Praktikum zur Numerischen Mathematik im Sommersemester 2021
Prof. Dr. Nils Rosehr, Hochschule München, 12. Mai 2021

Praktikum 3: Differentialgleichungen

Implementierung einer Bibliothek

Schreiben Sie eine Bibliothek `lib3` mit folgenden Funktionen:

- `Euler(f, a, xa, b, n)`
- `Mittelpunktsregel(f, a, xa, b, n)`
- `Heun(f, a, xa, b, n)`
- `Runge_Kutta(f, a, xa, b, n)`

Dabei sei f eine Funktion wie in der Vorlesung, die eine gewöhnliche explizite Differentialgleichung erster Ordnung definiert, und x eine Lösung des zugehörigen Anfangswertproblems mit $x(a) = x_a$ (in Python benutzen wir hierfür die Variable `xa`). Durch obige Funktionen sollen die entsprechenden Lösungsverfahren der Vorlesung implementiert werden – denken Sie an DRY. Der Rückgabewert aller Funktionen soll eine Liste der Länge $n + 1$ sein, bestehend aus den Lösungsiterationen $x_0, x_1, \ldots, x_n$ zur Approximation von $x(b)$. Hier ist es zweckmäßig, mit einer Liste `[xa]` zu starten und diese mit dem `append`-Befehl schrittweise zu erweitern.

Untersuchungen zur Konvergenz

- Definieren Sie in `main3.py` Funkionen `f1(t, x)` bis `f4(t, x)`, die die Differentialgleichungen aus Aufgabe 4.1 beschreiben.
- Definieren Sie ferner Funktionen `awp1(t, a, xa)` bis `awp4(t, a, xa)`, die die exakten Lösungen der Anfangswertprobleme mit $x(a) = x_a$ angeben. Sie können dann mit der Funktion `Lösung_AWP(awp, a, xa)` aus `vorgabe3.py` des Archivs `P3vorgabe.zip` von `moodle` eine Lösung des entsprechenden Anfangswertproblems erzeugen. (Der Rückgabewert von `Lösung_AWP` ist eine Funktion, die nur von `t` abhängt und sich außerdem gut mit `numpy` verträgt, siehe Code in `vorgabe3.py`.)
- Zur vollständigen Initialisierung der Verfahren zur Lösung des Anfangswertproblems sind die Daten a, $x_a =$ `xa` und b (Endzeitpunkt) sowie die Anzahl n der Iterationen nötig. Beispiele für diese Daten sind für jedes der vier Anfangswertproblem in Form der Listen `data[1]` bis `data[4]` bereits in `vorgabe3.py` definiert. Importieren Sie diese in `main3.py` durch `from vorgabe3 import data`. Lediglich der Parameter n muss jeweils in `main3.py` noch angepasst werden, beispielsweise durch `data[1][3] = 100`.
- Definieren Sie nun in `lib3` eine Funktion `plot(f, awp, datai)`, die mit obigen Daten das Richtungsfeld, die exakte Lösung und die vier numerischen Lösungen in ein Bild plottet, wobei `datai` die Form `[a, xa, b, n]` hat. Sie können dabei die vordefinierten Funktionen `Lösung_AWP(awp, a, xa)` und `plot_Richtungsfeld(f)` benutzen. Es ist sinnvoll zunächst die Funktionen zu plotten und dann `plot_Richtungsfeld(f)` aufzurufen, damit schon die Ausmaße des Plotbereichs bekannt sind. Rufen Sie in `main3.py` Ihre `plot`-Funktion für alle vier Anfangswertprobleme mit jeweils **aussagekräftigen** Werten für n auf, so dass man das unterschiedliche Konvergenzverhalten der Verfahren erkennen kann. Man sollte jetzt vier Bilder mit jeweils einem Funktionsgraphen, vier Polygonzügen und einem Richtungsfeld sehen.
- Ergänzen Sie schließlich `plot` so, dass für jedes der vier Anfangswertprobleme ein weiteres Bild geplottet wird, in dem der absolute Fehler (Differenz des Näherungswertes zum Endzeitpunkt b und des exakten Wertes $x(b)$) für alle vier Verfahren als Graph in Abhängigkeit von der Anzahl der Iterationen k für $k = n, n + 1, \ldots, 4n$ (für Ihr oben gewähltes n) geplottet wird. Möglicherweise müssen Sie hier Ihr oben gewähltes n noch einmal anpassen, um aussagekräftige Bilder zu erhalten. Es sollten nun insgesamt acht Bilder entstehen.

Systeme von Differentialgleichungen

Wie in der Vorlesung beschrieben lässt sich ein System

$$x_0' = f_0(t, x_0, x_1),$$
$$x_1' = f_1(t, x_0, x_1).$$

von Differentialgleichungen einfacher beschreiben, wenn man die Koordinatenfunktionen entsprechend $x = (x_0, x_1)$ und $f = (f_0, f_1)$ zu Vektoren zusammenfasst. Die Differentialgleichung geht über in die einfachere Form

$$x' = f(t, x).$$

Das bedeutet auch, dass wir ein System von Differentialgleichungen mit den bereits implementierten Funktionen lösen können. Wir müssen nur 2-Tupel von reellen Parametern übergeben. Dafür eignen sich am besten `NumPy`-Arrays; siehe „NumPy und Matplotlib“ bei `moodle`. Beispielsweise übergibt man den Anfangswert $x_a = (0, 1)$ als `np.array([0, 1])`. Siehe auch Beispiel 4.5.1 der Vorlesung.

