บทที่ 1

การใช้ Matlab ในการคำนวณทางวิศวกรรม

[24may2011 rev.1.2]

1.1 Matlab คืออะไร?

Matlab เป็นโปรแกรมสำหรับการคำนวณเชิงตัวเลข และ Visualization ที่มีประสิทธิภาพสูง ชื่อของโปรแกรม “Matlab” ย่อมาจากคำเต็มว่า MATrix LABoratory และเป็นเครื่องหมายการค้าของบริษัท MathWorks ซึ่งการทำงานภายในโปรแกรม Matlab อยู่บนพื้นฐานของการคำนวณทางเมตริกซ์ (Matrix Manipulation and Computation) เป็นแกนหลัก โปรแกรม Matlab สามารถทำงานแบบโต้ตอบ (interactive) ซึ่งคล้ายๆ กับ ภาษา Basic ในโปรแกรม QBasic และแบบ compiled mode คล้ายๆ กับภาษา C และ Pascal นอกจากนี้เราสามารถใช้ Matlab เป็นเสมือนเครื่องคิดเลข ทำการคำนวณทางคณิตศาสตร์ได้ทันที

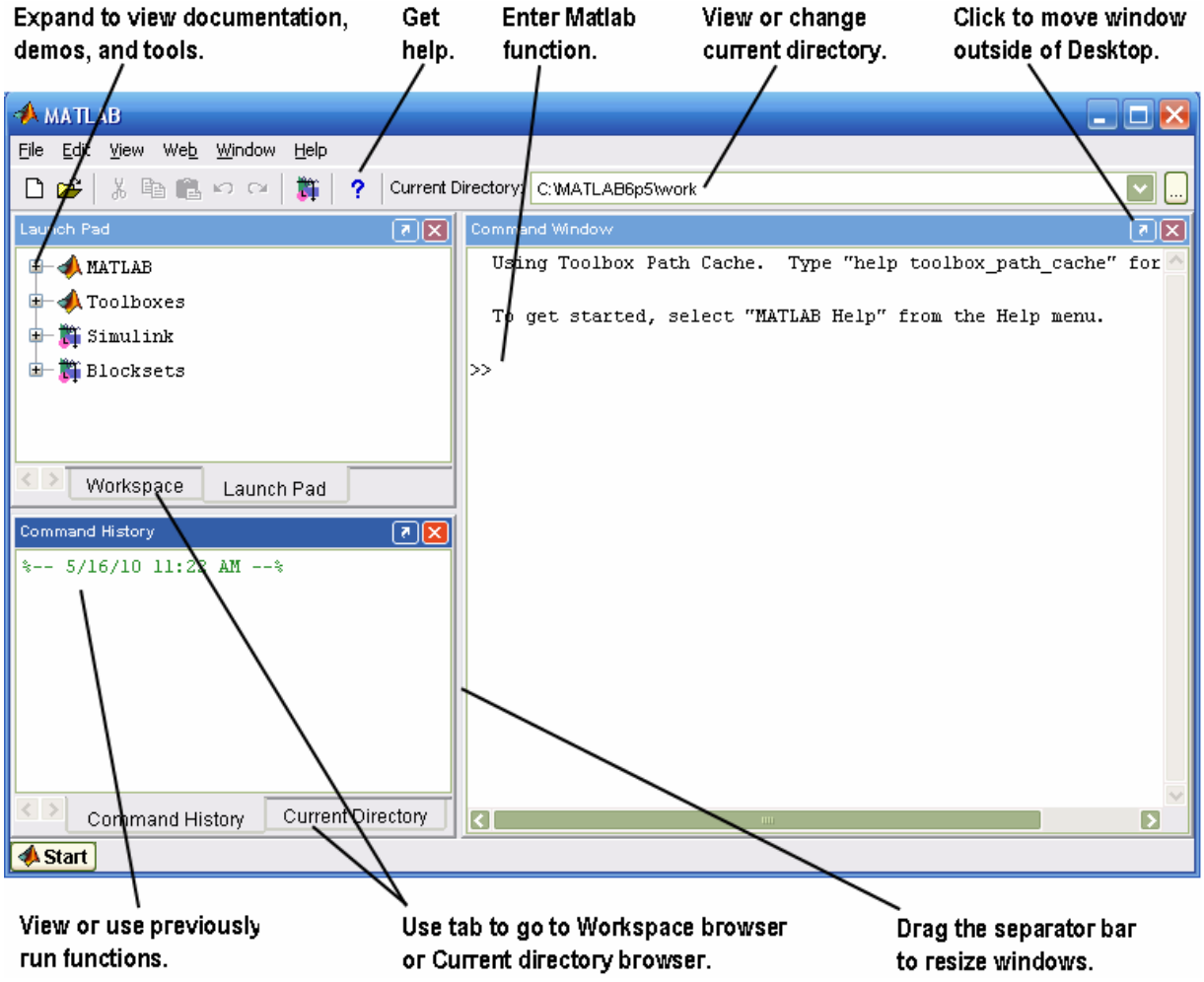
การโปรแกรม Matlab สามารถทำได้ง่าย เมื่อเทียบกับภาษาโปรแกรมอื่นๆ เราสามารถสร้างและกำหนดค่าตัวแปรได้ทันทีโดยไม่ต้องประกาศตัวแปรก่อน ซึ่งสะดวกในการเขียนโปรแกรม เพราะผู้ใช้ไม่ต้องกังวลในเรื่องโครงสร้างของภาษา เช่น ในเรื่องการต้องประกาศตัวแปรก่อนใช้งาน แต่มาสนใจอัลกอริทึมในการแก้โจทย์ปัญหาเป็นหลัก ข้อมูลไม่ว่าจะเป็นตัวเลข หรือตัวอักษร (strings) จะถูกจัดเก็บในรูปแบบของแถว และหลัก หรือ array ซึ่งก็คือ matrix นั่นเอง เช่น จำนวนสเกลล่าร์ (scalar) จะถูกแทนด้วยเมตริกซ์ขนาด 1x1 ข้อมูลที่เป็นเวกเตอร์จะถูกแทนที่ด้วยเมตริกซ์ที่มีเพียง 1 แถว ในกรณีที่เป็นเวกเตอร์แบบแถว (Row vector) หรือ ถูกแทนที่ด้วยเมตริกซ์ที่มีเพียง 1 หลัก ในกรณีที่เป็นเวกเตอร์แบบหลัก (Column vector) เป็นต้น การที่ Matlab ถูกออกแบบมาให้มีการทำงานภายในเช่นนี้ ทำให้การเขียนโปรแกรมแก้โจทย์ปัญหาที่มีลักษณะของ vector และ matrix เป็นเรื่องง่าย ตัวอย่างเช่น การแก้ระบบสมการเชิงเส้น ซึ่งระบบสมการ สามารถเขียนให้อยู่ในรูปแบบ $[A]\{x\} = \{b\}$ ได้ เป็นต้น (ดูรายละเอียดเพิ่มเติมใน บทที่ x) จะขอสรุปความสามารถของโปรแกรม Matlab เบื้องต้น ดังนี้

- Matlab เป็นโปรแกรมเพื่อการคำนวณและแสดงผลได้ทั้งตัวเลข และรูปภาพซึ่งมีประสิทธิภาพสูงสามารถทำการเขียนกราฟทั้ง 2 มิติ และ 3 มิติ ได้อย่างง่ายดาย และมีประสิทธิภาพ
- เราสามารถควบคุมการทำงานของ Matlab ด้วยชุดคำสั่ง (command line) และยังสามารถรวบรวมชุดคำสั่งเป็นโปรแกรม (script file) ได้ด้วย
- ลักษณะการเขียนโปรแกรมใน Matlab จะใกล้เคียงการเขียนสมการคณิตศาสตร์ที่เราคุ้นเคย จึงง่ายกว่าการเขียนโปรแกรมด้วยภาษาชั้นสูง เช่น ภาษา C, Pascal, Fortran และอื่นๆ
- Matlab มีฟังก์ชั่นสำเร็จรูป (built-in function) เพื่อทำงานเฉพาะทางมากมาย นอกจากนี้ผู้ใช้ยังสามารถเขียนฟังก์ชั่นขึ้นมาใหม่โดยใช้ประโยชน์จากฟังก์ชั่นที่มีอยู่เดิมได้เพื่อให้เหมาะสมกับงานของผู้ใช้แต่ละกลุ่ม สำหรับผู้ใช้ที่ต้องการใช้งานเฉพาะทางขั้นสูง เช่น งานด้าน Control, Image Processing, Artificial Neural Network หรืออื่นๆ Matlab ก็มี toolbox หรือชุด function พิเศษ เพื่อทำงานเฉพาะทางนั้นๆ ด้วย

