SVKM'S NMIMS Deemed to be UNIVERSITY | NAVI MUMBAI

MATLAB
Unit 2-Lecture 2

BTech (CSBS) -Semester VII

15 July 2022, 09:35AM

*few figures/content have been prepared/referred from internet source/books

Predefined variables

variable	description
ans	Value of last expression
eps	Smallest difference between 2 numbers
i	√-1
inf	Infinity
j	Same as i
NaN	Not a number
pi	The number ∏

Some Useful MATLAB commands

- who List known variables
- whos List known variables plus their size
- help >> help sqrt (Help on using sqrt)
- clear Clear all variables from work space
- clear x y Clear variables x and y from work space
- clc Clear the command window

Variable and assignment statement

variable name = expression

Command window:

```
>> mynum = 6
mynum =
     6
>>
```

Correction: The variable name must always be written on left, and expression on wright.

Now write in Command window:

```
>> 6 = mynum
   6 = mynum
     |
Error: The expression to the left of the equals sign is not a valid target
for an assignment.
>>
```

Initializing, incrementing & decrementing

Frequently, values of variables change, as shown previously. Putting the first or initial value in a variable is called ***initializing*** the variable.

Adding to a variable is called ***incrementing***. For example, the statement

```
mynum = mynum + 1
```

increments the variable *mynum* by 1.

Similarly, ***mynum=mynum-1***, will be ***decrementing*** variable.

Floating-Point Numbers

For floating point number there are two basic types:

- Double-Precision Floating Point
- Single-Precision Floating Point

The integer type are **int8, int16, int32, int64.**
These integers represent the **bits** used to store the value of data type.

type **char** is used to store the **character or string** eg. ‘cat’
type **logical** is used to store true/false.

Numerical Expression

Expressions can be created using values, variables that have already been created, operators, built-in functions, and parentheses. For numbers, these can include operators such as multiplication and functions such as trigonometric functions. An example of such an expression is:

```
>>2 * sin(1.4)
ans =
    1.9709
```

Fromat Command

This will remain in effect until the format is changed back to **short**, as demonstrated in the following.

```
>> format long
>>2 * sin(1.4)
ans =
    1.970899459976920

>> format short
>> 2 * sin(1.4)
ans =
    1.9709
```

Fromat Command

The **format** command can also be used to control the spacing between the MATLAB command or expression and the result; it can be either **loose** (the default) or **compact**.

```
>> format loose
>> 5*33
ans =

   165

>> format compact
>> 5*33
ans =
   165
>>
```

Nested Parentheses

Within a given precedence level, the expressions are evaluated from left to right (this is called *associativity*).

For the operators that have been covered thus far, the following is the precedence (from the highest to the lowest):

( )	parentheses
^	exponentiation
–	negation
*, /, \	all multiplication and division
+, –	addition and subtraction

Operator precedence rule:

Operators	Precedence
Parentheses: ( )	Highest
Power ^	
Unary: Negation (-), not (~)	
Multiplication, division *, /, \	
Addition, subtraction +, -	
Relational <, <=, >, >=, ==, ~=	
And &&	
Or ||	
Assignment =	Lowest

Practice problem:

1. Think about what the results would be for the following expressions, and then type them in to verify your answers:

```
1\2
-5 ^ 2
(-5) ^ 2
10 - 6/2
5 * 4/2 * 3
```

2. What would happen if you use the name of a function , eg abs, as a variable name?
3. Use plus operator and check the results.

Also, if a function name is typed incorrectly, MATLAB will suggest a correct name.

```
>> abso(-4)
Undefined function or variable 'abso'.
Did you mean:
>> abs(-4)
```

Constant/random number

pi	3.14159....
i	$\sqrt{-1}$
j	$\sqrt{-1}$
inf	infinity ∞
NaN	stands for "not a number," such as the result of 0/0

```
>> rand
ans =
    0.8147
>> rand
ans =
    0.9058
```

Practice problem:

Generate a random

- real number in the range (0,1)
- real number in the range (0, 100)
- real number in the range (20, 35)
- integer in the inclusive range from 1 to 100
- integer in the inclusive range from 20 to 35

Relational Expression

Expressions that are conceptually either true or false are called ***relational expressions***; they are also sometimes called ***Boolean expressions*** or ***logical expressions***. These expressions can use both ***relational operators***, which relate two expressions of compatible types, and ***logical operators***, which operate on **logical** operands.

The relational operators in MATLAB are:

Operator	Meaning
>	greater than
<	less than
>=	greater than or equals
<=	less than or equals
==	equality
~=	inequality

Example:

```
>> 3 < 5
ans =
     1

>> 2 > 9
ans =
     0
>> class(ans)
ans =
logical
```

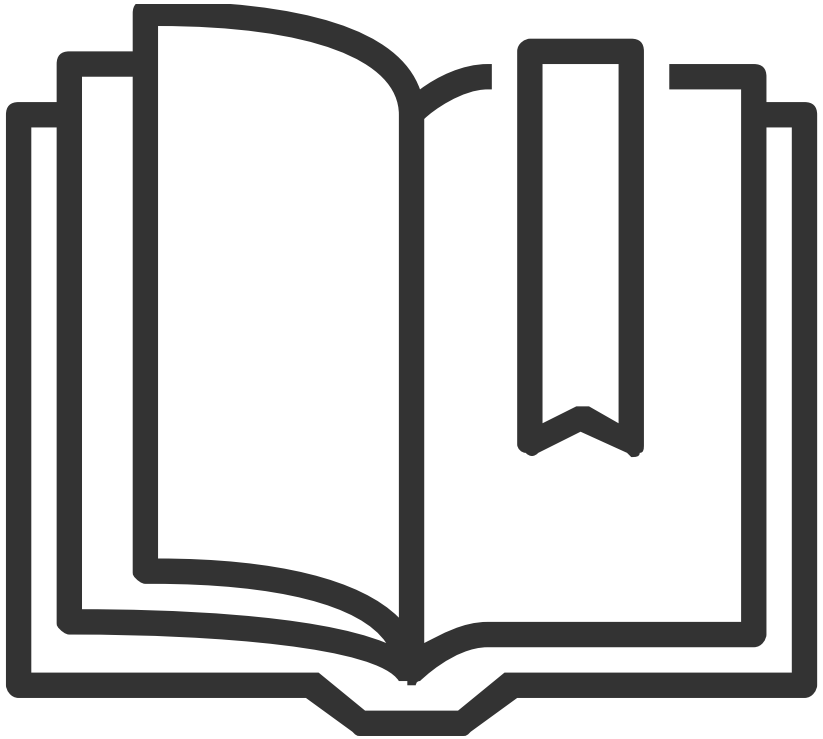
Practice question:

1. Assume that there is variable x that has been initialized, what would be the value of expression 3<x<5, if the value of x is 4? what if the value is 7?

