

II. Explizite Einschrittverfahren

- Ziele:
- Anfangswertproblem (gew. Diff.-Glb. + Anfangsbedingungen)
 - Lösbarkeit?
 - explizite Einschrittverfahren (Euler, Runge-Kutta)
 - Genauigkeit (Diskretisierungsfehler)

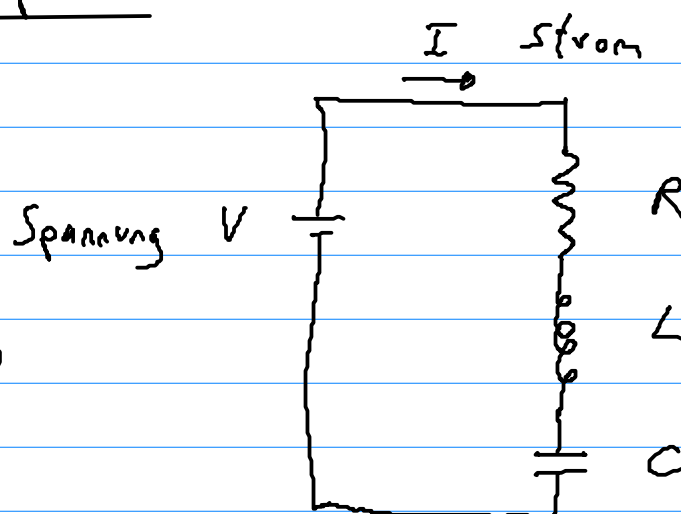
Wozu: Viele Anwendungen...

II.A Motivation und Beispiele

(1) Schwingkreis

$$\ddot{I} + \frac{R}{L} \dot{I} + \frac{1}{LC} I = 0$$

$\left(\frac{d^2 I}{dt^2} \right) \quad \left(\frac{dI}{dt} \right)$



(Skalare) gewöhnliche Differentialgleichung
(DGL) 2. Ordnung für Strom $I(t)$

Oft: ODE für Ordinary Differential Equation

(2) Molekular-Dynamik (MD)

$$m_i \cdot \ddot{\vec{x}}_i = - \vec{\nabla}_i U(\vec{x}_1, \dots, \vec{x}_N), \quad i=1, \dots, N$$

wobei N ... Anzahl Atome

\vec{x}_i ... Position des i -ten Atom

U ... Potential (hängt von allen Atomen ab!)

System gewöhnlicher DGLen 2. Ordnung für die Positionen der Atome $\vec{x}(t)$ (Im Prinzip einfach nur Newton's Bewegungsgleichungen).

(3) Maxwell-Gleichungen

$$\vec{\nabla} \times \vec{H} = \vec{j} + \frac{\partial \vec{D}}{\partial t}$$

magnetische Feldstärke Stromdichte elektrische Flussdichte

$$\vec{\nabla} \times \vec{E} = - \frac{\partial \vec{B}}{\partial t}$$

magnetische Flussdichte

elektrische Feldstärke

$$\vec{\nabla} \cdot \vec{D} = \rho$$

elektrische Ladungsdichte

$$\vec{\nabla} \cdot \vec{B} = 0$$

System von partiellen DGL

(weil partielle Ableitungen auftauchen!)

Durch diskretisieren der Orts-Ableitungen
 (z.B. mit finiten Differenzen/Volumen/Elementen)
 erhält man ein (großes) System von
 gewöhnlichen DGLen 1. Ordnung
 ↖ diese Vorlesung

oft: PDE für Partial Differential Equation

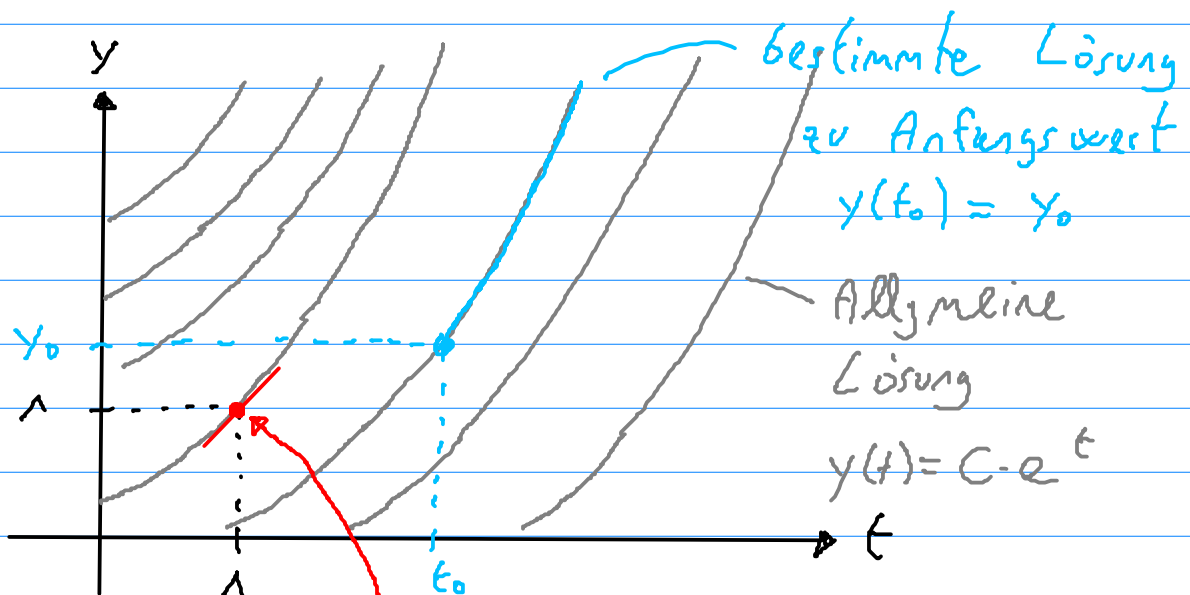
Zur Illustration betrachten wir eine
 (die?!) einfache skalare DGL

$$\dot{y}(t) = y(t) = f(t, y(t))$$

↖ sog. rechte Seite der DGL

(skalare)

Eine \checkmark DGL lässt sich graphisch mit einem
 Richtungsfeld/Vektorfeld darstellen:

