# Assignment 1

# DSECL ZC416 - Mathematical Foundations for Data Science

**Group Name:**

Group132

**Contribution Table:**

| Sl. No. | Name (as appears in Canvas) | ID NO |
|---|---|---|
| 1 | Prince Kumar | 2021SC04622 |
| 2 | Selvadurai M | 2021SC04620 |
| 3 | Chhattar Singh | 2021SC04621 |

**Q1) Gauss Seidel and Gauss Jacobi (5 marks)**

i Write a function to check whether a given square matrix is diagonally dominant. Test the function on a randomly generated 4 ×4 matrix.

**Program**

```
import numpy as np

def isDDM(m, n) :

    for i in range(0, n) :
        sum = 0

        for j in range(0, n) :
            sum = sum + abs(m[i][j])

        sum = sum - abs(m[i][i])

        if (abs(m[i][i]) < sum) :
                return print (f"Matrix is not diagonally dominant.\n {m}")

    return print (f"Matrix is diagonally dominant.\n {m}")

n = 4
m = np.random.randint(10,size=(4,4))
isDDM(m, n)
```

**Result**

```
D:\BITS-M.Tech\Semester1\SEM-1 MFDS\Assignment>python 1.py
Matrix is not diagonally dominant.
 [[8 9 3 9]
 [6 5 5 6]
 [8 2 2 6]
 [5 1 5 8]]
```

ii 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 along with the input matrix.

**Program**

```
import math
import sys
from numpy.linalg import norm
import numpy as np
from numpy import inf

maxVal = math.sqrt(sys.float_info.max)/3
def seidel(a, x ,b):
    n = len(a)
    for j in range(0, n):
        d = b[j]
        for i in range(0, n):
            if(j != i):
                d = d - (a[j][i] * x[i])
        x[j] = d / a[j][j]
    return x

tolerance = 1e-10
a = np.array( [
    [4, 1, 2, 5],[3, 5, 1, 8],[1, 1, 3, 3],[4, 1, 2, 9]
], dtype='f')

b = np.array( [4,7,3,6] )
x = x = np.zeros_like(b, dtype=np.float)
print('A the given matrix')
print(a)
print('B')
print(b)
print(x)

for i in range(0, 1000):
    x_old = x.copy()
    execute = True
    for num in x_old:
        if (abs(num) > maxVal):
            print("Value is diverging")
            execute = False
```

```
            break;
    if( execute == False):
        break
    x = seidel(a, x, b)
    if np.linalg.norm(x - x_old, ord=np.inf) / np.linalg.norm(x, ord=np.inf) <
tolerance:
        break
    print(x)
    print('First Norm')
    print(np.linalg.norm(x, ord=1))
    print('Infinite Norm')
    infintie_norm_of_x = norm(x,inf)
    print(infintie_norm_of_x)
    def frobeniusNorm(mat):
        sumSq = 0
        for i in range(len(mat)):
            sumSq += pow(mat[i], 2)
        res = math.sqrt(sumSq)
        return round(res, 5)

    print('Frobenius Norm')
    frobenius_norm_of_x = frobeniusNorm(x)
    print(frobenius_norm_of_x)
    print("Final Gauss Seidel Iteration")
    print(x)
```

### Result

```
1.386363635525996
Infinite Norm
0.5000000002724518
Frobenius Norm
0.76314
Final Gauss Seidel Iteration
[0.10227273 0.47727273 0.30681818 0.5        ]
[0.10227273 0.47727273 0.30681818 0.5        ]


First Norm
1.386363635910867
Infinite Norm
0.5000000001133733
Frobenius Norm
0.76314
Final Gauss Seidel Iteration
[0.10227273 0.47727273 0.30681818 0.5        ]
[0.10227273 0.47727273 0.30681818 0.5        ]


First Norm
1.3863636361767486
Infinite Norm
0.5000000000296465
Frobenius Norm
0.76314
Final Gauss Seidel Iteration
[0.10227273 0.47727273 0.30681818 0.5        ]
[0.10227273 0.47727273 0.30681818 0.5        ]


First Norm
1.386363636315765
Infinite Norm
0.4999999999962303
Frobenius Norm
0.76314
Final Gauss Seidel Iteration
[0.10227273 0.47727273 0.30681818 0.5        ]
```

