## Definition einer Funktion mit mehreren Variablen

Gegeben seine zwei Gleichungen

$$x_2 = -x_1^2 + 11$$
$$x_2 = \sqrt{-x_1 + 7}$$

In Funktionen = 0 umwandeln

$$f_1(x_1, x_2) = x_1^2 + x_2 - 11 = 0$$
$$f_2(x_1, x_2) = x_1 + x_2^2 - 7 = 0$$

In Gleichungssystem umwandeln

$$f(x) = \begin{pmatrix} f_1(x_1, x_2) \\ f_2(x_1, x_2) \end{pmatrix} = \begin{pmatrix} x_1^2 + x_2 - 11 \\ x_1 + x_2^2 - 7 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, f : \mathbb{R}^2 \to \mathbb{R}^2$$

Nullstellen einer nichtlinearen Funktion mit **n** unabhängigen Variablen mit Newton-Verfahren für Systeme

$$f : \mathbb{R} \to \mathbb{R}^n \text{ mit } f(x) = 0$$

## Partielle Ableitung

In einer Dimension

$$f'(x_0) = \lim_{\Delta x \to 0} \frac{f(x_0 + \Delta x) - f(x_0)}{\Delta x}$$

In mehreren Dimensionen
1. Ableitung nach x:

$$f_x = \frac{\partial f}{\partial x}(x, y) = \lim_{\Delta x \to 0} \frac{f(x + \Delta x, y) - f(x, y)}{\Delta x}$$

2. Ableitung nach y:

$$f_y = \frac{\partial f}{\partial y}(x, y) = \lim_{\Delta y \to 0} \frac{f(x, y + \Delta y) - f(x, y)}{\Delta y}$$

Beispiel

$$z = f(x, y) = 3xy^3 + 10x^2y + 5y + 3y \cdot \sin(5xy)$$

- $\frac{\partial f}{\partial x}(x, y) = 3 \cdot 1 \cdot y^3 + 10 \cdot 2x \cdot y + 0 + 3y \cdot \cos(5xy) \cdot 5 \cdot 1 \cdot y$
- $\frac{\partial f}{\partial y}(x, y) = 3x \cdot 3y^2 + 10x^2 \cdot 1 + 5 \cdot 1 + (3 \cdot 1 \cdot \sin(5xy) + 3y \cdot \cos(5xy) \cdot 5x \cdot 1)$

*Steigung der beiden Tangenten in $x$ – resp. $y$ –Richtung im Punkt* $(x_0, y_0) = (-1, 1)$

## Linearisierung von Funktionen mit mehreren Variablen

### Jacobi Matrix

Sei $f : \mathbb{R}^n \to \mathbb{R}^m$ mit $y = f(x)$ und $x = (x_1, x_2, ..., x_n)^T \in \mathbb{R}^n$. Die Jacobi-Matrix $Df(x)$ enthält sämtliche partiellen Ableitungen 1. Ordnung von $f$.

$$f(x) = \begin{pmatrix} y_1 = f_1(x) \\ y_2 = f_2(x) \\ \vdots \\ y_m = f_{m(x)} \end{pmatrix} Df(x) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(x) & \frac{\partial f_1}{\partial x_2}(x) & \dots & \frac{\partial f_1}{\partial x_n}(x) \\ \frac{\partial f_2}{\partial x_1}(x) & \frac{\partial f_2}{\partial x_2}(x) & \dots & \frac{\partial f_2}{\partial x_n}(x) \\ \vdots & \vdots & & \vdots \\ \frac{\partial f_m}{\partial x_1}(x) & \frac{\partial f_m}{\partial x_2}(x) & \dots & \frac{\partial f_m}{\partial x_n}(x) \end{pmatrix}$$

### Linearisieren - “verallgemeinerte Tangentengleichung”

Es gilt $\boldsymbol{f(x) \approx g(x)}$ in einer Umgebung des Vektors $x^{(0)}$

$$\boldsymbol{g(x) = f(x^{(0)}) + Df(x^{(0)}) \cdot (x - x^{(0)})}$$

- Für den speziellen Fall $f : \mathbb{R}^2 \to \mathbb{R}$ liefert die Linearisierung die Gleichung der **Tangentialebene**.

