Unit 2: Linear algebra with MATLAB

Introduction

Linear algebra is a branch of mathematics concerned with the study of matrices, vectors, vector spaces (also called linear spaces), linear maps (also called linear transformations), and systems of linear equations. MATLAB are well suited for studying Linear Algebra because its underlying structure is based on MATRICES.

There are many situations in which it would be too difficult and/or too tedious to compute or evaluate matrix operations more especially during processes of numerical differentiation or integration where large and denser matrices appear. In this unit, we will learn various ways of defining of matrices, vectors and other linear systems in MATLAB. Specifically, in this unit, we shall explore ways of basic matrix operations and other related problems in linear algebra.

Unit Objectives

On successful completion of the Unit, students should be able to:

- Describe vectors and matrices in MATLAB
- Define vector and matrix operations in MATLAB
- Load and read data files
- Solve system of linear equations using MATLB

Key Terms

As you go through this unit, ensure that you understand the key terms or phrases used in this unit as listed below:

- Vector
- Matrix
- Matrix operations
- Eigenvalue
- Coefficient matrix
- Inverse matrix

An order or size of any matrix A can be obtained via MATLAB command "***size(A)***". Thus, other than defining matrices, MATLAB has specific functions that support reading an entire file and creating a matrix of the data with one statement. We use a function ***load*** to create a matrix of data, and functions ***size*** and ***length*** to read the data files. Thus

- `>> load mydata.dat   % loads file into a matrix called mydata`
- `>> size(mydata)      % return number of rows and columns for the matrix mydata`
- `>> length(mydata)    % return total number of elements in matrix mydata`

Matrix functions

We discuss different functions in matrices and describe how each of these is defined in MATLAB.

1) Transpose of a matrix

Just as how we defined transpose of a vector, similarly we describe the transpose of a matrix A as a matrix of m- rows and n- columns defined as an $m \times n$ matrix

$$A^T = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} \in \mathbb{R}^{m \times n}$$

which is defined in MATLAB as

```
>> A'
ans =
    2    1    7
    3    2   11
   -5    4    3
    0    8    6
```

if A is originally defined as

$$A = \begin{bmatrix} 2 & 3 & -5 & 0 \\ 1 & 2 & 4 & 8 \\ 7 & 11 & 3 & 6 \end{bmatrix}$$

2) Diagonal matrix

Diagonal of a square matrix A is defined as a vector

$$\boldsymbol{diag(A)} = \begin{bmatrix} a_{11} \\ a_{22} \\ \vdots \\ a_{nn} \end{bmatrix} \in \mathbb{R}^{n=\min(p,m)}$$

Thus, it a column vector whose elements are diagonal entries of a matrix A. Thus, for a given matrix

$$A = \begin{bmatrix} 2 & 3 & -5 & 0 \\ 1 & 2 & 4 & 8 \\ 7 & 11 & 3 & 6 \\ 3 & -1 & 4 & 5 \end{bmatrix}$$

it's diagonal matrix is

$$diag(A) = \begin{bmatrix} 2 \\ 2 \\ 3 \\ 5 \end{bmatrix}$$

In MATLAB, we describe such operation as

```
>> A=[2 3 -5 0;1 2 4 8;7 11 3 6;3 -1 4 5]
A =
    2    3   -5    0
    1    2    4    8
    7   11    3    6
    3   -1    4    5
>> diag(A)
ans =
    2
    2
    3
    5
```

Let's look at a second example, a diagonal can still be obtained for non-square matrix such as

$$A = \begin{bmatrix} 2 & 3 & -5 & 0 \\ 1 & 2 & 4 & 8 \\ 7 & 11 & 3 & 6 \end{bmatrix}$$

whose diagonal is taken as

```
>> A'
ans =
    2    1    7
    3    2   11
   -5    4    3
    0    8    6
>> diag(ans)
ans =
    2
    2
    3
```