iii Repeat part (ii) for the Gauss Jacobi iteration

**Program**

```
import math
import sys
from numpy.linalg import norm
import numpy as np
from numpy import inf

maxVal = math.sqrt(sys.float_info.max)/3
def jacobi(A, b, x, n):

    D = np.diag(A)
    R = A - np.diagflat(D)

    for i in range(n):
        x = (b - np.dot(R,x))/ D
    return x

tolerance = 1e-10
a = np.array( [
    [4, 1, 2, 5],[3, 5, 1, 8],[1, 1, 3, 3],[4, 1, 2, 9]
], dtype='f')

b = np.array( [4,7,3,6] )
x = x = np.zeros_like(b, dtype=np.float)
n = len(a)
print('A the given matrix')
print(a)
print('B')
print(b)
print(x)

for i in range(0, 1000):
    x_old = x.copy()
    execute = True
    for num in x_old:
        if (abs(num) > maxVal):
            print("Value is diverging")
            execute = False
            break;
    if( execute == False):
        break
    x = jacobi(a, x, b,n)
```

```
    if np.linalg.norm(x - x_old, ord=np.inf) / np.linalg.norm(x, ord=np.inf) <
tolerance:
        break
    print(x)
    print("\n")
    print('First Norm')
    print(np.linalg.norm(x, ord=1))
    print('Infinite Norm')
    infintie_norm_of_x = norm(x,inf)
    print(infintie_norm_of_x)
    def frobeniusNorm(mat):
        sumSq = 0
        for i in range(len(mat)):
            sumSq += pow(mat[i], 2)
        res = math.sqrt(sumSq)
        return round(res, 5)

    print('Frobenius Norm')
    frobenius_norm_of_x = frobeniusNorm(x)
    print(frobenius_norm_of_x)
    print("Final Jacobi Iteration")
    print(x)
```

**Result**

```
[20.99956509 26.40910979 23.99356917  7.93572631]
[20.99956507 26.40910979 23.99356917  7.93572633]


First Norm
79.3379703480943
Infinite Norm
26.40910978685766
Frobenius Norm
42.15554
Final Jacobi Iteration
[20.99956507 26.40910979 23.99356917  7.93572633]
[20.99956506 26.40910978 23.99356917  7.93572634]


First Norm
79.3379703418717
Infinite Norm
26.409109783615254
Frobenius Norm
42.15554
Final Jacobi Iteration
[20.99956506 26.40910978 23.99356917  7.93572634]
[20.99956505 26.40910978 23.99356917  7.93572634]


First Norm
79.3379703391371
Infinite Norm
26.409109782192168
Frobenius Norm
42.15554
Final Jacobi Iteration
[20.99956505 26.40910978 23.99356917  7.93572634]
```

iv 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.

## Program

```
import numpy as np

import numpy.linalg as la

import time


def GaussSeidel(A,b):

       # dimension of the non-singular matrix

       n = len(A)


       # def. max iteration and criterions

       Kmax = 100;

       tol  = 1.0e-4;

       btol = la.norm(b)*tol


       x0   = np.zeros(n)

       k    = 0 ;

       stop = False

       x1   = np.empty(n)


       while not(stop) and k < Kmax:

           print ("begin while with k =", k)

           x1 = np.zeros(n)

           for i in range(n):          # rows of A
```

```
                x1[i] = ( b[i] - np.dot(A[i,0:i], x1[0:i]) - np.dot(A[i,i+1:n], x0[i+1:n]) ) / A[i,i]
                print("x1 =", x1)

            r    = b - np.dot(A,x1)
            stop = (la.norm(r) < btol) and (la.norm(x1-x0) < tol)
            print("end of for i ")
            print("x0 =", x0)
            print("btol = %e ; la.norm(r) = %e ; tol = %e ; la.norm(x1-x0) = %e; stop = %s " % (btol, la.norm(r), tol, la.norm(x1-x0), stop))
            x0   = x1
            print("x0 =", x0, end='')
            print("end of current while ")
            k    = k + 1

        if not(stop):   # or if k >= Kmax
            print("\n")
            print('Not converges in %d iterations' % Kmax)

        return x1, k

A = np.array( [
    [  3, -0.1, -0.2, 0.7],
    [0.1,    7, -0.3, 0.8],
    [0.3, -0.2,   10, 0.9],
    [0.5, -0.4,   7, 0.4]
], dtype='f')

b = np.array( [7.85, -19.3, 71.4, 65.3] )
```