## Newton-Verfahren

$$\boldsymbol{x^{(n+1)} = x^{(n)} - (Df(x^{(n)}))^{-1} \cdot f(x^{(n)})}$$

Durch die Substitution $\boldsymbol{\delta^{(n)} := -(Df(x^{(n)}))^{-1} \cdot f(x^{(n)})}$ kann man die Berechnung der Inverse umgehen:

1. LGS von $\delta^{(n)}$ lösen ($\delta$ einfach als Vektor behandeln):
   $\boldsymbol{Df(x^{(n)})\delta^{(n)} = -f(x^{(n)})}$
2. Iterationschritt:
   $\boldsymbol{x^{(n+1)} := x^{(n)} + \delta^{(n)}}$

### Quadratische Konvergenz

Das Newton-Verfahren konvergiert **lokal** quadratisch gegen $\overline{x}$, wenn $\boldsymbol{Df(\overline{x})}$ regulär ist und $\boldsymbol{f}$ 3-mal stetig differenzierbar ist (3-mal partiell ableiten möglich).

### Vereinfachtes Newton-Verfahren

1. LGS von $\delta^{(n)}$ lösen, dabei $\boldsymbol{Df(x^{(0)})}$ **einmal** auswerten:
   $\boldsymbol{Df(x^{(0)})\delta^{(n)} = -f(x^{(n)})}$
2. Iterationschritt:
   $\boldsymbol{x^{(n+1)} := x^{(n)} + \delta^{(n)}}$

### Gedämpftes Newton-Verfahren

für $n = 0, 1, ...$ :
- Berechne $\boldsymbol{\delta^{(n)}}$ als Lösung des LGS
  $$\boldsymbol{Df(x^{(n)})\delta^{(n)} = -f(x^{(n)})}$$
- Finde das minimale $k \in \{0, 1, ..., k_{\max}\}$ mit
  $$\| f\left(x^{(n)} + \frac{\delta^{(n)}}{2^k}\right) \|_2 < \| f(x^{(n)}) \|_2$$
- Falls kein minimales $k$ gefunden werden kann, rechne mit $k = 0$ weiter
- Setze
  $$\boldsymbol{x^{(n+1)} := x^{(n)} + \frac{\delta^{(n)}}{2^k}}$$

## Interpolation

### Polynom-Interpolation: Lagrange Interpolationsformel

$P_n(x)$ lautet in der Lagrangeform

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

Die Lagrangepolynome vom Grad $n$ sind definiert :durch

$$l_i(x) = \prod_{j=0, j \neq i}^{n} \frac{x - x_j}{x_i - x_j} \quad i = 0, 1, \cdots, n$$

Wobei $n + 1$ die Anzahl Stützpunkte/Datenpunkte ist.

- Sind die $y_i$ Funktionswerte einer genügend oft stetig differenzierbaren Funktion $f$ (also $y_i = f(x_i)$), dann ist der Interpolationsfehler an einer Stelle $x$ gegeben durch

$$|f(x) - P_{n(x)}| \leq \frac{|(x - x_0)(x - x_1) \cdots (x - x_n)|}{(n + 1)!} \max_{x_0 \leq \xi \leq x_n} |f^{(n+1)}(\xi)|$$

### Spline-Interpolation

**Interpolation**
- $S_0(x_0) = y_0$
- $S_1(x_1) = y_1$
- $S_2(x_2) = y_2$
- $S_3(x_3) = y_3$

**Stetiger Übergang**
- $S_0(x_1) = S_1(x_1)$
- $S_1(x_2) = S_2(x_2)$

**Gleiche Steigung**
- $S_0'(x_1) = S_1'(x_1)$
- $S_1'(x_2) = S_2'(x_2)$

**Gleiche Krümmung**
- $S_0''(x_1) = S_1''(x_1)$
- $S_1''(x_2) = S_2''(x_2)$

**Zwei Bedingungen sind frei wählbar**
- Natürliche kubische Splinefunktion
  - $S_0''(x_0) = 0$
  - $S_2''(x_3) = 0$
- Periodische kubische Splinefunktion
  - $S_0'(x_0) = S_2'(x_3)$
  - $S_0''(x_0) = S_2''(x_3)$
- «not a knot» kubische Splinefunktion
  - $S_0'''(x_1) = S_1'''(x_1)$
  - $S_1'''(x_2) = S_2'''(x_2)$

