Humboldt-Universität Berlin - Institut für Mathematik
Projektpraktikum I - WS18/19
Dr. Hella Rabus *16. Oktober 2018*

Serie 1, Teil 1

Projektplan

Im Rahmen dieser Serie wird der erste Teil eines übergreifenden Projektes bearbeitet. Abgesehen von Serie 4 werden alle Serien ein Teil dieses Projektes sein. Nachfolgend finden Sie eine grobe Vorschau der Inhalte der projektbezogenen Serien.

- Serie 1: Approximation von Ableitungen mithilfe von finiten Differenzen
- Serie 2: Darstellung von Differentialgleichungen als lineare Gleichungssysteme
- Serie 3: Lösen linearer Gleichungssysteme mit direkten Verfahren
- Serie 4: —
- Serie 5: Lösen linearer Gleichungssysteme mit iterativen Verfahren

Zielstellung

1. Wiederholung wesentlicher Elemente von Python (Methoden, Klassen, Module) und ggf. Erarbeitung neuer Konzepte
2. Wiederholung wesentlicher Elemente von LaTeX zur Erstellung einer Programmdokumentation und eines wissenschaftlichen Berichtes
3. Erlangen eines Grundverständnis von finiten Differenzen zur Approximation von Ableitungen von Funktionen
4. Analyse des Konvergenzverhalten eines Approximationsverfahrens und die Auswirkung von Rundungsfehlern

1 Theoretischer Hintergrund — Finite Differenzen

Es sei $f \in C^\infty([a,b])$ eine reellwertige und glatte Funktion auf einem Intervall $[a,b]$ gegeben. Mit Hilfe der Taylorreihe lässt sich der Funktionswert von f an der Stelle $x_+ \coloneqq x + h \in (a,b)$ mit $h > 0$ wie folgt approximieren:

$$f(x_+) = \sum_{n=0}^{\infty} \frac{f^{(n)}(x)}{n!} h^n = f(x) + f'(x)h + \frac{f''(x)}{2}h^2 + \cdots \tag{1}$$

Approximation von $f'(x)$ Hieraus lässt sich die Formel der finiten Differenz $D_h^{(1)}$ (hier der rechtsseitige Differenzenquotient) ableiten, mit der sich die erste Ableitung im Punkt x approximieren lässt. Es gilt

$$f'(x) = \underbrace{\frac{f(x_+) - f(x)}{h}}_{=:D_h^{(1)}x} + \sum_{n=2}^{\infty} \frac{f^{(n)}(x)}{n!} h^{n-1} = D_h^{(1)}x + \mathcal{O}(h).$$

Approximation von $f''(x)$ Betrachte nun zusätzlich die Taylorentwicklung von f in $x_- \coloneqq x - h$:

$$f(x_-) = \sum_{n=0}^{\infty} \frac{f^{(n)}(x)}{n!}(-h)^n = f(x) - f'(x)h + \frac{f''(x)}{2}h^2 - \cdots \tag{2}$$

Aus (1) und (2) bilden wir die Summe

$$f(x_+) - 2f(x) + f(x_-) = f''(x)h^2 + \frac{f^{(4)}(x)h^4}{24!} + \cdots$$

und erhalten den symmetrischen Differenzenquotienten $D_h^{(2)}$ zur Approximation der zweiten Ableitung von f

$$f''(x) = \underbrace{\frac{f(x_+) - 2f(x) + f(x_-)}{h^2}}_{=:D_h^{(2)}x} + \mathcal{O}(h^2)$$

und einer Fehlerabschätzung der Ordnung h^2.

Diese Formeln bieten die Grundlage für das numerische Approximieren von Ableitungen einer Funktion. Bei der Arbeit an den folgenden Aufgabenblättern werden diese Differenzenquotienten genutzt um partielle Differentialgleichungen mit Randbedingungen numerisch zu lösen.

