PVK Numerische Methoden
Tag 1

Lucas Böttcher

ETH Zürich
Institut für Baustoffe
Wolfgang-Pauli-Str. 27
HIT G 23.8
8093 Zürich

lucasb@ethz.ch

June 19, 2017

Übersicht

Daten und Zeiten:

Kursdauer: 19.06.2017-23.06.2017 von 08:00-12:30 Uhr
Erste Lektion: 08:00-09:30 Uhr
Pause: 09:30-09:45 Uhr
Zweite Lektion: 09:45-11:15 Uhr
Pause: 11:15-11:30 Uhr
Dritte Lektion: 11:30-12:30 Uhr

Übersicht

Ziel PVK: Vorbereitung auf die Prüfung (schriftlich, 180 min) am PC

Übersicht

Ziel PVK: Vorbereitung auf die Prüfung (schriftlich, 180 min) am PC

Erlaubte Hilfsmittel: 10 Seiten A4 eigenhändig verfasste Zusammenfassung, handgeschrieben, nicht ausgedruckt, nicht kopiert. Vorlesungsunterlagen stehen elektronisch während der Prüfung zur Verfügung. Sonst keine Hilfsmittel.

Übersicht

Kursziele laut VVZ:

Studenten sind im Stande

1. zwischen verschiedenen numerischen Methoden zu differenzieren, um zu wissen, welche Methode ein bestimmtes mathematisches Problem löst.

Übersicht

Kursziele laut VVZ:
Studenten sind im Stande

1. zwischen verschiedenen numerischen Methoden zu differenzieren, um zu wissen, welche Methode ein bestimmtes mathematisches Problem löst.
2. geeignete numerische Verfahren anzuwenden, um ein bestimmtes numerisches Problem der Physik zu lösen.

Übersicht

Kursziele laut VVZ:
Studenten sind im Stande

1. zwischen verschiedenen numerischen Methoden zu differenzieren, um zu wissen, welche Methode ein bestimmtes mathematisches Problem löst.
2. geeignete numerische Verfahren anzuwenden, um ein bestimmtes numerisches Problem der Physik zu lösen.
3. geeignete Software Repositories anzuwenden, welche hilfreich sind, um ein gegebenes numerisches Problem zu lösen.

Übersicht

Kursziele laut VVZ:
Studenten sind im Stande

1. zwischen verschiedenen numerischen Methoden zu differenzieren, um zu wissen, welche Methode ein bestimmtes mathematisches Problem löst.
2. geeignete numerische Verfahren anzuwenden, um ein bestimmtes numerisches Problem der Physik zu lösen.
3. geeignete Software Repositories anzuwenden, welche hilfreich sind, um ein gegebenes numerisches Problem zu lösen.
4. numerische Resultate, welche durch das numerische lösen mathematischer Probleme erhalten wurden, hinsichtlich Genauigkeit und Korrektheit zu interpretieren.

Übersicht

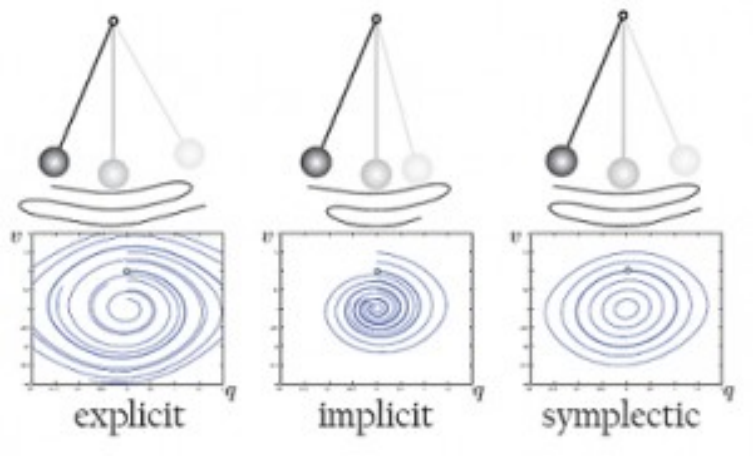
Kurstag	Inhalt
Tag 1	Differenzialgleichungen
Tag 2	Quadratur, Nichtlineare algebraische Gleichungen
Tag 3	Ausgleichsrechnung, Eigenwerte, Interpolation
Tag 4	Lineare ODEs, Exponentielle Integratoren, Splitting-Verfahren
Tag 5	Erweiterte Übungen

Kapitel I
Lösen gewöhnlicher Differentialgleichungen
(Sektionen 8.2, 8.5, 8.6 im Skript)

Lösen gewöhnlicher Differentialgleichungen

Lernziele: Numerische Lösung von Anfgangswertproblemen der Form $\dot{\mathbf{y}} = \mathbf{f}(t, \mathbf{y})$ mit $\mathbf{y}(0) = \mathbf{y}_0$.

Methoden: Eulerverfahren (explizit und implizit), implizite Mittelpunktsregel.



http://ftparmy.com/207981-pendulum-motion-in-phase-space-model.html

Explizites Eulerverfahren

Explizites Eulerverfahren

Grundidee: Approximieren der ODE $\dot{\mathbf{y}} = \mathbf{f}(t, \mathbf{y})$ durch
$\mathbf{y}_{n+1} = \mathbf{y}_n + h_n \mathbf{f}(t_n, \mathbf{y}_n)$, $n = 0, \ldots, N-1$.

Explizites Eulerverfahren

Explizites Eulerverfahren

Grundidee: Approximieren der ODE $\dot{\mathbf{y}} = \mathbf{f}(t, \mathbf{y})$ durch $\mathbf{y}_{n+1} = \mathbf{y}_n + h_n \mathbf{f}(t, \mathbf{y})$, $n = 0, \ldots, N-1$.

Denn der Vorwärts-Differenzen-Quotient ist $\mathbf{f}(t_n, \mathbf{y}(t_n)) = \dot{\mathbf{y}}(t_n) \approx \frac{\mathbf{y}(t_n+h_n)-\mathbf{y}(t_n)}{h_n}$.

Explizites Eulerverfahren

Explizites Eulerverfahren

Grundidee: Approximieren der ODE $\dot{\mathbf{y}} = \mathbf{f}(t, \mathbf{y})$ durch
$\mathbf{y}_{n+1} = \mathbf{y}_n + h_n \mathbf{f}(t, \mathbf{y})$, $n = 0, \ldots, N-1$.

Denn der Vorwärts-Differenzen-Quotient ist
$\mathbf{f}(t_n, \mathbf{y}(t_n)) = \dot{\mathbf{y}}(t_n) \approx \frac{\mathbf{y}(t_n+h_n)-\mathbf{y}(t_n)}{h_n}$.

Keine Energieerhaltung, cf. p. 226 im Skript.

Implizites Eulerverfahren

Implizites Eulerverfahren

Grundidee: Approximieren der ODE $\dot{\mathbf{y}} = \mathbf{f}(t, \mathbf{y})$ durch
$\mathbf{y}_{n+1} = \mathbf{y}_n + h_n \mathbf{f}(t_{n+1}, \mathbf{y}_{n+1})$, $n = 0, \ldots, N-1$.

Implizites Eulerverfahren

Implizites Eulerverfahren

Grundidee: Approximieren der ODE $\dot{\mathbf{y}} = \mathbf{f}(t, \mathbf{y})$ durch $\mathbf{y}_{n+1} = \mathbf{y}_n + h_n \mathbf{f}(t_{n+1}, \mathbf{y}_{n+1})$, $n = 0, \ldots, N-1$.

