Introduction to MATLAB

P. Sam Johnson

National Institute of Technology Karnataka (NITK)
Surathkal, Mangalore, India

Introduction

MATLAB developed by The MathWorks Inc., stands for **Matrix Laboratory**. It is a software package used to perform scientific computations and visualization. Its capability for analysis of various scientific problems, flexibility and powerful graphics makes it a very useful software package.

It provides an Integrated Development Environment (IDE) for programming with **numerous predefined functions** for technical computations and visualization.

Besides available **built-in functions**, user-defined functions can also be included which can be used just like any other built-in function.

MATLAB provides an excellent computational language.

MATLAB provides an excellent computational language, built-in state-of-the-art algorithms for mathematics and excellent visualization using ready-made functions.

Built-in functions for matrix operations such as computation of Singular Value Decomposition, LU factorization, specialized math functions like Bessel function, Beta function, Gamma function, Laplace Transforms, Fourier Transforms, Interpolation, polynomials and Ordinary Differential Equation solvers are included for saving the precious programming time of the user.

For visualization, large collections of **function programs** are included for two-dimensional and three-dimensional graphics.

Toolboxes

It provides collections of functions that act as tools for specific applications such as Control System Design, Digital Signal Processing, Neural Networks, Communications and Image Processing etc.

Such collections of programs are called '**Toolboxes**'.

In MATLAB version 7.5.0.342 (R2007b), **there are more than 50 toolboxes available** for a variety of special applications.

The toolboxes are optional and can be purchased as per user's requirement.

The basic building block in MATLAB is matrix.

As the name MATLAB comes from the two words 'MAT', which stands for Matrix, and 'LAB', which stands for Laboratory. MATLAB is Matrix Laboratory, thereby implying that it deals in matrices. Hence, **the basic building block in MATLAB is matrix.**

The basic data type, matrix, is defined as an array. The vectors, scalars etc. are all automatically handled as matrices. The scalars are treated as a matrix (array) of a single row and single column. A single-row matrix and a single-column matrix are treated as row vector and column vector respectively.

To make things simpler for the programmer, **it is not required to declare the dimensions of a matrix. There is also no need to declare any variables and their types.**

Variables can be simply used in the program anywhere one desires. MATLAB automatically sets the type of the variable according to the type

MATLAB Environment

The major components of the MATLAB environment are as follows:

- Command Window
- Command History
- Workspace Window
- Current Directory
- Editor / Edit Window
- Figure Window

Command Window

Whenever MATLAB is invoked, the main window called Command Window is activated. The Command Window displays the command prompt '>>' and a cursor where commands are entered and are executed instantaneously on pressing the 'Enter' key of the keyboard.

For example, if one wants to evaluate the expression $(20 * 3 - 5)$, it is entered at the command prompt, as shown below :

$$20 * 3 - 5.$$

The result obtained on pressing 'Enter' key is displayed in the Command Window as shown below.

```
ans =

     55
```

If the result of the expression had not been assigned to any variable, then MATLAB stores the result in a default variable, 'ans'.

The same expression can be evaluated again by assigning the result to a variable 'x', as given below:
$x = 20 * 3 - 5$.

The result obtained in the Command Window is as shown below:

$x =$

55

General Commands

The output display indicated that the results in now stored in the variable 'x'. If the output of a statement is not needed to be displayed, ':' (semicolon) is added at the end of the statement. For example,

$x = 20 * 3 - 5;$

On execution of this statement, no output is displayed in the Command Window and the result is stored in variable x.

`clock`	provides clock time and date as a vector
`date`	provides date as a strings
`ver`	gives the version of MATLAB installed

Command History Window

The Command History Window consists of a list of all the commands that are entered at the Command Window. It consists of commands of previous sessions also. These commands remain in the list until they are deleted. Any command may be executed by selecting and double clicking it with the mouse.

On right clicking the mouse, pop-up menu is displayed. A program file containing the selected commands can be created by choosing the 'Create M-file' option from the menu.

