MATLAB Premier

Asst. Prof. Dr. Melik DÖLEN

Middle East Technical University
Department of Mechanical Engineering
Outline

- Introduction
- Basic Features of MATLAB
  - Prompt Level and Basic Arithmetic Operations
  - Scalars, Vectors, and Matrices
  - Matrix Operations and MATLAB Functions
  - Graphics and Data Visualization
  - Programming in MATLAB
- MATLAB in Control Engineering
  - Operations on Transfer Functions
  - Simulation of LTI Systems
Introduction

- MATrix LABoratory “MATLAB” is an interactive software for numerical / symbolic computations and graphics.
- Originally designed for matrix computations, solving linear equation sets, data visualization, etc.
- The capabilities of MATLAB can be extended through programs written in its own programming language called “M-script”.
- Over the years, the capabilities of MATLAB in every aspect have widened rapidly.
- It has now become an “industry-standard” computing environment and an outstanding tool for engineering education.
To get started, type one of these: helpwin, helpdesk, or demo. For product information, type tour or visit www.mathworks.com.

» (45 + 7)/3
ans = 17.3333

» 100^2 – 3.5*ans
ans = 1.0061e+004

» sqrt(ans)
ans = 100.3029

ans (ANSwer) is a temporary “variable” storing the result of an instant calculation.

Previous answer can be recalled.

MATLAB offers a large number of predefined functions like sqrt, sin, cos, tan, exp, log, atan, abs, tanh, and more...

Arithmetic operations: +, -, *, /, ^

Scientific notation: 1.0061×10^4
MATLAB Objects

» a = 2.12
a =
2.1200

Result of various operations can be assigned to the variables you want to define. Variable names are case sensitive.

» b = -9.75;

Note that semicolon (;) suppresses the display.

» x = [1; 2; 3]
x =
1
2
3

Column vector (3x1) is defined:

\[ x = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \]

» y = [1 2 3]
y =
1 2 3

y is a row vector (1x3):

\[ y = [1 \ 2 \ 3] \]
MATLAB Objects (Cont’d)

» A = [1 2 3; 4 5 6; 7 8 9]
A =
1  2  3
4  5  6
7  8  9
A is a matrix (3x3):

» whos
Name      Size   Bytes  Class
A         3x3       72  double array
a         1x1        8  double array
ans       1x1        8  double array
b         1x1        8  double array
x         3x1       24  double array
y         1x3       24  double array

whos displays the names of all variables defined in the MATLAB “workspace” (memory).

Grand total is 18 elements using 144 bytes
Special Variables

By default, \( i \) and \( j \) are defined as imaginary unit \( (\sqrt{-1}) \).

MATLAB allows storage of complex numbers and relevant operations on them.

\( \cos(\pi) \)

\( a = \infty \) (see IEEE 754 standard!)
Important Notes

```matlab
» sin(x)
ans =
  0.8415
  0.9093
  0.1411
```

Most arithmetic operators and certain functions work for matrices and vectors as well.

```matlab
» help ans
ANS
ANS is the variable created automatically when expressions are not assigned to anything else. ANSWer.
```

An extensive online help is available.
Array Manipulations

```matlab
A = 12 345678922
    3

» A(2,3) Returns the element at 2nd row, 3rd column.
   ans = 6

» A(:,2) Returns all the elements at 2nd column.
   ans = 258

» A(3,:) Returns all the elements at 3rd row.
   ans = 789

» A(1:2,2:3) Returns submatrix.
   ans = 2356
```

Matrix A:

```
1 2 3
4 5 6
7 8 9
```
Array Manipulations (Cont’d)

» B = [A; zeros(1,3); y]
B =

\[
\begin{bmatrix}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9 \\
0 & 0 & 0 \\
1 & 2 & 3
\end{bmatrix}
\]

Forms a new matrix B using the variables previously defined. Here, the command `zeros(m,n)` creates an (m by n) matrix consisting of 0 elements.
Array Manipulations (Cont’d)

» B = [x ones(3,1) A]
B =
1     1     1     2     3
2     1     4     5     6
3     1     7     8     9

Redefines matrix B using the variables previously defined. Here, the command `ones(m, n)` creates an (m by n) matrix consisting of 1s.

