| | |
|---|---|
| **Name** | Usama Siddique |
| **Reg.#** | 2018−EE−413 |
| **Marks** | |

# LAB#02

# Introduction of Arrays, Two dimensional plots and Elementary sequence

## Arrays:

As mentioned earlier, the name MATLAB stands for Matrix Laboratory because MATLAB has been designed to work with *matrices.* A matrix is a rectangular object (e.g., a *table)* consisting of rows and columns. A *vector* is a special type of matrix, having only one row,or one column.

MATLAB handles vectors and matrices in the same way, but since vectors are easier to think about than matrices

A one-dimensional array is a list of number that is placed in a row or a column. The vector is created by typing the elements inside the square brackets [ ]

Variable name = [ type vector elements]

A vector is created with constant spacing by specifying the first term, the spacing, and the last term

Variable name = [first term: spacing: last term]

A vector in which the first element is *xi,* the last element is *xf,* and the number of elements is *n* is created by typing the linspace command:

Variable name =linspace (xi, xi, n)

A matrix is created by assigning the elements of the matrix to a variable. This is done by typing the elements, row by row, inside square brackets [ ]. First type the left bracket [ , then type the first row separating the elements with spaces or commas. To type the next row, type a semicolon or press Enter. Type the right bracket] at the end of the last row.

variable name = [ 1st row elements; 2nd row elements; 3rd row elements……....
last row elements]

Example:

```
>> a = [5 35 43; 4 76 81; 21 32 40]
```

A semicolon is typed before a new line is entered.

Rows of a matrix can also be entered as vectors using the notation for creating vectors with constant spacing,or the linspace command.

Example:

```
>> A = [1:2:11; 0:5:25; linspace (10,60,6); 67 2 43 68 4 13]
A =
     1    3    5    7    9   11
     0    5   10   15   20   25
    10   20   30   40   50   60
    67    2   43   68    4   13
```

In this example the first two rows were entered as vectors using the notation of constant spacing, the third row was entered using the linspace command, and in the last row the elements were entered individually.

## The zeros, ones and eye Commands:

The *zeros (m, n)*, the *ones (m, n)*, and *eye (n)* commands can be used to create matrices that have elements with special values. The *eye (n)* command creates a square matrix with *n* rows and *n* columns in which the diagonal elements are equal to **1**, and the rest of the elements are 0. This matrix is called the identity matrix.

Examples:

```
>> zr = zeros (3,4)
zr =
    0 0 0 0
    0 0 0 0
    0 0 0 0

>> ne=ones (3,5)

ne =
    1 1 1 1 1
    1 1 1 1 1
    1 1 1 1 1
>> idn= eye (5)
idn =
    0 0 0 0 0
    0 1 0 0 0
    0 0 1 0 0
    0 0 0 1 0
    0 0 0 0 1
```

**The Transpose Operator:-**

The transpose operator is applied by typing a single quote' following the variable to be transposed.

Examples:

```
>> C = [2 5 14 8 ; 9 5 32 1; 1 2 3 4]
C =
     2    5   14    8
     9    5   32    1
     1    2    3    4
>> D   = C'
D =
     2    9    1
     5    5    2
    14   32    3
     8    1    4
```

## Array Addressing:

Elements in an array (either vector or matrix) can be addressed individually or in subgroups.

## Vector:

The address of an element in a vector is its position in the row (or column). For a vector named ve, ve *(k)* refers to the element in position *k*. The first position is 1. For example, if the vector *ve* has nine elements: *ve=35* 46 78 2351481355 then
*ve (4)* = 23, *ve (7)* = 81, and *ve (1)* = 35.

**Example:**

```
>> VCT = [35 46 78 23 5 14 81 3 55]
        VCT =
           35 46 78 23 5 14 81   3 55
>> VCT(4)
        ans=   23

>> VCT(2)+VCT(8)


        ans=   49

>> MAT = [3 11 6 5; 4 7 10 2; 13 9 0 8]
        MAT =
           3      11  6     5
           4       7 10     2
           13    9     0     8
>> MAT(3,1)
        ans =
           13
>> MAT = [3 11 6 5; 4 7 10 2; 13 9 0 8]
        MAT =
           3      11  6     5
           4       7 10     2
           13    9     0     8
>> MAT(3,1)=20
        MAT =
           3      11  6     5
           4       7 10     2
           20    9     0     8
>> MAT(2,4)-MAT(1,2)
        ans =
           -9
```

## Using A Colon: In Addressing Arrays

A colon can be used to address a range of elements in a vector or a matrix.

**For a vector:**

*va (:)* Refers to all the elements of the vector *va* (either a row or a column vector). *va (m: n)* Refers to elements *m* through *n* of the vector *va.*

**Example:**

```
>> v = [4 15 8 12 34 2 50 23 11]
v =
15 8 12 34 2 50 23 11
>> u = v(3:7)
u =
   8 12 34 2 50
```

**For a matrix:**

*A(:,n)* Refers to the elements in all the rows of column *n* of the matrix *A.*
*A(n,:)* Refers to the elements in all the columns of row *n* of the matrix A.
*A(:,m:n)* Refers to the elements in all the rows between columns *m* and *n* of the matrix A.
*A(m:n,:)* Refers to the elements in all the columns between rows *m* and *n* of the matrix *A.*
*A(m:1},p:q)* Refers to the elements in rows *m* through *n* and columns *p* through *q* of the matrix A..

## Adding Elements to Existing Variables:

A variable that exists as a vector, or a matrix, can be changed by adding elements to it.

Example:

```
>> DF = 1:4
DF =
   1   2   3   4
>> DF(5:10)=10:5:35
DF =
     1   2   3   4 10     15 20 25 30 35
```

## Adding Elements to a Matrix:

Rows and/or columns can be added, to an existing matrix by assigning values to the new rows or columns. This can be done by assigning new values, or by appending existing variables. This must be done carefully since the size of the added rows or columns must fit the existing matrix. Examples are given below.

```
>> E = [1 2 3 4; 5 6 7 8]

    1   2    3   4
    5   6    7   8
 >> E(3,:)=[10:4:22]
 E =
    1   2    3   4
    5   6    7   8
    10 14 18 22
```

```
>> K = eye(3)
K =
    1    0    0
    0    1    0
    0    0    1
>> G = [E K]
G =
    1    2    3  4    1 0 0
    5    6    7  8    0 1 0
    10 14 18   22 0 0 1
```

## Built-in Functions for Handling Arrays:

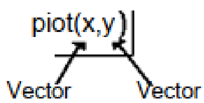
MATLAB has many built-in functions for managing and handling arrays. Some of these are listed below:

| Function | Description | Example |
|---|---|---|
| length (A) | Returns the number of elements in the vector A. | >> A = [5 9 2 4];<br>>> length(A)<br>Ans = 4 |
| size(A) | Returns a row vector [m, n], where m and n are the size *m* X *n* of the array A. | >> A = **[6** 1 4 0 12; 5 19 **6 8** 2];<br>>> size(A)<br>Ans =<br>2 5 |
| diag(A) | When A is a matrix, creates a vector from the diagonal elements of A. | >> A = [1 2 3; 4 5 **6**; 7 **8** 9]<br>A =<br>1 2 3<br>4 5 **6**<br>7 **8** 9<br>>> vec = diag(A)<br>vec = 1 5 9 |

## Two Dimensional Plots

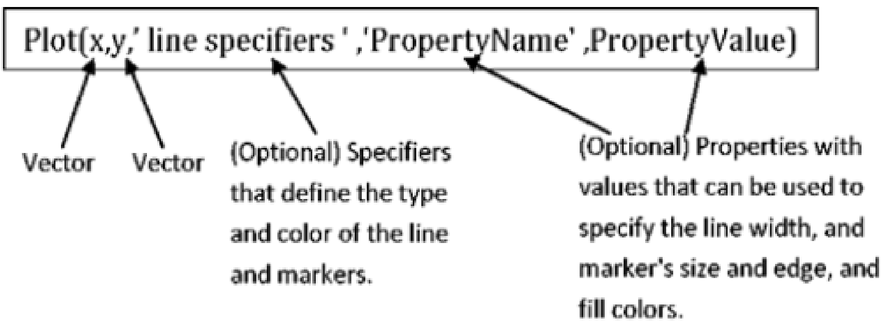
***The* plot *Command:-***

The plot command is used to create two-dimensional plots. The simplest form of the command is: The arguments *x* and *y* are each are vector (one-dimensional array). Both vectors must have the same




number of elements.

