DEPARTMENT OF SCIENCE & HUMANITIES

SCILAB MANUAL
LINEAR ALGEBRA- UE19MA251

IV Semester - B. Tech

January-May 2021

Compiled by

Mrs. RENNA SULTANA

Associate Professor in Mathematics

Department of Science & Humanities

PES University, RR Campus

Bangalore

Contents

SCILAB SYLLABUS

Gaussian Elimination , The LU Decomposition, Inverse of a Matrix by the Gauss-Jordan Method, The Span of Column Space of a Matrix, The Four Fundamental Subspaces, Projections by Least Squares, The Gram-Schmidt Orthogonalization, Eigen values and Eigen Vectors of a Matrix, The Largest Eigen Value of a Matrix by the Power Method.

Lesson Plan:

Class Number	Topic
1	Gaussian Elimination
2	The LU Decomposition
3	Inverse of a Matrix by the Gauss- Jordan Method
4	The Span of Column Space of a Matrix
5	The Four Fundamental Subspaces
6	Projections by Least Squares
7	The Gram-Schmidt Orthogonalization
8	Eigen values and Eigen Vectors of a Matrix
9-10	In Semester Assessment

ABOUT SCILAB

Overview of Scilab

Scilab is a programming language associated with a rich collection of numerical algorithms covering many aspects of scientific computing problems. From the software point of view, Scilab is an interpreted language. This generally allows to get faster development processes, because the user directly accesses a high-level language, with a rich set of features provided by the library. The Scilab language is meant to be extended so that user-defined data types can be defined with possibly overloaded operations. Scilab users can develop their own modules so that they can solve their particular problems. The Scilab language allows to dynamically compile and link other languages such as Fortran and C: this way, external libraries can be used as if they were a part of Scilab built-in features. Scilab also interfaces LabVIEW, a platform and development environment for a visual programming language from National Instruments. From the license point of view, Scilab is a free software in the sense that the user does not pay for it and Scilab is an open source software, provided under the Cecill license. The software is distributed with source code, so that the user has an access to Scilab's most internal aspects. Most of the time, the user downloads and installs a binary version of Scilab, since the Scilab consortium provides Windows, Linux and Mac OS executable versions. Online help is provided in many local languages. From the scientific point of view, Scilab comes with many features. At the very beginning of Scilab, features were focused on linear algebra. But, rapidly, the number of features extended to cover many areas of scientific computing.

The following is a short list of its capabilities:

• Linear algebra, sparse matrices

• Polynomials and rational functions

• Interpolation, approximation

• Linear, quadratic and non-linear optimization

• Ordinary Differential Equation solver and Differential Algebraic Equations solver

• Classic and robust control, Linear Matrix Inequality optimization

• Differentiable and non-differentiable optimization

• Signal processing

• Statistics

Scilab provides many graphics features including a set of plotting functions which allow us to create 2D and 3D plots as well as user interfaces. The Xcos environment provides a hybrid dynamic systems modeler and simulator.

How to get and install Scilab

Whatever the platform is (i.e. Windows, Linux or Mac), Scilab binaries can be downloaded directly from the Scilab homepage http://www.scilab.org or from the Download area http://www.scilab.org/download. Scilab binaries are provided for both 32 and 64-bit platforms so that they match the target installation machine. Scilab can also be downloaded in source form, so that we can compile Scilab by ourselves and produce our own binary. Compiling Scilab and generating a binary is especially interesting when we want to understand or debug an existing feature, or when we want to add a new feature. To compile Scilab, some prerequisites binary files are necessary, which are also provided in the Download center. Moreover, a Fortran and a C compiler are required.

Become familiar with Scilab

The useful workspace in Scilab consists of several window:

- The console for making calculations
- The editor for writing programs
- The graphics windows for displaying graphics
- The embedded help

The general environment and the console

After double clicking the icon to launch Scilab, the Scilab environment by default consists of the following docked windows: The console, files and variables browsers and the command history. In the console, after the prompt

```
-->
```

Just type a command and press the Enter key (Windows and Linux) or the Return key (Mac OS X) in the keyboard to obtain the corresponding result.

The menu bar

The menus listed below are particularly useful.

Applications

- The command history allows us to find all the commands from previous sessions to the current session.
- The variables browser allows us to find all variables previously used during the current session.

Edit

Preferences (in Scilab menu under Mac OS X) allows us to set and customize colours, fonts and font size in the console and in the editor, which is very useful for screen projection. Clicking on Clear Console clears the entire content of the console. In this case, the command history is still available and calculations made during the session remain in memory. Commands that have been erased are still available through the keyboard's arrow keys.

The Editor

Typing directly into the console has two disadvantages: it is not possible to save the commands and it is not easy to edit multiple lines of instruction. The editor is the appropriate tool to run multiple instructions.

Opening the Editor

To open the editor from the console, click on the first icon in the toolbar or on Applications → SciNotes in the menu bar. The editor opens with a default file named "Untitled 1 ".

Saving

Any file can be saved by clicking on File → Save as

The extension ".**sce** "at the end of a file name will launch automatically Scilab when opening it (except under Linux and Mac OS X).

Copying into the console, executing a program

In clicking on Execute in the menu bar, three options are available:

• Execute "…file with no echo "(Ctrl+Shift+E under Windows and Linux, Cmd+Shift+ E under Mac OS X): the file is executed without writing the program in the console (saving the file first is mandatory).

• Execute "… file with echo "(Ctrl+L under Windows and Linux, Cmd+L under Mac OS X): rewrites the file into the console and executes it.

• Execute "…until the caret, with echo "(Ctrl+E under Windows and Linux, Cmd+E under Mac OS X): rewrites the selection chosen with the mouse into the console and executes it or executes the file data until the caret position defined by the user. Standard copy/paste can also be used.

The graphics window

Opening a graphics window

A graphics window opens automatically when making any plot. It is possible to plot curves, surfaces and sequences of points.

Online help

To access the online help, click on ? →Scilab Help in the menu bar, or type in the console:

→ help

To get help with any function, type help in the console followed by the name of the appropriate function. For example:

→ help **sin**

displays help for **sine** function.

Programming

In the examples given in this manual, any line preceded by " --> " is a command, the other lines are the returns from commands (calculation results, error messages…). Do not write " --> " in the editor. They are introduced here to make the distinction between command lines and calculation results as it is displayed in the console after copying/pasting. When presented in a table (without " > " and no calculation result), the commands are depicted exactly as they should be typed in the editor.

Variables, assignment and display

Variables

Scilab is not a computer algebra system. It calculates only with numbers. All calculations are done with matrices, although this may go unnoticed. Even if the concept of matrices is unknown, vectors and sequences of numbers can explain it, as they are, in fact, matrices of dimension 1 × nor n × 1 and a number is itself a matrix of dimension 1 × 1. Variables do not need to be declared in advance, but any variable must have a value. For example, obtaining the value of a variable which has not been given a value produces an error.

```
-->a

 ! --error 4

 Undefined variable: a
```

If a value is assigned to the variable a, there is no longer an error:

```
--> a=%pi/4

a =
        0.7853982
 --> a

a =
        0.7853982
```

Variables may take any name that is not already defined by the system:

```
--> Piby2=%pi/2

Piby2 =
        1.5707963
```

The result of a calculation that is not assigned to a variable is automatically assigned to the variable called ans:

```
-->3*(4-2)

ans =
        6.
-->ans
ans =
        6.
```

Functions
Functions are the easiest and most natural way to make computations from variables a nd for obtaining results from variables.
A function definition begins with **function** and ends with **endfunction**.

Requesting the assignment of a variable
Assignment is made easily using the "= "operator.

Display

Writing

Typing the name of a variable displays its value, except when there is "; "at the end of the command line.

Brackets

Matrices are defined using square brackets.
As mentioned before, matrix computation is the basis of calculations in Scilab. A spac e or comma is used to separate columns and semicolons are used to separate rows.

To define a column vector and obtain a column display:

```
-->v=[3;-2;5]

v =

        3.

        - 2.

        5.
```

To define a row vector and obtain a row display:

```
--> v = [3,-2,5]

v =
        3.  - 2.    5.
```

To define a matrix and obtain a tabular display:

```
--> m = [1 2 3;4 5 6;7 8 9]

m =
     1.   2.   3.
     4.   5.   6.
     7.   8.   9.
```

disp function

The disp must always be used with parentheses. With the vector v previously defined:

```
-->v(2)
ans =
        - 2.
-->disp(v(2))
        - 2.
```

To display a string (usually a sentence), put it between quotes:

```
-->disp("The solution is")   will display the following
The solution is
```

Additional information on matrices and vectors

Accessing elements

Square braces are used to define matrices. A space or a comma is used to switch from one column to another and semicolons are used to switch from one line to another.

```
-->m= [1 2 3;4 5 6]
m =
        1.   2.   3.
        4.   5.   6.
```

Parentheses are used to access elements or to modify them.

```
-->m(2,3)
ans =
        6.
-->m(2,3)=23
 m  =
        1.   2.   3.
        4.   5.   23.
```

The “ : “ operator is used to designate all the rows or all columns of a matrix.
To view the second row of the matrix m, type:

```
-->m(2,:)
ans   =
        4.   5.   23.
```

To view the third column:

```
-->m(:,3)
ans   =
        3.
        23.
```

To obtain the transpose of a matrix or a vector use the single quote symbol “ ' “:

```
-->m'
ans =
      1.   4.
      2.   5.
      3.   23.
```

Operations

The operations   * “, “/ “are matrix operations. To make element wise operations, we need to put dot before the operating sign: “.* “, “. / “.