Before we describe another form of matrix, triangular matrix, let us have a description of the special type of diagonal matrix called “Identity” matrix ***I***, defined by

$$I = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

While in MATLAB, this type of matrix is defined using a special term called “eye”. Thus, “***eye(n)***” produces an $n \times n$ diagonal matrix. For example

```
>> eye(3)
ans =
    1    0    0
    0    1    0
    0    0    1
```

If such a matrix is made of “ones” entry throughout, then we have another special matrix called “ones matrix” which is generated in MATLAB using a function “***ones(n)***”. For example with “*ones(3)*”, we obtain

```
>> ones(3)
ans =
    1    1    1
    1    1    1
    1    1    1
```

Alternatively, if the matrix is made of “zero” entry throughout, then we have another special matrix called “zero matrix” which is generated in MATLAB using a function “***zeros(n)***”. For example with “*zeros(3)*”, we obtain

```
>> zeros(3)
ans =
    0    0    0
    0    0    0
    0    0    0
```

3) Triangular matrices

These are another type of special matrices defined as either the upper or lower part of the matrix including the diagonal.

a) Lower Triangular matrix L of a matrix A: is described as

$$L = \begin{bmatrix} a_{11} & 0 & 0 \\ \vdots & \ddots & 0 \\ a_{n1} & \cdots & a_{nn} \end{bmatrix}$$

b) Upper Triangular matrix U: is described as

$$U = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ 0 & \ddots & \vdots \\ 0 & 0 & a_{nn} \end{bmatrix}$$

In MATLAB, lower triangular matrix is defined as

```
>> tril(A)
ans =
    2    0    0    0
    1    2    0    0
    7   11    3    0
    3   -1    4    5
```

and upper triangular matrix is defined as

```
>> triu(A)
ans =
    2    3   -5    0
    0    2    4    8
    0    0    3    6
    0    0    0    5
```

4) Matrix addition and subtraction

Other kinds of matrix operations that are critical in numerical operations are matrix addition and subtraction. Both of these operations require two matrices A and B to be of the same order $n \times m$ and $p \times q$, respectively where $n = p$ and $m = q$. Thus,

a) Matrix addition

Given two matrices $A \in \mathbb{R}^{n \times m}$ and $B \in \mathbb{R}^{n \times m}$, then $C = A + B \in \mathbb{R}^{n \times m}$. Hence addition of two matrices is done component-wise, such that in MATLAB we have

```
>> tril(A)+triu(A)
ans =
    4    3   -5    0
    1    4    4    8
    7   11    6    6
    3   -1    4   10
```

for the addition of two lower and upper triangular matrices described above.

b) Matrix subtraction

Just as with addition, subtraction of two matrices is done component-wise for the defined resultant matrix $C = A + B \in \mathbb{R}^{n\times m}$, whenever $A \in \mathbb{R}^{n\times m}$ and $B \in \mathbb{R}^{n\times m}$. Thus, using the same two lower and upper triangular matrices, we have

```
>> tril(A)-triu(A)
ans =
    0   -3    5    0
    1    0   -4   -8
    7   11    0   -6
    3   -1    4    0
```

5) Matrix multiplication

In matrix multiplication, what is more important is the corresponding number of rows in one given matrix, corresponding to another matrix. Thus, given two matrices $A \in \mathbb{R}^{n\times m}$ and $B \in \mathbb{R}^{m\times p}$, then $C = AB \in \mathbb{R}^{n\times p}$. Hence multiplication of two matrices is done such that entries of the resultant matrix C are each defined as

$$c_{jk} = \sum_{i=1}^{n} a_{ji} b_{ik}$$

Hence, in MATLAB, given any two matrices $A_{2\times 3}$ and $B_{3\times 4}$, we get $C_{2\times 4}$, that is

```
>> A=[2 0 1;3 5 1];
>> B=[0 -3 5 0;1 0 -4 -8;3 -1 4 0];
>> C=A*B;
>> C
C =
    3   -7   14    0
    8  -10   -1  -40
```

One thing we have to remember about matrix multiplication is that $AB \neq BA$, that is, matrix multiplication is never commutative. But, the following associative operations are possible