### Spline-Interpolation: Natürliche kubische Spline

- Die Koeffizienten $a_i, b_i, c_i$ und $d_i$ der Polynome $S_{i(x)}$ für $i = 0, 1, ..., n - 1$ berechnen sich wie folgt:

1. $a_i = y_i$
2. $h_i = x_{i+1} - x_i$
3. $c_0 = 0, c_n = 0$
4. Berechnung der Koeffizienten $c_1, c_2, ..., c_{n-1}$ aus dem Gleichungssystem
   1. $i = 1$:
      $2(h_0 + h_1)c_1 + h_1 c_2 = 3\frac{y_2 - y_1}{h_1} - 3\frac{y_1 - y_0}{h_0}$
   2. falls $n \geq 4$ gilt für $i = 2, ..., n - 2$:
      $h_{i-1}c_{i-1} + 2(h_{i-1} + h_i)c_i + h_i c_{i+1} = 3\frac{y_{i+1} - y_i}{h_i} - 3\frac{y_i - y_{i-1}}{h_{i-1}}$
   3. $i = n - 1$:
      $h_{n-2}c_{n-2} + 2(h_{n-2} + h_{n-1})c_{n-1} = 3\frac{y_n - y_{n-1}}{h_{n-1}} - 3\frac{y_{n-1} - y_{n-2}}{h_{n-2}}$
5. $b_i = \frac{y_{i+1} - y_i}{h_i} - \frac{h_i}{3}(c_{i+1} + 2c_i)$
6. $d_i = \frac{1}{3h_i}(c_{i+1} - c_i)$

$$\boldsymbol{Ac = z}$$

$$\boldsymbol{A} = \begin{pmatrix} 2(h_0 + h_1) & h_1 & & & & \\ h_1 & 2(h_1 + h_2) & h_2 & & & \\ & h_2 & 2(h_2 + h_3) & h_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & h_{n-3} & 2(h_{n-3} + h_{n-2}) & h_{n-2} & \\ & & & h_{n-2} & 2(h_{n-2} + h_{n-1}) & \end{pmatrix}$$

$$\boldsymbol{c} = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_{n-1} \end{pmatrix}, \quad \boldsymbol{z} = \begin{pmatrix} 3\frac{y_2 - y_1}{h_1} - 3\frac{y_1 - y_0}{h_0} \\ 3\frac{y_3 - y_2}{h_2} - 3\frac{y_2 - y_1}{h_1} \\ \vdots \\ 3\frac{y_n - y_{n-1}}{h_{n-1}} - 3\frac{y_{n-1} - y_{n-2}}{h_{n-2}} \end{pmatrix}$$

## Ausgleichsrechnung

### Normalgleichung (lineares Ausgleichproblem / LSQ)

$$\boldsymbol{A^T A\lambda = A^T y}$$

$$\boldsymbol{A} = \begin{pmatrix} f_1(x_1) & f_2(x_1) & \cdots & f_{m(x_1)} \\ f_1(x_2) & f_2(x_2) & \cdots & f_{m(x_2)} \\ \vdots & & & \vdots \\ f_1(x_n) & f_2(x_n) & \cdots & f_{m(x_n)} \end{pmatrix}, \boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \boldsymbol{\lambda} = \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_m \end{pmatrix}$$

- $m \times m$ Matrix $\boldsymbol{A^T A}$ ist schlecht konditioniert, d.h. via Gauss-Algorithmus lösbar aber i.d.R. ungenau.
- Via QR-Zerlegung gibt es eine stabilere Methode, um die Normalengleichung zu lösen:
  $\boldsymbol{A = QR \Rightarrow R\lambda = Q^T y}$

### Nichtlineare Ausgleichsrechnung

Gegeben sind $n$ Wertepaare $(x_i, x_i)$ und Ansatzfunktionen $f_p = f_p(\lambda_1, ..., \lambda_m, x)$ mit $m$ Parametern $\lambda_j \in \mathbb{R}$.
Gesucht sind $m$ Parameter $\lambda = (\lambda_1, ..., \lambda_m)^T$ so, dass das **nichtlineare** Fehlerfunktional $E(f)$

$$E(f) = \sum_{i=1}^{m} (y_i - f_p(\lambda_1, ..., \lambda_m, x_i))^2 \to \min$$
$$\equiv \|y - f(\lambda)\|_2^2 \to \min$$