2 Aufgaben Teil 1 — Aufwärmübungen mit Python und LaTeX

Gegeben seien ein Intervall $[a, b] \subseteq \mathbb{R}$ eine reelle Funktion $f : [a, b] \to \mathbb{R}$, sowie ihre erste und zweite Ableitung (f', f''), eine Schrittweite h zur Steuerung der Genauigkeit der Approximationen der Ableitungen und eine Anzahl an Punkten $p \in \mathbb{N}$ für die Feinheit der Plots. Es seien $x_i \coloneqq a + i\frac{|b-a|}{p}$ für $i = 0, \ldots, p$ die Plotpunkte.

Aufgabe 1.1: Implementierung in Python

Erstellen Sie in Python eine Klasse die folgende Funktionalitäten erlaubt.

- Zeichnen der Funktion auf dem Intervall. Die Funktion soll dabei nur an den Plotpunkten ausgewertet werden.

- Approximation der ersten und zweiten Ableitung der Funktion an den Plotpunkten $D_h^{(1)} x_i, D_h^{(2)} x_i$, entsprechend der Formeln aus Abschnitt 1.
- Zeichnen der (exakten) ersten und zweiten Ableitung der Funktion, sowie die entsprechenden Approximationen auf dem Intervall $[a, b]$ (an den Plotpunkten).
- Berechnung der Fehler an den Plotpunkten entsprechend folgender Formel

$$e_h^{(k)} := \max_{i=0,\dots,p} |f^{(k)}(x_i) - D_h^{(k)} x_i|, \qquad k = 1, 2.$$

Aufgabe 1.2: Fehlerplot

Erzeugen Sie im Hauptprogramm einen Plot der Fehler. Mit folgenden Eigenschaften

- Der Plot zeigt die Fehler $e_h^{(1)}$ und $e_h^{(2)}$ in Abhängigkeit von h.
- Die Graphik ist doppelt-logarithmisch skaliert.
- Die Graphik enthält zusätzlich die Graphen der Funktionen $h \mapsto h^j$ für $j = 1, 2, 3$.

Schreiben Sie dazu eine Methode oder Funktion, die folgende Funktionalität realisiert.

- Für ein Objekt der Klasse aus Aufgabe 1.1 und eine Kollektion von Schrittweiten wird einem axis-Objekt der Plot der Fehler $e_h^{(1)}$ und $e_h^{(2)}$ in Abhängigkeit von h hinzugefügt.

Ein Beispiel wie auf diese Weise verschiedene Graphen in einen Plot gezeichnet werden können finden Sie in der Datei `plot_to_axis_example.py` in Moodle.

Aufgabe 1.3: Hauptprogramm

Gegeben sei $g_1(x) = \sin(x)$ auf $[0, \pi]$ und $p := 1000$. Schreiben Sie ein Hauptprogramm, welches die Funktionalität Ihrer Implementierung anhand der Funktion g_1 demonstriert. Verwenden Sie die von Ihnen bereits implementierte/n Klasse/n und Methode/n.

Aufgabe 1.4: Schnittstellendoku und Bedienungsanleitung

Erstellen Sie mit Sphinx oder LaTeX eine Schnittstellendoku und Bedienungsanleitung. Die benötigten Informationen sind aus Einführung in das wissenschaftliche Rechnen bekannt und in den *Hinweisen zur Abgabe* (z.B. in Moodle) noch einmal aufgelistet.

Allgemeine Hinweise

- Verwenden Sie pylint um Ihren Quelltext selbst zu bewerten und zu verbessern und achten Sie darauf, jeden Plot mit Titel, Legende und Achsenbeschriftung auszustatten.
- *Tipp:* Verwenden Sie Klassen, Methoden und *ndarrays* aus *numpy* um die Daten sinnvoll zwischen zu speichern
- Laden Sie Ihre Lösung zu Serie 1, bestehend aus einer oder mehreren python Dateien und der Doku als pdf, in Moodle hoch. Wählen Sie aussagekräftige Dateinamen.