Chapter-2

Introduction:

We shall present here a brief introduction to matrices and the MATLAB commands relevant to working with them.
Matrix algebra approach is a useful means of compactly presenting lengthy and sometimes complex formulas. A matrix is a rectangular array of elements arranged in horizontal rows and vertical columns. The number of rows and columns may vary from one matrix to another, so we conveniently describe the size of a matrix by giving its dimensions, that's, the number of its rows and columns. A matrix is said to have size n x m, read, "n by m" if it has n rows (horizontal lines) and m columns (vertical lines). The number of rows is always stated first. A square matrix is one for which m=n. The most direct way to create a matrix in MATLAB is to type the entries in the matrix between square brackets one row at a time. To separate the entries in the same row, type a comma or press the space bar. To indicate the beginning of a new row, type a semicolon (or a newline).

As an example, use MATLAB to create a 3-by-3 matrix A given by

$$A = \begin{bmatrix} 1 & 2 & 9 \\ 4 & 7 & 5 \\ 3 & 1 & 6 \end{bmatrix}$$

$$>> A = \left[\underbrace{1\ 2\ 9}_{first\ row};\ \underbrace{4\ 7\ 5}_{second\ row};\ \underbrace{3\ 1\ 6}_{third\ row} \right]$$ %define a 3-by-3 square matrix

This results in a 3x3 matrix, which looks like

A =

```
   1   2   9
   4   7   5
   3   1   6
```

MATLAB displays matrices without braces.

Building large matrices:

Large matrices can be assembled from smaller matrix blocks.

Practice:

```
>>A=[1 2 9; 4 7 5; 3 1 6];
>>B=[A; 11  12  13]              %add one row to matrix A
```

```
B =

    1    2    9
    4    7    5
    3    1    6
   11   12   13

>>C=[A; A; A]

C =

    1    2    9
    4    7    5
    3    1    6
    1    2    9
    4    7    5
    3    1    6
    1    2    9
    4    7    5
    3    1    6
```

We can determine the size of a vector or matrix by using the **size** command.

```
>>size(A)                    %return the size of A
>>size(A,1)                  %return the number of rows in A
>>size(A,2)                  %return the number of columns in A
```

Individual elements of a matrix can be referenced via indices enclosed within parentheses. The first index refers to the row number, and the second index refers to the column number. For eample, entering

```
>>A(2,1)                     %reference the second element of the first row
```

results in

```
ans=
     4
```

It is possible, under MATLAB, to alter the elements by reassigning their values.

```
>>A(2,1)=9                   %change the second element in the first column to 9
```

The colon (:) operator may be used to specify an entire row or column. The expression A(1,:) specifies the first row of A, A(:,3) designates the third column of A, and A(1:3,:) specifies the first three rows of A.

Practice:

$$A=\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\\ 7 & 9 & 2\end{bmatrix},\ B=\begin{bmatrix}1\\ 5\\ 9\end{bmatrix},\ C=\begin{bmatrix}2 & 1 & 0\end{bmatrix}$$

```
>>A=[1  2  3;  4  5  6;  7  9  2];        %specify matrix A
>>B=[1; 5; 9];                           %specify column vector B
>>C=[2  1  0];                           %specify the row vector C
>>A(:,1) = B;                            %replace the first column of A by B
>>A(2,:) = C;                            %replace the second row of A by C
```

Furthermore, the following commands are quite handy.

```
>>A(:,j);                                %correspond to the jth-column of A
>>A(i,:);                                %correspond to the ith-row of A
>>A(:,k:m);                              %correspond to [A(:,k)  A(:,k+1), …A(:,m)]
>>A(:,2)=[ ];                            %delete the second column of A
```

Note:

If a colon (:) is used by itself as in x(:,2), it refers to the entire range of the index.

Vectors can be viewed as special cases of matrices. Therefore, vectors are generated the same way as matrices.