- Matlab สามารถเชื่อมโยงหรือส่งข้อมูลแบบ Dynamic Link กับโปรแกรมอื่นๆ ได้ เช่น Excel หรือโปรแกรมที่เขียนขึ้นเองจากภาษา C หรือ Visual Basic ที่ร่วมทำงานอยู่บนระบบปฏิบัติการ Windows

1.2 เริ่มต้นสตาร์ทโปรแกรม Matlab

ดับเบิ้ลคลิ๊กที่ไอคอนของโปรแกรม Matlab บน desktop ก็จะทำการเปิดหน้าต่าง **Matlab Desktop** ขึ้นมา ซึ่งหน้าต่าง (window) นี้จะใช้เป็นตัวติดต่อสื่อสาร หรือ GUI (Graphic User Interface) สำหรับ Matlab เราอาจจะเห็นหน้าต่างย่อยหลายหน้าต่างบน Matlab desktop ดังรูปที่ 1.1 แต่ในบทนี้ขอให้สนใจในหน้าต่างที่ชื่อว่า Command window ซึ่งจะเป็นหน้าต่างที่ใช้มากที่สุด ที่หน้าต่างนี้จะมีเครื่องหมาย prompt, >> และต่อจากเครื่องหมาย prompt ก็จะเป็น cursor ที่กำลังกระพริบอยู่ แสดงว่า ตัวโปรแกรม Matlab พร้อมรับคำสั่งจากผู้ใช้



รูปที่ 1.1 หน้าต่างย่อยๆ บน Matlab Desktop

1.3 การใช้ Matlab เพื่อการคำนวณง่ายๆ

เราสามารถใช้โปรแกรม Matlab ทำการคำนวณคณิตศาสตร์ง่ายๆ เช่นเดียวกันกับเครื่องคิดเลข ให้ทดลองพิมพ์ แล้วกด “Enter”

```
>> 10+13
ans =
    23
```

Matlab จะทำการประมวลผล และให้คำตอบ (ans =) นอกจากนี้เรายังสามารถเก็บข้อมูลตัวเลขไว้ในตัวแปร หรือ variables เพื่อสะดวกในการใช้อ้างถึงภายหลัง และสามารถทำการคำนวณจากตัวแปรนั้นได้เลย เช่น

```
>> john = 10
john =
     10

>>sam = 13;

>> john + sam
ans =
    23
```

ตอนนี้จะสังเกตว่า Matlab ไม่ได้แสดง “sam = 13” เมื่อกด Enter นั้นก็เป็นเพราะเครื่องหมาย semicolon ที่ท้ายคำสั่งบอก Matlab ให้ทำการประมวลผล แต่ไม่ต้องแสดงผลออกมา

ข้อสังเกต `ans` เป็นตัวแปรหรือ variable ซึ่งเก็บค่าคำตอบที่เราคำนวณครั้งหลังสุดเอาไว้ คล้ายๆ กับการทำงานของเครื่องคิดเลขทางวิทยาศาสตร์ เช่น เครื่องคิดเลข Casio รุ่น fx-5500L ซึ่งสามารถอ้างถึงหรือนำคำตอบครั้งที่แล้วมาใช้คำนวณในครั้งต่อไปได้

ต่อไปจะลองหาค่า Sine ของมุม 90 องศา ให้พิมพ์ดังนี้

```
>> sin(pi/2)
ans =
     1
```

จะสังเกตว่า ฟังก์ชั่นตรีโกณมิติใน Matlab จะประมวลผลในหน่วยของเรเดียน (radian) ไม่ใช่องศา (degree) และ “pi” คือ ค่าคงที่ภายในของ Matlab มีค่า ~3.1416 เรเดียน

1.4 ความเข้าใจพื้นฐานในเรื่อง Vector(s) และ Matrix (Matrices)

เนื่องจากโปรแกรม Matlab จะทำงานกับข้อมูลที่เป็น matrix ซึ่งก็คือ ชุดของตัวเลข (an array of numbers) หรือสามารถเขียนอยู่ในรูปของตารางตัวเลข โดยมี m แถว และ n คอลัมน์ (m-row × n-column) เช่น ถ้า m=3 และ n=2 เมตริกซ์นี้จะมีมิติ เท่ากับ 3×2 ในภาษา Matlab การจะบอกว่าเป็นเมตริกซ์ ให้ใส่ไว้ภายใน Square brackets หรือ […] วิธีการใส่ตัวเลขในเมตริกซ์ก็จะใส่ทีละแถว โดยในแต่ละแถวจะแบ่งสมาชิกออกจากกันด้วย Space (หรือเครื่องหมาย Colon **“,”** หรือจุลภาค) และเมื่อจบแถวแล้ว ต้องการขึ้นแถวถัดไป (แถวที่ 2, 3, …) ให้ใส่เครื่องหมาย Semicolon หรือ **;** ดังตัวอย่าง

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

เวกเตอร์ คือ เมตริกซ์ที่มีจำนวนแถวหรือคอลัมน์ = 1 ฉะนั้นเวกเตอร์จะมี 2 ชนิด ก็คือ Row Vector (ตัวแปร B) และ Column Vector (ตัวแปร C) ดังจะแสดงตามลำดับ ดังนี้

```
>> B = [1 2 3]
B =
     1     2     3

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

จากความรู้เรื่องสมบัติ (Properties) ของเมตริกซ์ที่ได้เรียนมาแล้วทำให้ทราบว่า ถ้าทำการ transpose เมตริกซ์ B ก็จะได้เมตริกซ์ C หรือ C=B’ ในภาษา Matlab เราใช้ single quote (’) แทนกระบวนการ transpose เช่น

```
>> B'
ans =
     1
     2
     3
```

ตัวอย่างการนำเวกเตอร์ไปใช้งาน เช่น เราต้องการบันทึกอุณหภูมิตอนเที่ยงวันทุกวัน เป็นเวลา 1 สัปดาห์ จะแสดงใน Matlab ได้ดังนี้

```
>> T = [42.1 43.6 39.5 38.2 40.4 41.7 41.9]
T =
   42.1000   43.6000   39.5000   38.2000   40.4000   41.7000   41.9000
```

1.4.1 วิธีการอ้างถึงสมาชิกและกระทำต่อสมาชิกของเมตริกซ์

1. หากต้องการแสดงสมาชิกของเมตริกซ์ตัวหนึ่งตัวใด ให้ระบุตำแหน่งโดยรูปแบบ syntax ที่ใช้ คือ

`matrixname(row#,column#)`

จะสังเกตว่าในตอนนี้เราใช้ วงเล็บ หรือ Parentheses, (…) ไม่ได้ใช้ Square brackets, […]

```
>> T(1,3)  %Displays the element in the 1st row and 3rd column of T
ans =
      39.5000
```

ข้อสังเกต อะไรก็ตามที่อยู่หลังเครื่องหมาย % จะไม่ถูก Matlab ประมวลผล เรียกการ comment หรือคำอธิบายโปรแกรม ซึ่งจะมีประโยชน์เมื่อเราเขียนโปรแกรมด้วยภาษา Matlab โดยเฉพาะอย่างยิ่ง โปรแกรมที่มี code หลายบรรทัด

```
>> A(:,2) %Displays the 2nd column of A
```

```
>> A(3,:) %Displays the 3rd row of A
```

เราใช้เครื่องหมาย colon บอกให้ Matlab นำสมาชิก “ทุกๆ ตัว” ในแถว หรือ คอลัมน์ มาแสดง ตัวอย่างเช่น ถ้าเครื่องหมาย : อยู่ที่ตำแหน่ง row เช่น A(:,2) หมายความว่า ให้นำสมาชิกทุกๆ ตัวของคอลัมน์ที่ 2 มาแสดง ซึ่งก็คือสมาชิกของ A ในทุกๆ แถว เฉพาะในคอลัมน์ที่ 2

2. หากต้องการเมตริกซ์ย่อย จากเมตริกซ์ขนาดใหญ่ ให้ใช้ syntax ดังนี้

```
>> A(2:3,1:2)      %แสดงสมาชิกแถวที่ 2 ถึง 3 ในคอลัมน์แรกและคอลัมน์ที่ 2
ans =
     3     4
     5     6
```

ข้อสังเกต ในตอนนี้ เครื่องหมาย : จะแสดงถึงความต่อเนื่อง อาจใช้คำภาษาไทยว่า “ถึง” เช่น 2:3 หมายถึง จากแถว 2 ถึง แถว 3

3. หากต้องการสมาชิกเมตริกซ์ในแนวแทยงหลัก (major diagonal หรือ main diagonal) ให้ใช้คำสั่ง diag( ) คำสั่งนี้โดยส่วนใหญ่จะนิยมใช้กับ square matrix คือมีจำนวนแถวและคอลัมน์เท่ากัน