Workspace Window

A workspace is a collection of all the variables that have been generated so far in the current MATLAB session and shows their data type and size.

All the commands exectured from Command Window and all the script files exectuted from the Command Window share common workspace, so they can share all the variables. With these variables, a number of operations can be done, such as plottig by selecting a variable and right clicking the mouse (on value or the name) and selecting the desired option from the pop-up menu.

Workspace Window

The workspace information can also be obtained by typing commands at the command prompt. The '`who`' command will generate list of variables currently in the workspace.

The response is as given below:

Your variables are :

x, y, z

The '`whos`' command generates a list of variables currently in the workspace with their size (dimensions), number of bytes, and class/type of variable.

It also reports the total number of bytes used and the number of elements in the arrays.

Other Useful Commands

The other commands useful for workspace information are listed below :

`who`	lists the variables currently in the workspace \ memory
whos	same as who command but gives more information such as type and size
`what`	Lists .m, .mat and .mex files on the disk
`clear all`	Clears the variables in the workspace
`clear xyz`	Clears the variables specified in the command
`mlock fun`	Locks function fun so that it cannot be delted
`clc`	clears screan
`clf`	clears figure window

Current Directory

In the Current Directory Window, all the files and folders present in the Current Directory are lised.

All programs written in the Edit Window will automatically have (.m) extension. Once the *M*-file is saved, it can be executed by typing its name without the extension in the Command Window.

The Edit Window can also be used as a debugger to debug programs. For debugging, 'Debut' means is available on the toolbar.

Edit Window / Editor

An Edit Window is used to create a new prgram files, or to modify existing files. In this window, programs can be written, edited and saved. The programs written using the MATLAB editor are automatically assigned an extension (.m) by the editor and are known as *M*-files.

In the Edit Window, different features of the MATLAB language are shown in an *M*-file in different colours.

comments	green
variables and numbers	black
character strings	red/purple
MATLAB keywords	blue

Figure Window

A Figure Window is a sepatate window with default white background and is used to display MATLAB graphics. The results of all the graphic commands executed are displyaed in the Figure Window. **There can be any number of Figure Windows depending upon the system memory.**

For example, the following commands are given in Command Window for plotting a sine function. The output of plot function is displayed in the Figure Window.

$x = 0 : 0.05 : 20$; (x varying from 0 to 20 with step length 0.05)

$y = \sin(x)$;

plot (x, y);

Types of Files

There are three different types of files in the MATLAB :

1. *M*-files
2. *MAT*-files
3. *MEX*-files

***M*-files.** *M*-files are standard ASCII text files, with a (.m) extension to the filename. Any program written in a MATLAB editor is saved as *M*-files.

Types of Files

These files are futher classified as

- **Script Files.** An *M*-file with a set of valid MATLAB commands is called a script file. To run a script file, the filename (without the .m extension) is entered in the Command Window, just as any other MATLAB command. This is similar to executing the commands written inthe script file one by one. The script files work on global variables, that is, the variables currently present inthe workspace. These may have any number of commands including those which call built-in funcions of functions written by the user.
- **Function Files.** An *M*-file which begins with a function definitiion line is called a function file. If a file does not begin with function definition line, it becomes a script file. A function file is like a sub-program or sub-routine in FORTRAN, PASCAL, C etc. The function files can be called from the script files or another function file just as a fuction is called from another program in C.

Types of Files

MAT-files MAT-files are binary data-files, with an (.mat) extension to the filename. These are created by MATLAB when data is saved using save command. The sava command saves data from the current MATLAB workspace into a disk file. The syntax for save command is

save <filename> <var1, var2, var3, . . . >

To load the data saved in the MAT-file, the load command is used and it loads data from the file into the current MATLAB workspace. The syntax for the load command is

load <filename>

MEX-files. MEX-files is MATLAB callable FORTRAN and C program, with a (.mex) extension. This feature allows the user to integrate the code written in FORTRAN and C language into the programs developed using MATLAB.

