

### III. Adaptive Schrittweitensteuerung

Ziel: Eine gewünschte Genauigkeit erreichen mit möglichst wenig (Rechen-) Aufwand (Effizienz!)  $\rightarrow$  adaptive Schrittweite  $h \rightarrow h_j$

- Fehlerabschätzungen & Toleranzen (abs., rel.)
- Schrittweithalbierung & Eingebettete RK-Verfahren
- Adaptive Schrittweitensteuerung

Wozu: Praxis relevant!

#### III.1 Fehlerbetrachtungen und Toleranzen

Gegeben ein skalares AWP  $\dot{y}(t) = f(t, y(t))$  *der Einfachheit halber!*

$$y(t_0) = y_0$$

für  $t \in [t_0, T]$ .

Wir wollen dieses AWP näherungsweise bis auf eine Genauigkeit/Toleranz mit möglichst wenigen IntegrationsSchritten lösen

$$E = \max_{j=0, \dots, N} | \underbrace{y(t_j)}_{\text{exakt}} - \underbrace{y_j}_{\text{approx. Lösung}} | < \text{TOL}$$

Dieses Problem ist nicht mehr ganz so "einfach" zu behandeln wie bei der Quadratur weil gemachte Fehler sich verstärken können...

Satz II.3 gibt uns die Abschätzung

$$\epsilon \leq \left( |y(t_0) - y_0| + \sum_{j=1}^N |e_j| \right) \cdot e^{\tilde{\Sigma}(T-t_0)} < \text{TOL}$$

↑ Fehler in AW
↑ LDF
↑ "vernachlässigen"

Also

$$\sum_{j=1}^N |e_j| < \text{TOL}$$

$$|e_1| + \dots + |e_j| + \dots + |e_N| < \text{TOL}$$

Idee: Wähle  $|e_j| < \epsilon_{\text{ol}}$  ← lokal

↑ global

$$\text{Damit } \sum_{j=1}^N |e_j| = N \cdot \epsilon_{\text{ol}} < \text{TOL}$$

↑  
?

Problem: Wir kennen die Anzahl Schritte  $N$  nicht im Voraus!

Bei äquidistenter Schrittweite gilt

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

Damit liegt nahe

$$\frac{T-t_0}{h_j} \cdot \text{tol}_j \approx \text{TOL}$$

Schrittweite  
im j-ten Schritt

$$\text{tol}_j \approx h_j \cdot \frac{\text{TOL}}{T-t_0}$$

Also wähle  $h_j$  so, dass

$$|e_j| \leq \text{tol}_j \quad (\text{TKA})$$

Soweit so gut, aber es stellt sich dennoch die Frage ob die Vernachlässigung des exponentiellen Faktors eine gute Idee war...

**I.A. NEIN!**

D.h. es ist i.A. schwierig (bzw. unmöglich) von den LDF<sub>n</sub> auf die CDF<sub>n</sub> zu schliessen

4

Deshalb könnte man auch einfach nur die lokalen Fehler mit lokalen Toleranzen kontrollieren, z.B.

$$|e_j| \lesssim atol \quad (\text{absolute}) \quad (TK2)$$

$$|e_j| \lesssim |y_{j-1}| \cdot rtol \quad (\text{relative}) \quad (TK3)$$

$$|e_j| \lesssim atol + |y_{j-1}| \cdot rtol \quad (\text{abs. + rel.}) \quad (TK4)$$

Bem.: Es gibt keine Garantie, dass die Näherungslösung die gewünschte Genauigkeit hat!

In der Praxis liefert die adaptive Schrittweitensteuerung häufig dennoch gute Resultate.

## III.2 Lokale Fehlerschätzer

Nun müssen wir die LDF  $e_j$  lokal schätzen. Für diese haben wir die a priori Fehlerschätzer (s. II.4)

$$\begin{aligned} |e_j| &= |y(t_j) - y(t_{j-1}) - h_j \cdot \phi(t_{j-1}, y(t_{j-1}), h_j)| \\ &= \mathcal{O}(h^{p+1}) = C \cdot h^{p+1} \end{aligned}$$

Für ein ESV der KO  $p$ .

