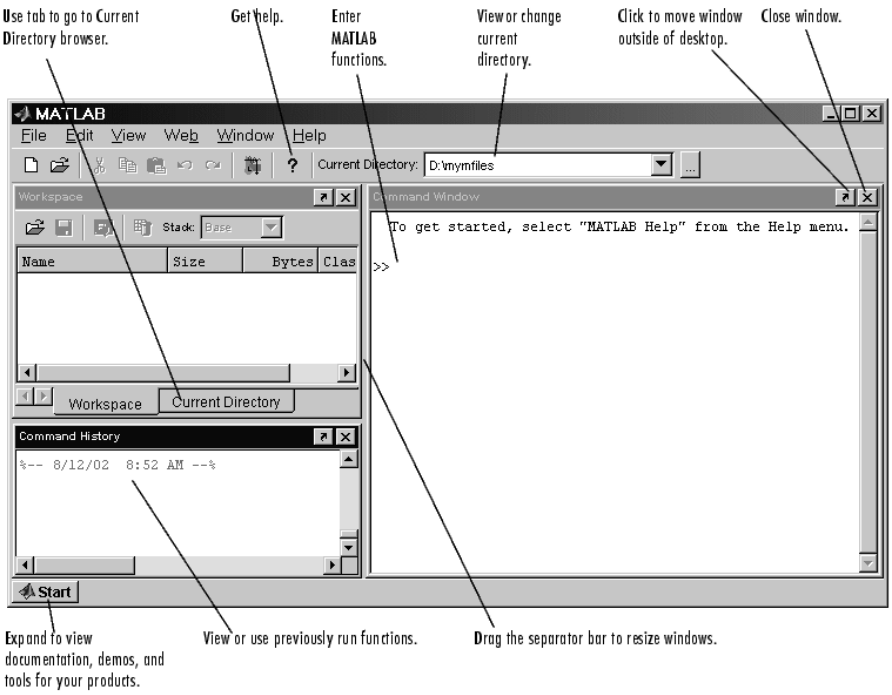
# การประมวลผลภาพด้วยโปรแกรม MATLAB

MATLAB เป็นโปรแกรมคอมพิวเตอร์ที่ถูกออกแบบมาเพื่อจัดการกับข้อมูลที่เป็นเมทริกซ์ จึงมีชื่อย่อมาจาก MATrix LABoratory เป็นโปรแกรมที่เหมาะสำหรับการพัฒนาอัลกอริทึมทางวิทยาศาสตร์และวิศวกรรม โปรแกรมมีคุณสมบัติของการคำนวณ การแสดงข้อมูลทางด้านกราฟิกส์แบบที่ทำให้เห็นภาพได้หรือวิสชวลไลเซชัน(Visualization) และการจำลอง (Simulations)การทำงานของระบบต่างๆ โดยเฉพาะการวิเคราะห์ข้อมูลและเครื่องมือของวิสชวลไลเซชัน ที่ถูกออกแบบให้รองรับข้อมูลที่เป็นเมทริกซ์พร้อมกับตัวดำเนินการต่างๆ นอกจากนี้โปรแกรม MATLAB ยังมีฟังก์ชันสำเร็จรูปหรือฟังก์ชันบิวล์ตอิน(Built-in function)ทางคณิตศาสตร์ให้เรียกใช้มากมาย ทำให้เราสามารถเขียนโปรแกรมและชุดคำสั่งที่เป็นสคริปต์ได้ง่าย สำหรับการประยุกต์ใช้งานต่างๆ จะมีเป็นกล่องเครื่องมือหรือทูลบ็อก(Toolboxes) ซึ่งเป็นฟังก์ชันพิเศษที่เขียนขึ้นโดยผู้เชี่ยวชาญของแต่ละสาขา อย่างในการประมวลผลภาพดิจิตอลจะเกี่ยวข้องกับทูลบ็อกต่างๆ เช่น Images(การประมวลผลภาพ), Imaq(Image acquisition: การนำเข้าข้อมูลภาพ), Signal(การประมวลผลสัญญาณ), Comm(Communication: การสื่อสาร), Stats(Statistics: สถิติ), Fuzzy(ฟัซซีเซต), Nnet(Neural networks: โครงข่ายประสาทเทียม) และ Wavelet (เวฟเล็ต) เป็นต้น