Solving linear systems

To solve the linear system AX = Y, in which A is a square matrix, use the backslash “\ “

X = A \ Y.

Be cautious, the operation Y / A will give
(only if the dimensions are correct) another result, that is to say the matrix Z for which Z A = Y.

Length

The length function returns the number of coordinates for a vector. The size function returns the dimensions (rows, columns) for a matrix.

```
-->U=[1:10]
U =
      1.   2.   3.   4.   5.   6.   7.   8.   9.   10.
-->length(U)
ans =
      10.
-->m=[1 2 3;4 5 6];
-->size(U)
ans =
   2.   3.
```

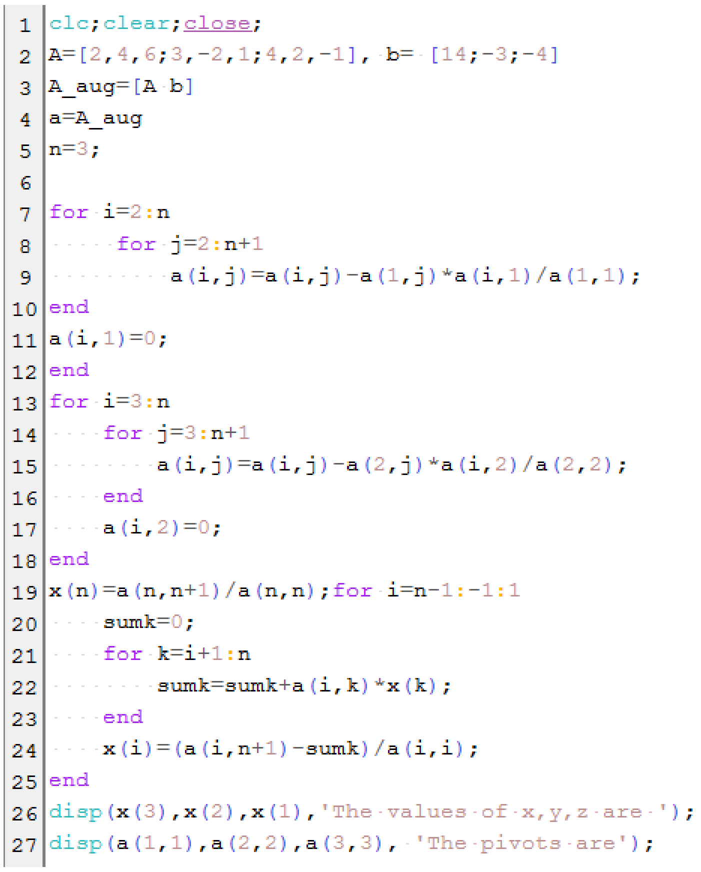
CLASS NUMBER - 1

Topic: Gaussian Elimination

Procedure: Given a system of n equations in n unknowns we use the method of pivoting to solve for the unknowns. The method starts by subtracting multiples of the first equation from the other equations. The aim is to eliminate the first unknown from the second equation onwards. We use the coefficient of the first unknown in the first equation as the first pivot to achieve this. At the end of the first stage of elimination, there will be a column of zeros below the first pivot. Next, the pivot for the second stage of elimination is located in the second row second column of the system. A multiple of the second equation will now be subtracted from the remaining equations using the second pivot to create zeros just below it in that column. The process is continued until the system is reduced to an upper triangular one. The system can now be solved backward bottom to top.

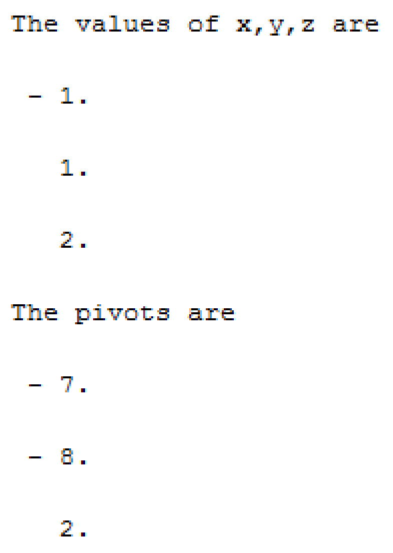
Example 1: Solve the system of equations $2x + 4y + 6z = 14$, $3x - 2y + z = -3$ and $4x + 2y - z = -4$ using Gaussian Elimination. Also, identify the pivots.

Solution:



```
clc;clear;close;
A=[2,4,6;3,-2,1;4,2,-1], b= [14;-3;-4]
A_aug=[A b]
a=A_aug
n=3;

for i=2:n
    for j=2:n+1
        a(i,j)=a(i,j)-a(1,j)*a(i,1)/a(1,1);
end
a(i,1)=0;
end
for i=3:n
    for j=3:n+1
        a(i,j)=a(i,j)-a(2,j)*a(i,2)/a(2,2);
    end
    a(i,2)=0;
end
x(n)=a(n,n+1)/a(n,n);for i=n-1:-1:1
    sumk=0;
    for k=i+1:n
        sumk=sumk+a(i,k)*x(k);
    end
    x(i)=(a(i,n+1)-sumk)/a(i,i);
end
disp(x(3),x(2),x(1),'The values of x,y,z are ');
disp(a(1,1),a(2,2),a(3,3), 'The pivots are');
```

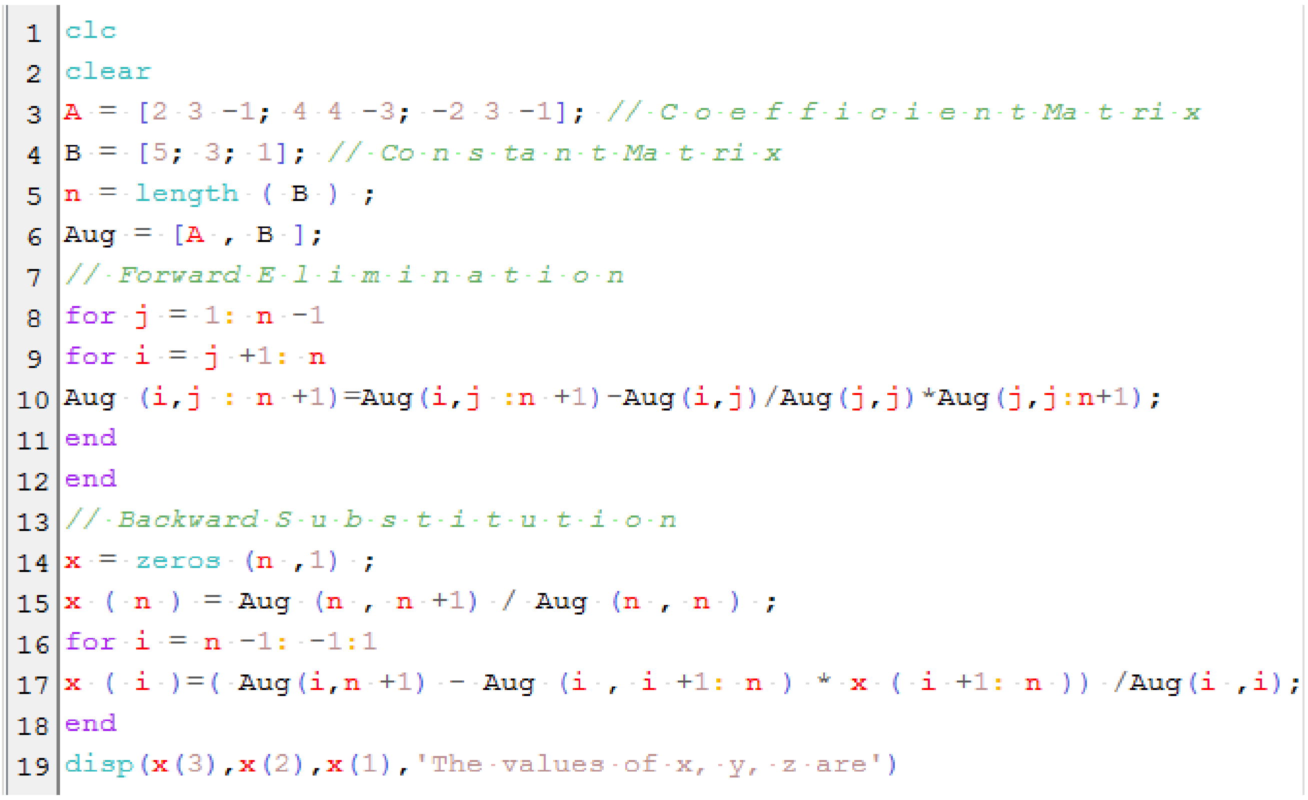
Output:



```
The values of x,y,z are

 - 1.

   1.

   2.

The pivots are

 - 7.

 - 8.

   2.
```

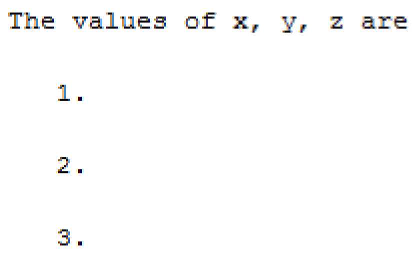
Example 2: Solve the system of equations 2x + 3y –z = 5, 4x + 4y -3 z = 3 and -2x + 3y –z = 1 by Gaussian Elimination.