**minimal** wird unter allen zulässigen **Belegungen** der Parameter $\lambda$, wobei

$$f(\lambda) := f(\lambda_1, ..., \lambda_m) := \begin{pmatrix} f_1(\lambda_1, ..., \lambda_m) \\ f_2(\lambda_1, ..., \lambda_m) \\ \vdots \\ f_n(\lambda_1, ..., \lambda_m) \end{pmatrix} := \begin{pmatrix} f_{p(\lambda_1, ..., \lambda_m, x_1)} \\ f_{p(\lambda_1, ..., \lambda_m, x_2)} \\ \vdots \\ f_{p(\lambda_1, ..., \lambda_m, x_n)} \end{pmatrix}$$

$$y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \quad \lambda = \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_m \end{pmatrix}$$

Das System $A\lambda = y$ heisst Fehlergleichungssystem.

### Gauss Newton

Berechne die Funktion $g(\lambda)$ sowie deren Jacobi-Matrix

$$g(\lambda) := y - f(\lambda), \quad Dg(\lambda)$$

Für $k = 0, 1...$:
**Schritt 1**: Berechne $\delta^{(k)}$ als Lösung des linearen Ausgleichsproblems

$$\min \left\| g(\lambda^{(k)}) + Dg(\lambda^{(k)}) \cdot \delta^{(k)} \right\|_2^2$$

d.h. löse konkret das folgende Normalgleichungssystem nach $\delta^{(k)}$ auf

$$Dg(\lambda^{(k)})^T Dg(\lambda^{(k)})\delta^{(k)} = -Dg(\lambda^{(k)})^T \cdot g(\lambda^{(k)})$$

Dies wird am stabilsten mit der QR-Zerlegung von $Dg(\delta^{(k)})$ erreicht

$$Dg(\lambda^{(k)}) = Q^{(k)} R^{(k)}, \quad R^{(k)}\delta^{(k)} = -Q^{(k)T} g(\lambda^{(k)})$$

**Schritt 2**: (Nur beim gedämpften Gauss-Newton)
Finde das minimale $p \in \{0, 1, ..., p_{\max}\}$ mit

$$\left\| g\left(\underbrace{\lambda^{(k)} + \frac{\delta^{(k)}}{2^p}}_{\lambda^{(k+1)}}\right) \right\|_2^2 < \|g(\lambda^{(k)})\|_2^2$$

Falls kein minimales $p$ gefunden werden kann, rechne mit $p = 0$ weiter
**Schritt 3**: (Gauss-Newton)

$$\lambda^{(k+1)} = \lambda^{(k)} + \delta^{(k)}$$

**Schritt 3**: (gedämpftes Gauss-Newton)

$$\lambda^{(k+1)} = \lambda^{(k)} + \frac{\delta^{(k)}}{2^p}$$

## Numerische Integration

### Rechteckregel / Trapezregel

Zur Approximation von

$$\int_a^b f(x)dx$$

$$Rf = f\left(\frac{a+b}{2}\right) c \cdot (b - a)$$

$$Tf = \frac{f(a) + f(b)}{2} c \cdot (b - a)$$

### Summierte Rechteckregel / Trapezregel

**Konstante** Breite $h = x_{i+1} - x_i = \frac{b-a}{n}$ und $x_i = a + i \cdot h$ für $i = 0, ..., n - 1$ und $x_n = b$.

$$Rf(h) = h \cdot \sum_{i=0}^{n-1} f\left(x_i + \frac{h}{2}\right)$$

$$Tf(h) = h \cdot \left(\frac{f(a) + f(b)}{2} + \sum_{i=1}^{n-1} f(x_i)\right)$$

### Summierte Simpsonregel

$$Sf(h) = \frac{1}{3}(Tf(h) + 2Rf(h))$$

### Fehlerabschätzung

$$\left| \int_a^b f(x)\,\mathrm{d}x - Rf(h) \right| \leq \frac{h^2}{24}(b - a) \cdot \max_{x \in [a,b]} |f''(x)|$$

$$\left| \int_a^b f(x)\,\mathrm{d}x - Tf(h) \right| \leq \frac{h^2}{12}(b - a) \cdot \max_{x \in [a,b]} |f''(x)|$$

$$\left| \int_a^b f(x)\,\mathrm{d}x - Sf(h) \right| \leq \frac{h^4}{2880}(b - a) \cdot \max_{x \in [a,b]} |f^{(4)}(x)|$$