Denn der Rückwärts-Differenzen-Quotient ist $\mathbf{f}(t_{n+1}, \mathbf{y}(t_{n+1})) = \dot{\mathbf{y}}(t_{n+1}) \approx \frac{\mathbf{y}(t_{n+1} - h_n) - \mathbf{y}(t_{n+1})}{-h_n}$.

Implizites Eulerverfahren

Implizites Eulerverfahren

Grundidee: Approximieren der ODE $\dot{\mathbf{y}} = \mathbf{f}(t, \mathbf{y})$ durch
$\mathbf{y}_{n+1} = \mathbf{y}_n + h_n \mathbf{f}(t_{n+1}, \mathbf{y}_{n+1})$, $n = 0, \ldots, N-1$.

Denn der Rückwärts-Differenzen-Quotient ist
$\mathbf{f}(t_{n+1}, \mathbf{y}(t_{n+1})) = \dot{\mathbf{y}}(t_{n+1}) \approx \frac{\mathbf{y}(t_{n+1}-h_n)-\mathbf{y}(t_{n+1})}{-h_n}$.

Lösen eines nichtlinearen Systems notwendig.

Implizites Eulerverfahren

Implizites Eulerverfahren

Grundidee: Approximieren der ODE $\dot{\mathbf{y}} = \mathbf{f}(t, \mathbf{y})$ durch
$\mathbf{y}_{n+1} = \mathbf{y}_n + h_n \mathbf{f}(t_{n+1}, \mathbf{y}_{n+1})$, $n = 0, \ldots, N-1$.

Denn der Rückwärts-Differenzen-Quotient ist
$\mathbf{f}(t_{n+1}, \mathbf{y}(t_{n+1})) = \dot{\mathbf{y}}(t_{n+1}) \approx \frac{\mathbf{y}(t_{n+1}-h_n)-\mathbf{y}(t_{n+1})}{-h_n}$.

Lösen eines nichtlinearen Systems notwendig.

Keine Energieerhaltung, cf. p. 226 im Skript.

Implizite Mittelpunktsregel

Implizite Mittelpunktsregel

Grundidee: Approximieren der ODE $\dot{\mathbf{y}} = \mathbf{f}(t, \mathbf{y})$ durch

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h_n \mathbf{f}\left(\frac{t_n + t_{n+1}}{2}, \frac{\mathbf{y}_{n+1} + \mathbf{y}_n}{2}\right), \; n = 0, \ldots, N-1.$$

Implizite Mittelpunktsregel

Implizite Mittelpunktsregel

Grundidee: Approximieren der ODE $\dot{\mathbf{y}} = \mathbf{f}(t, \mathbf{y})$ durch

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h_n \mathbf{f}\left(\frac{t_n + t_{n+1}}{2}, \frac{\mathbf{y}_{n+1} + \mathbf{y}_n}{2}\right), \; n = 0, \ldots, N-1.$$

Lösen eines nichtlinearen Systems notwendig.

Implizite Mittelpunktsregel

Implizite Mittelpunktsregel

Grundidee: Approximieren der ODE $\dot{\mathbf{y}} = \mathbf{f}(t, \mathbf{y})$ durch

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h_n \mathbf{f}\left(\frac{t_n + t_{n+1}}{2}, \frac{\mathbf{y}_{n+1} + \mathbf{y}_n}{2}\right), \; n = 0, \ldots, N-1.$$

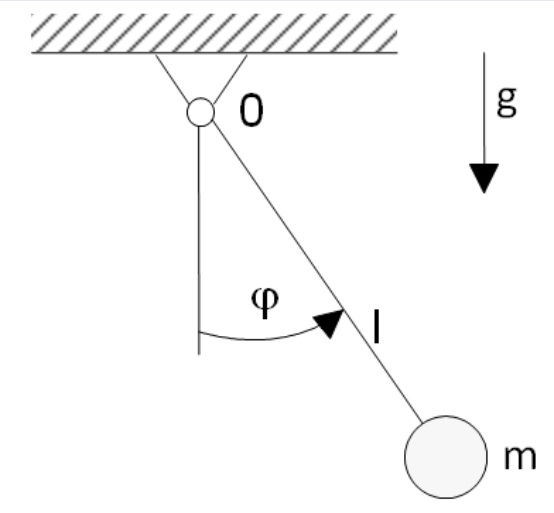
Lösen eines nichtlinearen Systems notwendig.

Energieerhaltung gegeben, cf. p. 227 im Skript.

Beispiel Mathematisches Pendel

Man schreibt die Differentialgleichung $\ddot{\phi} = -\frac{g}{l}\sin(\phi)$ um in ein System erster Ordnung:

$$\begin{pmatrix} \dot{\phi} \\ \dot{p} \end{pmatrix} = \begin{pmatrix} p \\ -\frac{g}{l}\sin(\phi) \end{pmatrix}. \tag{1}$$



http://moodle.autolab.uni-pannon.hu/Mecha_tananyag/mechatronikai_modellezes_angol/ch02.html

Untersuche das zeitliche Verhalten der Gesamtenergie für das explizite und implizite Eulerverfahren sowie für die Mittelpunktsregel.

Hinweis: Die Gesamtenergie ist ($m = 1$, $l = 1$) $E_{tot} = \frac{1}{2}\dot{\phi}^2 + g(1 - \cos(\phi))$.

Beispiel Mathematisches Pendel

```
f = lambda alpha, p: array([p, -9.81*sin(alpha)])

def explicit_euler(y0, dt, N):
    y_ee = []
    y_ee.append(y0)
    for i in range(N):
        y_ee.append( y_ee[-1] +
        dt*f(y_ee[-1][0],y_ee[-1][1]) )

    return asarray(y_ee)
```

Beispiel Mathematisches Pendel

```
g_impl = lambda x, y0, dt: y0+dt*f(x[0],x[1])-x

def implicit_euler(y0, dt, N):
    y_ie = []
    y_ie.append(y0)
    for i in range(N):
        y_ie.append( fsolve(g_impl, y_ie[-1],
        args=(y_ie[-1],dt)) )

    return asarray(y_ie)
```