Practice:

```
>>V=[3  5  7]  <enter>                   %define a row vector A
V=
        3   5   7
```

Special matrices:

A **diagonal matrix** is a matrix where only the diagonal entries are non-zero. An **identity matrix, I,** is the diagonal matrix with diagonal consisting of all 1's. An **upper triangular matrix** is a matrix whose entries lying below the diagonal are all zero. A **lower triangular matrix** is a matrix whose entries lying above the diagonal are all zero.
MATLAB has several built-in matrices. The command **eye(n)** produces a n-by-n identity matrix. The **zeros(n,m)** and **ones(n,m)** commands generate n-by-m matrices filled with zeros, and filled with ones, respectively. The **rand(n)** command will generates an n-by-n matrix whose elements are pseudo-random numbers uniformly distributed between 0 and 1, while **rand(n,m)** will create a n-by-m matrix with randomly generated entries distributed uniformly between 0 and 1. The **magic(n)** command generates a n-by-n square matrix whose entries constitute a magic square; i.e., the sum of elements along each row, column, or principal diagonal is the same value.

Practice:

```
>>D=eye(3) <enter>                       %create a 3-by-3 identity matrix

D=
```

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

>>H=zeros(2,3) <enter> %create a 2-by-3 null matrix

H=

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

>>W=ones(2,3) <enter> %create a 2-by-3 matrix with entries equal to 1

W=

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Practice:

You can allocate memory for 1-D arrays (vectors) using the **zeros** command. The following command allocates a 200-dimensional array:

>>y=zeros(200,1)

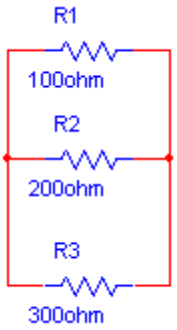
Similarly, you can allocate memory for 2-D matrices. The command

>>y=zeros(5,6)

defines a 5-by-6 matrix

Practice:

Write a MATLAB code to compute the equivalent resistance of three resistors connected in parallel.



>>% R1=100, R2=200, and R3=300
>>R=[100 200 300]; %specify vector of resistances

```
>>R_inv=ones(size®)./R;          %evaluate 1/R
>>R_eq=1/sum(R_inv);             %evaluate the equivalent resistance
>>R_eq  <enter>                  %print the equivalent resistance

R_eq =
                54.5459
```

The empty matrix:

The empty matrix is represented in MATLAB as []. This is a matrix with dimension zero-by-zero. Therefore, the statement

```
>>B=[]
```

enters a zero-by-zero matrix B into the workspace. The empty matrix has some special uses. It can serve, for example, to reduce the size of an existing matrix. For example

```
>>A([1 3],:)=[]
```

deletes rows one and three from A

Matrix operations:

Addition of matrices:

Matrix addition can be accomplished only if the matrices to be added have the same dimensions for rows and columns. If A and B are two matrices of the same size, then the sum A+B is the matrix obtained by adding the corresponding entries in A and B.

Practice:

```
>>A = [3  4;  5  6];             %define matrix A
>>B = [1  2;  9  7];             %define matrix B
>>C=A+B  <enter>                 %compute the sum

C =
```

$$\begin{bmatrix} 4 & 6 \\ 14 & 13 \end{bmatrix}$$

If B is any matrix, then the negative of B, denoted by –B, is the matrix whose entries are the negatives of the corresponding entries in B

***Note**: A scalar may be added to a matrix of any dimension. If A is a matrix, the expression A+4 is evaluated by adding 4 to each element of A.*

Difference of matrices:

If A and B are matrices with the same size, then the difference between A and B denoted A-B is the matrix defined by A-B=A+(-B).

Practice:

```
>>A=[3  4;  5  6];          %define matrix A
>>B=[1  2;  9  7];          %define matrix B
>>D=A-B  <enter>            %compute the difference
```

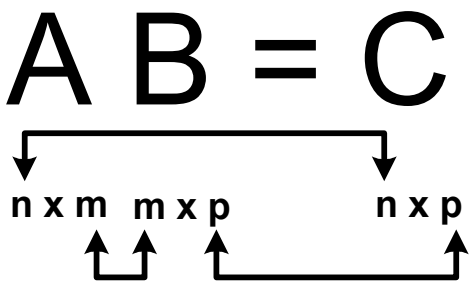
D =

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

Product of matrices:

We can form the product C=A x B only if the number of rows of B, the right matrix, is equal to the number of columns of A, the left matrix. Such matrices are said to be conformable. Given an m-by-n matrix A and a k-by-p matrix B, then A and B are conformable if and only if n = k. An element in the ith row and jth column of the product, AB, is obtained by multiplying the ith row of A by the jth column of B.