Help Feature

To get information about any function / command of MATLAB, help feature is available.

There are three methods of getting help in MATLAB:

1. Help Browser
2. help Command
3. lookfor Command

Help Browser

The Help Browser can be opened by selecting the Help icon from the desktop toolbar, or typing doc or helpdesk or helpwin in the Command Window. Once the Help Browser is opened, the details of any particular command function can be obtained. The Help navigator contains user-selectable tabs for contents, index, search and demos and it provides search facility for seeking help about a particular command. It also provides inded of all terms for which help is available in alphabetical order for easy search.

help Command In this method, help for any particular command can be obtained by typing help followed by the function name at the command prompt.

Help Browser

For example, type

```
help sin
```

The following help text will be displayed in the Command Window:

```
SIN Sine.

SIN(X) is the sine of the elements of X

Overloaded methods

help sym/sin.m
```

This will help in understanding the function sin.

lookfor Command

In this method, help can be obtained on any particular topic by typing lookfor, followed by function name on the command prompt.

The lookfor command is different from the help command as help command searches for an exact function match, whereas the lookfor command searches every function for matching the workds that are being looked for. Hence, lookfor command is slower than the help command but it can provide useful information.

For example, if help is sought for a function inverse and if function by the naem inverse is not available in MATLAB, then help command will not provide any result. The reponse of help command will be 'inverse.m not found'.

```
help inverse

inverse.m not found.
```

lookfor Command

For example, the 'lookfor' command for the inverse function

`lookfor inverse` with give the following result:

```
INVHILB Inverse Hilbert matrix.

ACOS Inverse cosine.

INV Matrix inverse.

PINV Pseudoinverse.

IEFT Inverse discrete Fourier transform.

NCX2INV Inverse of the noncentral chi-square cdf.

IDCT2 Compute 2-D inverse discrete cosine transform.
```

For simiplicity, all lines of the MathLab response are not shown.

Help Commands

`help`	lists the topics for which help is available
`help topic`	provides help for the topic specified in the command
`lookfor string`	lists the comment line of all .m files containing the string specified in the command
`helpwin`	opens the help window
`helpdesk`	opens the web browser for online help
demo	runs the demo program

Termination Commands

`Ctrl+C`	stops execution of function / command being run currently
`quit`	quits MATLAB
`exit`	Same as quit command

Directory Commands

`pwd`	provides present working directory, same as used in Unix
`cd`	changes present working directory, same as used in MSDOS
`dir`	Lists the files / folders in the current directory, same as used in MSDOS
`path`	displays MATLAB search path
`editpath`	modifies MATLAB search path
`copyfile`	copies a files
`mkdir`	makes a directory / folder, same as used in MSDOS

Exercises

1. List the major components of the MATLAB environment.
2. What is meant by workspace? How can the information obtained regarding variables stored in the workspace?
3. What is the difference between who and whos commands?
4. How can help be sought for various commands in MATLAB? Describe various commands used for seeking help.
5. What is the difference between a script file and a data file? What are the other types of files in MATLAB?
6. What are MEX-files? Describe the advantage of using MEX-files.
7. What is the difference between a script and a function file?

MatLab is a powerful computing language.

Programming languages process data consisting of numbers, characters and strings and provide output, known as useful information. Data processing is carried out by executing a sequence of instructions called a computer program. This group of instructions or program is written by the user, using certain symbols and words according to syntax rules of a particular language.

MatLab also has its own vocabulary and grammar. We discuss concepts of character set, constants, variables, data types, operators and expressions in the lecture.

MatLab is a powerful computing language.

Any MatLab command that can be given at the command line can be used in an *M*-file. The MatLab has more features as compared to any other programming language.

It contains an extensive collection of functions that are available in many toolboxes like numerical algebra, control systems and neural networks.

The commonly used IF, WHILE and FOR loops of any programming language such as 'C' and 'PASCAL' can also be used in *M*-files for control flow, although ther is slight differences in the syntax of these programming constructs.