```
>> Adiag = diag(A)
Adiag =
     1
     4
```

4. ในการจะลบแถวหรือคอลัมน์ออกจากเมตริกซ์ใหญ่ให้ assign เมตริกซ์ว่าง หรือ square brackets, [] ไปยังแถวหรือคอลัมน์ที่ต้องการกำจัด ตัวอย่างเช่น

```
>> A(:,2)=[]
A =
     1
     3
     5
```

5. ในการจะรวมเมตริกซ์ย่อยเป็น เมตริกซ์ใหญ่ (Concatenation) สามารถกระทำได้ดังนี้

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

>>E=[A A+12; A*3 A/2]
B =
    3.0000   7.0000    4.0000   15.0000   19.0000   16.0000
    5.0000   9.0000    2.0000   17.0000   21.0000   14.0000
    4.0000   6.0000    1.0000   16.0000   18.0000   13.0000
    9.0000  21.0000   12.0000    1.5000    3.5000    2.0000
   15.0000  27.0000    6.0000    2.5000    4.5000    1.0000
   12.0000  18.0000    3.0000    2.0000    3.0000    0.5000
```

1.4.2 เมตริกซ์ชนิดพิเศษ

ในการศึกษาเกี่ยวกับเมตริกซ์ ในเบื้องต้นนี้ จะมีเมตริกซ์ชนิดพิเศษ 3 ชนิด ที่ควรรู้จัก ก็คือ zero matrix, the identity matrix และ matrixes/vectors of ones เราใช้คำสั่งต่อไปนี้ในการสร้างเมตริกซ์เหล่านี้

```
>> F=zeros(3,3)  %displays a 3x3 matrix of zeros

>> G=eye(4,4)    %displays a 4x4 identity matrix

>> H=ones(3,1)   %displays a 3x1 matrix of ones
```

1.5 การใช้ฟังก์ชั่นอย่างง่ายและการเขียนกราฟ

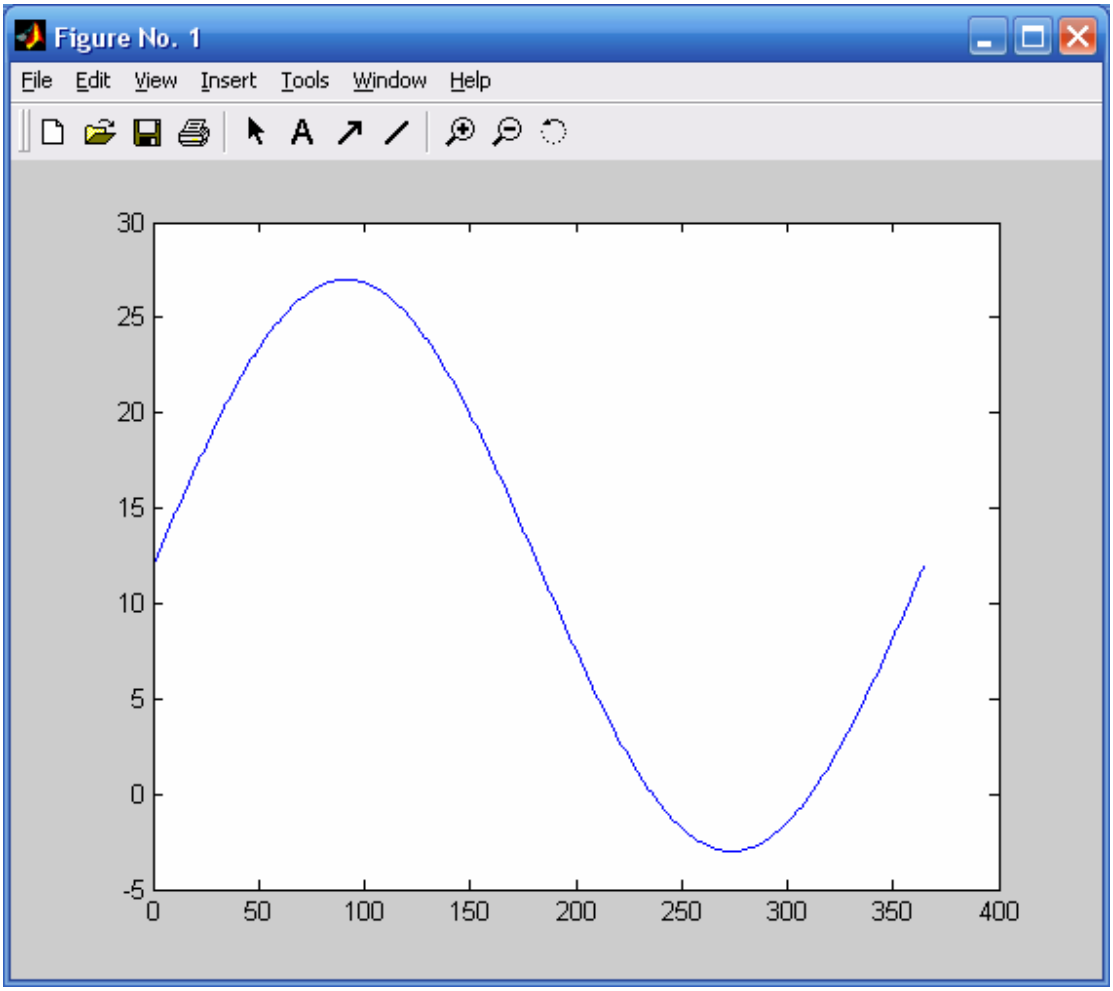
โปรแกรม Matlab มี built-in functions มากมาย ตัวอย่างเช่น ฟังก์ชั่นทางตรีโกณมิติ (trigonometry functions) สมมติว่า เรากำหนดให้รูปแบบการเปลี่ยนแปลงอุณหภูมิในรอบ 1 ปี เป็นฟังก์ชั่น Sine โดยในช่วงเวลา 1 ปี หรือ 365 วัน เราบันทึกข้อมูล ทุกๆ วัน กำหนดให้ ตัวแปร t เป็นลำดับเวลาการบันทึกข้อมูล จะได้ 365 ครั้ง

```
>> t=1:1:365

>> temp=15*sin(2*pi*t/365)+12
```

Syntax แรก จากข้างบน หมายความว่า ให้สร้าง Row vector ที่มีสมาชิกเริ่มตั้งแต่ 1 ถึง 365 โดยมี step การเพิ่มขึ้นทีละ +1 ซึ่ง syntax ข้างบนนี้ มีค่าเทียบเท่าเท่ากับ t=1:365 เพราะหากไม่ระบุ step ค่า default มีค่าเท่ากับ +1 ส่วน syntax ที่สองเป็นการให้ค่าอุณหภูมิ ซึ่งเรากำหนดไว้ว่าเป็นฟังก์ชั่น Sine ที่ขึ้นกับลำดับเวลา t ซึ่ง t มีอยู่ 365 ค่า ฉะนั้นก็จะได้ค่าอุณหภูมิ temp จำนวน 365 ค่าด้วย (ให้พิจารณาผลลัพธ์ที่ได้จาก Matlab ประกอบ) เราสามารถดูกราฟผลลัพธ์ ดังรูปที่ 1.2 โดยใช้ความสามารถทางกราฟิกของ Matlab ได้ดังนี้

```
>> plot(t,temp);
```