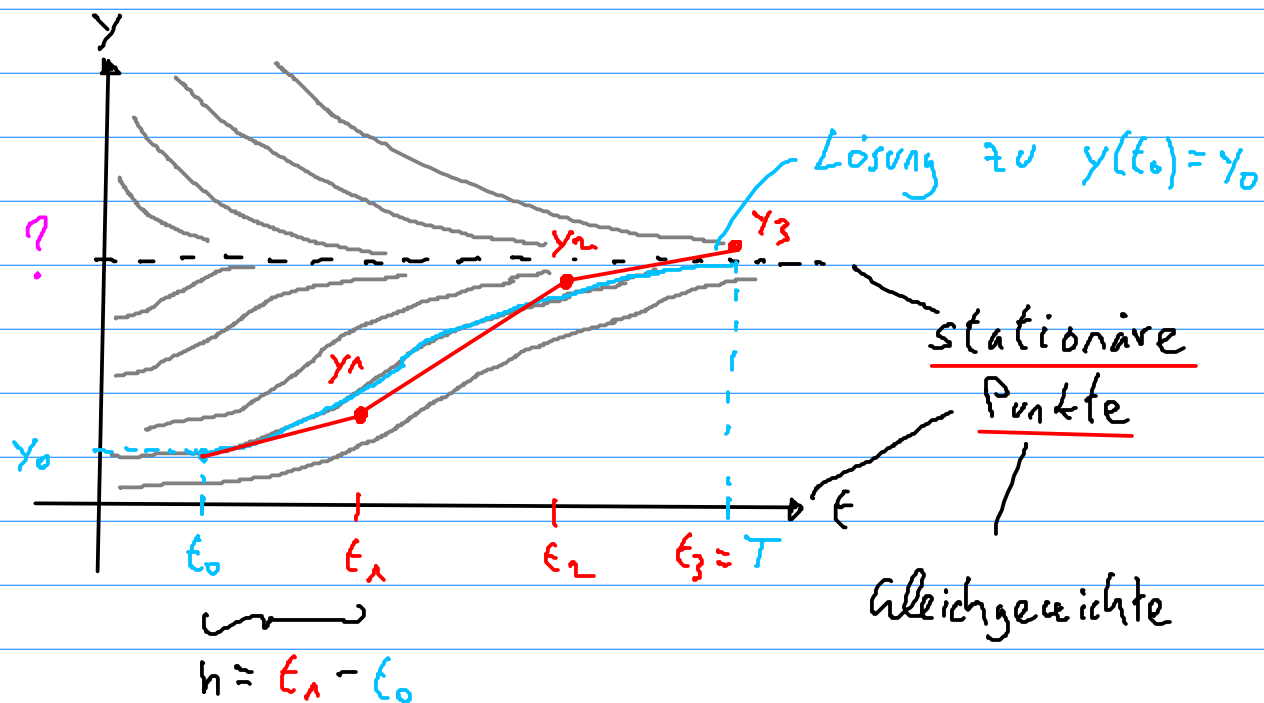


Steigung von $y(t)$ in $(t=1, y=1)$ ist 1

Eine etwas "interessantere" (skalare) DGL:

$$\dot{y}(t) = a \cdot y(t) - b \cdot y^2(t) = (a - b \cdot y(t)) y(t)$$

Graphisch:



Einfache Approximation der Lösung

① Diskretisiere das Zeitintervall $I = [t_0, T]$

$$t_k = t_0 + k \cdot h, \quad k = 0, 1, \dots, N$$

mit $h = \frac{T - t_0}{N}$

② Setze Anfangswert $y_0 = y(t_0)$

③ Berechne für $k = 0, 1, \dots, N-1$

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

Dies ist das Euler Verfahren

oder expliziter Euler

oder Vorwärts Euler

(Euler Forward)

I.A. stellen sich nun folgende Fragen:

(i) Macht es Sinn (approx.) Lösungen zu suchen? (Bedingungen an die rechte Seite $f(t, y(t))$ für Existenz und Eindeutigkeit)

(ii) DGLen höherer Ordnung? (z.B. (1) & (2) oben)

(iii) Konvergiert das Euler Verfahren gegen die exakten Lösungen für $h \rightarrow 0$?
Wie schnell?

(iv) Gibt's "bessere" Methoden?

II.2 Grundbegriffe

Def.: Ein skalares Anfangswertproblem (AWP) erster Ordnung:

Finde eine Funktion $y(t)$ einer Variablen (z.B. die Zeit) mit

$$\dot{y}(t) = f(t, y(t)) \quad (\text{gew. DGL 1. Ordnung})$$

auf dem Intervall $I = [t_0, T]$ und

$$y(t_0) = y_0 \quad (\text{Anfangswert (AW)})$$

auch Anfangsbedingung

Oft hat man nicht nur eine DGL, sondern ein ganzes System

Def.: Ein (allgemeines) AWP erster Ordnung:

Finde die Funktionen $y_1(t), \dots, y_n(t)$ einer Variablen (z.B. Zeit) mit

$$\begin{aligned} \dot{y}_1(t) &= f_1(t, y_1(t), \dots, y_n(t)) \\ &\vdots \\ \dot{y}_n(t) &= f_n(t, y_1(t), \dots, y_n(t)) \end{aligned} \quad \left(\begin{array}{l} n \text{ gew.} \\ \text{DGLen} \\ \text{erster} \\ \text{Ordnung} \end{array} \right)$$

auf dem Intervall $I = [t_0, T]$ und \rightarrow

$$\begin{aligned} y_1(t_0) &= y_{1,0} \\ &\vdots \\ y_n(t_0) &= y_{n,0} \end{aligned} \quad (n \text{ AWe})$$

Kürzer: $\dot{\vec{y}}(t) = \vec{F}(t, \vec{y})$

$$\vec{y}(t_0) = \vec{y}_0$$

wobei $\vec{y}(t) = \begin{pmatrix} y_1(t) \\ \vdots \\ y_n(t) \end{pmatrix}$, $\vec{y}_0 = \begin{pmatrix} y_{1,0} \\ \vdots \\ y_{n,0} \end{pmatrix}$

und $\vec{F}(t, \vec{y}) = \begin{pmatrix} f_1(t, y_1(t), \dots, y_n(t)) \\ \vdots \\ f_n(t, y_1(t), \dots, y_n(t)) \end{pmatrix}$

Bsp.: (*) Lineares DGL System

$$\dot{\vec{y}}(t) = A \vec{y}(t), \quad \vec{y} \in \mathbb{R}^n, \quad A \in \mathbb{R}^{n \times n}$$

$$\vec{y}(t_0) = \vec{y}_0 \quad \begin{matrix} / \\ n \times n \text{ Matrix} \end{matrix}$$

Lösung: $\vec{y}(t) = e^{A(t-t_0)} \vec{y}_0$
 $\underbrace{\hspace{1.5cm}}_{\text{Matrix Exp. Fkt.}}$

$$= \left(\sum_{i=0}^{\infty} \frac{1}{i!} A^i (t-t_0)^i \right) \vec{y}_0$$

8

Def.: Ein (skalares) AWP n-ter Ordnung:

finde die Funktion $y(t)$ einer Variablen (z.B. Zeit) mit

$$y^{(n)}(t) = f(t, y(t), \dot{y}(t), \ddot{y}(t), \dots, y^{(n-1)}(t))$$

(skalare gew. DGL n-ter Ordnung)

auf dem Intervall $I = [t_0, T]$ und

$$y(t_0) = y_0$$

$$\dot{y}(t_0) = \dot{y}_0$$

$$\ddot{y}(t_0) = \ddot{y}_0$$

$$\vdots$$

$$y^{(n-1)}(t_0) = y_0^{(n-1)}$$

(n AWP für y und die $(n-1)$ -ten Ableitungen)

Bsp.: (S) Newton's Bewegungsgleichungen

$$m \cdot \ddot{x} = F \quad \left(\begin{array}{l} \text{gek. DGL 2-ter} \\ \text{Ordnung} \end{array} \right)$$

Masse Beschleunigung Kraft

mit

$$x(t_0) = x_0 \quad (\text{Anfangs-Position})$$

$$\dot{x}(t_0) = \dot{x}_0 (= v_0) \quad (\text{Anfangs-Geschw.})$$

Im folgenden werden wir nur numerische Methoden für AWP erster Ordnung betrachten. Jedes AWP n -ter Ordnung lässt sich in ein System AWP erster Ordnung umschreiben:

Reduktion auf ein System erster Ordnung

Gegeben eine gew. DGL n -ter Ordnung

$$y^{(n)}(t) = f(t, y, \dot{y}, \dots, y^{(n-1)})$$

Wir definieren

$$z_0(t) = y(t)$$

$$z_1(t) = \dot{y}(t) = \dot{z}_0(t)$$

$$z_2(t) = \ddot{y}(t) = \dot{z}_1(t)$$

⋮

$$z_{n-1}(t) = y^{(n-1)}(t) = \dot{z}_{n-2}(t)$$

Beachte: $\dot{z}_{n-1}(t) = ?$

Dann können wir

$$y^{(n)}(t) = f(t, y(t), \dot{y}(t), \ddot{y}(t), \dots, y^{(n-1)}(t))$$

$$\left\{ \begin{array}{l} \dot{z}_{n-1}(t) = f(t, z_0(t), z_1(t), z_2(t), \dots, z_{n-1}(t)) \\ \text{gew. DGL erster Ordnung!} \end{array} \right.$$

