Introduction to Mathematical Software and Programming

Session 1

(w/c 17 February 2025)

Overview on MATLAB

❑ Introduction to MATLAB (Lab 1)

- Work in Command Window
- Built-in commands/functions
- Basic data structure: arrays (vector/matrix)
- Algebraic operations for arrays
- Array indexing

❑ 2-d Plot in MATLAB (Lab 2)

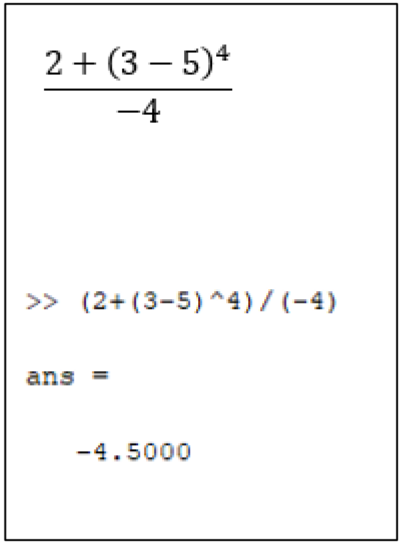
❑ Programming in MATLAB (Lab 3-5)

Use MATLAB as a calculator

Basic calculations can be executed at the prompt sign
>>
in the command window.

Precedence	Mathematical operations
( )	The contents of all parentheses are evaluated first, starting from the innermost parentheses and working outward.
^	All exponentials are evaluated, working from left to right
*, /	All multiplications and divisions are evaluated, working from left to right
+, -	All additions and subtractions are evaluated, starting from left to right

Example



$$\frac{2+(3-5)^4}{-4}$$

```
>> (2+(3-5)^4)/(-4)

ans =

   -4.5000
```

By default, MATLAB stores the calculation result in the variable **ans**

Precedence of operations

Highest	Parentheses ( )
	Power ^
Precedence ↓	Logical NOT ~
	Multiplication * Division /
	Remainder mod
	Addition + subtraction -
	Relational <, <=, >, >=, ==, ~=
	Logical AND &&
Lowest	Logical OR ||

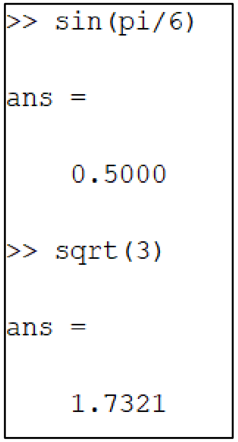
Note: selected from CELEN086 Seminar 1 slides for a complete table of precedence with arithmetic/relational/logical operators using MATLAB syntax.

Built-in Functions and Values

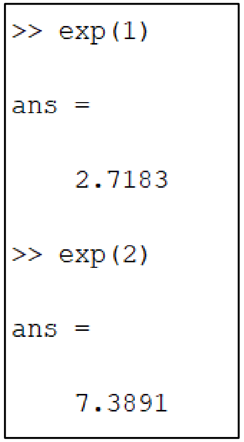
Elementary built-in functions

`cos(x)`	Cosine	`abs(x)`	Absolute value
`sin(x)`	Sine	`sign(x)`	Signum function
`tan(x)`	Tangent	`max(x)`	Maximum value
`acos(x)`	Arc cosine	`min(x)`	Minimum value
`asin(x)`	Arc sine	`ceil(x)`	Round towards $+\infty$
`atan(x)`	Arc tangent	`floor(x)`	Round towards $-\infty$
`exp(x)`	Exponential	`round(x)`	Round to nearest integer
`sqrt(x)`	Square root	`rem(x)`	Remainder after division
`log(x)`	Natural logarithm	`angle(x)`	Phase angle
`log10(x)`	Common logarithm	`conj(x)`	Complex conjugate

Examples



```
>> sin(pi/6)

ans =

    0.5000

>> sqrt(3)

ans =

    1.7321
```