### Gauss Formeln für Grad $n = 1, 2, 3$

Für ungleichmässige Stützstellen $x_i$ :

$$n = 1; \quad G_1 f = (b - a) \cdot f\left(\frac{b + a}{2}\right)$$

$$n = 2$$

$$G_2 f = \frac{b-a}{2}\left[f\left(-\tfrac{1}{\sqrt{3}} \cdot \tfrac{b-a}{2} + \tfrac{b+a}{2}\right) + f\left(\tfrac{1}{\sqrt{3}} \cdot \tfrac{b-a}{2} + \tfrac{b+a}{2}\right)\right]$$

$$n = 3$$

$$G_3 f = \frac{b-a}{2}\left[\frac{5}{9} \cdot f\left(-\sqrt{0.6} \cdot \frac{b-a}{2} + \frac{b+a}{2}\right) + \frac{8}{9} \cdot f\left(\frac{b+a}{2}\right)\right] + \frac{b-a}{2}\left[\frac{5}{9} \cdot f\left(\sqrt{0.6} \cdot \frac{b-a}{2} + \frac{b+a}{2}\right)\right]$$

### Romberg-Interpolation

$$T_{jk} = \frac{4^k \cdot T_{j+1,k-1} - T_{j,k-1}}{4^k - 1}$$

$$T_{j0} = Tf(h_j)$$

Mit $h_j = \frac{b-a}{2^j}$ und $n_j = 2^j$ und $x_i = a + ih_j$

### Romberg Schema für $m = 3$

$$\begin{array}{llll} \boldsymbol{T_{j0}} & \boldsymbol{T_{j1}} & \boldsymbol{T_{j2}} & \boldsymbol{T_{j3}} \\ T_{00} & & & \\ & \searrow & & \\ & T_{01} = \frac{4T_{10} - T_{00}}{3} & & \\ \nearrow & & \searrow & \\ T_{10} & & T_{02} = \frac{16T_{11} - T_{01}}{15} & \\ \searrow & \nearrow & & \searrow \\ & T_{11} = \frac{4T_{20} - T_{10}}{3} & & T_{03} = \frac{64T_{12} - T_{02}}{63} \\ \nearrow & & \searrow & \\ T_{20} & & T_{12} = \frac{16T_{21} - T_{11}}{15} & \\ \searrow & \nearrow & & \\ & T_{21} = \frac{4T_{30} - T_{20}}{3} & & \\ \nearrow & & & \\ T_{30} & & & \end{array}$$

Mit $j = 0$ in erster Spalte. Diese mit Tf(h_j) berechnen, dann nach rechts rechnen.

## Differentialgleichungen

### Euler-Verfahren

$x \in [a, b]$ Anfangswertproblem

$$y' = f(x, y) \quad \text{mit} \quad y(a) = y_0.$$
$$x_{i+1} = x_i + h$$
$$y_{i+1} = y_i + h \cdot f(x_i, y_i)$$

Mit $x_0 = a$, $x_i = a + ih$ für $i = 0, ..., n - 1$ und $h = \frac{b-a}{n}$

### Mittelpunk-Verfahren

$$x_{h/2} = x_i + \frac{h}{2}$$
$$y_{h/2} = y_i + \frac{h}{2} \cdot f(x_i, y_i)$$
$$x_{i+1} = x_i + h$$
$$y_{i+1} = y_i + h \cdot f\left(x_{h/2}, y_{h/2}\right)$$

### Modifiziertes Euler-Verfahren (Heun)

Klassische Euler-Verfahren durchführen und die erste Tangentensteigung in der Variable $k_1$ speichern:

$$x_{i+1} = x_i + h$$
$$y_{i+1}^{\text{Euler}} = y_i + h \cdot f(x_i, y_i)$$
$$k_1 = f(x_i, y_i)$$

Zweite Tangentensteigung am Punkt $(x_{i+1}, y_{i+1}^{\text{Euler}})$ berechnen und in $k_2$ speichern:

$$k_2 = f(x_{i+1}, y_{i+1}^{\text{Euler}})$$

Durchschnitt der Steigungen $\frac{k_1 + k_2}{2}$ bilden und damit einen Schritt $h$ vom ursprünglichen Punkt $(x_i, y_i)$ machen:

$$x_{i+1} = x_i + h$$
$$y_{i+1} = y_i + h \cdot \frac{k_1 + k_2}{2}$$

Schritte wiederholen ausgehend von $x_0 = a$.