ข้อมูลภาพดิจิตอล เป็นเซตของค่าตัวเลขที่แทนระดับความเข้มของสัญญาณที่เก็บอยู่ในรูปของเมทริกซ์ เนื่องจากโปรแกรม MATLAB รองรับข้อมูลพื้นฐานเป็นอะเรย์หลายมิติ รวมถึงเมทริกซ์และตัวดำเนินการต่างๆ ทำให้มีความเหมาะสมที่จะใช้โปรแกรมนี้ในการประมวลผลภาพ และมีกล่องเครื่องมือที่ช่วยให้การประมวลผลภาพทำได้สะดวกขึ้น ทั้งในส่วนของการประมวลผลภาพและการนำเข้าข้อมูลภาพ ซึ่งกล่องเครื่องมือเหล่านี้จะมีฟังก์ชันพิเศษที่จำเป็นสำหรับการประมวลผลภาพ เช่น เรขาคณิต พีชคณิต ตรรก รวมไปถึงการแปลงข้อมูลภาพ นอกจากนี้ทูลบ็อกยังรองรับข้อมูลภาพได้ทั้งภาพขาวดำ ภาพเกรย์สเกล และภาพสี

เมื่อประมวลผลโปรแกรม หน้าตาโปรแกรมของแต่ละเวอร์ชันมีความแตกต่างกัน แต่จะมีส่วนหลักๆดังปรากฏดังรูปที่ 1 ซึ่งมีองค์ประกอบที่สำคัญในหน้าต่างการใช้งานในแต่ละส่วนดังนี้

1. เมนูบาร์(Menu Bar) เป็นเมนูคำสั่งหลักที่ใช้ในการจัดการหน้าต่างของโปรแกรม
2. Workspace เป็นหน้าต่างแสดงสถานะของตัวแปรที่ใช้งาน พื้นที่หน่วยความจำ และขนาดตัวแปร
3. Command History เป็นหน้าต่างที่ใช้เก็บคำสั่งที่ใช้งานย้อนหลัง สามารถใช้คำสั่งที่อยู่ในวินโดว์นี้โดยการดับเบิลคลิก
4. Command Windows เป็นหน้าต่างที่ใช้เรียกคำสั่งโดยการพิมพ์คำสั่งที่ต้องการประมวลผล




รูปที่ 1 ลักษณะโครงสร้างโปรแกรม MATLAB 6.5 พร้อมใช้งาน

## พื้นฐานการใช้งานเบื้องต้น

รูปแบบการใช้งานที่ง่ายที่สุดของ MATLAB คือใช้เหมือนกับเครื่องคิดเลข เช่นเมื่อต้องการหาผลคูณของ 5 กับ 20 ก็พิมพ์คำสั่งลงในวินโดว์คำสั่ง

```
>>5*20
ans =
        100
```

เมื่อ ans (Answer) เป็นตัวแปรเก็บผลลัพธ์ แต่ถ้าเราต้องการเก็บผลคูณนี้ในตัวแปรอื่น เช่นต้องการเก็บในตัวแปร x ก็ทำได้โดย

```
>>x = 5*20
x =
        100
```

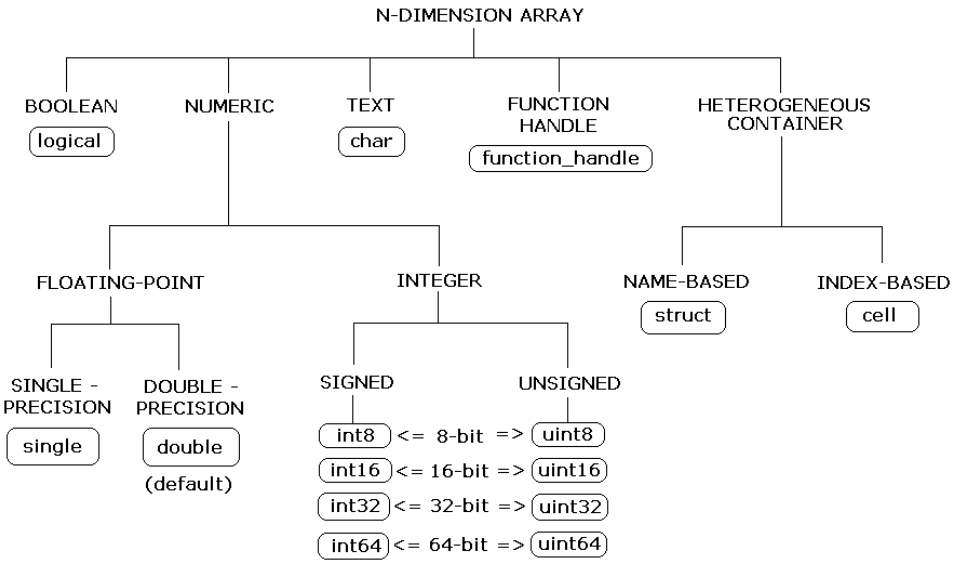
ตอนนี้ตัวแปร x มีค่าเท่ากับ 100 ถ้าเราพิมพ์คำสั่ง y = x*2 คอมพิวเตอร์ก็จะเอาค่าที่เก็บอยู่ในตัวแปร x มาคูณกับ 2 แล้วกำหนดค่าให้กับ y นั่นคือ

```
>>y=x*2
y =
        200
```