In der Konstanten  $C$  stecken höhere Ableitungen der (uns unbekannt!) Lösung. Deshalb, wie schon bei der adaptiven Quadratur, sind diese a priori Fehlerschätzer nicht direkt brauchbar.

Idee: Vergleiche das Resultat eines Verfahrens mit dem Resultat eines genaueren Verfahrens (sog. Kontroll-Verfahren (KV))

Wie bei der adaptiven Quadratur überlegt man sich folgende Möglichkeiten.

### III.2.1 Schrittweitenhalbierung

Gegeben ein Verfahren der KO  $p$ , berechne im  $j$ -ten Schritt ein Schritt mit Schrittweite  $h$

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

und 2 Schritte mit halber Schrittweite  $\frac{h}{2}$  (KV)

Hut / Für KV!

$$\hat{y}_{j+1/2} = y_j + \frac{h}{2} \cdot \phi(t_j, y_j, h/2)$$

$$\hat{y}_{j+1} = \hat{y}_{j+1/2} + \frac{h}{2} \phi(t_{j+1/2}, \hat{y}_{j+1/2}, h/2)$$

$t_j + h/2$

Der LDF ist dann

$$|e_{j+1}| = |y(t_{j+1}) - y_{j+1}| = C \cdot h^{p+1} + \mathcal{O}(h^{p+2})$$

und

Lösung mit AK  $y_j$        $\sim$  Satz II.3

$$|\hat{e}_{j+1}| = |y(t_{j+1}) - \hat{y}_{j+1}| = 2 \cdot C \cdot \left(\frac{h}{2}\right)^{p+1} + \mathcal{O}(h^{p+2})$$

$$= \frac{1}{2^p} C \cdot h^{p+1} + \mathcal{O}(h^{p+2})$$

$$= \frac{|e_{j+1}|}{2^p}$$

Nun wollen wir  $|e_{j+1}|$  und  $|\hat{e}_{j+1}|$  in Verbindung mit der Differenz der Resultate  $|\hat{y}_{j+1} - y_{j+1}|$  bringen:

$$|e_{j+1}| = \left| y(t_{j+1}) - \hat{y}_{j+1} + \hat{y}_{j+1} - y_{j+1} \right|$$

+ 0 Trick

$$\stackrel{\Delta\text{-U.G.}}{\leq} \underbrace{\left| y(t_{j+1}) - \hat{y}_{j+1} \right|}_{\frac{|e_{j+1}|}{2^p}} + \left| \hat{y}_{j+1} - y_{j+1} \right|$$

) Auflösen nach  $|e_{j+1}|$

$$\rightsquigarrow |e_{j+1}| \approx \frac{2^p}{2^p - 1} |\hat{y}_{j+1} - y_{j+1}| = \epsilon_{j+1}$$

und Schätzungen!

$$|\hat{e}_{j+1}| \approx \frac{1}{2^p - 1} |\hat{y}_{j+1} - y_{j+1}| = \hat{\epsilon}_{j+1}$$

Dies sind dann wieder sog. a posteriori Fehlerschätzer.

- Bsp.: (1) Euler mit  $h$  + Euler mit  $h/2$   
 (2) Heun  $\rightsquigarrow$  + Heun  $\rightsquigarrow$

### III.2.2 Eingebettete RK-Verfahren

Der Fehler wird geschätzt durch Vergleich eines Verfahrens der  $KO$   $p$  mit einem Verfahren höherer  $KO$ , z.B.  $p+1$ , (das  $KV$ ).

