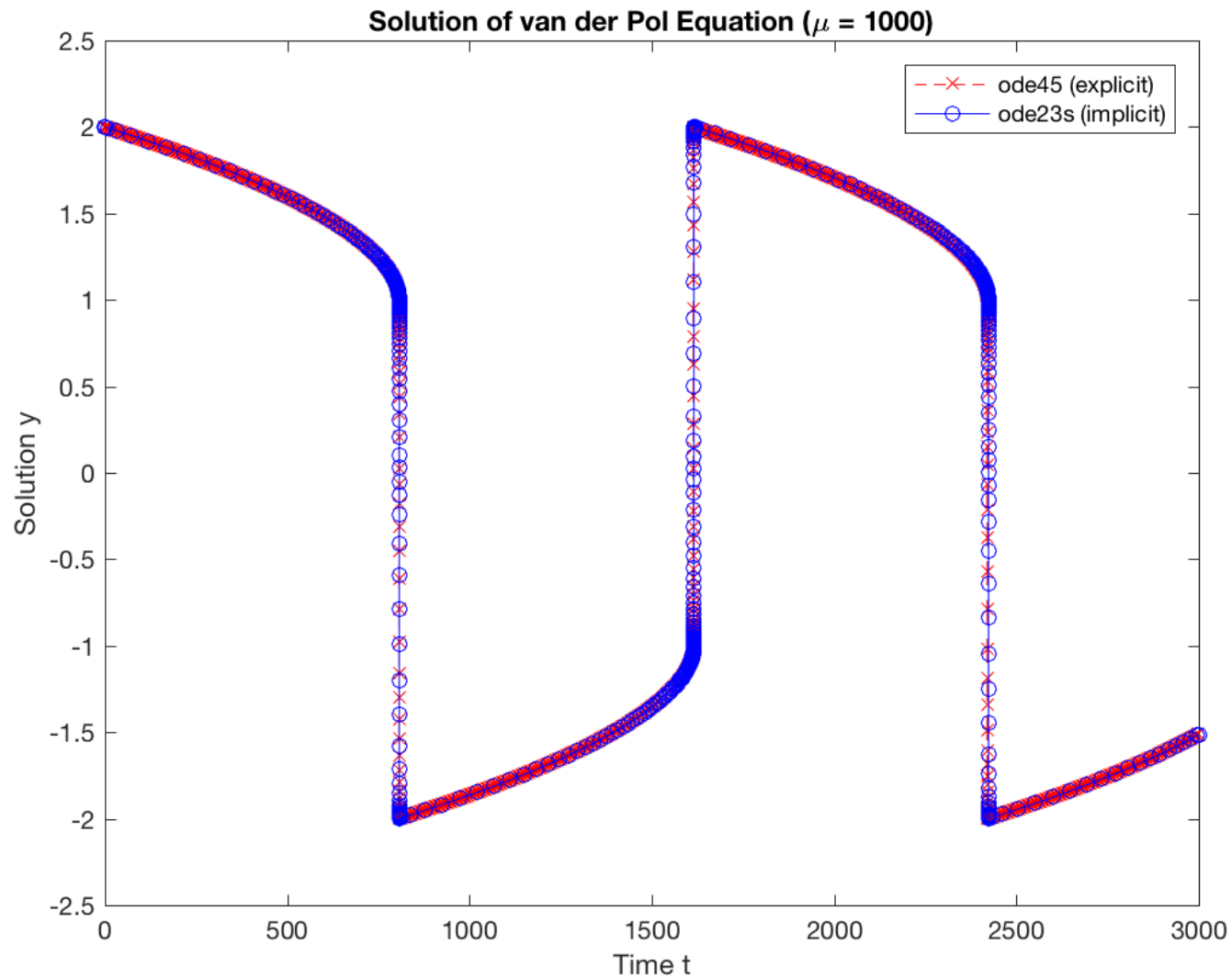


Motivation: Steifes Problem

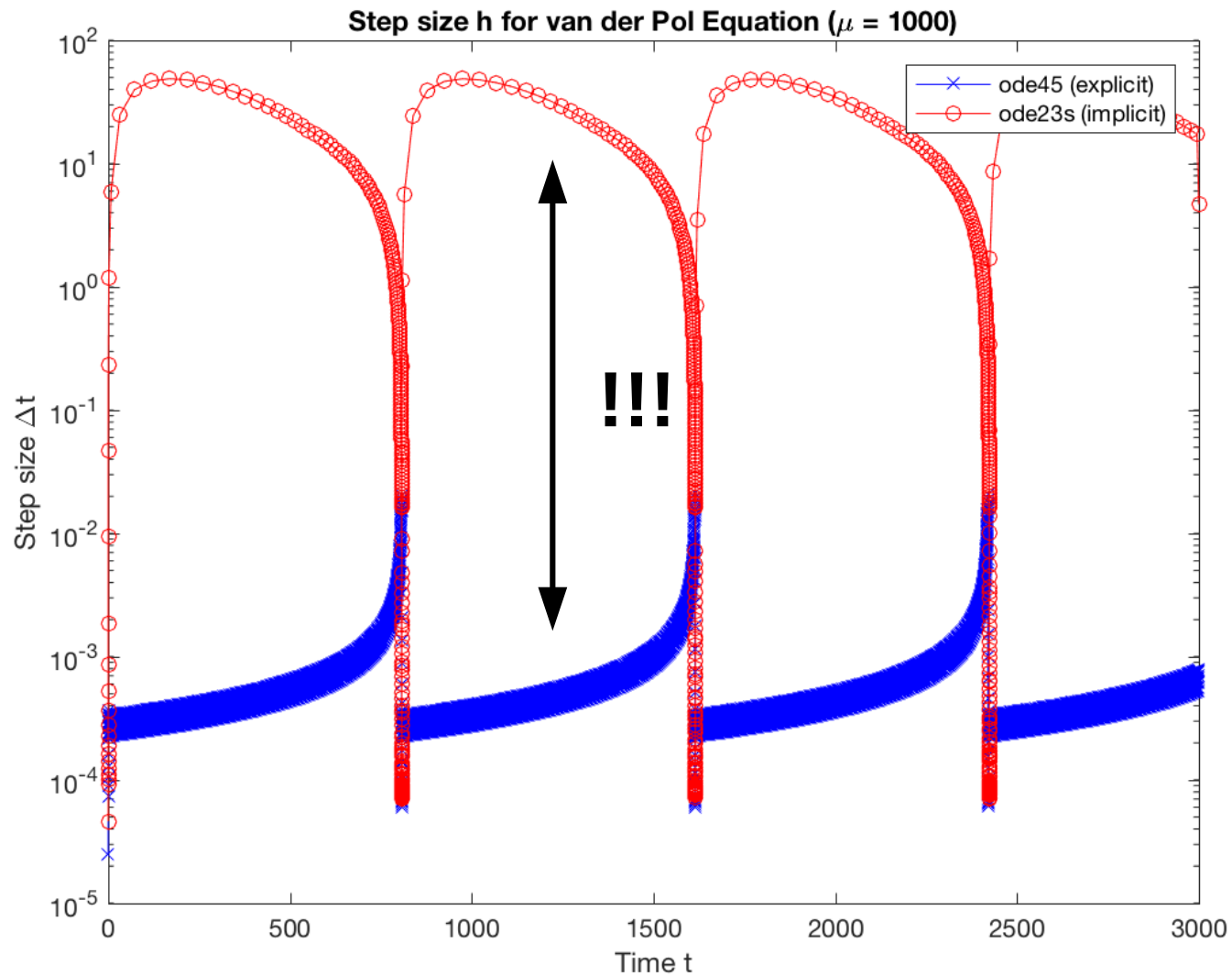
$$\begin{aligned} \dot{y}_1 &= y_2 \\ \dot{y}_2 &= \mu(1 - y_1^2)y_2 - y_1 \end{aligned} \quad \mu = 1000 \quad \mathbf{y}_0 = \begin{pmatrix} 0 \\ 2 \end{pmatrix} \quad t \in [0, 3000]$$

```
>> StiffVanDerPol
Loeser: ode45
Elapsed time is 73.490408 seconds.
Loser: ode23s
Elapsed time is 0.167961 seconds.
```

