Georg-August-Universität Göttingen
Institut für Theoretische Physik
Prof. Dr. F. Heidrich-Meisner
Dr. E. Bothmann
S. L. Villani
SoSe 2021

Übungen zur Vorlesung
Computergestütztes Wissenschaftliches Rechnen
Blatt 5

Lernziele dieses Übungsblattes

- Runge-Kutta-Verfahren: Anwendung und Herleitung
- Vergleich von Konvergenzordnungen

Aufgabenmodus

Die Aufgaben dieser Woche verfügen über keine gesonderten Tutorials. Sie sollten alle Aufgaben mit Ihren bisherigen Kenntnissen bearbeiten können. Dafür fallen die Aufgabenstellungen stellenweise etwas feinschrittiger aus.

Übungsaufgaben

Aufgabe 13 *Runge-Kutta 2. Ordnung (Pendel)*

In Aufgabe 11 auf dem letzten Übungsblatt haben Sie den expliziten Euler-Algorithmus für beliebige Zustandsvektoren `y[]` in der Funktion `euler_step` umgesetzt, der für das Pendelproblem in Sachen Performance und/oder Genauigkeit sehr schlecht abschneidet. Sie sollen daher im folgenden den dem Euler-Verfahren überlegenen zweistufigen Runge-Kutta-Algorithmus (RK2) implementieren:

$$\begin{aligned}\mathbf{k}_1 &= \mathbf{F}\left(t, \mathbf{y}\right) \cdot \Delta t,\\ \mathbf{k}_2 &= \mathbf{F}\left(t + \frac{\Delta t}{2}, \mathbf{y} + \frac{\mathbf{k}_1}{2}\right) \cdot \Delta t,\\ \mathbf{y}_{i+1} &= \mathbf{y}_i + \mathbf{k}_2.\end{aligned}$$

Fett-gesetzte Symbole stehen für im allgemeinen vektorielle Größen. Wie auf dem letzten Blatt verwenden wir die Notation $\mathbf{F} = \frac{\mathrm{d}\mathbf{y}}{\mathrm{d}t}$ (in den Vorlesungsnotizen wird stattdessen ein kleines $\mathbf{f}$ verwendet). Das Verfahren trägt in der Literatur oft den Namen “Mittelpunktsverfahren” oder “Modifiziertes Euler-Verfahren”.

1. Fügen Sie Ihrer Numerik-Bibliothek die neue Funktion

```
void rk2_step(double t, double delta_t, double y[],
              ode_func func, int dimension, void *params)
{
    ...
}
```

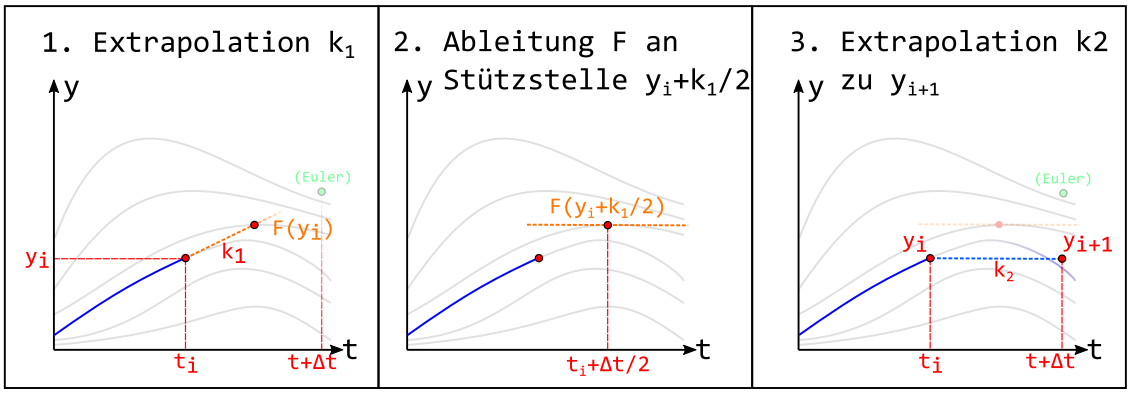

Abbildung 1: Illustration zum RK2-Verfahren.

hinzu. `t` ist die aktuelle Zeit, `delta_t` ist die Schrittweite des durchzuführenden Zeitschritts. `y[]` ist der aktuelle Zustandsvektor des Systems und hat die Größe `dimension`. Die Funktion `func` ist die Systemfunktion, die die Ableitungen der Systemgrößen berechnet. Der Aufruf

```
func(t, a, b, params);
```

berechnet die Funktion $\mathbf{F}\,(t, \mathbf{a})$ und speichert das Ergebnis im Array `b[]` ab.