MatLab is also equipped with powerful debugging features to locate and remove errors in *M*-files. The debugger provides a variety of useful functions such as breakpoint and line-by-line execution.

Character Set

The character set is a set of characters that form a port of the statements written using the programming language. Characters can be broadly classified into four categories :

1. Alphabets
2. Numerals
3. Special characters
4. White space characters

MatLab is case sensitive. Numerals $0 - 9$ also form part of its character set. The special characters including the escape sequence characters form part of the character set. White space characters like tab, blank, new line, vertical tab and line-feed are also included in the character set.

Data Types, Constants and Variables

Data Types. There are 14 fundamental data types (or classes) in MatLab. Each of these types is in the form of an array. This array is a minimum single element in size and can grow to an n-dimensional array of any size. Two-dimensional (2-D) versions of these arrays are called matrices.

Constants and Variables. Constants refer to fixed values which do not change during the execution of a program. Constants can be of two types:

1. Numeric constants
2. Character constants : Character constants can be single constants, string constants or escape sequence constants (for example, the symbol '\ b' means back space, '$\backslash t$' implies tab, '$\backslash n$' implies new line and so on.

Variables

Variables form an integral form of program in any language. The different types of data are identified by their names (variable names) for case in reference. Programming languages require a programmer to declare these variables and the type of data in advance at the beginning of the program, whereas no such declaration is required in MatLab.

MatLab does not require any variable or constant to be declared in advance at the start of a program. Blank spaces cannot be included in MatLab variable names. MatLab is case sensitive.

Operators

MatLab operators can be classified mainly into three categories :

1. Arithmetic operators that perform arithmetic computations like addition, multiplication, etc.
2. Relational operators that compare operands quantitatively like 'less than', 'not equal to', etc.
3. Logical operators that perform logical operations like AND, OR, NOT, etc.

Hierarchy of Operators

Generally several arithmetic operations are combined into a single expression. An expression is calculated by executing one arithmetic operation at a time (left to right).

The order in which the arithmetic operations are executed in an expression is called ‘hierarchy of operations’ or ’operator precedence’. The rules governing the precedence of operators generally follow the normal rules of algebra.

Hierarchy of Operators

The order in which the arithmetic operations are executed are given as follows:

1. The contents of all paraentheses () are evaluated first, starting from the innermost parentheses and working outwards.
2. Transpose, power, complex conjugate transpose, matrix power
3. Unary plus (+), unary minus (−), logical NOT ( )
4. Multiplication, right division, left division, matrix multiplication, matrix right division, matrix left division
5. Addition and subtraction
6. Colon operator
7. Less than, less than or equal to, greater than, greater than or equal to, equal to and not equal to
8. Logical AND
9. Logical OR

Built-in Functions

A mathematical function is an expression which has one or more input values and gives one or more values as outputs. Engineering calculations require functions which are quite complex as compared to simple arithmetic expressions.

Examples of commonly used functions are trigonometric functions, lograthimic functions and input \ output functions.

Besides the simple arithmetic operations, a very large number and variety of useful built-in functions for complex engineering calculations related to all disciplines are available in MatLab. The list of built-in functions in MatLab is very large and richer than in most of the languages like FORTRAN or C. A complete list of all MatLab functions can be seen through help browser.

Syntax

The syntax of a built-in function is

function name (variable name or expression)

Assignment Statement. General form of an assignment statement is given as follows:

variable name = expression ;

When an assignment statement is executed, the value of the expression to the right of the equality sign is first computed and the result obtained is assigned to the variable mentioned on the left of the equality sign.

List of Commands

`abs`	returns absolute number
`any`	returns true if any element of vector is non-zero
`&`	Finds logical AND
`|`	finds logical OR
`angle`	returns phase angle in radians
`all`	returns True if all element of vector are non-zero
`factorial`	gives factorial of a number
`pol2cart`	converts polar to Cartesian coordinates
`cart2pol`	convert cartesian to polar coordinates
`conj`	gives conjugate of a complex number
`real`	returns real part of complex number
`imag`	returns imaginary part of complex number
`log`	gives natural logarithm
`sqrt`	returns square root of a number
`tand`	returns tangent of an angle given in degrees
`xor`	finds exclusive OR

In MatLab, a matrix is a rectangular array of real or complex numbers.

