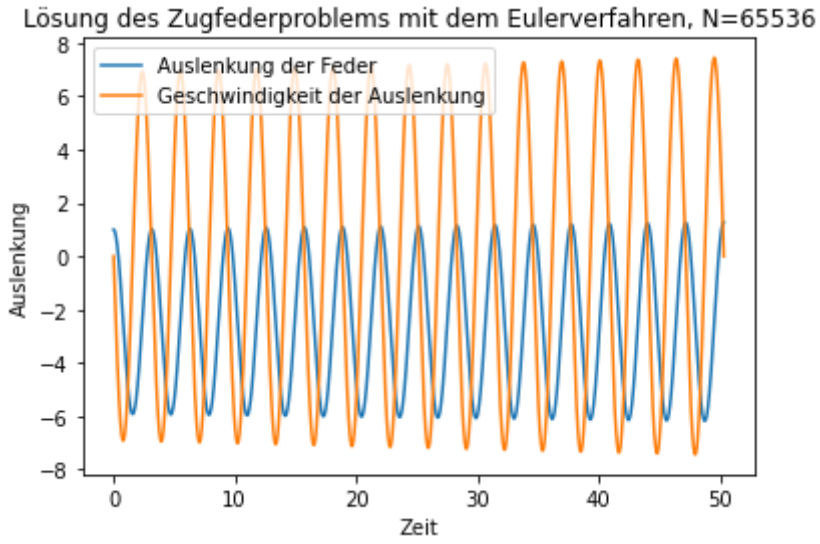
```
def daempfung(v):
    return d_faktor*v
def f(t,y):
    x1=y[1]
    x2=-g-c/m*y[0]-daempfung(y[1])
    return np.array((x1,x2))

g=9.81
c=4
m=1
perioden=16
a=0
b=perioden*math.pi
N=16*4096
d_faktor=0.1
y0=np.array((1,0))
(t,y)=euler(f,y0,a,b,N)
plotit(t,y)
```



Wieder passiert, was wir erwarten: Die Schwingung der Feder wird gedämpft. Sicherheitshalber testen wir nochmal mit Dämpfung=0. Hier sollten wir keine Dämpfung bekommen. Tatsächlich bekommen wir wegen des Energiewachstums-Effekts von oben sogar leicht größere Schwingungen, d.h. N ist noch nicht groß genug.

In [6]:

```
d_faktor=0
(t,y)=euler(f,y0,a,b,N)
plotit(t,y)
```



Genauso lässt sich jetzt das komplette Drei-Feder-Modell lösen, lediglich das f ist etwas komplizierter, und wir suchen jetzt halt 6 Funktionen. Wir übernehmen das f aus unserer Modellierung. Hierbei setzen wir jetzt natürlich

$$y = (x_1, x_2, x_3, v_1, v_2, v_3).$$

In [7]:

```
def d1(v):
    return d_1*v
def d2(v):
    return d_2*v
def d3(v):
    return d_3*v
g=9.81

# Massen
m1=120
m2=1000
m3=15

# Anfangsauslenkungen
dx1=0.1
dx2=0.15
dx3=0.03

# Federkonstanten
c1=m1*g/dx1
c2=m2*g/dx2
c3=m3*g/dx3

# Wegstrecke
def s(t):
    return 0
def ds(t):
    return 0

def f(t,y):
    # Zuerst die Ableitungen von x1,x2,x3=v1,v2,v3
    x1=y[3]
    x2=y[4]
    x3=y[5]
    # Jetzt für v1' die Differentialgleichungen aus der Modellierung
    v1=-g-c1/m1*(y[0]-y[1])-d1(y[3]-y[4])
    v2=-g-c2/m2*(y[1]-y[2])+c1/m2*(y[0]-y[1])-d2(y[4]-y[5])+d1(y[3]-y[4])
    v3=-g-c3/m3*(y[2]-s(t))+c2/m3*(y[1]-y[2])-d3(y[5]-ds(t))+d2(y[4]-y[5])
    return np.array((x1,x2,x3,v1,v2,v3))
```

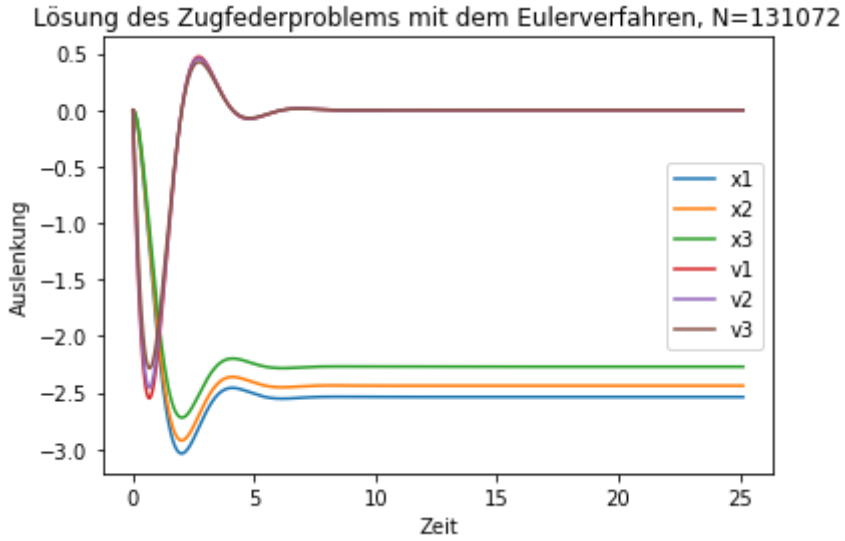
Wir setzen die Werte aus dem Buch ein.

In [8]:

```
a=0
b=8*math.pi
N=32*4096
d_1=50
d_2=50
d_3=50
y0=np.array((0,0,0,0,0,0))
(t,y)=euler(f,y0,a,b,N)
plotit(t,y)
plt.legend(['x1','x2','x3','v1','v2','v3'])
```

Out[8]:

```
<matplotlib.legend.Legend at 0x7fcb45cf0e50>
```



Erklärung: Wir starten mit den Federn in der Ruheposition. Diese werden in unterschiedliche Tiefen gezogen, durch die Dämpfung stellt sich schnell der zeitkonstante Wert ein.