Practice question:

Think about what would be produced by the following expressions, and then type them in to verify your answers.

```
3 == 5 + 2

'b' < 'a' + 1

10 > 5 + 2

(10 > 5) + 2

'c' == 'd' - 1 && 2 < 4

'c' == 'd' - 1  ||  2 > 4

xor('c' == 'd' - 1, 2 > 4)

xor('c' == 'd' - 1, 2 < 4)

10 > 5 > 2
```



Thank you for listening

10:45AM | IT Workshop/MATLAB

SVKM'S NMIMS Deemed to be UNIVERSITY | NAVI MUMBAI

MATLAB
Unit 2-Lecture 3

BTech (CSBS) -Semester VII

19 July 2022, 09:35AM

*few figures/content have been prepared/referred from internet source/books

Creating 2D array(matrix)

Variable_name = [1st row elements; 2nd row elements;……;last row elements]

- Matrix are used in science & eng to describe many physical quantities.
- All rows must have same number of elements

Zeros , ones and eye commands

- **zeros(m,n):** *mxn* matrix of 0's.
- **ones(m,n):** *mxn* matrix of 1' s.
- **eye(n):** *nxn* identity matrix

Array

List of numbers arranged in row and/or columns.

Simplest array

-1D array

-usually to represent vectors.

Complex array

-2D array

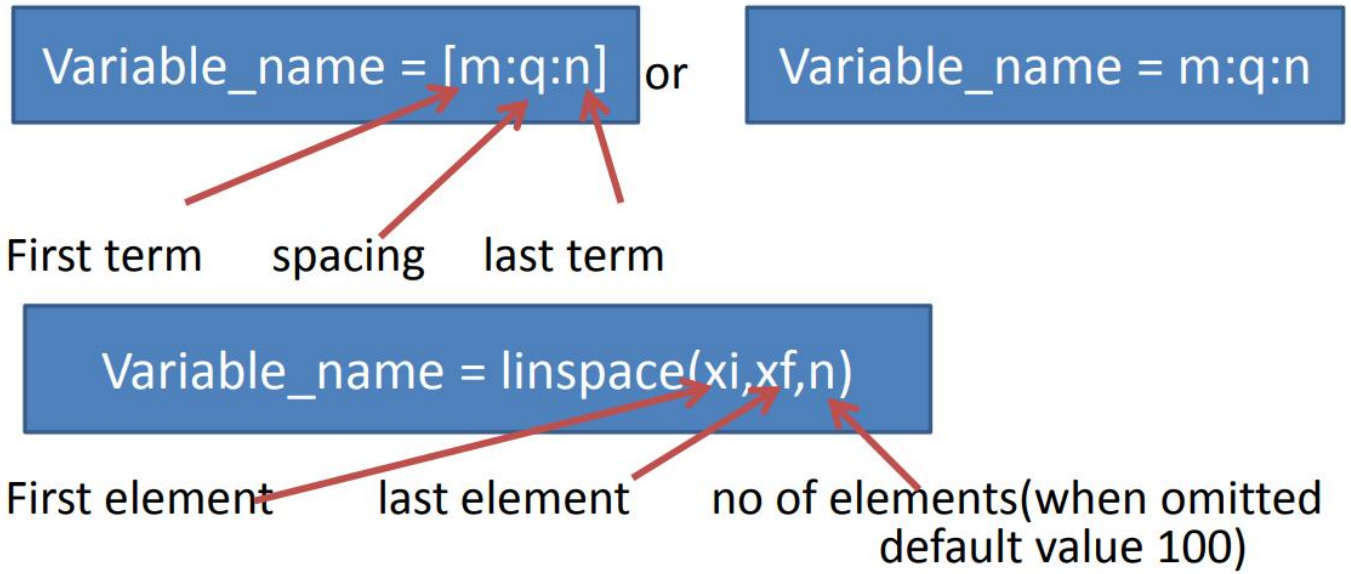
-represent matrixes

Creating vector from a known list of numbers

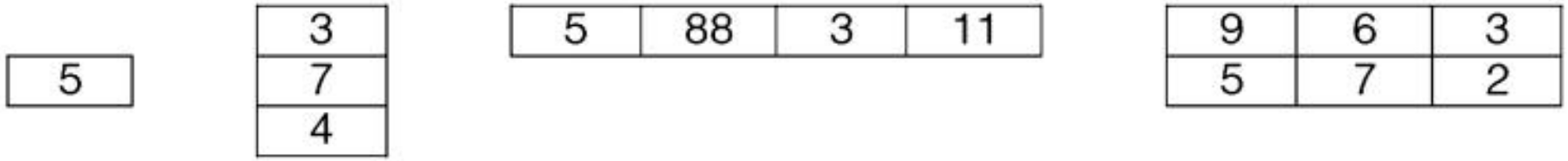
Variable_name = [type vector elements]

Row vector :- type elements with space or comma

Column vector :- type elements with semicolon (;) or press Enter key after each element



Vector and Matrix



The scalar is 1×1, the column vector is 3×1 (three rows by one column), the row vector is 1×4 (one row by four columns), and the matrix is 2×3 (two rows by three columns). All of the values stored in these matrices are stored in what are called ***elements***.

Creating row vector

There are several ways to create row vector variables. The most direct way is to put the values that you want in the vector in square brackets, separated by either spaces or commas. For example, both of these assignment statements create the same vector *v*:

```
>> v = [1   2   3   4]
   v =
     1   2   3   4

>> v = [1,2,3,4]
   v =
     1   2   3   4
```

Colon Operator

If, as in the preceding examples, the values in the vector are regularly spaced, the ***colon operator*** can be used to ***iterate*** through these values. For example, 2:6 results in all of the integers from 2 to 6 inclusive:

```
>> vec = 2:6
vec =
     2     3     4     5     6
```

In this vector, there are five elements; the vector is a 1×5 row vector. Note that in this case, the brackets [ ] are not necessary to define the vector.