In MatLab, a matrix is a rectangular array of real or complex numbers. Matrices with only one row or with only one column are called row and column vectors, respectively. A matrix having only one element is called a scalar.

Although other high-level programming languages work with one number at a time, in MatLab it is possible to work with the complex matrix simultaneously. This feature is very important as it removes the unnecessary loops and repetition of same statements.

In MatLab, matrix is chosen as a basic data element. All variables when used as a single data element are treated as single element matrix, that is a matrix with one row and one column.

Scalars and Vectors

A scalar is a 1×1 matrix containing a single element only. A column vector is a $m \times 1$ matrix that has m number of rows but a single column only.

A row vector is a $1 \times n$ matrix which has n number of columns and has only one row.

Assigning data to elements of a vector / scalar. All the elements of a **row vector** are separated by blank spaces or commas and are enclosed in square brackets. All the elements of a **column vector** are separated by semicolons and are enclosed in square brackets.

A **scalar** does not need any square brackets, for example, a scalar $p = 10$, can be entered just as

```
p = 10;
```

Vector Product and Vector Transpose

Vector Product : A row vector and a column vector can be multiplied with each other. However, these vectors should be of the same length. If x is a column vector and y is a row vector, then the vector products $y * x$ and $x * y$ are different.

Vector Transpose : Transpose operation turns a row vector into a column vector and vice versa. MatLab uses the apostrophe or single quote to denote transpose.

The command

$xt = x'$ will produce the transpose of the column vector x.

If x is a vector with complex entries, its conjugate can be obtained by using `conj(x)` command and the x' produces the **conjugate transpose** of the vector x.

Creation of Evenly Spaced Row Vectors

A row vector with evenly spaced, real elements can be created by using the following command.

```
variable name = k :  1 :  m,
```

where k is the initial value of the row vector, m is the final value of the row vector and 1 is the step size or increment which comes in between initial and final values.

- **In MatLab, negative and non-integer step sizes are also allowed.**
- **If a step size is not specified, $+1$ is taken as default value of the step size.**

Creation of Evenly Spaced Row Vectors

For example, the command

```
t = 1 :  2 :  11
```

gives

```
t = [1 3 5 7 9 11]
```

The command

```
t = 11 :  1 :  5
```

will produce **empty matrix**. Care should be taken that the initial and final values given must be appropriate.

Creating a Row Vector

A row vector can also be created by using `linspace` command. The syntax is given as follows:

`linspace`$(x1, x2, n)$

It generates a linearly spaced vector of length n, that is n equally spaced points between $x1$ and $x2$.

For example, the statement

`linspace`$(0, 10, 5)$

will produce

```
a = [0    2.5    5.0    7.5    10.0]
```

Creating a Row Vector

The command

```
logspace(a, b, n)
```

creates a logarithmically spaced vector of length n in the interval 10^a to 10^b.

For example, the statement

```
x = logspace(0,4,3)
```

will produce

```
x = [1    100    10000]
```

i.e. three points are generated in the interval 10^0 and 10^4.

Some Useful Commands

Let x be a row or a column vector.

`sum(`x`)`	the sum of all elements in the row/column
`mean(`x`)`	the average of all elements in the row/column
`length(`x`)`	the number of elements in the row/column
`max(`x`)`	the maximum value in the row/column
`min(`x`)`	the minimum value in the row/column
`prod(`x`)`	the product of all elements in the row/column

Some Useful Commands

The statement

`find(`x`)`

gives the locations of non-zero entries of the row/column vector.

For example, the statements

`x = [-1 3 0 11 0]` and `find(x)`

give the output

```
x = 1 2 4
```

because x has non-zero entries in fist, second and fourth positions.