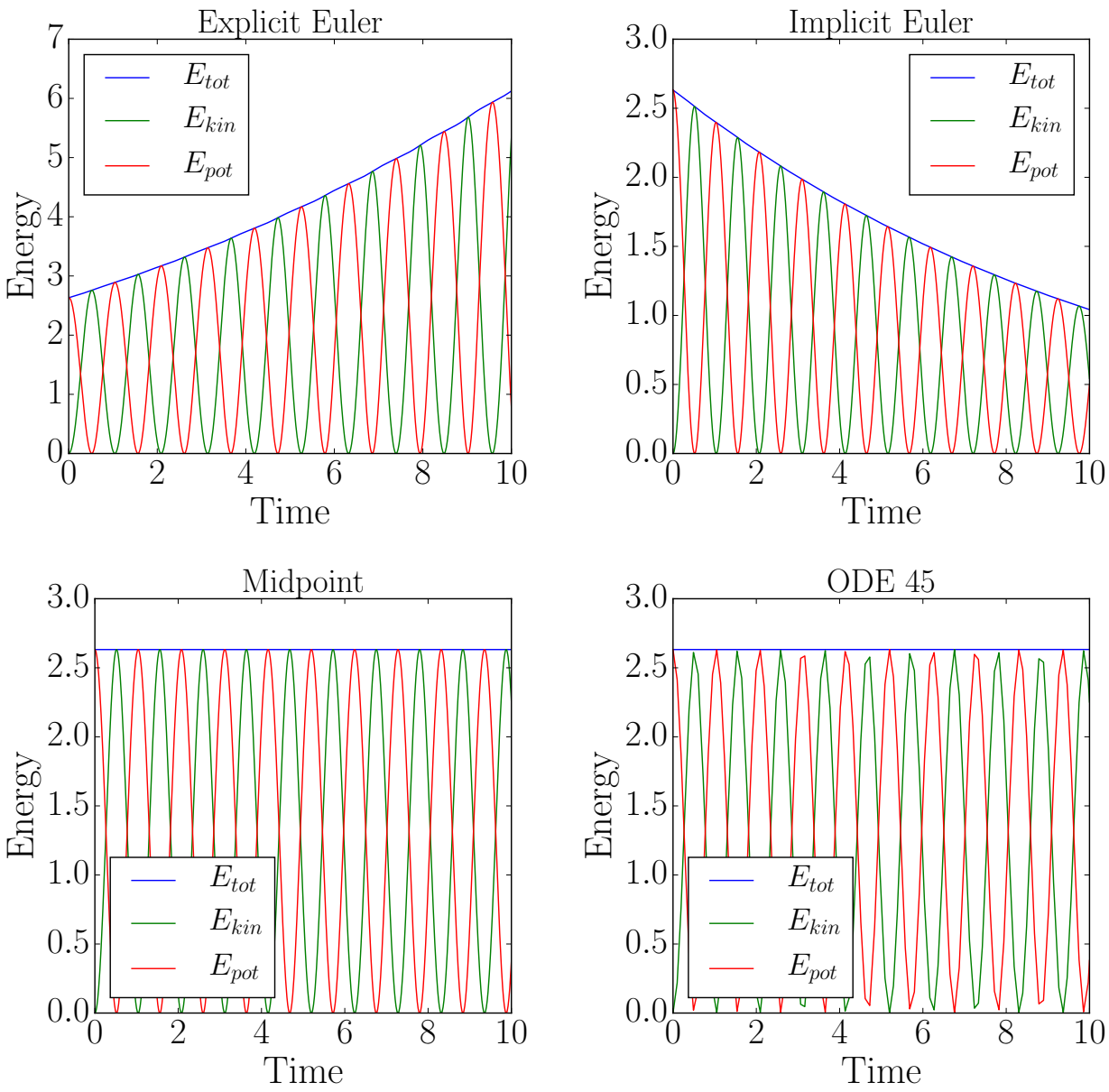
Beispiel Mathematisches Pendel

```
g_mid = lambda x, y0, dt: y0+
        dt*f(0.5*(x[0]+y0[0]),0.5*(x[1]+y0[1]))-x

def midpoint(y0, dt, N):
    y_mp = []
    y_mp.append(y0)
    for i in range(N):
        y_mp.append( fsolve(g_mid, y_mp[-1],
        args=(y_mp[-1],dt)) )

    return asarray(y_mp)
```

Beispiel Mathematisches Pendel



Verletverfahren

Lernziele: Integration von Differentialgleichungen der Form $\ddot{\mathbf{y}} = \mathbf{f}(t, \mathbf{y})$ (z.B.: Newtonsche Bewegungsgleichungen).

Methoden: Velocity-Verlet- und Störmer-Verlet-Verfahren.

Velocity-Verlet-Verfahren

Velocity-Verlet-Verfahren

Grundidee: Approximieren der ODE $\ddot{\mathbf{y}} = \mathbf{f}(t, \mathbf{y})$ mit Anfangsbedingungen $\mathbf{y}(0) = \mathbf{y}_0$ und $\dot{\mathbf{y}} = \mathbf{v}_0$.

In diesem Verfahren ist:
$\mathbf{x}_{n+1} = \mathbf{x}_n + h\mathbf{v}_n + \frac{h^2}{2} f(t_n, \mathbf{x}_n)$ und
$\mathbf{v}_{n+1} = \mathbf{v}_n + \frac{h}{2}\left(f(t_n, \mathbf{x}_n) + f(t_{n+1}, \mathbf{x}_{n+1})\right)$.

Energieerhaltung gegeben.

