MFDS: Assignment 1 Submission

Group Number: 323 and members are:

Name	ID No.	Email ID
RAGESH HAJELA	2021SC04877	2021SC04877@wilp.bits-pilani.ac.in
THAKARE KEDAR SATISHRAO	2021SC04878	2021SC04878@wilp.bits-pilani.ac.in
D V S RAMA KRISHNA	2021SC04879	2021SC04879@wilp.bits-pilani.ac.in

Q1) Gauss Seidel and Gauss Jacobi

Q1-i) Problem Statement

Part a) Write a function to check whether a given square matrix is diagonally dominant.

Part b) Test the function on a randomly generated 4 × 4 matrix.

Deliverable(s) : Code that performs the check and the results obtained for the matrix

Q1-i) Solution

Part a) Write a function to check whether a given square matrix is diagonally dominant.

```
def DiagonallyDominantCheck(a, b):
    Count = 0
    for i in range(0, b):
        Totalexceptdia = 0
        for j in range(0, b):
            if i != j:
                Totalexceptdia = Totalexceptdia + abs(a[i][j])
        if abs(a[i][i]) > Totalexceptdia:
            Count = Count + 1
    return Count
```

Part b) Generate 4 × 4 matrix and test the function.

```
import numpy as np

n = 4
A = np.random.randint(60, size=(n, n))
print("Randomly Generated Matrix : ", "\n", "\n", A, "\n")

K = DiagonallyDominantCheck(A, n)
if K == n:
    print("Randomly Generated", n, "X", n, "Matrix is Diagonally Dominant")
else:
    print("Randomly Generated", n, "X", n, "Matrix is not Diagonally Dominant")
```

Output:

```
Randomly Generated Matrix :

 [[ 6 54 41 27]
 [40 45 56 53]
 [41 39 24  0]
 [21 27 24 38]]

Randomly Generated 4 X 4 Matrix is not Diagonally Dominant
```

Q1-ii) Problem Statement

- Write a function to generate Gauss Seidel Iteration for a given square matrix.
- The function should also return the values of 1, ∞ and Frobenius norms of iteration matrix.
- Generate a random 4 ×4 matrix.
- Report the Iteration matrix and its norm values returned by the function alongwith the input matrix.
- Deliverable(s): The input matrix, iteration matrix and the three norms obtained

Q1-ii) Solution

```
import math
import numpy as np

n = 4  # No Rows/ Columns of Random Matrix
a = np.random.randint(60, size=(n, n))
b = np.random.randint(60, size=(n, 1))

print("Randomly Generated Matrix A: ", "\n", "\n", a, "\n")
print("Randomly Generated Matrix B: ", "\n", "\n", b, "\n")

def GuassSeidel(a, x, b, N):
    n = len(a)
    S = 0
    for k in range(0, N):
        for j in range(0, n):
            d = b[j]
            S = S + b[j] * b[j]  # Sum of squares of all elements - For Frobenius Norm
Calculations
            for i in range(0, n):
                if j != i:
                    d = d - a[j][i] * x[i]
            x[j] = d / a[j][j]

    # Calculation of Vector 1 Norm
    V1 = sum(abs(b))  # No of Columns = 1 hence Vector 1 Norm will be just sum of
absolute values of the Matrix

    # Calculation of Vector Infinity Norm
    Vinf = max(abs(b))  # No of Columns = 1 hence Vector infinity Norm will be just
element with max absolute value

    # Calculation of Vector Frobenius Norm
    Vfro = math.sqrt(S)  # No of Columns = 1 hence Vector 1 Norm will be just sum of
absolute values of the Matrix

    return V1, Vinf, Vfro


# set input variables
x = []
x = np.zeros_like(b, dtype=np.double)
print(f'Here is x: \n{x}')
N = 100  # No of iterations to be performed

# call GuassSeidel function and capture return norm values
V1, Vinf, Vfro = GuassSeidel(a, x, b, N)

print("\n", "Value of Vector 1 Norm : ", "\n", "\n", V1, "\n")
print("\n", "Value of Vector Infinity Norm : ", "\n", "\n", Vinf, "\n")
print("\n", "Value of Vector Frobenius Norm : ", "\n", "\n", Vfro, "\n")
```

Output:

```
Randomly Generated Matrix A:

 [[23 21 51 16]
 [17 42 29 37]
 [28 16  0 57]
 [49 59 12 34]]

Randomly Generated Matrix B:

 [[43]
 [32]
 [26]
 [28]]

Here is x:
[[0.]
 [0.]
 [0.]
 [0.]]

 Value of Vector 1 Norm :

 [129]

 Value of Vector Infinity Norm :

 [43]

 Value of Vector Frobenius Norm :

 658.2552696332936
```

Q1-iii) Problem Statement

Repeat part (ii) for the Gauss Jacobi iteration

Deliverable(s): The input matrix, iteration matrix and the three norms obtained.