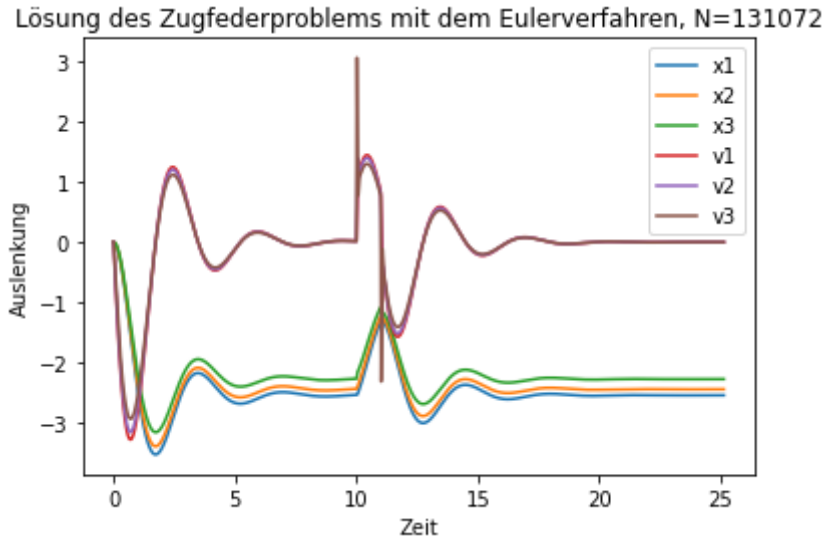
Jetzt wollen wir testen, was passiert, wenn das Auto über einen Buckel fährt, also s 1, dann wieder 0 wird.

In [9]:

```
def s(t):
    if (t>10)&(t<11):
        return 1
    else:
        return 0
d_1=30
d_2=30
d_3=30
y0=np.array((0,0,0,0,0,0))
(t,y)=euler(f,y0,a,b,N)
plotit(t,y)
plt.legend(['x1','x2','x3','v1','v2','v3'])
```

Out[9]:

```
<matplotlib.legend.Legend at 0x7fcb45624250>
```



Die Fahrerkabine wird aus der Ruhelage geworfen, der Gleichgewichtszustand stellt sich schnell wieder ein.

Das Ganze wird deutlich spannender mit dem Code aus dem Buch, der die einzelnen Federn und die Kabine zeichnet, sonst aber mehr oder weniger identisch ist (Vorführung Matlab).

In [ ]:

Induzierte Matrix norm

Sei im Folgenden $A \in \mathbb{R}^{n\times m}$, $A: \mathbb{R}^m \to \mathbb{R}^n$.

Sei $n=m$, $Ax = b$, $A \sim \tilde{A} = A + dA$
A invertier bar, $b \sim \tilde{b} = b + db \Rightarrow \tilde{A}\tilde{x} = \tilde{b}$, $\tilde{x} = x + dx$.

Fehler: $\frac{\|dx\|}{\|x\|}$ ($\|\cdot\|$ nicht notwendig $\|\cdot\|_2$)

Wie definiert man $\|A\|$ für eine Matrix?

Einfach: $\left(\sum_{i,k} |A_{ik}|^p\right)^{1/p}$. $p=2$: Frobenius-Norm.

Induziert: Angenommen, $dA = 0$.

$\Rightarrow \tilde{x} = A^{-1}\tilde{b} \Rightarrow \|dx\| = \|\tilde{x} - x\| = \|A^{-1} db\| \overset{\nabla}{\leq} \|A^{-1}\| \|db\|$

$\Rightarrow \frac{\|A^{-1} db\|}{\|db\|} \leq \|A^{-1}\| \quad \forall db \neq 0$

Also: (3.11) $\|A\| := \sup_{y\neq 0} \frac{\|Ay\|}{\|y\|}$. (Falls $m \neq n$: $\frac{\|Ay\|_{\mathbb{R}^n}}{\|y\|_{\mathbb{R}^m}}$)

$= \sup_{y\neq 0} \|A \frac{y}{\|y\|}\|$

$= \sup_{\|y\|=1} \|Ay\|$. Matrixnorm hängt von VR-Norm ab

Satz: $\|A\|$ ist eine Norm auf dem VR der Matrizen.

3.12 Matrizen. $\|A\|$ heißt induzierte Matrixnorm.

Eigenschaften: 1) $\|Ay\| \leq \|A\| \cdot \|y\|$

2) $\|AB\| = \sup_{\|y\|=1} \|ABy\|$

$\leq \sup_{\|y\|=1} \|A\| \|By\|$

$\leq \sup_{\|y\|=1} \|A\| \cdot \|B\| \cdot \|y\| = \|A\| \cdot \|B\|$

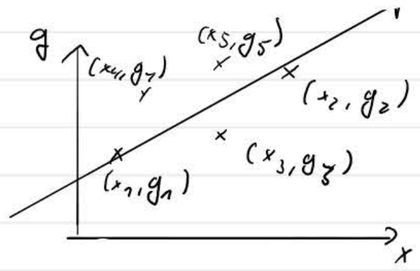
$\Rightarrow \|AB\| \leq \|A\| \cdot \|B\|$

Wie rechnet man das aus?

$\forall y \neq 0: \frac{\|Ay\|}{\|y\|} \leq \|A\|$ oder $\forall y: \|y\| = 1: \|Ay\| \leq \|A\|$

$\exists \bar{y}: \frac{\|A\bar{y}\|}{\|\bar{y}\|} = \|A\|$ $\exists \bar{y}: \|\bar{y}\| = 1: \|A\bar{y}\| = \|A\|$

Aufgabe: Löse $Ax = b$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $x \in \mathbb{R}^n$



1. F. U: $Ax = b$ hat keine Lösung

Suche ein „möglichst gutes" x^*, so dass $Ax^* \sim b$.

