|  |  |  |  |
| --- | --- | --- | --- |
| **paf_kiet_logo** | **COLLEGE OF COMPUTING AND INFORMATION SCIENCES** | | |
| **Final Assessment of Lab Exam (Summer 2021 Semester)** | | |
| **Class Id** | 107285 | **Course Title** | Numerical Computing LAB |
| **Program** | BSCS | **Campus / Shift** | MAIN MORNING |
| **Date** | 16-07-2021 | **Total Marks** | 20 |
| **Duration** | 2.5 hours | **Faculty Name** | Aziz Mehmoud Farooqi |
| **Student Id** | 10619 | **Student Name** | Muhammad Umar Khan |
| **Code** | A1 |  |  |

**Instructions:**

* Fill out your Student ID and Student Name in above header.
* Do not remove or change any part question paper.
* Write down your answers with title “Answer for Question# 00”.
* Handwritten text or image should be on A4 size page with clear visibility of contents.
* In case of CHEATING, COPIED material or any unfair means would result in negative marking or ZERO.
* **Caution:** Duration to perform Final Assessment is **02 hours & 30 mins only**. **If you failed to upload answer sheet on LMS (in PDF format) within 2.5 hours limit, you would be considered as ABSENT/FAILED.**



**Instructions:** Attempt all two questions.

**[10 Marks]**

**Question 1:**

Write a Dynamic Python code for solving any set of equations problem (3 unknown variables or 4 etc.) using **Gaussian Elimination Method**.

You have to define two returnable functions named ***funForwardEDynamically***(a) and ***funBackSubDynamically***(a) that takes only one array matrix and returns an another array.

After creating the functions, first you will call the ***funForwardEDynamically*** then ***funBackSubDynamically*** in the main function...

Solve the following two set of equations by Gaussian elimination one by one:

*3 variables & 3 equations:*

-x -5y -5z = 2

4x -5y +4z = 19

x +5y -z = -20

# Gauss Elimination

import numpy as np

def funDynamic(a):

n = len(a)

c = np.zeros(n)

for i in range(n):

for j in range(i + 1, n):

ratio = a[j][i]/a[i][i]

for k in range(n + 1):

a[j][k] = a[j][k] - ((ratio)\*a[i][k])

return a

def BackDub(a):

n = len(a) - 1

kvalues = np.zeros(n+1)

for i in range(n, -1, -1):

cons = a[i][n + 1]

if (i == n):

kvalues[i] = cons / a[i][i]

else:

mid = 0

for j in range(n, -1, -1):

if (a[i][j] != 0. and a[i][j] != a[j][j]):

a\_kvalues = kvalues[j]

a\_cellvalues = a[i][j]\*(-1)

mid = mid + (a\_kvalues \* a\_cellvalues)

kvalues[i] = (cons + mid) / a[i][i]

return kvalues

import numpy as np

def funForwardEDynamically(a):

a = np.array([[-1,-5,-5,2],[4,-5,4,19],[1,5,-1,-20]])

return a

a = np.array([[-1,-5,-5,2],[4,-5,4,19],[1,5,-1,-20]])

print(a)

n=3

for i in range(n):

print(i)

for j in range (i+1,n):

ratio = a[j][i]/a[i][i]

print(ratio)

for k in range (n+1):

a[j][k] = a[j][k]-((ratio)\*a[i][k])

print(a[j])

a3 = a[2][3] / a[2][2]

a2 = ((a[1][3]) + (-a[1][2]\*a3)) / (a[1][1])

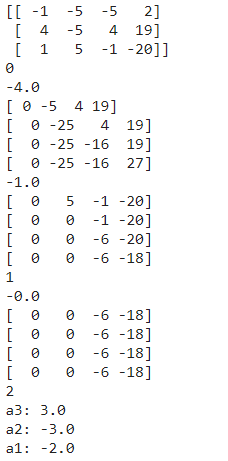
a1 = ((a[0][3]) + (-a[0][2]\*a3) + (-a[0][1]\*a2))/ a[0][0]

print('a3:', a3)

print('a2:', a2)

print('a1:', a1)

Output:



import numpy as np

eq3 = np.array([[-1,-5,-5,2],[4,-5,4,19],[1,5,-1,-20]])

forword = funDynamic(eq3)