- $A(BC) = (AB)C$
- $A(B + C) = AB + AC$
- $C(A + B) = CA + CB$

6) Determinant

For square matrices, determinant is a crucial parameter to describe either an invertible or non-invertible matrix. Given a matrix $A \in \mathbb{R}^{n \times n}$, then the determinant is given as $\det(A) = |A|$ such that for a 2×2 matrix

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \in R^{2\times 2}$$

then

$$\det(A) = |A| = a_{11}a_{22} - a_{12}a_{21}$$

For example, in MATLAB,

```
>> B=[2 -3;4 5];
>> det(B)
ans =
   22
```

Similarly for any matrix of order $n \times n$, the same MATLAB operator can be used to obtain the determinant. For example

```
>> C=[3 4 -5;2 6 1;7 3 4];
>> det(C)
ans =
  239
```

We can as well notice that

$$\det(AB) = \det(A)\det(B)$$

and

$$\det(A^T) = \det(A$$

For example

```
>> A=[-1 2;3 4];
>> B=[2 2;1 -5];
>> det(A*B)
ans =
   120
>> det(A)*det(B)
ans =
   120
>> det(A')
ans =
   -10
>> det(A)
ans =
   -10
```

7) Inverse

As stated before, determinant is a crucial parameter to describe either an invertible or non-invertible matrix. We see in the next section how such a parameter is used when computing inverse of matrices.

The inverse of a square matrix A is defined by A^{-1} if

$$A^{-1}A = AA^{-1} = I$$

Thus, for a 2×2 matrix

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \in R^{2\times 2}$$

then

$$A^{-1} = {}^{1}/_{\det(A)} \begin{bmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{bmatrix} \in R^{2\times 2}$$

But the formula is different for any square matrix of order n, $n > 2$. Nonetheless, the operator for computing inverse of a square matrix in MATLAB is the same, i.e. "***inv(A)***".

For example

```
>> B
B =
    2    2
    1   -5
>> inv(B)
ans =
   0.4167    0.1667
   0.0833   -0.1667
```

Another example

```
>> A=[2 3 -5 0;1 2 4 8;7 11 3 6;3 -1 4 5]

A =

    2    3   -5    0

    1    2    4    8

    7   11    3    6

    3   -1    4    5

>> inv(A)

ans =

   0.0777   -0.1894    0.0368    0.2589

  -0.0511    0.0456    0.0811   -0.1703

  -0.1996   -0.0484    0.0634    0.0014

   0.1029    0.1614   -0.0565    0.0095
```

8) Eigenvalues

Eigenvalues are as well another crucial parameter in describing roots of a characteristic polynomial and also in classifying dynamical systems. Thus, Given a matrix $A \in \mathbb{R}^{n\times n}$, then the eigenvalues are given as $\mathrm{eig(A)} = |A|$ such that for a 2×2 matrix

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \in R^{2\times 2}$$

then

$$\mathrm{eig}(A) = |\lambda I - A| = \det(\lambda I - A)$$

where $\lambda' s \in \mathbb{C}$ are the eigenvalues. For example, in MATLAB

```
>> A=[2 3 -5 0;1 2 4 8;7 11 3 6;3 -1 4 5];
>> eig(A)

ans =

  11.1973 + 0.0000i
   4.3970 + 0.0000i
  -1.7972 + 5.1562i
  -1.7972 - 5.1562i
```

From the solution, we observe that our eigenvalues are in the form of $a + ib$ where a is the real part while b is the imaginary part.

9) Rank

With a MATLAB matrix function "rank", we are able to compute the number of linearly independent rows or columns of a given matrix. For example

```
>> A=[1 2 3;4 5 6;7 8 0];
>> rank(A)
ans =
     3
```

Lastly, MATLAB provides another set of special matrices (see Table 5). These matrices have interesting properties that make them useful for constructing examples and for testing algorithms. For more information, see MATLAB documentation by typing "***help_name of a matrix***" in the command window.