Solution:



```
clc
clear
A = [2 3 -1; 4 4 -3; -2 3 -1]; // Coefficient Matrix
B = [5; 3; 1]; // Constant Matrix
n = length ( B ) ;
Aug = [A , B ];
// Forward Elimination
for j = 1: n -1
for i = j +1: n
Aug (i,j : n +1)=Aug(i,j :n +1)-Aug(i,j)/Aug(j,j)*Aug(j,j:n+1);
end
end
// Backward Substitution
x = zeros (n ,1) ;
x ( n ) = Aug (n , n +1) / Aug (n , n ) ;
for i = n -1: -1:1
x ( i )=( Aug(i,n +1) - Aug (i , i +1: n ) * x ( i +1: n )) /Aug(i ,i);
end
disp(x(3),x(2),x(1),'The values of x, y, z are')
```

Output:



```
The values of x, y, z are

   1.

   2.

   3.
```

Practice Problems

Solve the following system of equations by Gaussian Elimination. Identify the pivots in each case.

1. 2x + 5y + z = 0 , 4x + 8y + z = 2, y –z = 3
2. 2x + 3y + z = 8 , 4x + 7y + 5z = 20, -2y + 2z = 0
3. 2x – 3y = 3, 4x – 5y + z = 7, 2x – y – 3z = 5.

CLASS NUMBER - 2

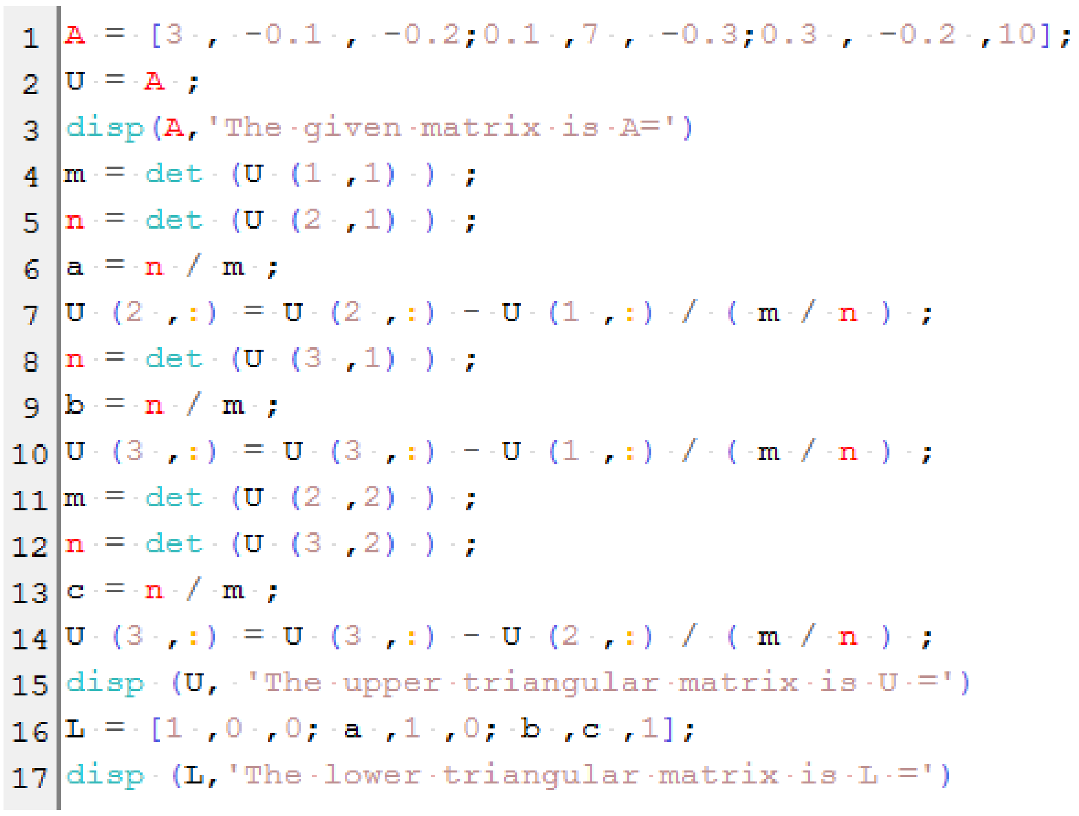
Topic: LU decomposition of a matrix

Procedure: Given a square matrix A, we can factorize A as a product of two matrices L and U. The matrix U is upper triangular with pivots on the main diagonal and the matrix L is lower triangular with 1's on the main diagonal and the multipliers in their respective positions. The factorization is carried out using the method of Gaussian Elimination.

Example 3: Find the triangular factors L and U for the matrix A =

$$\begin{pmatrix} 3 & -0.1 & -0.2 \\ 0.1 & 7 & -0.3 \\ 0.3 & -0.2 & 10 \end{pmatrix}$$

Solution:



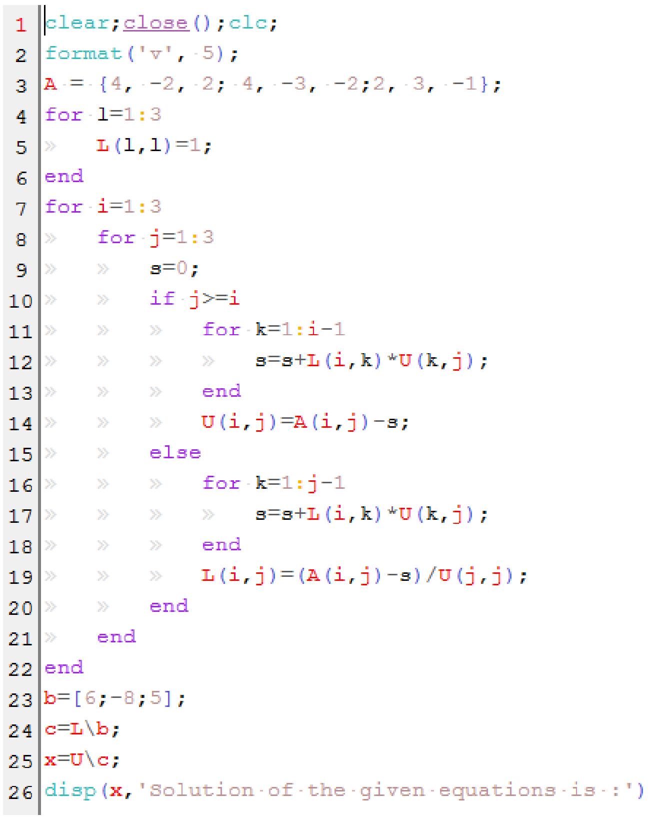
```
A = [3 , -0.1 , -0.2;0.1 ,7 , -0.3;0.3 , -0.2 ,10];
U = A ;
disp(A,'The given matrix is A=')
m = det (U (1 ,1) ) ;
n = det (U (2 ,1) ) ;
a = n / m ;
U (2 ,:) = U (2 ,:) - U (1 ,:) / ( m / n ) ;
n = det (U (3 ,1) ) ;
b = n / m ;
U (3 ,:) = U (3 ,:) - U (1 ,:) / ( m / n ) ;
m = det (U (2 ,2) ) ;
n = det (U (3 ,2) ) ;
c = n / m ;
U (3 ,:) = U (3 ,:) - U (2 ,:) / ( m / n ) ;
disp (U, 'The upper triangular matrix is U =')
L = [1 ,0 ,0; a ,1 ,0; b ,c ,1];
disp (L,'The lower triangular matrix is L =')
```

Output:

```
The given matrix is A=

   3.    - 0.1   - 0.2
   0.1     7.    - 0.3
   0.3   - 0.2     10.

The upper triangular matrix is U =

   3.   - 0.1          - 0.2
   0.     7.0033333    - 0.2933333
   0.     0.             10.012042

The lower triangular matrix is L =

   1.            0.            0.
   0.0333333     1.            0.
   0.1         - 0.0271299     1.
```

Example 4: Solve the system of equations 4x -2 y + 2z = 6, 4x -3 y -2 z = -8 and 2x + 3y –z = 5 by LU decomposition.

