Übungsblatt 6

Lotka-Volterra-Gleichungen und Fraktale

Netzwerke und komplexe Systeme

F. Klimm und B.F. Maier

Schülerakademie 5.2 (Roßleben 2016)

Aufgabe 6.1 Numerisches Lösen von Differentialgleichungen

Oft ist es nicht möglich Differentialgleichungen (DGL) analytisch zu lösen. Dann müssen wir auf numerische Lösungsverfahren zurückgreifen. Eines der einfachsten Verfahren ist das sogenannte *Euler-Verfahren*, auch intuitiv *Methode der kleinen Schritte* genannt.

Sei folgende DGL gegeben

$$\dot{y} = f(t, y). \tag{1}$$

mit dem Anfangswert $y(t_0) = y_0$. Dann berechnen wir iterativ aus dem derzeitigen Zustand (t_k, y_k) den jeweils folgenden Wert y_{k+1} als

$$\dot{y}_{k+1} = y_k + h \cdot f(t_k, y_k). \tag{2}$$

Dabei handelt es sich um eine *lineare* Approximation und h gibt die Schrittweite an. In der Regel resultieren kleinere Schrittweiten h in besseren numerischen Lösungen.

Teilaufgabe 6.1.1 Exponentielles Wachstum

Zunächst wollen wir uns mit einem eindimensionalen Problem beschäftigen und zwar mit dem exponentiellen Wachstum.

Wie lautet die analytische Lösung der folgenden DGL?

$$\dot{x}(t) = \lambda x(t) \tag{3}$$

$$x(t_0) = x_0 \tag{4}$$

Teilaufgabe 6.1.2 Eindimensionales Euler-Verfahren

Löse die obige DGL numerisch mit dem Euler-Verfahren. Setze zunächst $h = 1$, $x_0 = 3$ und $\lambda = 1.2$.

Stelle nun den zeitlichen Verlauf grafisch dar und vergleiche mit der analytischen Lösung. Wie verhält sich das System wenn die Schrittweite stark erhöht wird?
Zusatz: Variiere $\lambda \in \{-2, -0.1, 0.1, 2\}$. Stelle die verschiedenen Kurven $x(t)$ dar, wie verhalten sie sich?

Teilaufgabe 6.1.3 Numerisches überprüfen der Stabilität von Fixpunkten

Bestimme für die obige DGL alle Fixpunkte x^* analytisch. Nun wollen wir die Stabilität überprüfen. Dazu lenken wir die DGL ein wenig von dem Fixpunkt aus und überprüfen numerisch ob sich das System wieder zurück zum Fixpunkt bewegt oder sich davon entfernt. überprüfe dies für $\lambda = \{-1, 0, 1\}$.

Teilaufgabe 6.1.4 Zusatz: Fehlerabschätzung des Euler Verfahren

Da wir die analytische Lösung x_{ana} der obigen DGL kennen können wir überprüfen wie sich die Abweichung (oder auch der Fehler) in Abhängigkeit von der Schrittweite h verhält.

Stelle hierfür $\text{Fehler}(h) = |x_{\text{ana}}(t) - x_{\text{numerisch}}(t, h)|$ dar. Wähle zum Beispiel $t = 100$ und variiere die Schrittweite h. Wie sieht der Zusammenhang $\text{Fehler}(h)$ aus?

Teilaufgabe 6.1.5 Numerische Lösung der Lotka-Volterra Gleichungen

Wir können das Euler-Verfahren auch auf zweidimensionale DGLs anwenden. Hierzu wählen wir als Beispiel die *Lotka-Volterra* Gleichungen die wir bereits im Unterricht analytisch besprochen haben.

Dieses Räuber-Beute System wird wie folgt beschrieben:

$$\dot{x}(t) = x(3 - x - 2y) \tag{5}$$
$$\dot{y}(t) = y(2 - x - y)\,. \tag{6}$$

Das Euler-Verfahren funktioniert ähnlich wie im eindimensionalen Fall:

$$x_{k+1} = x_k + \dot{x}(t)_k \tag{7}$$
$$y_{k+1} = y_k + \dot{y}(t)_k\,. \tag{8}$$

Nutze das Euler-Verfahren um das Lotka-Volterra System zu lösen. Nutze verschiedene Anfangsbedingungen und analysiere zu welchen der analytisch beobachteten Fixpunkten sich das System hin bewegt. Stelle hierfür $x(t)$ und $y(t)$ dar. Welche der analytisch vorhergesagten Fixpunkte werden nicht beobachtet?

Teilaufgabe 6.1.6 Zusatz: Oszillationen im Lotka-Volterra System

Eine leicht veränderte Variante der Lotka-Volterra Gleichungen lautet

$$\dot{x}(t) = x(\alpha - \beta y) \tag{9}$$
$$\dot{y}(t) = y(\gamma - \delta y)\,. \tag{10}$$

Stelle verschiedene *Trajektorien* $(x(t), y(t))$ grafisch dar. Varriiere dafür die Parameter α, β, γ und δ. Wie verhält sich das System wenn du zum Beispiel $\alpha = 2/3$, $\beta = 4/3$, $\gamma = 1$ und $\delta = 1$ und $x_0 = y_0 = 0.9$ verwendest?

Teilaufgabe 6.1.7 Zusatz: *Basins of Attraction* Lotka-Volterra System

Recherchiere was die *Basins of Attraction* sind und ermittle diese numerisch für eine bestimmte Parameter Wahl für die Lotka-Volterra Gleichungen.