The plot command has additional optional arguments that can be used to specify the color and style of the line and the color and type of markers, if any are desired. With these options the command has the Graphs *in* the Same Plot:-




ln many situations there is a need to make several graphs in the same plot. There are three methods to plot multiple graphs in one figure.

• By using the plot command
• By using the hold on, hold off commands
• By using the line command.

## Formatting a Plot Using Commands:

The formatting commands are entered after the plot or the fplot commands. The various formatting commands are:

**The xlabel and ylabel commands:**

Labels can be placed next to the axes with the xlabel and ylabel commands which have the form:

xlabel ('text as string')

ylabel ('text as string')

**The title command:**

A title can be added to the plot with the command:

title ('text as string') [The text is placed at the top of the figure as a title.]

**The text command:**

A text label can be placed in the plot with the text or gtext commands:

text(x,y,'text as string')

gtext('text as string')

The text command places the text in the figure such that the first character is Positioned at the point with the coordinates x, y (according to the axes of the figure).

The gtext command places the text at a position specified by the user. When the command is executed, the Figure Window opens and the user specifies the Position with the mouse.

## Legend command:

The legend command places a legend on the plot. The legend shows a sample of the line type of each graph that is plotted, and places a label, specified by the user, beside the line sample. The command is:

legend ('string1', 'stringl', ............................................ ,pos )

The strings are the labels that are placed next to the line sample

## The axis command:

When the plot (x, y) command is executed, MATLAB creates axes with limits that are based on the minimum and maximum values of the elements of x and y. The axis command can be used to change the range and the appearance of the axes. In many situations a graph looks better if the range of the axes extend beyond the range of the data. The following are some of the possible forms of the axis command:

axis ([$x_{min}$, $x_{max}$]) Sets the Iimits of the x axis (xmin and xmax are numbers).

axis ([$x_{min}$, $x_{max}$, $y_{min}$, $y_{max}$]) Sets the limits of both the x and y axes.

## Plotting multiple plots on the same page:

Command is: subplot (m, n, p)

The command divides the Figure Window (page when printed) into *m* x *n* rectangular subplots where plots will be created. The subplots are arranged like elements in a *m* X *n* matrix where each element is a subplot. The subplots are numbered from 1 through *m. n.* The upper left is 1 and the lower right is the number *m* x *n.* The numbers increase from left to right within a row, from the first row to the last. The command subplot (m, n, p) makes the subplot P current.

## Elementary Sequence

A discrete time signal is represented as a sequence of numbers, called samples. These samples are denoted by *x(n)* where the variable n is integer valued and represents in discrete instances in time. An example of a discrete time signal is:

x(n) = {2 ,1, -1 **,0** ,1 ,4 ,3 ,7} ... (1)

where the up arrow indicates the sample at n = **0**

In MATLAB, a finite duration sequence is represented by a row vector. However, such a vector does not have any information about sample position n. Therefore, a correct representation *of x(n)* would require two vectors, one each for x and *n*,

To represent the sequence defined in eq1, the following MATLAB command can be used:

>> n = [-3, -2, -1,0,1,2,3,4] x= [2,1, -1,0,1,4,3,7]

We use several elementary sequences in digital signal processing for analysis purposes. Their definitions and MATLAB representations are given below.

## Unit Sample Sequence:

$$\delta(n) = \begin{cases} 1, & n = 0 \\ 0, & n \neq 0 \end{cases} = \left\{ \begin{matrix} \ldots\ldots\ldots, 0, 0, 1, 0, 0, \ldots\ldots \\ \uparrow \end{matrix} \right\}$$

In MATLAB the function zeros (1, N) generates a row vector of N zeros, which can be used to implement δ (n) over a finite interval. However, the logical relation n==0 is an elegant way of implementing δ (n) . For example, to implement

$$\delta(n - n_o) = \begin{cases} 1, & n = n_o \\ 0, & n \neq n_o \end{cases}$$

over the $n_1 \le n_0 \le n_2$ interval, we will use the following MATLAB function.

```
function [x,n] = impseq(n0,n1,n2)
% Generates x(n) = delta(n-nO);     n1 <= n <= n2
% ----------------------------------------------
% [x,n] = impseq(n0,n1,n2)
%
n= [n1:n2];
x = [(n-n0) == 0];
```

MATLAB Script:-

```
% Generation of a Unit Sample Sequence
% Generate a vector from -10 to 20
[x,n]=impseq(1,-10,20)
%plot the unit sample sequence
stem(n,u);
xlabel('time index n');ylabel('Amplitude');
title('Unit Sample Sequence');
axis([-10 20 0 1.2]);
```

## Unit Step Sequence:

$$u(n) = \begin{cases} 1, & n \ge 0 \\ 0, & n < 0 \end{cases} = \left\{ \begin{matrix} \ldots\ldots, 0, 0, 1, 1, 1, \ldots\ldots \\ \uparrow \end{matrix} \right\}$$

In MATLAB the function **ones(1,N)** generates a row vector of *N* ones. It can be used to generate u( *n)* over a finite interval. Once again an elegant approach is to use the logical relation n>=0. To implement

$$u(n - n_o) = \begin{cases} 1, & n \ge n_o \\ 0, & n < n_o \end{cases}$$

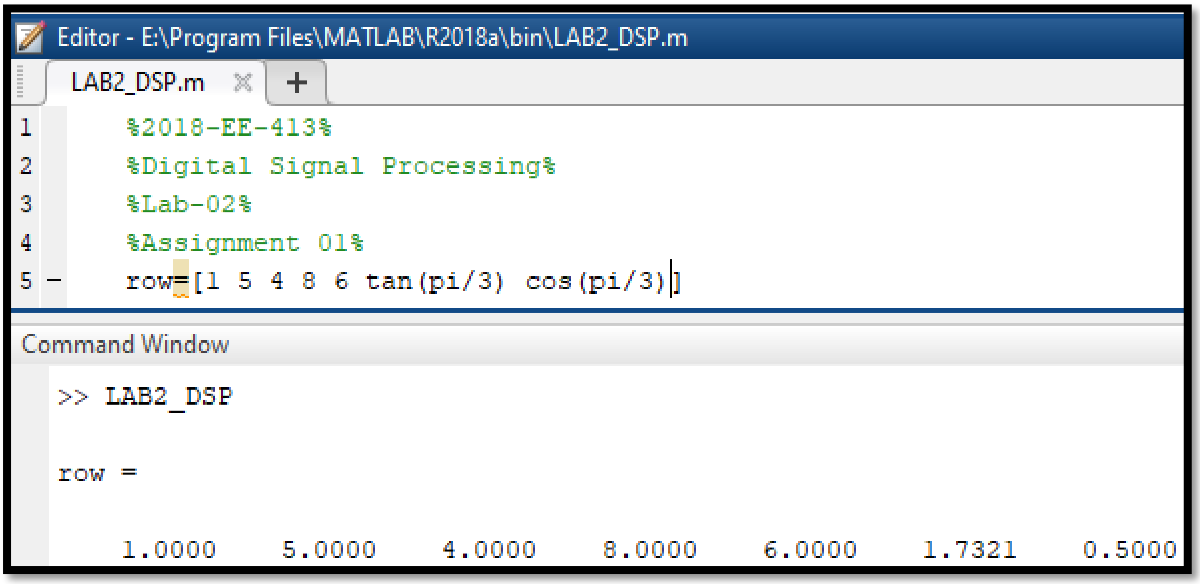
over the $n_1 \le n_0 \le n_2$ interval, we will use the following MATLAB function.

```
function [x,n] = stepseq(n0,n1,n2)
% Generates x(n)= u(n-nO); n1 <= n <= n2
%------------------------------------------------------------
% [x,n] = stepseq(n0,n1,n2)
%
n = [n1:n2]; x = [(n-n0) >= 0];
```

## Task#01:

Create a row vector that has the elements: 1, 5, 4, 8, 6, tan(pi/3) and cos(pi/3).