With the colon operator, a ***step value*** can also be specified by using another colon, in the form (first:step:last). For example, to create a vector with all integers from 1 to 9 in steps of 2:

```
>> nv = 1:2:9
nv =
     1   3   5   7   9
```

Line space function

The **linspace** function creates a linearly spaced vector; **linspace(x,y,n)** creates a vector with *n* values in the inclusive range of *x* to *y*. If *n* is omitted, the default is 100 points. For example, the following creates a vector with five values linearly spaced between 3 and 15, including the 3 and 15:

```
>> ls = linspace(3,15,5)
ls =
     3     6     9    12    15
```

Concatenating Vector

Vector variables can also be created using existing variables. For example, a new vector is created here consisting first of all of the values from *nv* followed by all values from *ls*:

```
>> newvec = [nv ls]
newvec =
   1   3   5   7   9   3   6   9   12   15
```

Putting two vectors together like this to create a new one is called ***concatenating*** the vectors.

Logspace function

Similarly, the **logspace** function creates a logarithmically spaced vector; **logspace(x,y,n)** creates a vector with *n* values in the inclusive range from 10^x to 10^y. If *n* is omitted, the default is 50 points. For example, logspace (1,4,4) creates a vector with four elements, logarithmically spaced between 10^1 and 10^4, or in other words 10^1, 10^2, 10^3, and 10^4.

```
>> logspace(1,4,4)
ans =
          10         100        1000       10000
```

SVKM'S NMIMS Deemed to be UNIVERSITY | NAVI MUMBAI

MATLAB
Unit 2-Lecture 4

BTech (CSBS) -Semester VII

22 July 2022, 09:35AM

*few figures/content have been prepared/referred from internet source/books

Workspace

- Managing the workspace
- Keeping track of your work session,
- Entering multiple statements per line

Workspace Browser

- The Workspace browser enables you to view and interactively manage the contents of the workspace in MATLAB®.
- For each variable or object in the workspace, the Workspace browser also can display statistics, when relevant, such as the minimum, maximum, and mean.
- You can edit the contents of scalar (1-by-1) variables directly in the Workspace browser. Right-click the variable and select Edit Value.
- To edit other variables, double-click the variable name in the Workspace browser to open it in the Variables editor.

Workspace Browser

Workspace

Name	Value	Class	Min	Max	Mean
A	*4x4 double*	double	1	16	8.5000
B	[1;2;3;4]	double	1	4	2.5000
filename	'myfile.txt'	char			
patient	*1x1 struct*	struct			
t	'Hello'	char			
val1	*2x3 cell*	cell			
val2	[17,21,42]	double	17	42	26.6667
x	325	double	325	325	325
y	[9900,26025,39600]	uint32	9900	39600	
z	-Inf	double	-Inf	-Inf	-Inf

Open the Workspace Browser

To open the Workspace browser if it is not currently visible, do one of the following:

- MATLAB Toolstrip: On the **Home** tab, in the **Environment** section, click **Layout**. Then, in the **Show** section, select **Workspace**.

- MATLAB command prompt: Enter workspace.

You also can minimize the Workspace browser by collapsing the panel in which it resides. eg. if the Workspace browser is in the left side panel, click the button at the bottom left corner of the panel to collapse the panel. To restore the panel, click the button. If the Workspace browser is in the left or right side panel and the panel contains multiple tools, you also can minimize it by clicking the button to the left of the Workspace browser title bar.

Workspace Variables

Functions

`load`	Load variables from file into workspace
`save`	Save workspace variables to file
`matfile`	Access and change variables in MAT-file without loading file into memory
`disp`	Display value of variable
`formattedDisplayText`	Capture display output as string
`who`	List variables in workspace
`whos`	List variables in workspace, with sizes and types
`clear`	Remove items from workspace, freeing up system memory
`clearvars`	Clear variables from memory
`openvar`	Open workspace variable in Variables editor or other graphical editing tool
Workspace Browser	Open Workspace browser to manage workspace

Save Workspace Variables

There are several ways to save workspace variables interactively:

- To save all workspace variables to a MAT-file, on the **Home** tab, in the **Variable** section, click **Save Workspace**.

- To save a subset of your workspace variables to a MAT-file, select the variables in the Workspace browser, right-click, and then select **Save As**. You also can drag the selected variables from the Workspace browser to the Current Folder browser.

- To save variables to a MATLAB script, click the **Save Workspace** button or select the **Save As** option, and in the **Save As** window, set the **Save as type** option to **MATLAB Script.** Variables that cannot be saved to a script are saved to a MAT-file with the same name as that of the script.

Save Workspace Variables

You also can save workspace variables programmatically using the **save** function.

```
save('june10')
```

To save only variables A and B to the file june10.mat, use the command

```
save('june10', 'A' , 'B')
```

Load Workspace Variables

- To load saved variables from a MAT-file into your workspace, double-click the MAT-file in the Current Folder browser.

- To load a subset of variables from a MAT-file on the **Home** tab, in the **Variable** section, click **Import Data**. Select the MAT-file you want to load and click **Open**. You also can drag the desired variables from the Current Folder browser Details panel of the selected MAT-file to the Workspace browser. The Details panel is not available in MATLAB Online.

- To load variables saved to a MATLAB script into the workspace, simply run the script.

Load Workspace Variables

You also can load saved variables programmatically, use the load function.

```
load('durer')
```

To load variables X and map from the file durer.mat

```
load('durer', 'X', 'map')
```

Keep a track of work

Write to a Diary File

To keep an activity log of your MATLAB® session, use the **diary** function. diary creates a verbatim copy of your MATLAB session in a disk file (excluding graphics).

entering multiple statements per line

- **Enter Multiple Lines Without Running Them**
- To enter multiple lines before running any of them, use **Shift+Enter** or **Shift+Return** after typing a line. This is useful, for example, when entering a set of statements containing keywords, such as if ... end. The cursor moves down to the next line, which does not show a prompt, where you can type the next line. Continue for more lines. Then press **Enter** or **Return** to run all of the lines.

- This allows you to edit any of the lines you entered before you pressing **Enter** or **Return**.

