Paul Simmerling
pws17001

Homework 4

ECE 3431: Numerical Methods
November 22, 2020

1 Question 22

Roots: [-j, 1.1, j, -3.2]

2 Question 23

Roots: [1, -1, 2, -2, 156]

3 Question 24

Roots: [1, -3, 2-j, 2+j, -3+j, -3-j]

4 Question 25

Roots: [0.5, 1-0.5j, 1+0.5j, -1-2j, -1.99999994-1.19248885e-07j, -2.00000006+1.19248885e-07j, -1+2j]

5 Question 26

Roots: [0.89745177+0.75566436j, -0.60963857-0.67275366j, 2.7121868+2.9170893j]

6 Question 27

Roots: [2, 2j, -3j, -7]

7 Question 28

FFD(x=0.8,h=0.05)=0.6784
error=-2.62761
FFD(x=0.8,h=0.1)=0.6597
error=-5.31166
FFD(x=0.8,h=0.25)=0.60024
error=-13.8461
FFD(x=0.8,h=0.5)=0.4924
error=-29.32464
FFD(x=0.8,h=1)=0.25649
error=-63.18537
FCD(x=0.8,h=0.05)=0.6964
error=-0.04402
FCD(x=0.8,h=0.1)=0.69555
error=-0.16603
FCD(x=0.8,h=0.25)=0.68946
error=-1.04014
FCD(x=0.8,h=0.5)=0.66804
error=-4.1146
FCD(x=0.8,h=1)=0.58626
error=-15.85268
Best FFD is h=0.05 at 2.6% error
Best FCD is h=0.05 at 0.04% error