จากการใช้งานแบบง่ายๆ พบว่า เราสามารถเก็บค่าที่ป้อนเข้าไปในตัวแปร และสามารถที่จะคำนวณได้ในรูปแบบที่เป็นสมการซึ่งเป็นนิพจน์ของการคำนวณและการกำหนดค่า

### 1. ตัวแปรและนิพนจ์

*ตัวแปร (variables)* ใน MATLAB เป็นตัวแปรที่ใช้สำหรับเก็บข้อมูลที่เป็นอะเรย์ N มิติ ซึ่งแบ่งเป็นคลาสของข้อมูลที่ประกอบด้วย ตัวแปรชนิดตรรก (Logic) ตัวเลข (Numeric) ตัวอักษร(char) ชื่อฟังก์ชัน(Function handle) และตัวแปรที่ใช้เก็บข้อมูลต่างชนิดกัน(Heterogeneous container) ประกอบด้วยตัวแปรโครงสร้าง (Structures) และ เซลล์ (Cells) ตัวแปรชนิดต่างๆ เหล่านี้ได้แสดงไว้ในรูปที่ 2 โดยคำที่ล้อมรอบด้วยวงรีเป็นชนิดของข้อมูล




รูปที่ 2 โครงสร้างรูปแบบของตัวแปรที่ประกาศในโปรแกรม

ตัวแปรใน MATLAB ถ้าไม่มีการกำหนดชนิดจะถูกกำหนดให้เป็นเลขทศนิยมชนิด double เช่นเมื่อเราคำนวณค่า 22*3 ก็จะได้ผลลัพธ์เป็นเลขทศนิยมเท่ากับ 66 แต่ถ้าต้องการใช้เป็นตัวอักษร จะต้องกำหนดชนิดข้อมูล เช่น

```
>>char(66)
ans =
        B
```

จากคำสั่ง char(66) เป็นการแปลงค่า 66 ซึ่งเป็นเลขฐานสิบให้เป็นตัวอักษร โดยมีชนิดข้อมูลคือ char เป็นตัวกำหนด ซึ่งค่า 66 หรือในเลขฐานสองขนาดแปดบิตคือ 01000010 ในรหัสแอสกี(ASCII[1]) ตรงกับตัวอักษร B

ในกรณีที่ต้องการกำหนดเป็นตัวแปรสตริง (string) สามารถกำหนดข้อความได้ดังนี้

```
>>s = 'image processing'
s =
        image processing
```

ตัวแปร s ที่ใช้เก็บข้อความ image processing เป็นตัวแปรอะเรย์ของตัวอักษร เราสามารถหาขนาดของอะเรย์ได้โดยใช้คำสั่ง size ดังนี้

```
>>size(s)
ans =
        1       16
```

[1] ASCII (American Standard Code for Information Interchange) เป็นรหัสเลขฐานสองขนาดแปดบิต ที่ใช้เป็นรหัสตัวอักษรและรหัสควบคุม(Control character)

ขนาดของอะเรย์ s คือ $1 \times 16$ คือมีขนาดหนึ่งแถวสิบหกคอลัมน์ แต่ตัวแปรอะเรย์ของตัวอักษรส่วนใหญ่เราต้องการทราบความยาวของอะเรย์ นอกจาก size ยังมีอีกฟังก์ชันหนึ่งที่ใช้หาความยาวของอะเรย์คือฟังก์ชัน length มีรูปแบบการใช้ดังนี้

```
>>length(s)
ans =
        16
```

เราสามารถเข้าถึงตัวอักษรในแต่ละตำแหน่งของอะเรย์ด้วยดัชนี โดยค่าดัชนีของอะเรย์ใน MATLAB จะเริ่มจาก 1 ดังนั้น s(1) ก็คือ i ส่วนตำแหน่ง s(3) คือ a จะเห็นว่าคำสั่งใน MATLAB ถ้าเราไม่กำหนดค่าให้ ก็จะมีตัวแปร ans ปรากฏในส่วนของการแสดงผลเสมอ ถ้าไม่ต้องการให้ ans ปรากฏ เราสามารถใช้ disp (Display) ซึ่งเป็นคำสั่งใช้แสดงค่า เช่น

```
>>disp(s(3))
a
```