Some Useful Commands

The command

```
a=find(x > 3)
```

gives the output

```
a = 4
```

that is, fourth element of x has a value greater than 3.

Some Useful Commands

For the vector

```
z = [1.4   10.7   -1.1   20.9],
```

command	description

`fix(z)`
 	
`a= fix(z)`
	rounds the elements of z
	to the nearest integers towards zero
	`a = [1  10  -1  20]`

`floor(z)`
 	
`a= floor(z)`
	rounds the elements of z
	to the nearest integers towards $-\infty$
	`a = [1  10  -2  20]`

`ceil(z)`
 	
`a= ceil(z)`
	rounds the elements of z
	to the nearest integers towards $+\infty$
	`a = [2  11  -1  21]`

`round(z)`
 	
`a= round(z)`
	rounds the elements of z
	to the nearest integers
	`a = [1  11  -1  21]`

Some Useful Commands

The command

```
sort(z)
```

lists the elements of z in the ascending order.

The command

```
sort(z, 'descend')
```

lists the elements of z in the descending order.

Entering Data in Matrices

The following points are to be noted for entering an explicit list of matrix elements at the command window:

- Start with a left-handed square bracket '['.
- Type the matrix elements row wise. Each row vector should have the same length to create a valid matrix.
- Enter the elements of the first row with one or more blanks or a comma separating each element.
- Separate the different rows of the matrix by a semicolon (;) or a carriage return.
- Enclose the entire list of elements of the matrix with in square bracket, that is [ ].
- Elements of a matrix can be a real number, complex number or a valid MatLab expression (provided that the values of unknown variables in the expression are supplied before the execution of the expression).

Some Useful Commands

```
A = [1 3 5 ; 7 9 11 ; 2 -9 0]
```

has three rows and three columns.

```
B = [1+2i 3i ; 4+4i 5]
```

has two rows and two columns and whose elements are complex numbers.

```
C = [1 -2 ; sqrt(3) exp(1)]
```

has two rows and two columns and whose elements are expressions

Line Continuation

Sometimes, it is not possible to type the entire data input on the same line. Thus, to enter the data in continuity two methods are used:

1. Using `ellipsis`. In this method, three consecutive periods (...) are used to indicate continuation of the data to the next line. There periods are called `ellipsis`.
2. Using `carriage return`. A matrix can also be entered multiple lines using `carriage return` at the end of the each row. In this case, the semicolons and ellipses at the end of each row may be omitted.

Matrix Subscripts / Indices

The elements of a matrix can be specified / accessed by specifying their respective row and column numbers. The first index refers to the row number and second index refers to column number. The element in ith row and the jth column of a matrix A is denoted by $A(i, j)$.

If an attempt is made to access an element outside of the given dimension of matrix A, it will give an error.

For example, if matrix A has dimension 3×3 and an attempt is made to access fourth row, say, element $A(2, 3)$, then an error message:

```
'index exceeds matrix dimensions'
```

will be displayed, in the command window.

Some Useful Commands

The command $A(i, j)$ is also used to expand the given matrix size.

Suppose

`A=[6 7 ; 8 9]` and $A(2, 3) = 15$

will result in the following change in the matrix A:

```
A=

    6     7     0

    8     9    10
```

The dimension of the matrix is extended to 2×3, element 15 is stored at the index $(2, 3)$ and remaining undefined elements are taken as zero.

Sub Matrices

Subsets of arrays can be selected by the following statements.

Statement `B(1:4; 2:8)` specifies rows 1 to 4 and columns 2 to 8 of the matrix B.

The colon when typed alone refers to all the elements in a row or column of a matrix.

Thus $A(:, 2:3)$ refers to elements of all rows and columns 2 to 3 of the matrix A.

Array as a single column vector.

An array can be regarded as one long column vector, which is formed from the columns of a given matrix.

The matrix element $A(i, j)$ can also be accessed by means of a single index $A(k)$, where $k = (j-1)m + i$ and m is the number of rows.