B = 
[x]_{3x1} [1]_{3x1} [A]_{3x3}

B =
\begin{bmatrix}
1 & 1 & 1 \\
2 & 1 & 4 \\
3 & 1 & 7 \\
\end{bmatrix}
\begin{bmatrix}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9 \\
\end{bmatrix}

Vector x
Unity vector
Matrix A
Matrix Operations

Let us define these matrices.

\[
A = \begin{bmatrix}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9 
\end{bmatrix}; \\
B = \begin{bmatrix}
1 & 2 & 0 \\
4 & 0 & 6 \\
0 & 8 & 9 
\end{bmatrix}; \\
C = \begin{bmatrix}
1 & 2 \\
3 & 4 \\
5 & 6 
\end{bmatrix};
\]

Matrix inversion using (') operator.
The command `transpose(C)` serves for the same purpose.

\[
C' \\
\text{ans} = \\
\begin{bmatrix}
1 & 3 & 5 \\
2 & 4 & 6 
\end{bmatrix}
\]

DETerminant of matrix A.

\[
det(A) \\
\text{ans} = \\
0
\]

INVerse of matrix B.

\[
\text{inv}(B) \\
\text{ans} = \\
\begin{bmatrix}
0.4000 & 0.1500 & -0.1000 \\
0.3000 & -0.0750 & 0.0500 \\
-0.2667 & 0.0667 & 0.0667 
\end{bmatrix}
\]
Matrix Operations (Cont’d)

MATLAB enables you to perform arithmetic operations (+, -, *) on matrices.

Matrices must be conformable to carry out a certain operation.

Matrix multiplication.

Invalid operation due to a mismatch in matrix inner dimensions.
Matrix Operations (Cont’d)

Scalars (1×1) are exceptions in matrix operations.

Multiples every element of the matrix A by 3.

Multiplies the corresponding elements of matrix A and B. The operator (.* ) is called array multiplication.
Solving Equation Sets

Consider this equation set:
\[\begin{align*}
    x_1 + x_2 - 4x_3 &= 8 \\
    2x_1 - 3x_2 + 2x_3 &= 1 \\
    5x_1 - 8x_2 + 10x_3 &= 2
\end{align*}\]

Alternatively, we have
\[\begin{bmatrix}
    1 & 1 & -4 \\
    2 & -3 & 2 \\
    5 & -8 & 10
\end{bmatrix}
\begin{bmatrix}
    x_1 \\
    x_2 \\
    x_3
\end{bmatrix}
= \begin{bmatrix}
    8 \\
    1 \\
    2
\end{bmatrix}
\Rightarrow A \cdot x = b
\]

Therefore, the solution becomes
\[A^{-1}A \cdot x = A^{-1}b \Rightarrow x = A^{-1}b\]

To implement this in MATLAB, one defines the following:

```matlab
» A = [1 1 -4; 2 -3 2; 5 -8 10];
» b = [8; 1; 2];
» x = A\b
x =
   5.5000
   3.5000
   0.2500
```

The operator (\) is called left matrix divide and it essentially solves the linear system using Gauss-Seidel method. Note that \(x = \text{inv}(A) \ast b\) yields the same result.
**Advanced Matrix Functions**

» eig(A)  

Returns all the eigenvalues of matrix A.

ans =
  5.0000
  4.0000
-1.0000

» [V,D] = eig(A)

V =
  0.7071  0.8058  -0.4082
  0.0000  0.0620  -0.8165
  0.7071 -0.5889  -0.4082

D =
  5.0000  0  0
  0  4.0000  0
  0  0 -1.0000

- **lu** (command) computes LU factorization of a matrix.
- **chol** computes the cholesky factorization of a symmetric positive definite matrix.
- **qr** computes the QR factorization of a matrix.
- **svd** obtains singular value decomposition of a matrix.
- **norm** computes various matrix or vector norms.
- **cond**, **condest**, **rcond** estimates various condition numbers.
Vectors

» t = 1:5
   t =
      1   2   3   4   5
» t = 0:.1:1
   t =
   Columns 1 through 7
        0   0.1000   0.2000   0.3000   0.4000   0.5000   0.6000
   Columns 8 through 11
       0.7000   0.8000   0.9000   1.0000

Creates a row vector whose components increase \textit{arithmetically}.

Components can change by \textit{non-unit} steps.

» t = linspace(0,1,11)
   
   t =
   Columns 1 through 7
       0   0.1000   0.2000   0.3000   0.4000   0.5000   0.6000
   Columns 8 through 11
       0.7000   0.8000   0.9000   1.0000

Generates a vector with 11 equally spaced elements that lie between 0 and 1.
MATLAB offers a wide variety of built-in data visualization functions.