8 From many h test code at end

Best FFD is h=0.0026 at 0.08% error
Best FCD is h=[0.0076, 0.0152, 0.0167, 0.0213, 0.0258, 0.0289] at 0.00047232% error.

```
#HW4:
#from evalPoly import *
import numpy as np
import cmath
from random import random
#Evaluate poly with complex coeffs a at point x
#return its first and second deriv at x
def evalPoly(a,x):
    n = len(a) - 1
    p = a[n]
    dp = 0.0 + 0.0j
    ddp = 0.0 + 0.0j
    for i in range(1,n+1):
      #p''_i = x*p''_{i-1} + 2*p'_{i-1}
        ddp = ddp*x + 2.0*dp
        #p'_i = x*p'_{i-1} + p_{i-1}
        dp = dp*x + p
        p = p*x + a[n-i]
    return p,dp,ddp

def polyRoots(a,tol=1.0e-12):
    def laguerre(a,tol):
      #choose random starting test root
        x = random()
        n = len(a)-1
        for i in range(30):
          #evaluate root at that point
            p,dp,ddp = evalPoly(a,x)
            if abs(p) < tol: return x#if in tol return root
            g = dp/p
            h = g*g - ddp/p#neg deriv of g
            #Use eq (x-r)=n/(G+-sqrt((n-1)[nH-G^2]))
            f = cmath.sqrt((n-1)*(n*h-g*g))#bottom sqrt^
            if abs(g+f) > abs(g-f): dx = n/(g+f)#determin if + or -
            else: dx = n/(g-f)
            x -= dx#Improve the root guess
            if abs(dx) < tol: return x#If dx is small, close to root
        print('Too many iterations')
    def deflPoly(a,root):
        n = len(a)-1
        b = [(0.0 + 0.0j)]*n
        b[n-1] = a[n]
        for i in range(n-2, -1, -1):
          #synthetically divide out root from poly
```

```
            b[i] = a[i+1] + root*b[i+1]
        return b

    n = len(a)-1
    #Make array with element for each root
    roots = np.zeros((n),dtype=complex)
    for i in range(n):
        x = laguerre(a,tol)
        #if real or imag are tiny, round off error
        if abs(x.imag) < tol: x = x.real
        if abs(x.real) < tol: x = x.imag*1j
        roots[i] = x
        #remove root and repeat process
        a = deflPoly(a,x)
    return roots




#FFD = (f(x+h)-f(x))/h, round everything to 5 sig fig
def firstForwardDifference(f,x,h,sig_figs=5):
    return round((round(f(x+h),sig_figs)\
                  -round(f(x),sig_figs)\
                  )/h,sig_figs)

#FCD = (f(x+h)-f(x-h))/(2h), round everything to 5 sig fig
def firstCentralDifference(f,x,h,sig_figs=5):
    return round((round(f(x+h),sig_figs)\
                  -round(f(x-h),sig_figs)\
                  )/(2*h),sig_figs)


if __name__ == "__main__":
    print("+-------------------------+")
    print("|                         |")
    print("|   ECE 3431 Homework 4   |")
    print("|     Paul Simmerling     |")
    print("|                         |")
    print("+-------------------------+")
    #P22: Page 181 Problem 10
    #P4(x)=x^4 + 2.1x^3 - 2.52x^2 + 2.1x - 3.52
    c22 = np.array([-3.52,2.1,-2.52,2.1,1])
    print("")
    print("+-----+")
    print("| Q22 |")
```

```
print("+-----+")
print("Roots:",polyRoots(c22))

#P23: Page 181 Problem 11
#P5(x) = x^5 - 156x^4 - 5x^3 + 780x^2 + 4x - 624
c23 = np.array([-624,4,780,-5,-156,1.0])
print("")
print("+-----+")
print("| Q23 |")
print("+-----+")
print("Roots:",polyRoots(c23))

#P24: Page 181 Problem 12
#P6(x) = x^6 + 4x^5 - 8x^4 - 34x^3 + 57x^2 + 130x - 150
c24 = np.array([-150,130,57,-34,-8,4,1.0])
print("")
print("+-----+")
print("| Q24 |")
print("+-----+")
print("Roots:",polyRoots(c24))

#P25: Page 181 Problem 13
#P7(x) = 8x^7 + 28x^6 + 34x^5 - 13x^4
#         - 124x^3 + 19x^2 + 220x - 100
c25 = np.array([-100,220,19,-124,-13,34,28,8.0])
print("")
print("+-----+")
print("| Q25 |")
print("+-----+")
print("Roots:",polyRoots(c25))

#P26: Page 181 Problem 14
#P3(x) = 2x^3 - 6(1 + i)x^2 + x - 6(1 - i)
c26 = np.array([-6*(1-1j),1,-6*(1+1j),2.0])
print("")
print("+-----+")
print("| Q26 |")
print("+-----+")
print("Roots:",polyRoots(c26))

#P27: Page 181 Problem 15
#P4(x) = x^4 + (5 + i)x^3 - (8 - 5i)x^2 + (30 - 14i)x - 84
c27 = np.array([-84,30-14j,-1*(8-5j),5+1j,1])
print("")
print("+-----+")
```

```
print("| Q27 |")
print("+-----+")
print("Roots:",polyRoots(c27))

#P28: Page 196 Problem 10
#Using five significant figures in the computations,
#write a program to determine d(sin x)/dx at x = 0.8
#from "the first forward difference approximation" and
#"the first central difference approximation."
#In each case, use h = 1, 0.5, 0.25, 0.1 or 0.05
#that gives the most accurate result (with least error).
#This requires experimentation.
print("")
print("+-----+")
print("| Q28 |")
print("+-----+")
import math
def f28(x): return math.sin(x)
df_at_x = math.cos(0.8)
#test all h with ffd
for h in [0.05, .1, .25, .5, 1]:
    ffd = firstForwardDifference(f28,0.8,h)
    error = round((ffd-df_at_x)/df_at_x*100,5)
    print("FFD(x=0.8,h={})={}".format(h,ffd))
    print("\terror={}\n".format(error))

#test all h with fcd
for h in [0.05, .1, .25, .5, 1]:
    fcd = firstCentralDifference(f28,0.8,h)
    error = round((fcd-df_at_x)/df_at_x*100,5)
    print("FCD(x=0.8,h={})={}".format(h,fcd))
    print("\terror={}\n".format(error))


#test many h to find best h for ffd
dh = 0.0001
h = np.array(np.arange(dh, 1000*dh, dh))
e = np.array([(0.0,0.0)]*len(h),dtype=[('h',float),('e',float)])
for i in range(0,len(h)):
    ffd = abs(firstCentralDifference(f28,0.8,h[i]))
    e[i] = (h[i],abs((ffd-df_at_x)/df_at_x*100))

e_sorted = np.sort(e,order='e')
print(e_sorted[:20])
```