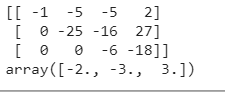
print(forword)

backword = BackDub(forword)

res = backword

res

Output :



*4 variables & 4 equations:*

x + z +2w = 6

y -2z = -3

x +2y - z = -2

2x + y +3z -2w = 0

# Gauss Elimination

import numpy as np

def funDynamic(a):

n = len(a)

c = np.zeros(n)

for i in range(n):

for j in range(i + 1, n):

ratio = a[j][i]/a[i][i]

for k in range(n + 1):

a[j][k] = a[j][k] - ((ratio)\*a[i][k])

return a

def BackDub(a):

n = len(a) - 1

kvalues = np.zeros(n+1)

for i in range(n, -1, -1):

cons = a[i][n + 1]

if (i == n):

kvalues[i] = cons / a[i][i]

else:

mid = 0

for j in range(n, -1, -1):

if (a[i][j] != 0. and a[i][j] != a[j][j]):

a\_kvalues = kvalues[j]

a\_cellvalues = a[i][j]\*(-1)

mid = mid + (a\_kvalues \* a\_cellvalues)

kvalues[i] = (cons + mid) / a[i][i]

return kvalues

import numpy as np

def funForwardEDynamically(a):

a = np.array([[1,0,1,2,6],[0,1,-2,0,-3],[1,2,-1,0,-2],[2,1,3,-2,0]])

return a

a = np.array([[1,0,1,2,6],[0,1,-2,0,-3],[1,2,-1,0,-2],[2,1,3,-2,0]])

print(a)

n=3

for i in range(n):

print(i)

for j in range (i+1,n):

ratio = a[j][i]/a[i][i]

print(ratio)

for k in range (n+1):

a[j][k] = a[j][k]-((ratio)\*a[i][k])

print(a[j])

a3 = a[2][3] / a[2][2]

a2 = ((a[1][3]) + (-a[1][2]\*a3)) / (a[1][1])

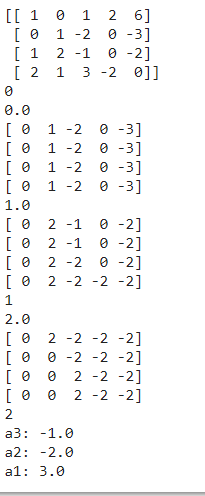
a1 = ((a[0][3]) + (-a[0][2]\*a3) + (-a[0][1]\*a2))/ a[0][0]

print('a3:', a3)

print('a2:', a2)

print('a1:', a1)

Output :



import numpy as np

eq3 = np.array([[1,0,1,2,6],[0,1,-2,0,-3],[1,2,-1,0,-2],[2,1,3,-2,0]])

forword = funDynamic(eq3)

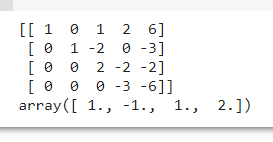
print(forword)

backword = BackDub(forword)

res = backword

res

Output:



**[10 Marks]**

**Question 2:**

Write a Static Python code for solving the particular set of equation problem (4 unknown variables and 4 equations) using **Jacobi Method**.

Solve the following set of equations by Jacobi:

*4 variables & 4 equations:*

10x - y + 2z = 6

- x +11y - z +3w = 25

2x - y +10z - w = -11

+ 3y - z +8w = 15

import numpy as np

import pandas as pd

DATAFRAME = pd.DataFrame(columns=['x','y','z'])

c = np.array([[10,-1,2,0,6],[-1,11,-1,3,25],[2,-1,10,-1,-11],[0,3,-1,8,15]])

x = np.ones(3)

x = np.zeros(len(c))

xcop = np.zeros(len(c))

for k in range(0,20):

for i in range(0,len(c)):

y = 0

for j in range(0,len(c)):

if i != j:

y = y + (c[i][j] \* x[j])

y = c[i][len(c)] - y

xcop[i] = y / c[i][i]

for i in range(0,len(x)):

x[i] = xcop[i]

DATAFRAME.loc[k+1] = [x[0],x[1],x[2]]

round(DATAFRAME,4)

Output :

