Supplemental Document 1

A General Tutorial on MATLAB

Interactive MATLAB Tutorial available in the following website
http://www.mathworks.com/academia/student_center/tutorials/register.html

1. The best teacher you can ask for help: the help function

help XYZ = help on function "XYZ"

Example: get the roots of $s^2 + 2s + 1 = 0$
function: roots

help roots
Explanation listed as follows
Polynomial: $C_1 s^N + C_2 s^{N-1} + \cdots + C_N s + C_{N+1}$
MATLAB: SYS $= [C_1 \ C_2 \ \ldots \ C_N \ C_{N+1}]$

Sys = [1 2 1]; %(don't show the result in the command line – the role of semicolon)
Sys = [1 2 1] %(show the result in the command line)
roots(Sys)
The answer is -1, -1

2. Algebra operation

- MATLAB can be used as a scientific calculator
- Scalar
  Example: 1, and -2
- Vector: a one-dimensional array of multiple elements
  Example: *v* = [1, 2, 3, 4, 5], or [1 2 3 4 5], or [1:1:5], or [1:5]
  The transpose of *v*: v^T = [1, 2, 3, 4, 5] ', or [1 2 3 4 5] ', or [1:1:5], or [1:5] '
  ones(3,1)
  ones(1,3)
  zeros(3,1)
  zeros(1,3)
- Matrix: a rectangular array of numbers, symbols or expressions
  M = [1, 2, 3; 4, 5, 6; 7, 8, 9], or *M* = [1 2 3; 4 5 6; 7 8 9], or
  M(1,1:3) = [1, 2, 3], M(2,1:3)= [4, 5, 6], M(3,1:3) = [7, 8, 9]

  The transpose of *M*: $(M^T)_{i,j} = (M)_{j,i}$
  M^T = [1, 2, 3; 4, 5, 6; 7, 8, 9]'
  ones(3,4)
  ones(4,3)
  eye(3)
  zeros(4,4)
  zeros(5,4)

2.1 Unary operation:
2.1.1 -: unary minus

Example: we can get the opposite of 2 by typing
-2

2.1.2 ^: power
Example: 20 to the power of twenty 20, i.e., 20^{20}
20^20
The answer is 1.0486e+026. MATLAB, however, uses "e" to represent "10^" for scientific notation

2.1.3 exp(*x*): the exponential of *x*
exp(1)
The answer is 2.7183.

2.1.4 log(*x*) for natural logarithm, and log10 for common (base 10) logarithm
Example
log(2.7183), log10(10^5)
The answers are 1 and 5, respectively

Note: all the above operations are applicable to vectors and matrixes. In this case, the operation is applied to each element in the vectors or matrixes.

2.2 Double element operation
2.2.1 +: plus
Example
v1 = [1:2:5];
v2 = [2:2:6];
v1 + v2
The answer is [3 7 11];

2.2.2 -: minus
Example
v1 = [1:2:5];
v2 = [2:2:6];
v1 - v2
The answer is [-1 -1 -1];

2.2.3 *: Matrix multiply, i.e. *M***N* with $M \in R^{m \times n}$, and $N \in R^{n \times m}$,
Example

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

M = [1 2; 2 1; 3 2];
N = [4 6 8; 5 7 9];
*M*N*
The answer is:

$$\begin{bmatrix} 14 & 20 & 26 \\ 13 & 19 & 25 \\ 22 & 32 & 42 \end{bmatrix}$$

2.2.4 ^: Square matrix power, i.e. *M*^2

Example

$$\begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}^{\wedge 2}$$

M = [1 2; 2 1];

M^2

The answer is:

$$\begin{bmatrix} 5 & 4 \\ 4 & 5 \end{bmatrix}$$

2.2.5 .*: Array multiply

v = v_1.*v_2, *v*(i) = $v_1(i)$*$v_2(i)$

Example

$$v_1 = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}, v_1 = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}, v_1 \times v_2 = ?$$

v1 = [1 2 3]';

v2 = [-1 1 -1]';

*v1.*v2*

The answer is:

$$\begin{bmatrix} -1 \\ 2 \\ -3 \end{bmatrix}$$

2.2.6 .^: Array power

v =v_1^n, *v*(i) = $v_1(i)\times v_1(i) \times \cdots \times v_1(i)$ (totally *n* times)

$$v_1 = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}, v_1^3 = ?$$

v1 = [1 2 3]';

v1.^3

The answer is:

$$\begin{bmatrix} 1 \\ 8 \\ 27 \end{bmatrix}$$

2.2.7 inv: the inverse of the square matrix X

A square matrix is singular if and only if its determinant is 0.