จากอะเรย์ของตัวอักษร ถ้าเราต้องการทราบว่าตัวอักษรใน s มีรหัสแอสกีของเลขฐานสิบเป็นค่าใดบ้าง ก็สามารถดูได้ด้วยการแปลงข้อมูลเป็น double หรือจำนวนเต็มแปดบิตแบบไม่มีเครื่องหมายหรือ uint8 ดังนี้

```
>>uint8(s)
ans =
  Columns 1 through 16
105  109   97  103  101   32  112  114  111   99     101  115  115  105  110  103
```

เราสามารถตรวจสอบรหัสแอสกีเหล่านี้ได้โดยการแปลงกลับไปเป็นตัวอักษร เช่น

```
>>disp(char(105))
i
>>disp(char([105  109  97 103  101  32 112 114 111 99 101 115  115  105 110 103]))
image processing
```

สำหรับข้อมูลภาพโดยทั่วไปเป็นชนิดเลขจำนวนเต็ม และจะเก็บเป็นชนิดแปดบิตไม่มีเครื่องหมาย(Unsigned integer) คือ uint8 เมื่อต้องการให้ตัวแปรเก็บข้อมูลชนิดอื่นที่ไม่ใช่ชนิด double จะต้องระบุชนิดข้อมูล สำหรับรายละเอียดของการกำหนดตัวแปรจะได้อธิบายในหัวข้อต่อไป

*นิพจน์(Expression)* รูปแบบของนิพจ์ทางคณิตศาสตร์ที่ใช้ในโปรแกรมมีลักษณะเป็นการกำหนดชื่อตัวแปรแล้วคั่นด้วยเครื่องหมายเท่ากับ “=” หรือเครื่องหมายกำหนดค่า (Assign to) ตามด้วยตัวแปร ฟังก์ชัน หรือค่าคงที่ นั่นคือ variable = expression เช่น A=B+2 ซึ่งเป็นการกำหนดค่า ด้วยค่าจากตัวแปร B บวกด้วย 2 แล้วเก็บค่าของผลลัพธ์ที่คำนวณได้ไว้ในตัวแปร A หรือจากการกำหนดค่าในตัวอย่างที่ผ่านมาคือ

```
>>s = ‘image processing’
```

สังเกตว่าตัวแปรใน MATLAB เราไม่ต้องประกาศและระบุชนิดของข้อมูลให้กับตัวแปร เพียงแต่กำหนดชื่อตัวแปรตามด้วยข้อมูลที่เรากำหนดค่า ตัวแปรที่อยู่ทางซ้ายของเครื่องหมายเท่ากับก็จะมีชนิดเดียวกับข้อมูลที่กำหนดให้ที่อยู่ทางขวาของเครื่องหมายเท่ากับ ดังนั้น s จึงมีชนิดเป็นอะเรย์ของตัวอักษร

หลักการตั้งชื่อตัวแปรจะคล้ายกับภาษาระดับสูงอื่นๆ เช่น ภาษาซี ปาสคาล เป็นต้น คือขึ้นต้นด้วยตัวอักษร ตามด้วยตัวเลขหรือตัวอักษร ตัวพิมพ์เล็กกับตัวพิมพ์ใหญ่ถือเป็นเป็นคนละตัวแปรกัน เช่น A กับ a ถือว่าเป็นคนละตัวแปรกัน การอ้างถึงหรือการเรียงใช้ตัวแปรจะต้องถูกกำหนดไว้ก่อนการเรียกใช้

## 2 พื้นฐานคำสั่งการจัดการ workspace

**who** ให้แสดงตัวแปรที่ประกาศในหน่วยความจำ

**whos** ให้แสดงตัวแปรที่ประกาศพร้อมแสดงรูปแบบตัวแปรพร้อมขนาดหน่วยความจำที่ใช้ไป

**clear** ลบตัวแปรจากหน่วยความจำ

**load** นำตัวแปรจากดิสก์สู่หน่วยความจำ

**save** เก็บตัวแปรจากหน่วยความจำสู่ดิสก์

**quit** ออกจากโปรแกรม MATLAB

**ตัวอย่างการเรียกใช้งานคำสั่งทีละบรรทัด**

```
>> a=1; b='digital image processing'; c=pi;
>> who
Your variables are:
a  b  c
>> whos
Name   Size            Bytes  Class Attributes
 a         1×1             8  double array
 b         1×24           48  char array
 c         1×1             8  double array
Grand total is 17 elements using 46 bytes
>> save number a b c
>>clear a b
>>whos
 Name     Size            Bytes  Class
 c        1x1             8  double array
Grand total is 1 element using 8 bytes
>>load number
>>who
Your variables are:
a  b  c
>>quit
```