- **Entering Multiple Functions in a Line**
- To enter multiple functions on a single line, separate the functions with a **comma** (,) or **semicolon** (;). Using the semicolon instead of the comma will suppress the output for the command preceding it. For example, put three functions on one line to build a table of logarithms by typing

```
format short; x = (1:10)'; logs = [x log10(x)]
```

and then press **Enter** or **Return**. The functions run in **left-to-right order**.

entering multiple statements per line

- **Entering Long Statements**
- For items in single quotation marks, such as strings, you must complete the string in the line on which it was started. For example, completing a string as shown here

```
headers = ['Author Last Name, Author First Name, ' ...
'Author Middle Initial']
results in

headers =
Author Last Name, Author First Name, Author Middle Initial
```

SVKM'S NMIMS Deemed to be UNIVERSITY | NAVI MUMBAI

MATLAB
Unit 3-Lecture 5

BTech (CSBS) -Semester VII

26 July 2022, 09:35AM

*few figures/content have been prepared/referred from internet source/books

Unit 3

- Matrix
- Array
- Basic mathematical functions

An array is MATLAB's basic data structure

Can have any number of dimensions. Most common are

- vector - one dimension (a single row or column)
- matrix - two or more dimensions
- Scalar - matrices with only one row and one column.

Arrays can have numbers or letters

Creating Matrices

In MATLAB, a vector is created by assigning the elements of the vector to a variable. This can be done in several ways depending on the source of the information.

—Enter an explicit list of elements

—Load matrices from external data files

—Using built-in functions

—Using own functions in M-files

A matrix can be created in MATLAB by typing the elements (numbers) inside square brackets [ ]

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

Creating Matrices

```
>> A = [2 -3 5; -1 4 5]  % Note MATLAB displays column vector vertically
A=
2 -3 5
-1 4 5
>> x = [1 4 7] % Note MATLAB displays row vector horizontally
x=
4
1 4 7
>> x = [1; 4; 7]  %Optional commas may be used between the elements.Type the semicolon (or
                                press Enter) to move to the next row
x=
1
4
7
```

Creating Matrices

```
>> cd=6; e=3; h=4;

>> Mat=[e cd*h cos(pi/3);h^2 sqrt(h*h/cd) 14]

Mat =
    3.0000   24.0000    0.5000
   16.0000    1.6330   14.0000
```

Concatenation of Matrices

```
Command Window
>>  a=[1 2;3 4];
b=[4 6 ;8 9];
A=[a,b]

A =

     1     2     4     6
     3     4     8     9

>> B=[a;b]

B =

     1     2
     3     4
     4     6
     8     9

fx >>
```

Row wise concate

Coloumn wise concate

Colon operator

The colon operator can be used to create a vector with constant spacing

```
x = m:q:n
```

- `m` is first number
- `n` is last number
- `q` is difference between consecutive numbers

```
>>x=[1:2:10]
x =
  1   3   5   7   9
```

If omit q, spacing is one

```
y = m:n
```

```
>> y=1:5
y =
  1 2 3 4 5
```

Colon operator

```
>> x=1:5:50

x =

     1     6    11    16    21    26    31    36    41    46

>> 1:5:50

ans =

     1     6    11    16    21    26    31    36    41    46
```

Question 1

How can you use the colon operator to generate the vector shown below?

```
9   7   5   3   1
```

linspace function

v = linspace(x_i, x_f, n)

- x_i is first number
- x_f is last number
- n is number of terms

(= 100 if omitted)

```
>> linspace (4,8,50)

ans =

  Columns 1 through 11

    4.0000    4.0816    4.1633    4.2449    4.3265    4.4082    4.4898    4.5714    4.6531    4.7347    4.8163

  Columns 12 through 22

    4.8980    4.9796    5.0612    5.1429    5.2245    5.3061    5.3878    5.4694    5.5510    5.6327    5.7143

  Columns 23 through 33

    5.7959    5.8776    5.9592    6.0408    6.1224    6.2041    6.2857    6.3673    6.4490    6.5306    6.6122

  Columns 34 through 44

    6.6939    6.7755    6.8571    6.9388    7.0204    7.1020    7.1837    7.2653    7.3469    7.4286    7.5102

  Columns 45 through 50

    7.5918    7.6735    7.7551    7.8367    7.9184    8.0000
```

Misc Matrix

- zeros(r,c) - makes matrix of r rows and c columns, all with zeros
- ones(r,c) - makes matrix of r rows and c columns, all with ones
- rand(r,c) - makes matrix of r rows and c columns, with random numbers
- eye(n) - makes square matrix of n rows and columns. Main diagonal (upper left to lower right) has ones, all other elements are zero
- magic(n) - makes a special square matrix of n rows and c columns, called Durer's matrix

Misc Matrix

- zeros(r,c) - makes matrix of r rows and c columns, all with zeros
- ones(r,c) - makes matrix of r rows and c columns, all with ones
- rand(r,c) - makes matrix of r rows and c columns, with random numbers
- eye(n) - makes square matrix of n rows and columns. Main diagonal (upper left to lower right) has ones, all other elements are zero
- magic(n) - makes a special square matrix of n rows and c columns, called Durer's matrix

Misc Matrix

```
>> a=zeros(4,3)

a =

     0     0     0
     0     0     0
     0     0     0
     0     0     0
```

```
>> b=ones(4,3)

b =

     1     1     1
     1     1     1
     1     1     1
     1     1     1
```

```
>> c=rand(4,3)

c =

    0.8147    0.6324    0.9575
    0.9058    0.0975    0.9649
    0.1270    0.2785    0.1576
    0.9134    0.5469    0.9706
```

```
>> d=eye(4)

d =

     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
```

```
>> e=magic(4)

e =

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

Creating Matrix variable

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

```
>> mat = [3 5 7; 1 2]
Error using vertcat
Dimensions of matrices being concatenated are not consistent.
```

```
>> mat = [2:4; 3:5]
mat =
    2   3   4
    3   4   5
```

Linear indexing

```
>> intmat = [100 77; 28 14]
intmat =
    100     77
     28     14
>> intmat(1)
an =
    100
>> intmat(2)
ans =
     28
>> intmat(3)
ans =
     77
>> intmat(4)
ans =
     14
```

Dimension

```
>> vec = -2:1
vec =
   -2   -1   0   1
>> length(vec)
ans =
   4
>> size(vec)
ans =
   1   4
```

```
>> mat = [1:3; 5:7]'
mat =
   1   5
   2   6
   3   7

>> [r, c] = size(mat)
r =
  3
c =
  2
```

```
>> size(mat)
ans =
   3   2
```

Question

How could you create a matrix of zeros with the same size as another matrix?

numel function

```
>> v=9:-2:1

v =

     9     7     5     3     1

>> numel(v)

ans =

     5
```

For vectors, **numel** is equivalent to the **length** of the vector. For matrices, it is the product of the number of rows and columns.

Question