- Es soll nun das Differentialgleichungssystem

$$x_0' = -cx_0 - x_1, \quad x_1' = x_0 - cx_1 \qquad \text{mit } c = \frac{\ln(2)}{2\pi} \text{ sowie Anfangswert } a = 0 \text{ und } x_a = (4, 0)$$

  gelöst werden.

- Stellen Sie hierfür die zweidimensionale Systemfunktion $f = (f_0, f_1)$ auf, und berechnen Sie die Lösung mit `Runge_Kutta(f, a, xa, b, n)`. Wenn Sie beim Programmieren vorsichtig waren, sollte das Verfahren ohne Änderungen auch im 2-dimensionalen Fall funktionieren. Insbesondere sollte der Operator += im Zusammenhang mit dem `append`-Befehl vermieden werden; siehe Einführung zu `NumPy`.

- Schreiben Sie nun eine Funktion `plot_2D(f, datai, solver)` (in `lib3`), die zwei Diagramme für das Lösungsverfahren `solver` zeichnet. Eines mit den beiden Graphen für die Koordinatenfunktionen x_0 und x_1 der Lösung in Abhängigkeit von t und ein zweites mit der Ortskurve x. Sie können auch die vordefinierte Funktion `plot_Richtungsfeld_2D(f)` aus `vorgabe3.py` aufrufen. (Dies ist nicht zu verwechseln mit dem Richtungsfeld, wie wir es in der Vorlesung für den eindimensionalen Fall definiert haben. In diesem Beispiel lässt sich auch der zweidimensionale Fall darstellen, weil die Differentialgleichung nicht zeitabhängig ist.)

- Welche geometrische Figur entsteht für $b = 8\pi$? Raten Sie die exakten Koordinaten $(x_0(b), x_1(b))$ des Endpunktes der Lösungskurve anhand der Figur, und speichern Sie diese als Tupel in die Variable `Endpunkt`.

Anwendung

Wie in der Vorlesung beschrieben lässt sich eine Differentialgleichung 2. Ordnung in ein System von zwei Differentialgleichungen überführen. Als Anwendung wollen wir die Periode des sogenannten mathematischen Pendels berechnen. Dabei handelt es sich um eine Masse in einem homogenen Gravitationsfeld, die an einer starren masselosen Verbindung um einen Punkt rotiert wie etwa bei einer Pendeluhr. Der Auslenkungswinkel der Masse ist gegeben durch folgende Differentialgleichung zweiter Ordnung:

$$x'' = -\sin(x)$$

Als System ergibt sich:

$$x_1' = -\sin(x_0), \quad x_0' = x_1$$

Lösen Sie numerisch die drei Anfangswertprobleme mit `Runge_Kutta` für x_a gleich $5°$, $30°$, und $177°$ jeweils für den Startzeitpunkt $a = 0$ und $x_a' = 0$. (Dabei ist $x°$ eine Abkürzung für das einheitenlose Bogenmaß $x° = x\frac{2\pi}{360}$, also etwa $90° = \frac{\pi}{2}$.) Plotten Sie die Lösungen mit `plot_2D`, und bestimmen Sie näherungsweise (durch Probieren oder eleganter durch Nullstellenbestimmung mit dem Sekantenverfahren) mit einem relativen Fehler von höchstens 1% die Periode der Schwingung (im Bogenmaß) in diesen drei Fällen. Weisen Sie Ihre drei gefundenen Werte der Variablen `Perioden` in `main3.py` als 3-Tupel zu.

Abgabe

- Laden Sie das Archiv `P3vorgabe.zip` von `moodle` herunter, entpacken Sie es, und testen Sie Ihre Programme, indem Sie `test3.py` im gleichen Verzeichnis mit `python` ausführen. Erhalten Sie `ERROR`, so entspricht Ihr Programm nicht der Spezifikation von oben. Erhalten Sie `FAIL`, so ist Ihr Programm zwar lauffähig, aber die berechneten Werte sind fehlerhaft.

- **Abgaben, bei denen der Test gar nicht durchläuft oder mit `ERROR`, sind ungültig**, `FAIL` führt nur zu Punktabzug.

- **Bitte füllen Sie die Datei** `info3.md` **aus, und vergessen Sie nicht die Quellenangabe.** Abgaben mit unvollständiger Datei `info3.md` können nicht gewertet werden.

- Komprimieren und bündeln Sie alle oben erzeugten oder geänderten Dateien, indem Sie ein ZIP-Archiv erstellen. Sollten Sie nicht wissen, wie das geht, konsultieren Sie dazu die Dokumentation Ihres Betriebssystems.

- Benennen Sie Ihr ZIP-Archiv `P3.zip`.

- Schreiben Sie eine Email an `rosehr@hm.edu` mit Betreff `Numerik Abgabe`, Dateianhang `P3.zip` und beliebigem sonstigen Inhalt. **Wichtig ist, dass Betreff und Dateiname des Anhangs stimmen!**

Praktikumstermine P3: keine; **Abgabetermin:** 01.06.