Newton-Raphson vs Bisection Method comparison

shem nicholas opiyo
SCT211-0509/2021

March 2023

1 Explanation of the Code

This code compares the runtime of bisection method and newton raphson method, when finding the root of $x^4 + x^3 - 2x^2 + x - 18$. It starts by describing the elements used by methods for the comparison. These include describing the function to be used, the function's derivative, the range i.e. 'a' and 'b', initial starting point i.e. Xo, and maximum number of iterations i.e. 100 iterations. The code then starts by implementing the bisection method and noting the time taken to find the root of the function. It then implements newton-raphson method and notes the time taken to find the function's root, then compares the runtime of both methods. Finally, the code identifies the method with the shortest time taken to find the root of the function and displays it.

2 Explanation of each section of the code

```
import math
import timeit

# Description of the code and its derivative

# function
def f(x) :
     return x**4+x**3-2*x**2+x-18
#Derivative of the function
def df(x):
    return 4*x**3+3*x**2-4*x+1

# these are the limits to be used in implementing bisection method,
# and indicating the start of the newton raphson method.
a = 1
b = 2
tolerance = 1e-6
```

```
x0 = a
max_iterations = 100

# This section implements the bisection method
start_time = timeit.default_timer()
for  i in range(max_iterations) :
    c = (a+b)/2

    if abs(f(c))<tolerance:
        print(f"Through the Bisection method, root found at x={c:.6f}")
        break
    elif f(c) * f(a) <0 :
        b= c
    else:
        a = c
bisection_time = timeit.default_timer() - start_time
print(f"BisectionTime : {bisection_time}")

# This section implements the newton raphson method
start_time = timeit.default_timer()
for  i in range(max_iterations) :
        fx = f(x0)
        dfx = df(x0)
        x1 = x0 - fx/dfx
        if abs( f(x1) ) < tolerance:
            print(f"Through Newton Raphson method, root found at x = {x1:.6f}
            break
        else:
            x0 = x1

newton_time = timeit.default_timer() - start_time
print(f"Newton Raphson Time : {newton_time}" )

# comparison of the runtime of bisection and newton raphson methods, and the
if newton_time < bisection_time :
    print(f"\n\n Juding from the comparison of the two method's run time, New
else:
    print("\n\nJudging from the comparison of the two methods's runtime, bise
```