Function \texttt{plot} is used to generate 2D graphs.

To obtain a graph of $y=f(x)$, one needs to create the domain $x$ and then form the range $y$ with the corresponding function values.

```matlab
>> x = linspace(0,2*pi,100);
>> y = sin(x);
>> plot(x,y);
>> xlabel('x [rad]');
>> ylabel('y');
>> grid on;
```
As the second example, let us plot the following function:

\[ y = f(x) = \frac{x}{1 + x^2} \]

The corresponding MATLAB commands are as follows:

```matlab
» x = linspace(-5,5,100);
» y = x./(1+x.*x);
» plot(x,y);
» xlabel('x');
» ylabel('y');
» grid on;
```
Programming

- Entering all the commands at the MATLAB prompt sequentially is a tedious and slow process.
- One can type in all the commands in a text file having an extension of “.m”. The file containing these commands is called M-script.
- When the name of this “.m” file is keyed in just like a command at the prompt, MATLAB executes all the commands sequentially.
- Similar to a high-level language, MATLAB provides a number of standard programming constructs such as loops and conditionals.
**Conditionals**

**IF Statement**

if <boolean expression 1>
    MATLAB statements
elseif <boolean expression 2>
    MATLAB statements
else
    MATLAB statements
end

**Example**

if x == 0
    y = 0;
elseif x == 1
    y = 1;
else
    y = 1 - 2*(x - .25);
end

- The use of **conditionals** in MATLAB programs are very similar to those of common high-level languages.
- The logical operators in MATLAB are <, >, <=, >=, == (logical equal), ~= (not equal).
- **Boolean expressions** take the values 1 (true) or 0 (false).
For Loop

FOR Statement
for variable = expression
    MATLAB statements
end

Example 1
for k = 1:4
    disp(k);
end

    1
    2
    3
    4

Example 2
for k = [1 3 7 11]
    disp(k);
end

    1
    3
    7
    11

- **for** loop repeats the statements as the loop index takes on the values in a given vector.
- Like an **if** construct, the loop must be terminated by **end** statement.
- **disp** command simply displays its argument without showing the variable name.
While Loop

**WHILE Statement**
while <boolean expression>
    MATLAB statements
end

**Example**
```
x = 1;
while 1+x > 1
    x = x/2;
end
disp(2*x)
```

**Output**
```
2.2204e-016
```

- **while** loop repeats the statements as long as the given boolean expression is true.
- Another important command is **break** which terminates the current **for** or **while** loop.
- The sample program is used to test the floating point accuracy ("machine epsilon") of a particular computer system.
MATLAB Functions

- While programming, it is often times necessary to define one’s own subroutines (procedures/functions).
- Unlike high-level languages, these functions cannot be included in the main program.
- Each function has to be an individual “m” file.
- This “m” file that begins with a line of the following form: `function [out1,out2,...] = cmd_name(in1,in2,...)`
- When a function is invoked, MATLAB creates a temporary workspace. The statements inside the function have no access to the variables used in the main workspace unless they are passed as inputs.
- When execution ends, all local variables are erased.
MATLAB Functions (Cont’d)

function [y,z] = fcn(x)
    z = 1 + x.*x;
    y = x./z;
    plot(x,y)

» x = linspace(-5,5,100);
» fcn(x)
   Calls M-file named “fcn.m” and passes the vector x.

» y1 = fcn(x);
» [y2,z2] = fcn(x);
   Different output arguments were passed from the MATLAB function fcn.
Useful MATLAB Functions

**Poles of a System**

\[ D(s) = s^5 + 3s^4 + 7s^3 + 5s^2 + 2s + 1 \]

```matlab
>> den = [1 3 7 5 2 1];
>> roots(den)
ans =
-1.0777 + 1.9230i
-1.0777 - 1.9230i
-0.7742
-0.0352 + 0.5144i
-0.0352 - 0.5144i
```

**Characteristic Polynomial**

Let the poles of a system be
\[ p_1 = -3; \quad p_2 = -2; \]
\[ p_3 = -1; \quad p_{3,4} = -5 \pm 5i; \]

```matlab
>> poles = [-3 -2 -1 -5+5*i -5-5*i];
>> poly(poles)
ans =
1 16 121 416 610 300
```

