

LAB WORK GUIDE

COMPUTER PROGRAMMING



By :  
Supardi, M.Si

PHYSICS EDUCATION DEPARTMENT  
FACULTY OF MATHEMATIC AND NATURAL SCIENCES  
YOGYAKARTA STATE UNIVERSITY  
2010

# Table of Contents

<b>UNIT 1</b> .....	<b>2</b>
Matlab Essential 1.....	2
<b>UNIT 2</b> .....	<b>8</b>
Matlab Essential 2.....	8
<b>UNIT 3</b> .....	<b>16</b>
Matlab Essential 3.....	16
<b>UNIT 4</b> .....	<b>28</b>
MATRIX.....	28
<b>UNIT 5</b> .....	<b>42</b>
BRANCHING STATEMENTS.....	42
<b>UNIT 7</b> .....	<b>55</b>
THE WHILE LOOP.....	55
<b>UNIT 8</b> .....	<b>60</b>
The Break and Continue Statements.....	60

# UNIT 1

## Matlab Essential 1

### Learning Objective

The objectives of this unit are to introduce some of fundamentals of Matlab programming, including

- 1) Operators, variables and expression
- 2) How to create a variable
- 3) Assignment operator

### Concepts

### Basic Operator

Basic arithmetic operators include addition (+), subtraction (-), multiplication (\*), division (/) and power (^). For example,

```
>> a=10
a =
    10
>> a^2
ans =
    100
>> a^3
ans =
   1000
>> 1+2*4/3
ans =
    3.6667
>> 1+2/4*3
ans =
    2.5000
```

Notice the example number 4 and 5. Here, we present the complex operation involving +, \* and /. We can write the operation more clearly of  $1+2*4/3$  with sign ( ) as

$$\begin{aligned}
 1 + ((2 * 4) / 3) &= 1 + 8 / 3 \\
 &= 1 + 2.667 \\
 &= 3.667
 \end{aligned}$$

For the operation of  $1 + 2 / 4 * 3$ , we can write again

$$\begin{aligned}
 1 + 2 / 4 * 3 &= 1 + (2 / 4) * 3 \\
 &= 1 + 0.5 * 3 \\
 &= 2.5000
 \end{aligned}$$

Matlab has some rules in executing the mathematical operation

- Matlab gives priority to the mathematical operation in the brackets.
- The mathematical operation involving the operator \* and / (could be \* / or / \*) work from left to right.
- The mathematical operation involving the operator + and - (could be + - or - +) also work from left to right.

### ***Assignment Operator***

The sign “=” is called as assignment operator. There are two forms of mathematical equation as below

$$x + 4 = 7 \quad \text{and} \quad x = 7 - 4$$

When we have an equation as point 1) and write it into Matlab statement, Matlab will show the error comment as below

```
>> x+4=7;
??? x+4=7;
|
Error: Missing operator, comma, or semicolon.
```



Whereas, x can express the value if it is given a command to calculate a specific operation.

```
>> x=7-4
x =
3
```

### ***Creating Variable***

Matlab does not need any variable declarations or dimension when we want to use the variable. In Matlab, variable will be created automatically and saved whenever finding the

new variable. Beside that, the very important thing is that Matlab is **case sensitive**. It means that upper case letter and lower case letter are differentiated. Followings are some rules when we will create a variable.

1. Variable could not be initiated by number, for example 2abc, 45y, 3ok43
2. Variable can consist of letter and number as ok45, ok45ok, abc432.
3. Variable could not use any special character as well as %, #, -, + or =. For example %ok, net-cost, %x, @sign dsb.
4. Avoid to create variable using special names like pi, eps, i, j. Because, the ones have been used by Matlab to assign any special value. As you know that pi=22/7, eps= , i and j assign the value  .
5. It is recommended that the variable we have created is simple but meaningful.

### Activity

1. State the problem

Calculate the value of z from the equation as shown below

$$z = \frac{a - \frac{b}{c-d}}{a + \frac{b}{c+d}}$$

2. Define the inputs and outputs

The inputs of this problem are any values of  $a$ ,  $b$ ,  $c$  and  $d$ . The values of input may not be imaginary. While, the output of this problem is  $z$  after applying any arithmetic operation of  $a, b, c$  and  $d$ .

2. Describe the algorithm

This problem can be broken down into three steps:

- 1) Get the values of inputs  $a, b, c,$  and  $d$
- 2) Apply the basic arithmetic operation to get  $z$ .
- 3) Write out the output (e.g the  $z$ )

### PROBLEMS

1. Determine which of the followings are valid variable. If invalid, explain why

a) b32

b) 2d

- c) s34d
- d) speed\_bicycle
- e) \_speed
- f) %velocity
- g) velocity&
- h) speed bicycle
- i) 'a'nu
- j) pi
- k) realmax
- l) a^3

2. Determine which of the following numbers are not accepted by Matlab.

- a) 2,34
- b) 2.32
- c) 0.32
- d) -3214
- e) 2.3e-4
- f) 5.2e+2
- g) 5e^3
- h) 3.43e5.3
- i) 34.2\*e^2

3. Interpret the following statements in Matlab

- a)  $ab + c$
- b)  $\frac{a}{b} - c$
- c)  $p + \frac{w}{u - v}$
- d)  $x^{yz}$
- e)  $\frac{-b + \sqrt{b^2 - 4ac}}{2a}$
- f)  $x^{y+z}$

4. Evaluate the following Matlab statement. Firstly, you must calculate manually and then check your answer with Matlab.

- a)  $\frac{\frac{1}{3}}{\frac{2}{4}}$
- b)  $\frac{2 \times 3}{4} + 5$
- c)  $2 - \frac{3 \times 4}{6}$
- d)  $\frac{(3 - 4 \times 2)}{4} - \frac{6}{2}$
- e)  $3 - \frac{4}{2 + 3 \times 5}$
- f)  $5 \left( \frac{9}{4} \right) + \frac{5}{3}$
- g)  $4^3 \left[ \frac{3}{4} + \frac{9}{(2)3} \right]$

## **Recommended Reading**

Chapman, S.J. 2002. *Matlab Programming for Engineers 2<sup>nd</sup>*, USA: Brooks/Cole.

Hahn, B.D; Valentine, D.T. 2007. *Essential Matlab for Engineers and Scientists 3<sup>rd</sup>*, Amsterdam: Elsevier

## **Reporting**

The results of this UNIT must be reported to lecturer in the next meeting. Each student must report individually and present their results to get the feedbacks from the lecturer.

# UNIT 2

## Matlab Essential 2

### Learning Objective

The objectives of this Unit are to introduce some of fundamentals of Matlab programming, including:

- 1) Numbering format and rounding number commands.
- 2) Additional Matlab command, built-in functions
- 3) Essential Matlab functions
- 4) Matlab special constants

### Concepts

#### Numbering Format

The followings are numbering format provided by Matlab. We can activate it by setting Matlab preference or writing manually.