2. Legen Sie in `rk2_step` zunächst drei `double`-Felder `support[]`, `k1[]` und `k2[]` der jeweiligen Länge `dimension` an. Benutzen Sie den Heap (also einen mit `malloc` allozierten Speicherbereich), damit die `rk2_step`-Funktion auch für große Systemdimensionen funktioniert.

3. Berechnen Sie mit `func` den Inhalt von `k1[]`. Vergessen Sie nicht die Multiplikation mit Δt.

4. Berechnen Sie die Position der nächsten Stützstelle und schreiben Sie das Ergebnis in `support[]`. Mit "Stützstelle" ist der Zustand (und die Zeit) am nächsten Auswertungspunkt gemeint:

$$\left(t + \frac{\Delta t}{2}, \mathbf{y} + \frac{\mathbf{k}_1}{2}\right).$$

Hinweis: Da der Inhalt von `k1` in diesem speziellen Verfahren nicht mehr benötigt wird, könnten Sie Speicher sparen und die Stützstelle stattdessen in das Feld `k1` schreiben. Bei den meisten Runge-Kutta-Verfahren ist dies aber nicht möglich. Daher ist es empfehlenswert, aus Gründen der Übersichtlichkeit auch hier das Array `support` als Zwischenspeicher zu verwenden.

5. Berechnen Sie nun `k2[]` und aktualisieren Sie dann den Zustandsvektor `y[]` auf den neuen Zeitpunkt. Geben Sie unbedingt den durch die drei Arrays `support[]`, `k1[]` und `k2[]` belegten Speicher wieder frei!

6. Ersetzen Sie in Ihrer Lösung von Aufgabe 11 `euler_step` durch `rk2_step`. Simulieren Sie das System und plotten Sie die Positionen der Pendel und die Gesamtenergie des Systems.

Aufgabe 14 *Konvergenzverhalten von RK2*

Bevor Sie einen numerischen Algorithmus zur Untersuchung von physikalischen Problemen einsetzen, ist es wichtig, zunächst seine korrekte Implementierung zu überprüfen. Das funktioniert am Besten an Referenzsystemen, bei denen Sie die gesuchte Lösung bereits kennen. Auch in der Prüfung wird es notwendig sein, dass Sie Ihre Umsetzung auf Fehler untersuchen. In dieser Aufgabe werden Sie die Konvergenzordnung von Ihrem `rk2_step`-Integrator überprüfen.

1. Bereiten Sie eine Pendelsimulation für ein Einzelpendel $N = 1$ vor. Die Federhärte betrage $k = 100\,\mathrm{N/m}$, die Masse $m = 1\,\mathrm{kg}$. Die Anfangsauslenkung sei $0\,\mathrm{m}$, die Anfangsgeschwindigkeit $1\,\mathrm{m/s}$.

2. Die analytische Lösung des Problems verrät eine Periodendauer von

$$T = 2 \cdot \pi \cdot \sqrt{\frac{m}{k}} \approx 0{,}6283... \text{ s}.$$

   Berechnen Sie diese Periodendauer mit `double`-Genauigkeit in Ihrem Programm, so dass Sie sie als Variable zur Verfügung stehen haben.

3. Gemäß der analytischen Lösung ist die Position $x(T/4)$ nach einer Viertelperiodendauer gerade

$$x(T/4) = v_0 \cdot \sqrt{\frac{m}{k}} = 0{,}1\,\mathrm{m}.$$

   Berechnen Sie auch hierfür eine Variable.

4. Entwerfen Sie eine Simulationsschleife, die das System für exakt $T/4$ simuliert. Weil der Zeitschritt Δt im Allgemeinen kein ganzzahliger Teiler von $T/4$ ist, müssen Sie in Ihrer Schleife die Schrittweite modifizieren:

   Der letzte Schritt der Simulation soll nicht die Länge Δt betragen, sondern einen Bruchteil davon, so dass das System exakt bei $T/4$ endet. Alle anderen Schritte werden regulär mit genau Δt durchgeführt.

5. Simulieren Sie das Pendel mit dem RK2-Verfahren 100 mal mit logarithmisch verteilten Δt im Bereich von 10^{-8} bis 10^{0}. Geben Sie jeweils den verwendeten Zeitschritt und den Betrag der Differenz ("Residuen") zwischen analytischer und numerischer Endposition des Pendels in eine Datei aus.

