Numerical Analysis assignment No. 1

B6TB1505 Daichi HAYASHI (Ohnishi Lab.)

October 9, 2019

1 Assignment Content

Using one of 4 schemes below, find root of the equation (1) and obtain ϕ.

- Interval halving (Bisection)
- False Position (Regula Falsi)
- Newton-Raphson (Newton method)
- Secant method

I choose False position. The reason is, I've already used Newton-Raphson method and I wanted to understand how False position works each iterarion by coding by myself.

$$f(\phi) = \frac{5}{3}\cos 40^\circ - \frac{5}{2}\cos\phi + \frac{11}{6} - \cos\left(40^\circ - \phi\right) \tag{1}$$

I show the Python script of False position in the next section.

2 Python Script and Result

In this script, the desired ϕ is x, and $\varepsilon = 1.0 \times 10^{-9}$. Under the script, result text will be shown.

```
# Numerical Analysis calss Assignment
# created by Daichi Hayashi (B6TB1505) Oct. 07, 2019.
# Python script of "False Position method" (Regula Falsi)
import numpy as np
import matplotlib.pyplot as plt

def func(x):
  alpha = 40.0 # input angle [deg.]
  return 5.0/3.0*np.cos(np.deg2rad(alpha)) - 5.0/2.0*np.cos(np.deg2rad(x)) \
  + 11.0/6.0 - np.cos(np.deg2rad(alpha-x))

def main():
  print('Start finding Root of function with the Regula Falsi...')
  n_max = 1000 # iteration max number
  num_a = 30.0 # smaller initial value [deg.]
  num_b = 40.0 # larger initial value [deg.]
```

```
  eps1  = 1e-9 # width threashold
  eps2  = 1e-9 # threashold

  if func(num_a)*func(num_b) > 0: # same sign -> initial value is wrong
    print("The initial values should be both side of root")
    return
  else:
    for n in range(n_max):
      fa    = func(num_a)
      fb    = func(num_b)
      num_c = (num_b*fa - num_a*fb)/(fa - fb)
      fc    = func(num_c)
      # print status
      print('loop {:>2d}, a={:.6f}, f(a)={:.8f}, b={:.1f}, x={:.7f}, \
f(x)={:.9f}'.format(n+1,num_a,fa,num_b,num_c,fc))
      if abs(fc) < eps2: # judge from function(c)
        print('Root has been found within accuracy (f(x)={:5.1e}, \
loop={:>2d}).'.format(eps2,n+1))
        return
      # judge from |b-a|, in almost case, this condition has NO meaning.
      elif abs(num_b - num_a) < eps1:
        print('Root has been found within the very small phi width \
(eps={:5.1e}, loop={:>2d}).'.format(eps1,n+1))
        return
      if fa * fc > 0:
        num_a = num_c
      elif fb * fc > 0:
        num_b = num_c

if __name__ == '__main__':
  main()
```

```
Start finding Root of function with the Regula Falsi…
loop  1, a=30.000000, f(a)=-0.03979719, b=40.0, x=31.6952277, f(x)=-0.006576877
loop  2, a=31.695228, f(a)=-0.00657688, b=40.0, x=31.9662384, f(x)=-0.001012328
loop  3, a=31.966238, f(a)=-0.00101233, b=40.0, x=32.0077375, f(x)=-0.000154096
loop  4, a=32.007738, f(a)=-0.00015410, b=40.0, x=32.0140495, f(x)=-0.000023417
loop  5, a=32.014050, f(a)=-0.00002342, b=40.0, x=32.0150086, f(x)=-0.000003557
loop  6, a=32.015009, f(a)=-0.00000356, b=40.0, x=32.0151543, f(x)=-0.000000540
loop  7, a=32.015154, f(a)=-0.00000054, b=40.0, x=32.0151764, f(x)=-0.000000082
loop  8, a=32.015176, f(a)=-0.00000008, b=40.0, x=32.0151798, f(x)=-0.000000012
loop  9, a=32.015180, f(a)=-0.00000001, b=40.0, x=32.0151803, f(x)=-0.000000002
loop 10, a=32.015180, f(a)=-0.00000000, b=40.0, x=32.0151803, f(x)=-0.000000000
Root has been found within accuracy (f(x)=1.0e-09, loop=10).
```

3 Algorithm and Ingenuity

In my script, if $f(a) \times f(b) > 0$, error would be returned. This means the guess values is same sign, so we don't have any loot between a and b. Because of this algorithm, I believe this script becomes more robust.

Like this, if $f(a) \times f(b) > 0$, cross point c will be next a because they have same sign.

4 Discussion

From comparison the result mine and the reference[1]'s, the loop count is larger than reference's. The difference of single and double precision cause this. Python is double precision, so the ε is not rounded. On the other hand, the text's is single precision, so the ε is rounded, and made different result.

I showed $f(\phi)$ one more digit. In loop 9, that is -0.000000002. If the last number was rounded, $f(\phi)$ became less than ε. This is the reason why result is different.

Expect the last digit, result numbers $(a,\ f(a),\ b,\ \phi,\ f(\phi))$ are good agreement with reference[1].

References

[1] Joe D. Hoffman, "*Numerical Methods for Engineers and Scientists, 2nd Edition Revised and Expanded*", CRC Press (2001).