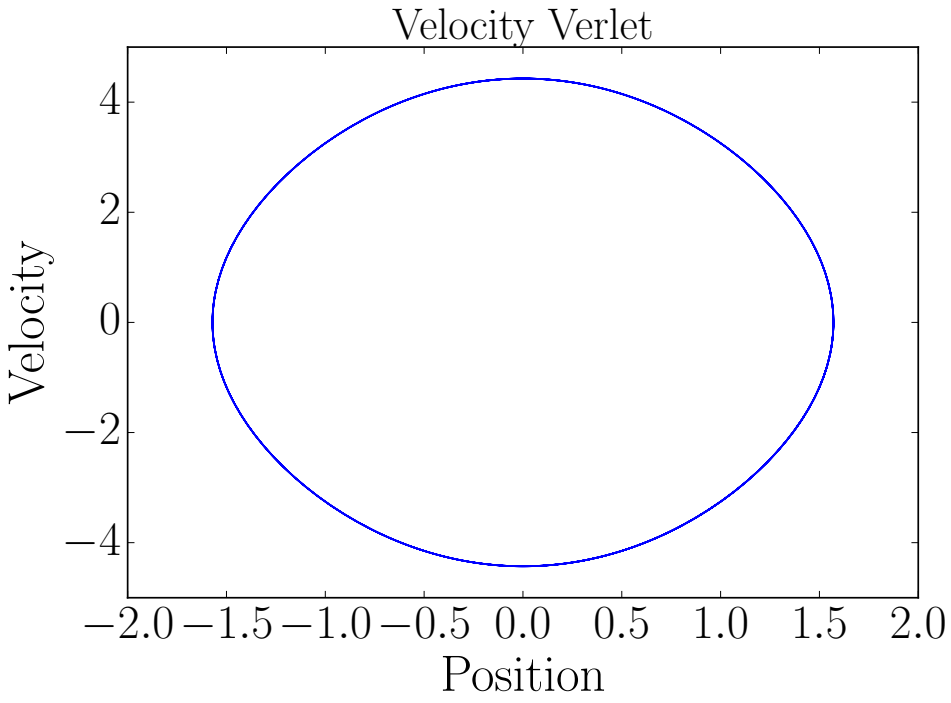
Velocity-Verlet-Verfahren

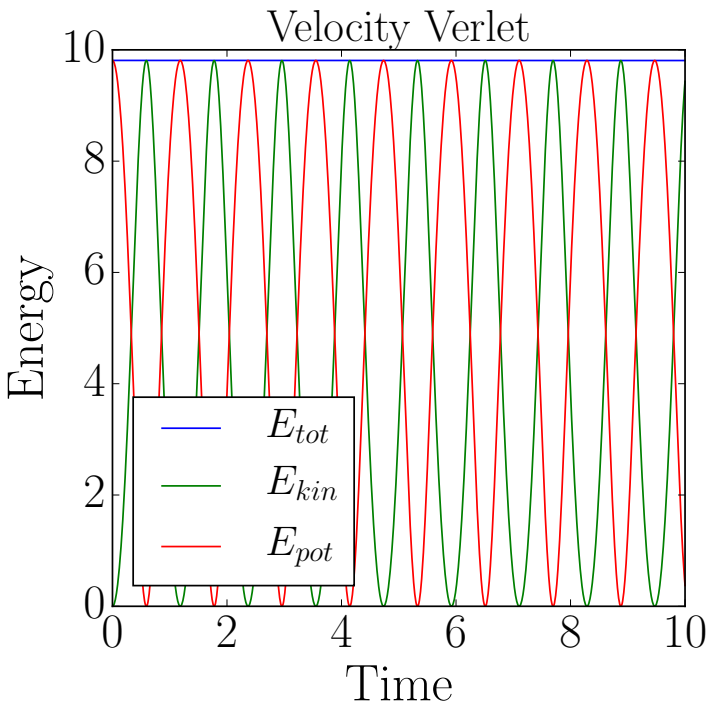
```
f = lambda phi: -9.81*sin(phi)

def velocity_verlet(x0, v0, dt, N):
    x_vv = []
    v_vv = []
    x_vv.append(x0)
    v_vv.append(v0)
    for i in range(N):
        x_vv.append( x_vv[-1] + dt*v_vv[-1] +
                     0.5*dt**2 * f(x_vv[-1]) )
        v_vv.append( v_vv[-1] +
                     0.5*dt*( f(x_vv[-1]) + f(x_vv[-2]) ))

    return asarray(x_vv), asarray(v_vv)
```

Velocity-Verlet-Verfahren





Das Velocity-Verlet-Verfahren erhält die Energie.

Störmer-Verlet-Verfahren

Störmer-Verlet-Verfahren

Grundidee: Approximieren der ODE $\ddot{\mathbf{y}} = \mathbf{f}(t, \mathbf{y})$ durch
$\mathbf{y}_{n+1} = -\mathbf{y}_{n-1} + 2\mathbf{y}_n + h^2\mathbf{f}(t_n, \mathbf{y}_n)$, $n = 0, \ldots, N-1$.

Störmer-Verlet-Verfahren

Störmer-Verlet-Verfahren

Grundidee: Approximieren der ODE $\ddot{\mathbf{y}} = \mathbf{f}(t, \mathbf{y})$ durch
$\mathbf{y}_{n+1} = -\mathbf{y}_{n-1} + 2\mathbf{y}_n + h^2\mathbf{f}(t_n, \mathbf{y}_n)$, $n = 0, \ldots, N-1$.

Denn man kann die zweite Ableitung approximieren durch
$\mathbf{f}(t, \mathbf{y}) = \ddot{\mathbf{y}} \approx \frac{y_{n+1} - 2y_n + y_{n-1}}{h^2}$.

Störmer-Verlet-Verfahren

Störmer-Verlet-Verfahren

Grundidee: Approximieren der ODE $\ddot{\mathbf{y}} = \mathbf{f}(t, \mathbf{y})$ durch
$\mathbf{y}_{n+1} = -\mathbf{y}_{n-1} + 2\mathbf{y}_n + h^2\mathbf{f}(t_n, \mathbf{y}_n)$, $n = 0, \ldots, N-1$.

Denn man kann die zweite Ableitung approximieren durch
$\mathbf{f}(t, \mathbf{y}) = \ddot{\mathbf{y}} \approx \frac{y_{n+1} - 2y_n + y_{n-1}}{h^2}$.

Typischerweise sind die Anfangswerte $\mathbf{y}(0) = \mathbf{y}_0$ und $\dot{\mathbf{y}} = \mathbf{v}_0$ gegeben.
Dann ist $\mathbf{y}_1 = \mathbf{y}_0 + h\mathbf{v}_0 + \frac{h^2}{2}\mathbf{f}(t_0, \mathbf{y}_0)$, cf. p. 228 im Skript.

Beispiel Streuung am Lennard-Jones Potential

Beispiel Streuung am Lennard-Jones Potential

Die Teilchenstreuung am Lennard-Jones Potential $U(x, y)$ ist beschrieben durch $\ddot{\mathbf{r}} = -\nabla U$, wobei $U(x, y) = 4\left(\left(\frac{2}{r}\right)^{12} - \left(\frac{1}{r}\right)^{8}\right)$.

Beispiel Streuung am Lennard-Jones Potential

Beispiel Streuung am Lennard-Jones Potential

Die Teilchenstreuung am Lennard-Jones Potential $U(x, y)$ ist beschrieben durch $\ddot{\mathbf{r}} = -\nabla U$, wobei $U(x, y) = 4\left(\left(\frac{2}{r}\right)^{12} - \left(\frac{1}{r}\right)^{8}\right)$.

Bestimme die Teilchentrajektorien für $\mathbf{r}_0 = (-10, b)^T$ mit $b \in \{0.15, 0.3, \ldots, 3\}$ und $\dot{\mathbf{r}}_0 = (1, 0)^T$. Der Zeitschritt ist $\Delta t = 0.02$ und $T_{tot} = 15$.

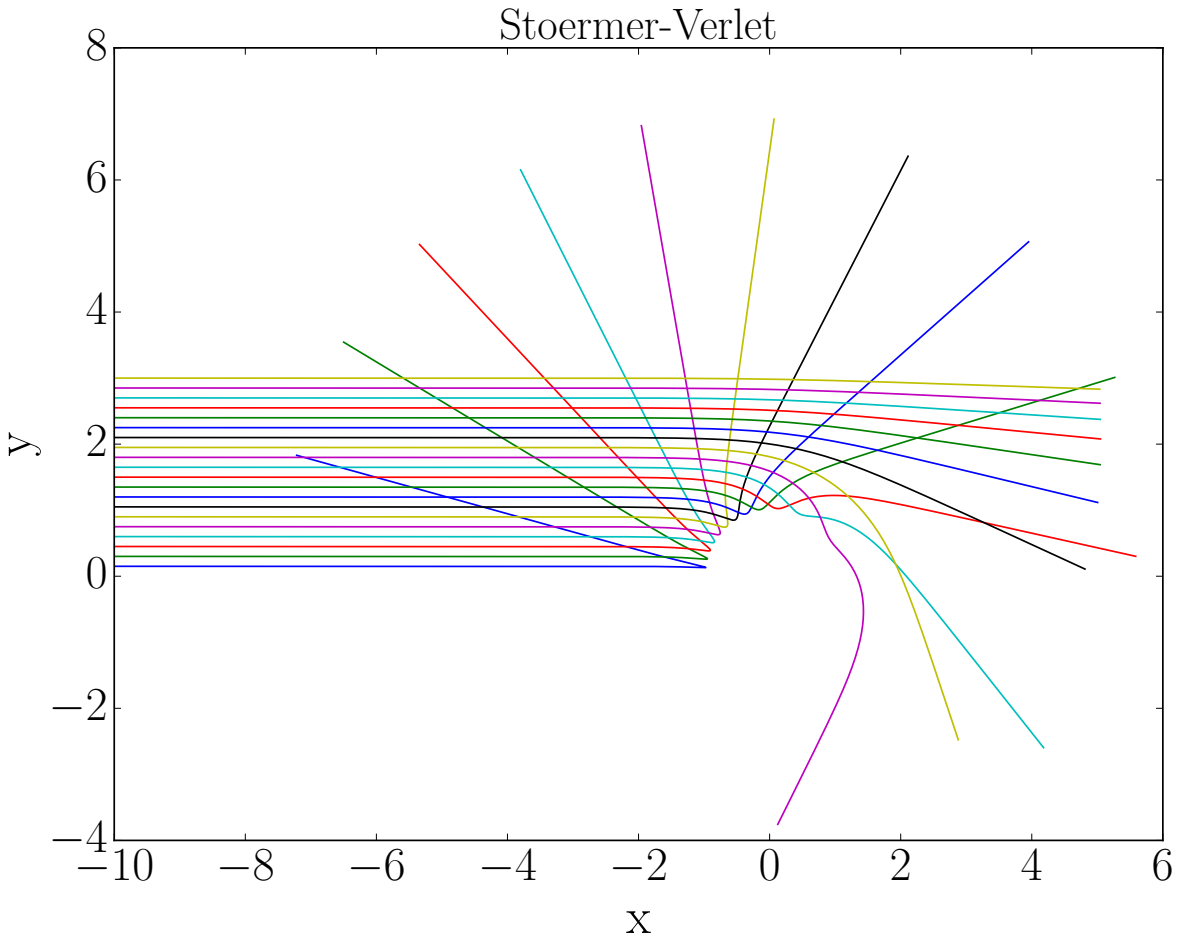
Beispiel Streuung am Lennard-Jones Potential

```
f = lambda x: 24*(2./sqrt(x[0]**2+x[1]**2)**14
              -1./sqrt(x[0]**2+x[1]**2)**8)*x

def stoermer_verlet(x0, v0, dt, T):
    x_sv = []
    x_sv.append(x0)
    x1 = x0 + dt*v0 + 0.5*dt**2*f(x0)
    x_sv.append(x1)
    for i in range(int(T/dt)):
        x_sv.append( -x_sv[-2]+2*x_sv[-1]
                     +dt**2*f(x_sv[-1]) )

    return asarray(x_sv)
```