Obige Definitionen sind auch $(n-1)$ gew. DGL erster Ordnung. Insgesamt haben wir also folgendes System von n gew. DGL erster Ordnung

$$\dot{\vec{z}}(t) = \vec{g}(t, \vec{z}(t))$$

wobei

$$\vec{z}(t) = \begin{pmatrix} z_0(t) \\ z_1(t) \\ \vdots \\ z_{n-2}(t) \\ z_{n-1}(t) \end{pmatrix}$$

und

$$\vec{g}(t, \vec{z}(t)) = \begin{pmatrix} z_1(t) \\ z_2(t) \\ \vdots \\ z_{n-1}(t) \\ f(t, z_0(t), z_1(t), \dots, z_{n-1}(t)) \end{pmatrix}$$

Für die AWe ergibt sich

$$\vec{z}(t_0) = \vec{z}_0 = \begin{pmatrix} y_0 \\ \dot{y}_0 \\ \ddot{y}_0 \\ \vdots \\ y_0^{(n-1)} \end{pmatrix}$$

Bsp.: (6) $\ddot{y}(t) = f(t, y(t), \dot{y}(t))$

$$y(t_0) = y_0$$

$$\dot{y}(t_0) = \dot{y}_0$$

Reduktion auf System erster Ordnung:

$$z_0(t) = y(t)$$

$$z_1(t) = \dot{y}(t) = \dot{z}_0(t)$$

$$z_2(t) = \ddot{y}(t) = \dot{z}_1(t) = f(t, z_0(t), z_1(t))$$

Somit $\dot{\vec{z}} = \vec{g}(t, \vec{z}(t))$

wobei

$$\vec{z} = \begin{pmatrix} z_0(t) \\ z_1(t) \end{pmatrix}, \quad \vec{g}(t, \vec{z}(t)) = \begin{pmatrix} z_1 \\ f(t, z_0(t), z_1(t)) \end{pmatrix}$$

und AWe

$$\vec{z}(t_0) = \vec{z}_0 = \begin{pmatrix} y_0 \\ \dot{y}_0 \end{pmatrix}$$

Genau gleich kann man ein System von n gew. DGLen n -ter Ordnung auf ein System von $n \cdot n$ gew. DGLen erster Ordnung reduzieren.

Def.: Eine gew. DGL heisst autonom, falls die rechte Seite die Form $\vec{f} = \vec{f}(\vec{y}(t))$ hat (anstatt $\vec{f} = \vec{f}(t, \vec{y}(t))$).

Autonomisieren von DGLen

Betrachte folgende gew. DGL

$$\dot{\vec{y}} = \vec{f}(t, \vec{y}(t)) \quad , \quad \vec{y} \in \mathbb{R}^n$$

Durch einführen der neuen Variabel

$$\vec{z}(t) = \begin{pmatrix} \vec{y}(t) \\ t \end{pmatrix} = \begin{pmatrix} z_1 \\ \vdots \\ z_{n+1} \end{pmatrix}$$

und der rechten Seite

$$\vec{g}(\vec{z}(t)) = \begin{pmatrix} \vec{f}(t, \vec{y}(t)) \\ 1 \end{pmatrix} = \begin{pmatrix} \vec{f}(z_{n+1}, \vec{z}_1) \\ 1 \end{pmatrix}$$

erhält man eine autonome gew. DGL

$$\dot{\vec{z}} = \vec{g}(\vec{z})$$

Bsp.: (+) $y'(t) = y(t)^2 + t^2$

$$\vec{z}(t) = \begin{pmatrix} z_1(t) \\ z_2(t) \end{pmatrix} = \begin{pmatrix} y(t) \\ t \end{pmatrix}$$

$$\vec{g}(\vec{z}(t)) = \begin{pmatrix} y(t)^2 + t^2 \\ 1 \end{pmatrix} = \begin{pmatrix} z_1(t)^2 + z_2(t)^2 \\ 1 \end{pmatrix}$$

$\leadsto \dot{\vec{z}}(t) = \vec{g}(\vec{z}(t))$ autonom ✓.

II.2.1 Existenz, Eindeutigkeit und Stabilität

Im folgenden wollen wir uns ein paar theoretische Fragen bez. AWPen anschauen:

- (i) Was muss erfüllt sein, dass es überhaupt Lösungen gibt? (Existenz)
- (ii) Gibt's mehrere Lösungen? Unter welcher Bedingung gibt es nur eine? (Eindeutigkeit)
- (iii) Wie hängt die Lösung vom AWP ab? (Stabilität)

Betrachten wir folgendes (allgemeines) AWP

$$\dot{\vec{y}}(t) = \vec{f}(t, \vec{y}(t)) \quad (\text{System von DGLen})$$

$$\vec{y}(t_0) = \vec{y}_0 \quad (\text{AWP})$$

wobei

$$\vec{y} : I = [t_0, T] \subset \mathbb{R} \rightarrow D \subset \mathbb{R}^n$$

(Zeit) Intervall
' Zustands- oder
Phasen-raum

$$\vec{f} : I \times D \rightarrow \mathbb{R}^n$$

Um (i) und (ii) von oben sicherzustellen, benötigen wir einen etwas stärkeren Begriff der Stetigkeit

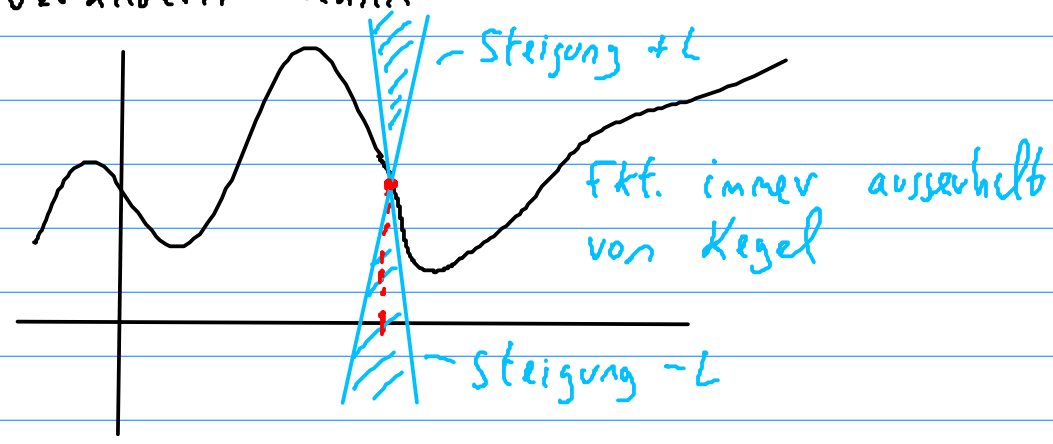
Def.: Eine Funktion $\vec{f}: I \times D \rightarrow \mathbb{R}^n$ ist Lipschitz-stetig in (Variabel) \vec{y} mit der Lipschitz-Konstanten $L \geq 0$, wenn für alle $t \in I$ und $\vec{y}, \vec{z} \in D$ gilt

$$\|\vec{f}(t, \vec{y}) - \vec{f}(t, \vec{z})\| \leq L \|\vec{y} - \vec{z}\|$$

Hier ist $\|\cdot\|$ eine Norm.

(1-, euklidische, max-Norm ...)

Bem.: (i) Lipschitz-Stetigkeit beschränkt wie stark eine Funktion um einen Punkt sich verändern kann



- (ii) Stetige diff.'bare Funktionen sind Lipschitz-stetig: Setze $L = \max_{y \in (a,b)} |f'(y)|$
- (iii) Auch nicht diff.'bare Funktionen können Lipschitz-stetig sein
z.B.: $f(y) = |y|$
- (iv) Manchmal auch Lipschitz-Bedingung genannt

Frage: Ist $f(y) = \sqrt{y}$, $y \in \mathbb{R}^+$, überall Lipschitz-stetig?