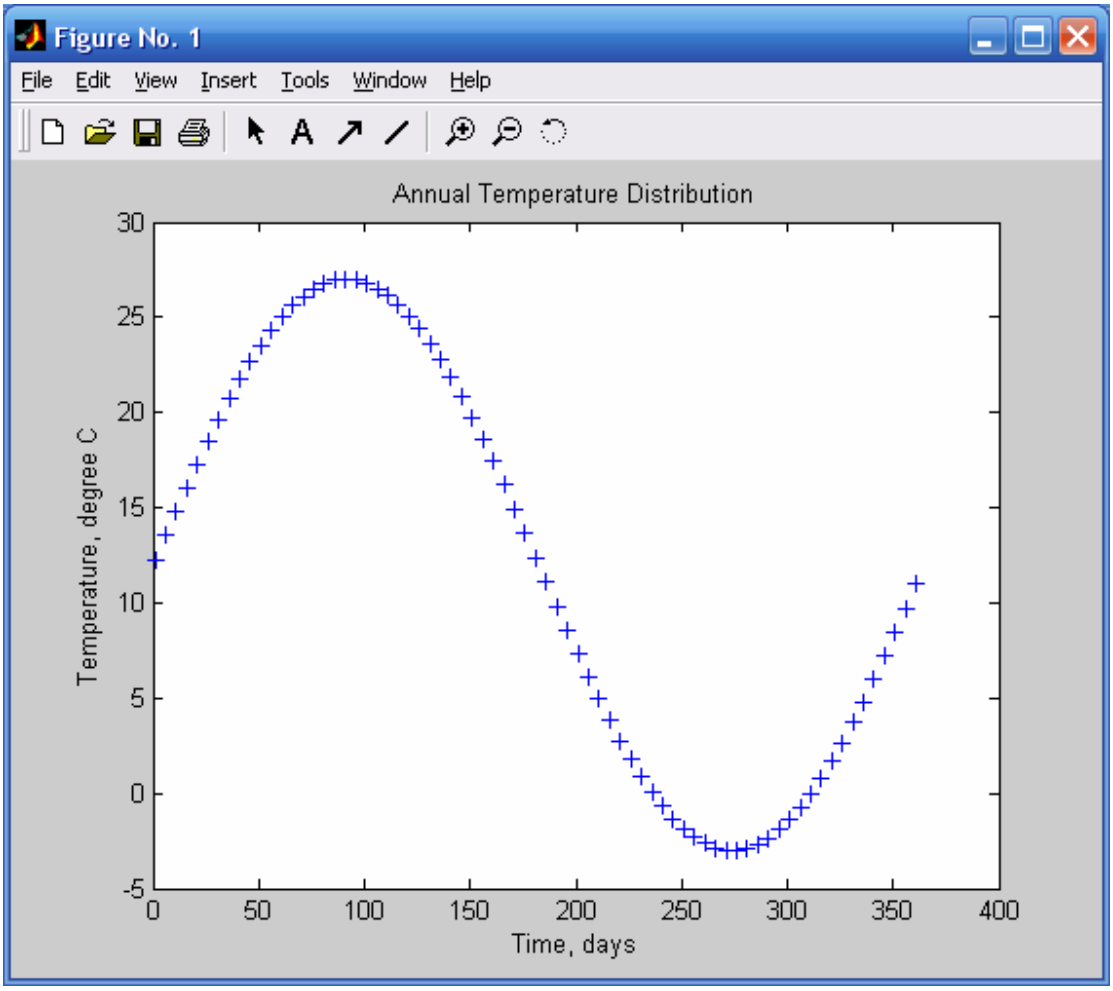


รูปที่ 1.2 กราฟของ sine โดยการพลอตแบบต่อเนื่อง (365 จุด)

ถ้าเราต้องการลดจำนวนข้อมูล ก็สามารถทำได้โดยการเปลี่ยนค่าของ step เช่น step = 5 และในแต่ละคำสั่งของ Matlab ก็มีตัวเลือก หรือ options อื่นๆ อีก ในที่นี้ แทนที่จะวาดกราฟเป็นเส้นต่อเนื่อง เราจะวาดกราฟเป็นจุด แสดงด้วยเครื่องหมาย '+' และทำการใส่ชื่อแกน x และแกน y จะขั้นตอนการทำและได้ผลลัพธ์ ดังรูปที่ 1.3

```
>> t=1:5:365;
```

```
>> temp=15*sin(2*pi*t/365)+12;
>> plot(t,temp,'+');
>> xlabel('Time, days');
>> ylabel('Temperature, degree C');
>> title('Annual Temperature Distribution');
```



รูปที่ 1.3 กราฟของ sine โดยการพลอตเป็นจุด แบบไม่ต่อเนื่อง

เราสามารถดู รายละเอียดปลีกย่อยของ built-in function ด้วยคำสั่ง help โดยการพิมพ์

```
>>help plot
```

หรือในบางครั้งเราไม่ทราบคำสั่งที่เฉพาะเจาะจงหรือชื่อ function ใน Matlab เราอาจใช้คำสั่ง lookfor ตามด้วย keyword หรือ topic เพื่อค้นหาในเบื้องต้นก่อน เมื่อทราบชื่อ function ที่ต้องการแล้ว ก็สามารถใช้คำสั่ง help อีกที เช่น

```
>>lookfor interpolation
```

1.6 การนำข้อมูลเข้าสู่ Matlab

เราสามารถนำข้อมูลเข้าสู่โปรแกรม Matlab ได้โดยไม่ต้องพิมพ์ทีละบรรทัด ซึ่งได้กล่าวถึงความสามารถของโปรแกรม Matlab ที่สามารถติดต่อกับโปรแกรมอื่นๆ ในระบบปฏิบัติการ Windows ตัวอย่างที่ง่ายที่สุด คือ ความสามารถในการนำข้อมูลจาก ASCI text file เข้าสู่ Matlab ได้โดยตรง โดยไม่ต้องเขียนชุดคำสั่งพิเศษเพิ่มเติม เช่นเรามีข้อมูล

```
0.0700 0.02
0.1400 0.02
0.2100 0.02
0.2700 0.02
0.3400 0.02
0.4100 0.02
0.4800 0.02
0.5500 0.02
0.6200 0.02
0.6800 0.000
0.7500 0.0000
0.8200 0.0000
0.8900 0.02
0.9600 0.1
1.0300 0.44
1.0900 0.78
1.1600 0.66
1.2300 0.3
1.3000 0.14
1.3700 0.06
1.4400 0.02
1.5000 0.02
```

สมมติว่าข้อมูลอยู่ในไฟล์ชื่อ **“bt.dat”** สร้างโดยโปรแกรม Notepad ซึ่งเป็น text editor ตัวหนึ่งใน Microsoft Windows เราใช้คำสั่ง **load** เพื่อนำข้อมูลเข้าสู่โปรแกรม Matlab

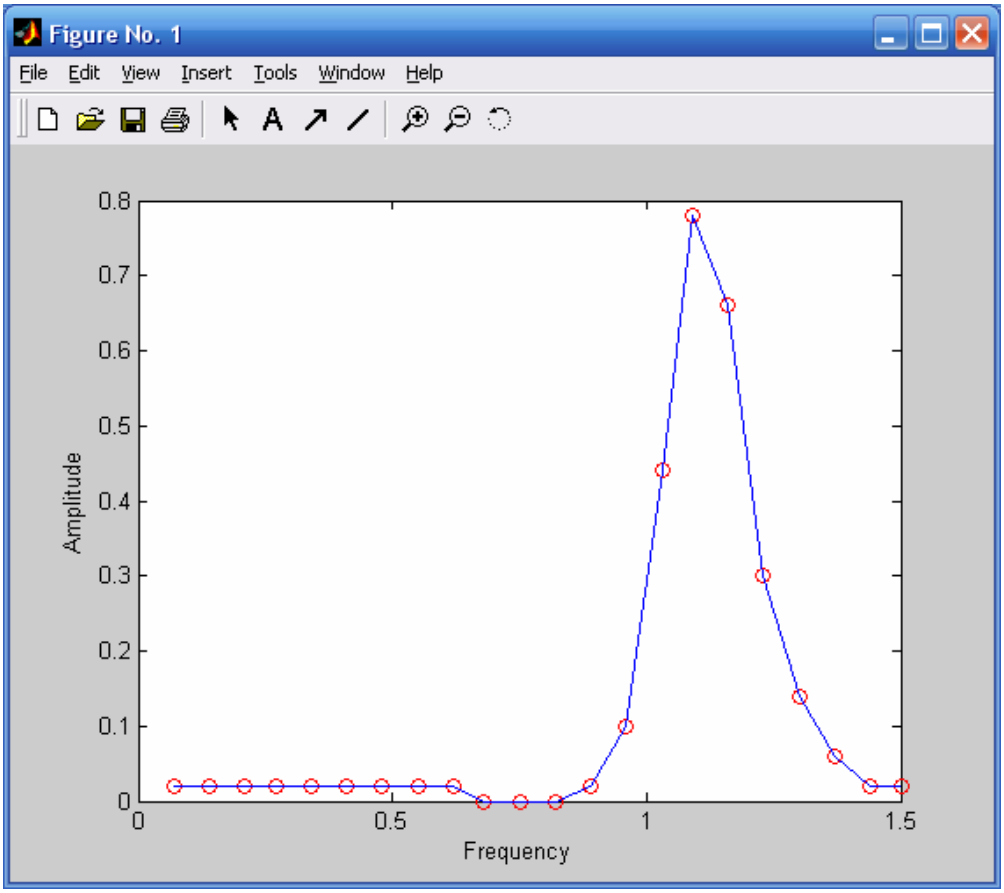
```
>>load bt.dat
```

ข้อสังเกต ชื่อไฟล์ข้อมูลที่นำเข้าด้วยคำสั่ง load ควรกำหนดให้มีนามสกุล ซึ่งอาจเป็นนามสกุลอื่นๆ ก็ได้ เช่น “bt.q” หรือ “bt.xxx” ถ้าระบุเฉพาะชื่อไฟล์ เช่น

```
>>load bt
```

