Mid-Semester Examination (P-346)

Sarbajit Mazumdar, Roll-1911147

Question 1:

Wein’s displacement law states that black body radiation for different temperatures peak at different wavelengths (λ_m) that are inversely proportional to the temperature T, i.e. $\lambda_m T = b$, where b is Wein’s constant.

It can be solved using solution of the equation

$$(x-5)e^x + 5 = 0$$

where $x = \frac{hc}{\lambda_m T k}$, Hence

$$\lambda_m T = b = \frac{hc}{kx}$$

Given:

$h = 6.626 \times 10^{-34} \ \ kg - m^2/s$

$k = 1.381 \times 10^{-23} \ \ - kg/Ks^2$

$c = 3 \times 10^8 \ m/s$

In [26]:

```
#Answer
# We will solve this Question using the Newton-Rhapson Method, to a precision of 10^(-
4) #sarbajitmazumdar_1911147

from My_Lib import *

import math

def f(x):
    return (x-5)*math.exp(x)+5    #Given Function for calculating wien's constant

root_val,a,b=Newton_Raphson_Method(f,10,10**(-4))#calling Newton Rhapson method,precisi
on=10^(-4)

print("The value of the Wien's constant is b = hc/kx =", 6.626*10**(-34)*3*10**8/(1.381
*10**(-23)*root_val),"m-Kelvin") #printing the result
```

```
The value of the Wien's constant is b = hc/kx = 0.0028990007255452407 m-Ke
lvin
```

Question 2:

Checking the inverse of a following matrix exists or not $\begin{bmatrix} 0 & 0 & 0 & 2 \\ 0 & 0 & 3 & 0 \\ 0 & 4 & 0 & 0 \\ 5 & 0 & 0 & 0 \end{bmatrix}$

Lastly I've verified whether the inverse exists or not Using $AA^{-1} = I$. And Inverse exists!!

In [39]:

```
# Printing the matrix in a readable form
list_C=[]
with open("matrix1.txt") as matC:
    for k in matC:
        list_C.append(list(map(float, k.split())))

#Condition for inverse exists or not
det=determinant_calc(list_C)
if det==0:
    print("Inverse doesn't exist !")
else:
    print("Determinant is ", det,",Which is not equal to zero, Hence inverse exists!")
#using the Gauss jordan function
a,b=Gauss_jordan(list_C)
print("\nThe Inverse of the given matrix is A[-1]:")

# Print the inverse matrix in readable form #SARBAJIT MAZUMDAR_1911147
for i in range(len(b)):
    for j in range(len(b)):
        print("%.2f"%b[i][j],end =' ') #each element of the matrix is rounded upto 2 pl
aces of decimal
    print()


##VERIFICATION PART
print("\n Verification of A*A[-1] = I(Identity matrix)")

list_d=matrix_mul(list_C,b)
for i in range(len(list_d)):
    for j in range(len(list_d)):
        print("%.2f"%list_d[i][j],end =' ') #each element of the matrix is rounded upto
2 places of decimal
    print()

print("Hence Inverse exists and verified!")
```

```
Determinant is  120.0 ,Which is not equal to zero, Hence inverse exists!

The Inverse of the given matrix is A[-1]:
0.20 0.00 0.00 0.00
0.00 0.25 0.00 0.00
0.00 0.00 0.33 0.00
0.00 0.00 0.00 0.50

 Verification of A*A[-1] = I(Identity matrix)
1.00 0.00 0.00 0.00
0.00 1.00 0.00 0.00
0.00 0.00 1.00 0.00
0.00 0.00 0.00 1.00
Hence Inverse exists and verified!
```

Question 3:

Solve the following set of linear equation using LU decomposition

$$3x_1 - 7x_2 - 2x_3 + 2x_4 = 0$$
$$-3x_1 + 5x_2 + x_3 = 5$$
$$6x_1 - 4x_2 - 5x_4 = 7$$
$$-9x_1 + 5x_2 - 5x_3 + 12x_4 = 11$$

In [54]:

```
#calling the augmented [A|b] matrix in readable form
list_C=[]
with open("matrix2.txt") as matC:
    for k in matC:
        list_C.append(list(map(float, k.split())))


#Printing the solutions of crout method #sarbajit mazumdar,1911147
x1 = linear_solver_crout(list_C)
if x1!=None:
    print("x_1 =", x1[0])
    print("x_2 =", x1[1])
    print("x_3 =", x1[2])
    print("x_4 =", x1[3])

#Verification
print("\nverification")
val=3*x1[0]-7*x1[1]-2*x1[2]+2*x1[3]
print("\nThe RHS of equation 1  IS coming out to be",val)
print("Hence the solutions are verified!")
```

```
The solutions of the system of linear equations by Crout's method is
x_1 = 3.0
x_2 = 4.0
x_3 = -6.0
x_4 = -1.0

verification

The RHS of equation 1  IS coming out to be -9.0
Hence the solutions are verified!
```

Question 4:

Find a real root of the following equation correct upto four decimal places in the interval $[0, 1]$ using both Midpoint and Regula-falsi method,

$$4e^{-x}\sin x - 1 = 0$$

Compare the convergence of the two methods.

COMMENTS

1. I have added one graph where I showed that for both methods(Bisection and Regula Falsi) $f(x_i)\ vs.\ i$ converges to zero after a certain number of iterations.
2. I have added one graph where I showed for both methods(Bisection and Regula Falsi) the a $|x_{i+1} - x_i|\ vs.\ i$ also converges to 0.
3. I have also added the tables for the both.