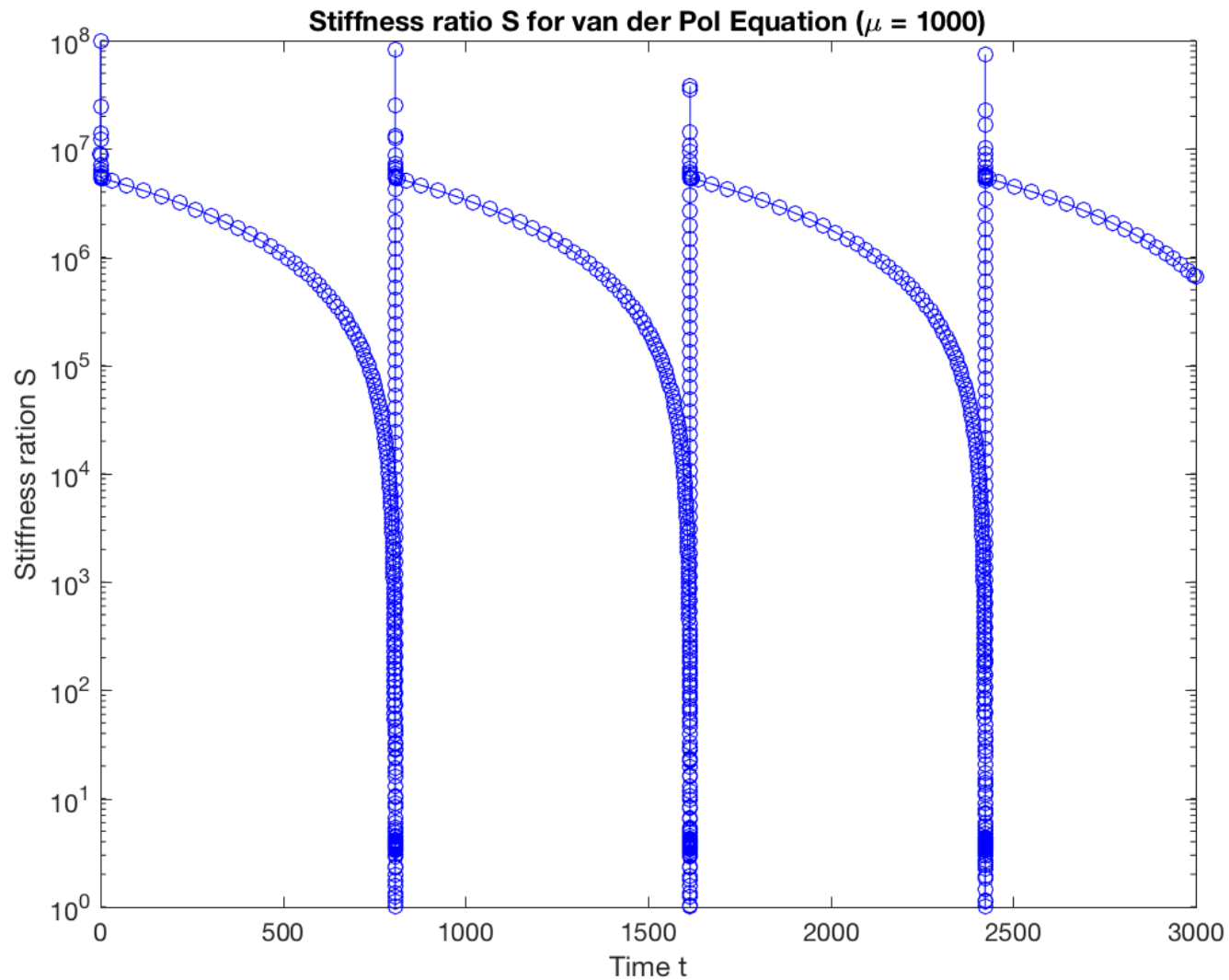
Motivation: Steifes Problem



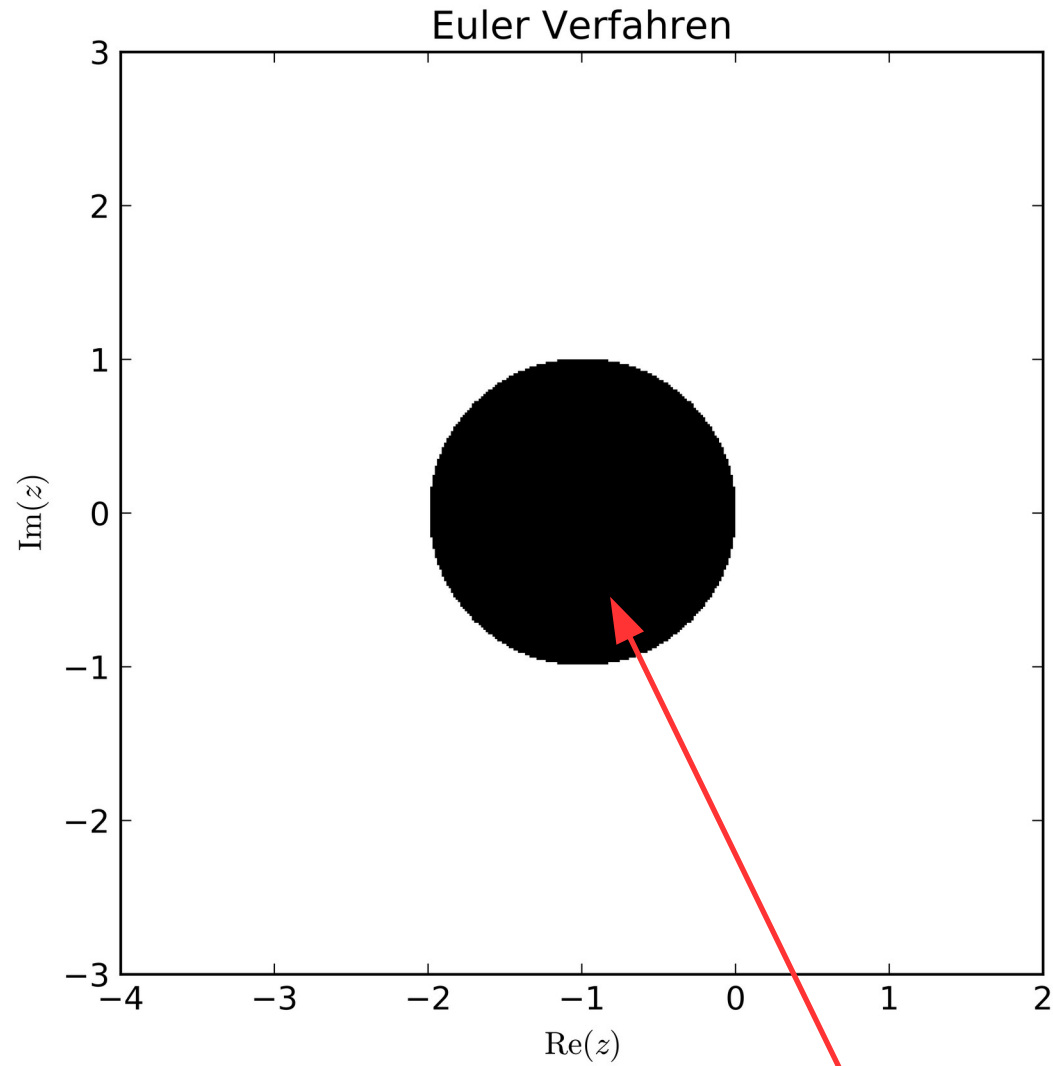
Motivation: Steifes Problem



Motivation: Steifes Problem

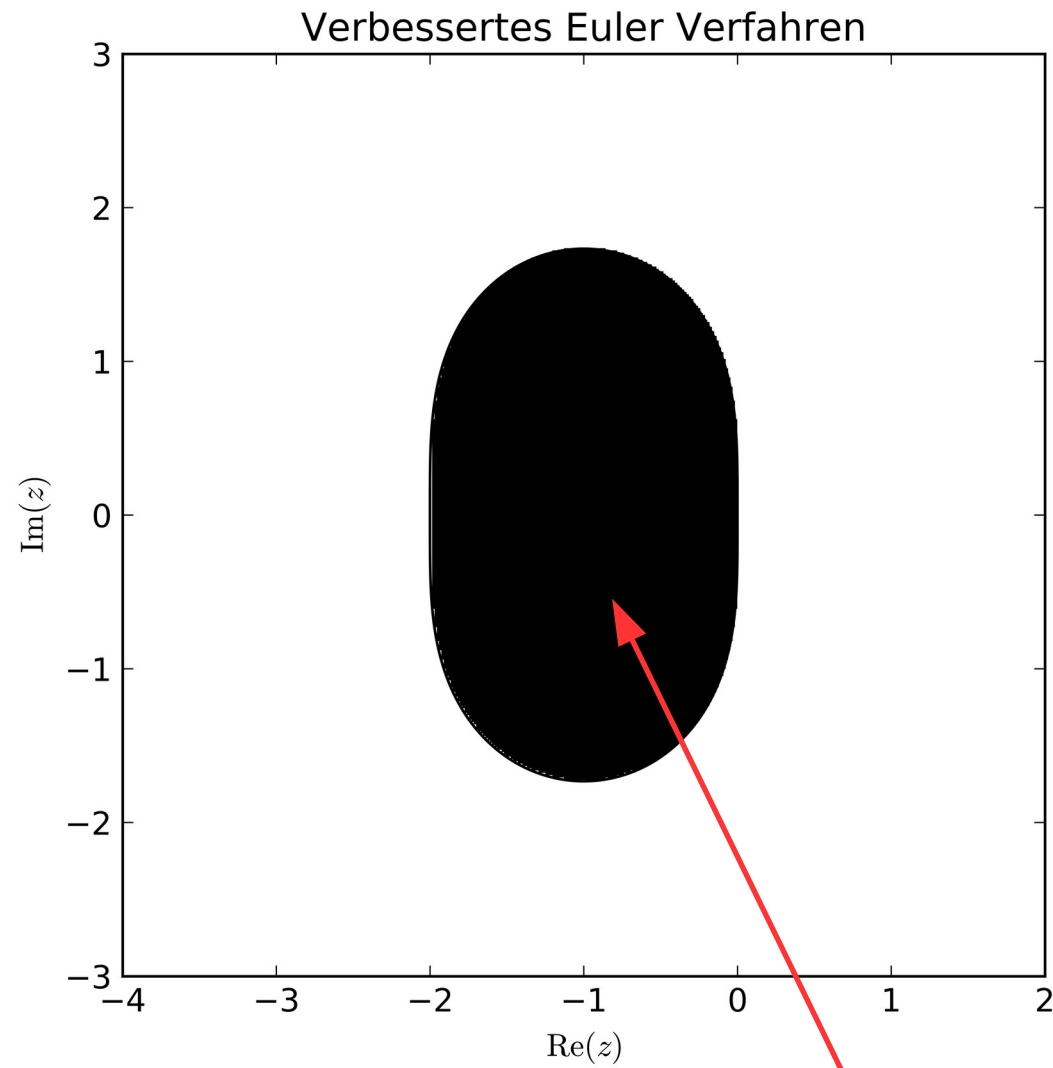


Stabilitätsgebiete



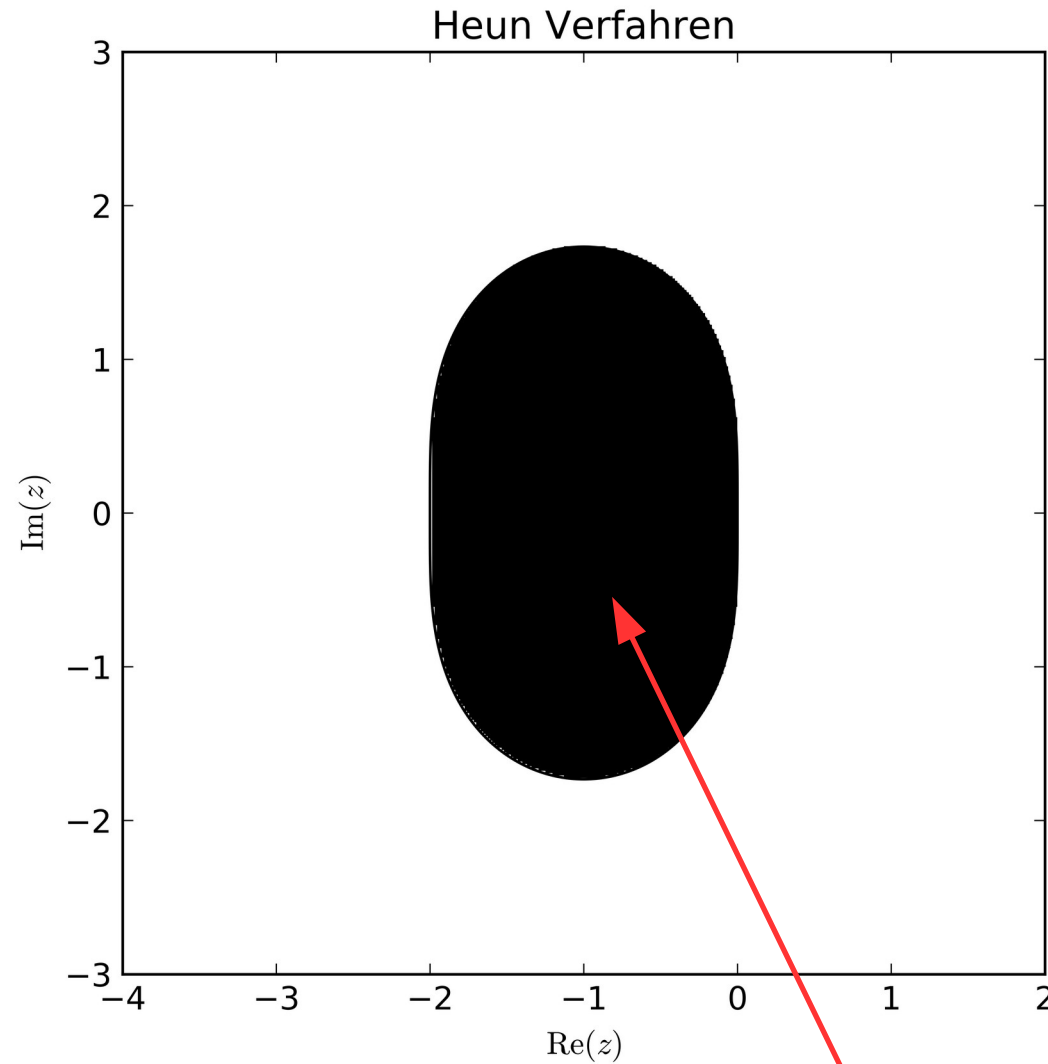
Stabilitätsgebiet

Stabilitätsgebiete



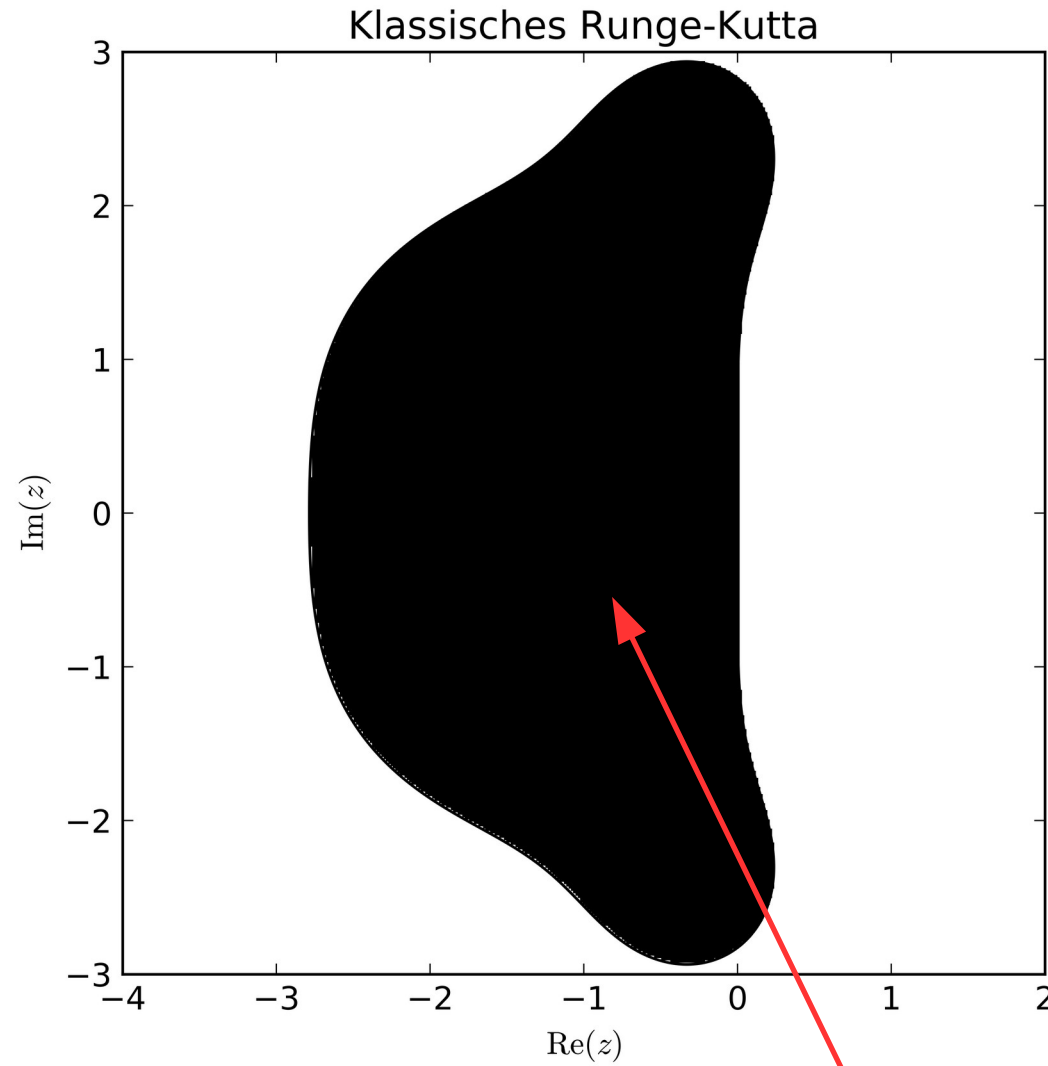
Stabilitätsgebiet

Stabilitätsgebiete



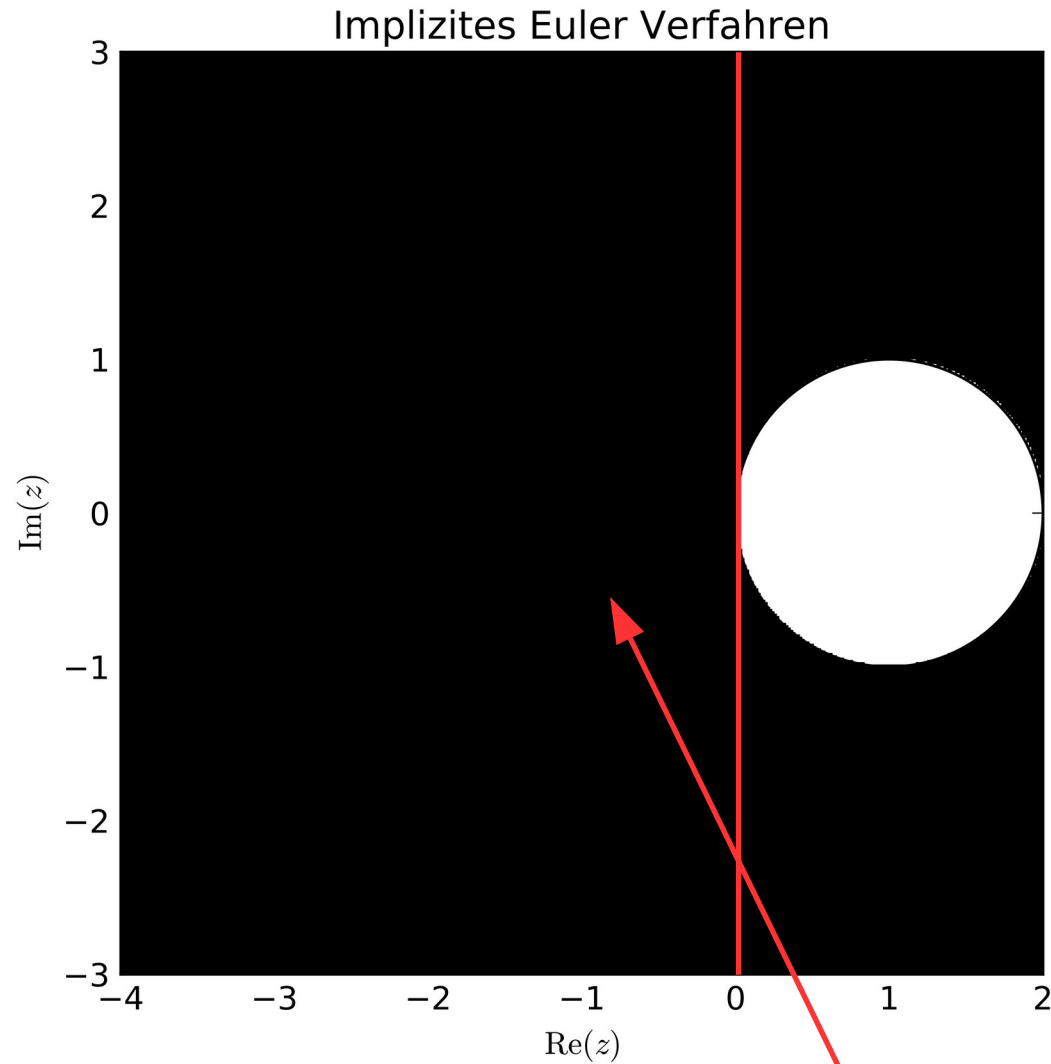
Stabilitätsgebiet

Stabilitätsgebiete



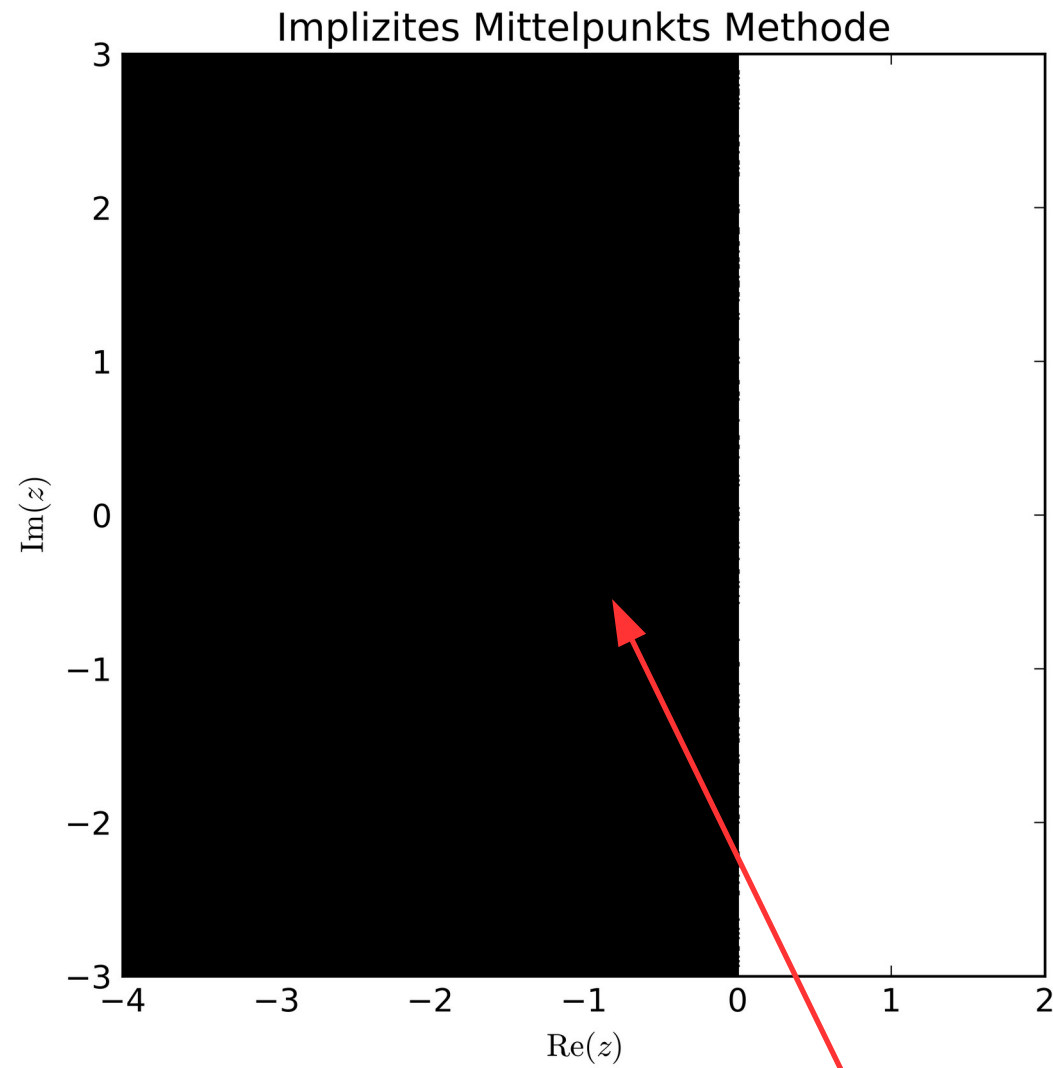
Stabilitätsgebiet

Stabilitätsgebiete



Stabilitätsgebiet

Stabilitätsgebiete



Stabilitätsgebiet

V.2 Implizite Runge-Kutta Verfahren