„Ausgleichsgerade"

$\Rightarrow \|Ax^* - b\|_2^2$ möglichst klein (Gauss: kleinste Quadrate)

Def: x^* heißt kleinste Quadrate-Lösg von $Ax = b$

$:\Leftrightarrow \|Ax^* - b\|_2^2 \leq \|Ax - b\|_2^2 \quad \forall x \in \mathbb{R}^n$.

Bemerkung: x^* ist nicht eindeutig.

Sei A die Nullmatrix, $x^* \in \mathbb{R}^n$ beliebig

$\Rightarrow \|Ax^* - b\|_2^2 = \|b\|_2^2 = \|Ax - b\|_2^2 \quad \forall x \in \mathbb{R}^n$

$\Rightarrow x^*$ ist kleinste Quadrate Lösung.

Das hilft uns nicht beim Rechnen.

Lemma: 1) $\text{Bild}(A)^\perp = \ker(A^t)$

Sei $y \in \ker(A^t) \Rightarrow A^t y = 0 \Rightarrow (A^t y, x) = 0 \quad \forall x \in \mathbb{R}^n$

$\Rightarrow (y, Ax) = 0 \Rightarrow y \in \text{Bild}(A)^\perp$.

Sei $y \in \text{Bild}(A)^\perp \Rightarrow (y, A(A^t y)) = 0$

$\Rightarrow 0 = (A^t y, A^t y) = \|A^t y\|_2^2$

$\Rightarrow y \in \ker(A^t)$.

2) $\mathbb{R}^m = \text{Bild}(A) \oplus \text{Bild}(A)^\perp$

$= \text{Bild}(A) \oplus \ker(A^t)$

3) $\ker(A^t A) = \ker(A)$: Übungen.

Unvermeidbare Fehler bei der Lösung von $Ax = b$

Sei $\widetilde{A} = A + dA$, $\widetilde{b} = b + db$, $\widetilde{A}\,\widetilde{x} = \widetilde{b}$, $\widetilde{x} = x + dx$.

$q = \|dA\| \cdot \|A^{-1}\| < 1$.

$$(A + dA)(x + dx) = b + db$$
$$Ax = b$$
$$\Rightarrow (A + dA)\,dx = db - dA \cdot x$$
$$\Rightarrow dx = (A + dA)^{-1}(db - dA \cdot x)$$
$$\Rightarrow \|dx\| \leq \|(A + dA)^{-1}\| \left(\|db\| + \|dA\| \cdot \|x\|\right)$$
$$\Rightarrow \frac{\|dx\|}{\|x\|} \leq \frac{\|A^{-1}\|}{1 - q}\left(\frac{\|db\|}{\|x\|} + \|dA\|\right)$$
$$= \frac{\|A\| \cdot \|A^{-1}\|}{1 - q}\left(\frac{\|db\|}{\|x\| \cdot \|A\|} + \frac{\|dA\|}{\|A\|}\right)$$

Kondition: $Ax = b \Rightarrow \|b\| \leq \|A\| \cdot \|x\|$

$$\frac{\|dx\|}{\|x\|} \leq \frac{\|A\| \cdot \|A^{-1}\|}{1 - q}\left(\frac{\|db\|}{\|b\|} + \frac{\|dA\|}{\|A\|}\right)$$

↳ rel. Fehler in b ↳ rel. Fehler in A

Mathematische Beschreibung des Gauss-Algorithmus: LR-Zerlegung

„Subtrahiere ein Vielfaches der i. Zeile von der k. Zeile"

$$\begin{pmatrix} 1 & & \\ -l_{21} & \ddots & 0 \\ \vdots & 0 & 1 \\ -l_{n1} & & \end{pmatrix} \begin{pmatrix} a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} a_1 \\ -l_{21}a_1 + a_2 \\ \vdots \\ -l_{n1}a_1 + a_n \end{pmatrix}$$

$$\begin{pmatrix} 1 & & & \\ & \ddots & & \\ & -l_{i+1,i} & & \\ & \vdots & 1 & \\ & -l_{n,i} & & \end{pmatrix} \begin{pmatrix} a_1 \\ \vdots \\ a_n \end{pmatrix}$$

i. Schritt

Permutation: Sei P eine Matrix mit einer 1 in jeder Zeile + Spalte, sonsten sei alles 0.

$\Rightarrow P_{i,k} = \begin{cases} 1, & k = \sigma_i \\ 0, & \text{sonst.} \end{cases}$

$$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} x_3 \\ x_2 \\ x_1 \end{pmatrix} \quad \sigma = (3,2,1)$$

Permutationsmatrix.

$$(P^t P)_{ik} = \sum_{j=1}^{n} \underbrace{P_{ji}}_{1,\, j=\sigma_i} \cdot \underbrace{P_{jk}}_{1,\, k=\sigma_j} = \begin{cases} 1, & i=k \\ 0, & \text{sonst} \end{cases} \Rightarrow P^t P = I.$$

Gegeben sei die Aufgabe $Ax = b$.

$PAx = Pb$ das Gleichungssystem, auf dem das Eliminationsverfahren ausgeführt wird.

Sei $L^{(1)} = \begin{pmatrix} 1 & & \\ -l_{21} & \ddots & 0 \\ \vdots & & 1 \\ -l_{n1} & & \end{pmatrix}$ mit $l_{k1} = \frac{(A^{(1)})_{k,1}}{(A^{(1)})_{1,1}}$.

Dann beschreibt $L^{(1)}PA$ den ersten Eliminationsschritt für A.