The inv is applied to any nonsingular or invertible matrix.

Example

$$M = \begin{bmatrix} 2 & 0 \\ 1 & 1 \end{bmatrix}, M^{-1} = ?$$

M = [2 0; 1 1];

inv(M)
eye(2)/M
The answer is

$$M^{-1} = \begin{bmatrix} 0.5 & 0 \\ -0.5 & 1 \end{bmatrix}$$

2.2.8 ./ : Right array divide
$v = v_1./v_2$, $v(\mathrm{i}) = v_1(i)/v_2(i)$
Example

$$v_1 = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}, v_1 = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}, v_1 \times v_2 = ?$$

v1 = [1 2 3]';
v2 = [-1 1 -1]';
v1./v2
The answer is:

$$\begin{bmatrix} -1 \\ 2 \\ -3 \end{bmatrix}$$

2.2.9 the size of a vector or matrix
The length of a vector, i.e., the number of elements in the vector: length(v)
Example
v = [0 1 3];
length(v)
The size of a matrix, i.e., the number of columns and rows in the matrix: size(M)
Example
What is the size of matrix $M = \begin{bmatrix} 2 & 3 \\ 3 & 4 \\ 4 & 5 \end{bmatrix}$,

M = [2 3; 3 4; 4 5];
size(M)
The answer is: 3, 2, meaning three rows and two columns.

3. Automating analysis with functions and scripts

3.1 saving all the commands into a *.m document
The procedure for building a *.m document

- Run MATLAB
- File -> New -> Script
- Input the commands into the file
- Save the file as a Document_name.m document
- Type Document_name in the command line to run the program

Example: to calculate $\sum_{i=1}^{10}(2i-1)^3$

- Input the following the commands into a document

```
clc % Clear command window.
clear % Removes all variables from the workspace.
v = [1:2:19];
v_3 = v.^3; % name criteria for MATLAB
sum_v_3 = sum(v_3)
```

- save the document as example_sum.m
- run example_sum in the command line

Tips: 1) add % for comments
2) use Ctrl + r to change an executable line to a comment line
2) use Ctrl + t to change the comment line to an executable line

3.2 writing a function with specified inputs

The procedure for building a function document

- Run MATLAB
- File -> New -> Function. And the following template pops up.

```
function [ output_args ] = Untitled3( input_args )
%UNTITLED3 Summary of this function goes here
%   Detailed explanation goes here

End
```

- Assign the name, i.e., `Untitled3`, for the function. The name of the function should be the same as the name of the document
- Assign the inputs (i.e., `input_args`) and outputs (i.e., `output_args`) of the function. It is possible that there are multiple inputs and multiple outputs in the defined function
- To run the function, assign the inputs and type the function name in the command line

Example: to calculate $\sum_{i=1}^{10}(2i-1)^3$

- Define a function to calculate $(2i\text{-}1)^3$

```
function y = cubic_i(x)
%UNTITLED3 Summary of this function goes here
%   Detailed explanation goes here
y = (2*x -1)^3;
end
```

- On the basis of the function *cubic_i*, calculate the output of this function for each input i, and then sum up all the outputs from the function

```
clc
clear
y_1 = cubic_i (1);
y_2 = cubic_i (2);
y_3 = cubic_i (3);
y_4 = cubic_i (4);
y_5 = cubic_i (5);
y_6 = cubic_i (6);
y_7 = cubic_i (7);
```

```
y_8 = cubic_i (8);
y_9 = cubic_i (9);
y_10 = cubic_i (10);
y_1 + y_2 + y_3 + y_4 + y_5 + y_6 + y_7 + y_8 + y_9 + y_10
```

Any better way? Repeated operations => circulation

4. Flow Control and Logical Operation

4.1 Flow control: we mainly focus on for loops and while loops in this training

4.1.1 for loops

for <variable> = <expression>
    <statement>
    ...
    <statement>
end

Example: to calculate $\sum_{i=1}^{10}(2i-1)^3$

```
Sum_i = 0;
for i = 1:10
    Sum_i = Sum_i + cubic_i(i);
end
Sum_i
```