```
mat = [1:3; 44 9  2; 5:-1:3]
mat(3,2)
mat(2,:)
size(mat)
mat(:,4) = [8;11;33]
numel(mat)
v = mat(3,:)
v(v(2))
v(1) = []
reshape(mat,2,6)
```

SVKM'S NMIMS Deemed to be UNIVERSITY | NAVI MUMBAI

MATLAB
Unit 3-Lecture 6

BTech (CSBS) -Semester VII

29 July 2022, 09:35AM

*few figures/content have been prepared/referred from internet source/books

Question

1) As the functions operate , how can we get an overall result for the matrix? Determine the overall maximum in the matrix?
2) Find the cummulative matrix from (1).
3) For vector *v* with a length *n*, *diff(v)* will be *n-1*. Create a random integer matrix and find difference on each coloumn.
4) Create a matrix of all 10's.
5) Create a vector variable and substract 3 from every element.

Question

6) Create a matrix variable and divide every element by 3.

7) Create a matrix variable and square every element.

8) You are provided with following vector

```
>>vec=[5 9 3 4 6 11]

>>v=[0 1 0 0 1 1]

>>vec(v)
```

Error: Array indices must be positive integers or logical values.

Define this error and give correction for it

Question

9) Find logical true or false of *vec* from (8), for value greater than 9

10) Find same for value less than 9

11) Assume a vector *vec* that erronouesly stores negative values, how can we elliminate those negative values?

12) With same *vec* from (11), using ‘find’ command and logical operators, instead of deleting negative values, retain only the positive values only.

Question

13) When two matrix have same dimension and are square, both array and matrix multiplication can be performed on them. For the following two matrices, perform A.*B, A*B and B*A.

$$A = \begin{bmatrix} 1 & 4 \\ 3 & 3 \end{bmatrix} \text{ and B} = \begin{bmatrix} 1 & 2 \\ -1 & 0 \end{bmatrix}$$

SVKM'S NMIMS Deemed to be UNIVERSITY | NAVI MUMBAI

MATLAB
Unit 3-Lecture 7

BTech (CSBS) -Semester VII

2 August 2022, 09:35AM

*few figures/content have been prepared/referred from internet source/books

Matrices

1. Create a matrix, *mat*. Is there a **rot180** function? Is there a rot90 function (to rotate clockwise)?
2. Use above *mat,* and try to flip the matrix from left to right.
3. Use same *mat*, and flip up to down.
4. Create an empty vector and find the length of this vector.
5. Create a vector in 1 to 10 range and delete 4 element from same vector.

Matrices

6. Create a vector, vec, ranging 3 to 15. Delete a subset vector range from 9 to 12
7. From above given vector, check if individual elements could be deleted.
8. From the matrix as obtained in (1), remove the second coloumn.
9. Find the absolute vaule of vector in range -5 to 1
10. Find the sign of below given matrix

$$\begin{matrix} -4 & 2 & 8 \\ 0 & -10 & -42 \\ -9 & 15 & 0 \end{matrix}$$

Matrices

11. Find if below strings are equal:

str1= “hello”;

str2= “howdy”;

SVKM'S NMIMS Deemed to be UNIVERSITY | NAVI MUMBAI

MATLAB
Unit 3-Lecture 8

BTech (CSBS) -Semester VII

5 August 2022, 09:35AM

*few figures/content have been prepared/referred from internet source/books

Solving Linear Equation

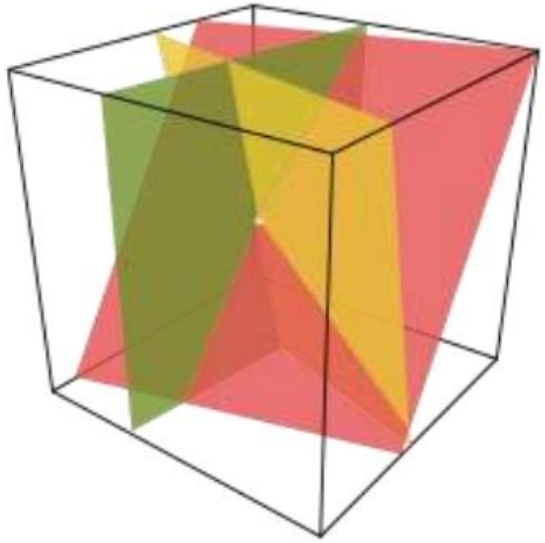
Solve System of Linear Equations Using linsolve:

A system of linear equations

$$a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1$$
$$a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2$$
$$\ldots$$
$$a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n = b_m$$

can be represented as the matrix equation $A.\vec{x} = \vec{b}$ where A is the coefficient matrix,

Solving Linear Equation...contd.



- Ex: Find values $x_1, x_2, x_3 \in \mathbb{R}$ that satisfy
  - $3x_1 + 2x_2 - x_3 - 1 = 0$
  - $2x_1 - 2x_2 + x_3 + 2 = 0$
  - $-x_1 - \frac{1}{2}x_2 - x_3 = 0$

Solution:

- Step 1: write the system of linear equations as a matrix equation

$$A = \begin{bmatrix} 3 & 2 & -1 \\ 2 & -2 & 1 \\ -1 & -\frac{1}{2} & -1 \end{bmatrix}, x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, b = \begin{bmatrix} 1 \\ -2 \\ 0 \end{bmatrix}.$$

- Step 2: Solve for $Ax = b$

Example 1

```
>> A = [5 -3 2; -3 8 4; 2 4 -9];   % Enter matrix A
>> b = [10; 20; 9];                % Enter column vector b
>> x = A\b                         % Solve for x
x =
    3.4442
    3.1982
    1.1868

>> c = A*x                         % check the solution
c =
   10.0000
   20.0000
    9.0000
```

The backslash (\) or the left division is used to solve a linear system of equations [**A**]{**x**} = {**b**}. For more information, type: `help slash`.

Question 1

- What's the A, x, and b for the following linear equations?
  - $2x_2 - 7 = 0$
  - $2x_1 - 3x_3 + 2 = 0$
  - $4x_2 + 2x_3 = 0$
  - $x_1 + 3x_3 = 0$

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

Question

LinearEquation.m

```
1  syms x y z
2  eqn1= 5*x + y + 4*z == 12;
3  eqn2= -x + y - 2*z == 43;
4  eqn3= x - y + z==-10;
5  [A,B]=equationsToMatrix([eqn1,eqn2,eqn3], [x,y,z])
6  X=linsolve(A,B)
```

Command Window

```
>> LinearEquation

A =

[ 5,  1,  4]
[-1,  1, -2]
[ 1, -1,  1]


B =

 12
 43
-10


X =

167/6
 29/6
  -33
```

Solve System of Linear Equations Using solve

Use **solve** instead of linsolve if you have the equations in the form of expressions and not a matrix of coefficients.