$\Rightarrow A^{(2)} = L^{(1)} \cdot A$

$$L^{(2)} = \begin{pmatrix} 1 & & & \\ & 1 & & \\ & -l_{32} & \ddots & \\ & \vdots & & 1 \\ & -l_{n2} & & \end{pmatrix} \Rightarrow L^{(2)} L^{(1)} \overset{P}{A}$$ zweiter Eliminationsschritt

$A^{(2)} = L^{(2)} \cdot A^{(2)}$

$A^{(n)} = L^{(n-1)} \cdots L^{(2)} \cdot L^{(1)} \cdot PA$

Bemerkungen: ① $\left(L^{(1)}\right)^{-1} = \begin{pmatrix} 1 & & \\ l_{21} & \ddots & \\ l_{n1} & & 1 \end{pmatrix}$

② $L^{(1)} L^{(2)} = \begin{pmatrix} 1 & & & \\ l_{21} & 1 & & \\ l_{31} & l_{32} & \ddots & \\ l_{n1} & l_{n2} & & 1 \end{pmatrix} \Rightarrow$ Überlagerung

$$\Rightarrow PA = \left(L^{(1)}\right)^{-1} \cdot \left(L^{(2)}\right)^{-1} \cdots \left(L^{(n-1)}\right)^{-1} \cdot A^{(n)}$$

$$= \underbrace{\begin{pmatrix} 1 & & \\ l_{21} & \ddots & 0 \\ l_{n1} & l_{n,n-1} & 1 \end{pmatrix}}_{\text{(zusammengef.) Linke untere Dreiecksmatrix}} \underbrace{\begin{pmatrix} * & & * \\ & \ddots & \\ 0 & & * \end{pmatrix}}_{\text{rechte obere Dreiecksmatrix}}$$

Satz: Für alle Matrizen $A \in \mathbb{R}^{n \times n}$ gibt es eine normierte linke untere Dreiecksmatrix L, eine rechte obere Dreiecksmatrix R, und eine Permutationsmatrix P so dass

$PA = LR \quad (P, L, R \in \mathbb{R}^{n \times n})$.

Rechner rechnen falsch!

Nicht alle Zahlen sind darstellbar.
- endlich viele Stellen
- $m_p\, m_{p-1} \dots m_1 \cdot m_{-1} \dots m_{-p}$ → Basis $b = 2$ oder 10.

$0, m_1 m_2 \dots m_p \cdot b^e$, $m_1 \neq 0$ (keine 0)

$100 = 0.1 \cdot 10^3$

$7 = 0.7 \cdot 10^1$

$p = 2$: 124 nicht darstellbar

p heißt Mantissenlänge,

M Menge der darstellbaren Zahlen (Maschinenzahlen).

$b = 2$, $p = 23$ oder $p = 52$.

Sei $x \in \mathbb{R}$. $rd : \mathbb{R} \to M$, „betragsmäßig nächste Zahl".

5 nach oben oder unten im 10er-System?

IEEE 754, 1985. Beispiel: $b = 10$, $p = 2$.

$rd(x) = 0.44 \cdot 10^e \Rightarrow x \in [0{,}435, 0{,}445] \cdot 10^e$

$|x - rd(x)| \leq 0.005 \cdot 10^e$

$= \frac{b}{2} \cdot b^{-p-1} \cdot b^e$

$m_1 \neq 0 \Rightarrow |rd(x)| \geq 0.1 \cdot b^e$

$= b^{e-1}$

$$\frac{|x - rd(x)|}{|rd(x)|} \leq \frac{\frac{b}{2} \cdot b^{-p-1} \cdot b^e}{b^{e-1}} = b^{-p+1} \cdot \frac{1}{2} = eps$$

Bemerkung: Zwischenergebnisse müssen gerundet werden.

$\underbrace{0.11}_{\in M} \cdot \underbrace{0.11}_{\in M} = 0.121 \notin M \Rightarrow M$ nicht abgeschlossen.

∞-Norm einer Matrix

$$\underset{A}{\begin{pmatrix} 2 & 1 \\ -1 & 3 \end{pmatrix}} \underset{x}{\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}} = \begin{pmatrix} 2x_1 + x_2 \\ -x_1 + 3x_2 \end{pmatrix}$$

$\| x \|_\infty = \max(|x_1|, |x_2|) \Rightarrow |x_1| \leq \|x\|_\infty, |x_2| \leq \|x\|_\infty$

$\| Ax \|_\infty = \max(|2x_1 + x_2|, |-x_1 + 3x_2|)$

$\leq 2|x_1| + |x_2| \quad \leq |x_1| + 3|x_2|$

$\leq 3\|x\|_\infty \quad \leq 4\|x\|_\infty$

$\Rightarrow \frac{\|Ax\|_\infty}{\|x\|_\infty} \leq 4.$ $\Rightarrow$ Max der Summe der Beträge in jeder Zeile

$x_1 = -1, x_2 = 1 : Ax = \begin{pmatrix} -1 \\ 4 \end{pmatrix}$

$\|x\| = 1, \quad \|Ax\| = 4, \quad \frac{\|Ax\|}{\|x\|} = 4$

Sei $A \in \mathbb{R}^{n \times m}$, $A = (a_{ij})$.

Sei $x \in \mathbb{R}^m$, $x = (x_j)$.

$\|x\|_\infty = \max\{|x_1|, \dots, |x_n|\}. \quad |x_j| \leq \|x\|_\infty.$

$|(Ax)_i| = \left| \sum_{j=1}^{m} A_{ij} x_j \right| \leq \sum_{j=1}^{m} |A_{ij}| |x_j| \leq \left(\sum_{j=1}^{m} |A_{ij}| \right) \cdot \|x\|_\infty$

$\|Ax\|_\infty = \max_i |(Ax)_i| \leq \underbrace{\left(\max_i \sum_{j=1}^{m} |A_{ij}| \right)}_{\|A\|_\infty} \cdot \|x\|_\infty$

Sei i ein Index, an dem das Maximum angenommen wird.

Wähle $\tilde{x} \in \mathbb{R}^m$: $\tilde{x}_j = \begin{cases} 1, & A_{ij} \geq 0 \\ -1, & A_{ij} < 0 \end{cases}$.

