Übungsblatt 6 zur Vorlesung Grundlagen der Programmierung

Übung 1
Kubikwurzelziehen mit Newton-Raphson
Implementieren Sie das Finden der Kubikwurzel mithilfe des Newton-Raphson-Verfahrens und erstellen Sie einen vergleichenden Benchmark mit der Implementierung auf Basis der binären Suche.

Übung 2
Nullstellen eines Polynoms
Gegeben sei die Funktion

$$f(x) = x^3 - 2x + 3x - 5$$

Finden Sie eine Nullstelle dieser Funktion ausgehend vom Startwert $x_0 = 2$ mithilfe der binären Suche sowie mithilfe des Newton-Raphson-Verfahrens.

Übung 3
Finde die Position des Minimums eines Polynoms
Wir haben in der letzten Übung die Position des Minimums einer beliebigen Quadratischen Funktion mit der binären Suche gelöst. Wir wollen die Aufgabe nun auch für das Polynom der Form $f(x) = ax^4 - bx + c$ lösen und erneut versuchen den Wert x zu finden, für den die Funktion ihr Minimum erreicht. Die Werte für a, b und c werden weiterhin zufällig in Schranken erzeugt, so dass die Funktion im Intervall $x \in [-10{,}10]$ immer ein Minimum erreicht. Abb. 1 zeigt drei Beispiele zufällig parametrisierte Funktionen dieser Form.

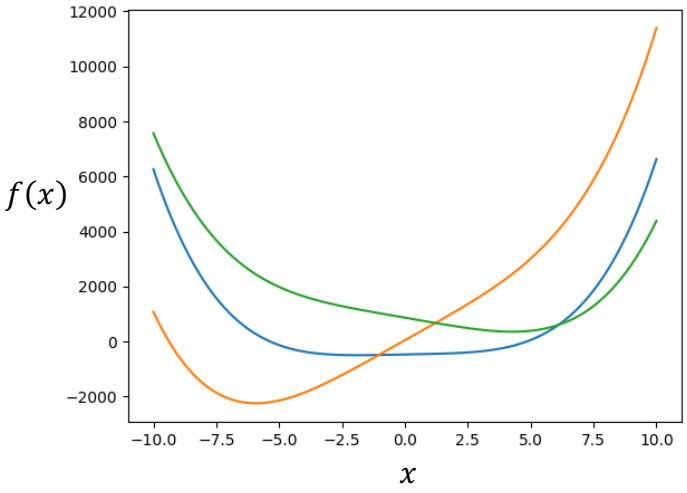

Abb. 1: Drei zufällig parametrisierte Polynome der Form $f(x) = ax^4 - bx + c$

Bitte implementieren Sie die Suche nach dem Minimum der Funktion unter Verwendung
1) der binären Suche, sowie

2) des Newton-Raphson Verfahrens.

Hinweise:

- Durch die Einschränkung der Werte für a, b und c finden Sie im Intervall $x \in [-10{,}10]$ immer das Minimum der Funktion
- Weiterhin ist die Ableitung $f'(x)$ im Intervall $x \in [-10{,}10]$ stetig steigend. Den Wendepunkt der Funktion finden Sie, wenn die Ableitung den Wert Null annimmt.

```
import random

a = random.uniform(0.5, 1.0)
b = random.uniform(0, 1000)
c = random.uniform(-1000, 1000)

def ableitungPolynom(x):
    '''
    x: Position
    returns: Wert der Ableitung an Position x
    '''
    return 4*a*x**3 - b


print("Finde die Position des Minimums des Polynoms f(x)=" +
      str(f"{a:.2f}") + "x^4 - " + str(f"{b:.2f}") + "x" + str(f"{c:+.2f}") +
      " im Intervall [-10,10]")

# Ihr Code kommt hier - das Ergebnis bitte in der
# Variable posMinimumPolynom ablegen und
# die Anzahl benötigter Schritte in anzSchritte:
posMinimumPolynom = 7
anzSchritte = 0

print("Anzahl Schritte= " + str(anzSchritte))
print("Angenähertes Ergebnis: " + str(f"{posMinimumPolynom:.2f}"))
print("Zum Testen: Korrektes Ergebnis: " + str(f"{(b/(4*a))**(1/3):.2f}"))
```

Übung 4

Runden binärer Zahlen auf eine feste Anzahl Nachkommastellen

Erweitern Sie den Code zur Umrechnung dezimaler Zahlen zu binären Zahlen aus dem Skript dahingehend, dass Sie die binäre Zahl auf eine feste Anzahl Nachkommastellen runden. Zusätzlich soll die Differenz zwischen der gerundeten Binärzahl und der ursprünglichen Dezimalzahl berechnet und angezeigt werden. Die Rundung erfolgt, indem die Nachkommastellen über der gewünschten Anzahl (n) betrachtet werden. Ist die erste abgeschnittene Stelle 1, wird der letzte erhaltene Nachkommateil aufgerundet, andernfalls abgerundet.

Probieren Sie aus: Welchen Einfluss hat die Bitlänge auf die Präzision Ihrer Rundung?