Solution:



```
clear;close();clc;
format('v', 5);
A = [4, -2, 2; 4, -3, -2;2, 3, -1];
for l=1:3
    L(l,l)=1;
end
for i=1:3
    for j=1:3
        s=0;
        if j>=i
            for k=1:i-1
                s=s+L(i,k)*U(k,j);
            end
            U(i,j)=A(i,j)-s;
        else
            for k=1:j-1
                s=s+L(i,k)*U(k,j);
            end
            L(i,j)=(A(i,j)-s)/U(j,j);
        end
    end
end
b=[6;-8;5];
c=L\b;
x=U\c;
disp(x,'Solution of the given equations is :')
```

Output:

```
Solution of the given equations is :

   1.
   2.
   3.
```

Practice Problems

1. Factorize the following matrices as A = LU

   (ι) $A=\begin{pmatrix} 2 & 3 & 1 \\ 4 & 7 & 5 \\ 1 & -2 & 2 \end{pmatrix}$ (ii) $A=\begin{pmatrix} 2 & -3 & 0 \\ 4 & -5 & 1 \\ 2 & -1 & -3 \end{pmatrix}$

2. Solve the system of equations by decomposing A as a product A = LU

   i) $2x + 3y + z = 8$, $4x + 7y + 5z = 20$, $-2y + 2z = 0$

   ii) $2x - 3y = 3$, $4x - 5y + z = 7$, $2x - y - 3z = 5$.

CLASS NUMBER – 3

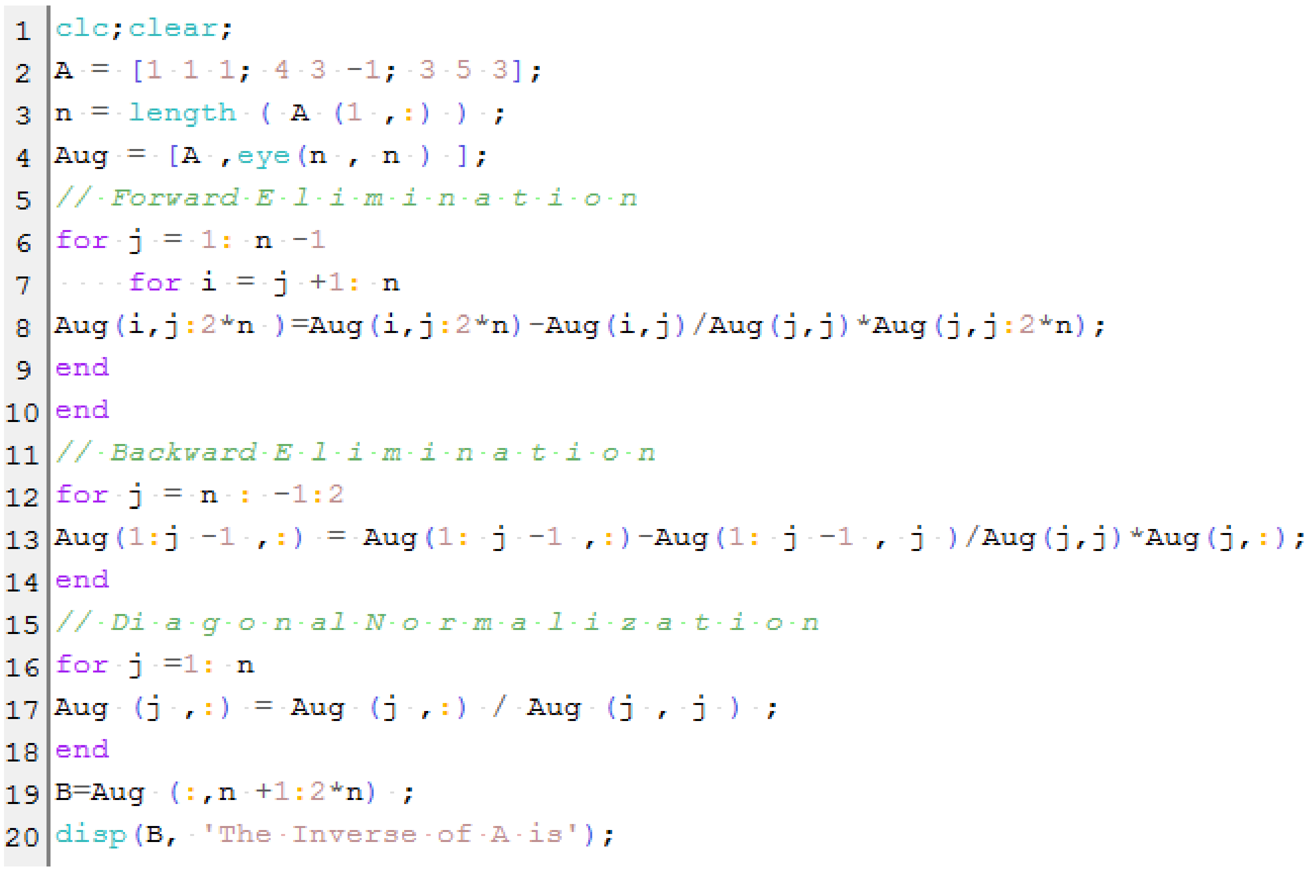
Topic: The Gauss - Jordan method of calculating A^{-1}

Procedure: To find the inverse of a non-singular matrix A, we begin with an augmented system [A I] where I is an identity matrix of the size same as that of A and reduce A to an upper triangular system by Gaussian Elimination. The system is further reduced to a diagonal one with pivots on the main diagonal and finally to [I A^{-1}].

Example 5: Find the inverse of the matrix A $= \begin{pmatrix} 1 & 1 & 1 \\ 4 & 3 & -1 \\ 3 & 5 & 3 \end{pmatrix}$

by Gauss – Jordan method.

Solution:



```
clc;clear;
A = [1 1 1; 4 3 -1; 3 5 3];
n = length ( A (1 ,:) ) ;
Aug = [A ,eye(n , n ) ];
// Forward E l i m i n a t i o n
for j = 1: n -1
    for i = j +1: n
Aug(i,j:2*n )=Aug(i,j:2*n)-Aug(i,j)/Aug(j,j)*Aug(j,j:2*n);
end
end
// Backward E l i m i n a t i o n
for j = n : -1:2
Aug(1:j -1 ,:) = Aug(1: j -1 ,:)-Aug(1: j -1 , j )/Aug(j,j)*Aug(j,:);
end
// Di a g o n al N o r m a l i z a t i o n
for j =1: n
Aug (j ,:) = Aug (j ,:) / Aug (j , j ) ;
end
B=Aug (:,n +1:2*n) ;
disp(B, 'The Inverse of A is');
```

Output:

```
 The Inverse of A is

     1.4    0.2  - 0.4
   - 1.5    0.     0.5
     1.1  - 0.2  - 0.1
```

Alternate Solution:

```
clc
clear
A = [1 1 1; 4 3 -1; 3 5 3];
n = length ( A (1 ,:) ) ;
Aug = [A ,eye(n , n ) ];
N = 1: n ;
for i = 1: n
dummy1 = N ;
dummy1 ( i ) = [];
index (i ,:) = dummy1 ;
end
// Forward E l i m i n a t i o n
for j = 1: n
[ dummy2 , t ] = max (abs( Aug ( j :n ,j ) ) ) ;
lrow = t +j -1;
Aug ([ j , lrow ] ,:) = Aug ([ lrow , j ] ,:) ;
Aug (j ,:) = Aug (j ,:) / Aug (j , j ) ;
for i = index (j ,:)
Aug (i ,:) = Aug (i ,:) - Aug (i , j ) / Aug (j , j ) *Aug (j ,:) ;
end
end
Inv_A = Aug (: , n +1:2* n ) ;
disp ( Inv_A)
```

Output:

```
     1.4    0.2  - 0.4
   - 1.5    0.     0.5
     1.1  - 0.2  - 0.1
```

<u>Practice Problems:</u>

1. Find the inverse of the following matrices:

$$\begin{pmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \end{pmatrix} \quad \begin{pmatrix} \mathbf{2} & \mathbf{-1} & \mathbf{0} \\ \mathbf{-1} & \mathbf{2} & \mathbf{-1} \\ \mathbf{0} & \mathbf{-1} & \mathbf{2} \end{pmatrix} \quad \begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{1} \\ \mathbf{0} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & \mathbf{1} \end{pmatrix}$$