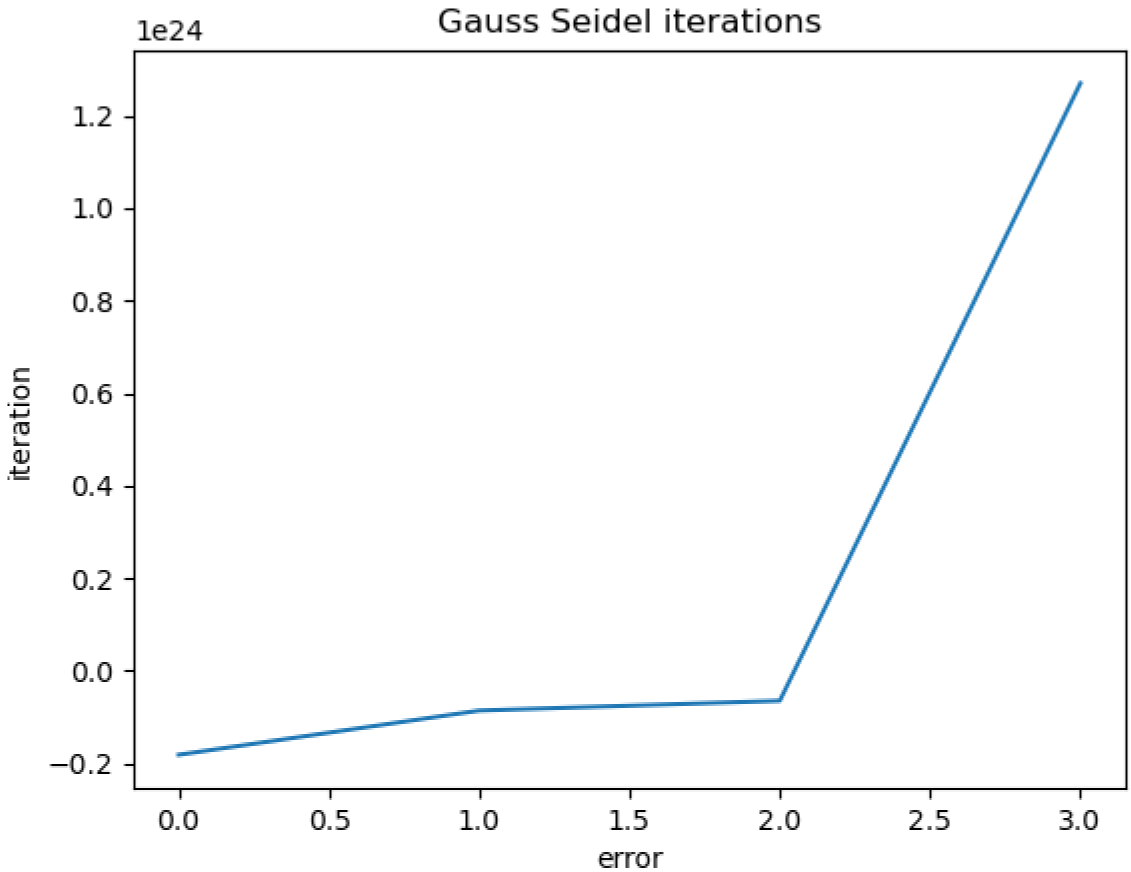
```python
xsol = la.solve(A,b)


start    = time.time()

x, k     = GaussSeidel(A,b)

ending   = time.time()

duration = ending-start

err      = la.norm(xsol-x)

print('Iter.=%d  duration=%f  err=%e' % (k,duration,err))


import matplotlib.pyplot as plt


plt.plot(x)

plt.xlabel('error')

plt.ylabel('iteration')

plt.title('Gauss Seidel iterations')

plt.show()
```