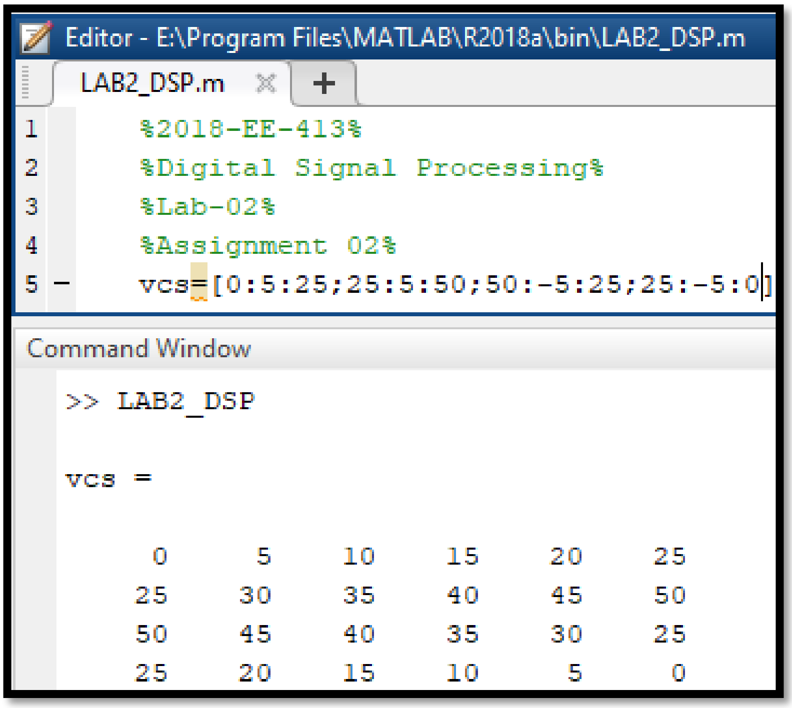
## Solution:




## Task#02:

Create a vector with constant spacing by specifying the first term, the spacing, and the last term.

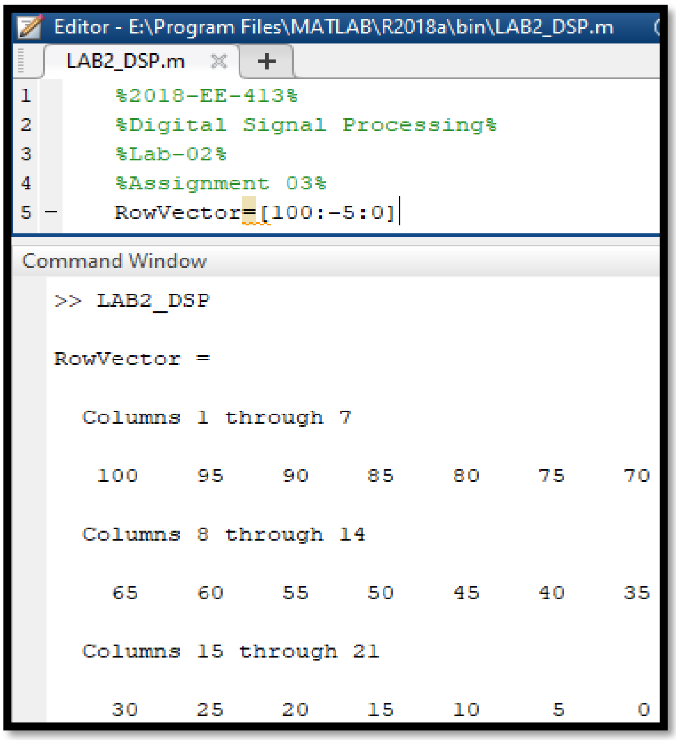
## Solution:



## Task#03:

Create a row vector in which the first element is 100, the elements decrease with increments of -5 and the last element is 0.

## Solution:

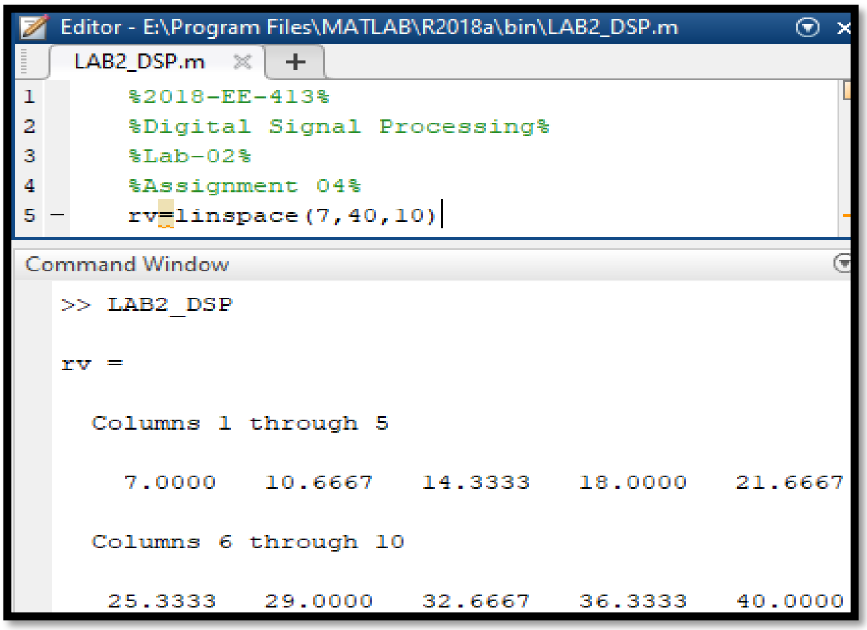


```
Editor - E:\Program Files\MATLAB\R2018a\bin\LAB2_DSP.m
LAB2_DSP.m
%2018-EE-413%
%Digital Signal Processing%
%Lab-02%
%Assignment 03%
RowVector=[100:-5:0]

Command Window
>> LAB2_DSP

RowVector =

  Columns 1 through 7

   100    95    90    85    80    75    70

  Columns 8 through 14

    65    60    55    50    45    40    35

  Columns 15 through 21

    30    25    20    15    10     5     0
```

## Task#04:

Create a row vector with 10 equally spaced elements in which the first element is 7 and the last element is 40.

## Solution:

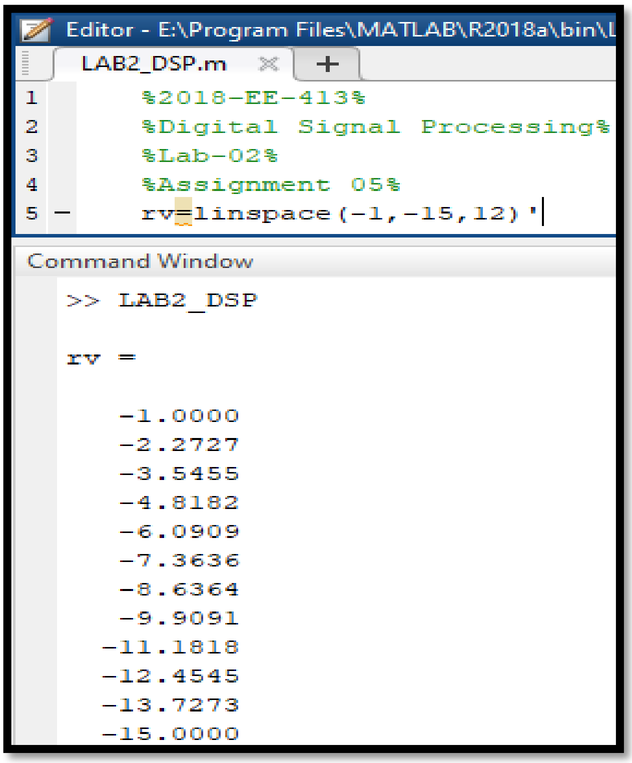


```
Editor - E:\Program Files\MATLAB\R2018a\bin\LAB2_DSP.m
LAB2_DSP.m
%2018-EE-413%
%Digital Signal Processing%
%Lab-02%
%Assignment 04%
rv=linspace(7,40,10)

Command Window
>> LAB2_DSP

rv =

  Columns 1 through 5

    7.0000   10.6667   14.3333   18.0000   21.6667

  Columns 6 through 10

   25.3333   29.0000   32.6667   36.3333   40.0000
```

## Task#05:

Create a column vector with 12 equally spaced elements in which the first element is -1 and the last element is -15.