CLASS NUMBER – 4

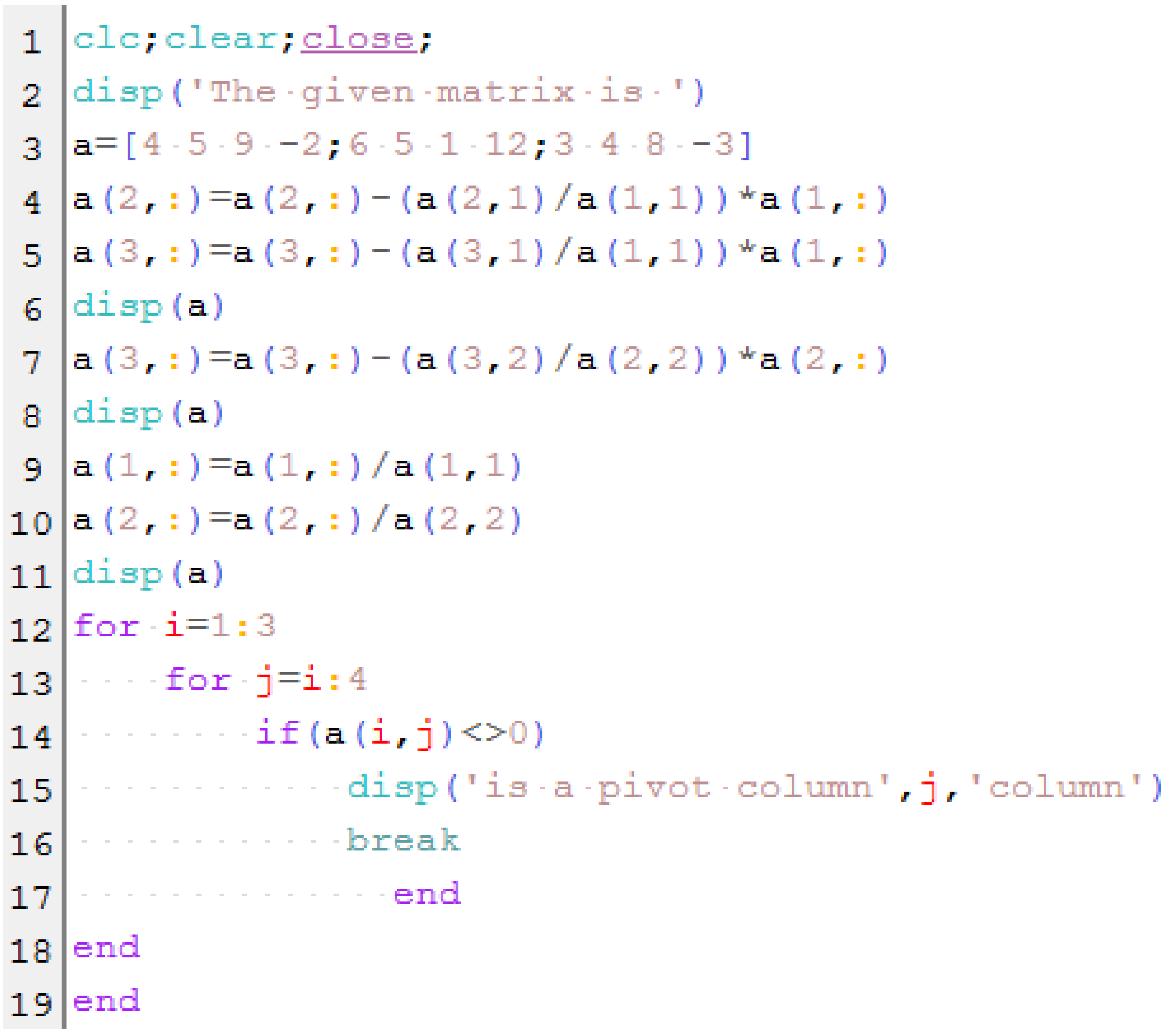
Topic: Span of the Column Space of A

Procedure: Given a matrix A, we reduce it to an upper triangular form using Gaussian Elimination. The columns that contain the pivots span the column space of A.

Example 6: Identify the columns that are in the column space of A where

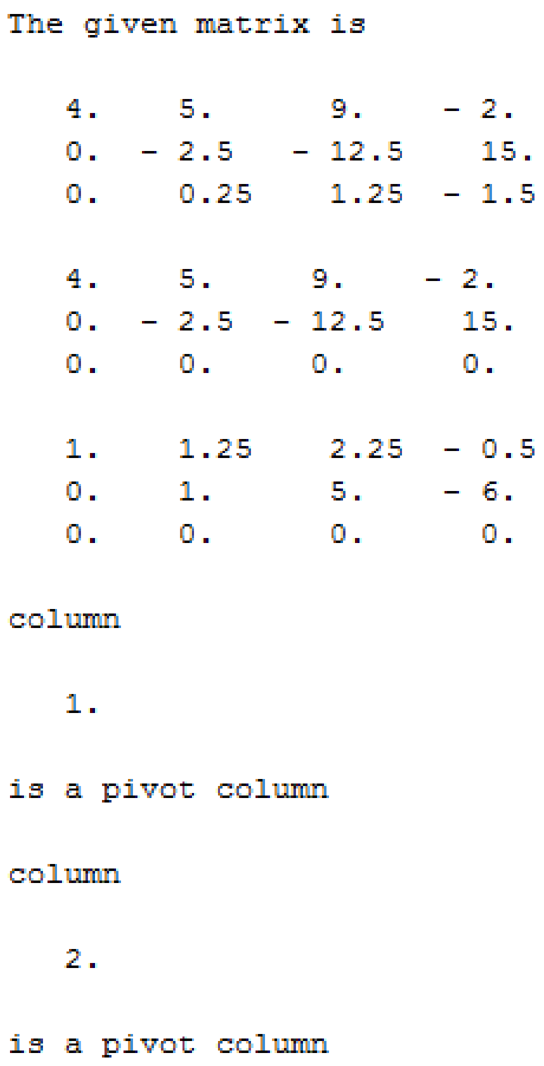
$$A=\begin{pmatrix} 4 & 5 & 9 & -2 \\ 6 & 5 & 1 & 12 \\ 3 & 4 & 8 & -3 \end{pmatrix}$$

Solution:



```
clc;clear;close;
disp('The given matrix is ')
a=[4 5 9 -2;6 5 1 12;3 4 8 -3]
a(2,:)=a(2,:)-(a(2,1)/a(1,1))*a(1,:)
a(3,:)=a(3,:)-(a(3,1)/a(1,1))*a(1,:)
disp(a)
a(3,:)=a(3,:)-(a(3,2)/a(2,2))*a(2,:)
disp(a)
a(1,:)=a(1,:)/a(1,1)
a(2,:)=a(2,:)/a(2,2)
disp(a)
for i=1:3
    for j=i:4
        if(a(i,j)<>0)
            disp('is a pivot column',j,'column')
            break
              end
end
end
```

Output:



```
The given matrix is

   4.     5.       9.     - 2.
   0.   - 2.5    - 12.5    15.
   0.     0.25     1.25   - 1.5

   4.     5.      9.     - 2.
   0.   - 2.5   - 12.5    15.
   0.     0.      0.       0.

   1.     1.25     2.25   - 0.5
   0.     1.       5.     - 6.
   0.     0.       0.       0.

column

   1.

is a pivot column

column

   2.

is a pivot column
```

Practice Problems:

Identify the columns that span the column space of A in the following cases.