Note: 1) pay attention to the indentation for different loops
2) loop operation is slower than the vector operation

4.1.2 while loops

while <logical expression>
    <statement>
    ...
    <statement>
end

Example: to calculate $\sum_{i=1}^{10}(2i-1)^3$

```
Sum_i = 0;
i = 1;
while i <= 10
    Sum_i = Sum_i + cubic_i(i);
    i = i + 1;
end
Sum_i
```

4.2 Logic operation

4.2.1 Relational operators.

- == - Equal
  Example:
  1==2
  The answer is false (i.e., 0)

1==1
The answer is true (i.e., 1)

- ~= - Not equal
- < - Less than
- > - Greater than
- <= - Less than or equal
- >= - Greater than or equal
- strcmp(a, b) Compares strings a and b.
  Example:
  strcmp('a','a')
  strcmp('a','b')

4.2.2 Logical operators

- & - Element-wise logical AND
  Example:
  1 & 0 = 0
  1 & 1 = 1
- | - Element-wise logical OR
  Example:
  1 | 0 = 1
  0 & 0 = 0
  1 | 1 = 1
- ~ - Logical NOT
  Example:
  ~ 0 = 1
  ~ 1 = 0

4.2. 3 if … end

if <logical expression>
  <statement>
  ...
  <statement>
end

Example: determine the maximum element in a vector, v = [2 3 1 5 8 9 2];

```
clc
clear
v = [2 3 1 5 8 9 2];
max_element = v(1);
for i = 2:length(v)
    if v(i) > max_element
        max_element = v(i);
    end
end
max_element
```

4.2.4 if … else … end (there are only two possible scenarios)

if <logical expression>
  <statement group 1>
else
  <statement group 2>
end

4.2.5 if … elseif … elseif … else … end

```
if <logical expression 1>
   <statement group 1>
elseif <logical expression 2>
   <statement group 2>
elseif <logical expression 3>
   <statement group 3>
   ...
elseif <logical expression r>
   <statement group r>
else
   <statement group r+1>
end
```

Example: determine how many '2', '3', and '1' in a vector, v = [2 3 1 5 8 9 2];

```
clc
clear
v = [2 3 1 5 8 9 2];
number_1 = 0;
number_2 = 0;
number_3 = 0;
number_others = 0;
for i = 1:length(v)
    if v(i) == 1
        number_1 = number_1 + 1;
    elseif v(i) == 2
        number_2 = number_2 + 1;
    elseif v(i) == 3
        number_3 = number_3 + 1;
    else
        number_others = number_others + 1;
    end
end
disp('# of 1')
number_1
disp('# of 2')
number_2
disp('# of 3')
number_3
```

5. Searching and Sorting

5.1 Searching

- This is about searching an element in an array.
- One example with using for loops and comparison operation is given in section 4.2.5
- Another way to conduct searching is to use the function *find*

Example: determine how many '2', '3', and '1' in a vector, v = [2 3 1 5 8 9 2];

```
clc
clear
v = [2 3 1 5 8 9 2];
number_1 = 0;
```

```
number_2 = 0;
number_3 = 0;
Index1 = find(v == 1);
number_1 = length(Index1)
Index2 = find(v == 2);
number_2 = length(Index2)
Index3 = find(v == 3);
number_3 = length(Index3)
disp('# of 1')
number_1
disp('# of 2')
number_2
disp('# of 3')
number_3
```

5.2 Sorting

- This is about sorting the elements in an array in an increasing or decreasing order
- Determining the maximum element in an array like the example given in Section 4.2.3 is one special case of sorting
- Bubble sort algorithm: repeatedly step through the list to be sorted, compare each pair of adjacent items and swap them if they are in the wrong order.
- Functions to get the maximum and minimum elements in one dimensional array: max() and min()
- The MATLAB function: arrange the data in the form of an array, and use the function *sortrows*

Example: rank the students' scores in Process Control final in an increasing order. The first column of the following matrix shows the students' ID, and the second column shows the students' score

[1 75
2 60
3 90
4 85]

```
clc
clear
M = [1 75
     2  60
     3  90
     4  85];
score_sort = sortrows(M,2) % the positive '2' for ascending order;
'-2' for  descending order
disp('the highest score')
score_sort(size(M,1),:)
disp('the lowest score')
score_sort(1,:)
```

The result is shown as:

```
score_sort =
```

```
2     60
1     75
4     85
3     90
```