A straightforward way of checking for conformity is to write the dimensions underneath the matrices as illustrated below:



Observe that the inner two numbers, giving the number of elements in a row of A and column of B, respectively, must be equal. The outer two numbers, indicating the number of rows of A and columns of B, give the dimensions of the product matrix C. Remember that the order of multiplication is important when dealing with matrices.

Practice:

For the given matrices, obtain the product C=A * B

$$A=\begin{bmatrix}1 & 2\\ 3 & 4\\ 5 & 6\end{bmatrix},\quad B=\begin{bmatrix}-2 & 3\\ 4 & 1\end{bmatrix}$$

```
>>A=[1  2;  3  4;  5  6];          %define matrix A
>>B=[-2  3;  4  1];                %define matrix B

>>C = A*B  <enter>                 %compute the product matrix C

C =
```

$$\begin{bmatrix}6 & 5\\ 10 & 13\\ 14 & 21\end{bmatrix}$$

Note:

Matrix multiplication is not commutative in general, i.e., $A*B \neq B*A$

Multiplication by a scalar:

If A is a matrix and k is a scalar, then the product k*A is defined to be the matrix obtained by multiplying each entry of A by the constant k.

Practice:

$$A=\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix},\quad B=\begin{bmatrix}7 & 0\\ 5 & 9\end{bmatrix}$$

```
>>A=[1  2;  3  4];          %define matrix A
>>B=[7  0;  5  9];          %define matrix B
>>C=2*A                     %scale A by 2
>>D=3*A+2*B                 % scale A by 3, B by 2, and add the results
```

answers:

$$C=\begin{bmatrix}2 & 4\\ 6 & 8\end{bmatrix},\quad D=\begin{bmatrix}17 & 6\\ 19 & 30\end{bmatrix}$$

Inner product:

Consider two vectors $X=[x_1,x_2]$, and $Y=[y_1,y_2]$. The inner product (or dot product) is defined as $X.Y=x_1\times y_1+x_2\times y_2=|X||Y|\cos(\theta)$.

Practice:

Find the inner product and the angle θ for the following two vectors:

A=[3 4]
B=[6 7]

```
>>A=[3  4];                    %specify vector A
>>B=[6  7];                    %specify vector B
>>C=A*B'                       %evaluate the inner product
>>%compute the value of the angle theta
>>theta=acos(C/(sqrt(A*A')*sqrt(B*B')))
```

Some properties of matrices:

1. A+B=B+A
2. A+(B+C)=(A+B)+C
3. A(B+C)=AB+AC
4. AI=IA=A % I is the identity matrix
5. 0A=0 %0 is the null matrix
6. A0=0
7. A+0=A

Determinant of a matrix:

Associated with any square matrix A is a scalar quantity called the **determinant** of the matrix A. A matrix whose determinant is non-zero is called a **non-singular** or **invertible**, otherwise it is called **singular**. For a 2-by-2 matrix A,

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

The determinant is equal to $a_{11}a_{22} - a_{21}a_{12}$. Similarly, for a 3-by-3 matrix B,

$$B = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

The determinant is equal to:

$$a_{11}\begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} - a_{12}\begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} + a_{13}\begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix} =$$

$$a_{11}\left(a_{22}a_{33} - a_{32}a_{23}\right) - a_{12}\left(a_{21}a_{33} - a_{31}a_{23}\right) + a_{13}\left(a_{21}a_{32} - a_{31}a_{22}\right)$$

The command **det** evaluates the determinant of a square matrix.