### Lokaler Fehler

Der **lokale Fehler** (nach einer Iteration) ist definiert als die Differenz zwischen dem exakten Wert und der Näherung:

$$\varphi(x_i, h) := y(x_{i+1}) - y_{i+1}$$

Num. Verfahren hat die **Konsistenzordnung p** falls gilt:

$$|\varphi(x_i, h)| \leq C \cdot h^{p+1}$$

für genügend kleine $h$ und einer Konstante $C > 0$, die von der Differentialgleichung abhängt.

### Globaler Fehler

Der **gesamt/globale Fehler** (nach $n$ Iterationen) ist definiert als die Diff. vom exakten Wert und der Näherung:

$$y(x_n) - y_n$$

Num Verfahren hat die **Konvergenzordnung p** falls gilt:

$$|y(x_n) - y_n| \leq C \cdot h^p$$

für genügend kleine $h$ und einer Konstante $C > 0$, die von der Differenzialgleichung abhängt.

### Klassisches vierstufiges Runge-Kutta Verfahren

$$\begin{aligned} k_1 &= f(x_i, y_i) \\ k_2 &= f(x_i + \tfrac{h}{2}, y_i + \tfrac{h}{2}k_1) \\ k_3 &= f(x_i + \tfrac{h}{2}, y_i + \tfrac{h}{2}k_2) \\ k_4 &= f(x_i + h, y_i + hk_3) \\ x_{i+1} &= x_i + h \\ y_{i+1} &= y_i + h \cdot \tfrac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \end{aligned}$$

### s-stufiges Runge-Kutta Verfahren

$$k_n = f\left(x_i + c_n h, y_i + h \sum_{m=1}^{n-1} a_{nm} k_m\right)$$

für $n = 1, ..., s$

$$y_{i+1} = y_i + h \sum_{n=1}^{s} b_n k_n$$

Butcher Tableau

$$\begin{array}{c|ccccc} c_1 & & & & & \\ c_2 & a_{21} & & & & \\ c_3 & a_{31} & a_{32} & & & \\ \vdots & & & & & \\ c_n & a_{n1} & a_{n2} & \dots & a_{n,n-1} & \\ \vdots & & & & & \\ c_s & a_{s1} & a_{s2} & \dots & a_{s,s-1} & \\ \hline & b_1 & b_2 & \dots & b_{s-1} & b_s \end{array}$$

- Hierbei ist $s \in \mathbb{N}$ die Stufenzahl und $a_{nm}$, $b_n$, $c_n$ sind Konstanten. Die Konsistenz- und Konvergenzordnung hängt von der Wahl dieser Konstanten ab.

### DGL Zurückführen auf 1. Ordnung (Beispiel 3. Ordnung)

$g(x)$ ist die Störfunktion (inhomogen, daher mit $+\boldsymbol{b}$)

$$y''' = c_0 y + c_1 y' + c_2 y'' + g(x)$$

Zustandsvektor definieren:

$$\boldsymbol{z} = \begin{pmatrix} z_0 \\ z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} y \\ y' \\ y'' \end{pmatrix}$$

- $z_0' = z_1$
- $z_1' = z_2$
- $z_2' = c_0 z_0 + c_1 z_1 + c_2 z_2 + g(x)$

Matrixform bilden ($\boldsymbol{z' = A \cdot z + b}$):

$$\begin{pmatrix} z_0' \\ z_1' \\ z_2' \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ c_0 & c_1 & c_2 \end{pmatrix} \cdot \begin{pmatrix} z_0 \\ z_1 \\ z_2 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ g(x) \end{pmatrix}$$

**Source Code**

https://github.com/josias-r/zhaw-sem5-summaries