(1)A=$\begin{pmatrix} 1 & 3 & 3 & 2 \\ 2 & 6 & 9 & 7 \\ -1 & -3 & 3 & 4 \end{pmatrix}$ (2) A=$\begin{pmatrix} 2 & 4 & 6 & 4 \\ 2 & 5 & 7 & 6 \\ 2 & 3 & 5 & 2 \end{pmatrix}$

CLASS NUMBER –5

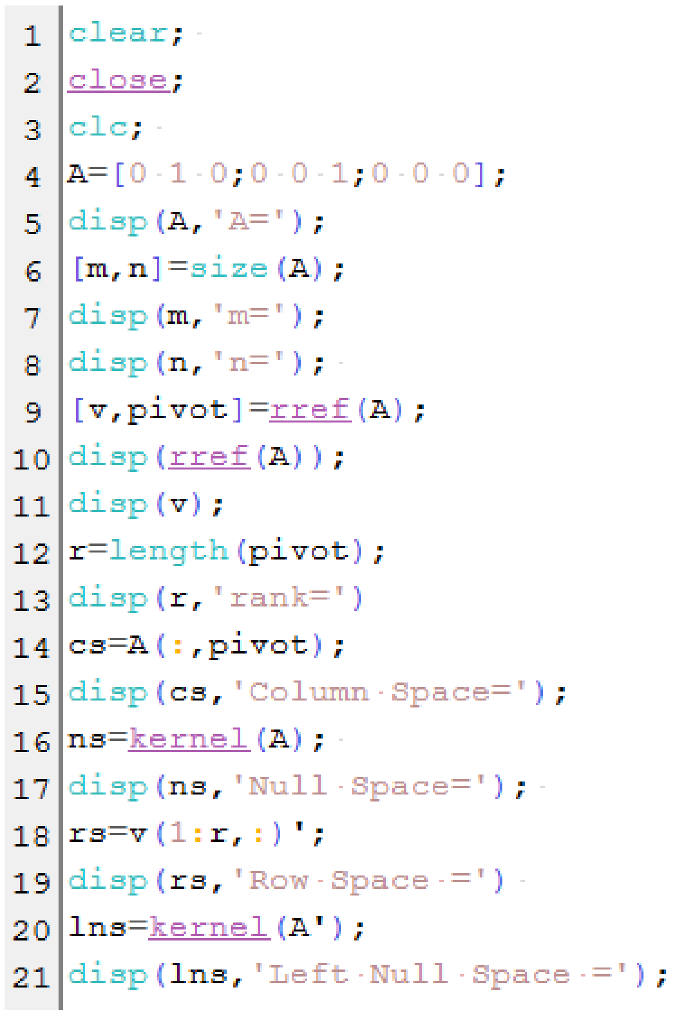
Topic: The Four Fundamental Subspaces

Procedure: Given a matrix A, we reduce it to row reduced form and find its rank by identifying the columns that contain the pivots. We then find the four fundamental subspaces viz, the column space C(A), the row space C(AT), the null space N(A) and the left null space N(AT).

Example 7: Find the four fundamental subspaces of

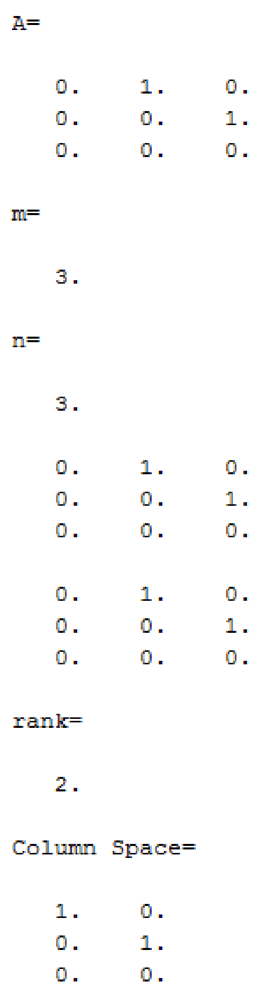
$$A=\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

Solution:



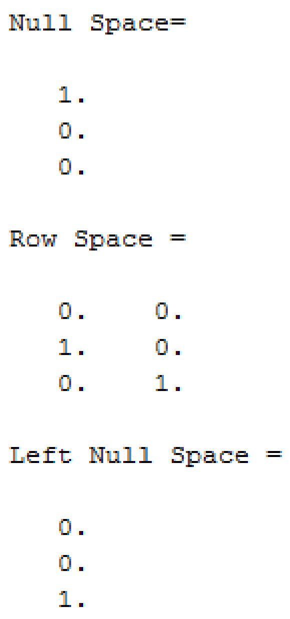
```
clear;
close;
clc;
A=[0 1 0;0 0 1;0 0 0];
disp(A,'A=');
[m,n]=size(A);
disp(m,'m=');
disp(n,'n=');
[v,pivot]=rref(A);
disp(rref(A));
disp(v);
r=length(pivot);
disp(r,'rank=')
cs=A(:,pivot);
disp(cs,'Column Space=');
ns=kernel(A);
disp(ns,'Null Space=');
rs=v(1:r,:)';
disp(rs,'Row Space =')
lns=kernel(A');
disp(lns,'Left Null Space =');
```

Output:



```
A=

   0.    1.    0.
   0.    0.    1.
   0.    0.    0.

m=

   3.

n=

   3.

   0.    1.    0.
   0.    0.    1.
   0.    0.    0.

   0.    1.    0.
   0.    0.    1.
   0.    0.    0.

rank=

   2.

Column Space=

   1.    0.
   0.    1.
   0.    0.
```

Figure XVI - The Four Fundamental Subspaces - Output 7 - 1



```
Null Space=

   1.
   0.
   0.

Row Space =

   0.    0.
   1.    0.
   0.    1.

Left Null Space =

   0.
   0.
   1.
```

Practice Problems:

1. Find the four fundamental subspaces of

$$A=\begin{pmatrix} 1 & 2 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 2 & 0 & 1 \end{pmatrix}$$

2. Find the four fundamental subspaces of

$$A=\begin{pmatrix} 1 & 3 & 3 & 2 \\ 2 & 6 & 9 & 7 \\ -1 & -3 & 3 & 4 \end{pmatrix}$$

CLASS NUMBER - 6

Topic: Projections by Least Squares

Procedure: Suppose we do a series of experiments and expect the output b to be a linear function of the input t. We look for a straight line b = C + Dt. If there are experimental errors then we have a system of equations

$$C + Dt_1 = b_1$$

$C + Dt_2 = b_2$ and so on.

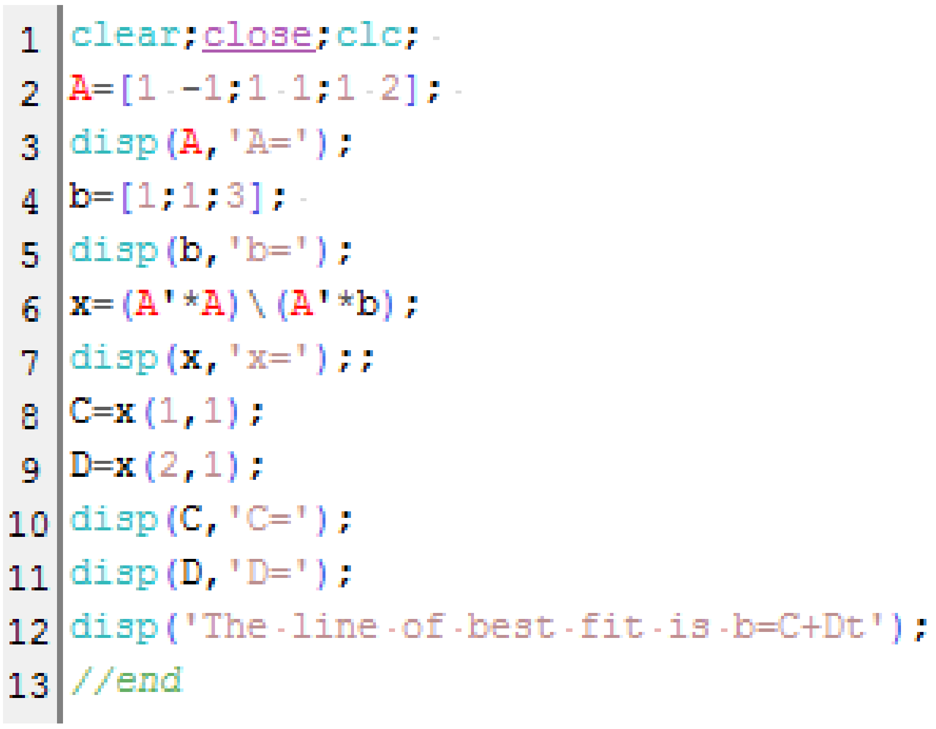
That is, we have the system of equations Ax = b. The best solution is obtained by minimizing the error

$$E^2 = \quad b - Ax \quad ^2 = (b_1 - C - Dt_1)^2 + (b_2 - C - Dt_2)^2 + \ldots\ldots + (b_m - C - Dt_m)^2$$

Example 8: Find the solution x = (C , D) of the system Ax = b and the line of best fit C + Dt = b given