จะหมายถึง load ไฟล์ที่ชื่อ “bt.mat” ซึ่ง .mat เป็นนามสกุล default ที่ Matlab รู้จัก เมื่อถึงขั้นตอนนี้ เราจะได้ ตัวแปรชื่อ bt ซึ่งเป็น matrix ขนาด 22×2 ขั้นตอนต่อไป เราจะทำการแยกข้อมูลทีละคอลัมน์จากตัวแปร bt ไปใส่ตัวแปรใหม่ชื่อ **x_freq** และ **y_ampli** ตามลำดับ แล้วทำการพลอตกราฟ (รูปที่ 1.4) ดังนี้

```
>>x_freq=bt(:,1)
>>y_amplitude=bt(:,2)
>>plot(x_freq,y_ampli,'or',x_freq,y_ampli,'b')
>>xlabel('Frequency')
>>ylabel('Amplitude')
```



รูปที่ 1.4 กราฟที่ได้จากการพลอตข้อมูลใน “bt.dat”

1.7 การคำนวณทางเมตริกซ์ (Matrix Operations)

สิ่งสำคัญในการคำนวณทางคณิตศาสตร์ (Mathematical operations) สำหรับเมตริกซ์ เช่น บวก ลบ คูณ หาร และอื่นๆ ก็คือ เมตริกซ์ทั้งสองต้องสอดคล้องกัน (conformable) ตามกฎของแต่ละการคำนวณ สัญลักษณ์ของการคำนวณเมตริกซ์พื้นฐาน มีดังต่อไปนี้

`+` แสดง การบวก (addition) —e.g., `>> A+B`

`-` แสดง การลบ (subtraction) —e.g., `>> A-B`

`*` แสดง การคูณ (multiplication) —e.g., `>> A*B`

`/ or \` แสดง การหาร (division) —e.g., `>> A/B` or `A\B` for “right” division or “left” division ตามลำดับ

`inv( )` แสดง การอินเวอร์สเมตริกซ์ (matrix inversion) —e.g., `>> Cinv = inv(C)`

`'` แสดง การทรานส์โพส (transposition) —e.g., `>> C'`

`det( )` แสดง การหาดีเทอร์มิแนนท์ (determinant) —e.g., `>> det(C)`

1.7.1 การบวก การลบ และอื่นๆ

เมตริกซ์ที่จะทำการบวก หรือลบกันได้ จะต้องมีขนาดหรือมิติเท่ากัน และจะทำการบวกหรือลบที่ระดับสมาชิก ตำแหน่งต่อตำแหน่ง ดังตัวอย่าง

```
>> A=[1 2 3; 4 5 6]
A =
      1  2  3
      4  5  6
>> B=[7 8 9; 10 11 12]
B =
       7  8  9
      10 11 12
>> A+B
ans =
       8 10 12
      14 16 18
>> A-B
ans =
      -6 -6 -6
      -6 -6 -6
```

ในการคูณเมตริกซ์ด้วยสเกลล่าร์ หรือค่าคงที่ ก็ทำได้ง่าย เช่น

```
>> 2*A
ans =
      2  4  6
      8 10 12
```

สำหรับฟังก์ชั่นทางคณิตศาสตร์พื้นฐานที่มีใน Matlab เช่น sin, cos, exp, log, sqrt และอื่นๆ ก็สามารถนำมาใช้กับเมตริกซ์ได้ทันที แต่จำไว้ว่าวิธีการคำนวณของฟังก์ชั่นจำพวกนี้จะกระทำกับสมาชิกแต่ละตัว (on each element) ตัวอย่างเช่น

```
>> sqrt(A)
ans =
      1.0000 1.4142 1.7321
      2.0000 2.2361 2.4495
```

ต่อไปจะขอสรุปคุณสมบัติของการบวก-ลบเมตริกซ์ และการคูณเมตริกซ์ด้วยสเกลล่าร์ เพื่อประโยชน์ในการเข้าใจที่มาของสูตรต่างๆ ด้วยวิธีพิสูจน์ ดังตารางที่ 1.1

ตารางที่ 1.1

Properties of Matrix Addition and Scalar Multiplication	
1. $A + B = B + A$	
2. $A + (B + C) = (A + B) + C$	
3. $A + 0 = 0 + A = A$	"*0*" is the zero matrix
4. $c(A + B) = cA + cB$	
5. $(a + b)C = aC + bC$	
6. $a(bC) = (ab)C$	

1.7.2 การคูณเมตริกซ์

เมตริกซ์สองตัวจะคูณกันได้ ก็ต่อเมื่อมี "มิติภายใน" (inner dimensions) เท่ากัน เช่น $CD = C_{m\,x\,n} \times D_{i\,x\,j}$ ในที่นี่ มิติภายในก็คือ $n = i$ และผลคูณที่ได้จะมีมิติเท่ากับ $m \times j$ (หรือ outer dimensions)

จะขอสาธิตขั้นตอนการคูณเมตริกซ์ $A_{2\,x\,3}$ กับ $B_{3\,x\,2}$ ในรูปสัญลักษณ์ และตามด้วยตัวอย่างใน Matlab ดังนี้

$$A = \begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \end{bmatrix}, B = \begin{bmatrix} b_1 & b_4 \\ b_2 & b_5 \\ b_3 & b_6 \end{bmatrix}$$

$$A \times B = \begin{bmatrix} (a_1b_1 + a_2b_2 + a_3b_3) & (a_1b_4 + a_2b_5 + a_3b_6) \\ (a_4b_1 + a_5b_2 + a_6b_3) & (a_4b_4 + a_5b_5 + a_6b_6) \end{bmatrix}$$

```
>> A*B
??? Error using ==> *
Inner matrix dimensions must agree.
```

จะเห็นว่าเมตริกซ์ $A_{2\,x\,3}$ และ $B_{2\,x\,3}$ ไม่สามารถคูณกันได้ โปรแกรม Matlab จึงฟ้อง Error ออกมา เราจึงทำการปรับมิติของเมตริกซ์ B โดยการทำ transpose เมตริกซ์ B จากนั้นจึงนำไปคูณกับเมตริกซ์ A ได้ ดังตัวอย่าง

```
>> BT=B'
BT =
      7 10
      8 11
      9 12
>> A*BT
ans =
       50  68
      122 167
```

ตอนนี้จะขอสรุปคุณสมบัติของการคูณเมตริกซ์ ตามตารางที่ 1.2

ตารางที่ 1.2

Properties of Matrix Multiplication	
1. $A(BC) = (AB)C$	
2. $A(B + C) = AB + AC$	
3. $(A + B)C = AC + BC$	
4. $AI_n = I_nA = A$	"I_n" is the identity matrix
5. $c(AB) = (cA)B = A(cB)$	

หมายเหตุ โปรดจำไว้ว่าโดยทั่วไปแล้ว $AB \neq BA$ เพราะการคูณเมตริกซ์ไม่สามารถสลับตำแหน่งได้ (not communicative) แนวคิดตรงนี้สำคัญ เนื่องจากจะนำไปใช้อธิบายในหัวข้อต่อๆ ไปด้วย

1.7.3 การคูณเมตริกซ์ในระดับสมาชิก (Multiplication on an element-by-element basis)

ในการคูณเมตริกซ์สองตัว บางครั้งเราเพียงสนใจที่จะคูณสมาชิกของเมตริกซ์ตำแหน่งต่อตำแหน่ง เช่น เราต้องการค่ากำลังสองของสมาชิกเมตริกซ์ A เราไม่สามารถใช้ A × A หรือ A^2 เพราะจะเป็นการคูณเมตริกซ์ ซึ่งก็ต้อง conform กับกฎของการคูณเมตริกซ์ จึงเป็นที่มาของการคำนวณในระดับสมาชิก ในภาษา Matlab เรียกว่า dot operation (ห้ามนำไปสับสนกับ dot product ของเวกเตอร์)

```
>> A^2      %Performs the usual matrix multiplication A*A
>> A.^2     %Squares each element of the matrix A
>> A.*B
ans =
     7    16    27
    40    55    72
```