Der folgende Satz stellt nun (i) und (ii) sicher

Satz II.1: (Picard-Lindelöf)

Sei \vec{f} stetig in (t, \vec{y}) und Lipschitz-stetig in \vec{y} auf $[t_0, t_0 + \delta] \times D$, mit $\delta > 0$ und D eine Umgebung vom AWP \vec{y}_0 .

Dann existiert eine eindeutige Lösung $\vec{y}(t)$ des AWP

$$\vec{y}'(t) = \vec{f}(t, \vec{y}(t)), \quad \vec{y}(t_0) = \vec{y}_0$$

für zumindest eine kurze Zeit $[t_0, t_0 + \epsilon]$, $\epsilon > 0$.

Bsp.: (8) $\dot{y}(t) = 2\sqrt{|y(t)|}$

$$y(0) = 0$$

Lösungen: $y(t) = 0$

$$y(t) = t \cdot |t|$$

Grund: $f(t, y) = 2\sqrt{|y|}$ stetig, aber nicht Lipschitz-stetig in $y=0$!

(9) Bsp. (8) mit $y(1) = 1$

$\leadsto y(t) = t \cdot |t|$ eindeutige Lösung!

(10) $\dot{y}(t) = y(t)^2$

$$y(0) = y_0 > 0$$

Lösung: $y(t) = \frac{y_0}{1 - y_0 \cdot t}$

Aber nur für $0 \leq t < \frac{1}{y_0}$!

(... "zumindest eine kurze Zeit" ...)

Grund: Lipschitz-Konstante L unbeschränkt

Der folgende Satz gibt Aufschluss über die eingehende Frage (iii) bez. der Abhängigkeit der Lösung von der AW:

Satz II.2: Die Funktion \vec{f} sei Lipschitz-stetig mit Lipschitz-Konstanten L in einer Umgebung der AKe \vec{y}_0, \vec{z}_0 und seien $\vec{y}(t), \vec{z}(t)$ die Lösungen der jeweiligen AWP's.

Dann gilt für $t \in [t_0, t_0 + \varepsilon]$

$$\|\vec{y}(t) - \vec{z}(t)\| \leq e^{L(t-t_0)} \|\vec{y}_0 - \vec{z}_0\|$$

Bem.: Satz II.2 stellt sicher, dass die Lösungen stetig vom AW abhängen.

Aber Lösungen können exponentiell in der Zeit auseinander driften

Bsp.: (11) $\dot{y}(t) = \lambda y(t), y(t_0) = y_0 \rightsquigarrow y(t) = y_0 e^{\lambda(t-t_0)}$

$\dot{z}(t) = \lambda z(t), z(t_0) = z_0 \rightsquigarrow z(t) = z_0 \cdot e^{\lambda(t-t_0)}$

Frage: Lipschitz-Konstante $L = ?$

$$\leadsto y(t) - z(t) = e^{\lambda(t-t_0)} (y_0 - z_0)$$

↑
Vgl. Satz II.2

II.3 Runge-Kutta Verfahren

Betrachten wir das (skalare) AWP

$$\dot{y}(t) = f(t, y(t))$$

$$y(t_0) = y_0$$

Die Lösung können wir formal durch Integration erhalten

$$y(t) = y_0 + \int_{t_0}^t f(\tau, y(\tau)) d\tau$$

Integral-Gleichung... Nicht unbedingt "besser" wie mit der gew. DGL...

Idee: Verwende Quadratur für das Integral der rechten Seite f

Konkret, versuchen wir die Lösung ausgehend von t_0, y_0 an einem Punkt t_1, \vec{y}_1 , wobei $t_1 = t_0 + h$, mittels Quadratur zu approx.:

$$y(t_1) = y_0 + \int_{t_0}^{t_1} f(\tau, y(\tau)) d\tau$$

$$= y_0 + h \cdot \int_0^1 f(t_0 + h\tau, y(t_0 + h\tau)) d\tau$$

... grenzen verschoben und skaliert ...

$$\approx y_0 + h \cdot \sum_{i=1}^s w_i \cdot f(t_0 + h \cdot c_i, y(t_0 + h \cdot c_i))$$

| Quadratur Gewichte / Knoten

Problem: $y(t_0 + h \cdot c_i)$ immer noch unbekannt!
 \rightsquigarrow Müssen auch approx. werden ...

Versuchen wir es mit der MR:

$$y(t_1) \approx y_0 + h \cdot f\left(t_0 + \frac{h}{2}, y\left(t_0 + \frac{h}{2}\right)\right)$$

Wie könnte man eine Approx. von $y\left(t_0 + \frac{h}{2}\right)$ berechnen? m.a. z.B. mit dem expliziten Euler-Verfahren!

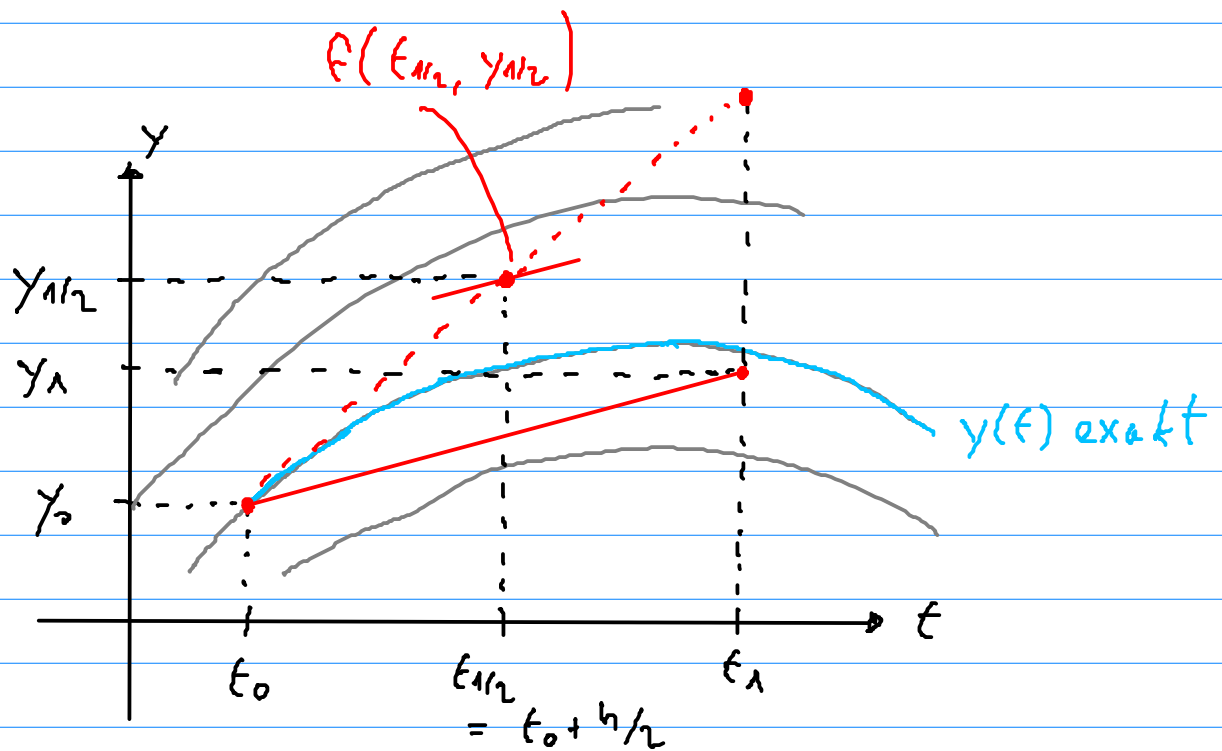
$$y\left(t_0 + \frac{h}{2}\right) \approx y_0 + \frac{h}{2} f\left(t_0, y_0\right) = y_{1/2}$$

\downarrow
 $y(t_0) = y_0$ AW

Setzen wir dies oben ein erhalten wir

$$y(t_1) \approx y_0 + h \cdot f\left(t_0 + \frac{h}{2}, y_{1/2}\right) = y_1$$

Graphisch:



Zusammenfassend, erhalten wir folgende Methode für AWP:

$$k_1 = f(t_j, y_j)$$

$$k_2 = f(t_j + h/2, y_j + h/2 \cdot k_1)$$

$$y_{j+1} = y_j + h \cdot k_2$$

für $j = 0, 1, \dots, N-1$. Dieses Verfahren ist bekannt als die verbesserte Polygonzug-Methode von Euler.