Um dies effizient zu berechnen, (d.h. mit möglichst wenigen Auswertungen der rechten Seite  $f$ ) verwendet man sog. eingebettete RK-Verfahren mit Butcher-Tableaux der Form

$c_1$					
$c_2$	$a_{21}$				
$c_3$	$a_{31}$	$a_{32}$			
$\vdots$	$\vdots$	$\vdots$	$\ddots$		
$c_s$	$a_{s1}$	$a_{s2}$	$\dots$	$a_{s,s-1}$	
	$b_1$	$b_2$	$\dots$	$b_{s-1}$	$b_s$
	$\hat{b}_1$	$\hat{b}_2$	$\dots$	$\hat{b}_{s-1}$	$\hat{b}_s$

←  $KV$

Bsp.: (3) Euler + Heun Verfahren

$0$			
$1$	$1$		
	$1$	$0$	← Euler
	$1/2$	$1/2$	← Heun

(4) Dormand und Prince Verfahren (MATLAB ode45)



Der a posteriori Fehlerschätzer ist dann

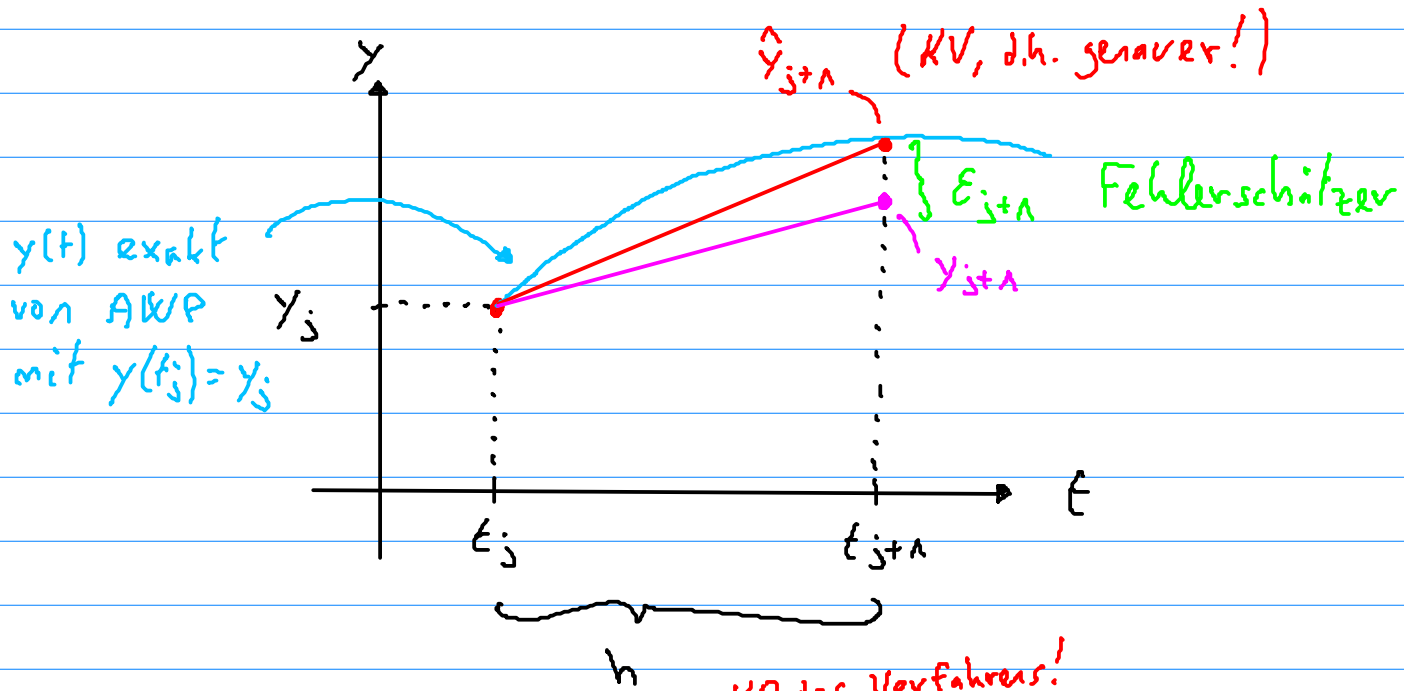
$$|e_{j+1}| \approx |\hat{y}_{j+1} - y_{j+1}| = \varepsilon_{j+1}$$

Bem.: Hier haben wir nur ein Fehlerschätzer  
für das weniger genaue Verfahren  
(D.h. kein Schätzer für  $\hat{\varepsilon}_{j+1}$  des KVs!)