## Solution:

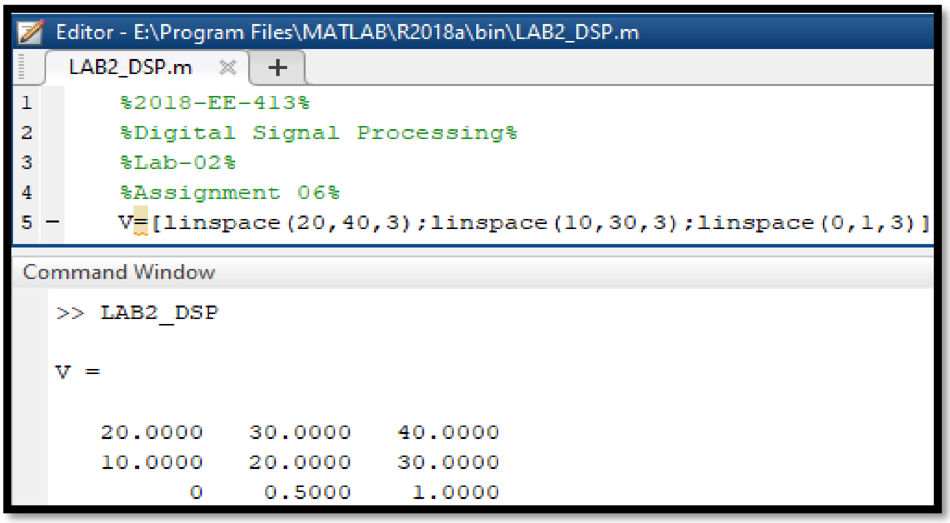



## Task#06:

Create the matrix shown below by using the vector notation for creating vectors with constant linspace command when entering the rows.

$$A = \begin{bmatrix} 20 & 30 & 40 \\ 10 & 20 & 30 \\ 0 & 0.5 & 1 \end{bmatrix}$$

## Solution:

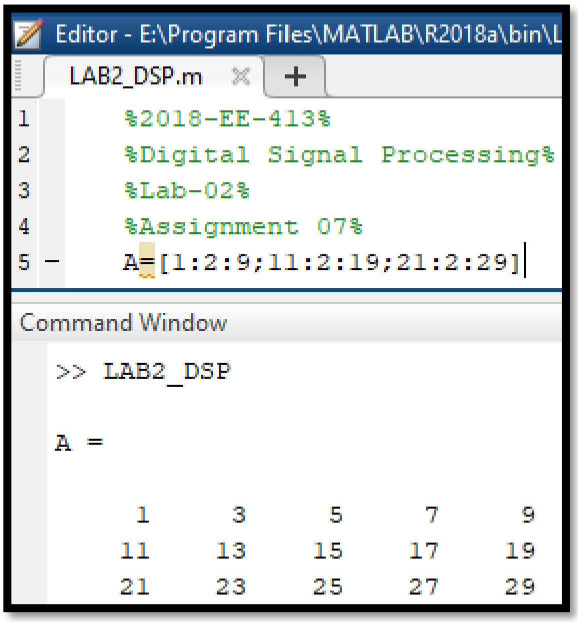


## Task#07:

Create the following matrix *A:*

$$A = \begin{bmatrix} 1 & 3 & 5 & 7 & 9 \\ 11 & 13 & 15 & 17 & 19 \\ 21 & 23 & 25 & 27 & 29 \end{bmatrix}$$

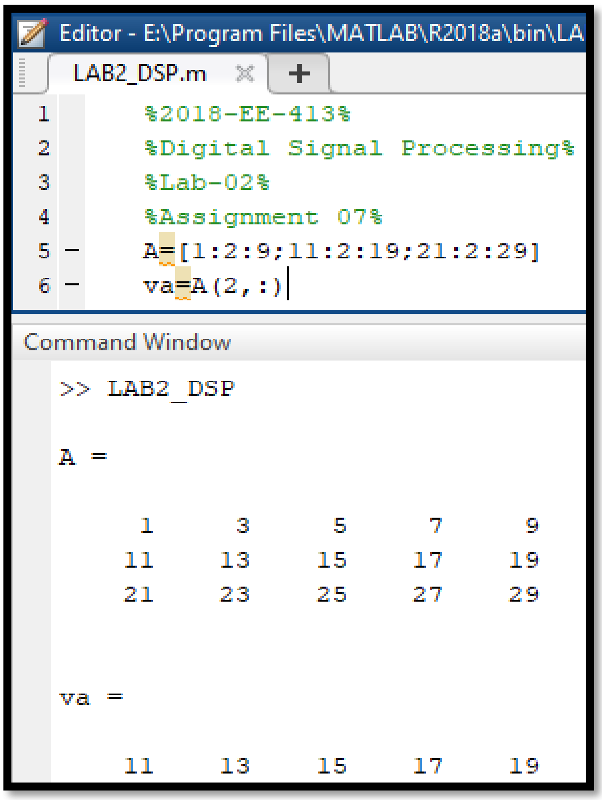
## Solution:




Use the matrix *A* to:

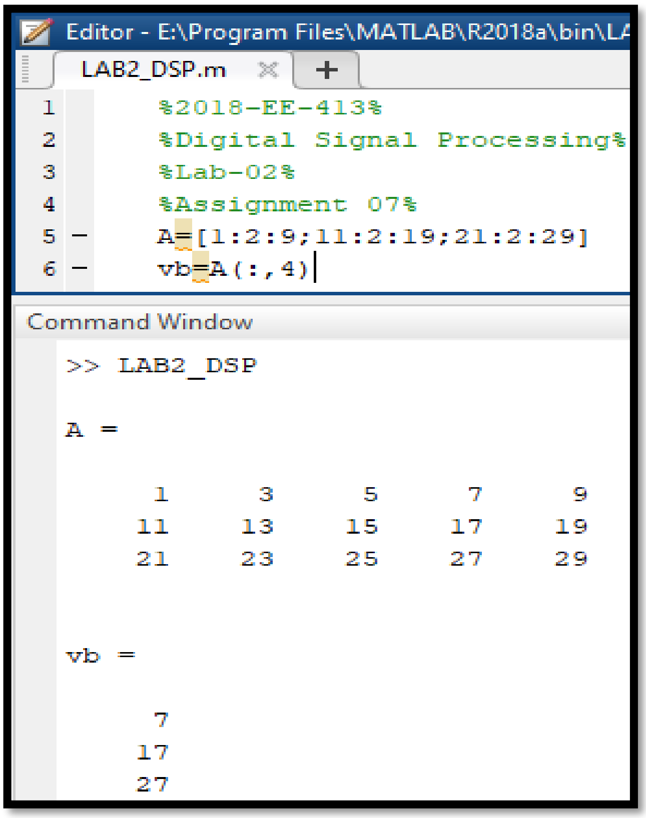
***a)*** Create a five-element row vector named va that contains the elements of the second row of *A*.

## Solution:



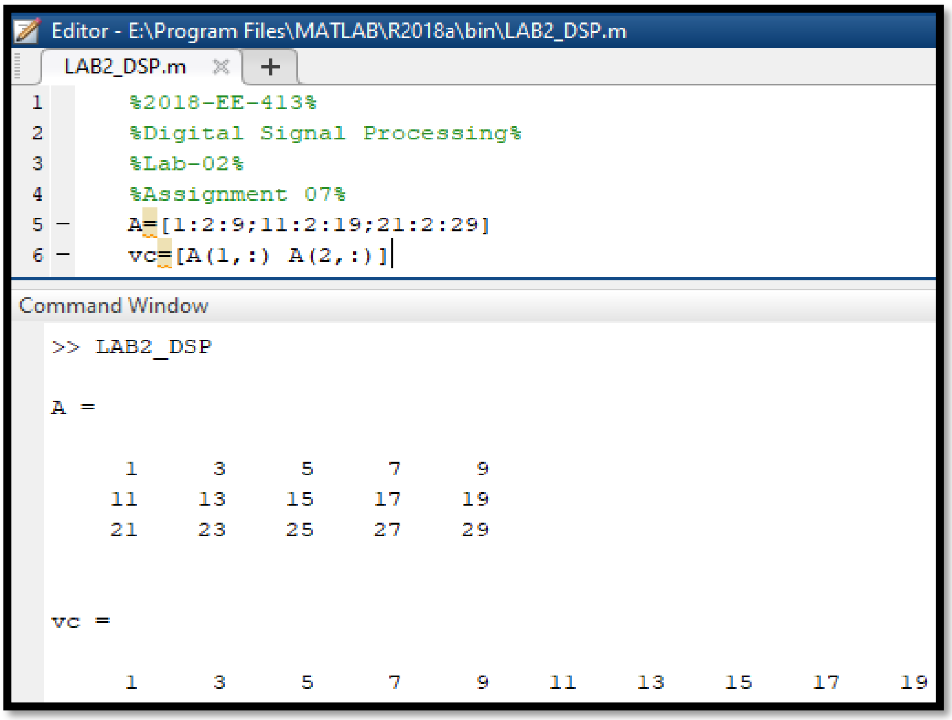
**b)** Create a three-element row vector named vb that contains the elements of the fourth column of A.