The keyword '`end`' refers to the last row or column. The command

```
A(:, end)
```

refers to all rows and last column of the matrix A.

Multi-dimensional Matrices and Arrays

To know the size of an existing matrix A, the command `size(A)`, can be used. The format is as follows :

```
size(A)
```

The command returns the number of elements of the matrix in each dimension.

The following command

```
[m,n]=size(A)
```

will assign the number of rows of the matrix A to variable 'm' and number of columns to variable 'n'.

Multi-dimensional Arrays

Multi-dimensional arrays are an extension of two-dimensional matrices described by their row and column indices. Multi-dimensional arrays use additiional indices.

A three-dimensional array, for example, uses three indices; first index refers to rows, second refers to column and third refers to the third dimension.

For example, a book can be viewed as a collection of two-dimensional pages with the page number being the third dimension.

To access the element in the third row, fourth column of page 5, the indices will be $(3, 4, 5)$. As the dimensions of an array are increased, the number of indicies is also increased correspondingly. For example, a five-dimensional array will require five indices. The first two refer to a row and column, the next three for third, fourth and fifth dimension, respectively.

Matrix Manipulations

The shape and size of a matrix can be altered by using the following MatLab commands:

Reshaping Matrices

Matrices can be reshaped into a vector or into any other appropriately sized matrix as described below:

1. **Reshaping of a matrix as a vector.** All the elements of a matrix A can be grouped into a single column vector b by the command `b=A(:)`. Matrix A will be stored column wise in vector b, i.e. first column is stored first, then second column and so on.
2. **Reshaping of matrix as a differently sized matrix.** If a given matrix A is a $p \times q$ matrix, it can be reshaped into a new $m \times n$ matrix B, as long as total elements of the two matrices are same. i.e. $p * q = m * n$. The reshaping command is as follows:

   `B=reshape(A,m,n)`.

Expanding Matrix Size

When a single element or a few elements are entered in a matrix, MatLab creates the matrix of proper dimensions to accommodate the entered element(s). Remaining unspecified elements are assumed to be zero.

For example, suppose a matrix C does not exist and a statement

```
C(2, 2) = 10
```

is entered at the command window. In such a case, matrix C of size (2×2) is generated with element $C(2, 2) = 10$ and with all other elements equal to zero as shown below.

```
C =

     0     0

     0    10
```

Appending a row / column to a matrix.

Similarly, the command

```
D(2, 1:2) = [3  4]
```

produces

```
D

   0   0

   3   4
```

Appending a row / column to a matrix : Sometimes, it is required that a column or a row to be added to a given matrix. A column can be appended by using following command :

```
A=[A  x]
```

It will append the column vector x to the columns of existing matrix A.

Appending a row / column to a matrix.

To append a row to a given matrix, the following command is used:

```
A=[A ; y]
```

This will append the row vector y to the rows of existing matrix A. Note that semicolon has been used to append the row vector.

Deleting a row / column of a matrix. Rows and columns of a matrix can be deleted by setting the corresponding row or column vector equal to a null vector. That is, a pair of empty square brackets [ ], without any element within the brackets.

To delete the first row from matrix F and first two columns from matrix G, the following commands are used.

```
F(1, :)=[ ]
```

```
G(:, 1:2)=[ ]
```

Error

To delete the particular element, the following command is given:

```
A(1,2)=[ ]
```

But this will result in an error because a single element. That is, $A(1,2)$ cannot be removed from a matrix and following message will be displayed:

```
??  Indexed empty matrix assignment is not allowed.
```

Concatenation of Matrices

Concatenation is a method of combining or connecting or joining small matrices together to make larger or bigger matrices. The pair of square brackets [ ] is the concatenation operator.

Let matrix A be given by $A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$. We can commute the matrix

$$B = \begin{pmatrix} A_1 & A_2 \\ A_3 & A_4 \end{pmatrix}$$

where the sub-matrices $A_1 = A, A_2 = A + 12, A_3 = A + 24, A_4 = A + 10$.