Oder auch: modifiziertes Euler Verfahren
explizite Mittelpunkts-Methode

Eine andere Möglichkeit wäre als QR
die TR zu verwenden

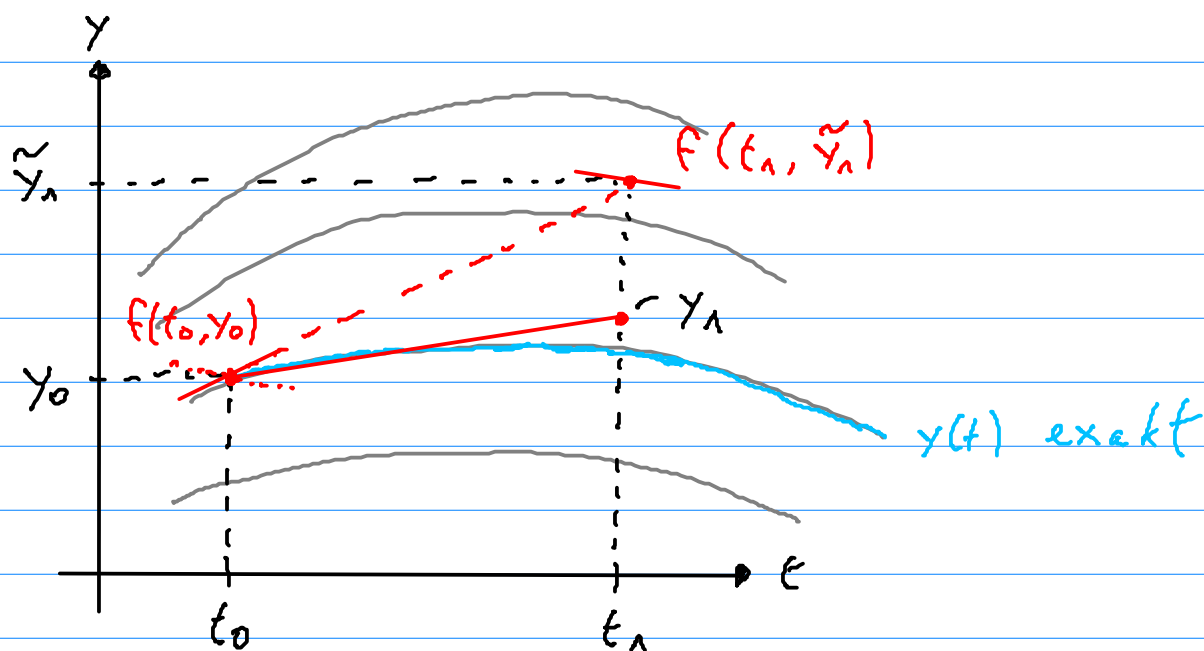
$$y(t_n) \approx y_0 + \frac{h}{2} \left(f(t_0, y_0) + f(t_n, y(t_n)) \right)$$

↑
?

Wie vorhin, approx. wir $y(t_n)$ mit dem
Euler-Verfahren

$$y(t_n) \approx y_0 + h \cdot f(t_0, y_0) = \tilde{y}_n$$

Graphisch:



Wir erhalten folgendes Verfahren

$$k_1 = f(t_j, y_j)$$

$$k_2 = f(t_j + h, y_j + h \cdot k_1)$$

$$y_{j+1} = y_j + \frac{h}{2} (k_1 + k_2)$$

für $j = 0, 1, \dots, N-1$. Dies ist das Verfahren von Heun.

Oder auch explizite Trapez-Methode.

25.03.19

Alle Verfahren die wir bis jetzt kennengelernt haben sind Teil einer grossen Familie von Einschrittverfahren (ESV):

Runge-Kutta Verfahren

Ein s-stufiges Runge-Kutta (Einschritt-) Verfahren (RK-ESV) ist definiert durch

$$y_{j+1} = y_j + h \cdot \sum_{i=1}^s b_i \cdot k_i$$

wobei die Stufen/Steigungen

$$k_i = f\left(t_j + c_i \cdot h, y_j + h \cdot \sum_{l=1}^s a_{il} \cdot k_l\right)$$

sind. Weiter nennt man

s	...	Anzahl Stufen
c _i	...	Knoten
b _i	...	Gewichte
a _{il}	...	Runge-Kutta Matrix / Koeffizienten

RK Verfahren schreibt man am besten in einem sog. Butcher-Tableau (BT)

$$\begin{array}{c|cccc}
 c_1 & a_{11} & a_{12} & \dots & a_{1s} \\
 c_2 & a_{21} & a_{22} & \dots & a_{2s} \\
 \vdots & & & & \\
 c_s & a_{s1} & a_{s2} & \dots & a_{ss} \\
 \hline
 & b_1 & b_2 & \dots & b_s
 \end{array}
 = \begin{array}{c|c}
 \vec{c} & A \\
 \hline
 & \vec{b}^T \\
 & \downarrow \\
 & \text{transponiert}
 \end{array}$$

Bsp.: (12) BT der verbesserten Polynomzug-
Methode von Euler

$$\begin{array}{c}
 \left. \begin{array}{l} 2 \text{ Stufen} \\ s=2 \end{array} \right\} \begin{array}{c|cc}
 \vec{c} \swarrow & 0 & 0 \\
 & 1/2 & 0 \\
 \hline
 & 0 & 1 \\
 \vec{b}^T \swarrow & &
 \end{array}
 \end{array}
 \begin{array}{l}
 \leftarrow A \\
 \leftarrow \text{die Nullen der RK-} \\
 \text{Matrix A schreibt} \\
 \text{man (oft) nicht}
 \end{array}$$

$$\begin{aligned}
 k_1 &= f\left(t_j + \overset{0}{c_1} \cdot h, y_j + h \cdot \left(\overset{0}{a_{11}} \cdot k_1 + \overset{0}{a_{12}} \cdot k_2 \right)\right) \\
 &= f(t_j, y_j)
 \end{aligned}$$

$$\begin{aligned}
 k_2 &= f\left(t_j + \overset{1/2}{c_2} \cdot h, y_j + h \cdot \left(\overset{1/2}{a_{21}} \cdot k_1 + \overset{0}{a_{22}} \cdot k_2 \right)\right) \\
 &= f\left(t_j + h/2, y_j + h/2 \cdot k_1\right)
 \end{aligned}$$

$$y_{j+1} = y_j + h \cdot (b_1 \cdot k_1 + b_2 \cdot k_2)$$

$$= y_j + h \cdot k_2$$

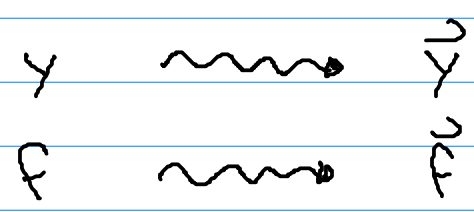
(13) BT der Methode von Heun

0	0	0
1	1	0
	1/2	1/2

(14) BT des Euler-Verfahrens

?	?
	?