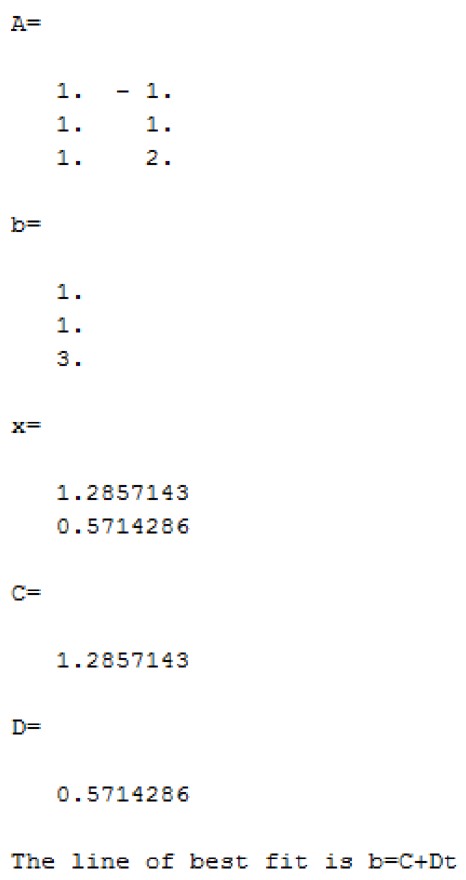
$$A=\begin{pmatrix} 1 & -1 \\ 1 & 1 \\ 1 & 2 \end{pmatrix} \quad b = (1, 1, 3)$$

Solution:



```
clear;close;clc;
A=[1 -1;1 1;1 2];
disp(A,'A=');
b=[1;1;3];
disp(b,'b=');
x=(A'*A)\(A'*b);
disp(x,'x=');;
C=x(1,1);
D=x(2,1);
disp(C,'C=');
disp(D,'D=');
disp('The line of best fit is b=C+Dt');
//end
```

Output:



```
A=

   1.  - 1.
   1.    1.
   1.    2.

b=

   1.
   1.
   3.

x=

   1.2857143
   0.5714286

C=

   1.2857143

D=

   0.5714286

The line of best fit is b=C+Dt
```

<u>Practice Problems:</u>

1. Solve Ax = b by least squares where

$$A=\begin{pmatrix}1 & 0\\ 0 & 1\\ 1 & 1\end{pmatrix} \quad b=\begin{pmatrix}1\\ 1\\ 0\end{pmatrix}$$

2. Find the line of best fit Ax=b for the following system

$$Ax=\begin{pmatrix}1 & -1\\ 1 & 0\\ 1 & 1\end{pmatrix}\begin{pmatrix}C\\ D\end{pmatrix},\ b=\begin{pmatrix}4\\ 5\\ 9\end{pmatrix}$$

3. Find the best straight line fit given b = (4, 3, 1, 0) at t = - 2 , - 1 , 0, 2.

4. Find a linear relationship between b and t that suits the following data the best:

$b = 2, 0, -3, -5$ at t = - 1 , 0, 1 , 2.

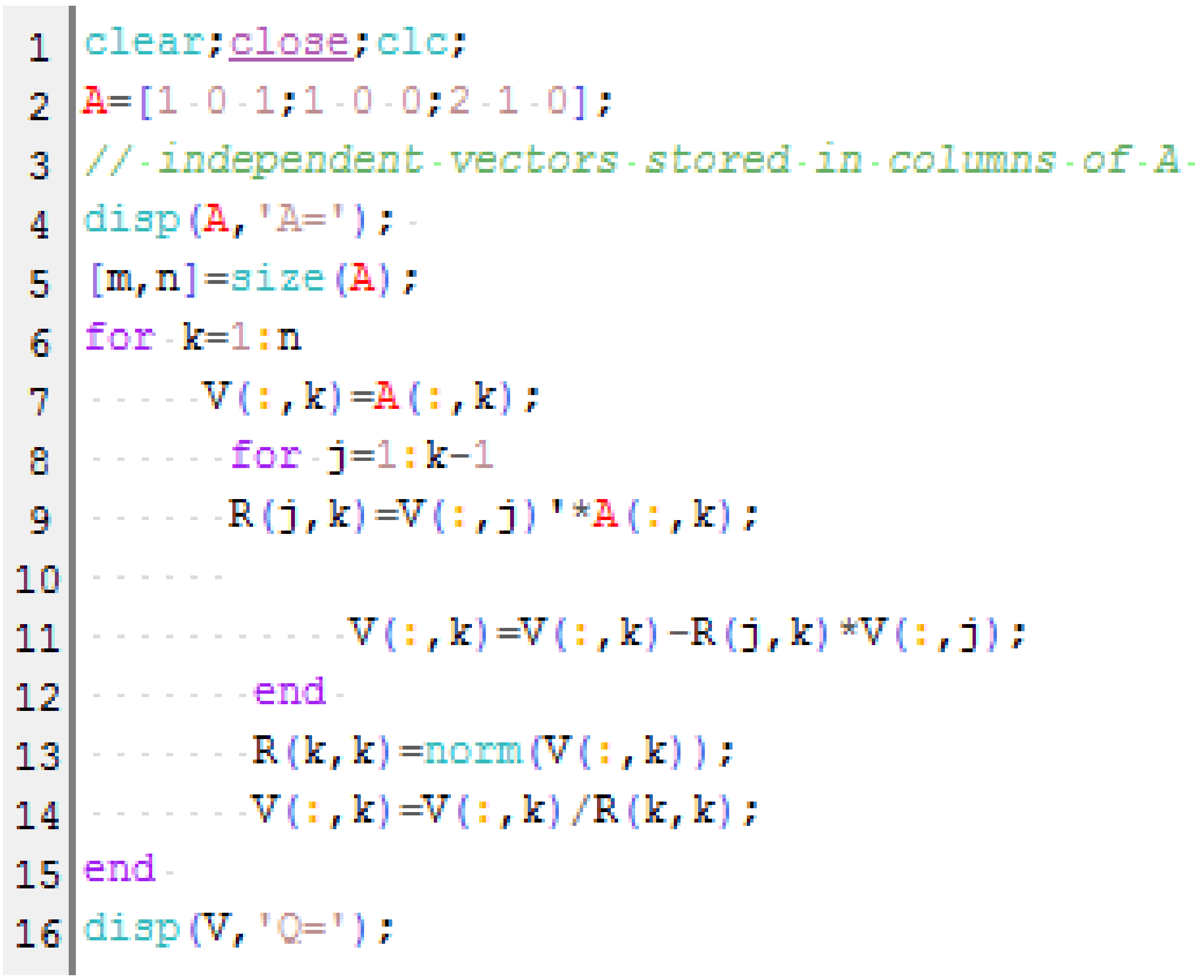
CLASS NUMBER -7

Topic: The Gram- Schmidt Orthogonalization

Procedure: Given a set of mutually independent vectors, we produce a set of orthonormal vectors by applying the Gram – Schmidt process.

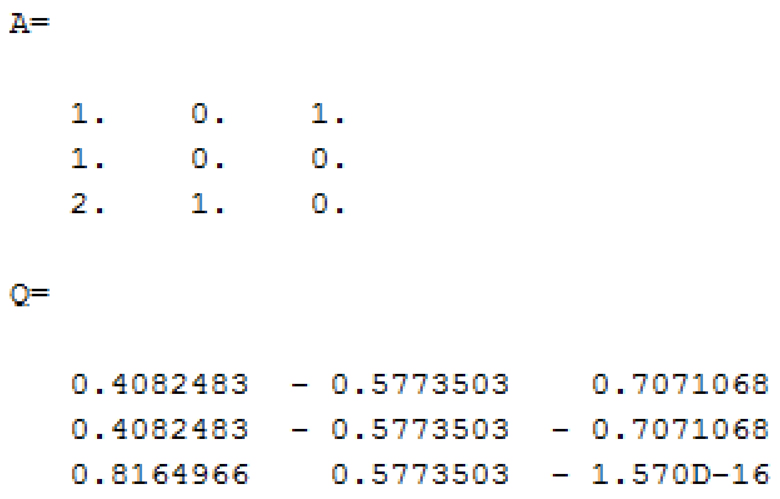
Example 9: Apply the Gram – Schmidt process to the vectors (1, 0, 1) , (1, 0, 0,) and (2, 1, 0) to produce a set of orthonormal vectors.