Ein allgemeines RK ESV mit s Stufen ist gegeben durch folgendes Butcher Tableau:

c_1	a_{11}	a_{12}	\dots	$a_{1,s-1}$	a_{1s}	$\vec{c} \mid A$ \vec{b}
c_2	a_{21}	a_{22}	\dots	$a_{2,s-1}$	a_{2s}	
\vdots	\vdots	\vdots	\ddots	\vdots	\vdots	
\vdots	\vdots	\vdots	\ddots	\vdots	\vdots	
c_s	a_{s1}	a_{s2}	\dots	$a_{s,s-1}$	a_{ss}	
	b_1	b_2	\dots	b_{s-1}	b_s	

Wenn A eine untere Dreiecksmatrix mit Nullen auf der Diagonalen ist, dann ist das RK Verfahren explizit.

Sonst ist es implizit ~~mo~~ i.A. muss ein nichtlineares Gleichungssystem gelöst werden!

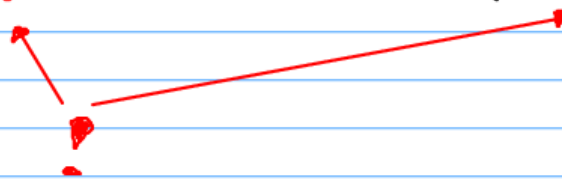
Ausgeschrieben

$$k_1 = f(t_j + c_1 \cdot h, y_j + h \cdot (a_{11} \cdot k_1 + a_{12} \cdot k_2 + \dots + a_{1s} \cdot k_s))$$

$$k_2 = f(t_j + c_2 \cdot h, y_j + h \cdot (a_{21} \cdot k_1 + a_{22} \cdot k_2 + \dots + a_{2s} \cdot k_s))$$

⋮

$$k_s = f(t_j + c_s \cdot h, y_j + h \cdot (a_{s1} \cdot k_1 + a_{s2} \cdot k_2 + \dots + a_{ss} \cdot k_s))$$



Für skalare DGL sind dies s i.A. nicht lineare Gleichungen für s Unbekannte (k_1, k_2, \dots, k_s) .

Für ein System von n DGLen sind dies $?$ i.A. nicht lineare Gleichungen für $?$ Unbekannte $(\vec{k}_1, \vec{k}_2, \dots, \vec{k}_s)$.

Dies ist natürlich sehr aufwendig und deshalb nutzt man implizite Verfahren nur wenn es sich lohnt!