Table 2.1 Numbering Format

No	Command	Description	Output
1	>> format short	Fixed-point with 4 decimal digits	3.1429 ( 4 angka di belakang koma)
2	>> format long	Fixed-point with 14 decimal digits	3.14285714285714
3	>> format short e	Scientific notation with 4 decimal digits	3.1429e+000
4	>> format long e	Scientific notation with 14 decimal digits	3.142857142857143e+000
5	>> format rational	Rational expression	22/7
6	>> format short g	Best of 4 digit fixed or floating-point	3.14286
7	>> format long g	Best of 15 digit fixed or floating-point	3.14285714285714
8	>> format bank	2 decimal digits	3.14



## *Rounding Number Commands*

Matlab has some commands to round any number to specific integer number.

- `ceil(x)` : Round  $x$  to the nearest integer number toward  $\infty$
- `floor(x)`: round  $x$  to the nearest integer number toward  $-\infty$  .
- `fix(x)` : round  $x$  to the nearest integer number toward 0.
- `round(x)`: round  $x$  to the nearest integer number.
- `mod(x,y)`: returns the remainder after  $x$  is divided by  $y$  with definition  $x-n*y$  where  $n=\text{floor}(x./y)$ .
- `abs(x)`: absolute number of  $x$ .
- `sign(x)`: sign of  $x$ .
- `factor(x)`: main factor of  $x$ .

## *Additional Matlab Commands*

Matlab have some additional commands. There are

1. `clc` : cleaning the screen on the command window
2. `close all` : closing all figures displiced before.
3. `clear` : deleting all data in the Matlab memory.
4. `cd` : changing the directory.
5. `pwd` : this command is used if we want to know where we are now.
6. `dir` : this command is used to list all files in the current directory.
7. `mkdir` : be used to create a new directory.
8. `delete` : be used to delate a file
9. `who` : displaying all variable in the current time.
10. `Whos` : displaying all variable in the current time together with information about size, byte, class etc.

11. what : displaying all files with extension .M (M-File)
12. lookfor : command for looking for a file with keyword.

## Built-In Functions

### Trigonometric Functions

There are many trigonometric functions we have known. The function include in Matlab built in functions. Here are some functions included in trigonometric functions: sin(), cos(), tan(), sinh(), cosh(), tanh(), asin(), acos(), atan(), asinh(), acosh()dan atanh(). The important thing is that argument must be in radian.

### Essential Matlab Function

Beside of trigonometric functions, Matlab also provides many essential functions. There are **abs()**, **sqrt()**, **exp()**, **log()**, **log10()**, **log2()**. For more clearly, notice table 2.2 below

Table 2.2 Essential Matlab Functions

No	Nama variabel	Keterangan
1	abs(x)	Returns absolute value of x, or $ x $
2	sqrt(x)	Returns square root of (x), or $\sqrt{x}$
3	exp(x)	Returns eponential value of x or $e^x$
4	log(x)	Returns the natural logarithm of x, or $\ln(x)$
5	log10(x)	Returns the base 10 algorithm of x, or $\log(x)$
6	log2(x)	Returns the base 2 algorithm of x, or ${}^2\log(x)$

### Matlab Special Constants

Matlab provides some special constants. We recommend you to avoid giving a name of variable using the same name with the constants. Table 2.3 shows some special functions:

Table 2.3 Special functions

No	Constants	Description
1	pi	3.14159265...

2	i	Imaginary unit $\sqrt{-1}$
3	j	The same as i
4	eps	Floating-point relative accuracy
5	realmin	The smallest floating-point number
6	realmax	The biggest floating-point number
7	inf	Infinite number
8	NaN	Not-a-Number

### Activity

1. State the problem

**Conversion from Celcius to Fahrenheit degree.** As we know that there are some types of thermometer that can be used to measure the temperature of bodies. There are Celcius (very common used), Fahrenheit, Reamur and Kelvin. Now, we want to calculate the conversion from celcius to Fahrenheit based on equation

$$C = \frac{5}{9}(F - 32)$$

2. Define the inputs and output

The inputs of this problem is just degree in F. While, the output of this problem is C, that is the resulting conversion.

3. Describe the algorithm

This problem can be broken down into three steps:

- 1) Read the values of input F.
- 2) Apply the basic arithmetic operation to get C.
- 3) Write out the output (e.g the z)

### PROBLEMS

1. Use Matlab to evaluate the following expression. Answer are in bracket again.

(a)  $\sqrt{2}$

(b)  $\frac{1+2}{2+3}$

(c) the sum of 3 and 9 divided by their product.

(d)  $2^{3^4}$

(e) the square of  $\pi$

(f)  $3\pi^3$

(g)  $1/\sqrt{2\pi}$

(h)  $\frac{1}{2\sqrt{\pi}}$

(i) the cube root of the product of 3.2 and

2.3

$$(j) \frac{1 + \frac{2}{3+4}}{1 + \frac{2}{4-3}}$$

2. Water freezes at 32° and boils at 212° on the Fahrenheit scale. If C and F are Celcius and Fahrenheit temperatures, the Formula

$$F = 9/5 C + 32$$

convert from Celcius to Fahrenheit Scale. Use the Matlab command line to convert a temperature of 37° C (normal human temperature).

3. Scientist often has to convert from one unit to another. Form example, convert 5 acres to hectares, given that an acre is 4840 square yards, a yard is 36 inches, an inch is 2.54 cm, and a hectar is 10000 m<sup>2</sup>. The best approach is to develop a formula to convert x acres to hectares. You can do this as follows.

$$\text{One square yard} = (36 \times 2.54)^2 \text{ cm}^2$$

$$\text{so one acre} = 4840 \times (36 \times 2.54)^2 \text{ cm}^2$$

$$= 0.4047 \times 10^8 \text{ cm}^2$$

$$= 0.4047 \text{ hectare}$$

$$\text{so } x \text{ acres} = 0.4047 x \text{ hectare}$$

Based on the formula, we can do this with Matlab

$$x = 3;$$

$$h = 0.4047 * x;$$

$$\text{disp}(h)$$

Develop formulae for the following conversions, and use Matlab to find the answer

- (a) Convert 22 yards to meters.
- (b) One pound = 454 grams. Convert 100 kilograms to pounds.
- (c) Convert 49 meters/second (terminal velocity for person-shaped object) to km/hour.
- (d) One atmosphere pressure = 14.7 pounds pre square inch (psi) = 101.325 kilo Pascals

(kPa). Convert 40 psi to kPa.

(e) One calorie = 4.184 joules. Convert 6.25 kilojoules to calories.

4. The following Matlab statements plot the function  $y(x)=2e^{-0.2x}$  for the range  $0 \leq x \leq 10$

```
x=0:0.1:10;
```

```
y=2 * exp(-0.2 * x);
```

```
plot(x,y)
```

Use the Matlab edit Window to create a new empty file, type the statements into the file, save the file with the name test1.m. Back to command window and execute the file by typing test1. What result do you get?

### Recommended Reading

Chapman, S.J. 2002. *Matlab Programming for Engineers 2<sup>nd</sup>*, USA: Brooks/Cole.