## Result

```python
A = np.array( [

    [  3, -0.1, -0.2, 0.7],

    [0.1,    7, -0.3, 0.8],

    [0.3, -0.2,   10, 0.9],

    [0.5, -0.4,   7, 0.4]

], dtype='f')


b = np.array( [7.85, -19.3, 71.4, 65.3] )
```

```
x1 = [-3.85911614e+22 -1.82562444e+22  0.00000000e+00  0.00000000e+00]
x1 = [-3.85911614e+22 -1.82562444e+22 -1.37414154e+22  0.00000000e+00]
x1 = [-3.85911614e+22 -1.82562444e+22 -1.37414154e+22  2.70457472e+23]
end of for i
x0 = [-2.30426424e+22 -1.09007373e+22 -8.20494925e+21  1.61489175e+23]
btol = 9.897557e-03 ; la.norm(r) = 1.536639e+23 ; tol = 1.000000e-04 ; la.norm(x1-x0) = 1.104563e+23; stop = False
x0 = [-3.85911614e+22 -1.82562444e+22 -1.37414154e+22  2.70457472e+23]end of current while
begin while with k = 97
x1 = [-6.46313783e+22  0.00000000e+00  0.00000000e+00  0.00000000e+00]
x1 = [-6.46313783e+22 -3.05750383e+22  0.00000000e+00  0.00000000e+00]
x1 = [-6.46313783e+22 -3.05750383e+22 -2.30137312e+22  0.00000000e+00]
x1 = [-6.46313783e+22 -3.05750383e+22 -2.30137312e+22  4.52954474e+23]
end of for i
x0 = [-3.85911614e+22 -1.82562444e+22 -1.37414154e+22  2.70457472e+23]
btol = 9.897557e-03 ; la.norm(r) = 2.573520e+23 ; tol = 1.000000e-04 ; la.norm(x1-x0) = 1.849891e+23; stop = False
x0 = [-6.46313783e+22 -3.05750383e+22 -2.30137312e+22  4.52954474e+23]end of current while
begin while with k = 98
x1 = [-1.08242792e+23  0.00000000e+00  0.00000000e+00  0.00000000e+00]
x1 = [-1.08242792e+23 -5.12062035e+22  0.00000000e+00  0.00000000e+00]
x1 = [-1.08242792e+23 -5.12062035e+22 -3.85427417e+22  0.00000000e+00]
x1 = [-1.08242792e+23 -5.12062035e+22 -3.85427417e+22  7.58595255e+23]
end of for i
x0 = [-6.46313783e+22 -3.05750383e+22 -2.30137312e+22  4.52954474e+23]
btol = 9.897557e-03 ; la.norm(r) = 4.310058e+23 ; tol = 1.000000e-04 ; la.norm(x1-x0) = 3.098145e+23; stop = False
x0 = [-1.08242792e+23 -5.12062035e+22 -3.85427417e+22  7.58595255e+23]end of current while
begin while with k = 99
x1 = [-1.81281946e+23  0.00000000e+00  0.00000000e+00  0.00000000e+00]
x1 = [-1.81281946e+23 -8.57586916e+22  0.00000000e+00  0.00000000e+00]
x1 = [-1.81281946e+23 -8.57586916e+22 -6.45502864e+22  0.00000000e+00]
x1 = [-1.81281946e+23 -8.57586916e+22 -6.45502864e+22  1.27047373e+24]
end of for i
x0 = [-1.08242792e+23 -5.12062035e+22 -3.85427417e+22  7.58595255e+23]
btol = 9.897557e-03 ; la.norm(r) = 7.218362e+23 ; tol = 1.000000e-04 ; la.norm(x1-x0) = 5.188685e+23; stop = False
x0 = [-1.81281946e+23 -8.57586916e+22 -6.45502864e+22  1.27047373e+24]end of current while


Not converges in 100 iterations
Iter.=100  duration=0.478719  err=1.287823e+24
```