```
8   %---------using solve function--------
9   sol=solve([eqn1, eqn2, eqn3], [x,y,z])
10  xSol = sol.x
11  ySol = sol.y
12  zSol = sol.z
```

```
sol =

  struct with fields:

    x: 167/6
    y: 29/6
    z: -33


xSol =

167/6


ySol =

29/6


zSol =

-33
```

Question

1. **Linear algebraic equations:** Find the solution of the following set of linear algebraic equations, as advised below.

$$\begin{aligned} x + 2y + 3z &= 1 \\ 3x + 3y + 4z &= 1 \\ 2x + 3y + 3z &= 2. \end{aligned}$$

- Write the equation in matrix form and solve for $\mathbf{x} = [x\ y\ z]^T$ using the left division \.

Gaussian elimination method

MATLAB has a built-in function, rref, that does precisely this reduction, i.e., transforms the matrix to its row reduced echelon form.

```
A=[1 2 3; 3 3 4; 2 3 3];
B=[1; 1; 2];
%C=B\A
%linsolve(A,B)
%transpose (C)
%-------------Gaussian ellimination method
C=[A B];
Cr=rref(C)
```

```
Cr =

    1.0000         0         0   -0.5000
         0    1.0000         0    1.5000
         0         0    1.0000   -0.5000
```

Find eigenvalues and eigenvectors

Step 1: Enter matrix A and type [V, D] = eig(A)