### III.3 Adaptive Schrittweitensteuerung

Ziel: Wähle Schrittweite klein genug um gewünschte lokale Präzision zu erreichen und gross genug um effizient zu sein

Ausgangslage im  $j$ -ten Schritt



$$\text{Haben: } E_{j+1} \approx C \cdot h^{p+1} \quad \Rightarrow \quad C \approx \frac{E_{j+1}}{h^{p+1}}$$

$$\text{Wollen: } E_{\text{tol}_{j+1}} \approx C \cdot H^{p+1} \quad \Rightarrow \quad H \approx h \cdot \left( \frac{E_{\text{tol}_{j+1}}}{E_{j+1}} \right)^{1/(p+1)}$$

$H$  ist ein schrittweiser-Vorschlag um

$$\varepsilon_{j+1} \approx \varepsilon_{ol_{j+1}}$$

zu erreichen.

Frage: ① Wenn  $\varepsilon_{j+1} > \varepsilon_{ol_{j+1}}$ , ist

$$H \lesssim h \quad ?$$

② Wenn  $\varepsilon_{j+1} < \varepsilon_{ol_{j+1}}$  ?

Daraus überlest man sich folgende adaptive  
schrittweiser-Wahl

$$H = h \cdot \min \left( \underset{\substack{\text{begrenzte Vergrößerung}}}{\text{facmax}}, \max \left( \underset{\substack{\text{begrenzte Verkleinerung}}}{\text{facmin}}, \underset{\substack{\text{"Sicherheits"-Faktor}}}{\text{fac}} \cdot \left( \frac{\varepsilon_{ol}}{\varepsilon} \right)^{\frac{1}{p+1}} \right) \right)$$

# Adaptive Schrittweitensteuerung (~MATLAB Pseudo-Code)

Function  $[t, y] = \text{adapt\_ode}(\dots)$

$$j = 0$$

$$h = h_0$$

$$t_{\text{ol}} = a_{\text{tol}} + |y_0| \cdot r_{\text{tol}}$$

$$\varepsilon = \varepsilon_{\text{ol}}$$

while  $(t_j < \tau)$

$$h = h \cdot \min \left( \text{fac}_{\text{max}}, \max \left( \text{fac}_{\text{min}}, \text{fac} \cdot \left( \frac{t_{\text{ol}}}{\varepsilon} \right)^{\frac{1}{p+1}} \right) \right)$$

Parameter

(\*Details zu h mo Übung)

$$y_{j+1} = y_j + \dots$$

$$\hat{y}_{j+1} = y_j + \dots$$

$$\varepsilon = |y_{j+1} - \hat{y}_{j+1}| \quad \text{oder} \quad \hat{\varepsilon} = \dots$$

mit h-Halb.-  
Methode

$$t_{\text{ol}} = a_{\text{tol}} + \max(|y_j|, |\hat{y}_{j+1}|) \cdot r_{\text{tol}} \quad (\text{TKS})$$

if  $(\varepsilon < t_{\text{ol}})$

$$t_{j+1} = t_j + h$$

$$y_{j+1} = \hat{y}_{j+1}$$

$$j = j + 1$$

end

end

← Nehmen das KV!

(TKA...4  
auch  
möglich)

Bem.: (i) Es wird stets das genauere Resultat  $\hat{y}_{j+1}$  genommen. Obwohl man im Fall eines eingebetteten RK-Verfahren eigentlich keinen Fehlerschätzer  $\hat{E}_{j+1}$  hat.

(ii) Es gibt keine Garantie, dass der globale Fehler  $E < \epsilon_{\text{tol}}$  ist !