$(A\tilde{x})_i = \sum_{j=1}^{m} A_{ij} \tilde{x}_j \leq \sum_{j=1}^{m} |A_{ij}| = \|A\|_\infty$

$\|A\|_\infty = \max_i \sum_{j=1}^{n} |A_{ij}|.$

Minimum Norm Lösung

Statt $Ax = b$ löse $A^t Ax = A^t b$

$\Rightarrow$ kleinste Quadrate Lösung, Existenz gesichert

Eindeutigkeit?

Idee von Gauss: Suche die kQL mit kleinster Norm.

Definition: x^+ Minimum Norm Lösung von $Ax = b$

$:\Leftrightarrow$ 1) x^+ ist kleinste Quadrate-Lösung von $Ax = b$

2) x^+ hat unter allen kQL die kleinste Norm, d.h.

$\|x^+\| \leq \|x\| \quad \forall x : x$ kQL.

Sei x^* kleinste Quadrate-Lösung.

$\mathbb{R}^n = \text{Bild}(A^t) \oplus \ker(A) \Rightarrow x^* = x^+ + x_2$

$x^+ \in \text{Bild}(A^t)$, $x_2 \in \ker(A)$

Beh.: x^+ ist MNL.

1) $A^t A x^+ = A^t A (x^+ + x_2) = A^t A x^* = A^t b$

2) Sei $\bar{x}$ eine kQL $\Rightarrow A^t A \bar{x} = A^t b$

$\Rightarrow A^t A (x^+ - \bar{x}) = 0 \Rightarrow A(x^+ - \bar{x}) = 0$

$\Rightarrow \bar{x} = x^+ + w, \quad w \in \ker(A)$.

$\bar{x} \in \text{Bild}(A^t)$, $w \in \ker(A)$

$\Rightarrow \|\bar{x}\|_2^2 = \|x^+\|_2^2 + \|w\|_2^2 \geq \|x^+\|_2^2$, "=" $\Leftrightarrow w = 0$

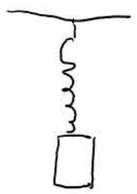
$\Leftrightarrow x^+ = \bar{x}$

$\Rightarrow x^+$ ist einzige MNL.

Satz: x^+ ist MNL von $Ax = b$

$\Leftrightarrow$ 1) $A^t A x^+ = A^t b$

2) $x^+ \in \text{Bild}(A^t)$



Federkonstante

x_0 · c

$-m \cdot g$

Erd-Beschleunigung

Gleichgewicht, (Hookesches Gesetz)

$0 = -m \cdot g - x_0 \cdot c \;,\; x_0 = -\frac{m \cdot g}{c}$

Auslenkung $x(t)$

$F(t) = -m \cdot g - x \cdot c$

$= m \cdot x''(t)$ (Kraft = Masse · Beschleunigung)

$\Rightarrow x''(t) = -g - \frac{c}{m} x(t)$

$x(t) = \cos\left(\sqrt{\frac{c}{m}} \cdot t\right) + x_0 \quad (\sin() + x_0)$

$\Rightarrow x''(t) = -\frac{c}{m}\left(\cos\left(\sqrt{\frac{c}{m}} \cdot t\right) + x_0\right) + \frac{c}{m} \cdot x_0$

$= -g \qquad -\frac{c}{m} \cdot x(t)$

Problem: Schwingt für immer!

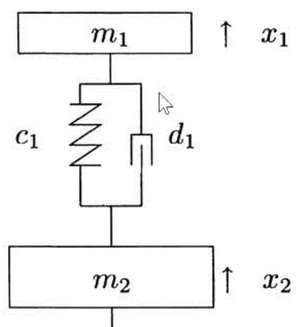
→ Dämpfung, (kinetische) Energie geht verloren.

$d(x'(t))$, z. B. $-d \cdot x'(t)$

$x''(t) = -g - \frac{c}{m} \cdot x(t) - \frac{d(x'(t))}{m}$ DGl 2. Ordg

$x'(t) = v(t)$

$x''(t) = v'(t) = -g - \frac{c}{m} \cdot x(t) - \frac{d(v(t))}{m}$ System von DGl 1. Ordg



$v_1'(t) = -g - \frac{c_1}{m_1} \cdot (x_1(t) - x_2(t)) - \frac{d_1(v_1(t) - v_2(t))}{m_1}$

$x_1'(t) = v_1(t)$

Neumannsche Reihe

Sei $A = I_2$, $\tilde{A} = A + dA$, $\tilde{A} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$.

$\Rightarrow \tilde{A}$ nicht invertierbar.

Satz: Sei $A \in \mathbb{R}^{n \times n}$, $\|A\| < 1$,

induzierte Matrixnorm.

Dann ist $(I - A)^{-1} = \sum_{k=0}^{\infty} A^k$.

Beweis: Geometrische Reihe!

$\| \sum_{k=0}^{n} A^k \| \leq \sum_{k=0}^{n} \|A\|^k \Rightarrow$ CF

$\Rightarrow \sum_{k=0}^{\infty} A^k$ konvergiert.

$(I - A) \sum_{k=0}^{n} A^k = I - A^{n+1} \xrightarrow{n \to \infty} I$.

$\Rightarrow \sum_{k=0}^{\infty} A^k = (I - A)^{-1}$.

Korollar: Sei $A \in \mathbb{R}^{n \times n}$ invertierbar,

$dA \in \mathbb{R}^{n \times n}$: $q := \|dA\| \cdot \|A^{-1}\| < 1$,

$\Rightarrow A + dA = A \underbrace{\left(I - \underbrace{(-A^{-1} dA)}_{\|\cdot\| \leq q < 1} \right)}_{\text{invertierbar}}$

(inv.)

$\Rightarrow A + dA$ invertierbar

$(A + dA)^{-1} = \sum_{k=0}^{\infty} (-A^{-1} dA)^k \cdot A^{-1}$

$\|(A + dA)^{-1}\| \leq \|A^{-1}\| \sum_{k=0}^{\infty} q^k = \|A^{-1}\| \cdot \frac{1}{1 - q}$

$A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $x \in \mathbb{R}^n$

2) $\mathbb{R}^m = Bild(A) \oplus Bild(A)^\perp$
$= Bild(A) \oplus ker(A^t)$

3) $ker(A^tA) = ker(A)$

$\|Ax^* - b\|^2 \leq \|Ax - b\|^2 \quad \forall x \in \mathbb{R}^n$.