Obwohl wir bisher nur skalare AWP betrachten haben, kann man die RK-Verfahren sehr einfach auf allgemeine AWP (also Systeme) verallgemeinern:



Desweiteren ist die Def. der RK-Verfahren sehr allgemein. Sie umfasst auch sog. implizite Verfahren

Bsp.: (15) Implizite Mittelpunkts-Methode

$$\frac{1/2}{\quad} \Bigg| \frac{1/2}{\quad}$$

$$\underline{k_1} = f\left(t_j + \frac{h}{2}, y_j + \frac{h}{2} \cdot \underline{k_1}\right)$$

$$y_{j+1} = y_j + h \cdot k_1$$

implizite Def. von k_1

d.h. man muss eventuell ein i.A. nicht-lineares Gl.-System lösen!

Implizite Methoden benötigt man bei sog. steifen Problemen Kap. V.

Explizite RK-Verfahren haben ein BT der Form

$$\begin{array}{c|cccc}
 c_1 & 0 & 0 & \dots & 0 \\
 c_2 & a_{21} & 0 & \dots & 0 \\
 \vdots & \vdots & \cdot & \cdot & \vdots \\
 c_s & a_{s1} & a_{s2} & \dots & a_{s,s-1} & 0 \\
 \hline
 & b_1 & b_2 & & b_{s-1} & b_s
 \end{array}$$

D.h. A ist eine Untere-Dreiecksmatrix mit Nullen auf der Diagonalen.

Das wohl bekannteste RK-Verfahren ist das sog. klassische RK-Verfahren

$$\begin{array}{c|cccc}
 0 & 0 & 0 & 0 & 0 \\
 1/2 & 1/2 & 0 & 0 & 0 \\
 1/2 & 0 & 1/2 & 0 & 0 \\
 1 & 0 & 0 & 1 & 0 \\
 \hline
 & 1/6 & 2/6 & 2/6 & 1/6
 \end{array}$$

Graphisch \rightsquigarrow Slides

Oft wird es auch nur als DAS RK-Verfahren oder RK4 bezeichnet.

\leftarrow 4 Stufen

II.4 Fehlerbetrachtungen für ESV

Hier wollen wir die Genauigkeit von ESV untersuchen. Dazu betrachten wir ein allgemeines ESV der Form

$$y_{j+1} = y_j + h \cdot \phi(t_j, y_j, h)$$

wobei ϕ die sog. Verfahrens- oder Inkrements-Funktion ist.

Bem.: (i) Bei RK-ESV ist

$$\phi(t_j, y_j, h) = \sum_{i=1}^s b_i \cdot k_i$$

(ii) Bei einem expliziten ESV kann man ϕ einfach durch einsetzen berechnen.

Bei einem impliziten ESV muss man um ϕ zu berechnen i.A. nicht-lineare Gleichungssysteme lösen.

Im folgenden betrachten wir das (skalare)

AWP

$$\dot{y}(t) = f(t, y(t)) \quad (\text{DGL})$$

$$y(t_0) = y_0 \quad (\text{AW})$$

auf dem Intervall $[t_0, T]$.

Anfangs-
Endzeit

Mit $y(t)$ bezeichnen wir die exakte Lösung.

Wir approx. die Lösung durch diskretisieren des Zeit-Intervalls in N Teil-Intervalle

$$t_j = t_0 + j \cdot h$$

mit der/dem Schrittweite / Zeitschritt

$$h = \frac{T - t_0}{N}$$

Die approx. Lösung zur Zeit t_j bezeichnen wir mit y_j .

Also $y(t_j) \approx y_j$

Def.: Der globale Diskretisierungsfehler (GDF) zur Zeit t_j ist definiert durch

$$E_j = \underbrace{y(t_j)}_{\text{exakte}} - \underbrace{y_j}_{\text{approx. Lösung zur Zeit } t_j}$$

D.h. E_j ist der Fehler (die Differenz zwischen der exakten und approx. Lösung) nach j Schritten.

Def.: Ein ESV heisst konvergent von der Ordnung p (oder hat Konvergenz-Ordnung (KO) p), wenn gilt

$$E = \max_{j=0, \dots, N} \underbrace{|y(t_j) - y_j|}_{E_j} = O(h^p)$$

für h klein genug.

Bsp.: (16) Empirische KO der bisher
angetrottenen RK-ESV

→ Slides

$$\text{Euler} \sim \mathcal{O}(h^?)$$

$$\text{verb. Euler} \sim \mathcal{O}(h^?)$$

$$\text{Heun} \sim \mathcal{O}(h^?)$$

$$\text{RK4} \sim \mathcal{O}(h^?)$$

Um die KO eines ESV theoretisch zu
untersuchen benötigen wir:

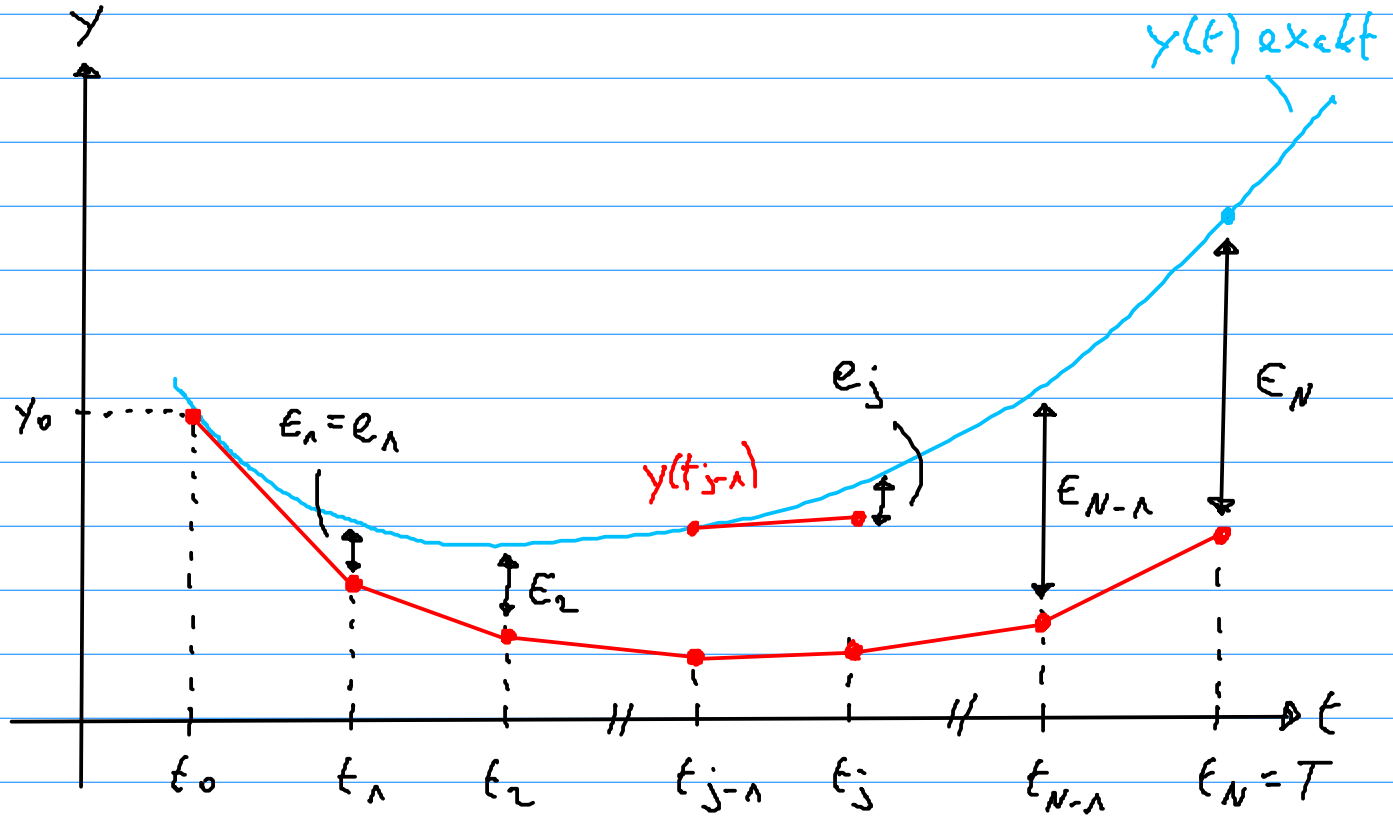
Def.: Der lokale Diskretisierungsfehler (LDF)
zur Zeit t_j ist definiert durch

$$e_j = y(t_j) - \left(y(t_{j-1}) + h \cdot \phi(t_{j-1}, y(t_{j-1}), h) \right)$$

↑ exakte Lösung bei t_{j-1} !