Hahn, B.D; Valentine, D.T. 2007. *Essential Matlab for Engineers and Scientists 3<sup>rd</sup>*, Amsterdam: Elsevier

### Reporting

The results of this UNIT must be reported to lecturer in the next meeting. Each student must report individually and present their results to get the feedbacks from the lecturer.

# UNIT 3

## Matlab Essential 3

### Learning Objective

The objectives of this unit are to introduce some of basic concepts of Matlab programming, including:

1. How to use the built-in functions provided by Matlab: meshgrid, feval, polyval, polyfit, polyder, roots, poly, conv and deconv.
2. Applying the built-in functions to solve the problems.

### Concepts

#### *meshgrid*

Meshgrid function is used to create the grids on the x-y plane. This function will be useful when we want to plot 3 dimensional graph.

$[X,Y] = \text{meshgrid}(x,y)$  transforms the domain specified by vectors  $x$  and  $y$  into arrays  $X$  and  $Y$ , which can be used to evaluate functions of two variables and three-dimensional mesh/surface plots. The rows of the output array  $X$  are copies of the vector  $x$ ; columns of the output array  $Y$  are copies of the vector  $y$ .

$[X,Y] = \text{meshgrid}(x)$  is the same as  $[X,Y] = \text{meshgrid}(x,x)$ .

$[X,Y,Z] = \text{meshgrid}(x,y,z)$  produces three-dimensional arrays used to evaluate functions of three variables and three-dimensional volumetric plots.

### Example

Determine graph of the function  $z = x^2 - y^2$  with specified domain  $0 \leq x \leq 5$  and  $0 \leq y \leq 0$

Solution

Firstly, we must create the grids on x-y plane by using meshgrid function.

```
>> x=0:5;
```

```
>> y=0:5;
```

```
>> [X Y]=meshgrid(x,y);
```

Effect of applying meshgrid function is that elements of matrix  $X$  are formed by  $x$  and the elements of matrix  $Y$  are formed by  $y$ . Therefore, the value of  $z$  can be determined by

```
>> z=X.^2-Y.^2;
```

As example, the point of grid (3,4) has value  $z=3^2-2^2=5$  . To display the graph, we can use *mesh* function.

```
>> mesh(X,Y,z)
```

## ***feval()***

Feval() function is usually used to evaluate a function. To do that, the first step is to create the function we want to evaluate. In this example, we will use a function provided by Matlab, that is *humps*.

To evaluate the *humps* function, we must create a *handle* function by using notation @ (read et).

```
>> fhandle=@humps;
```

```
>> feval(fhandle,1)
```

```
ans =
```

```
16
```

## ***polyval***

polyval is used to determine the value of polynomial in the form

$$p(x)=a_0+a_1x^1+a_2x^2+a_3x^3+a_4x^4+\dots+a_{n-1}x^{n-1}+a_nx^n$$

Matlab has a simple way to express polynomial function as below

$$p=[ a_n \ a_{n-1} \ \dots \ a_3 \ a_2 \ a_1 \ a_0 ]$$

### **Example**

Given a polynomial  $p(x)=x^4+3x^2+4x+5$  . The polynomial will be evaluated at  $x=2, -3$  and 4.

### **Solution**

- Firstly, we must write the polynomial on the Matlab expression, that is  $p=[1 \ 0 \ 3 \ 4 \ 5]$ .
- Secondly, we determine the point of evaluation, that is  $x=[2,-3,4]$
- Thirdly, evaluate the polynomial on point x, that is  $\text{polyval}(p,x)$

If we write on the command window

```
>> p=[1 0 3 4 5];
```

```
>> x=[2,-3,4];
```

```
>> polyval(p,x)
ans =
    41   101   325
```

## ***Polyfit***

Polyfit function will be very useful when we have to fit the data we have obtained from an experiment. We can fit our experiment data with linear, quadratic, third order polynomial, etc depending on behaviour of the data. The general form of fit function is

$$p = \text{polyfit}(x,y,n)$$

with  $n$  is order of the polynomial we are applied to fit the experiment data.

## ***polyder***

Polyder function is used to differentiate a polynomial. The general form of the function is

$$k = \text{polyder}(p)$$

or

$$k = \text{polyder}(a,b)$$

### **Example**

Differentiate the polynomial below

$$p(x) = 4x^4 + 3x^2 + 4x + 5$$

### **Solution**

```
p=[4 0 3 4 5];
```

```
polyder(p)
```

```
ans =
```

```
    16     0     6     4
```

We can write the result on the mathematical expression as

$$16x^3 + 6x + 4$$

### **Example**

Differentiate the polynomial below

$$p(x) = (x^4 + 3x^2 + 4x + 5)(2x^3 + x^2 + 3x + 1)$$



## Solution

```
a=[4 0 3 4 5];
```

```
b=[2 1 3 1];
```

```
p=polyder(a,b)
```

```
p =
```

```
56 24 90 60 69 40 19
```

or

$$p(x) = 56x^6 + 24x^5 + 90x^4 + 60x^3 + 69x^2 + 40x + 19$$

## roots

Roots function is used to find the roots of a n order polynomial. The general form of the function is

$$r = \text{roots}(c)$$

## Example

Given a polynomial  $p(x) = 4x^4 + 3x^2 + 4x + 5$ . Find roots of the polynomial using roots function.

## Solution

Polynomial  $p(x) = 4x^4 + 3x^2 + 4x + 5$  can be written in Matlab expression as

$$p = [4 \ 0 \ 3 \ 4 \ 5]$$

By using roots function

```
roots(p)
```

```
ans =
```

```
0.6364 + 1.0830i
```

```
0.6364 - 1.0830i
```

```
-0.6364 + 0.6222i
```

```
-0.6364 - 0.6222i
```

## poly

Poly function is used to obtain a polynomial when the roots of polynomial have been determined. The general form of poly function is

$$p = \text{poly}(r)$$

### Example

Given roots of polynomial p are 1,2,3,4 and 5. Find form of the polynomial

### Solution

```
>> r=[1 2 3 4 5];
```

```
>> poly(r)
```

```
ans =
```

```
1 -15 85 -225 274 -120
```

or

$$p(x) = x^5 - 15x^4 + 85x^3 - 225x^2 + 274x - 120$$

### **conv**

Conv function is used to multiply two polynomials. The general form of the function is

$$w = \text{conv}(u,v)$$

### Example

Given two polynomials  $u(x) = x^2 + 3x + 2$  and  $v(x) = x^3 + 2x^2 + 3x + 1$ . Find the multiplication result of the two polynomials.

### Solution

```
>> u=[1 3 2];
```

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

```
>> conv(u,v)
```

```
ans =
```

```
1 5 11 14 9 2
```

or

$$x^5 + 5x^4 + 11x^3 + 14x^2 + 9x + 2$$

### **deconv**

*Deconv* function is opposite to *conv*. This function will perform division operation of the functions. General form of this function is

$$[q,r] = \text{deconv}(v,u)$$

where  $q$  and  $r$  are result and residue respectively.