Beispiel Streuung am Lennard-Jones Potential



Runge-Kutta-Verfahren

Lernziele: Integration von Differentialgleichungen der Form $\dot{\mathbf{y}} = \mathbf{f}(t, \mathbf{y})$ mit Verfahren höherer Konvergenzordnung.

Methoden: Runge-Kutta zweiter und vierter Ordnung, Kollokationsverfahren.

Runge-Kutta-Verfahren

Runge-Kutta-Verfahren

Grundidee: Man nutzt eine Taylorentwicklung höherer Ordnung, um eine bessere Approximation der Differentialgleichung zu bekommen.

Runge-Kutta-Verfahren

Runge-Kutta-Verfahren

Grundidee: Man nutzt eine Taylorentwicklung höherer Ordnung, um eine bessere Approximation der Differentialgleichung zu bekommen.

Man behält alle Terme bis zur Ordnung h^s:

$$y_{n+1} = y_n + \frac{h}{1!}\frac{dy}{dt} + \frac{h^2}{2!}\frac{d^2y}{dt^2} + \cdots + \frac{h^s}{s!}\frac{d^qy}{dt^s} + \mathcal{O}\left(h^{s+1}\right).$$

Runge-Kutta-Verfahren

Grundidee: Man nutzt eine Taylorentwicklung höherer Ordnung, um eine bessere Approximation der Differentialgleichung zu bekommen.

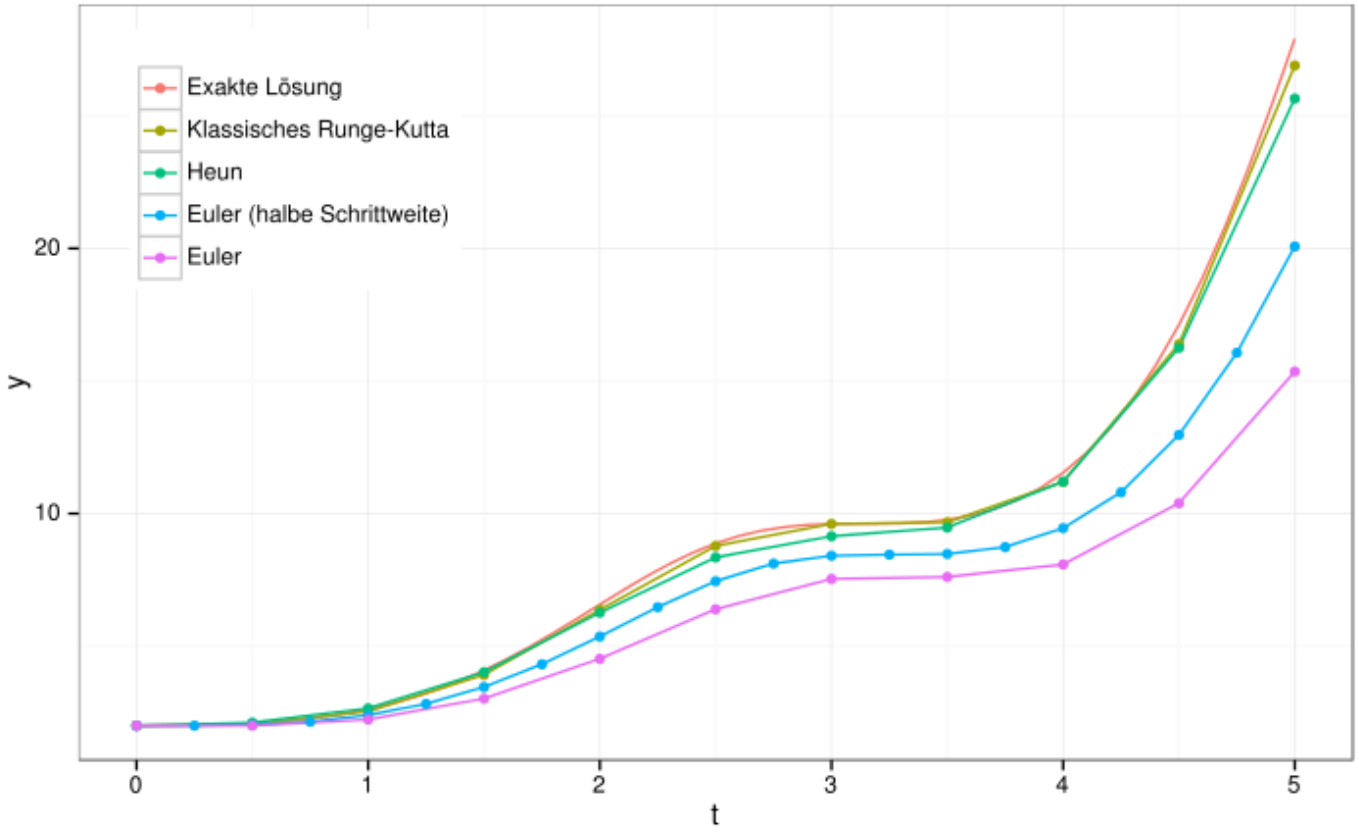
Man behält alle Terme bis zur Ordnung h^s:

$$y_{n+1} = y_n + \frac{h}{1!}\frac{dy}{dt} + \frac{h^2}{2!}\frac{d^2y}{dt^2} + \cdots + \frac{h^s}{s!}\frac{d^sy}{dt^s} + \mathcal{O}\left(h^{s+1}\right).$$

