CPL_Midsem

October 9, 2021

[ ]:
```
%run Function_Library.ipynb
```

[30]:
```
#Q1
#Kavin.A.S.B(1911085)
def h(x):
    return (x-5)*math.exp(x)+5

eps=10**-6

p=1
q=10
a,b=bracketing(p,q,h)
print("\nFrom Newton Raphson Method")
x=5
root=newton_raphson(x,h)
print("Nearest root of the given function for " + str(x) + " is x = "+str(root))
#b = kx/hc
b = 344.9453
print("The value of b =" + str(b))
#Kavin.A.S.B(1911085)
```

```
From Newton Raphson Method
Nearest root of the given function for 5 is x = 4.965114231744276
The value of b =344.9453
```

[36]:
```
A = [[0,0,0,2],[0,0,3,0],[0,4,0,0],[5,0,0,0]]
C = [[0,0,0,2,1,0,0,0],[0,0,3,0,0,1,0,0],[0,4,0,0,0,0,1,0],[5,0,0,0,0,0,0,1]] 
 ↪# augmented matrix
print("The augmented matrix is: ")
print_matrix(C,4,8)
GJ, d=gauss_jordan(C,4,8)
if GJ!=None:
    M=get_inv(C,4)
    M=round_matrix(M,2)
    print("The inverse matrix is: ")
    print_matrix(M,4,4)
else:
```

```
    print("No unique solution exists.")
```

```
The augmented matrix is:

0    0    0    2    1    0    0    0

0    0    3    0    0    1    0    0

0    4    0    0    0    0    1    0

5    0    0    0    0    0    0    1
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-36-5deb19ca9f8f> in <module>
      3 print("The augmented matrix is: ")
      4 print_matrix(C,4,8)
----> 5 GJ, d=gauss_jordan(C,4,8)
      6 if GJ!=None:
      7     M=get_inv(C,4)

<ipython-input-4-6c373844fedc> in gauss_jordan(Ab, nrows, ncols)
     53                 # does partial pivoting
     54                 Ab = partial_pivot(Ab,r,nrows)
---> 55             fact=Ab[r][r]
     56             det=det*fact # calculates the determinant
     57             for c in range(r,ncols):

TypeError: 'int' object is not subscriptable
```

[35]:

```
#Q3
#Kavin.A.S.B(1911085)
# LU decomposition using Doolittle's condition L[i][i]=1



print("The matrix is: ")
A,r,c = read_matrix('A.txt')
print_matrix(A,r,c)

vect=[-9,5,7,11]

# partial pivoting
A, vector = partial_pivot_LU(A, vect, r)
A = LU_doolittle(A,r)
```

```
print("The transformed LU matrix is ")
print_matrix(A,r,r)

x = [0 for i in range(r)]

x = for_back_subs_doolittle(A,r,vect)

print("Solutions are : ")
for i in range(r):
    print("x["+str(i)+"] = "+str(x[i]))



#Kavin.A.S.B(1911085)
```

```
The matrix is:

3.0    -7.0    -2.0    2.0

-3.0    5.0    1.0    0.0

6.0    -4.0    0.0    -5.0

-9.0    5.0    -5.0    12.0


The transformed LU matrix is

3.0    -7.0    -2.0    2.0

-1.0    -2.0    -1.0    2.0

-3.0    8.0    -3.0    2.0

2.0    -5.0    0.3333333333333333    0.33333333333333304


Solutions are :
x[0] = 3.0
x[1] = 4.0
x[2] = -6.0
x[3] = -1.0
```

[14]:
```
#Q4
#Kavin.A.S.B(1911085)
def f(x):
    return 4*math.sin(x)*math.exp(-x)-1
```

```
eps=10**-6

p=0
q=1
a,b=bracketing(p,q,f)

print("\nFrom Bisection Method")
root=bisection(a,b,f)
if p==a and q==b:
    print("Root of the function in the interval (" + str(p) + "," + str(q) + ")␣
 ↪= "+str(root))
else:
    print("Root does not lie in the given range (" + str(p) + "," + str(q)+")")
    print("We change the interval to (" + str(a) + "," + str(b)+")")
    print("Root of the given function in the interval (" + str(a) + "," + str(b)␣
 ↪+ ") is "+str(root))

print("\nFrom Regular Falsi Method")
root=regula_falsi(a,b,f)
if p==a and q==b:
    print("Root of the function in the interval (" + str(p) + "," + str(q) + ")␣
 ↪= "+str(root))
else:
    print("Root does not lie in the given range (" + str(p) + "," + str(q)+")")
    print("We change the interval to (" + str(a) + "," + str(b)+")")
    print("Root of the given function in the interval (" + str(a) + "," + str(b)␣
 ↪+ ") is "+str(root))
#Kavin.A.S.B(1911085)
```

```
From Bisection Method
Root of the function in the interval (0,1) = 0.3705587387084961

From Regular Falsi Method
Root of the function in the interval (0,1) = 0.3705584003334566
```

[16]:

```
#Kavin.A.S.B(1911085)
import math
import matplotlib.pyplot as plt

plt.figure(figsize=(9,6))
p=0
q=1
a,b=bracketing(p,q,f)
x_bis, y_bis =bisection_for_plotting(a,b,f)
x_rf, y_rf =regula_falsi_for_plotting(a,b,f)
```

```python
print("\nBISECTION METHOD")
a=pd.DataFrame(y_bis,x_bis)
print(a)
print("\n\nREGULA FALSI METHOD")
b=pd.DataFrame(y_rf,x_rf)
print(b)

plt.plot(x_bis, y_bis, 'r-o', label='Bisection')
plt.plot(x_rf, y_rf, 'g-o', label='Regula Falsi')

plt.grid(color='b', ls = '-.', lw = 0.5)
plt.xlabel('No. of terms in taylor expansion')
plt.ylabel('Error')
plt.title('Error vs No. of terms curve')
plt.legend()
plt.show()
#Kavin.A.S.B(1911085)
```

```
BISECTION METHOD
               0
1  -2.292864e-01
2   6.940729e-03
3  -1.002927e-01
4  -4.406794e-02
5  -1.792540e-02
6  -5.334495e-03
7   8.423635e-04
8  -2.236228e-03
9  -6.944759e-04
10  7.455742e-05
11 -3.098058e-04
12 -1.175858e-04
13 -2.150461e-05
14  2.652880e-05
15  2.512695e-06
16 -9.495808e-06
17 -3.491519e-06
18 -4.894027e-07
19  1.011648e-06


REGULA FALSI METHOD
               0
1   2.889616e-01
2   2.534652e-01
```

```
3    1.630040e-01
4    8.445109e-02
5    3.886823e-02
6    1.691148e-02
7    7.177851e-03
8    3.014426e-03
9    1.260307e-03
10   5.259410e-04
11   2.193102e-04
12   9.141959e-05
13   3.810315e-05
14   1.588027e-05
15   6.618275e-06
16   2.758210e-06
17   1.149498e-06
18   4.790581e-07
```

output_5_1.png

[ ]:

```

```