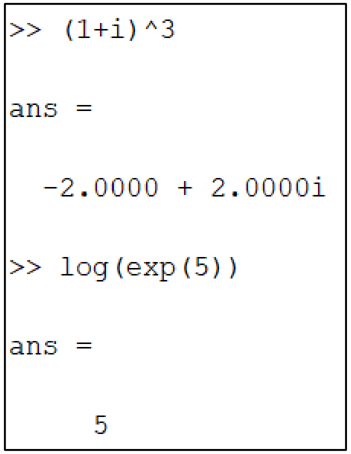
```
>> exp(1)

ans =

    2.7183

>> exp(2)

ans =

    7.3891
```

Predefined values

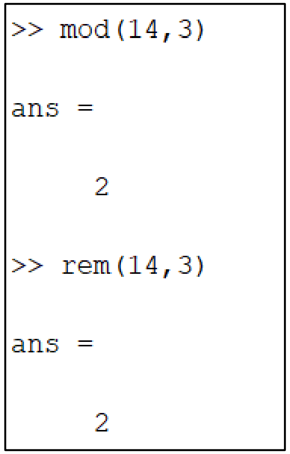
`pi`	The π number, $\pi = 3.14159\ldots$
`i,j`	The imaginary unit i, $\sqrt{-1}$
`Inf`	The infinity, ∞
`NaN`	Not a number

Note:

`exp(1)` *Euler's number e*



```
>> (1+i)^3

ans =

  -2.0000 + 2.0000i

>> log(exp(5))

ans =

     5
```



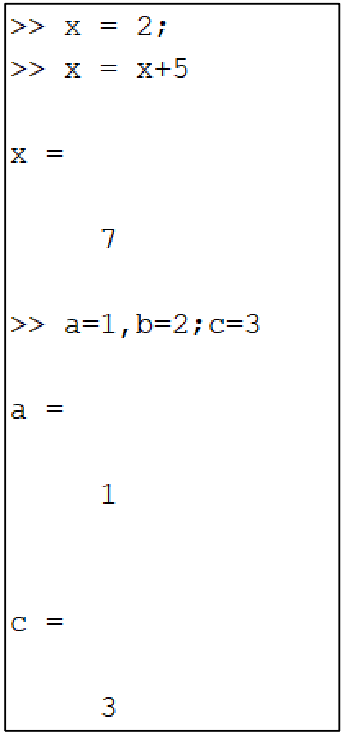
```
>> mod(14,3)

ans =

     2

>> rem(14,3)

ans =

     2
```

Defining variables

Variable like x cannot be called or used directly, unless we assign a value to it using the statement:

`variableName = value/expression`

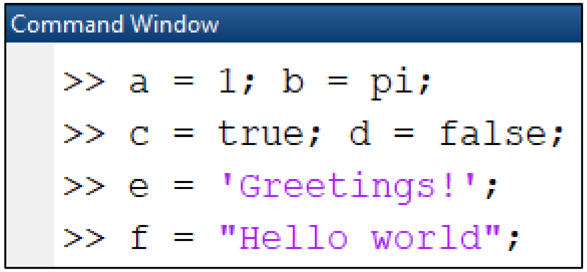
- Expressions may consist of multiple variables, operators and functions
- Created variables are stored in Workspace
- We can suppress the Command Window output by adding a semicolon **;** after the statement
- We can assign multiple variables at one command line, separating each by a comma **,** or a semicolon **;**



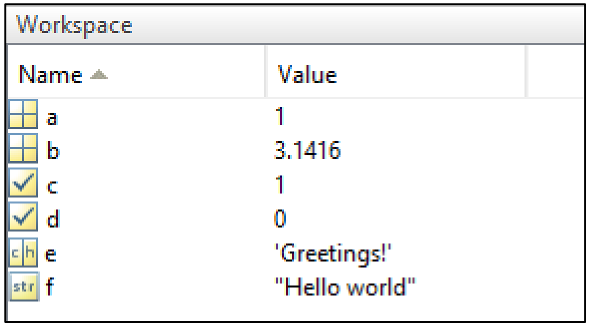
```
>> x = 2;
>> x = x+5

x =

     7

>> a=1,b=2;c=3

a =

     1


c =

     3
```

Data classes

- Numerical values in MATLAB are stored as "double" class by default
- Other commonly used data classes are logical, char, string...