↳ Steife Probleme

Bsp.: (4) Impliziter Euler $\begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array}$

(5) Implizite Mittelpunkts-Methode (KO $p=2$)

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

(6) Implizite Trapez-Methode (KO $p=2$)

Ausgeschrieben $\begin{array}{c|cc} 0 & & \\ \hline 1 & 1/2 & 1/2 \\ \hline & 1/2 & 1/2 \end{array}$

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

$$k_2 = f(t_j + h, y_j + \frac{h}{2}(k_1 + k_2))$$

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

oft wird sie geschrieben als

$$y_{j+1} = y_j + \frac{h}{2} (f(t_j, y_j) + f(t_{j+1}, y_{j+1}))$$

(7) RK-Gauss Verfahren (KO $p=4$)

	$1/2 - \sqrt{3}/6$	$1/4$	$1/4 - \sqrt{3}/6$
Knoten \rightarrow	$1/2 + \sqrt{3}/6$	$1/4 + \sqrt{3}/6$	$1/4$
Gauss-Legendre Gewichte \rightarrow		$1/2$	$1/2$

(8) SDIRK (KO $p=3$)

Singly Diagonal
Implicit RK

$$\begin{array}{c|cc} \gamma & \gamma & \\ \hline 1-\gamma & 1-2\gamma & \gamma \\ \hline & 1/2 & 1/2 \end{array}$$

$$\gamma = \frac{3 \pm \sqrt{5}}{6}$$

Hier muss man auch nichtlineare Gleichungen lösen... Aber was ist ein Vorteil von SDIRK Methoden?

Konvergenz impliziter RK-ESV

Schreiben wir die RK-ESV Stufen so:

$$\vec{v} = \vec{y}_j \vec{\lambda} + h A \vec{F}(t_j, \vec{v})$$

RK-Matrix

$$\begin{pmatrix} \vec{v}^{(1)} \\ \vdots \\ \vec{v}^{(s)} \end{pmatrix} = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \begin{pmatrix} \vec{F}(t_j + c_1 h, \vec{v}^{(1)}) \\ \vdots \\ \vec{F}(t_j + c_s h, \vec{v}^{(s)}) \end{pmatrix}$$

Damit

$$\vec{y}_{j+1} = \vec{y}_j + h \vec{z}^T \vec{F}(t_j, \vec{v})$$

Man muss also in jedem Schritt ein i.A. nicht-lineares s. n System lösen

Anzahl Stufen Lösungskomponenten

$$\vec{G}(\vec{v}) = \vec{v} - \vec{y}_j \vec{\lambda} - h A \vec{F}(t_j, \vec{v}) \stackrel{!}{=} 0$$

Konvergenz impliziter RK-ESV

In der Praxis verwendet man deshalb Newton-Methoden. Die Jacobi-Matrix hat die Form

$$J(\vec{y}) = \left(\delta_{il} - h a_{il} \frac{\partial f_i}{\partial y_l} (t_j + c_i h, \vec{y}^{(i)}) \right)_{i,l=1}^s$$

Dies ist immer noch aufwendig!

Deshalb verwendet man oft sog. "vereinfachte" Newton Verfahren welche die Jacobi-Matrix nur einmal auswerten $J(\vec{y}_j)$ und einfrieren (d.h. die LU/LR-Zerlegung von J muss man nur einmal machen).

sog. linearen Mehrschrittmethoden von der Form:

$$\sum_{l=0}^k \alpha_l \cdot y_{j+l-1} = h \cdot \sum_{l=0}^k \beta_l \cdot f_{j+l-1}$$

wobei $f_{j+l-1} = f(t_{j+l-1}, y_{j+l-1})$ und α_l, β_l Koeffizienten sind.

Spezialfälle beschreiben folgende Verfahren:

- Adams-Bashforth: $\alpha_0 = 1, \alpha_1 = -1$ und $\alpha_l = 0$ für $l > 1$
↳ Übungen

- $\beta_0 = 0$ ← explizit

- Adams-Moulton: $\alpha_0 = 1, \alpha_1 = -1$ und $\alpha_l = 0$ für $l > 1$
↳ Übungen

- $\beta_0 \neq 0$ ← implizit

- Rückwärtsdifferenziermethoden (Backward Differencing Methods BDF): $\beta_0 \neq 0$ und $\beta_l = 0$ für $l \geq 1$
↳ implizit

(g) BDF1: $k=1$

Bestimme das Interpolationspolynom durch

y_{j+1}, y_j :

$$p_1(t) = y_{j+1} \cdot \frac{t - t_j}{h} - y_j \cdot \frac{t - t_{j+1}}{h}$$

Die Ableitung zur Zeit t_{j+1} ist dann

$$\left. \frac{d}{dt} p_1(t) \right|_{t=t_{j+1}} = \frac{y_{j+1} - y_j}{h} \approx \dot{y}(t)$$

Und damit

$$\frac{y_{j+1} - y_j}{h} = f(t_{j+1}, y_{j+1})$$

Oder

$$y_{j+1} - y_j = h \cdot f(t_{j+1}, y_{j+1})$$

BDF1 entspricht dem impliziten Euler-Verfahren

(10) BDF2: k=2

Bestimme das Interpolationspolynom durch

y_{j+1}, y_j, y_{j-1} :

$$\begin{aligned} p_2(t) &= y_{j-1} \cdot \frac{1}{2h^2} (t-t_j)(t-t_{j+1}) \\ &\quad - y_j \cdot \frac{1}{h^2} (t-t_{j-1})(t-t_{j+1}) \\ &\quad + y_{j+1} \cdot \frac{1}{2h^2} (t-t_{j-1})(t-t_j) \end{aligned}$$

Die Ableitung zur Zeit t_{j+1} ist

$$\begin{aligned} \left. \frac{d}{dt} p_2(t) \right|_{t=t_{j+1}} &= y_{j-1} \cdot \frac{1}{2h} - y_j \cdot \frac{2}{h} + y_{j+1} \cdot \frac{3}{2h} \\ &\approx \dot{y}(t) \end{aligned}$$

Und damit

$$\frac{1}{h} \left(\frac{1}{2} y_{j-1} - 2y_j + \frac{3}{2} y_{j+1} \right) = F(t_{j+1}, y_{j+1})$$

Oder

$$y_{j+1} - \frac{4}{3} y_j + \frac{1}{3} y_{j-1} = \frac{2}{3} h F(t_{j+1}, y_{j+1})$$

$\alpha_0 = 1$ α_1 α_2 β_0

Konvergenz von linearen Mehrschrittverfahren

Die Konsistenzordnung wird wie bei ESV über den LDF definiert

$$e_{j+1} = y(t_{j+1}) - \frac{1}{\alpha_0} \left(- \sum_{l=1}^k \alpha_l y(t_{j+1-l}) + \sum_{l=0}^k \beta_l f(t_{j+1-l}, y(t_{j+1-l})) \right)$$

Also "einfach" um wieviel die exakte Lösung die diskrete bzw. approximierete Diff.-Gleichung nicht erfüllt. (Taylor-Entwicklung ...)

Der Konsistenzfehler ist

$$\tau_{j+1} = \frac{e_{j+1}}{h}$$

und die Konsistenzordnung \tilde{p} wie bei ESV $\tau = \mathcal{O}(h^{\tilde{p}})$. (s. Übung 509A04)

Aber: Im Gegensatz zu ESV impliziert Konsistenz noch nicht Konvergenz.

(s. z.B. Dahmen & Reusken (2006))

Steifes lineares AWP

$$\dot{\mathbf{y}}(t) = A\mathbf{y}(t)$$

$$\mathbf{y} = \begin{pmatrix} y_1(t) \\ y_2(t) \\ y_3(t) \end{pmatrix} \quad A = \begin{pmatrix} -\frac{1}{2} & -\frac{869}{10} & \frac{1521}{5} \\ 0 & -\frac{227}{2} & \frac{591}{2} \\ 0 & \frac{591}{2} & -\frac{1803}{2} \end{pmatrix}$$

$$\mathbf{y}_0 = \begin{pmatrix} 1 \\ 6 \\ 2 \end{pmatrix} \quad 0 \leq t \leq 2$$

Steifes lineares AWP

$$\begin{aligned}\dot{\mathbf{y}}(t) &= A\mathbf{y}(t) \\ &= PDP^{-1}\mathbf{y}(t)\end{aligned}$$

Diagonal!

D.h. „von Links“

$\times P^{-1} \rightarrow$

$$P^{-1}\dot{\mathbf{y}}(t) = D \underbrace{P^{-1}\mathbf{y}(t)}_{\mathbf{z}(t)}$$

$$\dot{\mathbf{z}}(t) = D\mathbf{z}(t) \quad \text{ENTKOPPELT!!!}$$

Steifes lineares AWP

Durch rechnen (z.B. mit einem CAS!):

$$D = \begin{pmatrix} -\frac{1}{2} & 0 & 0 \\ 0 & -15 & 0 \\ 0 & 0 & -1000 \end{pmatrix}$$