Aufgabe 6.2 Fraktale und Selbstähnlichkeit

Saskia und Konrad haben beide die Koch-Kurve eingeführt, Konrad als Lindenmayer-System und Saskia als Fraktal. Wir wollen nun dieses Fraktal für endliche Iterationsschritte n zeichen. Es wird bestimmt durch das Lindenmayer-System

$$\text{Variablen} = \{F\} \tag{11a}$$
$$\text{Konstanten} = \{+, -\} \tag{11b}$$
$$\text{Produktionsregeln} = \{F \to F - F + +F - F\} \tag{11c}$$
$$\text{Axiom} = F. \tag{11d}$$

Hier bedeutet F, dass eine *turtle* (bzw. ein Zeichenstift) eine bestimmte Strecke vorwärts geht, $-$, dass sich die *turtle* um 60° nach links dreht und $+$, dass sich die *turtle* um 60° nach rechts dreht.

Teilaufgabe 6.2.1 Anwenden der Iteration in L-System

Schreibe eine Funktion in Python, die für die Koch-Kurve die Produktionsregeln n mal auf das Axiom anwendet und so ein Wort produziert, mit dem einer *turtle* vermittelt werden kann, wie gezeichnet werden soll.

Teilaufgabe 6.2.2 Teenage Mutant L-System Turtle

Das folgende Programm bietet einen Einstieg in das Zeichnen mit einer *turtle*. [1] Spiele mit den Befehlen im ersten Teil des Codes herum. Was macht die Funktion im letzten Teil des Codes?

[1] Eine ausführliche Dokumentation der *turtle*-Befehle findet ihr auf `https://docs.python.org/2/library/turtle.html`

```python
import turtle
import numpy as np

# setze die Geschwindigkeit des Zeichners auf Maximum
turtle.speed(0)

# Hebe den Stift des Zeichners (sodass bei Bewegung
# nicht gezeichnet wird
turtle.penup()

# Bewege den Zeichner zur neuen Startposition
turtle.setpos(-250,250)

# Setze den Stift nieder, sodass nun gezeichnet wird
turtle.down()

# Bewege den Zeichner 30 pixel vorwaerts (zeichne Linie)
turtle.forward(30)

# Drehe den Zeichner 60 Grad nach links
turtle.left(60)

# Gehe 30 Pixel nach vorne (zeichne dabei), drehe 60 Grad nach
    rechts
turtle.forward(30)
turtle.right(60)

# Bestimme, dass das Zeichenfenster erst bei einem Klick
   geschlossen wird
turtle.exitonclick()

# Was macht diese Funktion?
def draw_mysterious_thingy(n,base_r=100):

    forward_length = 2 * base_r * np.sin(np.pi/n)
    degree = 360./n

    turtle.penup()
    turtle.setpos(0,0)
    turtle.pendown()

    for i in range(n):
        turtle.forward(forward_length)
        turtle.left(degree)
```

Teilaufgabe 6.2.3 Koch-Kurve zeichnen

Schreibe eine Funktion, die das Ergebnis der n-fachen Iteration des L-Systems (11) mithilfe der *turtle* zeichnet. Beachte, dass der Befehl ausgelöst durch F skaliert werden muss mit variierendem n (d.h. dass der forward-Befehl mit $1/3^n$ der ursprünglichen Länge aufgerufen werden muss).

Wie muss das Axiom verändert werden, damit statt der Koch-Kurve die Koch'sche Schneeflocke gezeichnet wird?

Teilaufgabe 6.2.4 Allgemeine L-Systeme

Verallgemeinere deine Funktion aus Aufgabe 6.2.1, sodass beliebige Lindenmayer-Systeme produziert werden, die n-fach iteriert werden.

Aufgabe 6.3 Fraktale erzeugen

Betrachte das L-System

$$\text{Variablen} = \{A, B\} \tag{12a}$$
$$\text{Konstanten} = \{+, -\} \tag{12b}$$
$$\text{Produktionsregeln} = \{A \to +B-A-B+,\ B \to -A+B+A-\} \tag{12c}$$
$$\text{Axiom} = A. \tag{12d}$$

Hier stehen sowohl A als auch B dafür, dass die *turtle* vorwärts läuft. Bei $-$ soll sie sich um 60° nach links drehen, bei $+$ um 60° nach rechts.

Welches Fraktal wird erzeugt?

Teilaufgabe 6.3.1 L-Systeme mit Speicher

Bei Systemen mit Speicher merkt sich die *turtle* ihre aktuelle Position im zweidimensionalen Raum, sobald sie in der Zeichenkette eine Klammer [findet. Sie folgt dann weiter den Befehlen in der Zeichenkette, bis sie auf das Ende des momentanen Befehls trifft, markiert durch]. Dann springt sie auf die Position im Raum zurück, die sie bei [hatte und folgt weiter der Zeichenkette. Betrachte das L-System

$$\text{Variablen} = \{F\} \tag{13a}$$
$$\text{Konstanten} = \{+, -, [,]\} \tag{13b}$$
$$\text{Produktionsregeln} = \{F \to F[+F]F[-F]F\} \tag{13c}$$
$$\text{Axiom} = F. \tag{13d}$$

Der Winkel, den $+, -$ beschreiben, soll diesmal $180°/7$ betragen. Welchem Objekt ähnelt das Ergebnis?