# Graph


Gauss Seidel iterations
1e24
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
iteration
0.0
0.5
1.0
1.5
2.0
2.5
3.0
error

v) Repeat part (iv) for the Gauss Jacobi iteration

## Program

```
import matplotlib.pyplot as plt
from numpy import array, zeros, diag, diagflat, dot
from pprint import pprint
import numpy as np
from numpy import array, inf
from numpy.linalg import norm
from scipy.linalg import solve

np.seterr(divide='ignore', invalid='ignore')


RandomMatrix = np.random.randint(100, size=(4, 4))
B = np.random.randint(100, size=(4, 1))

# Funnction to check diagonally dominant
def IsDiagonallyDominant(MyMatrix):
    D = np.diag(np.abs(MyMatrix))
    S = np.sum(np.abs(MyMatrix), axis=1) - D
    if np.all(D > S):
        print('matrix is diagonally dominant')
    else:
        print('NOT diagonally dominant')
    return

IsDiagonallyDominant(RandomMatrix)
```

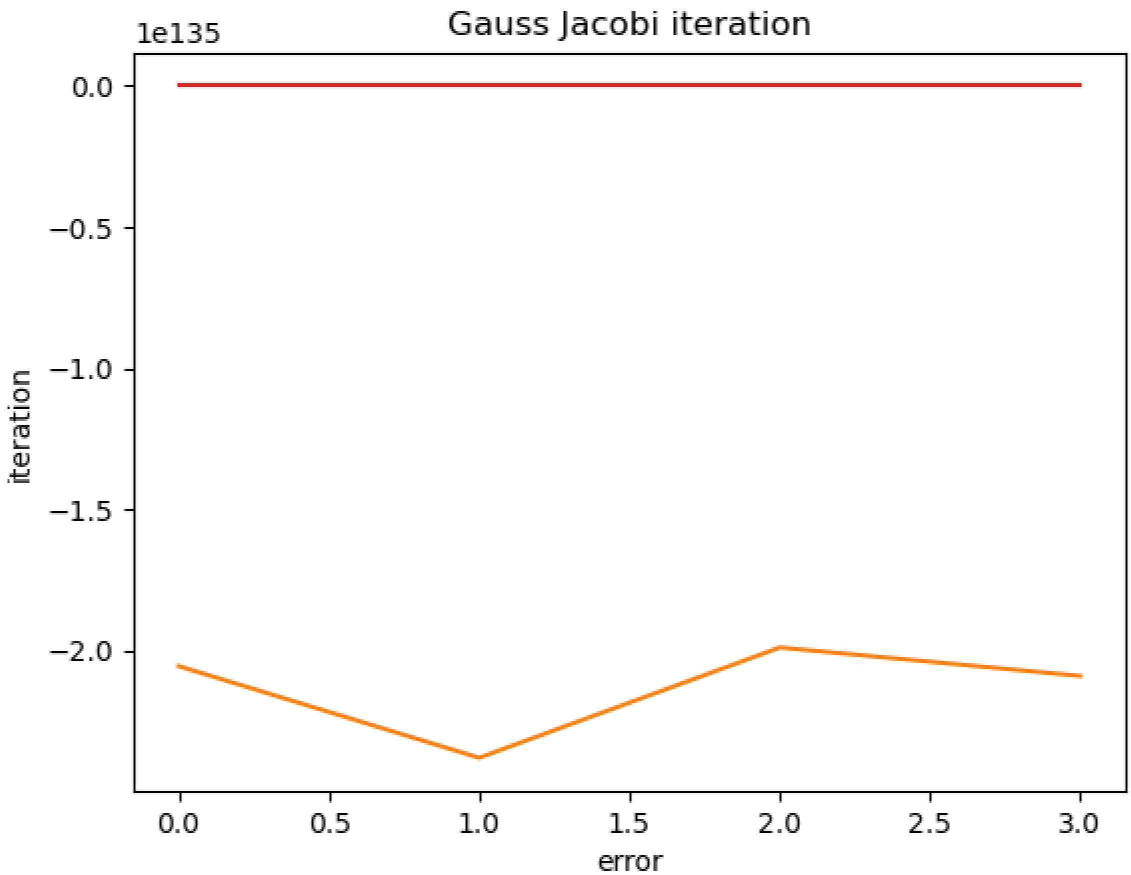
```python
def jacobi(A, b, N=99, x=None):
    """Solves the equation Ax=b via the Jacobi iterative method."""
    if x is None:
        x = zeros(len(A[0]))
    D = diag(A)
    R = A - diagflat(D)

    # Iterate for N times
    for i in range(N):
        x = x.astype('float64')
        x = (b - dot(R, x)) / D
    return x



A = RandomMatrix
b = B
N = 99
x = np.ones((4, 1), dtype=int)

print(A)
print(b)
print(N)
print(x)

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

print("A:")
pprint(A)
```

```
print("b:")
pprint(b)


print("x:")
pprint(sol)


import matplotlib.pyplot as plt


plt.plot(sol)
plt.ylabel('error')
plt.xlabel('iteration')
plt.title('Gauss Jacobi iteration')
plt.show()
```