Table 5: Special matrices

hilb	Hilbert matrix
invhilb	Inverse Hilbert matrix
magic	Magic square matrix
pascal	Pascal matrix
toeplitz	Toeplitz matrix
vander	Vandermonde matrix
wilkinson	Wilkinson's eigenvalue test matrix
hadamard	Hadamard matrix

Activity 2 b

a) Define any 4×4 matrix A and extract a submatrix B consisting of rows 2 and 3 and columns 1 and 2 of the matrix A.

b) Verify using MATLAB that a matrix

$$R = \begin{bmatrix} 0 & 2 \\ 0 & 3 \end{bmatrix}$$

does not have an inverse.

c) Use MATLAB to show that for any given square invertible matrix of order n, it is not always true that

$$A^{-1}A = AA^{-1} = I$$

d) Given any two matrices S and T of order $n \times m$ and $m \times p$, respectively. Show using MATLAB that the resultant matrix ST is non-singular.

e) From a matrix $A = [1\ 2\ 3; 4\ 5\ 6; 7\ 8\ 9]$, define a ***concatenated*** matrix $B = [A\ 10 * A; -A\ eye(3)]$.

f) In Discrete Mathematics module, you learn that $H_{n\times n}$ is said to be a Hadamard matrix if and only if $H = (a_{ij})_{n\times n}$ such that $HH^T = nI_n$ where $a_{ij} = -1$ or 1. Given that

$$H = \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}$$

   a. Demonstrate that H is Hadamard matrix.
   b. Hence, draw up a concatenated Hadamard matrix

$$H^* = \begin{pmatrix} H & H \\ H & -H \end{pmatrix}$$

Linear system of equations

During applications that involve usage of differential equations, such as in modelling environmental changes, diseases, interactions, etc. often do land into a system of equations of the form $A\boldsymbol{X} = B$ where X is a vector of decision variables for the given system of differential equations, while A and B are the coefficient matrix and system solution vector, respectively. For example, a system of the form

$$\begin{aligned} 2x_1 + \quad x_2 - \quad x_3 &= 0 \\ x_1 + \quad 7x_2 + \quad 2x_3 &= 1 \\ 3x_1 - \quad 3x_2 + \quad 4x_3 &= 2 \end{aligned}$$

can be re-arranged in form of $A\boldsymbol{X} = B$ where

$$A = \begin{bmatrix} 2 & 1 & -1 \\ 1 & 7 & 2 \\ 3 & -3 & 4 \end{bmatrix}, X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix}$$

In this unit, we shall then describe means of solving such given systems of linear equations in MATLAB based on the relation

$$\boldsymbol{X} = A^{-1}B$$

given that A^{-1} exists.

Since we have already looked at how to compute matrix multiplication and also inverse of a matrix, then easier it will be to compute

$$\boldsymbol{X} = A^{-1}B$$

For example, for the equations

$$\begin{aligned} x + \quad 2y &= 5 \\ 3x + \quad 4y &= 6 \end{aligned}$$

may be written in the form of $A\boldsymbol{X} = B$ as

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 5 \\ 6 \end{bmatrix}$$

where

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, X = \begin{bmatrix} x \\ y \end{bmatrix}, B = \begin{bmatrix} 5 \\ 6 \end{bmatrix}$$

Hence, the solution in MATLAB may be written as

```
>> A=[1 2;3 4]
A =
    1    2
    3    4
>> B=[5;6]
B =
    5
    6
>> X=inv(A)*B
X =
  -4.0000
   4.5000
```

Take note that the inverse syntax "***X=inv(A)*B***" can as well be written in another form as "***X=A\B***" where the latter syntax evaluates even if the matrix A does not have an inverse. But in actual sense, the two syntaxes should be able to give the same result. For example

```
>> A=[1 2;3 4]
A =
    1    2
    3    4
>> B=[5;6]
B =
    5
    6

>> X=A\B
X =
  -4.0000
   4.5000
```

For very large coefficient and ill-conditioned matrix A, this method of solving the system $A\boldsymbol{X} = B$ by the operation

$$\boldsymbol{X} = A^{-1}B$$

is not efficient. Thus, other methods exist which we shall discuss them as either iterative or non-iterative methods, derived based on programming techniques. However, since we have not gone into details yet of writing MATLAB programming files, in this unit, we only discuss these methods using MATLAB syntaxes.

