pr1_ii_math548_midterm_Lazizbek

March 17, 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 (ii) 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)

[3]:
```python
# In which interval it has a solution?

import numpy as np
import math

# h(x) = 2.7951 - 8.954*x + 10.56*x**2 - 5.4*x**3 + x**4
h0 = 2.7951 - 8.954*(0) + 10.56*(0)**2 - 5.4*(0)**3 + (0)**4 # h(0) = 2.7951
hrightb = h0
hleftb = h0
ha = h0
hc = h0
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):
  hrightb = 2.7951 - 8.954*(b) + 10.56*(b)**2 - 5.4*(b)**3 + (b)**4
  hleftb = 2.7951 - 8.954*(-b) + 10.56*(-b)**2 - 5.4*(-b)**3 + (-b)**4

  if np.sign(hleftb) * np.sign(ha) <= 0: # hleftb <= 0 as ha = h0 = 2.7951 > 0:
    print(" = b ", b, ";", "hleftb = ", hleftb)
    solution_ints.append([-b, a])
```

```
    a = -b
    ha = hleftb

  if np.sign(hrightb) * np.sign(hc) <= 0: # hrightb <= 0 as ha = h0 = 2.7951 >
↪0:
    print("b = ", b, ";", "hrightb = ", hrightb)
    solution_ints.append([c, b])
    c = b
    hc = hrightb

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

```
b =  1.1 ; hrightb =  -6.661338147750939e-16
solution intervals =  [[0, 1.1]]
```

solution intervals = [ ], empty list, means that I have only multiple roots for my 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)

[4]:
```
# 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 h(x) in {solution_int} is as follows: ")
  x0 = solution_int[1] # by the result of (32), x0=2 would give faster
↪convergence even though I am not necessarily starting with x0 = 2
  v = 2.7951 - 8.954*(x0) + 10.56*(x0)**2 - 5.4*(x0)**3 + (x0)**4  # starting
↪function value at x0.

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

  if abs(v) < epsilon:
    print(0, "\t ", x0, "\t", v)
    x1 = x0
    mnumer = ( - 8.954 + 21.12*(x1) - 16.2*(x1)**2 + 4*(x1)**3)**2
    mdenom = mnumer - (2.7951 - 8.954*(x1) + 10.56*(x1)**2 - 5.4*(x1)**3 +
↪(x1)**4)*(21.12 - 32.4*(x1) + 12*(x1)**2)+0.000001 # in case x0 is a desired
↪root
    print("mnumer = ", mnumer)
```

```
  print("mdenom = ", mdenom)
  m = mnumer/mdenom
  print("m = ", m)
  print(f"multipilicity of the root {x1} is {round(m)}")

  # 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: 2.7951 - 8.954*(x) + 10.56*(x)**2 - 5.4*(x)**3 + (x)**4
  print("To check our work, the solution of h(x) = 0 using Python's fsolve function, fsolve(f, [0, 2]) = ",  *fsolve(f, solution_int[i]), '\n')

else:
  for k in range(1, M):
    x1 = x0 - v/( - 8.954 + 21.12*(x0) - 16.2*(x0)**2 + 4*(x0)**3)
    v = 2.7951 - 8.954*(x1) + 10.56*(x1)**2 - 5.4*(x1)**3 + (x1)**4
    print(k, "\t ", x1, "\t", v)
    if abs(v) < epsilon:
      print(f"\nSo the approximate solution {i+1} after {k} steps is = {x1}")

      mnumer = ( - 8.954 + 21.12*(x1) - 16.2*(x1)**2 + 4*(x1)**3)**2
      mdenom = mnumer - (2.7951 - 8.954*(x1) + 10.56*(x1)**2 - 5.4*(x1)**3 + (x1)**4)*(21.12 - 32.4*(x1) + 12*(x1)**2)+0.000001 # in case x1 is the actual(real) root
      print("mnumer = ", mnumer)
      print("mdenom = ", mdenom)
      m = mnumer/mdenom
      print("m = ", m)
      print(f"multipilicity of the root {x1} is {round(m)}")

      # 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: 2.7951 - 8.954*(x) + 10.56*(x)**2 - 5.4*(x)**3 + (x)**4
      print("To check our work, the solution of h(x) = 0 using Python's fsolve function, fsolve(f, [0, 2]) = ",  *fsolve(f, solution_int[i]), '\n')

      break
    x0 = x1
```

```
Root 1 of h(x) in [0, 1.1] is as follows:
steps     x                      h(x)
0         1.1    -6.661338147750939e-16
mnumer =  0.0
```

```
mdenom =  1e-06
m =  0.0
multipilicity of the root 1.1 is 0
To check our work, the solution of h(x) = 0 using Python's fsolve function,
fsolve(f, [0, 2]) =  1.0999932258418257
```

2 Here how I checked the roots of h(x), h'(x), h''(x), h'''(x) to see what's going on with m:

[5]:
```
# h(x) = 0
f = lambda x: 2.7951 - 8.954*(x) + 10.56*(x)**2 - 5.4*(x)**3 + (x)**4
print("To check our work, the solution of h(x) = 0 using Python's fsolve␣
 ↪function, fsolve(f, [0, 2]) = ",  *fsolve(f, 1000), '\n')
```

```
To check our work, the solution of h(x) = 0 using Python's fsolve function,
fsolve(f, [0, 2]) =  2.100000000000002
```

[16]:
```
# h'(x) = 0
f = lambda x: - 8.954 + 21.12*(x) - 16.2*(x)**2 + 4*(x)**3
print("To check our work, the solution of h'(x) = 0 using Python's fsolve␣
 ↪function, fsolve(f, [0, 2]) = ",  *fsolve(f, solution_int[i]), '\n')
```

```
To check our work, the solution of h'(x) = 0 using Python's fsolve function,
fsolve(f, [0, 2]) =  1.0999999815114856
```

[17]:
```
# h''(x) = 0
f = lambda x: 21.12 - 32.4*(x) + 12*(x)**2
print("To check our work, the solution of h''(x) = 0 using Python's fsolve␣
 ↪function, fsolve(f, [0, 2]) = ",  *fsolve(f, solution_int[i]), '\n')
```

```
To check our work, the solution of h''(x) = 0 using Python's fsolve function,
fsolve(f, [0, 2]) =  1.0999999999999992
```

[18]:
```
# h'''(x) = 0
f = lambda x: - 32.4 + 24*(x)
print("To check our work, the solution of h'''(x) = 0 using Python's fsolve␣
 ↪function, fsolve(f, [0, 2]) = ",  *fsolve(f, solution_int[i]), '\n')
```

```
To check our work, the solution of h'''(x) = 0 using Python's fsolve function,
fsolve(f, [0, 2]) =  1.35
```

[15]:
```
# !sudo apt-get install texlive-xetex texlive-fonts-recommended␣
↪texlive-plain-generic
```

[14]:
```
# !jupyter nbconvert --to pdf /content/pr1_ii_math548_midterm_Lazizbek.ipynb
```