1.7.4 การ transpose เมตริกซ์

ในการ transpose เมตริกซ์ A, เขียนแทนด้วย A^T หรือ A' หมายความว่า เมตริกซ์ที่ได้ จะมีคอลัมน์ที่เกิดจากแถวของเมตริกซ์ตั้งต้น ตารางที่ 1.3 แสดงคุณสมบัติของการ transpose

ตารางที่ 1.3

Properties of Transpose	
1. $(A + B)' = A' + B'$	
2. $(cA)' = cA'$	
3. $(AB)' = B'A'$	***
4. $(A')' = A$	

หากทำการ transpose เมตริกซ์จตุรัส (square matrix) ใดๆ แล้วได้เท่ากับเมตริกซ์ตั้งต้น เราเรียกเมตริกซ์นั้นว่า "Symmetric matrix" หรือเมตริกซ์สมมาตร ข้อสังเกตง่ายๆ สำหรับ symmetric matrix ก็คือ สมาชิกของเมตริกซ์จะสมมาตรแบบ mirror ที่แกน main diagonal line ต่อไปนี้เป็นตัวอย่างเมตริกซ์สมมาตร

$$\begin{bmatrix} 2 & 5 \\ 5 & -4 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1 & -4 \\ 1 & 7 & 8 \\ -4 & 8 & 3 \end{bmatrix}$$

1.7.5 การอินเวอร์สเมตริกซ์ (The Inverse of a Matrix)

แนวคิดในการอินเวอร์สเมตริกซ์ ต่อยอดมาจากการคูณส่วนกลับของจำนวนจริง (multiplicative inverse of a real number) เช่น 1/4 (:= b) เป็นส่วนกลับหรือค่า inverse ของ 4 (:= a) จะได้ว่า (1/4)4 = 4(1/4) = 1 หรือ

$$ab = 1 \quad \text{และ} \quad ba = 1$$

ตอนนี้ กำหนดให้ A และ B เป็นเมตริกซ์จตุรัส และ I_n เป็น identity matrix ในทำนองเดียวกัน เราได้ว่า

$$AB = BA = I_n$$

ถ้าสมการข้างบนเป็นจริง เราเรียกว่าเมตริกซ์ A ที่สามารถหาค่าอินเวอร์สได้ว่าเป็น Nonsingular matrix เรียก B ว่าเป็นอินเวอร์สของ A มีสัญลักษณ์ คือ A^{-1} จะได้ว่า

$$AA^{-1} = A^{-1}A = I_n \qquad \text{(eqn. 1.1)}$$

ถ้าเมตริกซ์ A ไม่สามารถหาค่าอินเวอร์สได้ เรียก A ว่าเป็น Singular matrix

วิธีการหาค่า inverse matrix มีอยู่ 2 วิธี คือ (1) วิธี Cofactor และ (2) วิธี Row Reduction ซึ่งในขณะนี้เราจะศึกษาเฉพาะวิธีที่สองซึ่งจะเข้าใจได้ง่ายกว่า ส่วนวิธี Cofactor ผู้ที่สนใจสามารถดูรายละเอียดได้จากเอกสารอ้างอิงอื่นๆ ในหัวข้อ Matrix Algebra

- วิธีการหา Inverse Matrix ด้วยวิธี Row Reduction

วิธีนี้เป็นการหาค่า inverse matrix ของ nonsingular matrix A โดยการคำนวณมือ ด้วยกระบวนการ row reduction ของเมตริกซ์ A และ identity matrix, I_n ไปพร้อมๆ กัน เช่น ต้องการหาค่า inverse ของเมตริกซ์ A

$$A=\begin{bmatrix}4 & 10\\ 10 & 30\end{bmatrix}$$

เริ่มต้นกระบวนการโดยนำ identity matrix, I_n มาวางคู่กับเมตริกซ์ที่จะทำการหาค่า inverse จากนั้นเป้าหมายของเราก็คือ จะทำการเปลี่ยนเมตริกซ์ A ไปเป็น identity matrix ด้วยวิธี row reduction (หรือวิธี Gauss-Jordan ซึ่งจะกล่าวถึงในบทต่อไป) ในขณะเดียวกัน I_n ก็จะกลายเป็นค่า inverse ของเมตริกซ์ A

1.	$\begin{bmatrix}4 & 10\\ 10 & 30\end{bmatrix}\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$	- วาง [A] คู่กับ [I_n]
2.	$\begin{bmatrix}1 & 2.5\\ 10 & 30\end{bmatrix}\begin{bmatrix}0.25 & 0\\ 0 & 1\end{bmatrix}$	- ทำให้ตัวแรกของ Row1 มีค่าเป็น 1 โดยการหารทั้งแถวที่ 1 ด้วย 4 หรือ (Row1)/4
3.	$\begin{bmatrix}1 & 2.5\\ 0 & 5\end{bmatrix}\begin{bmatrix}0.25 & 0\\ -2.5 & 1\end{bmatrix}$	- นำ Row1 มาขจัด แถวที่เหลือ เพื่อให้ Column1 ของทุกแถวเป็น 0 โดย Row2 – (10 x Row1)
4.	$\begin{bmatrix}1 & 2.5\\ 0 & 1\end{bmatrix}\begin{bmatrix}0.25 & 0\\ -0.5 & 0.2\end{bmatrix}$	- ทำให้ตัวสุดท้าย ของ Row สุดท้าย มีค่าเป็น 1 โดย (Row2)/5
5.	$\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}1.5 & -0.5\\ -0.5 & 0.2\end{bmatrix}$	- ต่อไปก็จะทำให้ upper triangular เป็น 0 โดย Row1 – (2.5 x Row2) ซึ่งจะได้ค่า [I_n][A]$^{-1}$

เราสามารถตรวจสอบความถูกต้องโดยการนำ [A] คูณกับ [A]$^{-1}$ จะได้ค่า I_n

$$\begin{bmatrix}4 & 10\\ 10 & 30\end{bmatrix}\begin{bmatrix}1.5 & -0.5\\ -0.5 & 0.2\end{bmatrix}=\begin{bmatrix}6+(-5) & -2+2\\ 15-15 & -5+6\end{bmatrix}=\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$$

ในการหาค่า inverse matrix ของเมตริกซ์ขนาด 3 x 3 หรือใหญ่กว่า ก็จะทำในลักษณะเดียวกัน จะเห็นว่าขั้นตอนการทำ row reduction ก็จะยาวขึ้น และซับซ้อนมากขึ้น ดังนั้นในทางปฏิบัติ ส่วนใหญ่การหาค่า inverse matrixจะกระทำในโปรแกรมคอมพิวเตอร์ ด้วยวิธีเชิงตัวเลขมากกว่า โดยใน Matlab จะใช้คำสั่ง **inv(…)** ถ้าเมตริกซ์ที่นำมาคำนวณเป็น singular matrix หรือเป็นเมตริกซ์ที่ไม่สามารถหาค่า inverse ได้ โปรแกรม Matlab ก็จะแสดงข้อความบอกความผิดพลาด ตัวอย่างเช่น

```
>> A=[1 4;8 9];
>> inv(A)
ans =
      -0.3913   0.1739
       0.3478  -0.0435
>> B=[1 2;2 4];
>> inv(B)
Warning: Matrix is singular to working precision.
(Type "warning off MATLAB:singularMatrix" to suppress this warning.)
ans =
   Inf   Inf
   Inf   Inf
```

1.7.5 การหารเมตริกซ์

ตามความเป็นจริงแล้ว matrix ไม่มีคุณสมบัติของการหาร แต่พอจะเทียบเคียงการหารได้กับการคูณด้วยค่าอินเวอร์ส หรืออธิบายในเชิงสัญลักษณ์ ก็คือ $X/Y = X * 1/Y = X * Y^{-1}$ ดังนั้นกระบวนการ "/ Y" ก็เทียบได้กับ "* Y^{-1}" โปรแกรม Matlab ได้สร้าง function การหารเมตริกซ์ขึ้นมา 2 ลักษณะ คือ

- เครื่องหมาย / (อ่านว่า slash) ใน MATLAB จะเรียก right matrix division

หมายถึง $A / B = A * B^{-1}$ -คำอธิบายเพิ่มเติม ใช้เฉพาะใน Matlab บางครั้งตัวหารอาจเป็นค่าคงที่ ก็จะเป็นการนำค่าคงที่นั้นไปหารสมาชิกของ A ทุกตัว คล้ายๆ เป็นการ scaling เมตริกซ์ A