The resulting characteristic polynomial is

\[ D(s) = s^5 + 16s^4 + 121s^3 + 416s^2 + 610s + 300 \]
**Useful Commands (Cont’d)**

**Partial Fractions Expansion**

Consider the following TF:

\[
G(s) = \frac{2s^3 + 5s^2 + 3s + 6}{s^3 + 6s^2 + 11s + 6}
\]

The resulting fractions are

\[
G(s) = \sum_{i=1}^{N} \frac{r_i}{s - p_i} + k
\]

\[
G(s) = \frac{-6}{s+3} + \frac{-4}{s+2} + \frac{3}{s+1} + 2
\]

» num = [2 5 3 6];
» den = [1 6 11 6];
» [r,p,k] = residue(num,den)

\[
r = \\
-6.0000 \\
-4.0000 \\
3.0000
\]

\[
p = \\
-3.0000 \\
-2.0000 \\
-1.0000
\]

k = 2
Useful Commands (Cont’d)

**Polynomial Multiplication**

\[
P(s) = s^3 + 2s^2 + 3s + 1 \quad Q(s) = s^2 + s + 5 \implies D(s) = P(s)Q(s) = ?
\]

```matlab
» P = [1 2 3 1];
» Q = [1 1 5];
» D = conv(P,Q);
```

\[D = 1 \quad 3 \quad 10 \quad 14 \quad 16 \quad 5\]

The resulting polynomial becomes

\[D(s) = s^5 + 3s^4 + 10s^3 + 14s^2 + 16s + 5\]

**Zero/Pole Locations**

```matlab
» num = [2 5 3 6];
» den = [1 6 11 6];
» pzmap(num,den)
```

![Pole-Zero Map](image)
Block Diagram Reduction

Transfer Functions in Series

\[
\frac{Y(s)}{X(s)} = G(s) = G_1(s)G_2(s)
\]

As an illustration, let

\[
G_1(s) = \frac{s + 5}{s^2 + 3s + 5}; \quad G_2(s) = \frac{1}{s + 1}
\]

The resulting TF becomes

\[
G(s) = \frac{s + 5}{s^3 + 4s^2 + 8s + 5}
\]

» num1 = [1 5]; den1 = [1 3 5];
» num2 = [1]; den2 = [1 1];
» [num, den] = series(num1, den1, num2, den2)

num =
\[
0 \quad 0 \quad 1 \quad 5
\]

den =
\[
1 \quad 4 \quad 8 \quad 5
\]

NUMerator
DENominator
Transfer Functions in Parallel

As an illustration, let

\[
\frac{G_1(s)}{X(s)} = \frac{G_1(s)}{S} = \frac{1}{s+1}
\]

\[
G_2(s) = \frac{1}{s+1}
\]

The resulting TF becomes

\[
G(s) = \frac{2s^2 + 9s + 10}{s^3 + 4s^2 + 8s + 5}
\]
Closed Loop Transfer Functions

\[
\begin{align*}
X(s) & \quad + \quad G_1(s) \quad Y(s) \\
- & \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad G_2(s)
\end{align*}
\]

As an illustration, let

\[
G_1(s) = \frac{s + 5}{s^2 + 3s + 5}; \quad G_2(s) = \frac{1}{s + 1}
\]

The resulting TF becomes

\[
G(s) = \frac{s^2 + 6s + 5}{s^3 + 4s^2 + 9s + 10}
\]

\[
\begin{align*}
\text{num1} &= [1 5]; \quad \text{den1} = [1 3 5]; \\
\text{num2} &= [1]; \quad \text{den2} = [1 1]; \\
\text{num,den} &= \text{feedback}(\text{num1,den1,\text{num2,den2}})
\end{align*}
\]

\[
\begin{align*}
\text{num} &= \\
&= [0, 1, 6, 5] \\
\text{den} &= \\
&= [1, 4, 9, 10]
\end{align*}
\]
As an illustration, let

\[ G(s) = \frac{2s + 4}{s^2 + 2s + 4} \]

MATLAB commands are

```matlab
» num = [2 4]; den = [1 2 4];
» impulse(num,den,10)
```

Duration of simulation
(10 sec.)
As an illustration, let

\[ G(s) = \frac{2s + 4}{s^2 + 2s + 4} \]

MATLAB commands are

```matlab
» num = [2 4]; den = [1 2 4];
» step(num,den,10)
```

Duration of simulation
(10 sec.)