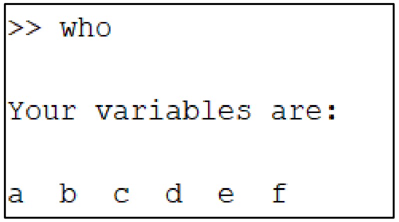
```
>> a = 1; b = pi;
>> c = true; d = false;
>> e = 'Greetings!';
>> f = "Hello world";
```



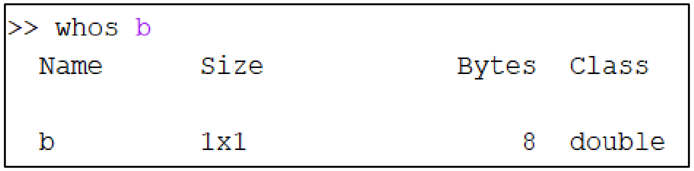
Workspace

Name	Value
a	1
b	3.1416
c	1
d	0
e	'Greetings!'
f	"Hello world"

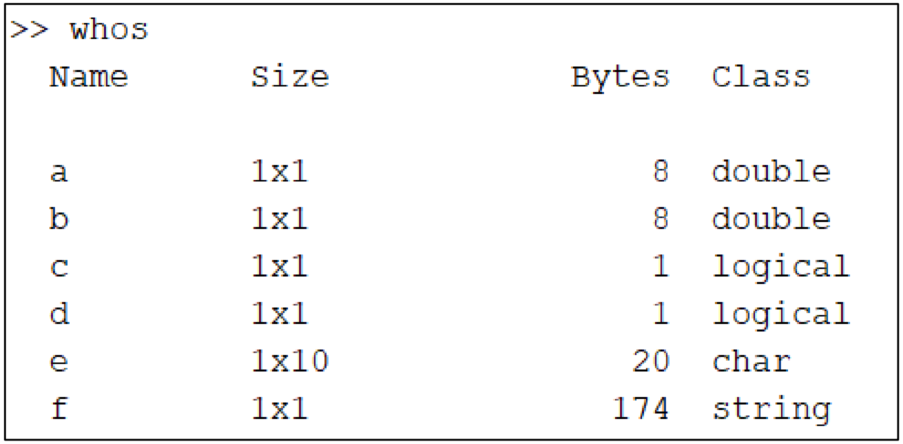
Following commands can display the variables and detailed information currently created in the Workspace.



```
>> who

Your variables are:

a  b  c  d  e  f
```



```
>> whos b
  Name      Size            Bytes  Class

  b         1x1                 8  double
```



```
>> whos
  Name      Size            Bytes  Class

  a         1x1                 8  double
  b         1x1                 8  double
  c         1x1                 1  logical
  d         1x1                 1  logical
  e         1x10               20  char
  f         1x1               174  string
```

Manage Workspace

All assigned variables are stored in Workspace.

- The data in Workspace persist between the executions of separate commands.
- The data in Workspace will be removed when you close MATLAB.

Following commands are helpful in managing data in Workspace:

`clear x`	Delete variable x from the Workspace
`clear`	Delete all variables in the current Workspace
`clc`	Clear the Command Window without deleting assigned variables

Array

Arrays are elementary data structures for storing values in MATLAB.