Allgemein schreibt man die Approximation als $y_{n+1} = y_n + h\sum_{i=1}^{s} b_i k_i$, wobei $k_i = f\left(t_n + c_i h, y_n + \sum_{j=1}^{s} a_{ij} k_j\right)$.

$$\begin{array}{c|ccc} c_1 & a_{11} & \dots & a_{1s} \\ \vdots & \vdots & & \vdots \\ c_s & a_{s1} & \dots & a_{ss} \\ \hline & b_1 & \dots & b_s \end{array}$$

Runge-Kutta-Verfahren



https://commons.wikimedia.org/wiki/File:Runge-kutta.svg

Beispiel Runge-Kutta-Verfahren 4. Ordnung

Beispiel Runge-Kutta-Verfahren 4. Ordnung

Butcher Tableau RK4

$$\begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array}$$

1. Definiere die 4 Koeffizienten:

$$k_1 = f(t_n, y_n)$$
$$k_2 = f(t_n + 0.5h, y_n + 0.5hk_1)$$
$$k_3 = f(t_n + 0.5h, y_n + 0.5hk_2)$$
$$k_4 = f(t_n + h, y_n + hk_3)$$

2. Berechne den nächsten Schritt:

$$y_{n+1} = y_n + h\left(\frac{1}{6}k_1 + \frac{1}{3}k_2 + \frac{1}{3}k_3 + \frac{1}{6}k_4\right).$$

Implementiere das RK Verfahren, um die Airy Gleichung $\ddot{u}(t) - tu(t) = 0$ für $u(0) = 0.35503$ und $\dot{u}(0) = 0.25882$ im Zeitintervall $[-40, 0]$ zu lösen.

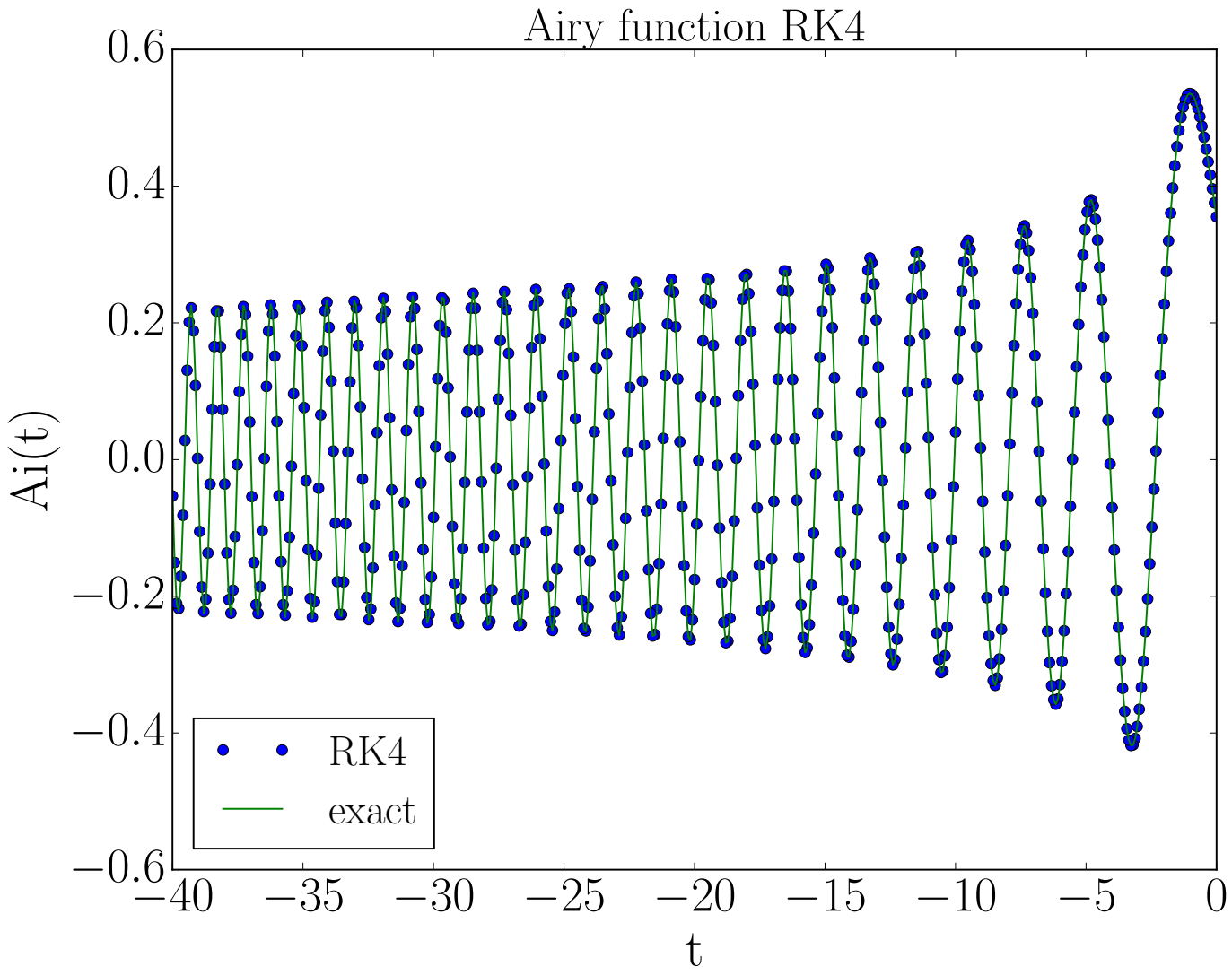
Beispiel Runge-Kutta-Verfahren 4. Ordnung

```
f = lambda t,y: array([y[1], t*y[0]])

def RK4(t0, y0, N, T):
    y_RK4 = []
    y_RK4.append(y0)
    t_RK4, h = linspace(t0, T, N+1, retstep=True)

    for i in range(N):
        k1 = f(t_RK4[i], y_RK4[-1])
        k2 = f(t_RK4[i]+0.5*h, y_RK4[-1]+0.5*h*k1)
        k3 = f(t_RK4[i]+0.5*h, y_RK4[-1]+0.5*h*k2)
        k4 = f(t_RK4[i]+h, y_RK4[-1]+h*k3)
        y_RK4.append(y_RK4[-1]+h*(1./6*k1+
                        1./3*k2+1./3*k3+1./6*k4))

    return t_RK4, asarray(y_RK4)
```


Airy function RK4
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
Ai(t)
−40
−35
−30
−25
−20
−15
−10
−5
0
t
RK4
exact

Runge-Kutta-Verfahren und Kollokation

Grundidee: Kollokationsverfahren approximieren eine Differential, sodass Anfangsbedingung und die Differentialgleichung an Kollokationspunkten erfüllt ist

Runge-Kutta-Verfahren und Kollokation

Runge-Kutta-Verfahren und Kollokation

Grundidee: Kollokationsverfahren approximieren eine Differential, sodass Anfangsbedingung und die Differentialgleichung an Kollokationspunkten erfüllt ist

Kollokationsmethoden erzeugen implizize Runge-Kutta-Verfahren.