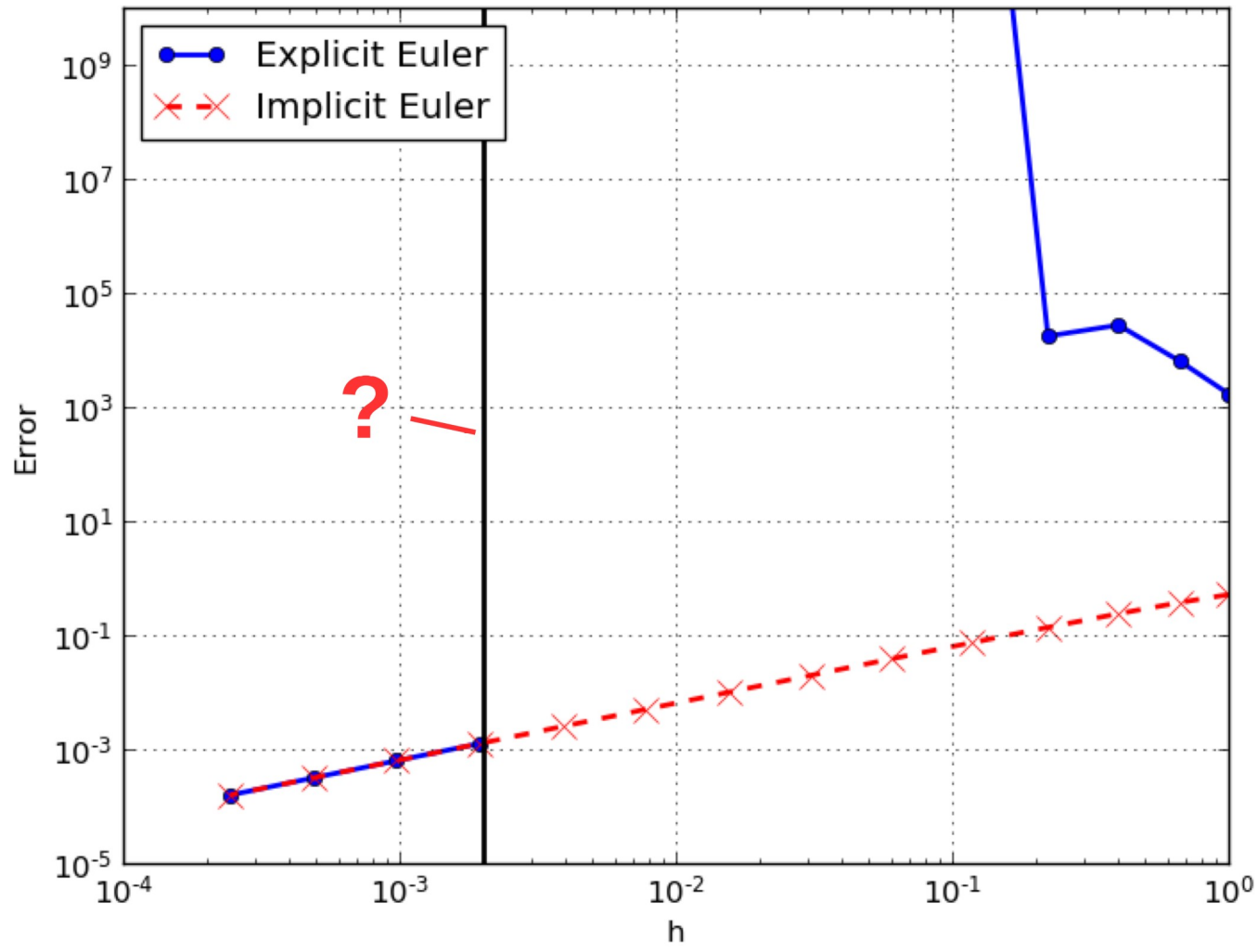
$$P = \begin{pmatrix} 15 & -12 & 1 \\ 0 & 12 & 1 \\ 0 & 4 & -3 \end{pmatrix} \quad P^{-1} = \begin{pmatrix} \frac{1}{15} & \frac{4}{75} & \frac{1}{25} \\ 0 & \frac{3}{40} & \frac{1}{40} \\ 0 & \frac{1}{10} & -\frac{3}{10} \end{pmatrix}$$

Durch einsetzen der AW:

$$\mathbf{y}(t) = \begin{pmatrix} 7e^{-\frac{1}{2}t} - 6e^{-15t} \\ 6e^{-15t} \\ 2e^{-15t} \end{pmatrix}$$

Beachte: der schnellste ablingende Mode ist nicht Teil der exakten Lösung!

Steifes lineares AWP



Erinnerung: Aus Bsp. (2) wissen wir für den expliziten Euler $0 < h < \frac{2}{\lambda}$

Steifigkeit tritt auch oft bei nichtlinearen DGLen auf

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

nicht lineare Vektorwertige Fkt.

Hier definiert man ein lokales Mass der Steifheit durch linearisieren an einem (interessanten) Punkt t_n, \vec{y}_n :

Jacobi-Matrix $\frac{\partial \vec{F}}{\partial \vec{y}}$

$$\vec{F}(t, \vec{y}(t)) = \vec{F}(t_n, \vec{y}_n) + \frac{\partial \vec{F}}{\partial t}(t_n, \vec{y}_n) \cdot (t - t_n) + \mathcal{J}(t_n, \vec{y}_n) \cdot (\vec{y} - \vec{y}_n)$$

Durch rearrangieren der Terme, erhält man ein inhom. lin. System

$$\dot{\vec{y}}(t) = \underbrace{\mathcal{J}(t_n, \vec{y}_n)}_A \vec{y}(t) + \left(\underbrace{\vec{F}(t_n, \vec{y}_n) + \frac{\partial \vec{F}}{\partial t}(t_n, \vec{y}_n) (t - t_n) - \mathcal{J}(t_n, \vec{y}_n) \vec{y}_n}_{\vec{b}} \right)$$

Ist die Linearisierung steif, so nennt man das nichtlineare System von DGLen **lokal steif**

Steifes nichtlineares AWP

$$\dot{y}_A = -0.1y_A + 100y_B y_C$$

$$\dot{y}_B = +0.1y_A - 100y_B y_C - 500y_B^2$$

$$\dot{y}_C = +500y_B^2 - 0.5y_C$$

$$y_A(0) = 0.5 \quad y_B(0) = 0.5 \quad y_C(0) = 0.5$$

$$0 \leq t \leq 1$$

Steifes nichtlineares AWP

$$\dot{y}_A = -0.1y_A + 100y_B y_C$$

$$\dot{y}_B = +0.1y_A - 100y_B y_C - 500y_B^2$$

$$\dot{y}_C = +500y_B^2 - 0.5y_C$$

NICHTLINEAR!!!

$$\mathbf{y} = \begin{pmatrix} y_A \\ y_B \\ y_C \end{pmatrix} \quad \mathbf{f}(x, \mathbf{y}) = \begin{pmatrix} -0.1y_A + 100y_B y_C \\ +0.1y_A - 100y_B y_C - 500y_B^2 \\ +500y_B^2 - 0.5y_C \end{pmatrix}$$

$$y_A(0) = 0.5 \quad y_B(0) = 0.5 \quad y_C(0) = 0.5 \quad \mathbf{y}_0 = \begin{pmatrix} 0.5 \\ 0.5 \\ 0.5 \end{pmatrix}$$

Steifes nichtlineares AWP

Steif?

Linearisieren wir die rechte Seite der DGL:

$$\mathbf{f}(t, \mathbf{y}) = \begin{pmatrix} -0.1y_A + 100y_B y_C \\ +0.1y_A - 100y_B y_C - 500y_B^2 \\ +500y_B^2 - 0.5y_C \end{pmatrix}$$

$$\mathbf{f}(t, \mathbf{y}) \approx \mathbf{f}(t_1, \mathbf{y}_1) + \underbrace{\frac{\partial \mathbf{f}}{\partial t}(t_1, \mathbf{y}_1)}_0 + \underbrace{\frac{\partial \mathbf{f}}{\partial \mathbf{y}}(t_1, \mathbf{y}_1)}_{/}$$

$$J(t, \mathbf{y}) = \frac{\partial \mathbf{f}}{\partial \mathbf{y}}(t, \mathbf{y}) = \begin{pmatrix} -0.1 & 100y_C & 100y_B \\ +0.1 & -1000y_B - 100y_C & -100y_B \\ 0 & 1000y_B & -0.5 \end{pmatrix}$$

Steifes nichtlineares AWP

Lokal Steif?

Linearisieren wir um den AW: $t_0 = 0$ $\mathbf{y}_0 = \begin{pmatrix} 0.5 \\ 0.5 \\ 0.5 \end{pmatrix}$

$$J(t_0, \mathbf{y}_0) = \frac{\partial \mathbf{f}}{\partial \mathbf{y}}(t_0, \mathbf{y}_0) = \begin{pmatrix} -0.1 & 50 & 50 \\ +0.1 & -550 & -50 \\ 0 & 500 & -0.5 \end{pmatrix}$$

