pr1_i_math548_midterm_Lazizbek

March 16, 2024

1 Math 548, Midterm. Problem 1. LS

Problem 1

(20 points) Find the roots and their multiplicities of the following functions in the interval [0,2] using the Newton's method. Comment on your results and discuss any commonalities between the results of (i) and (ii).

i. f(x) = 21.12 - 32.4x + 12x^2

ii. h(x) = 2.7951 - 8.954x + 10.56x^2 - 5.4x^3 + x^4

#In which interval (i) has a solution? Determine all of the solution intervals using the sign comparison of the function at boundaries of the intervals [-b, a] or [c, b] starting symmetrically around both sides of the origin, 0, with a tolerance b as 10^(-3)

[6]:
```
# In which interval it has a solution?

import numpy as np
import math

# f(x) = 21.12 - 32.4*(x) + 12*(x)**2
f0 =  21.12 - 32.4*(0) + 12*(0)**2 # f0 = 21.12
frightb = f0
fleftb = f0
fa = f0
fc = f0
solution_ints = list()
b = 0
a = 0
c = 0
# we'll run through [-b, a] or [c, b]

for b in np.arange(0.0, 2.0, 0.001):
  frightb = 21.12 - 32.4*(b) + 12*(b)**2
  fleftb = 21.12 - 32.4*(-b) + 12*(-b)**2

  if np.sign(fleftb) * np.sign(fa) <= 0: # fleftb <= 0 as fa = f0 = 21.12 > 0:
    print(" = b ", b, ";", "fleftb = ", fleftb)
    solution_ints.append([-b, a])
```

```
    a = -b
    fa = fleftb

  if np.sign(frightb) * np.sign(fc) <= 0: # frightb <= 0 as fc = f0 = 21.12 > 0:
    print("b = ", b, ";", "frightb = ", frightb)
    solution_ints.append([c, b])
    c = b
    fc = frightb

print("solution intervals = ", solution_ints)
```

```
b =  1.101 ; frightb =  -0.005987999999996774
b =  1.6 ; frightb =  3.552713678800501e-15
solution intervals =  [[0, 1.101], [1.101, 1.6]]
```

solution intervals = [] means, I have only multiple roots for my square function because if the given function has any negative values my code would capture it with tolerance of b = 10^(-3), and insert it into solution_ints list as a solution interval.

#Determine the solutions using the Newton's method with a tolerance as 10^(-6)

[2]:
```
# Determine this solution using the Newton's method with a tolerance   as␣
↪10^(-6)
from scipy.optimize import fsolve


M = 10**6 # in case the program goes into infinite loops
epsilon = 10**(-8)

for i in range(len(solution_ints)):
  solution_int = solution_ints[i]

  print(f"\nRoot {i+1} of f(x) in {solution_int} is as follows: ")
  x0 = solution_int[1] # by the result of (12), x0=2 would give faster␣
↪convergence
  v = 21.12 - 32.4*(x0) + 12*(x0)**2  # starting function value at x0.

  print("steps", "\t ", "x", "\t", "\t", "\t", "f(x)")

  if abs(v) < epsilon:
    print(0, "\t ", x0, "\t", v)
    x1 = x0
    mnumer = (-32.4 + 24*(x1))**2
    mdenom = mnumer - (21.12 - 32.4*(x1) + 12*(x1)**2)*(24)+0.00000001 # in␣
↪case x0 is a desired root
    m = round(mnumer/mdenom)
    print("m = ", m)
    print(f"multipilicity of the root {x1} is {m}")
```

```
    else:
      for k in range(1, M):
        x1 = x0 - v/(-32.4 + 24*(x0))
        v = 21.12 - 32.4*(x1) + 12*(x1)**2
        print(k, "\t ", x1, "\t", v)
        if abs(v) < epsilon:
          print(f"\nSo the approximate solution {i+1} after {k} steps is = {x1}")
          # The function fsolve takes in many arguments that you can find in the␣
↪documentation,
          # but the most important two is the function you want to find the root,␣
↪and the initial guess interval.
          f = lambda x: 21.12 - 32.4*(x) + 12*(x)**2
          mnumer = (-32.4 + 24*(x1))**2
          mdenom = mnumer - (21.12 - 32.4*(x1) + 12*(x1)**2)*(24)+0.00000001 # in␣
↪case x0 is a desired root
          m = mnumer/mdenom

          print("Using Python's fsolve function, fsolve(f, [0, 2]) = ", ␣
↪*fsolve(f, solution_int[i]), '\n')
          print("m = ", m)
          print(f"multipilicity of the root {x1} is {round(m)}")
          break
        x0 = x1
```

```
Root 1 of f(x) in [0, 1.101] is as follows:
steps     x                      f(x)
1         1.099997991967872      1.2048241156747963e-05
2         1.099999999991936      4.838796030526282e-11

So the approximate solution 1 after 2 steps is = 1.099999999991936
Using Python's fsolve function, fsolve(f, [0, 2]) =  1.0999999999999992

m =  0.9999999997544808
multipilicity of the root 1.099999999991936 is 1

Root 2 of f(x) in [1.101, 1.6] is as follows:
steps     x                      f(x)
0         1.6    3.552713678800501e-15
m =  1
multipilicity of the root 1.6 is 1
```

#In case I calculated the roots of (i) by hand:

[5]:
```
print(32.4**2-4*21.12*12)
print((32.4+6)/(2*12))
```

```python
print((32.4-6)/(2*12))
```

```
36.0
1.5999999999999999
1.0999999999999999
```