Practice:

Find the determinant of the square matrix A depicted below.

$$A=\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\\ 7 & 1 & 2\end{bmatrix}$$

```
>>A=[1  2  3;  4  5  6;  7  1  2];        %specify matrix A
>>det(A)                                  %compute the determinant
```
answer: -21

Note:

For the diagonal matrix B,

$$B=\begin{bmatrix}b_{11} & 0 & 0\\ 0 & b_{22} & 0\\ 0 & 0 & b_{33}\end{bmatrix}$$

the determinant is equal to the product of the diagonal elements, i.e., det(B)=$b_{11}b_{22}b_{33}$.

Diagonal matrix:

The diagonal matrix A is one whose elements off the main diagonal are all equal to zero, while those along the main diagonal are non-zero.
The command **diag** will generate a diagonal matrix with the specified elements on the main diagonal.

Practice:

```
>>A=diag([1  2  3])              %generate a diagonal matrix
```

Identity matrix:

If A is any matrix, the identity matrix for multiplication is a matrix **I** which satisfies the following relation.

$AI = A$ and $IA = A$

This matrix, called the identity matrix, is the square matrix

$$I = \begin{bmatrix} 1 & 0 & 0 & 0 & \dots \\ 0 & 1 & 0 & 0 & \dots \\ 0 & 0 & 1 & 0 & \dots \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

That is, all elements in the main diagonal of the matrix, running from top left to bottom right, are equal to 1, all other elements equal zero. Note that the identity matrix is always indicated by the symbol *I*.

Rank of a matrix:

The rank of a matrix, A, equals the number of linearly independent rows or columns. The rank can be determined by finding the highest-order square sub-matrix that is non-singular. The command **rank** provides the rank of a given matrix.

Practice:

```
>>A=[1  2  3;  4  5  6;  7  8  9];  %define matrix A
>>rank(A)                           %determine the rank of matrix A
```

Inverse of matrix:

If A and B are square matrices such that AB=BA=I, then matrix B is called the **inverse** of A and we usually write it as B= A^{-1}. Only a square matrix whose determinant is not zero has an inverse. Such a matrix is called nonsingular. If any row (or column) of a square matrix is some multiple or linear combination of any other row(s) [or column(s)] the matrix will be singular, i.e., have a determinant of zero and have no inverse. The command **inv** provides the inverse of a matrix.

Practice:

Find the inverse of the matrix A given by

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 2 \end{bmatrix}$$

```
>>A=[1  2  3;  4  5  6;  7  8  2];          %define matrix A
>>B=inv(A)  <enter>                        %compute the inverse of A store result in B
```

B =

$$\begin{bmatrix} -1.8095 & 0.9524 & -0.1429 \\ 1.6190 & -0.9048 & 0.2857 \\ -0.1429 & 0.2857 & -0.1429 \end{bmatrix}$$

Note:

For the diagonal matrix B given by

$$B = \begin{bmatrix} b_{11} & 0 & 0 \\ 0 & b_{22} & 0 \\ 0 & 0 & b_{33} \end{bmatrix}$$

The inverse is also a diagonal matrix C,

$$C = \begin{bmatrix} \frac{1}{b_{11}} & 0 & 0 \\ 0 & \frac{1}{b_{22}} & 0 \\ 0 & 0 & \frac{1}{b_{33}} \end{bmatrix}$$

Quick method for finding the inverse of a 2-by-2 matrix:

1. Interchange the elements of the main diagonal
2. Change the signs of element on the secondary diagonal
3. Divide by the determinant

Transpose of a matrix:

The transpose of a matrix is the result of interchanging rows and columns. The transpose is found by using the prime operator (apostrophe), [‘]. In particular, the transpose of a row vector is equal to a column vector and vice versa. For a matrix with complex entries the prime operator yields the *complex conjugate* transpose. To obtain the transpose without the complex conjugate use [.’] instead.