- เครื่องหมาย \ (อ่านว่า back slash) ใน MATLAB จะเรียก left matrix division

หมายถึง $A \backslash B = A^{-1} * B$ -คำอธิบาย ถ้าพิจารณาโดยอ่านจากหลังไปหน้า ว่าเป็น B หารด้วย A ก็เทียบได้กับ $B * A^{-1}$ แต่เนื่องจากการคูณเมตริกซ์ไม่สามารถสลับตำแหน่ง จึงต้องคงลำดับเอาไว้ จึงกลายเป็น B คูณกับ A^{-1} ที่อยู่ข้างหน้า

เครื่องหมายหารแบบ left division นี้ ประดิษฐ์ขึ้นมาเพื่อสะดวกในการแก้ระบบสมการเชิงเส้น (system of linear equations) โดยเฉพาะ ลองพิจารณา

$[A]\{x\} = \{b\}$ หรือในรูปฟอร์มง่ายๆ เป็น $Ax = b$

เราต้องการหาผลเฉลย หรือ solutions ของระบบสมการ ซึ่งก็คือค่า unknown {x} หรือ x ดังนั้นจึงทำการกำจัดสัมประสิทธิ์ A โดยการคูณด้วย A^{-1} เข้าไปข้างหน้าของทั้งสองด้านของสมการ

$A^{-1}Ax = A^{-1}b$

$Ix = A^{-1}b$ -identity matrix, I คูณกับเมตริกซ์หรือเวกเตอร์ใดๆ ก็ได้ตัวเดิม จะได้ว่า

$x = A^{-1}b$ (eqn. 1.2)

ฉะนั้นค่า $A^{-1}b$ ก็คือ solutions ของระบบสมการเชิงเส้น ใน Matlab เราสามารถคำนวณค่า $A^{-1}b$ โดยใช้ คำสั่ง **A\b** สำหรับตัวอย่างจะอยู่ในหัวข้อการหาค่าคำตอบของระบบสมการเชิงเส้น ซึ่งจะกล่าวถึงในบทต่อไป

1.7.6 การหา determinant ของเมตริกซ์

ค่า determinant ของเมตริกซ์จะเป็นค่าตัวเลข หรือค่า scalar ที่สังเคราะห์มาจากเมตริกซ์หนึ่ง ในขั้นตอนการหา determinant จะทำได้เฉพาะเมตริกซ์จตุรัส (square matrix) เท่านั้น ค่า determinant ของเมตริกซ์ A เขียนแทนด้วยสัญลักษณ์ det A หรือ | A | สมการที่ 1.3 แสดงวิธีการหาค่า determinant ของเมตริกซ์ขนาด 2 x 2

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

$$| A | = a_{11}a_{22} - a_{21}a_{12} \qquad \text{(eqn. 1.3)}$$

สำหรับเมตริกซ์ขนาด 3 x 3 มีสูตรการหาค่า determinant ดังสมการ (1.4)

$$B = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix}$$

$$| B | = b_{11}\begin{vmatrix} b_{22} & b_{23} \\ b_{32} & b_{33} \end{vmatrix} - b_{12}\begin{vmatrix} b_{21} & b_{23} \\ b_{31} & b_{33} \end{vmatrix} + b_{13}\begin{vmatrix} b_{21} & b_{22} \\ b_{31} & b_{32} \end{vmatrix} \qquad \text{(eqn. 1.4)}$$

ในกรณีที่เมตริกซ์มีขนาดใหญ่กว่านี้ เช่น 4 x 4 จะเป็นการสะดวกกว่าที่เราจะใช้สูตรทั่วไปในการหาค่า determinant ดังสมการ...

คำสั่งในการหาค่า determinant ของ square matrix a ใน Matlab ก็คือ **det(…)** สำหรับเมตริกซ์ที่มี determinant เท่ากับศูนย์ แสดงว่าเมตริกซ์นั้นเป็น singular matrix ตัวอย่างการหา determinant โดย Matlab

```
>> a=[1 2 3; 2 7 6; 5 4 8];
>> det(a)
ans =
      -21
```

1.8 การเขียนโปรแกรมใน Matlab

คราวนี้มาถึงหัวข้อสำคัญของ Matlab สำหรับการศึกษาวิชาทางด้าน Numerical Method คือการเขียนโปรแกรม แม้ว่าการทำงานกับ Matlab โดยปกติผู้ใช้สามารถทำการพิมพ์คำสั่งทีละคำสั่งใน Command window แล้วดูผลลัพธ์ได้ทันที อย่างไรก็ตามการคำนวณบางอย่างจำเป็นต้องใช้ชุดคำสั่งซ้ำๆ ซึ่งจะไม่สะดวกนัก ที่ผู้ใช้จะพิมพ์คำสั่งเหล่านี้ใหม่ทุกครั้ง ดังนั้น Matlab จึงอนุญาตให้ผู้ใช้สามารถเขียนโปรแกรมที่ประกอบด้วยชุดคำสั่งขึ้นมาเองได้ การเรียกใช้งานโปรแกรมที่ผู้ใช้เขียนขึ้นเองนั้น ในเบื้องต้นนี้ เราจะเรียกใช้โดยผ่าน Command window ของ Matlab เราสามารถแบ่งการโปรแกรมใน Matlab ออกเป็น 2 ประเภท ดังนี้ คือ

1.8.1 การเขียนสคริปต์ (Script M-files)

การเขียนสคริปต์ คือการรวมเอาคำสั่งต่างๆ ของ Matlab ที่เกี่ยวข้องกับการคำนวณนั้นๆ ไว้ในไฟล์ที่มีนามสกุล (*.m) หรือ M-file ซึ่งมีโครงสร้างภายในเป็น ASCII text file เราสามารถเรียก M-file ขึ้นมาแก้ไขชุดคำสั่งที่ได้บันทึกไว้ โดยผู้ใช้ไม่จำเป็นต้องป้อนคำสั่งใหม่ที่ละคำสั่งอีกต่อไป เมื่อผู้ใช้แก้ไขตัวแปรหรือคำสั่งต่างๆ ใน (text) editor แล้ว สามารถสั่งให้โปรแกรมทำงานใหม่ เพื่อทดสอบการทำงานของโปรแกรมได้ทันที ตัวอย่างง่ายๆ เช่น เราต้องการเขียนสคริปต์เพื่อวาดกราฟของ data ที่อยู่มีในไฟล์ชื่อ “bt.dat” เราจะกระทำการพิมพ์ข้อความเหล่านี้ใน text editor เช่น Notepad หรืออาจใช้ M-file editor ที่มากับ Matlab ก็ได้ และบันทึกเป็นไฟล์ชื่อ “plotbt.m” ดังนี้

```
%% file name: plotbt.m
%% M-file for plotting data in bt.dat
%%
load bt.dat
x_freq=bt(:,1)
y_ampli=bt(:,2)
plot(x_freq,y_ampli,'or',x_freq,y_ampli,'b')
xlabel('Frequency')
ylabel('Amplitude')
```

เมื่อได้ไฟล์ชื่อ plotbt.m แล้ว ตรวจสอบการมีอยู่ของไฟล์โดยใช้คำสั่ง **dir** หรือ **ls** ที่ Matlab command prompt จากนั้นทำการเรียกชื่อสคริปต์ไฟล์ที่ต้องการ run และกด enter

```
>> plotbt.m
```

ก็จะได้กราฟดังรูปที่ 1.4 ข้อดีของสคริปต์ไฟล์ ก็คือ เราสามารถเก็บโปรแกรมที่เราเขียนขึ้นไว้ใช้ในโอกาสต่อๆ ไปได้สะดวก เราสามารถแก้ไขข้อความ เช่น ชื่อไฟล์ข้อมูลอื่น เปลี่ยนรูปแบบของกราฟ โดยไม่ต้องเขียน code ใหม่ทั้งหมด

คำสั่งการนำข้อมูลเข้าและการแสดงผลแบบโต้ตอบ

ตัวอย่างต่อไปนี้จะแสดงการรับข้อมูลและแสดงผลแบบโต้ตอบที่ command window อย่างง่าย โดยโปรแกรมจะคำนวณความยาวด้านตรงข้ามมุมฉาก เมื่อป้อนข้อมูลของด้านประกอบมุมฉากทั้งสอง ตามกฎของ Pythagoras ได้ดังนี้

```
% filename : pytha.m
% my Pythagoras
a = input('Please input the first side ');
b = input('Please input the second side ');
c = sqrt(a^2 + b^2);
disp('The third side = ');
disp(c);
```