D.h. e_j ist der Fehler bei t_j nach einem
Schritt ausgehend von der exakten Lösung
bei t_{j-1} .

Graphisch:



Def.: Der Konsistenzfehler (KF) zur Zeit t_j ist definiert durch

$$\begin{aligned} \tau_j &= \frac{e_j}{h} \\ &= \frac{y(t_j) - y(t_{j-1})}{h} - \phi(t_{j-1}, y(t_{j-1}), h) \end{aligned}$$

Def.: Ein ESV heisst konsistent von der Ordnung p (oder hat Konsistenzordnung p), wenn gilt

$$\tau = \max_{j=0, \dots, N} |\tau_j| = \mathcal{O}(h^p)$$

für h klein genug.

Ein ESV heisst konsistent falls $p \geq 1$.

LDF, KF und Konsistenzordnung sind lokale Grössen. Diese kann man (relativ) einfach mittels Taylor-Entwicklungen bestimmen:

$$\begin{aligned} \tau_{j+n} &= \frac{y(t_j + \overbrace{h}^{t_{j+n}}) - y(t_j)}{h} - \phi(t_j, y(t_j), h) \\ &= \frac{\cancel{y(t_j)} + h \cdot \dot{y}(t_j) + \frac{h^2}{2} \ddot{y}(t_j) + \dots - \cancel{y(t_j)}}{h} \\ &\quad - \left(\phi(t_j, y(t_j), 0) + h \cdot \phi'(t_j, y(t_j), 0) + \frac{h^2}{2} \phi''(t_j, y(t_j), 0) + \dots \right) \\ &= \dot{y}(t_j) - \phi(t_j, y(t_j), 0) \\ &\quad + \frac{h}{2} \left(\ddot{y}(t_j) - 2 \cdot \phi'(t_j, y(t_j), 0) \right) \\ &\quad + \frac{h^2}{6} \left(\dddot{y}(t_j) - 3 \cdot \phi''(t_j, y(t_j), 0) \right) \\ &\quad + \dots + \mathcal{O}(h^p) + \dots \end{aligned}$$

Berechnen wir zunächst die Ableitungen der Lösung

$$\overset{\text{DGL}}{\dot{y}(t)} = f(t, y(t))$$

$$\ddot{y}(t) = \frac{d}{dt} f(t, y(t))$$

Ketten-
Regel! = $\frac{\partial f}{\partial t}(t, y(t)) + \frac{\partial f}{\partial y}(t, y(t)) \cdot \dot{y}(t)$ " f(t, y(t)) DGL!

$$= \frac{\partial f}{\partial t}(t, y(t)) + \frac{\partial f}{\partial y}(t, y(t)) \cdot f(t, y(t))$$

$$\ddot{y}(t) = \frac{d}{dt} \left(\frac{\partial f}{\partial t}(t, y(t)) + \frac{\partial f}{\partial y}(t, y(t)) \cdot f(t, y(t)) \right)$$

$$= \frac{\partial^2 f}{\partial t^2}(t, y(t)) + \frac{\partial^2 f}{\partial y \partial t}(t, y(t)) \cdot \dot{y}(t) = f(t, y(t))$$

$$\overset{\frac{\partial^2 f}{\partial t \partial y} = \frac{\partial^2 f}{\partial y \partial t}}{+} \left(\frac{\partial^2 f}{\partial t \partial y}(t, y(t)) + \frac{\partial^2 f}{\partial y^2}(t, y(t)) \cdot \dot{y}(t) \right) \cdot f(t, y(t))$$

$$+ \frac{\partial f}{\partial y}(t, y(t)) \cdot \left(\frac{\partial f}{\partial t}(t, y(t)) + \frac{\partial f}{\partial y}(t, y(t)) \cdot \dot{y}(t) \right)$$

$$= \frac{\partial^2 f}{\partial t^2}(t, y(t)) + 2 \cdot \frac{\partial^2 f}{\partial t \partial y}(t, y(t)) \cdot f(t, y(t))$$

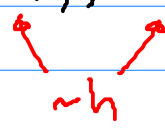
$$+ \frac{\partial^2 f}{\partial y^2}(t, y(t)) \cdot f(t, y(t))^2 + \frac{\partial f}{\partial t}(t, y(t)) \cdot \frac{\partial f}{\partial y}(t, y(t))$$

$$+ \left(\frac{\partial f}{\partial y}(t, y(t)) \right)^2 \cdot f(t, y(t))$$

$$\ddot{y}^{(k)}(t) = \left(\frac{\partial}{\partial t} + f(t, y(t)) \frac{\partial}{\partial y} \right)^{k-1} f(t, y(t))$$

Zur Berechnung der Entwicklung der Verfahrens-
Funktion benötigt man die zwei-dimensionale
Taylor-Reihe:

$$\begin{aligned}
 f(t + \Delta t, y + \Delta y) &= f(t, y) \\
 &+ \frac{\partial f}{\partial t}(t, y) \cdot \Delta t + \frac{\partial f}{\partial y}(t, y) \cdot \Delta y \\
 &+ \frac{1}{2} \frac{\partial^2 f}{\partial t^2}(t, y) \cdot \Delta t^2 + \frac{\partial^2 f}{\partial t \partial y}(t, y) \cdot \Delta t \cdot \Delta y + \frac{1}{2} \frac{\partial^2 f}{\partial y^2}(t, y) \cdot \Delta y^2 \\
 &+ \dots
 \end{aligned}$$



 $\sim h$

Am besten ein paar Beispiele

Bsp.: (17) Euler-Verfahren

$$\phi(t_j, y(t_j), h) = f(t_j, y(t_j))$$

$$\leadsto \phi(t_j, y(t_j), 0) = f(t_j, y(t_j)) = \dot{y}(t_j) \checkmark$$

$$\phi'(t_j, y(t_j), 0) = ?$$

Und damit Konsistenzordnung

$$p = ?$$

d.h. $\tau = \mathcal{O}(h^?)$ wie gemessen
in Bsp. (16)!

(18) Verbesserter Euler

$$k_1 = f(t_j, y(t_j))$$

$$k_2 = f(t_j + h/2, y(t_j) + h/2 k_1)$$

$$y_{j+1} = y(t_j) + h \cdot k_2$$

Entwickeln der Verfahrens-Funktion

$$\phi(t_j, y(t_j), h) = f\left(t_j + \frac{h}{2}, y(t_j) + \frac{h}{2} k_1\right)$$

$$= f(t_j, y(t_j)) \leftarrow \phi(t_j, y(t_j), 0)$$

$$\left. \phi'(t_j, y(t_j), 0) \cdot h \right\} + \frac{\partial f}{\partial t}(t_j, y(t_j)) \cdot \frac{h}{2} + \frac{\partial f}{\partial y}(t_j, y(t_j)) \cdot \frac{h}{2} k_1$$

$$\left. \phi''(t_j, y(t_j), 0) \cdot \frac{h^2}{2} \right\} \left\{ \begin{array}{l} + \frac{1}{2} \frac{\partial^2 f}{\partial t^2}(t_j, y(t_j)) \cdot \left(\frac{h}{2}\right)^2 + \frac{\partial^2 f}{\partial t \partial y}(t_j, y(t_j)) \cdot \left(\frac{h}{2}\right)^2 \cdot k_1 \\ + \frac{1}{2} \frac{\partial^2 f}{\partial y^2}(t_j, y(t_j)) \cdot \left(\frac{h}{2} k_1\right)^2 \end{array} \right.$$

+ ...