Q1-iii) Solution

```
import numpy as np
import math

n = 4  # No Rows/ Columns of Random Matrix
N = 100  # No of iterations to be performed
a = np.random.randint(60, size=(n, n))
b = np.random.randint(60, size=(n, 1))

print("Randomly Generated Matrix A: ", "\n", "\n", a, "\n")
print("Randomly Generated Matrix B: ", "\n", "\n", b, "\n")


def jacobi(A, x, b, N=99):
    n = len(A)
    V1 = 0
    Vinf = 0
    Vfro = 0
    S = 0
    for k in range(0, N):
```

```python
        for j in range(0, n):
            d = b[j]

            S = S + b[j] * b[j]  # S
    if x is None:
        x = zeros(len(A[0]))
    # Vector from diagonal elements of A and subtract from A
    D = np.diag(A)
    R = A - np.diagflat(D)
    c = []
    count = 0
    # Iterate N times
    for i in range(N - 1):
        x = x.astype('float64')
        x = (b - np.dot(R, x)) / D
        c = x[:, 0].reshape(4, 1)

    # Calculation of Vector 1 Norm
    V1 = sum(abs(b))  # No of Columns = 1 hence Vector 1 Norm will be just sum of
absolute values of the Matrix
    print("\n", "Value of Vector 1 Norm : ", "\n", "\n", V1, "\n")

    # Calculation of Vector Infinity Norm
    Vinf = max(abs(b))  # No of Columns = 1 hence Vector infinity Norm will be just
element with max absolute value
    print("\n", "Value of Vector Infinity Norm : ", "\n", "\n", Vinf, "\n")

    # Calculation of Vector Frobenius Norm
    Vfro = math.sqrt(S)  # No of Columns = 1 hence Vector 1 Norm will be just sum of
absolute values of the Matrix
    print("\n", "Value of Vector Frobenius Norm : ", "\n", "\n", Vfro, "\n")
    return c


x = []
x = np.zeros_like(b, dtype=np.double)
print("\n", "Initial value of x: ", "\n", "\n", x, "\n")

x = jacobi(a, x, b, N)
```

Output:

```
Randomly Generated Matrix A:

 [[32  1 58 41]
 [16 32  6 52]
 [18 29 39 38]
 [31 47 43 51]]

Randomly Generated Matrix B:

 [[ 6]
 [57]
 [34]
 [15]]


 Initial value of x:

 [[0.]
 [0.]
 [0.]
```

```
[0.]]


Value of Vector 1 Norm :

[112]


Value of Vector Infinity Norm :

[57]


Value of Vector Frobenius Norm :

683.0812543175226
```

Q1-iv) Problem Statement

- Write a function that perform Gauss Seidel iterations. Generate a random 4 × 4 matrix A and generate a random b ∈ R4. Report the results of passing this matrix to function written in
    - (i). Solve linear system Ax = b by using function in
    - (ii). Generate a plot of ‖xk+1 − xk‖2 for first 100 iterations.
    - Does it converge ? or is it diverging? Specify your observation.
- Take a screenshot of plot and paste it in the assignment document.
- Deliverable(s): The input matrix and the vector, the 10 successive iterates and the plot

Q1-iv) Solution

```
import numpy as np
import math

n = 4
N = 100

A = np.random.randint(60, size=(n, n))
b = np.random.randint(60, size=(n, 1))
print("Randomly Generated Matrix : ", "\n", "\n", A, "\n")


# Function for checking Diagonally Dominant Matrix - Part a
def DiagonallyDominantCheck(a, b):
    Count = 0
    for i in range(0, b):
        Totalexceptdia = 0

        for j in range(0, b):
            if (i != j):
                Totalexceptdia = Totalexceptdia + abs(a[i][j])

        if (abs(a[i][i]) > Totalexceptdia):
            Count = Count + 1

    return Count


K = (DiagonallyDominantCheck(A, n))
```