คำสั่งควบคุมขั้นตอนการทำงานของ M-file

เรื่องสำคัญเรื่องหนึ่งในการเขียนโปรแกรม ก็คือ การใช้คำสั่ง Control Flow ซึ่งในวิชานี้คำสั่งที่ใช้มากได้แก่ **For Loops, While Loops** และ **If-Else-End** ส่วนคำสั่ง control flow อื่นๆ นิสิตสามารถศึกษาด้วยตัวเอง ตามความสนใจและความต้องการใช้งานในอนาคต ต่อไปนี้จะเป็นการศึกษาคำสั่ง control flow จากโปรแกรมตัวอย่างที่ผู้อื่นได้เขียนไว้แล้ว

1. คำสั่ง **For Loops**

เป็นการ execute ชุดคำสั่งที่อยู่ในลูป เป็นจำนวนกี่ครั้งตามตัวเลขที่กำหนด ลองศึกษาจากตัวอย่าง M-file ต่อไปนี้

```
clear;           %this clears all variables from the workspace
n=10;
beta=zeros(n,1); %Create an nx1 vector to hold Beta vector
for i=1:n        %set # of times to execute the following commands
    beta(i,1)=i+1; %formula for the ith element of beta
end              %end the For loop
beta             %print beta in the command window
```

2. คำสั่ง **While Loops**

เป็นการ execute ชุดคำสั่งที่อยู่ในลูป จนกระทั้งประโยคควบคุม (Control expression) หรือประโยคเงื่อนไขไม่เป็นจริง ลองศึกษาจากตัวอย่าง ต่อไปนี้

```
clear;
b=0;t=0;         %Enter the initial values of variables b and t
while 2^b<200    %Enter the controlling expression
    b=b+1;
    t=t+2^b;     %These two lines are the commands to be executed
end              %End the While Loop
b                %show the value of b
t                %show the value of t
```

3. คำสั่ง **If-Else-End**

จะเป็นการทดสอบ logical expression ที่ตามหลัง **If** ถ้าพบว่าเป็นจริง (true) ก็จะ execute ชุดคำสั่งที่อยู่ในบรรทัดถัดไปต่อจาก logical expression ถ้าเป็นเท็จ (false) ก็จะข้ามไปทำชุดคำสั่งหลัง **else** ลองศึกษาจากตัวอย่างต่อไปนี้

```
clear;
b=randn;     %pick b from the N(0,1) distribution
if b>0
    count=1; %The variable count will be equal to 1 if b>0
else
    count=0; %count is zero otherwise
end
count        %show count in the command window
```

โปรดจำไว้ว่า คำสั่ง control flow ที่กล่าวไปแล้วนี้ สามารถนำมาใช้ด้วยกันได้ หรืออาจเขียนเป็นลูปซ้อนลูป (nested structure) ซึ่งจะเป็นการเขียนโปรแกรมที่มีความซับซ้อนมากยิ่งขึ้น

1.8.2 การเขียนฟังก์ชั่น (Function M-files)

หากชุดคำสั่ง M-file ใด โดยเฉพาะเมื่อทำการประมวลผลแล้วมีการส่งค่าผลลัพธ์กลับออกมา มีความจำเป็นต้องเรียกใช้บ่อยครั้ง เราสามารถนำชุดคำสั่งนั้นมาเขียนเป็นฟังก์ชั่น ไฟล์ที่เก็บฟังก์ชั่นจะต้องมีนามสกุลเป็น (*.m) และชื่อไฟล์จะต้องเป็นชื่อเดียวกันกับชื่อของฟังก์ชั่น ข้อแตกต่างระหว่างการเขียนฟังก์ชั่นและการเขียนสคริปต์ คือ ฟังก์ชั่นจะมีการรับค่า input arguments และส่งค่า output arguments ออกมา เวลาฟังก์ชั่นทำงานใน Matlab จะมี workspace แยกต่างหาก ออกมาจาก workspace ปกติของ command window ข้อแตกต่างอีกประการของฟังก์ชั่น คือ บรรทัดแรกจะต้องมีคำว่า function ไว้หน้าชื่อฟังก์ชั่น เวลา save file ก็บันทึกเป็นชื่อฟังก์ชั่นมีนามสกุล '.m' เป็น 'myfunc.m' ในรูปแบบ

```
function a = myfunc(c,d)
```

คำอธิบาย a คือ ค่า output ที่ต้องการ อาจจะเป็น scalar หรือ matrix ก็ได้

c, d เป็น input parameter

ฟังก์ชั่นอีกรูปแบบหนึ่ง

```
function[a,b,c] = yourfunc(c,d,e,f)
```

จะเป็นการสร้าง function file ชื่อ yourfunc (save ในชื่อ 'yourfunc.m') โดยมี input เป็น d, e, f และ g โดยมีค่าตัวแปร a, b และ c เป็น output

ตัวอย่างการเขียนฟังก์ชั่น เช่น เราต้องการเขียนฟังก์ชั่นเพื่อคำนวณค่า sine ของมุมในหน่วยองศา โดยให้มีชื่อว่า sinedeg แล้วทำการบันทึกชื่อไฟล์ว่า sinedeg.m และในไฟล์นั้นจะมีชุดคำสั่ง ดังนี้

```
% filename : sinedeg.m
% This function computes Sine with degree input
function x = sinedeg(deg)
rad = deg*pi/180;
x=sin(rad);
```

จะเรียกใช้โดยการพิมพ์ที่ command prompt เช่น

```
>> sinedeg(30)
ans =
    0.5000
```

1.9 บทส่งท้าย (Closures)

ในบทส่งท้ายเกี่ยวกับการใช้ Matlab ในการคำนวณทางวิศวกรรมนี้ ได้รวบรวมคำสั่งเพื่ออำนวยความสะดวกในการทำงานในสภาพแวดล้อมของโปรแกรม Matlab ซึ่งเป็นคำสั่งเบื้องต้นที่ใช้ใน Command window ที่ไม่เกี่ยวข้องกับการคำนวณ และเทคนิคการทำงานกับโปรแกรม Matlab ที่ควรทราบ มีดังต่อไปนี้

quit หรือ **exit**	เลิกการทำงานของ Matlab
clc	ลบข้อความที่บรรจุอยู่ใน Command Window แต่ไม่มีการลบค่าตัวแปรใดๆ
clf	ลบรูปภาพที่บรรจุอยู่ใน Graphic Window
clear	ลบตัวแปรทุกตัวออกจากหน่วยความจำ
save	เป็นการรวบรวมค่าตัวแปรทุกตัวที่มีอยู่ในขณะนั้นบันทึกลงบน harddisk
edit	แก้ไขหรือสร้างไฟล์ M-file (.m)
Ctrl + c	หากต้องการยกเลิกการคำนวณในขณะที่ Matlab ยังทำการคำนวณไม่เรียบร้อย

การกำหนด Current Directory และ Search Path

ในการที่จะใช้ M-file ที่เราเขียนขึ้นมาโดยการเรียกที่ command prompt นั้น โปรแกรม Matlab จะต้องหาไฟล์นี้เจอเสียก่อน โดย Matlab จะทำการหาที่ Current Directory และที่ Search Path ที่เรากำหนดไว้

- ใช้ **dir** หรือ **ls**	เพื่อดูรายชื่อไฟล์ ที่มีอยู่ใน current directory
- ใช้ **pwd**	เพื่อดูชื่อ current directory
- ใช้ **cd** <dir>	เพื่อเปลี่ยนไปยัง directory
- ใช้ **path**(path, '*dir*')	เพื่อกำหนด search path เช่น `>> path(path, 'z:\')`

ในกรณีที่คำสั่งใน command line ยาวมาก จนต้องเขียนต่อบรรทัดใหม่ ให้ใช้สัญลักษณ์ ellipsis หรือ continuation symbol คือ จุด 3 จุด เช่น

```
>> a = [1, 2, 3, ...
4 ,5, 6]
```

การขอความช่วยเหลือ (help) ในการใช้โปรแกรม และรูปแบบการใช้ฟังก์ชั่นต่างๆ

- ใช้ **help** อย่างเดียวเพื่อให้ Matlab แสดงชื่อ function ที่มีบรรจุอยู่ใน help
- ใช้ **help** แล้วตามด้วยชื่อ function เพื่อให้ Matlab แสดงรายละเอียดของ function นั้น
- ใช้ **helpwin** จะเปิดหน้าต่าง Help Windows ขึ้นมาโดยเฉพาะ ซึ่งจะสะดวกในการค้นหามากขึ้น
- ใช้คำสั่ง **demo** หากต้องการดูการสาธิตการทำงานของ Matlab

* * * * *