**Solution:**




***c)*** Create a ten-element row vector named vc that contains the elements of the first and second rows of *A*.

**Solution:**



***d)*** Create a six-element row vector named vd that contains the elements of the second and fifth columns of *A*.

**Solution:**

```
Editor - E:\Program Files\MATLAB\R2018a\bin\LAB2_DSP
LAB2_DSP.m

%2018-EE-413%
%Digital Signal Processing%
%Lab-02%
%Assignment 07%
A=[1:2:9;11:2:19;21:2:29]
vd=[A(:,2)' A(:,5)']
```

```
Command Window
>> LAB2_DSP

A =

     1     3     5     7     9
    11    13    15    17    19
    21    23    25    27    29


vd =

     3    13    23     9    19    29
```

**Task#08:**

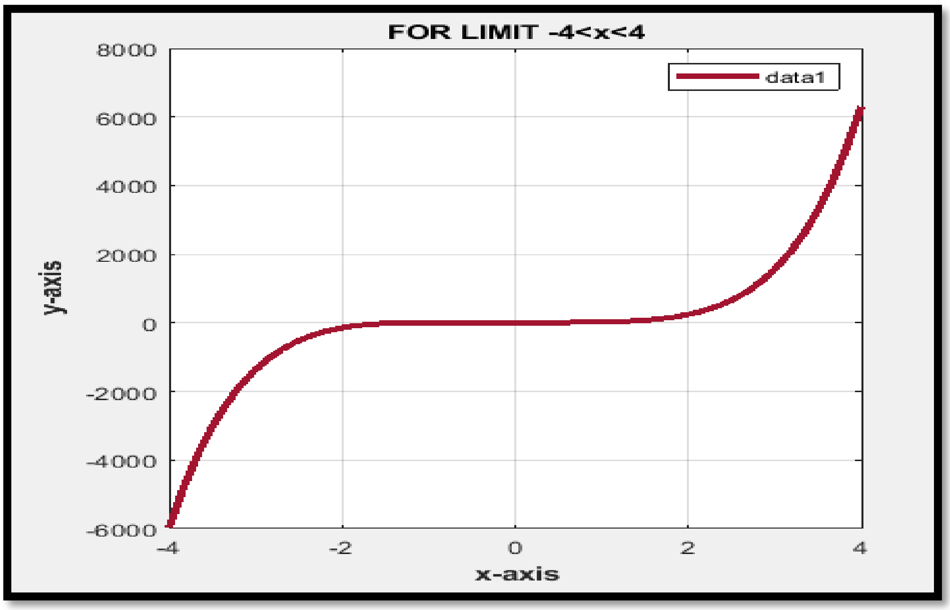
Solve the following problems in MATLAB Command Window.

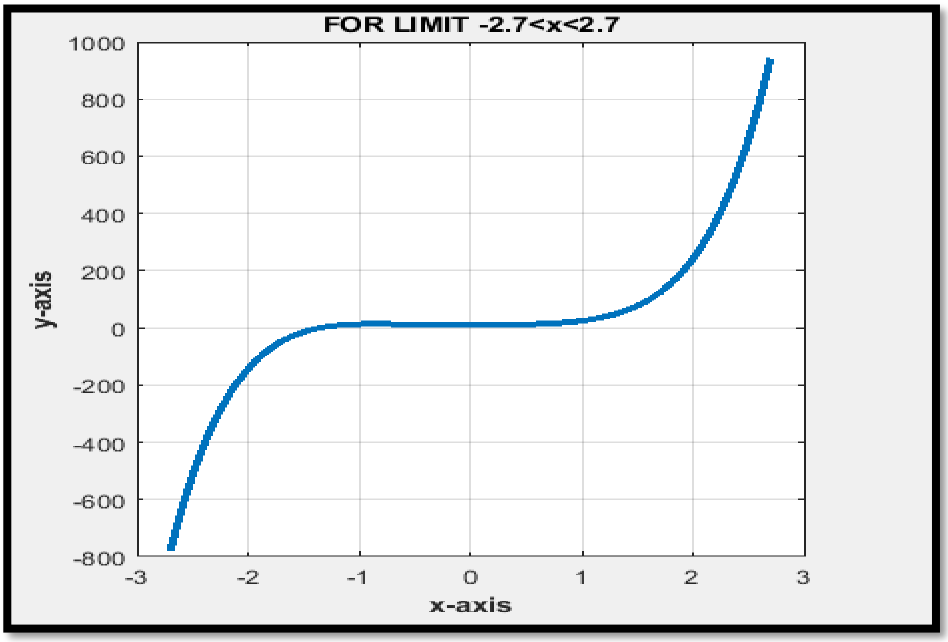
**1)** Make two separate plots of the function $f(x) = 6x^5 + 10x^2 + 9$, one plot for -4≤x≤4, and one for -2.7≤x≤2.7.

**Solution:**

```
Editor - E:\Program Files\MATLAB\R2018a\bin\LAB2_DSP.m
LAB2_DSP.m

%2018-EE-413%
%Digital Signal Processing%
%Lab-02%
%Assignment 08%
x=linspace(-4,4,50)
fx=(6*x.^5)+(10*x.^2)+9
plot(x,fx)
grid on
xlabel('x-axis');ylabel('y-axis');title('FOR LIMIT -4<X<4');
```

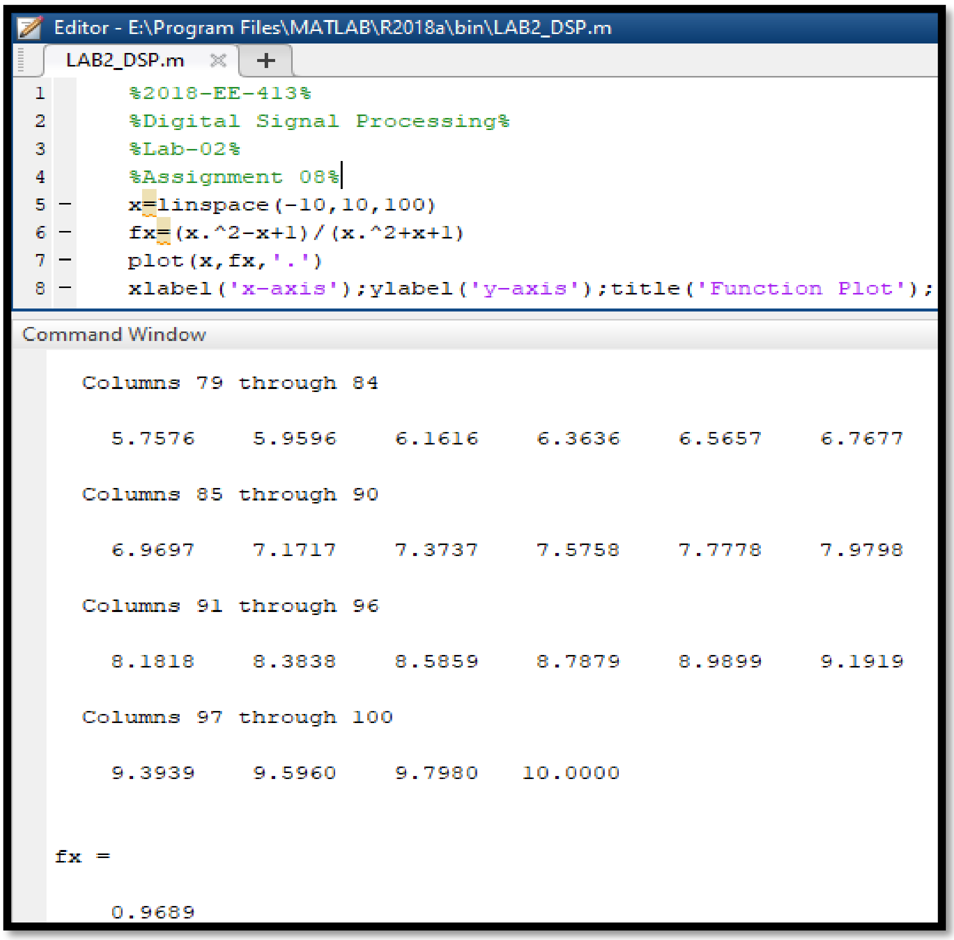
```
fx =

  1.0e+03 *

 Columns 1 through 13

  -5.9750   -4.8322   -3.8696   -3.0653   -2.3989   -1.8521   -1.4080   -1.0517   -0.7696   -0.5497   -0.3811   -0.2545   -0.1618

 Columns 14 through 26

  -0.0957   -0.0504   -0.0208   -0.0026    0.0075    0.0122    0.0136    0.0131    0.0119    0.0106    0.0096    0.0091    0.0091

 Columns 27 through 39

   0.0096    0.0107    0.0126    0.0157    0.0206    0.0283    0.0405    0.0591    0.0869    0.1272    0.1842    0.2631    0.3697

 Columns 40 through 50

   0.5112    0.6957    0.9328    1.2330    1.6085    2.0728    2.6410    3.3297    4.1575    5.1446    6.3130
```