Array Dimension	Mathematical Terminology	Example
`0-dimension`	Scalar	$1, \pi, 2.71828$
`1-dimension`	Vector (row vector, column vector)	$\begin{pmatrix} 1 & 2 & 3 \end{pmatrix}$ $\begin{pmatrix} 10 \\ -5 \\ 8 \end{pmatrix}$
`2-dimension`	Matrix m-by-n or $m \times n$ matrix (m rows, n columns)	$\begin{pmatrix} 2 & 4 \\ 1 & 10 \end{pmatrix}$ $\begin{pmatrix} -9 & 0 \\ 2 & -1 \\ 5 & 1 \end{pmatrix}$

By convention, we can treat: scalar as 1×1 matrix;   and

row vector as $1 \times n$ matrix   column vector as $n \times 1$ matrix   depending on the length of vectors.

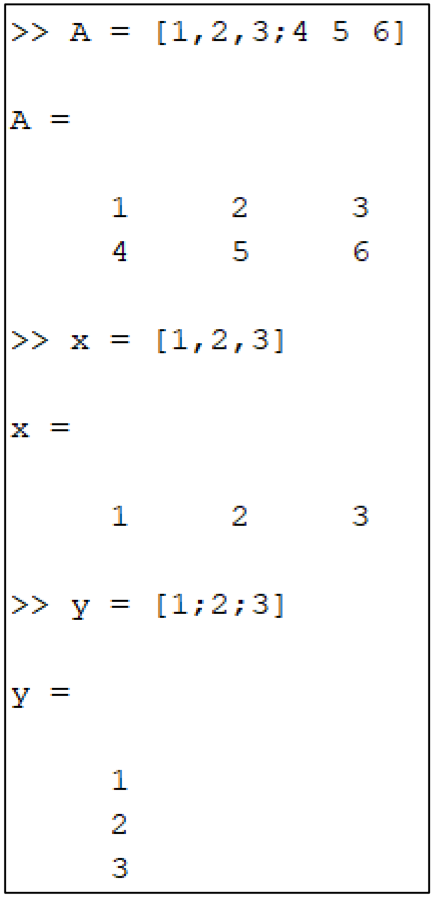
Matrix

Elements in matrix can be identified by the row index and column index.

$$A_{m\times n} = (a_{ij}) \qquad A_{2\times 3} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{pmatrix}$$

where a_{ij} are all elements located in the i-th row and j-th column with $1 \le i \le m, 1 \le j \le n$.

- In MATLAB, one matrix is entered row by row inside the square brackets [ ]
- Use comma ,(or space) to separate elements within a row, use semicolon ; to separate each row



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

A =

     1     2     3
     4     5     6

>> x = [1,2,3]

x =

     1     2     3

>> y = [1;2;3]

y =

     1
     2
     3
```

Built-in Matrix functions

Matlab provides functions that generate elementary or special matrices.

Function	Meaning
`eye(m,n)`	Return an m-by-n matrix with 1 on the main diagonal
`eye(n)`	Return an n-by-n square identity matrix
`zeros(m,n)`	Return an m-by-n matrix of zeros
`ones(m,n)`	Return an m-by-n matrix of ones
`rand(m,n)`	Return an m-by-n matrix of random numbers between 0 and 1
`magic(n)`	Use help command to see its meaning by yourself

Vector indexing and built-in command

In MATLAB, array index starts from 1.

`v(i)`	Return the i-th element in vector v
`length(v)`	Return the total number of elements in vector v
`sum(v)`	Return the sum of all elements in vector v
`max(v), min(v)`	Return the maximum/minimum element in vector v
`v’`	Return transposed vector

Evenly-spaced Vector

Two ways of creating evenly-spaced vector:

- Use colon operator **:** `StartValue:Increment:EndValue`
- Use linspace command: `linspace(StartValue,EndValue,PointNumber)`

Example: create a vector x consisting of numbers [-10,-9,-8,…,0,1,2,…,10]

