Signals & Systems Laboratory

CSE- 301L

Lab # 02

OBJECTIVES OF THE LAB

In this lab, we will cover the following topics:

- *Built in Matrix Functions*
- *Indexing Matrices*
- *Sub Matrices*
- *Matrix element level operations*
- *Round Floating Point numbers to Integers*

2.1 MATRICES

MATLAB works with essentially only one kind of object, a rectangular numerical matrix possibly, with complex entries. Every MATLAB variable refers to a matrix [a number is a 1 by 1 matrix]. In some situations, 1-by-1 matrices are interpreted as scalars, and matrices with only one row or one column are interpreted as vectors.

A matrix is a rectangular array of numbers. For example:

$$\begin{matrix} 3 & 6 & 9 & 2 \\ 1 & 4 & 8 & 5 \\ 2 & 8 & 7 & 5 \\ 1 & 4 & 2 & 3 \end{matrix}$$

defines a matrix with 4 rows, 4 columns, and 16 elements.

Defining Matrices in MatLab

MATLAB is designed to make definition of matrices and matrix manipulation as simple as possible.

Matrices can be introduced into MATLAB in several different ways. For example, either of the statements

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

and

```
>> A = [ 1 2 3
        4 5 6
        7 8 9 ]
```

creates the obvious 3-by-3 matrix and assigns it to a variable A.

Note that:

- The elements within a row of a matrix may be separated by commas as well as a blank.
- The elements of a matrix being entered are enclosed by brackets;
- A matrix is entered in "row-major order" [i.e. all of the first row, then all of the second row, etc];
- Rows are separated by a semicolon [or a newline], and the elements of the row may be separated by either a comma or a space. [Caution: Watch out for extra spaces!]

The matrix element located in the i^{th} row and j^{th} column of A is referred to in the usual way:

```
>>A(1,2), A(2,3)

ans =

     2

ans =

     6
```

It's very easy to modify matrices: >>

```
A(2,3) = 10;
```

2.1.2 Building Matrices from a Block

Large matrices can be assembled from smaller matrix blocks. For example, with matrix A in hand, we can enter the following commands:

```
>> C = [A; 10 11 12];        <== generates a (4x3) matrix
>> D = [A; A; A];            <== generates a (9x3) matrix
>> E = [A, A, A];            <== generates a (3x9) matrix
```

As with variables, use of a semicolon with matrices suppresses output. This feature can be especially useful when large matrices are being generated.

2.1.3 Built-in matrix functions

MATLAB has many types of matrices which are built into its system. For example,

```
Function Description
================================================
diag returns diagonal M.E. as vector
eye identity matrix
hilb Hilbert matrix
magic magic square
ones matrix of ones
rand randomly generated matrix
triu upper triangular part of a matrix
tril lower triangular part of a matrix
zeros matrix of zeros
================================================
```

Here are some examples:

i. **Matrices of Random Entries:** A 3 by 3 matrix with random entries is produced by typing

```
>> rand(3)

ans =

0.0470    0.9347    0.8310
0.6789    0.3835    0.0346
0.6793    0.5194    0.0535
```

General m-by-n matrices of random entries are generated with

```
>>rand(m,n);
```

ii. **Magic Squares:** A magic square is a square matrix which has equal sums along all its rows and columns. For example:

```
>> magic(4)

ans =

16    2     3    13
5    11    10     8
9     7     6    12
4    14    15     1
```

The elements of each row and column sum to 34.

iii. **Matrices of Ones:** The functions

- eye (m,n) produces an m-by-n matrix of ones.
- eye (n) produces an n-by-n matrix of ones.

iv. **Matrices of Zeros:** The commands

- zeros (m,n) produces an m-by-n matrix of zeros.
- zeros (n) produces an n-by-n one;

If A is a matrix, then zeros (A) produces a matrix of zeros of the same size as A.

v. **Diagonal Matrices:** If x is a vector, diag(x) is the diagonal matrix with x down the diagonal.

If A is a square matrix, then diag(A) is a vector consisting of the diagonal of A. What is diag (diag (A))? Try it.

2.2 MATRIX OPERATIONS

The following matrix operations are available in MATLAB:

```
Operator Description Operator Description
================================================
+ addition         ' transpose
- subtraction      \ left division
* multiplication   / right division
^ power
================================================
```

These matrix operations apply, of course, to scalars (1-by-1 matrices) as well. If the sizes of the matrices are incompatible for the matrix operation, an error message will result, except in the case of scalar-matrix operations (for addition, subtraction, and division as well as for multiplication) in which case each entry of the matrix is operated on by the scalar.