MATLAB: eig \longrightarrow

$$\lambda_1 \approx -5.00 \times 10^2$$

$$\lambda_2 \approx -9.87 \times 10^{-4}$$

$$\lambda_3 \approx -5.07 \times 10^1$$

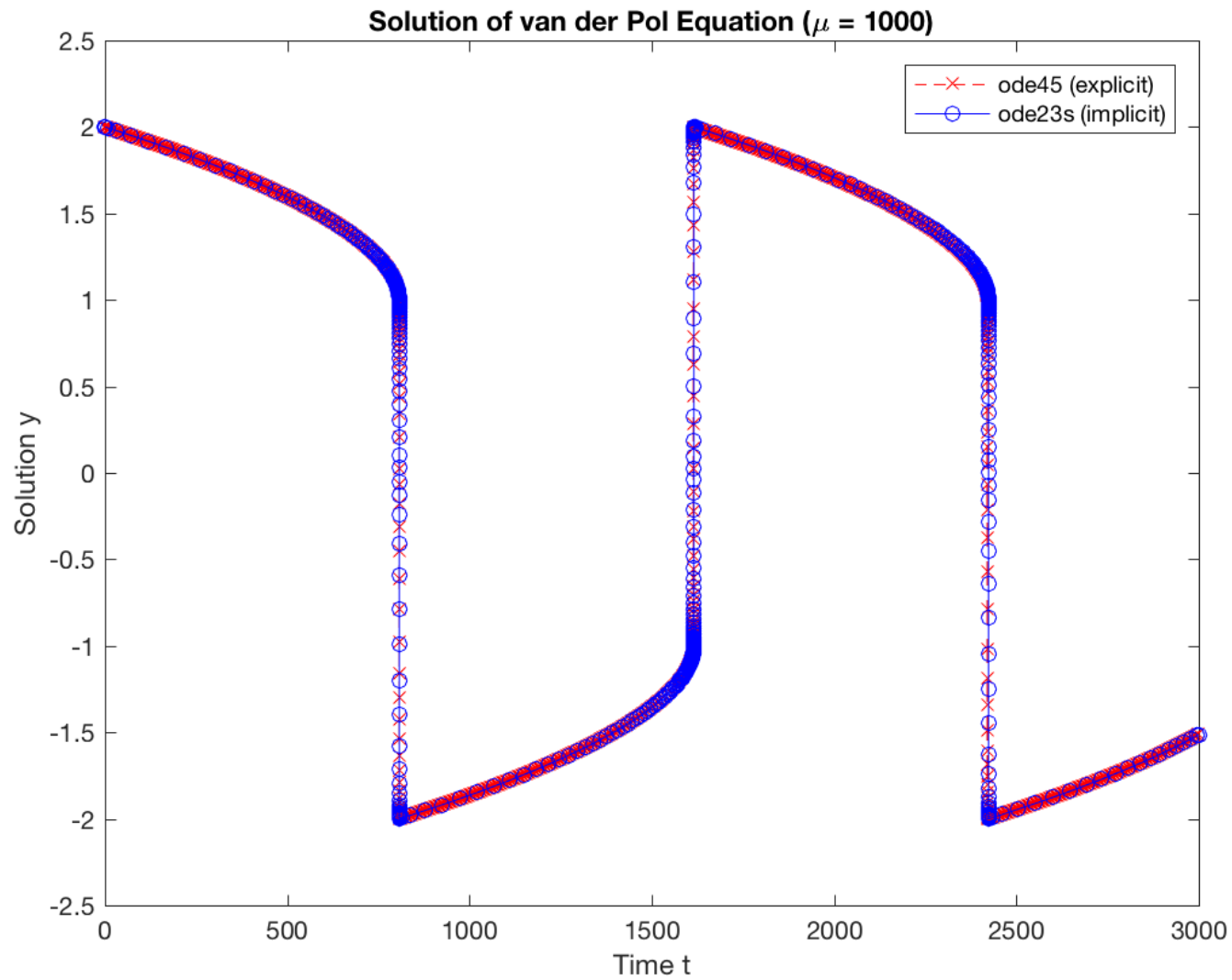
$$S = \frac{\max_j |\operatorname{Re}(\lambda_j)|}{\min_j |\operatorname{Re}(\lambda_j)|} \approx 5.06 \times 10^5 \quad \mathbf{JA!!!}$$

Van der Pol Oszillator

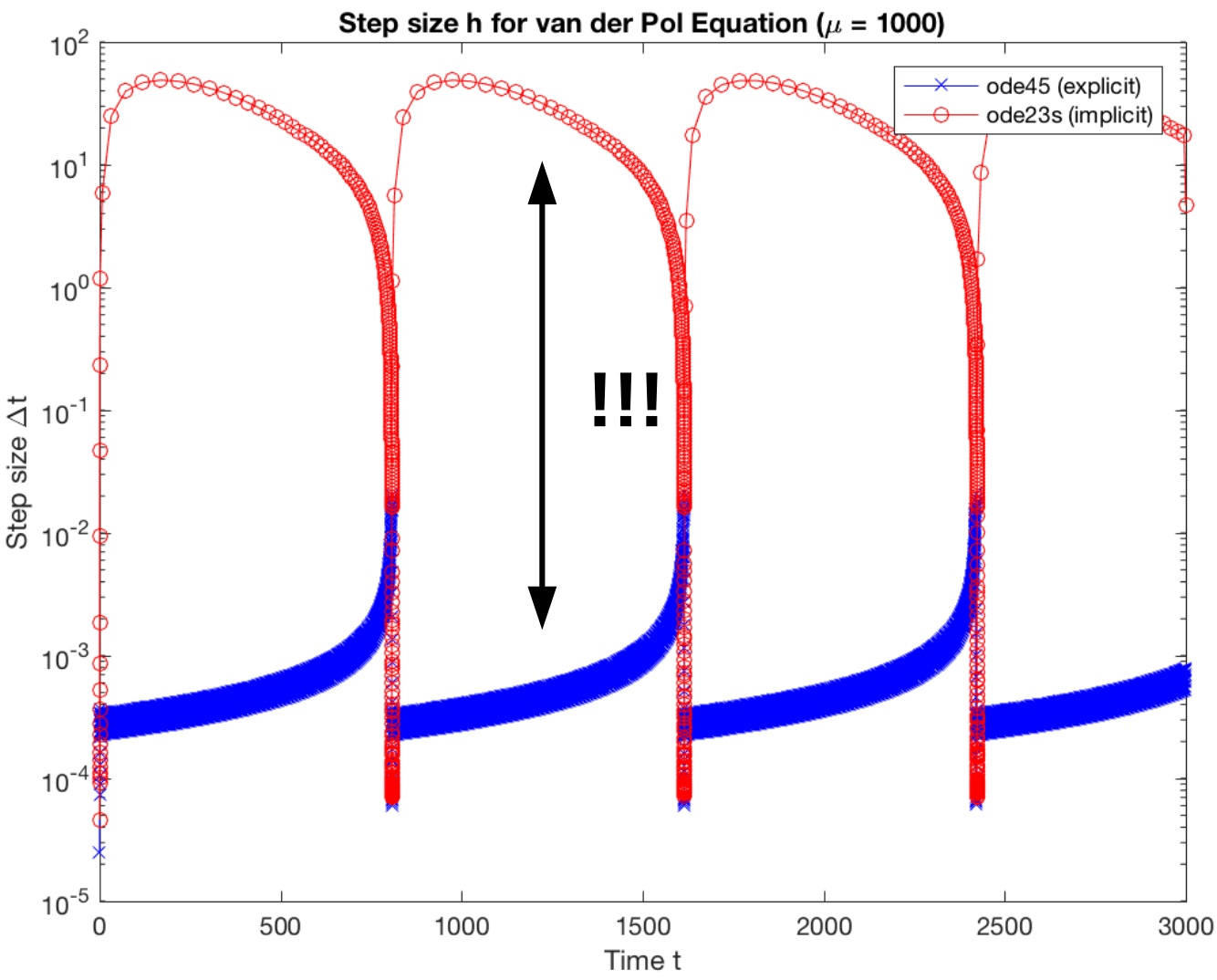
$$\begin{aligned} \dot{y}_1 &= y_2 \\ \dot{y}_2 &= \mu(1 - y_1^2)y_2 - y_1 \end{aligned} \quad \mu = 1000 \quad \mathbf{y}_0 = \begin{pmatrix} 0 \\ 2 \end{pmatrix} \quad t \in [0, 3000]$$

```
>> StiffVanDerPol
Loeser: ode45
Elapsed time is 73.490408 seconds.
Loser: ode23s
Elapsed time is 0.167961 seconds.
```