```
>> A = [ 5 -3 2; -3 8 4; 2 4 -9];
>> [V, D] = eig(A)
V =

    -0.1709    0.8729    0.4570
    -0.2365    0.4139   -0.8791
     0.9565    0.2583   -0.1357
D =

   -10.3463         0         0
          0    4.1693         0
          0         0   10.1770
```

Step 2: Extract what you need:

'V' is an 'n x n' matrix whose columns are eigenvectors

D is an 'n x n' diagonal matrix that has the eigenvalues of 'A' on its diagonal.

Question

2. **Eigenvalues and eigenvectors:** Consider the following matrix.

$$\mathbf{A} = \begin{bmatrix} 3 & -3 & 4 \\ 2 & -3 & 4 \\ 0 & -1 & 1 \end{bmatrix}$$

- Find the eigenvalues and eigenvectors of $\mathbf{A}$.
- Show, by computation, that the eigenvalues of $\mathbf{A}^2$ are square of the eigenvalues of $\mathbf{A}$.
- Compute the square of the eigenvalues of $\mathbf{A}^2$. You have now obtained the eigenvalues of $\mathbf{A}^4$. From these eigenvalues, can you guess the structure of $\mathbf{A}^4$?
- Compute $\mathbf{A}^4$. Can you compute $\mathbf{A}^{-1}$ without using the `inv` function?

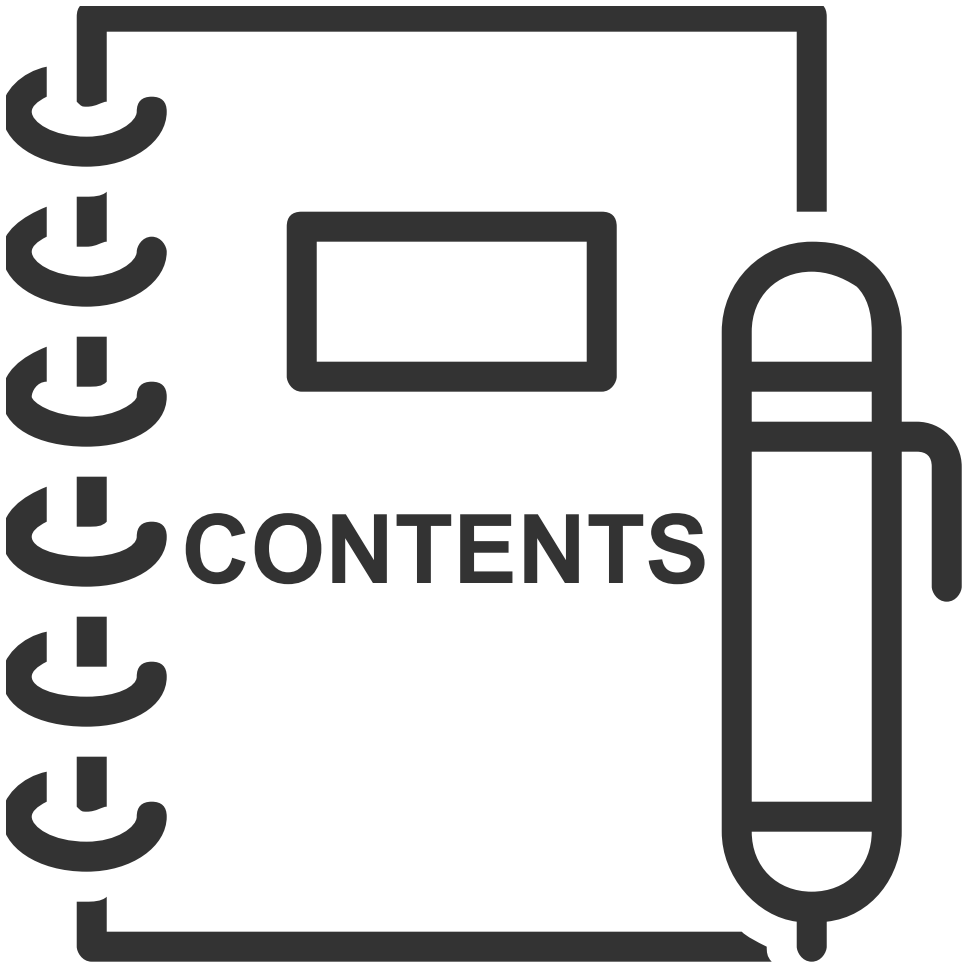
SVKM'S NMIMS Deemed to be UNIVERSITY | NAVI MUMBAI

MATLAB
Unit 1-Lecture 1

BTech (CSBS) -Semester VII

12 July 2022, 09:35AM

*few figures/content have been prepared/referred from internet source/books



- Teaching plan
- Assessment analysis
- Text/Reference books
- Unit 1-Lecture 1

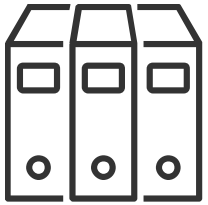
Teaching plan/Assessment analysis

Teaching plan

Teaching Scheme				Evaluation Scheme	
Lecture (Hours/week)	**Practical (Hours/week)**	**Tutorial (Hours/week)**	**Credit**	**Internal Continuous Assessment (ICA) As per Institute Norms (50 Marks)**	**Theory (3 Hrs, 100 Marks)**
2	2	0	3	Marks Scaled to 50	Marks Scaled to 50

Assessment analysis

Assessment Component	**ICA (100 Marks) (Marks scaled to 50)**					**TEE (100 marks) (Marks scaled to 50)**
	Lab Performance	**Lab Exam and Viva**	**Research activity (beyond syllabus)**	**Class Test 1 and Class Test 2**	**Class Partition**	
Weightage	10%	5%	10%	20%	5%	50%
Marks	20	10	20	20+20	10	100
Date/week of activity	Weekly	10^{th} and 11^{th}week	14^{th} week	T1: 16-23 August, 2022 T2: 10-15 October, 2022	2^{nd} and 13^{th} Week	16^{th} Nov.2022 to 2^{nd} Dec., 2022



Text/Reference books

Text Books:

1) Rafael C. Gonzalez, Richard E. Woods, Steven Eddins, "*Digital Image Processing using MATLAB*", Pearson Education, Inc., Second Edition, 2004.
2) Stormy Attaway, Butterworth-Heinemann, "*MATLAB: A Practical Introduction to Programming and Problem Solving*", Butterworth-Heinemann is an imprint of Elsevier, Fourth Edition, 2017.

Reference Books:

1) Clever Moler, *"Experiments with Matlab"*, MathWorks, Inc., 2011.

Unit 1-Lecture 1→Agenda

1) Desktop Basics
2) Numbers &Arithmetic Operations
3) Workspace Variables

1: Introduction

When you start MATLAB, the desktop appears in its default layout.

- The desktop includes these panels:
- **Current Folder** — Access your files.
- **Command Window** — Enter commands at the command line, indicated by the prompt (>>).
- **Workspace** — Explore data that you create or import from files.