2.2.1 Matrix Transpose

The transpose of a matrix is the result of interchanging rows and columns. MATLAB denotes the [conjugate] transpose by following the matrix with the single-quote [apostrophe]. For example:

```
        >> A'
ans =
          1        4        7
          2        5        8
          3        6        9
         >> B = [1+i       2 + 2*i            3 - 3*i]'
B =
         1.0000 - 1.0000i
         2.0000 - 2.0000i
         3.0000 + 3.0000i
```

2.2.2 Matrix Addition/Subtraction

Let matrix "A" have m rows and n columns, and matrix "B" have p rows and q columns. The matrix sum "A + B" is defined only when m equals p and n equals q, the result is n-by-m matrix

having the element-by-element sum of components in A and B.

For example:

```
>> E = [ 2 3; 4 5.0; 6 7];
>> F = [ 1 -2; 3 6.5; 10 -45];
>> E+F
ans =
    3.0000    1.0000
    7.0000   11.5000
   16.0000  -38.0000
```

2.2.3 Matrix Multiplication

Matrix multiplication requires that the sizes match. If they don't, an error message is generated.

```
>> A*B, B*A;
>> B'*A;
>> A*A', A'*A;
>> B'*B, B*B';
```

Scalars multiply matrices as expected, and matrices may be added in the usual way (both are done "element by element”):

```
>> 2*A, A/4;
>> A + [b,b,b];
```

Example:

We can use matrix multiplication to check the "magic" property of magic squares.

```
>> A = magic(5);
>> b = ones(5,1);
>> A*b;                <== (5x1) matrix containing row sums.
>> v = ones(1,5);
>> v*A;                <== (1x5) matrix containing column sum.
```

2.2.4 Matrix Functions "any" and "all”

There is a function to determine if a matrix has at least one nonzero entry, any, as well as a function to determine if all the entries are nonzero, all.

```
>> A = zeros(1,4)
>> any(A)
>> D = ones(1,4)
>> any(D)
>> all(A)
```

2.2.5 Returning more than One Value

Some MATLAB functions can return more than one value.

In the case of max the interpreter returns the maximum value and also the column index where the maximum value occurs.

```
>> [m, i] = max(B)
>> min(A)
>> b = 2*ones(A)
>> A*B
>> A
```

2.2.6 Size of Matrix

Size of a matrix can be calculate by using function 'size '.

```
>> x = [1 2 3 ;1 2 3];
>> s = size(x)
s =
     2   3
```

2.2.7 Length of Array

Length of an array can be found using function 'length'.

```
>> n = [-3:1:3];
>> l = length(n)
 l =
      7
```

2.2.8 Finding an element in a matrix

This function can be used to find index of any particular value. Say given array is

x= [0 2 4 6 8];

To find the indices of all values that are greater than 4, following is used

```
>> y = find(x>4)
y =
 4    5
```

----------TASK 01----------

Write a program to generate a new matrix B from the matrix A given below such that each column in the new matrix except the first one is the result of subtraction of that column from the previous one i.e. 2nd new column is the result of subtraction of 2nd column and 1st column and so on. Copy the first column as it is in the new matrix.

$$A = \begin{matrix} 13 & 6 & 9 \\ 1 & 4 & 8 \\ 2 & 8 & 17 \end{matrix}$$

----------TASK 02----------

Generate two 2500 sampled random discrete time signals (1 dimensional) using rand() function i.e. rand(1, 2500). Write a program to add the two such random signals together using simple vector addition.

2.3 SUB-MATRICES

2.3.1 A note on colon notation

A central part of the MATLAB language syntax is the "colon operator," which produces a list. For example:

```
>> x = -3:3
x =
```

-3 -2 -1 0 1 2 3

The default increment is by 1, but that can be changed. For

example: >> x = -3 : .5 : 3

x =

Columns 1 through 7

-3.0000 -2.5000 -2.0000 -1.5000 -1.0000 -0.5000 0

Columns 8 through 13

0.5000 1.0000 1.5000 2.0000 2.5000 3.0000

This can be read: "x is the name of the list, which begins at -3, and whose entries increase by .5, until 3 is surpassed." You may think of x as a list, a vector, or a matrix, whichever you like.