```
-10:1:10
```

```
linspace(-10,10,21)
```

Note:

- *If we do not specify the increment in the colon operator, by default it is 1.*
- *If we do not specify the point number in the linspace command, by default it is 100.*

Matrix indexing and built-in command

Suppose M is a matrix with 5 rows and 6 columns,
and v is a row/column vector of length 10.

Command	Meaning
M (3 , 4)	The element with row index 3 and column index 4
M (2 , :)	Row 2 of M
M (: , 2)	Column 2 of M
M (2 : 4 , :)	Row 2 to Row 4 of M
v (2)	Second element of v
v (2 : end)	Second element to last element of v
M (2 , :) = [ ]	Delete row 2 from M

Command	Meaning
sum(sum(M))	Sum of all elements in matrix M
size(M)	Return the dimension of matrix M (row and column numbers)

Matrix operation

Suppose A and B are two matrices.

Command	Meaning	Comment
size(A)	Return the size of matrix A	(row number, column number)
A’	Transpose matrix of A	
inv(A)	Inverse matrix of A	A must be a square matrix
det(A)	Determinant of matrix A	
A^2	Matrix product A*A	
A+B	Sum of A and B	A and B must have same sizes
A-B	Difference of A and B	
A*B	Matrix product of A and B Note: (A*B is not equal to B*A)	The sizes of A and B must match, i.e., column number of A = row number of B

Element-wise operation

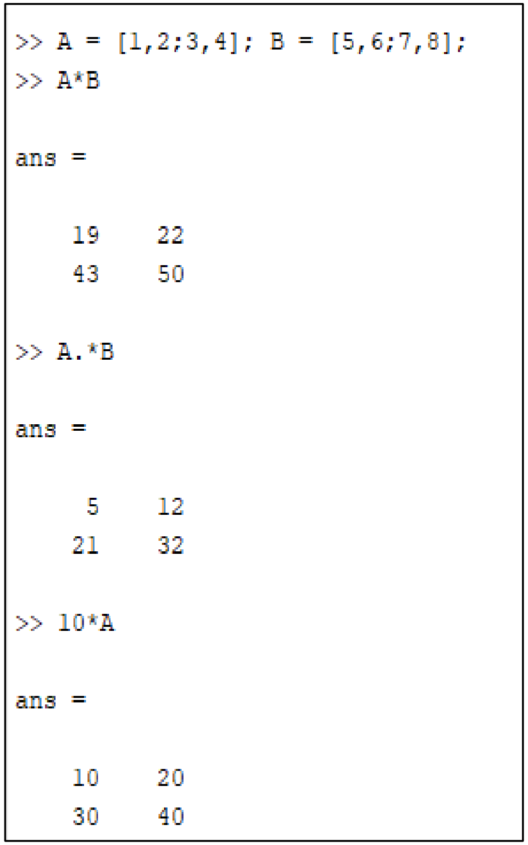
Note the difference between

- A*B matrix multiplication; Sizes of A and B must match
- A.*B element-by-element/element-wise multiplication; Sizes of A and B must be same

Operation sign	Meaning
.*	Element-wise multiplication
./	Element-wise division
.^	Element-wise exponentiation

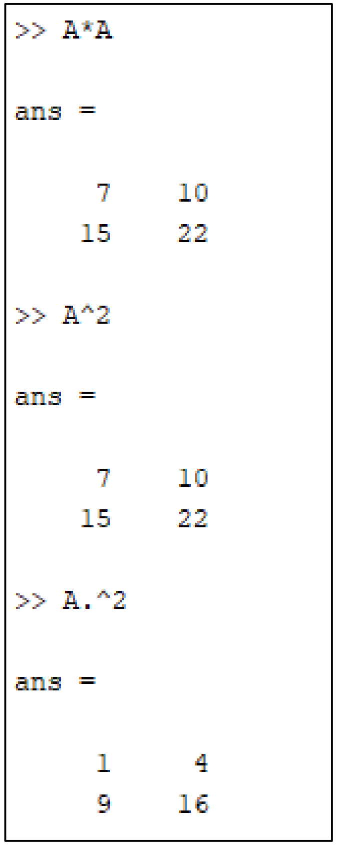
Example

Example 1



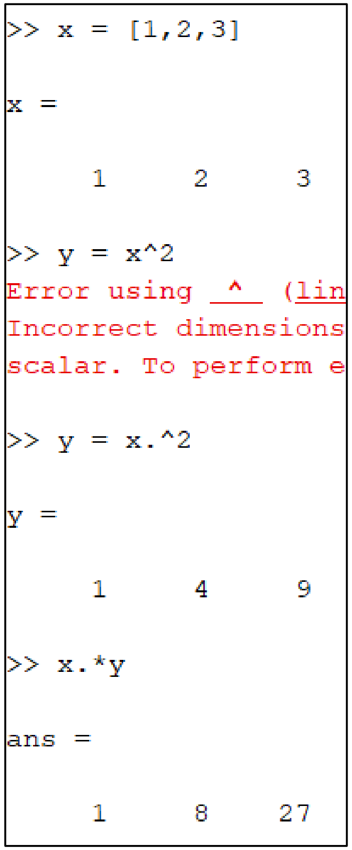
```
>> A = [1,2;3,4]; B = [5,6;7,8];
>> A*B