- **Command History** — View or rerun commands that you entered at the command line.

Command Window

- type commands

Current Directory

- View folders and m-files

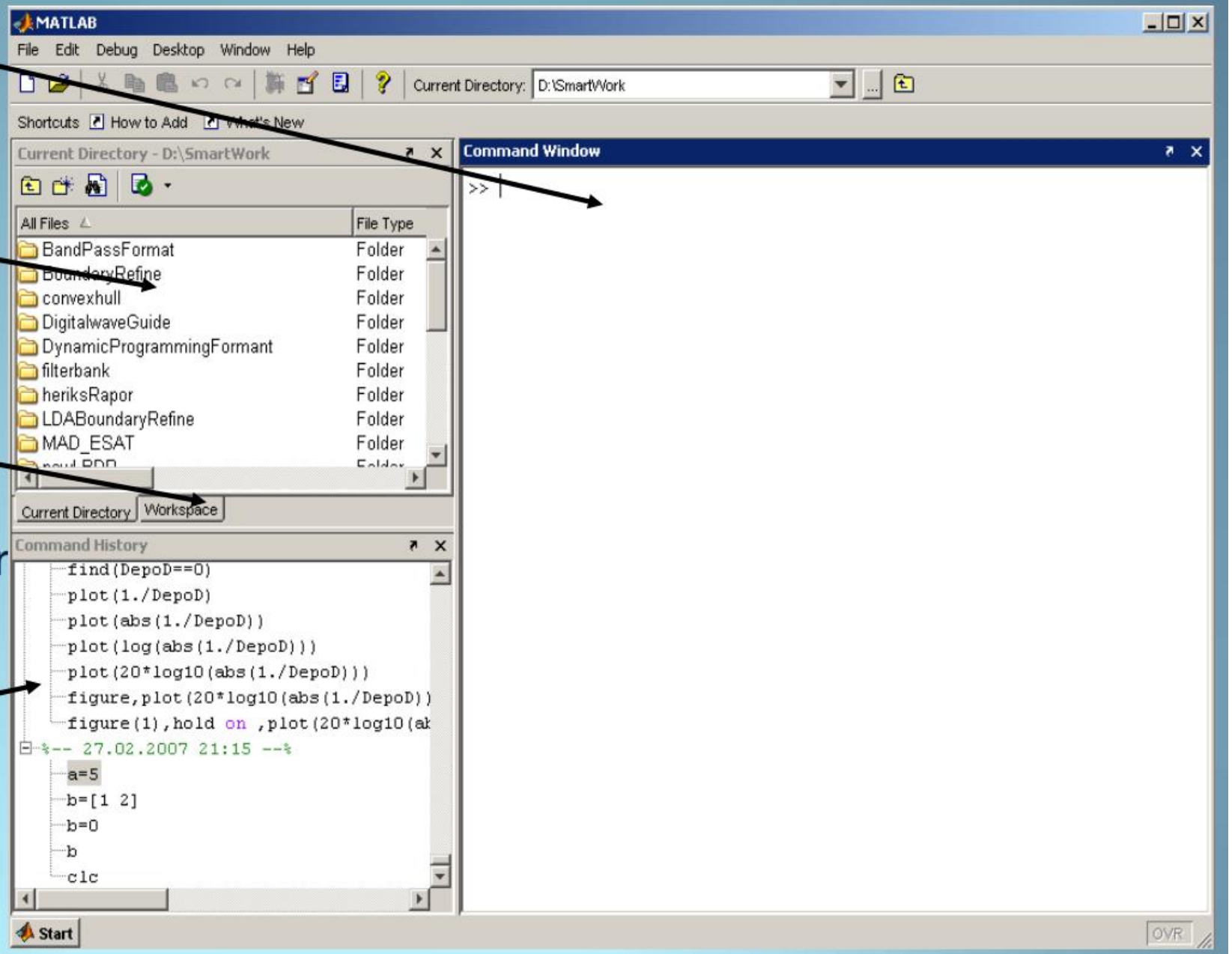
Workspace

- View program variables
- Double click on a variable
- to see it in the Array Editor

Command History

- view past commands
- save a whole session

using diary



MATLAB window

- Figure Window -contains output from graphic commands
- Help Window -provides help information
- Editor Window -creates and debugs script and function files
- Current directory Window -shows files in current directory
- Launch Pad Window -provides access to tools,demos and documentation

Command window

>> type code

Press enter

Command executed and output displayed

semicolon (;)

output not displayed

Ellipsis(…) if a command is too long to fit in one line

Command can continue line after line up to 4096 characters.

comman commands

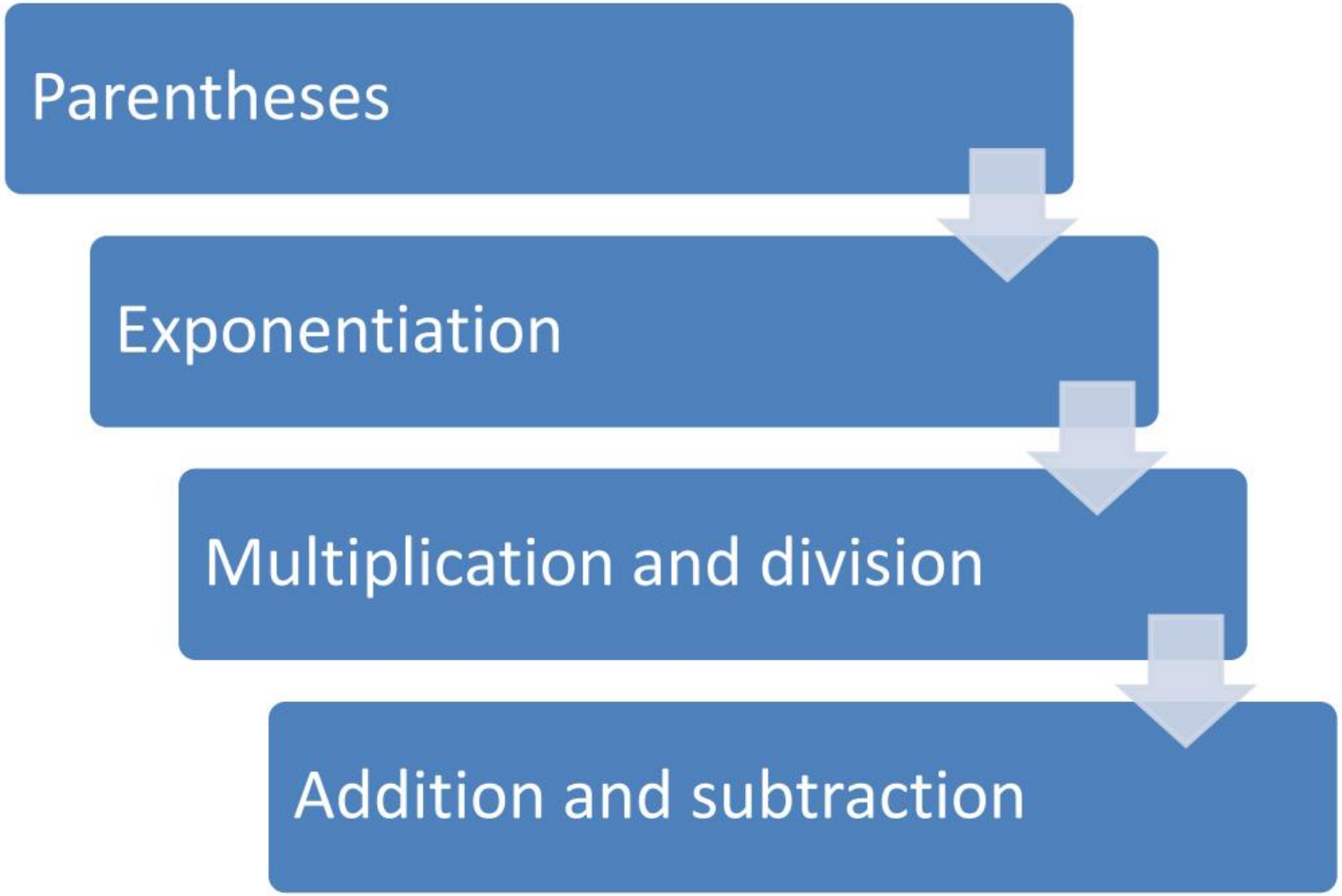
Matlab commands **case sensitive**

%	-comment
clc	-clear screen
↑	-recall previously typed commands
↓	-move down to previously typed

Arithmetic Operations With Scalars

Operation	Symbol	Example
Addition	+	5+3
Subtraction	-	5-3
Multiplication	*	5*3
Right division	/	5/3
Left division	\	5\3=3/5
Exponentiation	^	5^3=125

Order of Precedence



Display Formats

Command	Description
format short	Fixed-point with 4 decimal digits
format long	Fixed-point with 14 decimal digits
format bank	2 decimal digits
format compact	Eliminates empty lines
format loose	Adds empty lines

Elementary Math functions

Function	Description
sqrt (x)	Square root
exp (x)	Exponential (e^x)
abs (x)	Absolute value
log (x)	Natural logarithm Base e logarithm
Log10(x)	Base 10 logarithm
factorial(x)	Factorial function x!

Trigonometric/rounding Math functions

sin(x),cos(x),
tan(x),cot(x)

Rounding function

Function	Description
round(x)	Round to the nearest integer
fix(x)	Round towards zero
ceil(x)	Round towards infinity
floor(x)	Round towards minus infinity
rem(x,y)	Returns remainder after x is divided by y

Trigonometric/rounding Math functions

Elementary functions (sin, cos, sqrt, abs, exp, log10, round)

-type **help elfun**

Advanced functions (bessel, beta, gamma, erf)

-type **help specfun**

-type **help elmat**

Defining scalar variables

variable is a name made of a letter or a combination of several letters that is assigned a numerical value

- actually name of a memory location

-assignment operator “=”

eg. >>x=15

>>x=3*x-12

When new variable is created matlab assigns appropriate memory space where assigned value can be stored

- When variable is used stored data is used
- If assigned new value content of memory is replaced

eg. >>ABB=72;

>>ABB=9;

>>ABB

ABB=

9

Rules about variable names

- Variable names are case sensitive.
- Variable names can contain up to 63 characters (as of MATLAB 6.5 and newer).
- Variable names must start with a letter followed by letters, digits, and underscores.
- Must begin with a letter.
- Avoid using names of built-in functions for variable.