### Example

Given two polynomials  $p_1(x)=2x^4+3x^3+x^2+4x+5$  and  $p_2(x)=x^2+3x+4$ . Find result of the division operation and its residue from  $p_1$  and  $p_2$ .

### Solution

Using `deconv` function, we can find the division result and its residue,

```
>> p1=[2 3 1 4 5];  
>> p2=[1 3 4];  
>> [q r]=deconv(p1,p2)
```

$q =$

2 -3 2

$r =$

0 0 0 10 -3

or, if we express into mathematical expression

$$q=2x^2-3x+2 \text{ dan } r=10x-3$$

### Activity

1. State the problem

Suppose that we performed an experiment, and then we obtained experiment data as shown belows

x	1	2	3	4	5	6	7	8	9	10
y	1.3	3.2	11.3	15.1	25.5	38.2	47.1	68.2	81.3	98.2

Based on data we have obtained, the first thing we have to think about data is to guess the trend of the data. Is it tend to linear, quadratic or other. It is crucial because one is to determine the order of polynomial.

2. Define the inputs and output

The inputs of this program are data as shown on data table above. There are data of independent variables  $x$  and data of dependent variables  $y$ . Make them as vector. Then, the outputs of this data fitting program are a specific polynomial  $p$  and data evaluation of  $p$ . You can use `polyval` to evaluate the polynomial.

3. Describe the algorithm

This problem can be broken down into three steps:

- (a) Read the values of input, namely,  $x$  and  $y$  as a vector of each.
- (b) Apply the predefined functions, these are *polyfit* to fit the data to specific polynomial and *polyval* to evaluate the polynomial at any points.
- (c) Write out the output (e.g the polynomial  $p$  and evaluation data table of  $p$  )

## Problems

1. The ideal gas law relates the pressure  $P$ , volume  $V$ , absolute temperature  $T$ , and amount of gas  $n$ . The law is

$$P = \frac{nRT}{V}$$

where  $R$  is the gas constant. An engineer must design a large natural gas storage tank to be expandable to maintain the pressure constant at 2.2 atmospheres. In December when the temperature is 4°F (-15°C), the volume of gas in the tank is 28.500 ft<sup>3</sup>. What will the volume of the same quantity of gas be in July when the temperature is 88°F (31°C)? (Hint: Use the fact that  $n, R$  and  $P$  are constant in this problem. Note also that  $K = ^\circ C + 273.2$ ).

2. Use Matlab to calculate

- (a)  $e^{(-2.1)^3} + 3.47 \log(14) + \sqrt[4]{287}$

- (b)  $(3.4)^7 \log(14) + \sqrt[4]{287}$

- (c)  $\cos^2\left(\frac{4.12\pi}{6}\right)$

- (d)  $\cos\left(\frac{4.12\pi}{6}\right)^2$

3. Use Matlab to evaluate

$$\frac{8x^3 - 9x^2 - 7}{10x^3 + 5x^2 - 3x - 7} \quad \text{at } x=5$$

4. Given a function with two variables

$$z = \sin\left(\frac{r}{r}\right), \quad \text{where } r = \sqrt{x^2 + y^2} \quad \text{and} \quad -8 \leq x \leq 8 \quad \text{and} \quad -8 \leq y \leq 8$$

- (a) Find array  $X$  and  $Y$  as result of meshgrid from the variable  $x$  and  $y$ .

- (b) Plot the result in 3D graph

5. Given a polynomial

$$P(x) = x^8 - 3x^5 + 7x^2 - 10x + 1$$

Find the value of  $P$  at point  $x=1, -4, 10$ , and  $3$ .

## **Recommended Reading**

Chapman, S.J. 2002. *Matlab Programming for Engineers 2<sup>nd</sup>*, USA: Brooks/Cole.

Hahn, B.D; Valentine, D.T. 2007. *Essential Matlab for Engineers and Scientists 3<sup>rd</sup>*, Amsterdam: Elsevier

## **Reporting**

The results of this UNIT must be reported to lecturer in the next meeting. Each student must report individually and present their results to get the feedbacks from the lecturer.

# UNIT 4

## MATRIX

### Learning Objectives

The objective of this unit are to introduce some of basic concepts related to matrix:

1. Create a matrix including coulumn and row matrix.
2. Identify individual element of a matrix.
3. Use the colon operator and arithmetic operator.
4. Delete the row and coulumn of a matrix and manipulate the elements .
5. Create a sparse matrix

### Concepts

As we know that Matlab stands for *Matrix Laboratory*, because Matlab is designed specially to work with data arranged in the form of matrices. In this chapter, matrix has two distint meanings:

1. an arrangement of data in rows and coulomns
2. a mathematical object, for which mathematical operations are defined.

### Creating Matrices

Matlab have ability to create matrices simply. The complicated matrices can be constructed by simpler one. Notice how to make matrices below. Use semicolon to indicate the end of a row when entering a matrix.

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

```
>> x=[4,5,6];
```

```
>> B=[A;x]
```

```
B =
```

```
1 2 3
```

```
4 3 4
```

```
3 2 1
```

```
4 5 6
```

### Subscript

Individual element of matrix are referenced with two subscripts, the first for row and the second for colomn.

```
>> A=[1,2,3;4,3,4;3,2,1];
>> A(3,3)
ans =
    1
```

### Transpose

Transpose is defined to change elements of matrix in row to column and reverse elements in column to row .

```
>> A=[1,2;3,4];
>> b=A'
b =
    1    3
    2    4
```

## *The colon operator*

The colon operator has extremely powerful, and provides very efficient ways of handling matrices. Notice example below

```
>> A=[1,2,3;4,3,4;3,2,1];
>> A(1:2;2:3)
>> A(1:2,2:3)
ans =
    2    3
    3    4
```

(returns first and second rows, second and third columns). The statement

( 1 , : ) result in

```
1    2    3
```

(returns first row), and statement A(1:2,2:3)=zeros(2)

A(1:2,2:3)=zeros(2) result in

```
1    0    0
4    0    0
3    2    1
```

We can also construct a table based on the colon operator. Suppose we want a table *trig* of the sines and cosines of the angles 0° to 180° in steps of 30°. The following statement achieve this:

```
>> x=[0:30:180]';
>> trig(:,1)=x;
>> trig(:,2)=sin(pi/180*x);
>> trig(:,3)=cos(pi/180*x);
>> trig
trig =
     0     0  1.0000
 30.0000  0.5000  0.8660
 60.0000  0.8660  0.5000
 90.0000  1.0000  0.0000
120.0000  0.8660 -0.5000
150.0000  0.5000 -0.8660
180.0000  0.0000 -1.0000
```

The colon operator is ideal for the sort of row operation performed in Gauss reduction. For example, if A is the matrix

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

the statement

```
A(2,:)=A(2,.)-A(2,1)/A(1,1)*A(1,.)
```

resulting in

```
 1  2  3
 0 -5 -8
 3  2  1
```

and then the statement

```
>> A(3,:)=A(3,.)-A(3,1)/A(1,1)*A(1,.)
```

```
A =
```

```
 1  2  3
 0 -5 -8
 0 -4 -8
```

```
>> A(3,:)=A(3,.)-A(3,2)/A(2,2)*A(2,.)
```

```
A =
```

```
 1.0000  2.0000  3.0000
```



```
0   -5.0000  -8.0000
0    0       -1.6000
```

The keyword `end` refers to the last elements of a vector. Note that vector is special case of matrix, because it only has one row or column. For example, if `A` is a vector, the statement

```
sum(A(1:end))
```

returns the sum of array elements from 1 to last one.