**2)** Plot the function $f(x) = \frac{x^2-x+1}{x^2+x+1}$ for $-10 \leq x \leq 10$

## Solution:



Editor - E:\Program Files\MATLAB\R2018a\bin\LAB2_DSP.m

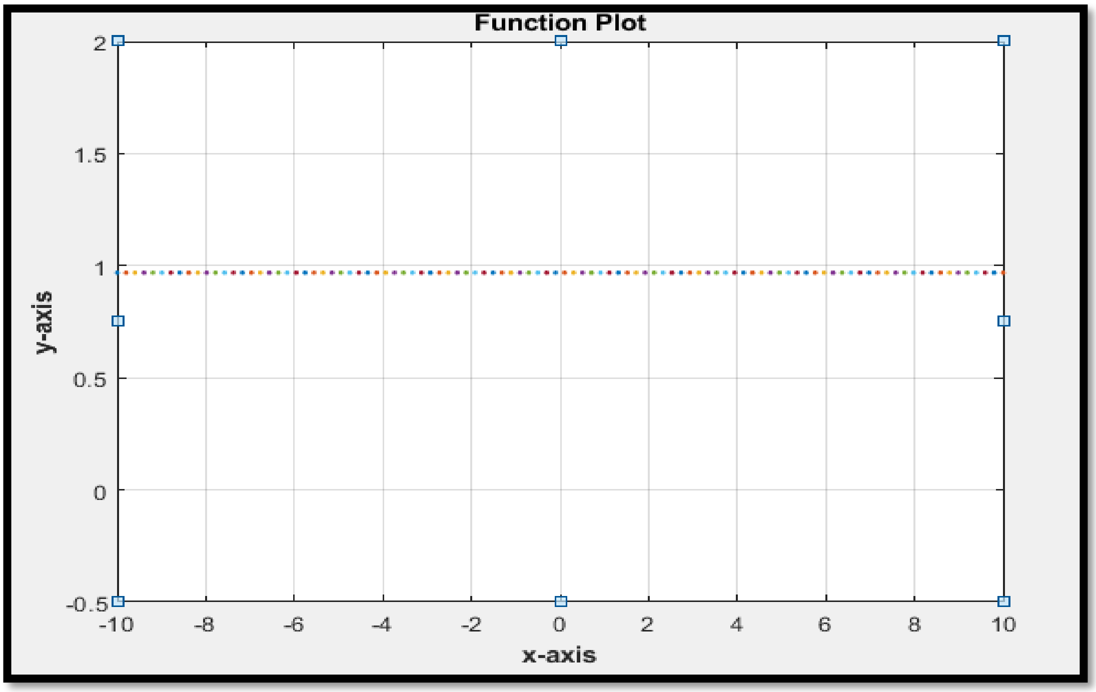
LAB2_DSP.m

```
%2018-EE-413%
%Digital Signal Processing%
%Lab-02%
%Assignment 08%
x=linspace(-10,10,100)
fx=(x.^2-x+1)/(x.^2+x+1)
plot(x,fx,'.')
xlabel('x-axis');ylabel('y-axis');title('Function Plot');
```

Command Window

```
  Columns 79 through 84

    5.7576    5.9596    6.1616    6.3636    6.5657    6.7677

  Columns 85 through 90

    6.9697    7.1717    7.3737    7.5758    7.7778    7.9798

  Columns 91 through 96

    8.1818    8.3838    8.5859    8.7879    8.9899    9.1919

  Columns 97 through 100

    9.3939    9.5960    9.7980   10.0000


fx =

    0.9689
```

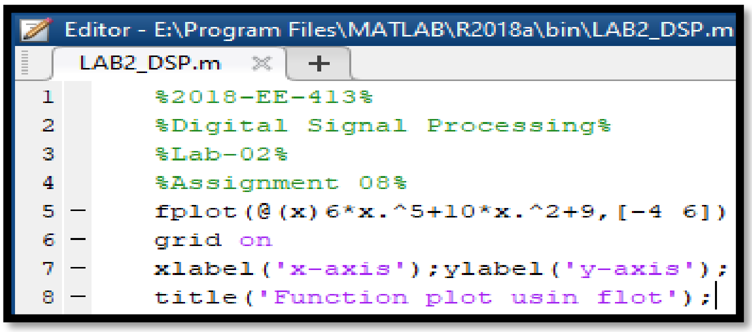
## Plot:



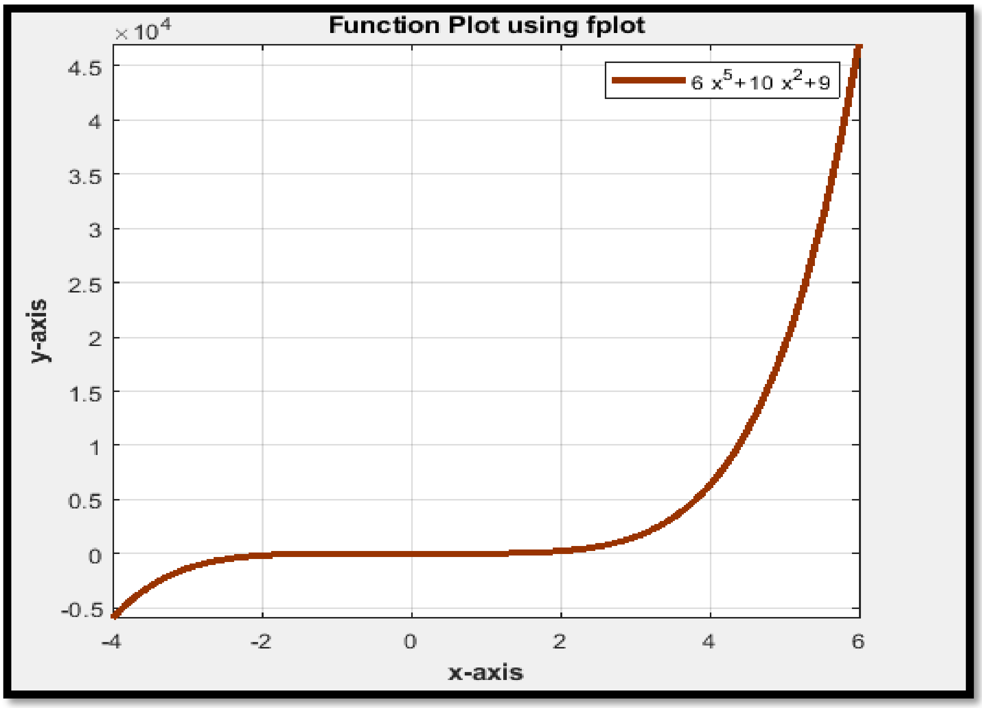
**3)** Use the fplot command to plot the function: $f(x) = 6x^5 + 10x^2 + 9$

In the domain $-4 \le x \le 6$

## Solution:




```
%2018-EE-413%
%Digital Signal Processing%
%Lab-02%
%Assignment 08%
fplot(@(x)6*x.^5+10*x.^2+9,[-4 6])
grid on
xlabel('x-axis');ylabel('y-axis');
title('Function plot usin flot');
```




## Task#09:

Plot the following data in MATLAB

| **Year** | **2000** | **2001** | **2002** | **2003** | **2004** | **2005** |
|---|---|---|---|---|---|---|
| **Minimum Marks** | 23 | 25 | 20 | 30 | 28 | 35 |
| **Maximum Marks** | 95 | 92 | 100 | 85 | 90 | 93 |

***a)*** Label x Axis as year and y axis as marks
***b)*** The color of the Minimum marks graph should be green and maximum marks graph should be red.
***c)*** The range of x Axis should be 2000 to 2005 and y axis 20 to 100
***d)*** The title of the graph should be "six years marks record".