Für weitere Informationen:
`https://lp.uni-goettingen.de/get/text/1983`

Beispiel Gausskollokation 4. Ordnung

Butcher Tableau

$$\begin{array}{c|cc} \frac{1}{2} - \frac{\sqrt{3}}{6} & \frac{1}{4} & \frac{1}{4} - \frac{\sqrt{3}}{6} \\ \frac{1}{2} + \frac{\sqrt{3}}{6} & \frac{1}{4} + \frac{\sqrt{3}}{6} & \frac{1}{4} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

Gausskollokationsmethoden haben die Ordnung $p = 2s$, also für das nebenstehende Verfahren findet man $p = 4$.

Implementiere ein Gausskollokationsverfahren 4. Ordnung, um die Airy Gleichung $\ddot{u}(t) - tu(t) = 0$ für $u(0) = 0.35503$ und $\dot{u}(0) = 0.25882$ im Zeitintervall $[-40, 0]$ zu lösen.

Steife Differentialgleichungen

Lernziele: Integration von steifen Differentialgleichungen, was für manche numerische Verfahren nur bei sehr kleiner Schrittweite möglich ist.

Methoden: Beispiel explizites und implizites Eulerverfahren, Stabilitätsgebiet von Runge-Kutta-Verfahren, Radau-Verfahren und Rosenbrock-Wanner-Verfahren.

Beispiel Dahlquist'sche Testgleichung

Beispiel Dahlquist'sche Testgleichung

Wir betrachten das Beispiel $\dot{y} = \lambda y$ mit $y(0) = 1$ und $\lambda < 0$.

Beispiel Dahlquist'sche Testgleichung

Beispiel Dahlquist'sche Testgleichung

Wir betrachten das Beispiel $\dot{y} = \lambda y$ mit $y(0) = 1$ und $\lambda < 0$.

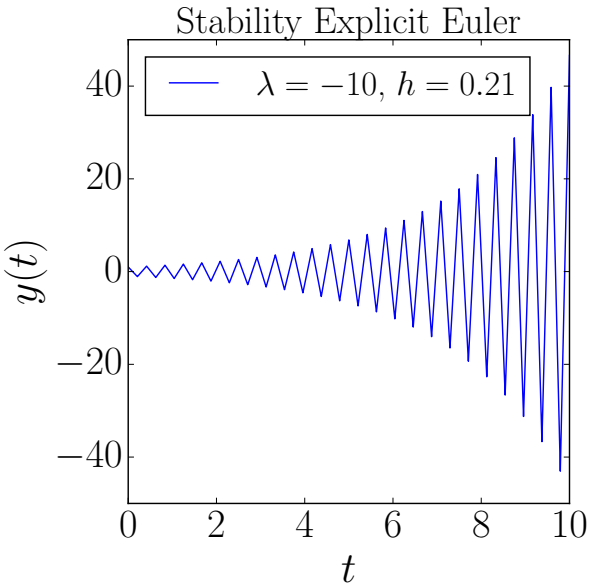
Das explizite Eulerverfahren gibt $y_k = y_{k-1} + \lambda h y_{k-1} = (1 - |\lambda| h)^k y_0$, also $|1 - |\lambda| h| < 1$ und damit $0 < h < \frac{2}{|\lambda|}$.

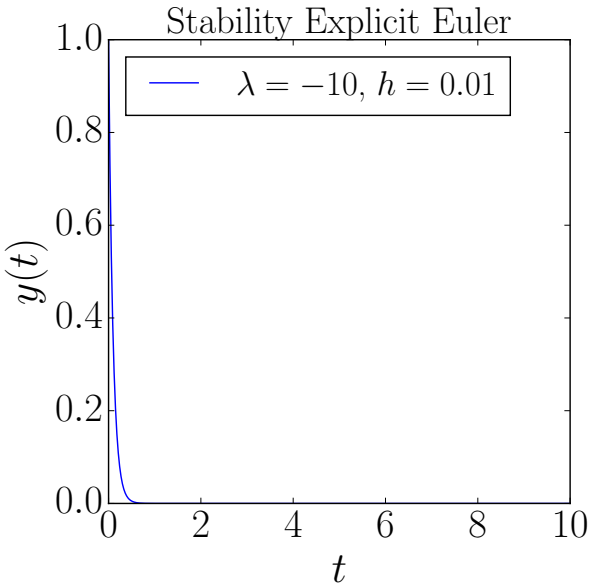
Beispiel Dahlquist'sche Testgleichung

Beispiel Dahlquist'sche Testgleichung

Wir betrachten das Beispiel $\dot{y} = \lambda y$ mit $y(0) = 1$ und $\lambda < 0$.

Das explizite Eulerverfahren gibt $y_k = y_{k-1} + \lambda h y_{k-1} = (1 - |\lambda| h)^k y_0$, also $|1 - |\lambda| h| < 1$ und damit $0 < h < \frac{2}{|\lambda|}$.
Für zu grosse Schrittweiten $h > \frac{2}{|\lambda|}$ wird das explizite Eulerverfahren instabil. Das passiert bei impliziten Methoden nicht.





Stabilitätsfunktionen für Runge-Kutta-Verfahren

Stabilitätsfunktionen für Runge-Kutta-Verfahren

Für ein Butcher-Tableau von Stufe s

$$\begin{array}{c|c} \mathbf{c} & \mathbf{U} \\ \hline & \mathbf{b}^T \end{array}$$

und die Differentialgleichung $\dot{y} = \lambda y$ mit $\lambda \in \mathbb{C}$ ist die Stabilitätsfunktion $S(z) = 1 + z\mathbf{b}^T (\mathbb{1} - z\mathbf{U})^{-1} \mathbf{1} = \frac{\det\left(\mathbb{1} - z\mathbf{U} + z\mathbb{1}\mathbf{b}^T\right)}{\det(\mathbb{1} - z\mathbf{U})}$, wobei $z := \lambda h$ und $\mathbf{1} = (1, \ldots, 1)^T \in \mathbb{R}^s$, cf. p. 280 im Skript.

Stabilitätsfunktionen für Runge-Kutta-Verfahren

Stabilitätsfunktionen für Runge-Kutta-Verfahren

Für ein Butcher-Tableau von Stufe s

$$\begin{array}{c|c} \mathbf{c} & \mathbf{U} \\ \hline & \mathbf{b}^T \end{array}$$

und die Differentialgleichung $\dot{y} = \lambda y$ mit $\lambda \in \mathbb{C}$ ist die Stabilitätsfunktion $S(z) = 1 + z\mathbf{b}^T (\mathbb{1} - z\mathbf{U})^{-1} \mathbf{1} = \frac{\det\left(\mathbb{1} - z\mathbf{U} + z\mathbb{1}\mathbf{b}^T\right)}{\det(\mathbb{1} - z\mathbf{U})}$, wobei $z := \lambda h$ und $\mathbf{1} = (1, \ldots, 1)^T \in \mathbb{R}^s$, cf. p. 280 im Skript.

Zum Beispiel ist für das explizite Eulerverfahren mit $\mathbf{c} = 0$, $\mathbf{U} = 0$ und $\mathbf{b}^T = 1$ die Stabilitätsfunktion $S(z) = 1 + z$.

Radau-Verfahren

Radau-Verfahren

Im Gegensatz zur Gausskollokation, welche die maximale Ordnung $p = 2s$ haben, erreichen Radau-Verfahren die Ordnung $p = 2s - 1$. Es handelt sich ebenfalls um ein implizites Runge-Kutta-Verfahren.