## Duplicating Row and Column

Sometimes, it is useful to create a matrix where all rows or columns are the same. It is can be done with the `repmat` function. Suppose `A` is a row vector,

```
>> A=[1,2,3];
>> repmat(A,[3 1])
ans =
     1     2     3
     1     2     3
     1     2     3
>> repmat(A,[1 3])
ans =
     1     2     3     1     2     3     1     2     3
```

The alternative syntax for `repmat` is,

```
>> repmat(A,3,1)
```

## Deleting rows or Columns

We can use column operator and the empty array to delete entire rows or columns. For example

```
>> A=[1,2,3;4,3,4;3,2,1];
>> A(1,:)=[]
A =
     4     3     4
     3     2     1
>> A(:,1)=[]
A =
```

```
3 4
2 1
```

## Specialized Matrices

The followings are functions that could be used to generate arbitrary matrices when we can not think of one to generate our self.

1. *pascal* (*n*), the function is used to generate a pascal matrix.
2. *magic*(*n*), the function is used to generate a magic matrix.
3. *zeros*(*n*), the function is used to generate an array of all zeros.
4. *ones*(*n*) generates an array of all ones
5. *rand*(*n*) generates an array with uniformly random distributed elements.
6. *randn*(*n*) generates an array with normally random distributed elements.

## Manipulating Matrices

Here are some function s for manipulating matrices. See help for details

1. *diag* extracts or creates a diagonal
2. *fliplr* flip from left to right.
3. *flipud* flip from top to down
4. *rot90* rotate 90°
5. *tril* extracts the lower triangular part, e.g, the statement

```
>> A=pascal(4);
>> tril(A)
ans =
    1    0    0    0
    1    2    0    0
    1    3    6    0
    1    4   10   20
```

## Matrix Multiplication

The matrix product  $C = AB$  is defined when the column dimension of A is equal to the row

dimension of B, or when one of them is a scalar. If A is m-by-p and B is p-by-n, their product C is m-by-n. The product can actually be defined using MATLAB for loops, colon notation, and vector dot products. A = pascal(3);

```
B = magic(3);  
m = 3; n = 3;  
for i = 1:m  
    for j = 1:n  
        C(i,j) = A(i,:)*B(:,j);  
    end  
end
```

MATLAB uses a single asterisk to denote matrix multiplication. The next two examples illustrate the fact that matrix multiplication is not commutative; AB is usually not equal to BA. X = A\*B

```
X =  
    15    15    15  
    26    38    26  
    41    70    39
```

```
Y = B*A
```

```
Y =  
    15    28    47  
    15    34    60  
    15    28    43
```

A matrix can be multiplied on the right by a column vector and on the left by a row vector. u = [3; 1; 4];

```
x = A*u  
x =  
     8
```

17

30

$v = [2 \ 0 \ -1];$

$y = v*B$

$y =$

12 -7 10

Rectangular matrix multiplications must satisfy the dimension compatibility conditions.  $C =$   
`fix(10*rand(3,2));`

$X = A*C$

$X =$

17 19

31 41

51 70

$Y = C*A$

Error using `==> *`

Inner matrix dimensions must agree.

Anything can be multiplied by a scalar.  $s = 7;$

$w = s*v$

$w =$

14 0 -7

## ***Sparse Matrix***

In the physics and engineering, we often find the problems involving matrix containing a large enough percentage of zeros. The density of a matrix is the number of non-zero elements divided by the total elements of matrix. The low density matrix are often good candidates for use of the sparse format. The general form of using *sparse* function

*sparse(row,colomn,input,m,n)*

For example, if we want to create a 5-by-5 sparse matrix as below

$$\begin{bmatrix} 2 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{bmatrix}$$

we can create simply,

```
>> p=sparse(1:5,1:5,2,5,5);
>> p=p+sparse(2:5,1:4,-1,5,5);
>> p=p+sparse(1:4,2:5,-1,5,5);
>> full(p)
ans =
    2.00    -1.00     0         0         0
   -1.00     2.00    -1.00     0         0
    0      -1.00     2.00    -1.00     0
    0         0     -1.00     2.00    -1.00
    0         0         0     -1.00     2.00
```

## Activity

### 1. State the program

In the first program, we will solve a problem related to sparse matrix. Because it is often applied in physics. The form of a sparse matrix we have to create is

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

### 2. The Inputs and Output of program

In this program we will apply the sparse matrix function. The inputs of this program are non-zero elements (-1 and 2), vector of row and columns indices and the number of row and column

of the matrix. Output of this program is the form of matrix that it is shown as above.

### 3. Design the algorithm

This program can be broken down into three steps:

*Read the inputs*

*Manipulate the sparse matrix*

*Write out the outputs*

We will discuss one-by-one these steps to make us clearly. The resulting pseudocode for these steps are just follows:

- a) *Prompt the user to input a vector of row and column indices (i,j, respectfully), non-zero elements (-1 and 2), the number of row and columns (m and n).*
- b) *Read the inputs i, j, non-zero, m and n.*
- c) *A <----- zeros(m,n)*  
*A <----- sparse(i:m, j:n,2,m,n);*  
*A <----- A+sparse(i+1:m, j : n-1, -1 , m,n);*  
*A <----- A+sparse(i:m-1, j+1: n,-1, m , n);*  
*full(A)*

### Problems

1. Given two vectors  $\mathbf{A}=(1 \ -3 \ 2 \ 5 \ 6)$  and  $\mathbf{B}=(4; 1; 3; 5; 1)$ . Find
  - (a) magnitude of the two vectors
  - (b) multiplication product of the two vectors
2. Given vector  $\mathbf{A}=(2+i \ -3+5i \ 5 \ 1-3i \ 2)$ .
  - (a) Find the transpose of matrix A
  - (b) Find the transpose conjugate of matrix A
  - (c) Find the magnitude of  $\mathbf{A}$ .
3. Suppose that we have a few numbers 3,2,3,1 and 6. Create a column and row vector with the

given number elements..

4. Given  $A=[1,2,3,5,3]$  and  $B=[4;3;2;5;2]$ . Find

(a) Cross product of vector A and B.

(b) transpose of B

(c) Dot product of vector A and B.

5. Given two matrices A and B

$$A = \begin{pmatrix} 3 & 2 & -1 & 2 & 4 \\ -2 & 1 & 1 & 3 & 2 \\ 8 & 2 & -4 & 3 & -5 \end{pmatrix} \quad B = \begin{pmatrix} 4 & 1 & -1 & 2 & 7 \\ 6 & 3 & 0 & 3 & 6 \\ 1 & -2 & 14 & 2 & -5 \end{pmatrix}$$

(a) Find multiplication product of A and B

(b) Find invers of A and B

(c) Find  $AA^{-1}$ ,  $AB$ ,  $BB^{-1}$  and  $A^{-1}B^{-1}$ .