Solution:



```
clear;close;clc;
A=[1 0 1;1 0 0;2 1 0];
// independent vectors stored in columns of A 
disp(A,'A='); 
[m,n]=size(A);
for k=1:n
     V(:,k)=A(:,k);
      for j=1:k-1
      R(j,k)=V(:,j)'*A(:,k);
      
            V(:,k)=V(:,k)-R(j,k)*V(:,j);
       end 
       R(k,k)=norm(V(:,k));
       V(:,k)=V(:,k)/R(k,k);
end 
disp(V,'Q=');
```

Output:



```
A=

   1.    0.    1.
   1.    0.    0.
   2.    1.    0.

Q=

   0.4082483  - 0.5773503    0.7071068
   0.4082483  - 0.5773503  - 0.7071068
   0.8164966    0.5773503  - 1.570D-16
```

<u>Practice Problems:</u>

Apply the Gram – Schmidt process to the following set of vectors and find the orthogonal matrix:

1. (1, 1, 0) , (1, 0, 1) , (0, 1, 1)

2. (0, 0, 1) , (0, 1, 1) , (1, 1, 1)

3. (1, 0, 1) , (1, 1, 0) , (2, 1, 1)

CLASS NUMBER - 8

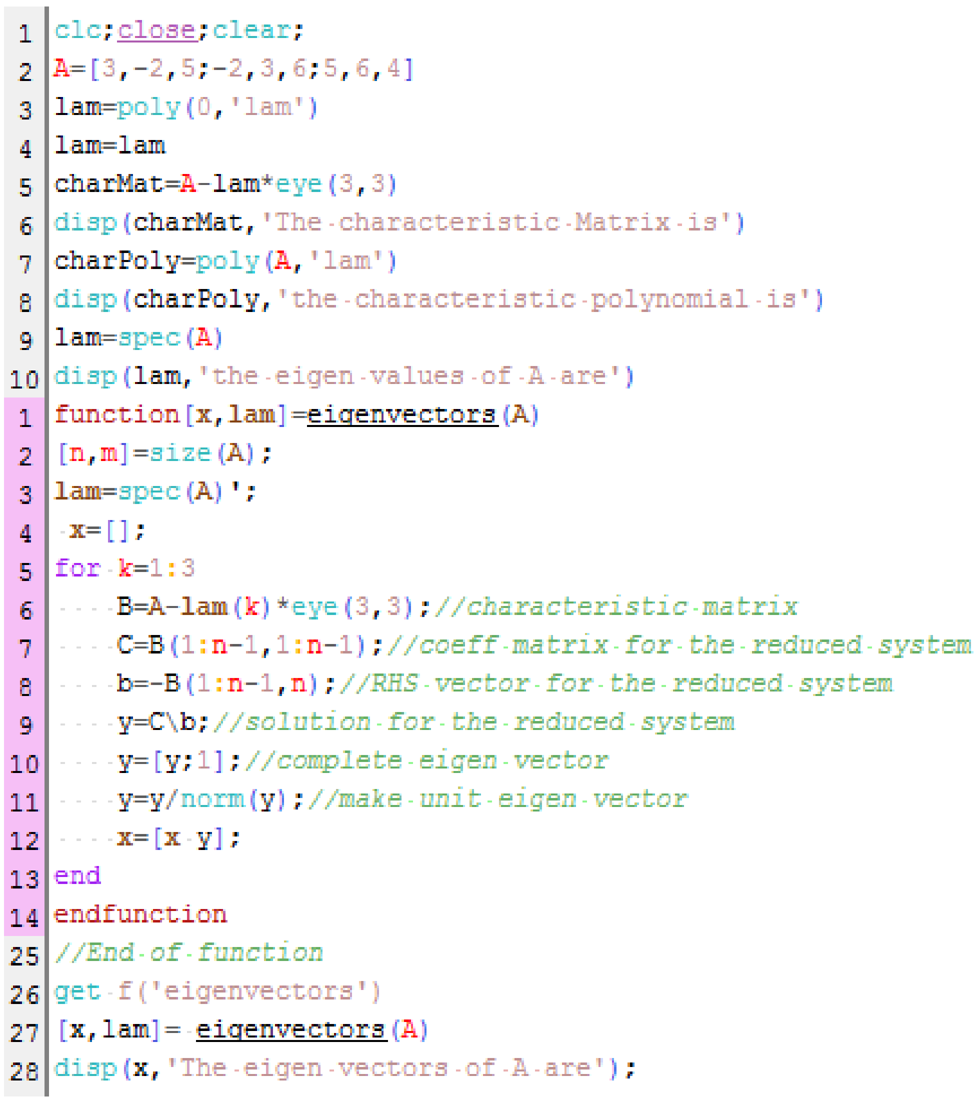
Topic: Eigen values and Eigen vectors of a given square matrix

Procedure: Given a square matrix A, we find the characteristic polynomial of A by expanding the matrix equation | A – λ I |. The Eigen values of A are obtained solving the characteristic equation | A – λ I | = 0. The corresponding Eigen vectors are obtained by solving the system of equations Ax = λx.

Example 10: Find the Eigen values and the corresponding Eigen vectors of

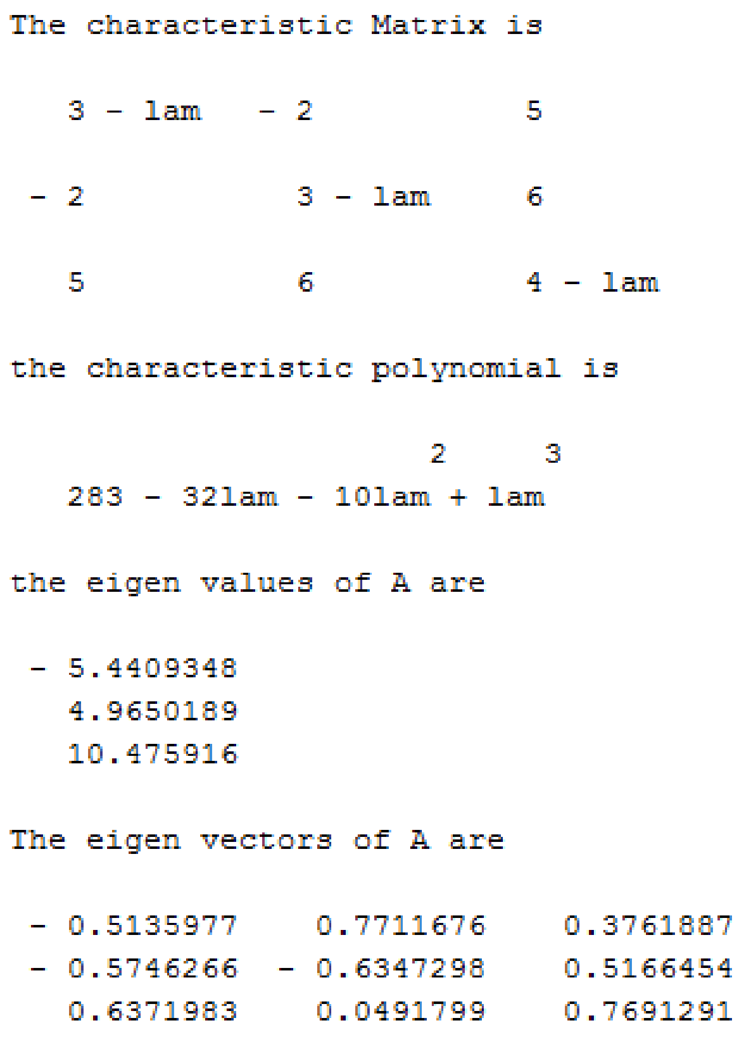
$$A=\begin{pmatrix} 3 & -2 & 5 \\ -2 & 3 & 6 \\ 5 & 6 & 4 \end{pmatrix}$$

Solution:



```
clc;close;clear;
A=[3,-2,5;-2,3,6;5,6,4]
lam=poly(0,'lam')
lam=lam
charMat=A-lam*eye(3,3)
disp(charMat,'The characteristic Matrix is')
charPoly=poly(A,'lam')
disp(charPoly,'the characteristic polynomial is')
lam=spec(A)
disp(lam,'the eigen values of A are')
function[x,lam]=eigenvectors(A)
[n,m]=size(A);
lam=spec(A)';
 x=[];
for k=1:3
    B=A-lam(k)*eye(3,3);//characteristic matrix
    C=B(1:n-1,1:n-1);//coeff matrix for the reduced system
    b=-B(1:n-1,n);//RHS vector for the reduced system
    y=C\b;//solution for the reduced system
    y=[y;1];//complete eigen vector
    y=y/norm(y);//make unit eigen vector
    x=[x y];
end
endfunction
//End of function
get f('eigenvectors')
[x,lam]= eigenvectors(A)
disp(x,'The eigen vectors of A are');
```

Output:



```
The characteristic Matrix is

   3 - lam    - 2            5

 - 2            3 - lam     6

   5            6            4 - lam

the characteristic polynomial is

                     2      3
   283 - 32lam - 10lam + lam

the eigen values of A are

 - 5.4409348
   4.9650189
   10.475916

The eigen vectors of A are

 - 0.5135977    0.7711676    0.3761887
 - 0.5746266  - 0.6347298    0.5166454
   0.6371983    0.0491799    0.7691291
```

<u>**PracticeProblems:**</u>

Find the Eigen values and the corresponding Eigen vectors of the following matrices:

(ι) $\begin{pmatrix} 8 & -6 & 2 \\ -6 & 7 & -4 \\ 2 & -4 & 3 \end{pmatrix}$ (2) $\begin{pmatrix} 5 & -6 & 2 \\ -6 & 4 & -4 \\ 2 & -4 & 0 \end{pmatrix}$ (3) $\begin{pmatrix} 2 & 2 & 1 \\ 1 & 3 & 1 \\ 1 & 2 & 2 \end{pmatrix}$