## Result

```
NOT diagonally dominant
[[71 66 46 65]
 [71 20 72 14]
 [12 13 55 70]
 [33 92 96  1]]
[[50]
 [90]
 [47]
 [85]]
99
[[1]
 [1]
 [1]
 [1]]
A:
array([[71, 66, 46, 65],
       [71, 20, 72, 14],
       [12, 13, 55, 70],
       [33, 92, 96,  1]])
b:
array([[50],
       [90],
       [47],
       [85]])
x:
array([[-8.04998191e+033, -2.09877494e+088, -7.41964068e+044,
        -1.23092992e+217],
       [-6.80406055e+033, -1.77394085e+088, -6.27127924e+044,
        -1.04041497e+217],
       [-5.13678893e+033, -1.33925318e+088, -4.73456072e+044,
        -7.85470982e+216],
       [-8.84501472e+033, -2.30605428e+088, -8.15241969e+044,
        -1.35249910e+217]])
```

## Graph



Gauss Jacobi iteration
1e135
0.0
−0.5
−1.0
−1.5
−2.0
iteration
0.0
0.5
1.0
1.5
2.0
2.5
3.0
error

**Q2) LU Decomposition,Vector Spaces and LT**

i Find the LU decomposition of the matrix A = $\begin{matrix} 1 & 0 & 1 \\ a & a & a \\ b & b & a \end{matrix}$

For which real numbers a and b does it exist?

DSECL ZC416 - Mathematical Foundation for Data Science.

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 & b exists.

Solution :-

$$A = LU$$

$$A = \begin{bmatrix} 1 & 0 & 1 \\ a & a & a \\ b & b & a \end{bmatrix}; \quad L = \begin{bmatrix} 1 & 0 & 0 \\ c & 1 & 0 \\ d & e & 1 \end{bmatrix}; \quad U = \begin{bmatrix} f & g & h \\ 0 & i & j \\ 0 & 0 & k \end{bmatrix}$$

Reducing the matrix,

$$R_2 \Rightarrow R - aR_1$$

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & a & 0 \\ b & b & a \end{bmatrix}$$

For L taking the identity matrix

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

writing Coefficient of a in L at;

Row = 2

Column = 1 :

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

reducing the matrix

$$R_3 \to R_3 - bR_1$$

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

writing coefficient of b in L at row 3, Column 1.

This was based on the reduced matrix form

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

$$R_3 \to R_3 - \frac{b}{a} R_2$$

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

writing the above coefficient $\frac{b}{a}$ in L at Row = 3, Column = 2.

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

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

hence; $A_1 = 1$

Leading sub matrices are

$$A_1 = 1$$

$$A_2 = \begin{bmatrix} 1 & 0 \\ a & a \end{bmatrix}$$

$$A_3 = \begin{bmatrix} 1 & 0 & 1 \\ a & a & a \\ b & b & a \end{bmatrix}$$

i.e $|A_1| = 1 \neq 0$

$$|A_2| = a - 0 \neq 0 = a \neq 0$$

$$|A_3| = 1(a^2 - ab) - 0 + 1(ab - ba) \neq 0$$

$$= a(a-b) \neq 0$$

hence; $a \neq 0$ / $a \neq b$

Thus $a, b \in R$ Except $a \neq 0$ and $a \neq b$

ii Find the dimension of the vector space 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.