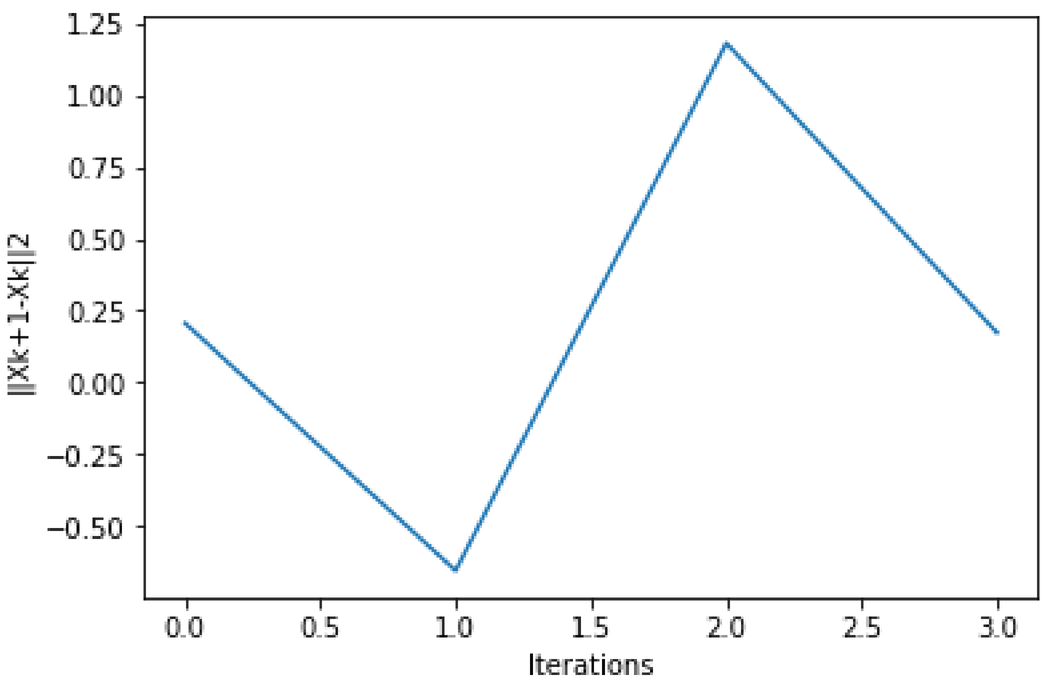
```python
if (K == n):
    print("Randomly Generated ", n, "X", n, " Matrix is Diagonally Dominant")
else:
    print("Randomly Generated ", n, "X", n, " Matrix is not Diagonally Dominant", "\n")

print("Randomly Generated Matrix B: ", "\n", "\n", b, "\n")


def GuassSeidel(a, x, b, N):
    n = len(a)
    V1 = 0
    Vinf = 0
    Vfro = 0
    S = 0
    count = 0
    for k in range(0, N):

        for j in range(0, n):

            d = b[j]

            S = S + b[j] * b[j]  # Sum of squares of all elements - For Frobenius Norm
Calculations

            for i in range(0, n):

                if (j != i):
                    d = d - a[j][i] * x[i]

            x[j] = d / a[j][j]

            if count <= 10:
                print("\n", "The value of x after ", count, " iterations", "\n", x, "\n")
            count += 1

    # Calculation of Vector 1 Norm
    V1 = sum(abs(b))  # No of Columns = 1 hence Vector 1 Norm will be just sum of
absolute values of the Matrix
    print("\n", "Value of Vector 1 Norm : ", "\n", "\n", V1, "\n")

    # Calculation of Vector Infinity Norm
    Vinf = max(abs(b))  # No of Columns = 1 hence Vector infinity Norm will be just
element with max absolute value
    print("\n", "Value of Vector Infinity Norm : ", "\n", "\n", Vinf, "\n")

    # Calculation of Vector Frobenius Norm
    Vfro = math.sqrt(S)  # No of Columns = 1 hence Vector 1 Norm will be just sum of
absolute values of the Matrix
    print("\n", "Value of Vector Frobenius Norm : ", "\n", "\n", Vfro, "\n")

    return x

x = []
x = np.zeros_like(b, dtype=np.double)
print("\n", "Initial value of x: ", "\n", "\n", x, "\n")

x = GuassSeidel(A, x, b, N)

import matplotlib.pyplot as plt
plt.plot(x)
plt.xlabel('Iterations')
plt.ylabel('||Xk+1-Xk||2')
plt.show()
```