## Solution:

Editor - E:\Program Files\MATLAB\R2018a\bin\LAB2_DSP.m

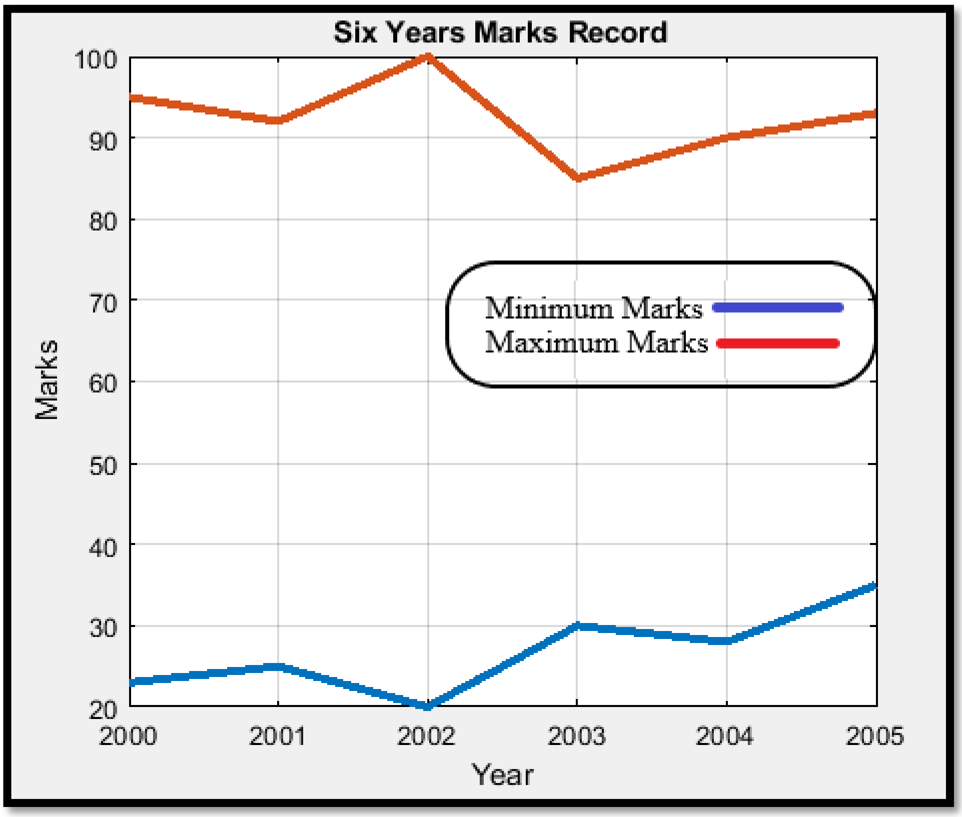
LAB2_DSP.m

```
%2018-EE-413%
%Digital Signal Processing%
%Lab-02%
%Assignment 09%
x=linspace(2000,2005,6)
minmarks=[23 25 20 30 28 35]
maxmarks=[95 92 100 85 90 93]
plot(x,minmarks,x,maxmarks);
xlim([2000 2005]);ylim([20 100]);
grid on
xlabel('Year');ylabel('Marks');
title('Six Years Marks Record');
```

Command Window

```
>> LAB2_DSP

x =

        2000         2001         2002         2003         2004         2005


minmarks =

    23    25    20    30    28    35


maxmarks =

    95    92   100    85    90    93
```

## Plot:



## Task#10:

Generate and plot the following sequence

## Functions being Used:

Editor - E:\Program Files\MATLAB\R2018a\bin\stepseq.

LAB2_DSP.m | stepseq.m | imseq.m

```
function [x,n]=stepseq(n0,n1,n2)
n=[n1:n2];
x=[(n-n0)>=0];
end
```

Editor - E:\Program Files\MATLAB\R2018a\bin\imseq

LAB2_DSP.m | stepseq.m | imseq.m

```
function [x,n]=imseq(n0,n1,n2)
n=[n1:n2];
x=[(n-n0)==0];
end
```

$u(n-5) \qquad -20 \le n \le 10$

## Solution:

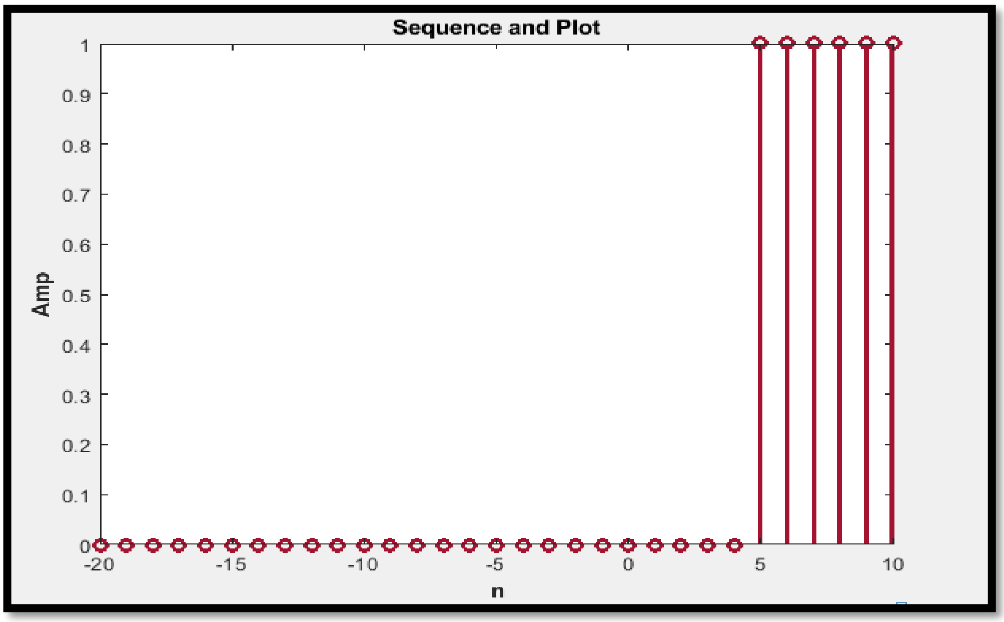
```
%2018-EE-413%
%Digital Signal Processing%
%Lab-02%
%Assignment 10%
[x,n]=stepseq(5,-20,10)
plot(n,x)
stem(n,x)
xlabel('n');ylabel('Amp');
title('Sequence and Plot');
```

## Sequence Generated:

Command Window

```
>> LAB2_DSP

x =

  1×31 logical array

  Columns 1 through 23

   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0

  Columns 24 through 31

   0   0   1   1   1   1   1   1


n =

  Columns 1 through 15

   -20   -19   -18   -17   -16   -15   -14   -13   -12   -11   -10    -9    -8    -7    -6

  Columns 16 through 30

    -5    -4    -3    -2    -1     0     1     2     3     4     5     6     7     8     9

  Column 31

    10
```