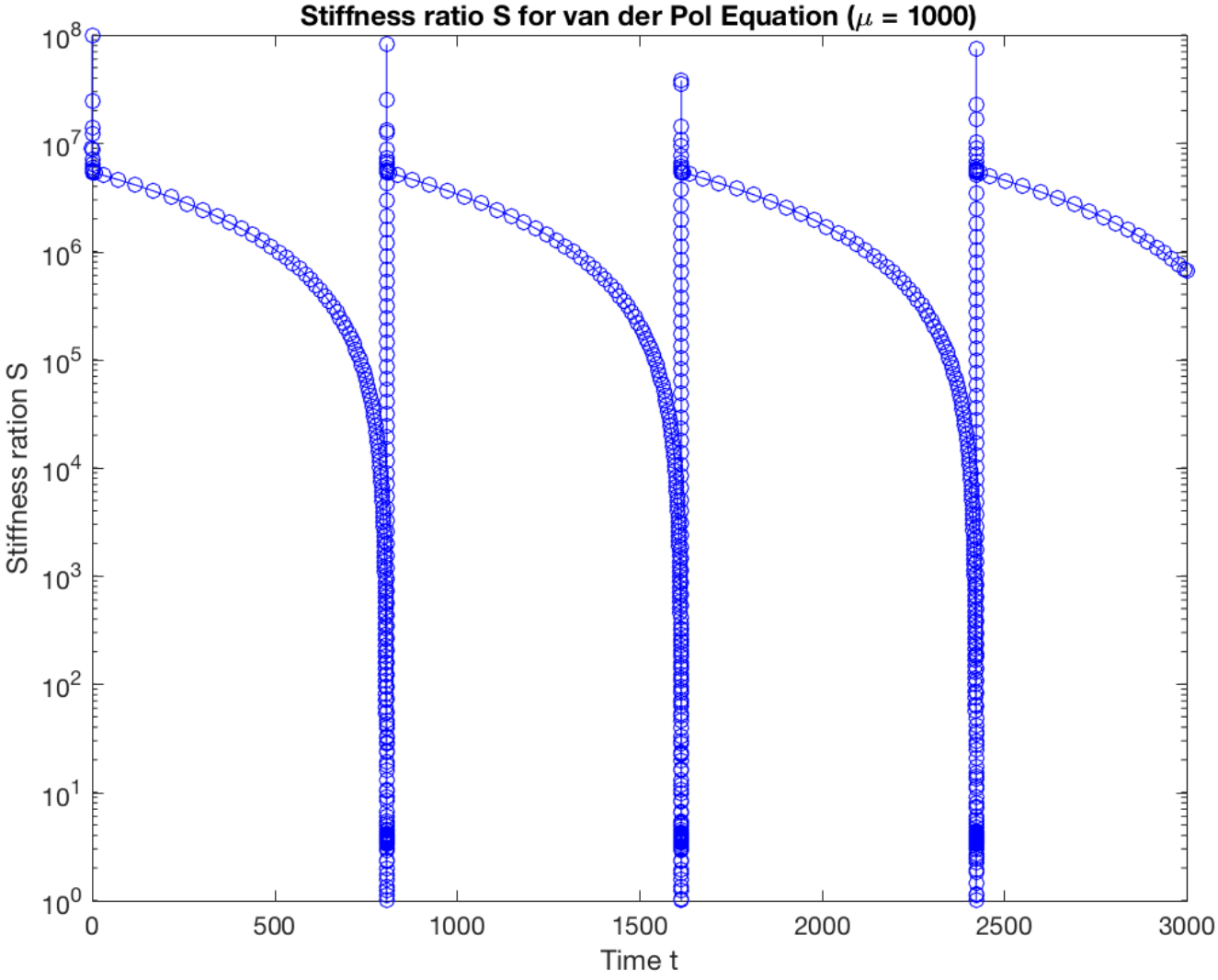
Van der Pol Oscillator



Van der Pol Oscillator



Van der Pol Oscillator



Zur numerischen Behandlung steifer Probleme
folgen wir aus Bsp. (11)-(14), dass
explizite Verfahren ungeeignet sind.

s. nächste
Seite

D.h. ineffizient da die Schrittweite
aus Stabilität- und NICHT
Genauigkeits-Gründen gewählt
werden muss

explizit		implizit
günstig pro Schritt		teuer pro Schritt
Schrittweite limitiert durch schnellste abfallende Komponente		Schrittweite nur durch gewünschte Genauigkeit limitiert

Bsp. (14)

Bsp.: (14) Die Wärmeleitungsgleichung (in ihrer einfachsten Form) in 2D für die Temperatur $u(x, y, t)$

$$\frac{\partial u}{\partial t} = \Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$$

wobei $(x, y) \in [0, 1]^2$ und Zeit $t \geq 0$.

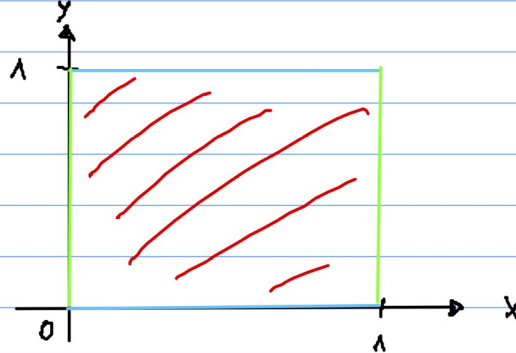
Die Anfangswerte seien

$$u(x, y, 0) = \underline{\underline{0(x, y)}}$$

und die Randwerte

$$\underline{u(x, 0/1, t)} = \underline{u(0/1, y, t)} = 0$$

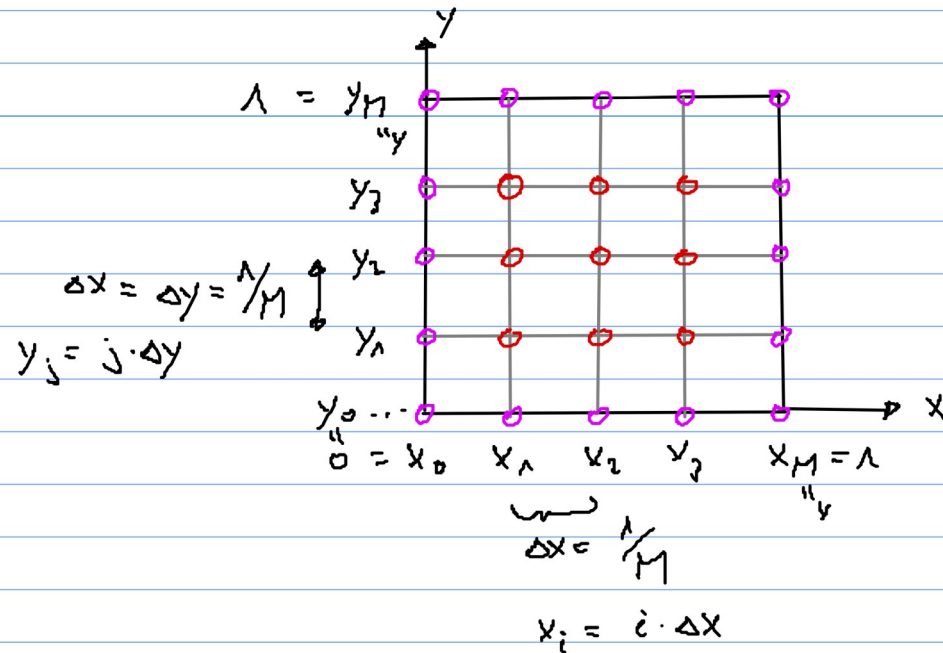
Graphisch



Also ein Anfangs-Randwert-Problem.

Bsp. (14)

Verwenden wir die sog. Linien-
Methode (method of lines) und
diskretisieren das Problem zuerst
in den Ortsvariablen. Dazu
führen wir ein $(M+1) \times (M+1)$ Gitter
(grid/mesh) ein:



Die Temperatur an den Gitter-Knoten
wollen wir approximieren:

$$u(x_i, y_j, t) \approx u_{i,j}(t)$$

Bsp. (14)

Dazu müssen wir die PDE approximieren:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$$

Finite Differenzen
s. Kap. 1

$$\approx \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{\Delta y^2} = \frac{du_{i,j}}{dt}$$

Für $1 \leq i, j \leq M-1$ (0-Knoten)

$u_{i,j} = 0$ am Rand (0-Knoten)

Dies ist ein lineares System

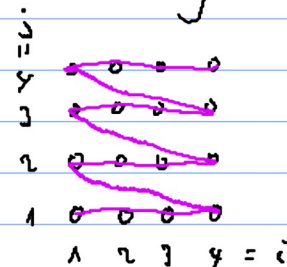
von $(M-1) \times (M-1)$ DGLen mit AW

$$u_{i,j}(t=0) = \bar{u}(x_i, y_j) \text{ für } 1 \leq i, j \leq M-1.$$

Die Temperaturen an den Gitter-Knoten packen wir in einen Lösungsvektor

$$\vec{y} = \begin{pmatrix} u_{1,1} \\ u_{2,1} \\ \vdots \\ u_{M-1, M-1} \end{pmatrix}$$

$(M-1)^2$ Komponenten



Bsp. (14)

und schreiben das DGL System wie folgt

$$\dot{\vec{y}} = A \vec{y}$$

wobei die (Block-) Matrix

$$A = \begin{pmatrix} \gamma - I & & & 0 \\ -I & \gamma & & \\ & & \ddots & \\ & & & \gamma - I \\ 0 & & & -I & \gamma \end{pmatrix} \quad \begin{matrix} (n-1)^2 \times (n-1)^2 \\ \text{Matrix} \end{matrix}$$

und

$$\gamma = \begin{pmatrix} \gamma - \lambda & & & 0 \\ -\lambda & \gamma & & \\ & & \ddots & \\ & & & \gamma - \lambda \\ 0 & & & -\lambda & \gamma \end{pmatrix} \quad \begin{matrix} (n-1) \times (n-1) \\ \text{Matrix} \end{matrix}$$

$$I = \begin{pmatrix} \lambda & & & 0 \\ & & & \\ & & \ddots & \\ 0 & & & \lambda \end{pmatrix} \quad \begin{matrix} (n-1) \times (n-1) \\ \text{Einheitsmatrix} \end{matrix}$$

Die Matrix A ist dünn besetzt (sparse, viele Nullen!).

Bsp. (14)

Für $M=100$ ($\Delta x=0.01$) ergibt sich
bereits ein System von $99^2=9801$ DGLen!

(Man stellt sich leicht vor, dass in
der Praxis viel grössere Systeme
gibt.)

Nun müssen wir dieses AWP
noch lösen. Um das Lösen eines
grossen LGS zu vermeiden ist
man versucht einen expliziten
Löser zu verwenden, z.B. das
explizite Euler Verfahren

$$\vec{y}^{n+1} = \vec{y}^n + h A \vec{y}^n \quad n=0,1,\dots$$

Zeitindex

Aber man merkt schnell, dass dies
eine schlechte Idee ist.

Das DGL System ist nämlich sehr
steif. Man kann zeigen, dass die
Eigenwerte von A mit $\frac{1}{\Delta x^2}$
skalieren. ▽

Bsp. (14)

Für $\Delta x = 0.01$ ($M = 100$) ergeben
sich so ungefähr 12'000 Zeit-
Schritte bis zu $t = 0.1$.