Durch Vergleich

$$\phi(t_j, y(t_j), 0) = f(t_j, y(t_j)) = \dot{y}(t_j) \quad \checkmark$$

$$2. \quad \phi'(t_j, y(t_j), 0) = \left(\frac{\partial f}{\partial t}(t_j, y(t_j)) + \frac{\partial f}{\partial y}(t_j, y(t_j)) \cdot f(t_j, y(t_j)) \right)$$

$$= \ddot{y}(t_j) \quad \checkmark$$

$$3. \quad \phi''(t_j, y(t_j), 0) = \dots \quad \times$$

Und damit Konsistenzordnung $p=2$

D.h. $\tau = O(h^2)$ wie gemessen
in Bsp. (16)!

(19) Allgemeines 3-stufiges explizites RK

c_1			
c_2	a_{21}		
c_3	a_{31}	a_{32}	
	b_1	b_2	b_3

Durch entwickeln der Verfahrensfunktion
und Vergleich mit der exakten
Entwicklung erhält man folgende
Bedingungen:

$$b_1 + b_2 + b_3 = 1$$

1. Ordnung

$$b_1 c_1 + b_2 c_2 + b_3 c_3 = 1/2$$

2. Ordnung

$$a_{21} b_2 + (a_{31} + a_{32}) b_3 = 1/2$$

$$b_1 c_1^2 + b_2 c_2^2 + b_3 c_3^2 = 1/3$$

$$a_{21} b_2 c_2 + (a_{31} + a_{32}) b_3 c_3 = 1/3$$

$$a_{21}^2 b_2 + (a_{31} + a_{32})^2 b_3 = 1/3$$

3. Ordnung

$$a_{21} b_2 c_1 + a_{31} b_3 c_1 + a_{32} b_3 c_2 = 1/6$$

$$a_{21} a_{32} b_3 = 1/6$$

8 nicht lineare Gleichungen für 3 Variablen!

Fazit: die lokalen Fehlermasse (LDF, KF, Konsistenzordnung) lassen sich mittels Taylor-Reihenentwicklungen abschätzen

↳ Dabei wird natürlich immer stillschweigend vorausgesetzt, dass die rechte Seite der DGL genügend glatt ist! $\leftarrow F$

Aber eigentlich sind wir ja an globalen Fehlergrößen (CDF & MO) interessiert.

Bsp. (16), (17) und (18) deuten bereits an, dass zwischen den lokalen und globalen Größen ein Zusammenhang besteht. Der folgende Satz untermauert dies:

Satz II.3: Falls die rechte Seite der DGL $\vec{f}(t, \vec{y})$ und die Verfahrens-funktion $\vec{\phi}(t, \vec{y}, t)$ Lipschitz-stetig in \vec{y} sind, dann gilt für das ESV folgende (globale) Fehlerabschätzung

$$\epsilon = \max_{j=0, \dots, N} \|\vec{y}(t_j) - \vec{y}_j\|$$

$$\leq \left(\underbrace{\|\vec{y}(t_0) - \vec{y}_0\|}_{\text{AW Fehler}} + \sum_{j=1}^N \underbrace{\|\vec{e}_j\|}_{\text{fehler in jedem Schritt summieren sich schlimmstenfalls}} \right) \cdot e^{\tilde{L}(t_N - t_0)}$$

wobei \tilde{L} die Lipschitz-Konstante der Verfahrens-funktion $\vec{\phi}$ ist.

Obige Aussage impliziert

Konsistenz der Ordnung p

\Rightarrow Konvergenz der Ordnung p

Bem.: (i) Satz II.3 macht eine Aussage über die Stabilität von ESV, d.h. eine kontrollierte Fehlerfortpflanzung (AW + LDF \blacktriangledown)

(ii) \tilde{L} hängt über $\vec{\phi}$ mit der rechten Seite \vec{f} zusammen

Betrachten wir den GDF zur Endzeit $T = t_N$:

$$\begin{aligned}
 |e_N| &\stackrel{\text{Def.}}{=} |y(t_N) - y_N| \\
 &= \left| \underbrace{y(t_{N-1})}_{\text{Trick}} + h \cdot \frac{y(t_N) - y(t_{N-1})}{h} + 0 \text{ Trick} \right. \\
 &\quad \left. - h \left(\underbrace{\phi(t_{N-1}, y(t_{N-1}), h)}_{\text{Trick}} - \underbrace{\phi(t_{N-1}, y(t_{N-1}), h)} \right) \right. \\
 &\quad \left. - \underbrace{\left(y_{N-1} + h \cdot \phi(t_{N-1}, y_{N-1}, h) \right)}_{y_N} \right|
 \end{aligned}$$

$$\begin{aligned}
 &\stackrel{\Delta\text{-UG}}{\leq} \underbrace{|y(t_{N-1}) - y_{N-1}|}_{= e_{N-1}} \\
 &+ h \cdot \left| \frac{y(t_N) - y(t_{N-1})}{h} - \phi(t_{N-1}, y(t_{N-1}), h) \right| \stackrel{= h \cdot \tau_N}{=} e_N \\
 &+ h \cdot \left| \phi(t_{N-1}, y(t_{N-1}), h) - \phi(t_{N-1}, y_{N-1}, h) \right| \\
 &\quad \stackrel{= h \tilde{L} |y(t_{N-1}) - y_{N-1}| = h \tilde{L} |e_{N-1}|}{=}
 \end{aligned}$$

$$= (\lambda + h \tilde{L}) |e_{N-1}| + |e_N|$$

$$\stackrel{\text{Rekursiv einsetzen}}{=} (\lambda + h \tilde{L}) \left((\lambda + h \tilde{L}) |e_{N-2}| + |e_{N-1}| \right) + |e_N|$$

$$\begin{aligned}
 &\vdots \\
 &= (\lambda + \tilde{L} h)^N |e_0| + \sum_{j=1}^N \underbrace{(\lambda + h \tilde{L})^{N-j}}_{\leq (\lambda + h \tilde{L})^N} |e_j|
 \end{aligned}$$

$$\begin{aligned}
&\leq \left(|\epsilon_0| + \sum_{j=1}^N |e_j| \right) \underbrace{(\lambda + h\tilde{L})^N}_{\leq e^{h\tilde{L}}} \\
&\leq \left(|\epsilon_0| + \sum_{j=1}^N |e_j| \right) e^{h\tilde{L}N} \\
&\quad h = \frac{T-t_0}{N} \\
&= \left(|\epsilon_0| + \sum_{j=1}^N |e_j| \right) \cdot e^{\tilde{L}(T-t_0)} \\
&= \left(\underbrace{|y(t_0) - y_0|}_{\substack{\uparrow \\ \text{AW Fehler}}} + \sum_{j=1}^N \underbrace{|e_j|}_{\substack{\uparrow \\ \text{Fehler in jedem} \\ \text{Schritt}}} \right) \cdot e^{\tilde{L}(T-t_0)}
\end{aligned}$$

Und damit hätten wir Satz II.3 für den skalaren Fall begründet! \blacksquare