6. Create command lines to construct a sparse matrix with order 7-by-7 as belows

$$(a) \quad A = \begin{pmatrix} 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 \end{pmatrix}$$

$$(b) \quad A = \begin{pmatrix} 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 4 & 3 & 0 & 0 & 0 & 0 & 0 \\ 5 & 0 & 3 & 0 & 0 & 0 & 0 \\ 6 & 0 & 0 & 3 & 0 & 0 & 0 \\ 7 & 0 & 0 & 0 & 0 & 0 & 0 \\ 8 & 0 & 0 & 0 & 0 & 3 & 0 \\ 9 & 0 & 0 & 0 & 0 & 0 & 3 \end{pmatrix}$$

## Recommended Reading

Chapman, S.J. 2002. *Matlab Programming for Engineers 2<sup>nd</sup>*, USA: Brooks/Cole.

Hahn, B.D; Valentine, D.T. 2007. *Essential Matlab for Engineers and Scientists 3<sup>rd</sup>*, Amsterdam: Elsevier

## Reporting

The results of this UNIT must be reported to lecturer in the next meeting. Each student must report individually and present their results to get the feedbacks from the lecturer.





# UNIT 5

## BRANCHING STATEMENTS

### Learning Objective

The objectives of this unit are to introduce some of basic concepts of Matlab, including:

1. Logic operator and its application.
2. The if constructor and its applications to solve the mathematical problems.

### Concepts

In this topic we will discuss the branching statements that allow us to control the order in which statements are executed in a program. Before that, we are going to introduce some relational operator which will be very important .

Relational Operator

<i>Operator</i>	<i>Operation</i>
==	Equal to
~=	Not equal to
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to

### Logic Operator

Logic operators are operators with one or two logical operands that yield logical result. There three main logic operators: AND, OR and EXCLUSIVE OR (XOR)

<i>Operator</i>	<i>Operation</i>
&	Logical AND
	Logical OR
xor	Logical XOR
~	Logical NOT

### Branches

The if Construct

The if construct has the form

```
if control_expr_1
```

```

Statement 1      )
Statement 2      ) BLOCK 1
....            )
else if (control_expr_2)
Statement 1      )
Statement 2      ) BLOCK 2
....            )
else
Statement 1      )
Statement 2      ) BLOCK 3
....            )
end

```

where the control expression control operation of the if construct. If the value of control expression is non-zero, the statements in Block 1 will be executed. Otherwise, the control expression 2 will be checked. If the value of control expression 2 is true, then statements in Block 2 will be executed. If control expression 1 and 2 are false, then statements in Block 3 will be executed.

## Activity

### 1. State the problems

Calculate and write out the roots of a quadratic equation. We must give the category of the roots, whether the roots are real or imaginary.

### 2. Define Inputs and Outputs

The inputs required in this program are the coefficients  $a$ ,  $b$  and  $c$  of the quadratic equation

$$ax^2 + bx + c = 0$$

The outputs of this program are the roots  $x_1$  and  $x_2$ . In addition, we must write out type of the roots, whether they are real or imaginary.

### 3. Design the algorithm

This program can be broken down into three steps:

- 1) Read the inputs data

2) Calculate the roots

3) Write out the roots

As you know that there are three possible ways to calculate the roots, depending on the value of its discriminant. If the value of discriminant is greater than 0, so the two roots are real. If the value of the discriminant is equal to 0, the two roots are the same and if the discriminant is less than 0, so the roots are imaginary. Considering the problem, it is logical to implement this algorithm with a three-branched if construct. The resulting pseudocode is

*Prompt the user to input the three coefficients a,b and c*

*Read the coefficients a, b and c*

*Determine the discriminant  $D = b^2 - 4ac$*

*if  $D > 0$*

$$x_1 \leftarrow \frac{-b + \sqrt{D}}{2a}$$
$$x_2 \leftarrow \frac{-b - \sqrt{D}}{2a}$$

*Give a message that the equation has two distinct roots and the roots are real. Write out the roots.*

*Else if  $D == 0$*

$$x_1 = x_2 \leftarrow \frac{-b + \sqrt{D}}{2a}$$

*Give a message that the two roots are the same and real.*

*Write out the roots.*

*Else*

$$\text{real\_part} \leftarrow -b/(2*a)$$

$$\text{imag\_part} \leftarrow \text{sqrt}(\text{abs}(D))/(2*a)$$

*Give a message that the equation has two distinct complex roots.*

*Write out the roots.*

## Problems

1. Write a program to evaluate a function  $f(x,y)$  for any user-specified values  $x$  and  $y$ . The function  $f(x,y)$  is defined as follows

$$f(x,y) = \begin{cases} x+y, & \text{for } x \geq 0, y \geq 0 \\ x+y^2, & \text{for } x \geq 0, y < 0 \\ x^2+y, & \text{for } x < 0, y \geq 0 \\ x^2+y^2, & \text{for } x < 0, y < 0 \end{cases}$$

2. Suppose that we must give the final grade after the final examination. The program will read in a numerical grade and assign a letter grade to it according to the following table

$grade > 95$	$A$
$95 \geq grade > 86$	$B$
$86 \geq grade > 76$	$C$
$76 \geq grade > 66$	$D$
$66 \geq grade > 0$	$E$

Write an if construct to assign the grade as described above using (a) multiple elseif clauses and (b) nested if construct.

## Recommended Reading

Chapman, S.J. 2002. *Matlab Programming for Engineers 2<sup>nd</sup>*, USA: Brooks/Cole.

Hahn, B.D; Valentine, D.T. 2007. *Essential Matlab for Engineers and Scientists 3<sup>rd</sup>*, Amsterdam: Elsevier

## Reporting

The results of this lab work must be reported to lecturer in the next meeting. Each student must report individually and present their results to get the feedbacks from the lecturer.

## UNIT 6

### REPEATING WITH FOR

#### Learning Objectives

The objectives of this unit are to introduce some of basic concepts of Matlab including:

1. The basic construct of repeating *for*.
2. The use of the repeating *for*; its application to a numerical method.
3. Avoiding *for* with vectorizing

#### Concepts

Statement *for* is very useful in constructing computer programming. To describe how the statement do, notice statement s below:

```
for i =1:5; disp(i);  
for i=1:3; disp(i);  
for i=1:0; disp(i)
```

#### Square root with Newton's Method

The square root  $x$  of any positive number  $a$  may be found using only arithmetic operation of addition, subtraction and division with Newton's method. This is an iterative method that refines an initial guess. The procedure for finding the square root of  $x$  is the following

1. initial guess  $a$
2. initialize  $x$  to  $a/2$
3. repeat 6 times (say)
  1. replace  $x$  by  $(x+a/x)/2$
  2. display  $x$
4. Stop

#### Factorial

Factorial of an integer number  $n$  is defined as

$$n! = 1 \times 2 \times 3 \times 4 \cdots (n-1) \times n$$

Here are structure plan to create factorial program

1. Determine the number to be factored, say  $N$
2. initialize fact to 1
3. repeat  $N$  times
  1. replace fact by  $k \times \text{fact}$

2. display fact
4. Stop

### The Basic Construct of for Repetition

In general the most common form of the *for* repetition (for use in a program, not on the command line) is

```
for index=j:k
    statements
end
```

or

```
for index=j:m:k
    statements
end
```

Note the following points carefully:

1.  $j:k$  is a vector with elements  $j, j+1, j+2, j+3, \dots, k$
2.  $j:m:k$  is a vector with elements  $j, j+m, j+2m, j+3m, \dots$  such that the last element does not exceed  $k$  if  $m > 0$ .