Radau-Verfahren

Radau-Verfahren

Im Gegensatz zur Gausskollokation, welche die maximale Ordnung $p = 2s$ haben, erreichen Radau-Verfahren die Ordnung $p = 2s - 1$. Es handelt sich ebenfalls um ein implizites Runge-Kutta-Verfahren.

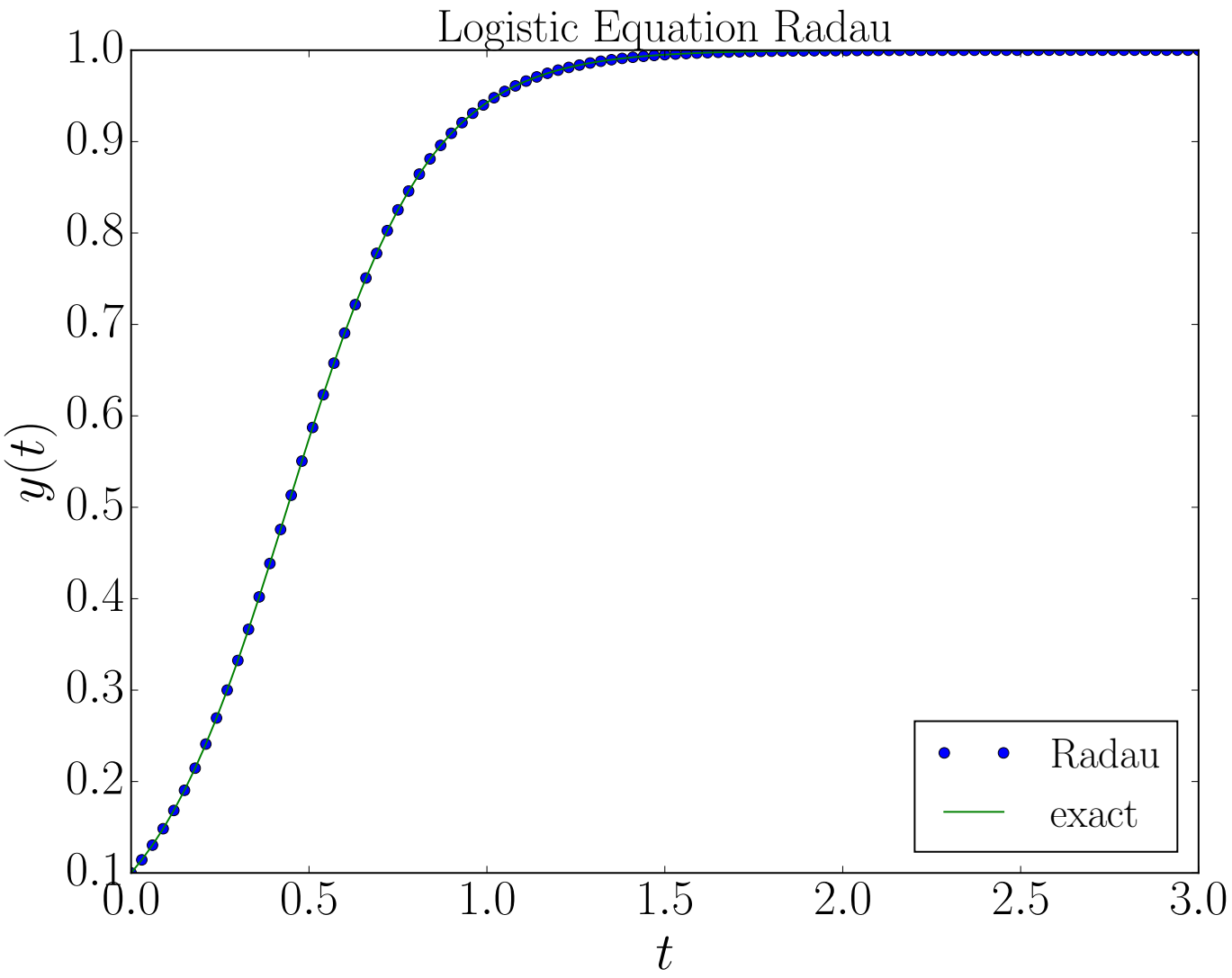
Beispiel Lösung der logistischen Gleichung

Implementiere ein 2-stufiges Radau-Verfahren, um die logistische Gleichung $\dot{y} = \lambda y(1 - y)$ mit $y(0) = 0.1$ und $\lambda = 5$ zu lösen.

Das dazugehörige Butcher-Tableau ist, cf. p. 290 im Skript.

$$\begin{array}{c|cc} \frac{1}{3} & \frac{5}{12} & -\frac{1}{12} \\ 1 & \frac{3}{4} & \frac{1}{4} \\ \hline & \frac{3}{4} & \frac{1}{4} \end{array}$$

Radau-Verfahren



Rosenbrock-Wanner-Verfahren

Rosenbrock-Wanner-Verfahren

Grundidee: Implizite Runge-Kutta-Verfahren eignen sich gut zum Lösen steifer Anfangswertprobleme. Jedoch beinhaltet das die Lösung von Gleichungssystemen in jedem Iterationsschritt.

Rosenbrock-Wanner-Verfahren

Rosenbrock-Wanner-Verfahren

Grundidee: Implizite Runge-Kutta-Verfahren eignen sich gut zum Lösen steifer Anfangswertprobleme. Jedoch beinhaltet dies die Lösung von Gleichungssystemen in jedem Iterationsschritt.

Linear implizite Verfahren, wie das Rosenbrock-Wanner-Verfahren, nutzen approximierte lineare Gleichungssysteme.

Beispiel Lösung der logistischen Gleichung

Implementiere Rosenbrock-Wanner-Verfahren der Ordnung 2 und 3, um die logistische Gleichung $\dot{y} = \lambda y(1-y)$ mit $y(0) = 0.1$ und $\lambda = 5$ zu lösen.

Rosenbrock-Wanner-Verfahren

Rosenbrock-Wanner-Verfahren

Grundidee: Implizite Runge-Kutta-Verfahren eignen sich gut zum Lösen steifer Anfangswertprobleme. Jedoch beinhaltet dies die Lösung von Gleichungssystemen in jedem Iterationsschritt.

Linear implizite Verfahren, wie das Rosenbrock-Wanner-Verfahren, nutzen approximierte lineare Gleichungssysteme.

Rosenbrock-Wanner-Verfahren

Rosenbrock-Wanner-Verfahren

Grundidee: Implizite Runge-Kutta-Verfahren eignen sich gut zum Lösen steifer Anfangswertprobleme. Jedoch beinhaltet dies die Lösung von Gleichungssystemen in jedem Iterationsschritt.

Linear implizite Verfahren, wie das Rosenbrock-Wanner-Verfahren, nutzen approximierte lineare Gleichungssysteme.

Beispiel Lösung der logistischen Gleichung

Implementiere Rosenbrock-Wanner-Verfahren der Ordnung 2 und 3 (p. 293 im Skript), um die logistische Gleichung $\dot{y} = \lambda y(1-y)$ mit $y(0) = 0.1$ und $\lambda = 5$ zu lösen.

Weitere Informationen: `https://www.math.hu-berlin.de/~lamour/Lehre/ODEs/script_NumDGL_Tischendorf.pdf`

Rosenbrock-Wanner-Verfahren