Practice:

Find the transpose of the matrix A

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 2 \end{bmatrix}$$

```
>>A=[1  2  3;  4  5  6;  7  8  2];          %specify matrix A
```

B=A’ <enter> %compute the transpose of A and store result in B

B=

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

Symmetric matrix:

A symmetric matrix can be defined as one which is equal to its transpose. Clearly only square matrices can be symmetric. It is easily shown that the sum or difference of two symmetric matrices is also symmetric.

Practice:

$$A = \begin{bmatrix} 3 & 7 & 8 \\ 7 & 4 & 2 \\ 8 & 2 & 5 \end{bmatrix}$$

```
>>A=[3 7 8; 7 4 2; 8 2 5];          %symmetric matrix
>>B=A'                              %transpose of A
```

Trace of a matrix:

The trace of a square matrix is equal to the sum of its diagonal elements. The **trace** command provides the trace of a given matrix.

Practice:

```
>>%compute the trace of a matrix
>>A=[1  2  3;  4  5  6;  7  8  2];    %define matrix A
>>trace(A)  <enter                    %calculate the trace of matrix A
```

ans:

8

Additional properties of matrices:

1. inv(A*B)=inv(A)*inv(B)
2. $\left(A * B\right)^{'} = B^{'} * A^{'}$
3. $\left(A + B\right)^{'} = A^{'} + B^{'}$

Eigenvalues and eigenvectors:

The eigenvalues and eigenvectors of a square matrix can be determined using the function **eig**. There are two flavors of this function, one just finds the eigenvalues, and the other finds both the eigenvalues and the eigenvectors.

Practice:

```
>>A=[1  2;  3  4];          %define matrix A
>>eig(A)  <enter)           %compute the engenvalues of A
```

The same command **eig** can be used to produce both the eigenvalues and eigenvectors. Here is an illustration:

```
>>A=[1  2;  3  4];          %define matrix A
>>[V,D]=eig(A)              %eigenvalues and eigenvectors
```

The eigenvalues of matrix A are stored as the diagonal entries of the diagonal matrix D and the corresponding eigenvectors are stored in columns of the matrix V.

Solving systems of linear equations.

To solve the linear system $Ax = b$, where A is known n-by-n matrix, b is a known column vector of length n, and x is an unknown column vector of length n.

$Ax = b$

multiply each side by the inverse matrix A^{-1}

$\therefore A^{-1}Ax = A^{-1}b$

but $A^{-1}A = I \Rightarrow Ix = A^{-1}b$

$\therefore x = A^{-1}b$

This procedure turns out to be slow. A quicker, and sometimes more accurate method to solve systems of linear equation is based on the Gaussian elimination scheme, where one systematically eliminates one unknown from the equations down to the point where there is only one unknown left. The solution by Gaussian elimination is implemented via the left division operation (backslash),"\":

$x = A \backslash b$

Practice:

Solve the following system of linear equation.

$$\begin{cases} x + 5y + 7z = 1 \\ 3x + 2y + 4z = 2 \\ 7x + 9y + z = 3 \end{cases}$$

This system of linear equation can be written in the form:

$$\begin{bmatrix} 1 & 5 & 7 \\ 3 & 2 & 4 \\ 7 & 9 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & 5 & 7 \\ 3 & 2 & 4 \\ 7 & 9 & 1 \end{bmatrix}, \qquad w = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \qquad b = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

First method:

```
>>%Slower procedure
>>A=[1 5 7; 3 2 4; 7 9 1];          %specify matrix A
>>b=[1 2 3]';                       %specify the column vector b
>>w=inv(A)*b;                       %compute the solution of the linear system
```

Second method:

```
>>%Faster procedure
>>A=[1 5 7; 3 2 4; 7 9 1];          %specify matrix A
>>b=[1 2 3]';                       %specify the column vector b
>>w=A\b                             %compute the solution of the linear system
```

Cramer's rule:

```
>>A=[1 5 7; 3 2 4; 7 9 1];          %specify matrix A
>>b=[1 2 3]';                       %specify vector b
>>for n=1:3
        D=A;
        D(: , n)=b;
        C=D;
        w(n)=det(C)/det(A);
end

>>w=w'

w =
        0.5495
        -0.1099
        0.1429
```