Output:

```
Randomly Generated Matrix :

 [[16 17 37 24]
 [31 10  5 13]
 [35 46 54 24]
 [33  0  1 52]]

Randomly Generated  4 X 4  Matrix is not Diagonally Dominant

Randomly Generated Matrix B:

 [[40]
 [ 8]
 [45]
 [17]]


 Initial value of x:

 [[0.]
 [0.]
 [0.]
 [0.]]


 The value of x after  0  iterations
 [[2.5]
 [0. ]
 [0. ]
 [0. ]]


 The value of x after  1  iterations
 [[ 2.5 ]
 [-6.95]
 [ 0.  ]
 [ 0.  ]]


 The value of x after  2  iterations
 [[ 2.5       ]
 [-6.95      ]
 [ 5.13333333]
 [ 0.        ]]


 The value of x after  3  iterations
 [[ 2.5       ]
```

```
[-6.95      ]
[ 5.13333333]
[-1.35833333]]


The value of x after  4  iterations
[[ 0.05104167]
[-6.95      ]
[ 5.13333333]
[-1.35833333]]


The value of x after  5  iterations
[[ 0.05104167]
[-0.1590625 ]
[ 5.13333333]
[-1.35833333]]


The value of x after  6  iterations
[[ 0.05104167]
[-0.1590625 ]
[ 1.53945216]
[-1.35833333]]


The value of x after  7  iterations
[[ 0.05104167]
[-0.1590625 ]
[ 1.53945216]
[ 0.2649264 ]]


The value of x after  8  iterations
[[-1.28836882]
[-0.1590625 ]
[ 1.53945216]
[ 0.2649264 ]]


The value of x after  9  iterations
[[-1.28836882]
[ 3.67981293]
[ 1.53945216]
[ 0.2649264 ]]


The value of x after  10  iterations
[[-1.28836882]
```

```
[ 3.67981293]
[-1.58401333]
[ 0.2649264 ]]


Value of Vector 1 Norm :

[110]


Value of Vector Infinity Norm :

[45]


Value of Vector Frobenius Norm :

630.7138812488591
```



Observations:

Plotted graph shows neither convergence, nor divergence. This is because the matrix A does not meet the criteria for the convergence or the divergence as it is randomly generated. With every run, graph shows a different plot due to randomly generated input matrices.

Q1-v) Problem Statement

- Repeat part (iv) for the Gauss Jacobi iteration
- Deliverable(s): The input matrix and the vector, the 10 successive iterates and the plot

Q1-v) Solution

```
import numpy as np
import math

n = 4
N = 100
A = np.random.randint(60, size=(n, n))
b = np.random.randint(60, size=(n, 1))
print("Randomly Generated Matrix : ", "\n", "\n", A, "\n")


# Function for checking Diagonally Dominant Matrix - Part a

def DiagonallyDominantCheck(a, b):
    Count = 0
    for i in range(0, b):
        Totalexceptdia = 0

        for j in range(0, b):
            if (i != j):
                Totalexceptdia = Totalexceptdia + abs(a[i][j])

        if (abs(a[i][i]) > Totalexceptdia):
            Count = Count + 1

    return Count


K = (DiagonallyDominantCheck(A, n))

if (K == n):
    print("Randomly Generated ", n, "X", n, " Matrix is Diagonally Dominant")
else:
    print("Randomly Generated ", n, "X", n, " Matrix is not Diagonally Dominant", "\n")

print("Randomly Generated Matrix B: ", "\n", "\n", b, "\n")


def jacobi(A, x, b, N=99):
    n = len(A)
    V1 = 0
    Vinf = 0
    Vfro = 0
    S = 0
    for k in range(0, N):

        for j in range(0, n):
            d = b[j]

            S = S + b[j] * b[j]  # S
    if x is None:
        x = zeros(len(A[0]))
    # Vector from diagonal elements of A and subtract from A
    D = np.diag(A)
    R = A - np.diagflat(D)
    c = []
```