All this can be done using the following statements:

```
A = [1  2 ; 3  4];

B = [A  A+12 ; A+24  A+10]
```

Generation of Special Matrices

The following MatLab commands / functions, which generate special matrices, (like zero matrix, matrix with all entries are one and the identity matrix) are often used in engineering computations involving matrices:

`A = zeros(2,3)` will generate a 2×3 matrix with all elements equal to zero.

`A = zeros(10)` will generate a 10×10 square matrix with all elements equal to zero.

`A = ones(2,3)` will generate a 2×3 matrix with all elements equal to one.

`A = ones(10)` will generate a 10×10 square matrix with all elements equal to one.

Generation of Special Matrices

`A = eye(2,3)` will generate a 2×3 matrix with all main diagonal elements equal to one.

`A = eye(10)` will generate a 10×10 square matrix with all main diagonal elements equal to zero.

Vandermonde Matrix. The command

```
V = vander(v)
```

returns Vandermonde matrix V whose columns are powers of the vector

$$v = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{pmatrix}.$$

Generation of Special Matrices

Vandermonde matrix is an $m \times n$ matrix with monomial terms of a geometric progression in each row as shown below:

$$v = \begin{pmatrix} v_1^{n-1} & \cdots & v_1^2 & v_1 & 1 \\ v_2^{n-1} & \cdots & v_2^2 & v_2 & 1 \\ \vdots & \cdots & \vdots & \vdots & \vdots \\ v_m^{n-1} & \cdots & v_m^2 & v_m & 1 \end{pmatrix}.$$

Let $v = [1\ 2\ 3]$. To generate a Vandermonde matrix, the following command is used:

```
v = [1  2  3]

vander(v)
```

Generation of Special Matrices

If v is a row or column vector with n elements, it returns a diagonal matrix, with vector v on the diagonal. The syntax for this command is

```
A = diag(v, k)
```

This will return a square matrix A, of order '$n + abs(k)$' with the elements of vector v on the $k$the diagonal.

When $k = 0$, the vector v will come in main diagonals of A. We may simply use

```
A = diag(v).
```

Generation of Special Matrices

When $k = 1$, the vector v will come in 1st upper main diagonals of A.

When $k = -1$, the vector v will come in 1st lower main diagonals of A.

Let x be an $m \times n$ matrix, then the command

```
A = diag(x)
```

will return a column vector formed from the elements on the main diagonal of the matrix x.

```
A = diag(x, k)
```

will return a column vector formed from the elements of the kth diagonal of the matrix x.

Some Useful Commands Related To Matrices

Let A be a matrix.

`det(`A`)`	the determinant of a given square matrix A
`rank(`A`)`	the rank of a given rectangular matrix A
`trace(`A`)`	the sum of the diagonal elements of a rectangular matrix A
`inv(`A`)`	the inverse of non-singular square matrix A
`norm(`A`)`	the Euclidean norm of a given rectangular matrix A
A'`(`A `transpose)`	the transpose of a given rectangular matrix A

Some Useful Commands Related To Matrices

Let A be a matrix.

$x = A\backslash b$ (Left Division)	the solution vector x of a linear system $Ax = b$

`poly(`A`)`
`p = poly(`A`)`
gives `p = 1 -5 4`
	the coefficients of the characteristic polynomial of a given square matrix A
	The characteristic polynomial is $s^2 - 5s + 4$.

`eig(`A`)`	the eigenvalues of A
`[v, x] = eig(A)`	returns the eigenvectors v (columns of v); eigenvalues are main diagonal of matrix x
`eigs(`A`)`	returns six largest magnitude eigenvalues of A
`org(`A`)`	orthogonalizes the rectangular matrix A

References

- Misza Kalechman, "*Practical MATLAB Basics for Engineers*", CRC Press, London, 2009.
- Raj Kumar Bansal, Ashok Kumar Goel and Manoj Kumar Sharma, "*MATLAB and its applications in engineering*", Pearson, New Delhi, 2009.