ans =

    19    22
    43    50

>> A.*B

ans =

     5    12
    21    32

>> 10*A

ans =

    10    20
    30    40
```

Example 2



```
>> A*A

ans =

     7    10
    15    22

>> A^2

ans =

     7    10
    15    22

>> A.^2

ans =

     1     4
     9    16
```

Example 3



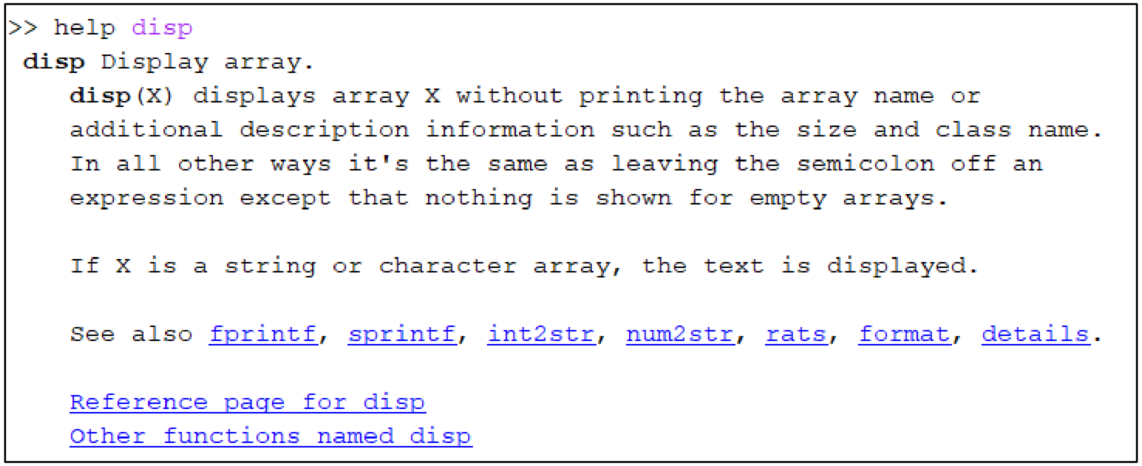
```
>> x = [1,2,3]