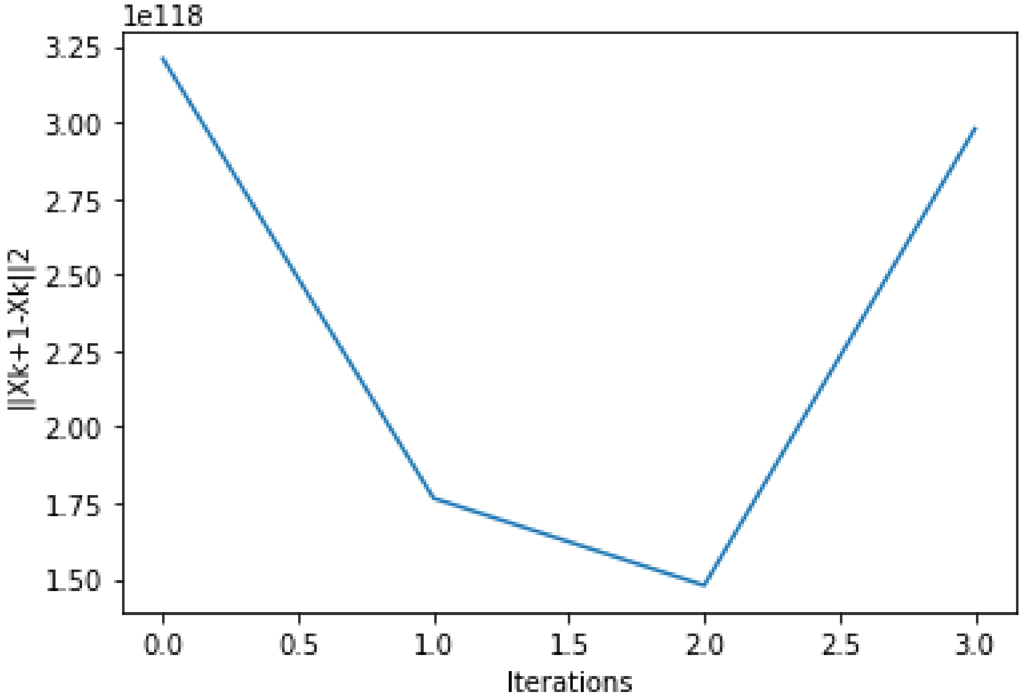
```
    count = 0
    # Iterate N times
    for i in range(N - 1):
        x = x.astype('float64')
        x = (b - np.dot(R, x)) / D
        c = x[:, 0].reshape(4, 1)
        if count < 10:
            print("\n", "The value of x after ", count + 1, " iterations", "\n", c, "\n")
        count += 1
    # Calculation of Vector 1 Norm
    V1 = sum(abs(b))  # No of Columns = 1 hence Vector 1 Norm will be just sum of
absolute values of the Matrix
    print("\n", "Value of Vector 1 Norm : ", "\n", "\n", V1, "\n")

    # Calculation of Vector Infinity Norm
    Vinf = max(abs(b))  # No of Columns = 1 hence Vector infinity Norm will be just
element with max absolute value
    print("\n", "Value of Vector Infinity Norm : ", "\n", "\n", Vinf, "\n")

    # Calculation of Vector Frobenius Norm
    Vfro = math.sqrt(S)  # No of Columns = 1 hence Vector 1 Norm will be just sum of
absolute values of the Matrix
    print("\n", "Value of Vector Frobenius Norm : ", "\n", "\n", Vfro, "\n")
    return c


x = []
x = np.zeros_like(b, dtype=np.double)
print("\n", "Initial value of x: ", "\n", "\n", x, "\n")

x = jacobi(A, x, b, N)

import matplotlib.pyplot as plt

plt.plot(x)
plt.xlabel('Iterations')
plt.ylabel('||Xk+1-Xk||2')
plt.show()
```

