pr4_i_math548_midterm_Lazizbek

March 19, 2024

1 Math 548, Midterm. Problem 4. LS

Problem 4

(i)

(a) Use the Newton's method for finding the root of

( ) = ^ − 2 cos( ) = 0

in the interval [0,2] with the starting value x0 = 2.

(b) Carry out 6 iterations and comment on what you observe. If the observation needs any improvement, how will you go about it?

(c) Show that the Newton's method's convergence is of second order.

(ii)

(iii) Now, use the iteration method given below to find the root of

( ) = ^ − 2 cos( ) = 0 in the interval [0,2].

Choose your starting values as x0 = 0.6 and y0 = 0.3388.

yn+1 = yn (2 – f'(xn) yn)

xn+1 = xn – yn+1 f(xn)

(e) Compare your methods and results in (i) and (ii) and discuss any connections between (i) and (ii).

#Problem 4. (i)

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

import numpy as np
import math

# f(x) = math.exp(x) - 2*math.cos(x)

f0 =  math.exp(0) - 2*math.cos(0) # f0 = - 1
frightb = f0
fleftb = f0
fa = f0
```

```python
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 = math.exp(b) - 2*math.cos(b)
  fleftb = math.exp(-b) - 2*math.cos(-b)

  if np.sign(fleftb) * np.sign(fa) <= 0: # fleftb >= 0 as fa = f0 = -1  < 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 = -1 < 0:
    # print("b = ", b, ";", "frightb = ", frightb)
    solution_ints.append([c, b])
    c = b
    fc = frightb

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

```
solution intervals =  [[0, 0.54], [-1.454, 0]]
```

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


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

for i in range(1):
  solution_int = [0, 2]

  print(f"\nRoot {i+1} of f(x) in {solution_int} is as follows: ")
  x0 = solution_int[1] #  x0=2 as required
  v = math.exp(x0) - 2*math.cos(x0)  # 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 = (math.exp(x1) + 2*math.sin(x1))**2
    mdenom = mnumer - (math.exp(x1) - 2*math.cos(x1))*(math.exp(x1) + 2*math.
↪cos(x1))+0.00000001 # in case x0 is a desired root
    m = mnumer/mdenom
    print("m = ", m)
    print(f"multipilicity of the root {x1} is {round(m)}")

  else:
    for k in range(1, M):
      x1 = x0 - v/(math.exp(x0) + 2*math.sin(x0))
      v = math.exp(x1) - 2*math.cos(x1)
      print(k, "\t ", x1, "\t", v)
      if abs(v) < epsilon:
        mnumer = (math.exp(x1) + 2*math.sin(x1))**2
        mdenom = mnumer - (math.exp(x1) - 2*math.cos(x1))*(math.exp(x1) +␣
↪2*math.cos(x1))+0.00000001 # in case x0 is a desired root
        m = mnumer/mdenom
        print("\nm = ", m)
        print(f"multipilicity of the root {x1} is {round(m)}")
        break
      x0 = x1
```

```
Root 1 of f(x) in [0, 2] is as follows:
steps    x                      f(x)
1        1.1071175683826828     2.1311417447814685
2        0.664462389989102      0.3689486075980428
3        0.5483209267201882     0.023543300836922132
4        0.5398302921852223     0.00012382330442339828
5        0.5397851620829291     3.494306399787206e-09

m =  1.0000000002643719
multipilicity of the root 0.5397851620829291 is 1
```

(c) Show that the Newton's method's convergence is of second order.

For this we need to show g(x*) = 0;

g(x*) = 1-1/m = 1-1/1 = 0.

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

```
[ ]: # !jupyter nbconvert --to pdf /content/Math548_hw6_Lazizbek.ipynb
```