   Hinweis: Für sehr kleine Δt wird das Problem sehr schnell ziemlich rechenaufwendig. Passen Sie die Kompilier- und Linkbefehle in der `makefile` an und führen Sie alle `gcc`-Befehle mit Optimierung aus:

   ```
   gcc -Wall -Ofast -flto ...
   ```

6. Tragen Sie die Residuen doppellogarithmisch gegen die jeweilige Schrittweite auf. Bestimmen Sie aus der Steigung die Konvergenzordnung des Algorithmus. Wenn die Ordnung schlechter ist als Sie erwarten, ist die Implementierung fehlerhaft. Finden Sie in diesem Fall Ihre Fehler und korrigieren Sie sie.

7. **Optional:** Untersuchen Sie auch das Euler-Verfahren und vergleichen Sie beide Integratoren in einem gemeinsamen Plot.

Aufgabe 15 *Herleitung von RK4*

In dieser Aufgabe leiten wir den Runge-Kutta-Algorithmus 4. Ordnung her. Dabei folgen wir der Herleitung für die 2. Ordnung aus der Vorlesung. Hier hatten wir den Ansatz $y_{i+1} = y_i + \sum_{i=1}^{2} w_i k_i$ mit $k_1 = \Delta x f(y_i, x_i)$ und $k_2 = \Delta x f(y_i + \alpha k_1, x_i + \alpha \Delta x)$. Für die 4. Ordnung schreiben wir nun den Ansatz

$$\begin{aligned} y_{i+1} &= y_i + \sum_{i=1}^{4} w_i k_i, \\ k_1 &= \Delta x f(y_i, x_i), \\ k_2 &= \Delta x f(y_i + \alpha_1 k_1, x_i + \alpha_1 \Delta x), \\ k_3 &= \Delta x f(y_i + \alpha_2 k_2, x_i + \alpha_2 \Delta x), \\ k_4 &= \Delta x f(y_i + \alpha_3 k_3, x_i + \alpha_3 \Delta x). \end{aligned} \tag{1}$$

1. Für die 2. Ordnung haben wir in der Vorlesung (mit $\alpha \equiv \alpha_1$) bereits bestimmt, dass

   $$\begin{aligned} k_1 &= \Delta x f, \\ k_2 &= \Delta x f + \Delta x^2 \alpha_1 f', \end{aligned}$$

   mit $f \equiv f(y_i, x_i)$ und $f' \equiv \frac{\mathrm{d}f}{\mathrm{d}x}$. Zeigen Sie nun auch mit Hilfe der Entwicklung von f in Δx die entsprechenden Ausdrücke für k_3 und k_4:

   $$\begin{aligned} k_3 &= \Delta x f + \Delta x^2 \alpha_2 f' + \Delta x^3 \alpha_2 \alpha_1 f'', \\ k_4 &= \Delta x f + \Delta x^2 \alpha_3 f' + \Delta x^3 \alpha_3 \alpha_2 f'' + \Delta x^4 \alpha_3 \alpha_2 \alpha_1 f'''. \end{aligned}$$

2. Taylorn Sie $y(x_i + \Delta x) - y(x_i)$ um $\Delta x = 0$ bis zur 4. Ordnung. Verwenden Sie $y' = f$, um einen Koeffizientenvergleich mit dem Ansatz (1) durchzuführen, wobei Sie die k_i aus Aufgabenteil 1 einsetzen. Sie erhalten 4 Gleichungen für 7 Unbekannte (w_1, w_2, w_3, w_4, α_1, α_2, α_3).

3. Wählen Sie nun $\alpha_1 = \alpha_2 = \frac{1}{2}$ und $\alpha_3 = 1$, also zwei Stützstellen in der Mitte und eine Stützstelle am Ende des Intervalls $[x_i, x_i + \Delta x]$. Zeigen Sie, dass dann $w_1 = w_4 = \frac{1}{6}$ und $w_2 = w_3 = \frac{1}{3}$.

4. Implementieren Sie `rk4_step` und wiederholen Sie damit Aufgabenteil 7 aus Aufgabe 14, um `rk4_step` mit `rk2_step` (und ggf. auch dem Euler-Verfahren) in einem gemeinsamen Plot zu vergleichen.

Selbsttest

- Ist RK2 genauer als Euler? Falls ja, warum?
- Ist RK4 genauer als RK2? Falls ja, warum?
- RK2 erzeugt pro Zeitschritt mindestens den doppelten Rechenaufwand gegenüber Euler. Welchen Einfluss hat das auf Ihre Wahl des Integrators für ein Problem?