Output:

```
Randomly Generated Matrix :

 [[ 5 41 26 48]
 [38 23  8  2]
 [28  7  1  5]
 [42 37 24 17]]

Randomly Generated  4 X 4  Matrix is not Diagonally Dominant

Randomly Generated Matrix B:

 [[52]
 [30]
 [ 5]
 [58]]
```

```
Initial value of x:

[[0.]
[0.]
[0.]
[0.]]


The value of x after  1  iterations
[[10.4]
[ 6. ]
[ 1. ]
[11.6]]


The value of x after  2  iterations
[[-155.36]
[ -79.28]
[ -77.24]
[-124.96]]


The value of x after  3  iterations
[[2261.76 ]
[1360.304]
[1106.968]
[2274.048]]


The value of x after  4  iterations
[[-38731.1872]
[-19864.144 ]
[-16843.3296]
[-34366.88  ]]


The value of x after  5  iterations
[[580403.74272]
[335059.10208]
[279072.32992]
[553196.22016]]


The value of x after  6  iterations
[[-9509334.066176]
[-5078856.660608]
[-4272538.922304]
[-8694364.377856]]
```

```
The value of x after  7  iterations
[[1.47329735e+08]
[8.25847529e+07]
[6.90570355e+07]
[1.37970144e+08]]


The value of x after  8  iterations
[[-2.36080493e+09]
[-1.28538530e+09]
[-1.07863532e+09]
[-2.18017071e+09]]


The value of x after  9  iterations
[[3.70787019e+10]
[2.05400023e+10]
[1.72002177e+10]
[3.45200621e+10]]


The value of x after  10  iterations
[[-5.89261747e+11]
[-3.23126508e+11]
[-2.70916796e+11]
[-5.46018158e+11]]


Value of Vector 1 Norm :

[145]


Value of Vector Infinity Norm :

[58]


Value of Vector Frobenius Norm :

836.2415918859813
```



Observations:

Plotted graph shows neither convergence, nor divergence. This is because the matrix A does not meet the criteria for the convergence or the divergence as it is randomly generated. With every run, graph shows a different plot due to randomly generated input matrices.

Q2) LU Decomposition, Vector Space and LT

1

MFDS ASSIGNMENT – 01

GROUP NUMBER - 323

PROBLEM STATEMENT

Q2) LU Decomposition, Vector Spaces and LT.

i) Find the LU decomposition of the matrix $A = \begin{bmatrix} 1 & 0 & 1 \\ a & a & a \\ b & b & a \end{bmatrix}$ when it exists. For which real numbers a and b does it exist?

SOLUTION

Taking an example of an $n \times n$ matrix A, it can be reduced to its row echelon form U without any row interchanges by only using the elementary row operations

$$R_i \longrightarrow R_i - m_{ij} R_j \quad \text{for } j = 1, 2, \ldots, n-1, \text{ and}$$
$$i = j+1, j+2, \ldots, n.$$

We can define an $n \times n$ matrix $L = [l_{ij}]$ as follows.

$$l_{ij} = \begin{cases} m_{ij} & , \text{if } i > j \\ 1 & , \text{if } i = j \\ 0 & , \text{if } i < j \end{cases}$$

Using this theorem, we can row reduce the matrix A given in the problem, without interchanging the rows.

(2)

Here, $A = \begin{bmatrix} 1 & 0 & 1 \\ a & a & a \\ b & b & a \end{bmatrix}$

$$\Rightarrow \begin{bmatrix} 1 & 0 & 1 \\ 0 & a & 0 \\ 0 & b & a-b \end{bmatrix}$$ <u>Applied</u> $R_2 \leftarrow R_2 - (a) R_1$, $R_3 \leftarrow R_3 - (b) R_1$

$$\Rightarrow \begin{bmatrix} 1 & 0 & 1 \\ 0 & a & 0 \\ 0 & 0 & a-b \end{bmatrix}$$ <u>Applied</u> $R_3 \leftarrow R_3 - \left(\frac{b}{a}\right) R_2$

last step $R_3 \leftarrow R_3 - \left(\frac{b}{a}\right) R_2$ is only possible when $a \neq 0$. Because we need the pivot element to be non-zero.

So based on the elementary row operation we observe the following

Elementary row operation	From the Theorem
$R_2 \leftarrow R_2 - (a) R_1$	$m_{21} = a$
$R_3 \leftarrow R_3 - (b) R_1$	$m_{31} = b$
$R_3 \leftarrow R_3 - \left(\frac{b}{a}\right) R_2$	$m_{32} = \frac{b}{a}$

③

Therefore we can have the L matrix form as follows.

$$L = \begin{bmatrix} 1 & 0 & 0 \\ m_{21} & 1 & 0 \\ m_{31} & m_{32} & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ a & 1 & 0 \\ b & b/a & 1 \end{bmatrix}$$

Finally, we get the following. $\boxed{A = LU}$

where $L = \begin{bmatrix} 1 & 0 & 0 \\ a & 1 & 0 \\ b & b/a & 1 \end{bmatrix}$

and $U = \begin{bmatrix} 1 & 0 & 1 \\ 0 & a & 0 \\ 0 & 0 & a-b \end{bmatrix}$

And $A = LU$ will only be possible when $a \neq 0$

So, a can be any real number other than zero

ie. a non-zero real number

And b can be any real number.