### Avoid *for* repetition by vectorizing

There are situations where the *for* repetition is essential, as many examples presented here. However, the way Matlab designed, the *for* repetition tend to be inefficient in terms of computing time. For example, suppose we want to evaluate

$$\sum_{n=1}^{100000} n^2$$

Here is how to do it with the *for* repetition

```
t0=clock;

N=100000;

sum=0;

for n=1:N

    sum=sum+n^2;
```

```
end  
etime(clock, t0)
```

Now, try to vectorize the calculation (before looking at the solution). Here it is

```
t0=clock;  
N=100000;  
n=1:N;  
s=sum(1./n.^2);  
etime(clock, t0)
```

The last way takes only 0.0160 second on the dual core, it is more than 12.4 times faster with the same computer. Now suppose, we want to calculate

$$\sum_{n=1}^{100000} \frac{1}{n^2}$$

Here is the for repetition with tic and toc to write the elapsed time used to be

```
tic  
N=100000;  
sum=0;  
for n=1:N  
    sum=sum+1/n^2;  
end  
toc
```

This way takes about 0.4060. No if we try to vectorize the sum

```
tic  
N=100000;  
n=1:N;  
s=sum(1./n.^2);
```

toc

The elapsed time of this calculation takes only 0.016 second or 26.4 times faster compared with the first way.

### A Common Mistakes: for less loop

A very common mistakes is to omit the word *for* in the repetition. It will give the terrible result, of course. For example

```
tic
N=100000;
sum=0;
n=1:N
    sum=sum+1/n^2;
end
toc
```

### Problems

1. Write Matlab programs to find the followings sum (a) with for repetition and (b) by vectorization. Time both version each case.

(a)  $1^2+2^2+3^2+4^2+\dots+2000^2$

(b)  $1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\frac{1}{9}-\dots-\frac{1}{1003}$

- (c) sum the left hand side of the series

$$\frac{1}{1^2 \cdot 3^2} + \frac{1}{3^2 \cdot 5^2} + \frac{1}{5^2 \cdot 7^2} + \dots = \frac{\pi^2 - 8}{16}$$

2. One of method to approach the integral expression is Simpson 1/3 rule. The method can be defined as

$$I = \frac{h}{3} \left[ f_0 + f_N + 4 \sum_{i=1}^{N/2} f_{2i-1} + 2 \sum_{i=1}^{N/2} f_{2i} \right]$$



if  $f = x e^{-x}$ , try to obtain result of the calculation with *for* loop and vectorization.

3. Calculate the squares of every integer number from 1 to 10000 in a for loop without initializing the array of the squares first.
4. Calculate the squares of every integer number from 1 to 10000 in a for loop using the zeros function to pre-allocate the array of squares first.
5. Calculate the squares of every integer number from 1 to 10000 with vector.

### **Recommended Reading**

Chapman, S.J. 2002. *Matlab Programming for Engineers 2<sup>nd</sup>*, USA: Brooks/Cole.

Hahn, B.D; Valentine, D.T. 2007. *Essential Matlab for Engineers and Scientists 3<sup>rd</sup>*, Amsterdam: Elsevier

### **Reporting**

The results of this lab work must be reported to lecturer in the next meeting. Each student must report individually and present their results to get the feedbacks from the lecturer.

# UNIT 7

## THE WHILE LOOP

### Learning Objectives

The objectives of this unit are to introduce some of basic concepts of Matlab including:

1. The basic construct of repeating *while loop*.
2. The use of the repeating while loop, its application to stastical physics

### Concepts

The *while loop* is a block of statements that is repeated indefinitely as long as the condition is satisfied. The general form of a while loop is

```
while (condition)
    statements
    .....
    .....
end
```

If the condition is non zero (true), the statements between the word while and end will be executed, and control will return to the while statement. If the condition is still non zero, the statements will be executed again. The process will be executed until the condition become zero (false). When control returns to the while statement and the condition is zero, the program will execute the first statement after the end.

### Statistical Analysis

It is very common in physics experiment, that we get a large set of data. What can we do with the data? If we measure a physics quantity repeatedly, so we must get the best point. The best point of numbers is called average number and defined as

$$\bar{x} = \frac{1}{N} \sum_{i=1}^N x_i \quad (6-1)$$

where  $x_i$  is sample  $i$  of  $N$  data. If all of the input values are available in an array, the everage of a set of numbers can be calculated either directly from equation (6-1) or elseby built-in Matlab function

mean.

The standard deviation of a set numbers is defined as

$$s = \sqrt{\frac{N \sum_i x_i^2 - \left(\sum_i x_i\right)^2}{N(N-1)}}$$

Standard deviation is a measure of the amount of scatter on the measurements. The greater standard deviation, the more scattered are the points in the data set. If all of the input values are available in an array, the standard deviation can be calculated either directly from equation (6-2) or else by Matlab built-in function *std*.

## Activity

### 1. State the problem

Calculate the average and standard deviation of a set of measurement data set, assuming that all data are positive or zero and assuming that we do not know how many measurements are included in the data set. A negative data input will mark the end of the set of measurements.

### 2. Define the inputs and outputs

The data required in this program are an unknown number of positive or zero numbers. The outputs of this program are a printout of the average and standard deviation of the inputs data set. In addition, the number of inputs is also printed out to check that the inputs are read correctly.

### 3. Design the algorithm

This program can be broken down into three major steps.

*Accumulate the input data*

*Calculate the average and standard deviation*

*Write out the average, standard deviation and number of points.*

The pseudocode for these steps is:

*Initialize N, sum\_x, sum\_x2 to 0*

*Prompt user for first number*

*Read in first x*

*while x >= 0*

*n <----- n+1*

*sum\_x <----- sum\_x + x*

*sum\_x2 <----- sum\_x + x^2*

*Prompt user for next number*

*Read in next x*

*end*

Next, we must calculate the average and standard deviation. The pseudocode of the steps are just like equation (6-1) and (6-2).

*x\_avg <----- 1/N \* sum\_x*

*x\_std <----- sqrt( (N \* sum\_x2 - (sum\_x)^2) / (N \* (N-1)) )*

Finally, we must write out the result

*write out the average value x\_avg*

*write out the standard deviation*

*write out the number of input data points.*

## Problems

1. Write Matlab programs to find the followings sum with the *while* statement, so that the last term less than  $10^{-5}$ . Finish the problems using the steps we have done. If you want, time how long does the process take.

