Praktikum zur Numerischen Mathematik im Sommersemester 2021
Prof. Dr. Nils Rosehr, Hochschule München, 28. April 2021

Praktikum 2: Nullstellenbestimmung

Implementierung einer Bibliothek

Schreiben Sie eine Bibliothek `lib2`, indem Sie die folgenden Funktionen implementieren.

- `Bisektion(f, a, b, n)`,
- `Regula_falsi(f, a, b, n)`,
- `Sekantenverfahren(f, a, b, delta, n)` und
- `Newton_Verfahren(f, df, a, delta, n)`.

Dabei sei f eine Funktion und df deren Ableitung. Durch die Funktionen sollen die entsprechenden Nullstellenverfahren der Vorlesung implementiert werden. Achten Sie hier darauf, genau die Formeln aus dem Skript zu verwenden (Die Gleitpunktarithmetik ist nicht assoziativ, und Umklammerungen, können zu anderen Ergebnissen führen als bei meiner Referenzimplementation.) Der Rückgabewert aller Funktionen soll eine Liste von Näherungswerten sein, die in den ersten drei Fällen wie `[a, b, m1, m2, ...]` und beim Newton-Verfahren wie `[a, x2, x3, ...]` beginnt. Die Iteration soll höchstens so oft durchgeführt werden, dass die Rückgabeliste höchstens die Länge n hat und dass in den letzten beiden Fällen der Absolutbetrag der Differenz der letzten beiden Iterationswerte kleiner oder gleich `delta` ist.

Untersuchungen zur Konvergenz

- Importieren Sie `vorgabe2.py` in `main2.py` mit `from vorgabe2 import *`. Nun sind Tupel `data[1]` bis `data[5]` vordefiniert, die die Werte `f, df, a, b, n, t` wie oben enthalten; dabei ist `t` eine exakte Nullstelle der Funktion `f`.
- Plotten Sie die Funktionen mit Hilfe von `numpy` und `matplotlib`, um sich einen Überblick zu verschaffen.
- Erstellen Sie eine übersichtliche Ausgabe der Näherungswerte Ihrer Verfahren und der zugehörigen absoluten Fehler. Sie können dafür die Funktion `Tabelle(f, df, a, b, n, delta)` und `print_Tabelle(T)` verwenden. Die erste erzeugt eine $n \times 4$-Matrix `T` mit den Nullstellenapproximationen Ihrer Verfahren, und die zweite macht eine Ausgabe solch einer Tabelle. Man beachte, dass `T - t` eine Tabelle der absoluten Fehler (mit Vorzeichen) ist, wenn `t` eine exakte Lösung ist. (Intern werden hier `numpy`-Arrays benutzt; das ist aber für die Anwendung hier nicht wichtig.)
- Sie sollten nun in der Lage sein, folgende Fragen beantworten zu können.
  (1) Wie beurteilen Sie die Konvergenz der verschiedenen Verfahren für `data[1]`?
  (2) Woran liegt es, dass im Vergleich zu `data[1]` die Konvergenz der Regula falsi für `data[2]` so gut ist?
  (3) Was ist bei `data[3]` los?
  (4) Warum konvergiert das Newton-Verfahren für `data[4]` nicht?
  (5) Warum konvergiert das Newton-Verfahren für `data[5]` so langsam? Wie kann man dies verbessern?
  (6) Wie ist das Konvergenzverhalten der Intervallbreite bei der Regula falsi für `data[5]`? Woran liegt das?

Anwendung auf Polynome

Ergänzen Sie `lib2` durch folgende Funktionen:

- `Hornerschema(a, x0)`,
- `Polynom_Funktion(a)`,
- `Diff_Polynom(a)` und
- `Nullstellen_Polynom(a, delta, k)`

Dabei soll das Hornerschema für ein Polynom $p(x) = a_0x^n + a_1x^{n-1} + \cdots + a_{n-1}x + a_n$ und den Linearfaktor $x - x_0$ wie in der Vorlesung beschrieben implementiert werden. Der Übergabeparameter `a` ist eine Liste der Länge $n + 1 \geq 1$, die wie oben die Koeffizienten beginnend mit dem Leitkoeffizient a_0 enthält. Der Rückgabewert ist die Liste $b_0, b_1, \ldots, b_n$ gleicher Länge wie in der Vorlesung.
Die Funktion `Polynom_Funktion(a)` soll aus der Koeffizientenliste `a` die zugehörige Polynom**funktion** $x \mapsto p(x)$ wie oben machen und diese zurückgeben. Benutzen Sie hierzu einfach `Hornerschema(a, x)[-1]` (nach Vorlesung ist dies gleich $b_n = p(x)$), und beachten Sie, dass Sie innerhalb von `Polynom_Funktion` eine Funktion definieren müssen und diese zurückgeben müssen.
Die Funktion `Diff_Polynom(a)` soll die Koeffizientenliste der Ableitung des zu `a` gehörigen Polynoms p zurückgeben.
Schließlich soll die Funktion `Nullstellen_Polynom(a, delta, k)` alle komplexen Nullstellen des Polynoms $p(x)$ in Form einer Liste der Länge n zurückgeben und dazu den in der Vorlesung beschriebenen Algorithmus verwenden. Intern soll das Newton-Verfahren mit Genauigkeitsparametern `delta` und `k` (`k` heißt oben `n`) aufgerufen werden.
Mit der Funktion `np.random.uniform(a, b)` lassen sich gleichverteilte Pseudo-Zufallswerte zwischen a und b erzeugen. Eine komplexe Zahl, etwa $3 - 2j$, kann z.B. mit `complex(3, -2)` oder direkt mit `3-2j` erzeugt werden.

Berechnen Sie die Nullstellen der Polynome

$$x^3 - 6x^2 + 11x - 6,$$

$$x^5 + (-1 + j)x^4 + (-3 + 3j)x^3 + (1 - 11j)x^2 + (2 - 3j)x + 10j$$

und

$$x^{50} + 8x^5 - \frac{80}{3}x^4 + 20x^3 - 2x + 1$$

mit Parametern `delta` $= 10^{-5}$ und `k` $= 50$.

Abgabe

- Laden Sie das Archiv `P2vorgabe.zip` von moodle herunter, entpacken Sie es, und testen Sie Ihre Programme, indem Sie `test2.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 `info2.md` aus, und vergessen Sie nicht die Quellenangabe.** Abgaben mit unvollständiger Datei `info2.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 mit `P2.zip`.
- Schreiben Sie eine Email an `rosehr@hm.edu` mit Betreff `Numerik Abgabe`, Dateianhang `P2.zip` und beliebigem sonstigen Inhalt. **Wichtig ist, dass Betreff und Dateiname des Anhangs stimmen!**

Praktikumstermine P2: keine; **Abgabetermin:** 18.05.