Satz: x^* ist kleinste Quadrate-Lösung von $Ax = b$
$\Leftrightarrow A^tAx^* = A^tb$
(„Gausssche Normalgleichung")

Beweis: $b \in \mathbb{R}^m \Rightarrow b = \underbrace{b_1}_{\in Bild(A)} \oplus b_2 \leftarrow \in ker(A^t)$

Sei $x \in \mathbb{R}^n$ beliebig.

$\|Ax - b\|_2^2 = \|\underbrace{Ax - b_1}_{\in Bild(A)} - \underbrace{b_2}_{\in ker(A^t)}\|_2^2 \overset{\text{Pythagoras}}{=} \|Ax - b_1\|_2^2 + \|b_2\|_2^2$

minimal, falls $Ax = b_1$.

Da $b_1 \in Bild(A)$, gibt es mindestens ein x^* mit $Ax^* = b_1$, also eine kQL.

x^* kQL $\Rightarrow Ax^* = b_1 \Rightarrow A^tAx^* = A^tb_1 + \underbrace{A^tb_2}_{=0} = A^tb$

Sei $A^tAx = A^tb = A^tb_1 = A^tAx^*$
$\Rightarrow A^tA(x - x^*) = 0$
$\Rightarrow A(x - x^*) = 0$, denn $ker(A^tA) = ker(A)$.
$\Rightarrow Ax = Ax^* = b_1 \Rightarrow x$ ist kQL.

Insgesamt: x^* kQL $\Leftrightarrow A^tAx^* = A^tb$.

pq-Formel

April 21, 2020

1 Beispiel: Verstärkungsfaktoren und Stabilität für die pq-Formel

Abschließend wollen wir uns an *dem* Standardbeispiel für programmierbare Formeln aus der Schule anschauen, ob die Standardformel korrekt ist, an der pq-Formel. Da die Rechnung etwas aufwändiger ist, lassen wir dies Python symbolisch tun. Jeder andere symbolische Rechner (Maple, Mathematica, Wolfram Alpha) tut es genauso.

[1]:
```
import sympy
```

[2]:
```
p=sympy.Symbol('p')
q=sympy.Symbol('q')
x=sympy.Symbol('x')
x1=-p/2-sympy.sqrt((p/2)**2-q)
x2=-p/2+sympy.sqrt((p/2)**2-q)
display(x1)
display(x2)
```

$$-\frac{p}{2} - \sqrt{\frac{p^2}{4} - q}$$

$$-\frac{p}{2} + \sqrt{\frac{p^2}{4} - q}$$

Wir schauen uns die zweite Formel für positives p und $q \sim 0$ an. Dann werden hier zwei fast gleichgroße Zahlen voneinander subtrahiert.

Wir wählen also $p = 2, q = 1e - 20$.

[3]:
```
pwahr=2
qwahr=1e-20
x2.subs(p,pwahr).subs(q,qwahr)
```

[3]:
$$0$$

[6]:
```
dx2p=sympy.simplify(p*sympy.diff(x2,p)/x2)
dx2q=sympy.simplify(q*sympy.diff(x2,q)/x2)
display(dx2p)
display(dx2q)
```

$$-\frac{p}{\sqrt{p^2 - 4q}}$$

$$\frac{2q}{-p^2 + p\sqrt{p^2 - 4q} + 4q}$$

[7]:
```
display(dx2p.subs(p,pwahr).subs(q,qwahr))
display(dx2q.subs(p,pwahr).subs(q,qwahr))
```

-1.0

0.5

[13]:
```
x1numerisch=x1.subs(p,pwahr).subs(q,qwahr)
print(x1numerisch)
```

```
-2.00000000000000
```

[14]:
```
print(qwahr/x1numerisch)
```

```
-5.00000000000000e-21
```

[ ]:
```

```

Stabilität eines Algorithmus

Ein Algorithmus ist gut (stabil), wenn der entstehende Fehler in der Größenordnung des unvermeidlichen Fehlers liegt.

① $x+y$, $x=1, y=1$

② $(x+2)+(y-2)$, $2=1000 \Rightarrow$ Fehler 100%.

$1001 \quad -999$

$0.1 \cdot 10^4 \rightarrow -0.1 \cdot 10^4$

$+$

0

$x_1 = -\frac{p}{2} + \sqrt{\left(\frac{p}{2}\right)^2 - q}$ $\quad |q| \ll p \Rightarrow$ Auslöschung $\rightarrow 0$

③ Variation der Konstanten

$y'(t) = \alpha(t) \cdot \beta(t) \cdot y(t)\,, y(a) = y_0$

$y(t) = c(t) \cdot e^{\int_a^t \beta(s)\,ds}$

$y'(t) = c'(t) \cdot e^{\int_a^t \beta(s)\,ds} + c(t) \cdot e^{\int_a^t \beta(s)\,ds} \cdot \beta(t)$

$= c'(t) \cdot e^{\int_a^t \beta(s)\,ds} + y(t) \cdot \beta(t)$

$\Rightarrow c'(t) \cdot e^{\int_a^t \beta(s)\,ds} = \alpha(t)$

$\Rightarrow c'(t) = \alpha(t) \cdot e^{-\int_a^t \beta(s)\,ds}$

$\Rightarrow c(t) = \int_a^t \alpha(\tau) \cdot e^{-\int_a^\tau \beta(s)\,ds}\,d\tau + C$

$C = y_0 \Rightarrow c(a) = y_0$

$\Rightarrow y(t) = y_0$

$y(t) = (y_0 + \int_a^t \alpha(\tau) \cdot e^{-\int_a^\tau \beta(s)\,ds}\,d\tau) \cdot e^{\int_a^t \beta(s)\,ds}$

$= y_0 \cdot e^{\int_a^t \beta(s)\,ds} + \int_a^t \alpha(\tau) \cdot e^{\int_\tau^t \beta(s)\,ds}\,d\tau$

⑤

Aufgabe 2: (4 Punkte)
Überprüfen Sie, ob die Beispiele 3.5 aus 11.3 die Voraussetzungen des Satzes von Picard-Lindelöf erfüllen. Geben Sie ggf. das größtmögliche Existenzintervall an, das Sie mit dem globalen Satz erhalten können.