In our third example, the following statements generate a table of sine.

```
>> x = [0.0:0.1:2.0]';
>> y = sin(x);
>> [x y]
```

Try it. Note that since sin operates entry-wise, it produces a vector y from the vector x.

The colon notation can also be combined with the earlier method of constructing matrices. >> a = [1:6 ; 2:7 ; 4:9]

----------TASK 03----------

Using colon notation, generate the following sequence:

-65.25, -57.75, -50.25. . . . . ., 54.75,62.25, 69.75

2.3.2 Sub-matrix extraction using colon notation

Colon notation can be used to generate vectors. A very common use of the colon notation is to extract rows, or columns, as a sort of "wild-card" operator which produces a default list. For example,

- A(1:4,3) is the column vector consisting of the first four entries of the third column of A
- A(:,3) is the third column of A. A colon by itself denotes an entire row or column
- A(1:4,:) is the first four rows of A.

An arbitrary integral vector can be used as subscripts. The statement

A(:,[2 4])

contains as columns, columns 2 and 4 of matrix A.

This subscripting scheme can be used on both sides of an assignment

statement: A(:,[2 4 5]) = B(:,1:3)

It replaces columns 2, 4, and 5 of matrix A with the first three columns of matrix B. Note that the "entire" altered matrix A is printed and assigned. Try it.

----------TASK 04----------

Given the matrices:

A=[-12,34,61,-9;65,78,90,12; 14,78,45,12; 60,25,3,8

B=[34,67,8,9; 12,-91,12,9; 89,-8,0,2; 16,9,23,67]

Find the following:

1) Array addition; store the result in matrix C
2) Array subtraction; store the result in matrix D
3) Array multiplication using .* operator; store the result in matrix E
4) Array division using ./ operator; store the result in matrix F
5) Array exponentiation using .^ operator; store the result in matrix G
6) Take sin of A and store the result in H, Take sqrt of B and store the result in I. Find H*I and store the result in J.

----------TASK 05----------

Type the given matrix in MatLab:

$$A = \begin{bmatrix} 3 & 7 & -4 & 12 \\ -5 & 9 & 10 & 2 \\ 6 & 13 & 8 & 11 \\ 15 & 5 & 4 & 1 \end{bmatrix}$$

Find the following:

1) Create 4x3 array B consisting of all elements in the second through fourth columns of A
2) Create 3x4 array C consisting of all elements in the second through fourth rows of A
3) Create 2x3 array D consisting of all elements in the first two rows and the last three columns of A

----------TASK 06----------

MATLAB has functions to round floating point numbers to integers. These are round, fix, ceil, and floor. Test how these functions work. Determine the output of the following:

```
>> f = [-.5 .1 .5];
>> round(f)
>> fix(f)
>> ceil(f)
>> floor(f)
```

----------TASK 07----------

Given the following matrix:

$$A = \begin{matrix} -3 & 5 \\ 4 & 8 \end{matrix}$$

Find the following:

1) Column-wise sum of all elements of A using **sum** function; for information about sum function, type help sum in matlab
2) Column-wise product of all elements of A using **prod** function; for information about prod function, type help prod in matlab
3) Length of matrix A
4) Size of matrix A

----------TASK 08----------

The end command is used to access the last row or column of a matrix. Use the end command to delete and update the last row and column of the following matrix.

Matrix A = [3 23 34 12 34 5 56 23; 12 34 34 32 23 23 45 1; 67 23 2 4 4 5 6 456; 4 5 1 1 2 34 45

56; 67 67 45 67 78 7 8 5; 6 35 5 3 5 56 7 8]

Hint:

For deleting a column use A(3 , :)=[ ];

For deleting last column use A(: , end)=[ ];

and vice versa.

----------TASK 09----------

Try the following commands in MatLab and comment on them:

(i) A(3,end)

(ii) A(:)

(iii) A(: , end)

(iv) Y = linspace(20,100)

(v) Y = linspace(20,100,50)

----------TASK 10----------

Use the inverse (inv(A)) function to find the inverse of A for finding the unknowns for the Linear equation.

x + 2y + 3z = 1

4x + 5y + 6z = 2

7x + 8y = 1

where:

X= [x y z]

A= [1 2 3 ;4 5 6; 7 8 9]

b= [1;2;3]

Ax=b

Hint:

x = inv(A)*b or x = A\b

----------TASK 11----------

Solve Task 10 by taking the equations from user.

Hint: Take the matrix A and b from user.