(a)  $1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \dots$

(b)  $1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \dots$

(c)  $1 - \frac{1}{3^2} + \frac{1}{5^2} - \frac{1}{7^2} + \frac{1}{9^2} - \dots$

### **Recommended Reading**

Chapman, S.J. 2002. *Matlab Programming for Engineers 2<sup>nd</sup>*, USA: Brooks/Cole.

Hahn, B.D; Valentine, D.T. 2007. *Essential Matlab for Engineers and Scientists 3<sup>rd</sup>*, Amsterdam: Elsevier

### **Reporting**

The results of this lab work must be reported to lecturer in the next meeting. Each student must report individually and present their results to get the feedbacks from the lecturer.

# UNIT 8

## *The Break and Continue Statements*

### Learning Objectives

The objectives of this unit are to introduce some of concepts, including:

1. Basic concept of using break and continue statements
2. Nesting loop
3. Fitting a line to a set of noisy measurement

### Concepts

There are two additional statements that can be used to control the while and for loop. There are break and continue statements. The break statement will terminate the execution of loop and pass control to the next statement after the end of the loop, while the continue statement will terminate the current passing through the loop and return to the top of the loop. For more clearly, notice the example below

```
for i=1:5
    if i==3
        break;
    end
    fprintf('%i ',i);
end
disp('berakhir');
```

resulting in

```
1 2 berakhir
```

The following is the example for the continue standard

```
for i=1:5
    if i==3
        break;
    end
    fprintf('%i ',i);
end
disp('berakhir');
```

resulting in

```
1 2 4 5 berakhir
```

### Nesting Loops

It is possible for one loop to be completely inside another loop. If one loop is completely inside another loop, it is called **nested loops**. Notice the example below

```
for i=1:5
    for j=1:5
        z=i + j;
        printf('%i + %i = %i \n',i,j,z);
    end
end
```

resulting in

```
1 + 1 = 2
1 + 2 = 3
1 + 3 = 4
2 + 1 = 3
2 + 2 = 4
2 + 3 = 5
3 + 1 = 4
3 + 2 = 5
3 + 3 = 6
```

### Fitting a line to a set of noisy measurement

The velocity of an object in the presence of a constant gravitational field is given by

$$v(t) = v_0 + at \quad (7-1)$$

where  $v(t)$  is the velocity of an object each time,  $v_0$  is the initial velocity,  $t$  is time and  $a$  is the acceleration due to gravity. From the function  $v$ , we can plot the velocity versus time. As you know that the graphic is straight line. However, when we go out from the laboratory and attempt to measure the velocity versus time of an object and plot the data, our data will not fall along a straight line. They may come close, but never line up perfectly. Why not? Because we never measure perfectly. In this case, we need the *linear regression*. Given a noisy set of measurements  $(x,y)$  that appear to fall along a straight

line, how can we find the equation of the line

$$y = mx + b \quad (7-2)$$

which best fits the measurements? If we can find the regression coefficient  $m$  and  $b$ , then we can predict  $y$  at any given  $x$ . A standard method for finding the coefficients  $m$  and  $b$  is the *method of least square*. The slope of the least square line is given by

$$m = \frac{(\sum xy) - (\sum x)\bar{y}}{(\sum x^2) - (\sum x)x} \quad (7-3)$$

and the intercept of the least square line is given by

$$b = \bar{y} - m\bar{x} \quad (7-4).$$

## Activity

### 1. State of the problem

Calculate the slope  $m$  and intercept  $b$  of the least square line to produce the line such as (7-2).

The input data  $(x,y)$  is read from the keyboard. Plot both the input data points and the fitted line.

### 1. Define Inputs and Outputs

The inputs required in this program are the number of data points, plus the pair of points  $(x,y)$ .

The output of this program are the slope  $m$  and the intercept  $b$  of the best line fitted.

### 2. Describe the algorithm

This program can be broken down into six major steps:

- (1) Get the number of inputs data points
- (2) Read the input data values
- (3) Calculate the required statistics
- (4) Calculate the slope and intercept
- (5) Write out the slope and intercept
- (6) Plot the data points and fitted line.

The pseudocode of these steps is shown below



```

Print message describing purpose of the program
n_points <----- input('input the number of data points');
for i=1:n_points
    temp <----- input('input data values [ x y] pair')
    x(i) <----- temp(1)
    y(i) <----- temp(2)
end

```

Next, we must accumulate the required statistics data. There are the sums  $\sum xy$ ,  $\sum x$ ,  $\sum x^2$  and  $\sum y$ . The pseudocode of this program is

initialize sum\_x, sum\_xy, sum\_x2 and sum\_y to 0

```

for i=1 : n_points
    sum_x <-----sum_x + x
    sum_xy <----- sum_xy + x*y
    sum_x2 <----- sum_x2 + x^2
    sum_y <----- sum_y + y
end

```

Next, we must calculate the slope and intercept of the least square line. The pseudocode of this program is

```

y_bar <----- y / n_points
x_bar <----- x / n_points
slope <----- (sum_xy - sum_x * y_bar) / (sum_x2 - sum_x * x_bar)
y_int <----- y_bar - m * x_bar

```

Finally, we must write out and plot the results. We must plot the data points and fitted line in one frame. To do that, we must use the *hold* command. Recall, if we want to add a picture into previous figure, we write *hold on*. In the end of program, do not forget to write *hold off*.

## Problems

1. When a ball is thrown upward, trajectory of the motion will be parabolic. As you know that there are two components of motion, x-component and y-component. When we start throwing, we have two initial velocity components, these are  $v_{0x} = v_0 \cos \theta$  and  $v_{0y} = v_0 \sin \theta$ . The height of the ball at any time after it is thrown is given by

$$y(t) = y_0 + v_{0y}t + \frac{1}{2}gt^2$$

while the horizontal distance travelled by the ball after it is thrown is given by

$$x(t) = x_0 + v_{0x}t$$

- (a) Create a program to calculate  $y$  and  $x$  at any time if the initial velocity is 3 and angle  $\theta = 60^\circ$
  - (b) Find when does the ball achieve maximum height.
2. The  $n$ th Fibonacci number is defined by the following recursion equation

$$f(1) = 1; f(2) = 2; f(3) = 3; \dots$$
$$f(n) = f(n-1) + f(n-2)$$

Therefore,  $f(3) = f(1) + f(2) = 2 + 1 = 3$ , and so forth for higher numbers. Create a program to calculate and write out the Fibonacci number for  $n > 2$ .

## Recommended Reading

Chapman, S.J. 2002. *Matlab Programming for Engineers 2<sup>nd</sup>*, USA: Brooks/Cole.

Hahn, B.D; Valentine, D.T. 2007. *Essential Matlab for Engineers and Scientists 3<sup>rd</sup>*, Amsterdam: Elsevier

## Reporting

The results of this lab work must be reported to lecturer in the next meeting. Each student must report individually and present their results to get the feedbacks from the lecturer.