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

Solution:

$$\left.\begin{array}{c}\text{Given}\\ \text{Vectors}\end{array}\right\} \Rightarrow \begin{bmatrix}1\\1\\-2\\0\\1\end{bmatrix}, \begin{bmatrix}1\\2\\0\\-4\\1\end{bmatrix}, \begin{bmatrix}0\\1\\3\\-3\\2\end{bmatrix} \begin{bmatrix}2\\3\\0\\-2\\0\end{bmatrix}$$

There are many ways to find a bias.

Here we are Using Row Space

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

Where

$$R_2 \to R_2 - R_1$$

$$R_2 = \begin{bmatrix}1 & 1 & -2 & 0 & 1\\ 0 & 1 & 2 & -4 & 0\\ 0 & 1 & 3 & -3 & 2\\ 2 & 3 & 0 & -2 & 0\end{bmatrix}$$

So,

$$R_2 = \begin{bmatrix}1 & 1 & -2 & 0 & 1\\ 0 & 1 & 2 & -4 & 0\\ 0 & 1 & 3 & -3 & 2\\ 2 & 3 & 0 & -2 & 0\end{bmatrix}$$

Calculating: $R_4$

$$R_4 \rightarrow R_4 - 2R_1$$

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

then making $R_3$

$$R_3 \rightarrow R_3 - R_2$$

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

Substituting $R_4$

$$R_4 \rightarrow R_4 - R_2$$

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

Subsituting $R_4$ again

$$R_4 \rightarrow R_4 - 2R_3$$

$$\begin{bmatrix} 1 & 1 & -2 & 0 & 1 \\ 0 & 1 & 2 & -4 & 0 \\ 0 & 0 & 1 & 1 & 2 \\ 0 & 0 & 0 & 0 & -6 \end{bmatrix}$$

Span the Columns and rows to the Nominal form

The basis is;

$$\Rightarrow \left\{ \begin{bmatrix} 1 \\ 1 \\ -2 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 2 \\ -4 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 2 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ -6 \end{bmatrix} \right\}$$

Also dimension of the vector space

= Number of linearly independent Vectors / Matrix

hence, Vector Space = 4

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

$$Ax = \begin{bmatrix} 1 \\ 4 \\ 1 \\ 1 \end{bmatrix} \text{ is of th1e form } : x = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} + c \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix}, \ c \in R$$

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

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

iii) a) What can be said about columns of matrix (A)

b) Find the dimension of null space & rank of matrix A?

Solution:

a)

$$[A]_{m\times n} \begin{bmatrix} 0 \\ 1+2c \\ 1+c \end{bmatrix}_{3\times 1} = \begin{bmatrix} 1 \\ 4 \\ 1 \\ 1 \end{bmatrix}_{4\times 1}$$

We know that matrix multiplication is possible by only by when

Number of column in Matrix A } = Number of column in Matrix B }

Order of matrix A = 4 * 3

hence Matrix 'A' has '3' columns.

b)

$$Ax = \underbrace{\begin{bmatrix} 1 \\ 4 \\ 1 \\ 1 \end{bmatrix}}_{A} \underbrace{\left[\begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} + c \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix}\right]}_{x}$$

To find the null space

Reduce the Row echelam form;

We got $\begin{bmatrix} 1 & 4 & 1 & 1 \end{bmatrix}$

$$x = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} + c\begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix}$$

$$x = \begin{bmatrix} 0 \\ 1+2c \\ 1+c \end{bmatrix}$$

For null space;

$$\begin{bmatrix} 1 & 4 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1+2c \\ 1+c \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ 4(1+2c) \\ 1(1+c) \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \end{bmatrix}$$

$$4 + 8c = x_2$$
$$1 + c = x_3$$
$$4c = 0$$
$$c = 0$$

, The matrix $= \begin{bmatrix} 0 \\ 4 \\ 1 \\ 0 \end{bmatrix}$

hence

The nullity of the Matrix $= 1$

Rank also $= 1$