$f \in C^0(\Omega, \mathbb{R}^n)$, $\Omega = \{(t, y_1, \dots, y_n) : t \in \mathbb{R}, y_1, \dots, y_n \in \mathbb{R}\}$

$f : K_\delta(y_0) \subset G_1$, $M = \sup_{(t,y) \in \Omega} \|f(t,y)\| < \infty$, $\alpha = \frac{\delta}{M}$.

$0 < \frac{\delta}{M} < \alpha(t) = \alpha(0) + \beta(0)\,y(0) + \gamma_0 = \text{const} > 0$

$\Rightarrow f(t,y) = \alpha(t) + \beta \cdot (t) \cdot y$

$\Rightarrow \|f(t,y) - f(t,y_2)\| \le \|\beta(t)\| \cdot \|y_1 - y_2\|$

$\Rightarrow \|f(t,y)\| \le \|\alpha(t)\| + \|\beta(t)\| \cdot \|y\| \le M$

$\Rightarrow \frac{\delta}{M} \ge \delta$ / Lösung $u = y_0 + \beta(t) \cdot (\|y_0\| + \delta) \Rightarrow \overrightarrow{t_0 u} \in K_\delta(y_0) \Rightarrow \{u, \alpha \cdot \overrightarrow{u_0 t_0}\}$

$y(t) = 1 = y(t)^2,\ y(0) = 0.$

$f(t,y) = t \cdot y^2$, direkt mit Satz 4 aus 11.2 lösbar.

$\Rightarrow$ Lipschitz-Bedingung.

$\Rightarrow$ f ist nicht Lipschitz

$0 < K_\delta(y_0) = [-\delta, \delta]$, $M = \eta + \delta^2$.

$\{u, \alpha \cdot \frac{\delta}{M}\}$, $\frac{\delta}{M} = \frac{\delta}{\eta + \delta^2}$ $(\eta \neq 0) \cdot \delta^2 + \delta \cdot \eta - \alpha \delta \ge 0$.

$y(0) = 1, y(0) = 0$

$f(t,y) = y^{-2/3}$.

$\frac{1}{3} \cdot y^{-2/3}$

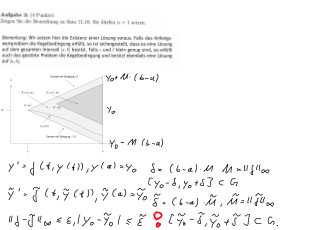
$\|f(t,y) - f(t,0)\| = y^{-2/3} \not\le L \cdot y$ $\Rightarrow L \ge y^{-2/3} \to \infty$

$\Rightarrow f$ ist nicht Lipschitz

$\Rightarrow f$ erfüllt nicht Picard-Lindelöf

Aufgabe 3: (4 Punkte)
Zeigen Sie die Behauptung im Satz 11.10. Sie dürfen $n = 1$ setzen.

Bemerkung: Wir setzen hier die Existenz einer Lösung voraus. Falls das Anfangsproblem die Kegelbedingung erfüllt, so ist sichergestellt, dass es eine Lösung auf dem gesamten Intervall $[a, b]$ besitzt. Falls γ und δ klein genug sind, so erhält man auch das gestörte Problem die Kegelbedingung und besitzt ebenfalls eine Lösung auf $[a, b]$.



$y' = f(t, y(t)), y(a) = y_0 \quad \delta = (b-a) \cdot M \quad M = \|f\|_\infty$

$C y_0 - \delta, y_0 + \delta] \subset G$

$\tilde{y}' = \tilde{f}(t, \tilde{y}(t)), \tilde{y}(a) = \tilde{y}_0 \quad \tilde{\delta} = (b-a) \cdot \tilde{M}, \tilde{M} = \|\tilde{f}\|_\infty$

$\|f - \tilde{f}\|_\infty \le \varepsilon, |y_0 - \tilde{y}_0| \le \tilde{\varepsilon}$? $C \tilde{y}_0 - \tilde{\delta}, \tilde{y}_0 + \tilde{\delta}] \subset G$.

$\Rightarrow \|\tilde{f}\|_\infty \le \|f\|_\infty + \varepsilon \Rightarrow \tilde{M} \le M + \varepsilon$

$\tilde{\delta} = (b-a) \cdot \tilde{M} \le \delta + (b-a) \cdot \varepsilon$

$C \tilde{y}_0 - \tilde{\delta}, \tilde{y}_0 + \tilde{\delta}] \subset [y_0 - (\delta + (b-a) \cdot \varepsilon),$
$y_0 + \tilde{\varepsilon} + (\delta + (b-a) \cdot \varepsilon)]$

Goursat $\Rightarrow [y_0 - \delta - \frac{\varepsilon}{2}, y_0 + \delta + \frac{\varepsilon}{2} + \frac{\varepsilon}{2}] \subset G$

$\tilde{\varepsilon} < \frac{\varepsilon}{2}$, $(b-a) \cdot \varepsilon < \frac{\varepsilon}{2}$

$\subset [y_0 - \delta - \frac{\varepsilon}{2} - \frac{\varepsilon}{2}, y_0 + \delta + \frac{\varepsilon}{2} + \frac{\varepsilon}{2}] \subset G$

$\Rightarrow$ ist stetig auf $[a, b]$.

Aufgabe 4: (4 Punkte)
Zeigen Sie die folgenden Lemma von Gronwall (durch Induktion).
Seien $(\alpha_k), (\beta_k), (\gamma_k)$ reelle nichtnegative Folgen und

$\forall_{n \in \mathbb{N}} : \alpha_n \le \gamma_n + (1 + \beta_n) \cdot \alpha_n \ge 0$.