## Plot:

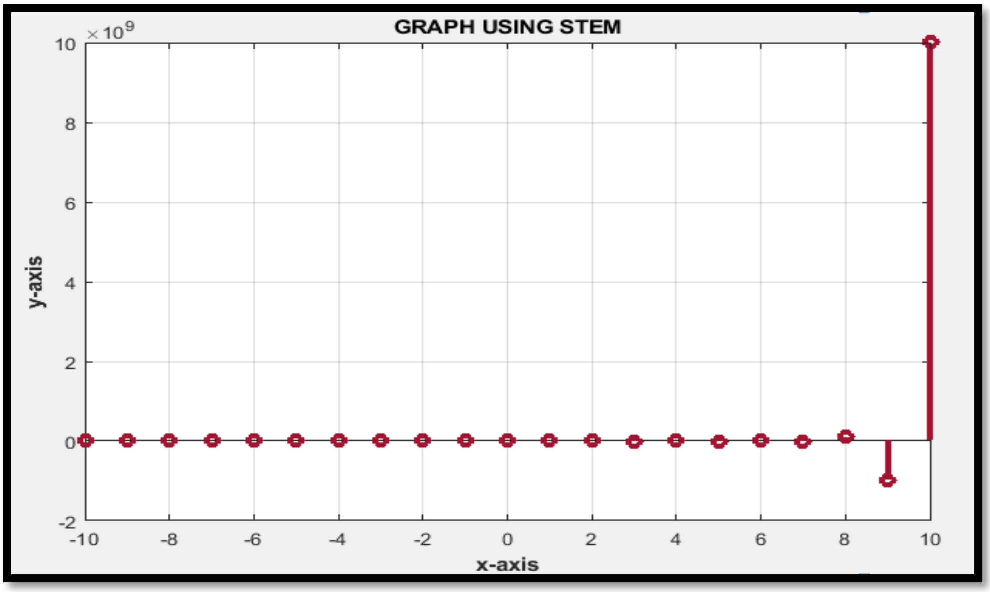



$$x(n) = (-10)^n \quad -10 \le n \le 10$$

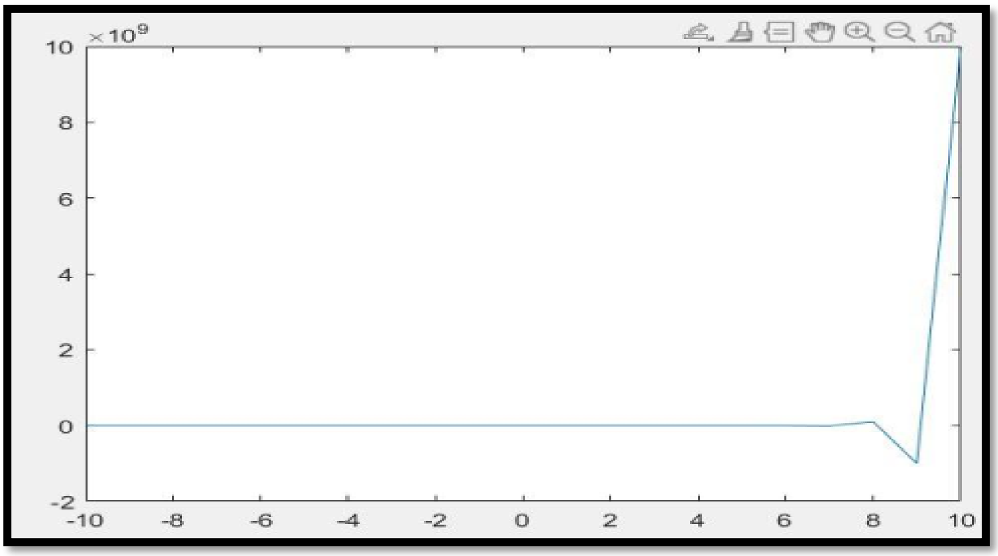
## Solution:

```
%2018-EE-413%
%Digital Signal Processing%
%Lab-02%
%Assignment 10%
n=[-10:10];
x=(-10).^n
stem(n,x)
```

## PLOT:



## Graph using Plot Command:

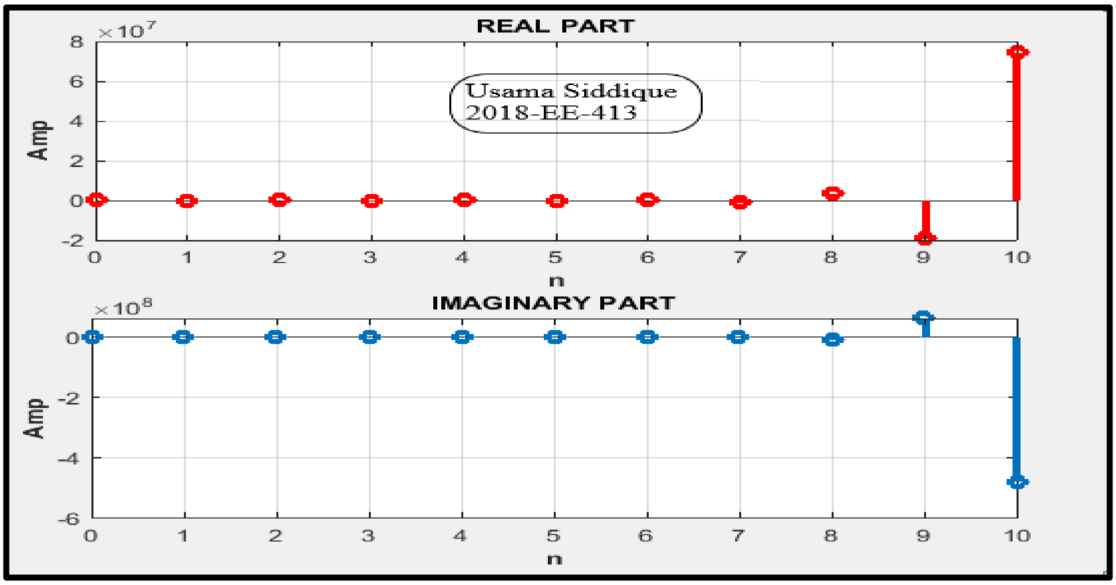



$$x(n) = \exp\,[(2 + j3)\,n]\,,\quad 0 \le n \le 10$$

## Solution:

```
%2018-EE-413%
%Digital Signal Processing%
%Lab-02%
%Assignment 10%
n=[0:10];
x=exp((2+3j)*n);
subplot(2,1,1);
stem(n,real(x));
xlabel('n');ylabel('Amp');
title('Real Part');
subplot(2,1,2);
stem(n,imag(x));
xlabel('n');ylabel('amp');
title('Imaginary Part');
```

## Plot:



$$x(n) = n[u(n) - u(n-10)] + 10e^{-.03(n-10)}[u(n-10) - u(n-20)] \qquad 0 \le n \le 20$$

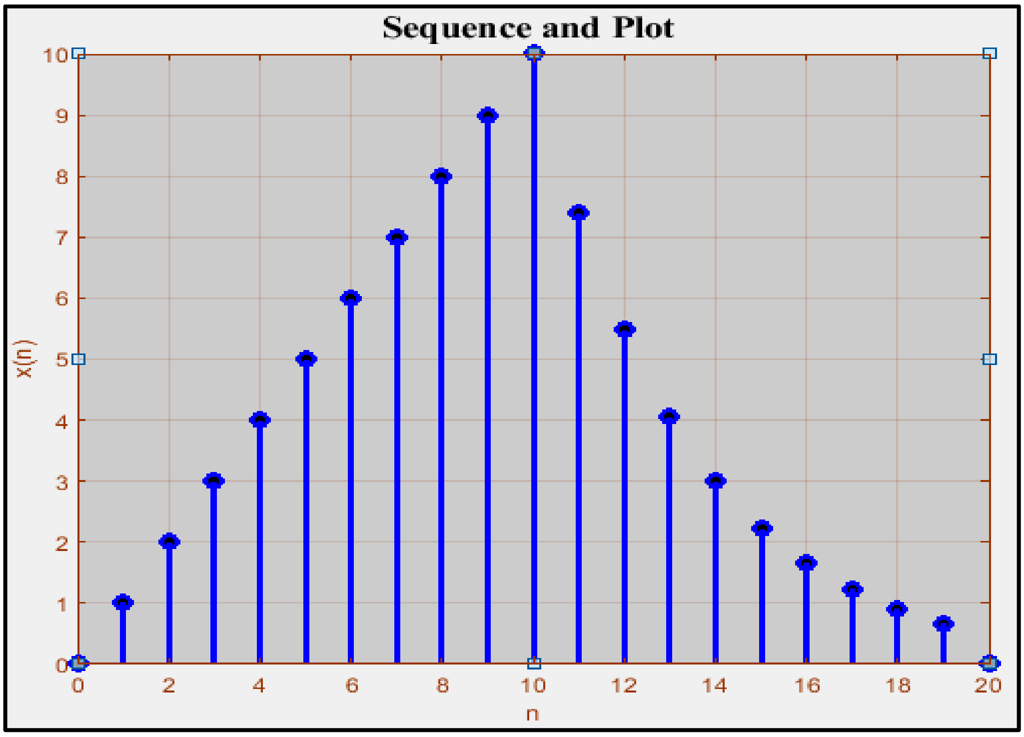
**Solution:**

Editor - E:\Program Files\MATLAB\R2018a\bin\LAB2_DSP.m

LAB2_DSP.m | stepseq.m | imseq.m

```
%2018-EE-413%
%Digital Signal Processing%
%Lab-02%
%Assignment 10%
n=[0:20];
x1=n.*(stepseq(0,0,20)-stepseq(10,0,20));
x2=10*exp(-0.3*(n-10)).*(stepseq(10,0,20)-stepseq(20,0,20));
x=x1+x2;
subplot(2,2,3);
stem(n,x);
xlabel('n');ylabel('x(n)');
title('Sequence and Plot');
```

**Plot:**




## Conclusion:

In this lab, we have learned the Introduction of Arrays, Two dimensional plots and Elementary sequence. MATLAB has been designed to work with *matrices.* A matrix is a rectangular object (e.g., a *table)* consisting of rows and columns. A *vector* is a special type of matrix, having only one row,or one column. MATLAB handles vectors and matrices in the same way, but since vectors are easier to think about than matrices.