a) LU Factorization method

This method is built on factorization of lower L and upper U triangular matrices of coefficient matrix A. Thus, LU factorization of a matrix $A \in \mathbb{R}^{n\times n}$ is given by

$$A = LU$$

where L and U are the lower and upper triangular matrices, respectively. The MATLAB syntax for LU factorization of matrix $A \in \mathbb{R}^{n\times n}$ is therefore

$$LU = lu(A)$$

For example

```
>> A=[1 2;3 4]
A =
    1    2
    3    4
>> [L U]=lu(A)
L =
   0.3333    1.0000
   1.0000         0
U =
   3.0000    4.0000

        0    0.6667
```

Or sometimes, LU factorization of matrix $A \in \mathbb{R}^{n\times n}$ is given by

$$A = LU = LDU$$

where D is the diagonal matrix. Hence, the MATLAB syntax for this is

$$[L, P, U] = lu(A)$$

For example

```
>> A=[1 2;3 4]
A =
    1    2
    3    4
>> [L,U,P]=lu(A)
L =
   1.0000        0
   0.3333   1.0000
U =
   3.0000   4.0000
        0   0.6667
P =
    0    1
    1    0
```

Hence, in order to solve the system $A\boldsymbol{X} = B$ by LU factorization method, we have $A\boldsymbol{X} = LU\boldsymbol{X} = B$ such that

$$\boldsymbol{X} = (LU)^{-1}B = L^{-1}U^{-1}B$$

Or alternatively, we evaluate $LU\boldsymbol{X} = B$ in two phases

- $U\boldsymbol{X} = Z$ and
- $L\boldsymbol{Z} = B$

such that the unknowns of vector $\boldsymbol{Z}$ are determined by Forward substation method, while the unknowns of vector $\boldsymbol{X}$ are determined by backward substitution.

For example, using

$$\boldsymbol{Z} = L^{-1}B, \quad \boldsymbol{X} = U^{-1}Z$$

we have

```
>> A=[1 2;3 4]
A =
    1    2
    3    4
>> [L U]=lu(A)
L =
   0.3333    1.0000
   1.0000         0
U =
   3.0000    4.0000
        0    0.6667
>> B=[5;6]
B =
    5
    6
>> Z=inv(L)*B
Z =
    6
    3
>> X=inv(U)*Z
X =
  -4.0000
   4.5000
```

b) Singular Value Decomposition technique

The singular value decomposition (SVD) of a given matrix $A \in \mathbb{R}^{n\times n}$ is given by

$$A = USV^T$$

where U is an orthogonal matrix, V is an orthogonal matrix, and S is a diagonal singular matrix such that the MATLAB syntax is defined by

$$[U, S, V] = svd(A)$$

For example,

```
>> A=[1 2;3 4]
A =
    1    2
    3    4
>> [U,S,V]=svd(A)
U =
  -0.4046   -0.9145
  -0.9145    0.4046
S =
   5.4650         0
        0    0.3660
V =
  -0.5760    0.8174
  -0.8174   -0.5760
```

Activity 2 c

a) Consider the following system of linear equations

$$\begin{cases} x + 2y + 3z = 1 \\ 4x + 5y + 6z = 1 \\ 7x + 8y \qquad = 1 \end{cases}$$

a. Derive an expression of the form $A\boldsymbol{x} = \boldsymbol{b}$ in MATLAB

b. Hence, solve for $\boldsymbol{x}$ using inverse and $A\backslash b$ methods and compare.

b) By the system of linear equations presented in a) above, compute

a. Inverse of its coefficient matrix

b. Diagonal of the coefficient matrix

c) Consider the following system of linear equations

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

a. Derive an expression of the form $A\boldsymbol{x} = \boldsymbol{b}$ in MATLAB

b. Hence, solve for $\boldsymbol{x}$ using LU factorization method and compare with $A\backslash b$ method.

d) From each matrix A of problems a) and c), deduce U, S, V via $[U, S, V] = svd(A)$.