x =

     1     2     3

>> y = x^2
Error using  ^  (lin
Incorrect dimensions
scalar. To perform e

>> y = x.^2

y =

     1     4     9

>> x.*y

ans =

     1     8    27
```

*Note: If a is scalar, then a*A or A*a means multiplying each element of A by a.*

Help command and Documentation

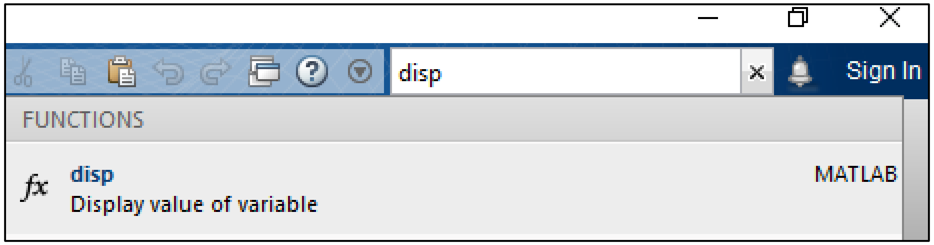
MATLAB built-in command **`help`** can be used for checking command information

`help CommandName`



```
>> help disp
 disp Display array.
    disp(X) displays array X without printing the array name or
    additional description information such as the size and class name.
    In all other ways it's the same as leaving the semicolon off an
    expression except that nothing is shown for empty arrays.

    If X is a string or character array, the text is displayed.

    See also fprintf, sprintf, int2str, num2str, rats, format, details.

    Reference page for disp
    Other functions named disp
```

You can also check from MATLAB documentation bar, which then direct you to the built-in documentation with examples

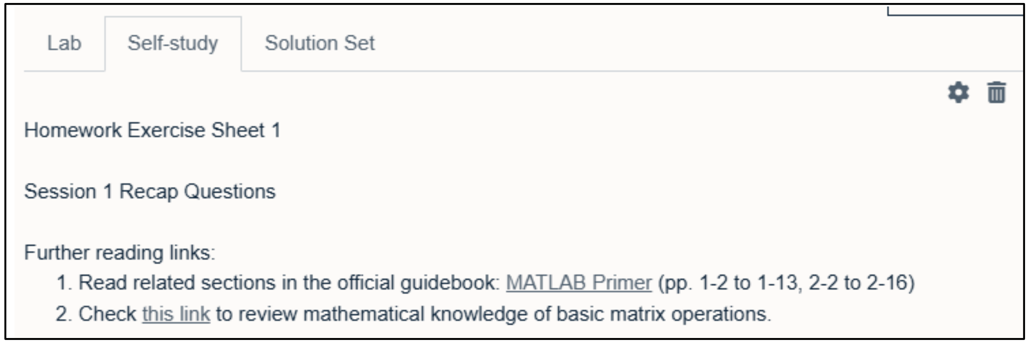


disp

fx disp
Display value of variable
MATLAB

Self-study

- Try all examples/commands in this lab note
- Complete lab worksheet 1
- Complete self-study materials on Moodle



Homework Exercise Sheet 1

Session 1 Recap Questions

Further reading links:

1. Read related sections in the official guidebook: MATLAB Primer (pp. 1-2 to 1-13, 2-2 to 2-16)
2. Check this link to review mathematical knowledge of basic matrix operations.

Please note:

You are expected to contribute 8 hours to your self-study for this 10-credit module in each week.

Extra note on Matrix

For a matrix $A_{m \times n}$, the transpose matrix $B_{n \times m} = A^T$ satisfies $b_{ij} = a_{ji}$.

For two matrices $A_{m \times p}$ and $B_{p \times n}$, the matrix multiplication AB produces a matrix $C_{m \times n}$, where the element in row i and column j in C is calculated through (the vector product of row i from A and column j from B)

$$c_{ij} = \sum_{k=1}^{p} a_{ik} \cdot b_{kj}$$

For a square matrix $A_{n \times n}$, the inverse matrix $B_{n \times n} = A^{-1}$ is the matrix such that $AB = BA = I$, where I is the identity matrix (or unit matrix) with ones on the main diagonal and zeros elsewhere.

For a square matrix $A_{n \times n}$, the determinant is a scalar value related to A, usually denoted by det(A).

Note:
You can review related topics about matrices from Foundation Algebra Textbook or check more online examples using the link provided in Moodle self-study resources.