Dann gilt:

$\alpha_n \le \gamma_n + \sum_{k=0}^{n-1} \gamma_k \cdot \exp\left(\sum_{j=k}^{n-1} \beta_j\right)$

Bemerkung: Das diskrete Lemma erinnert uns dann Lemma von Gronwall durch Diskretisierung der Funktionen α, γ, β mittels $\alpha_n = \alpha(n)$ usw.

$k = 0 : \checkmark$

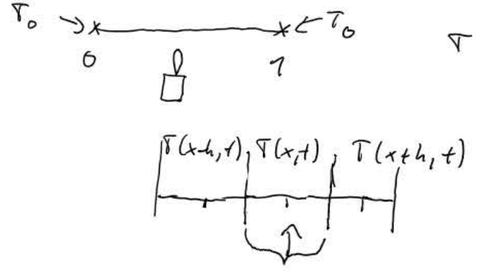
$k \to k+1 : (\alpha_0 + \sum_{i=0}^{k} \alpha_i) \cdot e^{\sum_{i=0}^{k} \beta_i}$

$\overset{\text{I.V.}}{\le} \alpha_0 \cdot e^{\sum_{i=0}^{k} \beta_i} + \gamma_0 \cdot e^{\sum_{i=0}^{k} \beta_i} + \sum_{i=1}^{k} \gamma_i \cdot e^{\sum_{i=0}^{k} \beta_i}$

$\Rightarrow \alpha_{k+1} \le \underbrace{\alpha_0 \cdot e^{\beta_0}}_{\Rightarrow \alpha_1} + \gamma_{k+1} \cdot \underbrace{e^{\sum_{i=0}^{k} \beta_i + \beta_k}}_{\Rightarrow \alpha_1}$

$\Rightarrow (\alpha_1 \beta_1) \cdot e_1 + \alpha_1 e$

$\Rightarrow$ klar



$T(x,t)$ = Temperatur zum Zeitpunkt t an Stelle x

$T(x,0) = T_0$

$$h \cdot T(x, t+dt) = h \cdot T(x,t) + \frac{h}{2}\left(T(x+h,t) - T(x,t)\right) \cdot dt + \frac{h}{2}\left(T(x-h,t) - T(x,t)\right) \cdot dt + s(x) \cdot h \cdot dt$$

$$\frac{T(x,t+dt) - T(x,t)}{dt} = \frac{1}{h^2}\left(T(x+h,t) - 2T(x,t) + T(x-h,t)\right)$$

$$\to \frac{\partial}{\partial t} T(x,t) = T(x,t) + h\frac{\partial}{\partial x} T(x,t) + \frac{h^2}{2}\frac{\partial^2}{\partial x^2} T(x,t) + \dots$$
$$T(x,t) - h\frac{\partial}{\partial x} T(x,t) + \frac{h^2}{2}\frac{\partial^2}{\partial x^2} T(x,t) + \dots$$

$$\approx \frac{\partial^2}{\partial x^2} T(x,t) + s(x)$$

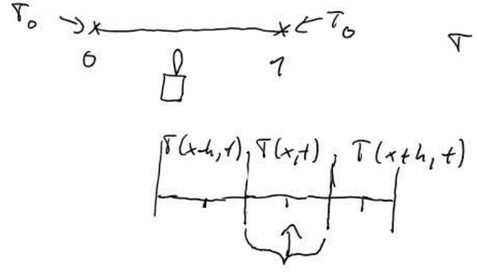
$$\left(\frac{\partial}{\partial t} T(x,t)\right) = \frac{\partial^2}{\partial x^2} T(x,t) \quad \text{Wärmeleitungsgleichung}$$
$$= 0 \qquad + s(x)$$

stationär: $-\frac{\partial^2}{\partial x^2} T(x) = s(x)$

$$\begin{cases} -T''(x) = s(x) \\ T(0) = T_0,\ T(1) = T_0. \\ \text{Randwertproblem.} \end{cases}$$

$T(x,y,t)$

$$-\frac{\partial^2}{\partial x^2} T(x,y,t) - \frac{\partial^2}{\partial y^2} T(x,y,t) = s(x)$$



$T(x,t)$ = Temperatur zum Zeitpunkt t an Stelle x

$T(x,0) = T_0$

$$h \cdot T(x, t+dt) = h \cdot T(x,t) + \frac{h}{2}\left(T(x+h,t) - T(x,t)\right) \cdot dt + \frac{h}{2}\left(T(x-h,t) - T(x,t)\right) \cdot dt + s(x) \cdot h \cdot dt$$

$$\frac{T(x,t+dt) - T(x,t)}{dt} = \frac{1}{h^2}\left(T(x+h,t) - 2T(x,t) + T(x-h,t)\right)$$

$$\to \frac{\partial}{\partial t} T(x,t) = T(x,t) + h\frac{\partial}{\partial x} T(x,t) + \frac{h^2}{2}\frac{\partial^2}{\partial x^2} T(x,t) + \dots$$
$$T(x,t) - h\frac{\partial}{\partial x} T(x,t) + \frac{h^2}{2}\frac{\partial^2}{\partial x^2} T(x,t) + \dots$$

$$\approx \frac{\partial^2}{\partial x^2} T(x,t) + s(x)$$

$$\left(\frac{\partial}{\partial t} T(x,t)\right) = \frac{\partial^2}{\partial x^2} T(x,t) \quad \text{Wärmeleitungsgleichung}$$
$$= 0 \qquad + s(x)$$

stationär: $-\frac{\partial^2}{\partial x^2} T(x) = s(x)$

$$\begin{cases} -T''(x) = s(x) \\ T(0) = T_0,\ T(1) = T_0. \\ \text{Randwertproblem.} \end{cases}$$

$T(x,y,t)$

$$-\frac{\partial^2}{\partial x^2} T(x,y,t) - \frac{\partial^2}{\partial y^2} T(x,y,t) = s(x)$$