In [55]:

```
from My_Lib import *

import math
import matplotlib.pyplot as plt

# Function definition
def f(x):
    return 4*math.exp(-x)*math.sin(x)-1 #Given function #sarbajit mazumdar,1911147


root_val_1, x_error_1,f_x_i_1, iterations_1 = bisection_method(f, 0, 1, 10**(-5))   #ca
lling bisection function, braclet [0,1]
root_val_2, x_error_2,f_x_i_2, iterations_2 = regula_falsi_method(f, 0, 1, 10**(-5)) #c
alling Regul falsi function, bracket [0,1]



print("Root of the equation via Bisection Method is(upto 4 decimal places rounding): %.
4f"%root_val_1) #Printing root value for bisection
print("Root of the equation via Regula Falsi Method is(upto 4 decimal places rounding):
%.4f"%root_val_2)#Printing root value for Regula Falsi



################################################################################

#
# MAKING TABLES FOR BOTH THE METHODS
#



print ("\n------------Bisection Method--------------")
print("| |x_(i+1)-x_(i)\t|\ti\t|\t")
print ("-----------------------------------------")
for i in range (1,len(iterations_1)):
    print("|\t%.5f"%x_error_1[i],"\t|\t",iterations_1[i],"\t|")

print ("-----------------------------------------")

print ("------------Regula Falsi Method----------")
print("| |x_(i+1)-x_i\t|\t|\ti\t|\t")
print ("-----------------------------------------")
for i in range (1,len(iterations_2)):
    print("|\t%.11f"%x_error_2[i],"\t|\t",iterations_2[i],"\t|")

print ("-----------------------------------------")

##############################################################################3

#Plotting of Two methods:

plt.plot(iterations_1, f_x_i_1,'m-o', label="Bisection Method")
plt.plot(iterations_2, f_x_i_2,'r-o' ,label="Regular Falsi Method")

plt.title("Convergence of functional value for Bisection and Regula Falsi Method")
plt.xlabel("Iterations  $i$")
plt.ylabel("functional value ($f(x_i)$)")
plt.legend()
```

```
plt.grid()
plt.show()

# Removing zeroes from the lists
iterations_2.remove(0)
iterations_1.remove(0)
x_error_2.remove(0)
x_error_1.remove(0)

# plotting convergence of roots

plt.plot(iterations_1, x_error_1,'m-o', label="Bisection Method")
plt.plot(iterations_2, x_error_2,'r-o' ,label="Regular Falsi Method")

plt.title("Convergence of root for Bisection and Regula Falsi Method")
plt.xlabel("Iterations  $i$")
plt.ylabel("Absolute convergence $|x_{i+1}-x_{i}|$")
plt.legend()
plt.grid()
plt.show()
```

```
Root of the equation via Bisection Method is(upto 4 decimal places roundin
g): 0.3706
Root of the equation via Regula Falsi Method is(upto 4 decimal places roun
ding): 0.3706

------------Bisection Method--------------
| |x_(i+1)-x_(i)        |       i       |
-----------------------------------------
|       0.25000         |       1       |
|       0.12500         |       2       |
|       0.06250         |       3       |
|       0.03125         |       4       |
|       0.01562         |       5       |
|       0.00781         |       6       |
|       0.00391         |       7       |
|       0.00195         |       8       |
|       0.00098         |       9       |
|       0.00049         |       10      |
|       0.00024         |       11      |
|       0.00012         |       12      |
|       0.00006         |       13      |
|       0.00003         |       14      |
|       0.00002         |       15      |
|       0.00001         |       16      |
-----------------------------------------
------------Regula Falsi Method----------
| |x_(i+1)-x_i  |       |       i       |
-----------------------------------------
|       0.18104871698   |       1       |
|       0.12669555641   |       2       |
|       0.07005839543   |       3       |
|       0.03347011363   |       4       |
|       0.01482812595   |       5       |
|       0.00634439186   |       6       |
|       0.00267360110   |       7       |
|       0.00111943680   |       8       |
|       0.00046743846   |       9       |
|       0.00019496498   |       10      |
|       0.00008127990   |       11      |
|       0.00003387848   |       12      |
|       0.00001411982   |       13      |
|       0.00000588463   |       14      |
|       0.00000245247   |       15      |
|       0.00000102208   |       16      |
|       0.00000042596   |       17      |
|       0.00000017752   |       18      |
|       0.00000007398   |       19      |
|       0.00000003083   |       20      |
|       0.00000001285   |       21      |
|       0.00000000536   |       22      |
|       0.00000000223   |       23      |
|       0.00000000093   |       24      |
|       0.00000000039   |       25      |
|       0.00000000016   |       26      |
|       0.00000000007   |       27      |
|       0.00000000003   |       28      |
|       0.00000000001   |       29      |
|       0.00000000000   |       30      |
|       0.00000000000   |       31      |
|       0.00000000000   |       32      |
|       0.00000000000   |       33      |
```

```
|        0.00000000000    |         34        |
|        0.00000000000    |         35        |
|        0.00000000000    |         36        |
|        0.00000000000    |         37        |
|        0.00000000000    |         38        |
|        0.00000000000    |         39        |
|        0.00000000000    |         40        |
|        0.00000000000    |         41        |
|        0.00000000000    |         42        |
|        0.00000000000    |         43        |
-----------------------------------------------
```