Therefore, a can be any non-zero real number, and b can be any real number.

PROBLEM STATEMENT

Q2) ii) Find the dimension of the vector spanned by the vectors $\{[1,1,-2,0,1], [1,2,0,-4,1], [0,1,3,-3,2], [2,3,0,-2,0]\}$ and find a basis for the space.

SOLUTION

Lets say the vector input is 'A', create the matrix 'A' from input

$$A = \begin{bmatrix} 1 & 1 & -2 & 0 & 1 \\ 1 & 2 & 0 & -4 & 1 \\ 0 & 1 & 3 & -3 & 2 \\ 2 & 3 & 0 & -2 & 0 \end{bmatrix}$$

Applied

$$\Rightarrow \begin{bmatrix} 1 & 1 & -2 & 0 & 1 \\ 0 & 1 & 2 & -4 & 0 \\ 0 & 1 & 3 & -3 & 2 \\ 0 & 1 & 4 & -2 & -2 \end{bmatrix} \begin{matrix} \\ R_2 \leftarrow R_2 - R_1 \\ \\ R_4 \leftarrow R_4 - 2R_1 \end{matrix}$$

Applied

$$\Rightarrow \begin{bmatrix} 1 & 1 & -2 & 0 & 1 \\ 0 & 1 & 2 & -4 & 0 \\ 0 & 0 & 1 & 1 & 2 \\ 0 & 0 & 2 & 2 & -2 \end{bmatrix} \begin{matrix} \\ \\ R_3 \leftarrow R_3 - R_2 \\ R_4 \leftarrow R_4 - R_2 \end{matrix}$$

$$\Rightarrow \begin{bmatrix} 1 & 1 & -2 & 0 & 1 \\ 0 & 1 & 2 & -4 & 0 \\ 0 & 0 & 1 & 1 & 2 \\ 0 & 0 & 0 & 0 & -6 \end{bmatrix} \begin{array}{l} \text{Applied} \\ \\ \\ R_4 \leftarrow R_4 - 2R_3. \end{array}$$

We get four linearly independent vectors, and these vectors will span the vector space of A.

Therefore, the basis are

$$\boxed{\{[1,1,-2,0,1],\ [0,1,2,-4,0],\ [0,0,1,1,2],\ [0,0,0,0,-6]\}}$$

As, dimension of the vector space is the number of linearly independent vectors, so for 'A' vector space -

$$\boxed{\text{Dimension of the vector space} = 4}$$

⑥

PROBLEM STATEMENT

Q2) iii) Suppose that A is a matrix such that the complete solution to.

$$Ax = \begin{bmatrix} 1 \\ 4 \\ 1 \\ 1 \end{bmatrix}$$

is of the form:

$$x = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} + c \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix}, \quad c \in R$$

a) What can be said about the colums of matrix A?

b) Find the dimension of null space and rank of matrix A.

SOLUTION

As observed from the input.

$$A_{(4 \times m)} \times x_{(m \times 1)} = \begin{bmatrix} 1 \\ 4 \\ 1 \\ 1 \end{bmatrix}_{4 \times 1}$$

Here, dimension of A is $(4 \times m)$ and dimension of x is $(m \times 1)$.

From the composite solution

$$x = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} + c \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix}, c \in R.$$

So, dimension of x is (3×1).

7

So, dimension of $x = (m\times 1) = (3\times 1) \Rightarrow m=3$.

thus, dimension of $A = (4\times m) = (4\times 3)$

Therefore, $\boxed{\text{matrix } A \text{ has } 3 \text{ columns}}$

The complete solution for A is.

$$x = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} + c \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix}, \quad c \in R.$$

Here,

$$x = X_p + X_s$$

where X_p is particular solution

X_s is special solution, and

x is the complete solution

The number of special solution will give us the dimension of null space. So, $\boxed{\dim(\text{null space}) = 1}$

Applying Rank - Nullity Theorem, we have

$$\text{Rank}(A) + \text{Nullity}(A) = \text{Columns}(A).$$

As we already know that $\text{Nullity}(A) = 1$, and
$\text{columns}(A) = 3$

So finally, $\boxed{\text{Rank}(A) = 2}$

This solution answers regarding columns of matrix A, dimension of null space and rank of matrix A.