## 3 เมทริกซ์และเวกเตอร์

ตัวแปรที่ใช้เก็บเมทริกซ์และเวกเตอร์ใน MATLAB ทำได้โดยการสร้างตัวแปรอะเรย์ ถ้าเป็นอะเรย์หนึ่งมิติก็จะใช้เก็บข้อมูลที่เป็นเวกเตอร์ ส่วนอะเรย์สองมิติใช้เก็บข้อมูลที่อยู่ในรูปของเมทริกซ์ เครื่องหมายคอมม่า (,) ใช้คั่นสมาชิกแต่ละตัวของแต่ละคอลัมน์หรืออาจจะเว้นวรรค การเว้นวรรคมากหรือน้อยไม่มีผลต่อการกำหนดสมาชิกของอะเรย์ สำหรับเครื่องหมายเซมิโคลอน (Semicolon ;) เป็นตัวแบ่งสมาชิกแต่ละแถว สมาชิกของอะเรย์จะถูกกำหนดอยู่ในวงเล็บเหลี่ยม [ ]

หน้าที่ของเครื่องหมายเซมิโคลอนอีกอย่างหนึ่งคือ ใช้บอกการสิ้นสุดคำสั่งเพื่อไม่ให้มีการแสดงผลออกมาที่จอภาพ เพราะนิพจน์หรือคำสั่งใดๆที่มีผลลัพธ์ถ้าไม่จบด้วยเซมิโคลอน เมื่อคำสั่งถูกประมวลผล ผลลัพธ์จะถูกส่งไปเก็บที่ตัวแปรทางซ้ายของเครื่องหมายเท่ากับ แล้วแสดงผลไปที่ command window

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

```
>> a = [1 2 3;4 5 6;7 8 9];
```

หรือจะใช้คอมม่าคั่นสมาชิกแต่ละตัว

```
>> a = [1, 2, 3;4, 5, 6;7, 8, 9];
```

ตัวอย่างการป้อนเวกเตอร์ $b=[1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6\ \ 7\ \ 8\ \ 9\ \ 10]$ สามารถกำหนดได้หลายแบบ เช่น

```
>>b=[1,2,3,4,5,6,7,8,9,10];        %หรือ
>>b=[1 2 3 4 5 6 7 8 9 10];        %หรือ
>>b=[1:10];                        %รูปแบบใช้ได้เฉพาะกรณีที่มีการเพิ่มค่าที่ละหนึ่งเสมอ หรือ
>>b=[1:1:10];                      %รูปแบบคือ [ค่าเริ่มต้น : เพิ่มที่ละ : ค่าสุดท้าย]
```

การกำหนดเวกเตอร์ b ของสองคำสั่งสุดท้ายเป็นกรณีเฉพาะ เมื่อสมาชิกของเวกเตอร์มีค่าเริ่มต้น แล้วสมาชิกตัวต่อๆไปได้จากการเพิ่มค่าครั้งละเท่าๆกัน จึงกำหนดค่าด้วยรูปแบบของ MATLAB คือ j : i : k เมื่อ j คือค่าเริ่มต้น i เป็นขนาดของการเพิ่มค่า และ k เป็นค่าสุดท้าย เช่นในคำสั่งสุดท้าย 1:1:10 จะเห็นว่า j=1, i=1, และ k=10 ความหมายของการกำหนดค่าแบบนี้คือ [j, j+i, j+2i, ..., j+mi ] ซึ่งค่าสุดท้ายคือ j+mi มีค่าเท่ากับ k ตัวดำเนินการโคลอนจะอธิบายอีกครั้งในเรื่องการเข้าถึงสมาชิกของอะเรย์

การป้อนเวคเตอร์แบบคอลัมน์ $c=\begin{bmatrix}1\\2\\3\\4\end{bmatrix}$

```
>> c = [1; 2; 3; 4];        %หรือ
>> c = [1 2 3 4]';          %ป้อนเป็นเวกเตอร์ แล้วใช้ตัวดำเนินการทรานสโพสเพื่อทำให้เป็นคอลัมน์เวกเตอร์
```

**ตัวดำเนินการเมทริกซ์**

เมทริกซ์สามารถบวกลบคูณและหารได้ตามกฎการดำเนินการของเมทริกซ์ ใน MATLAB ประกอบด้วยตัวดำเนินการบวก (+) การลบ (-) และการคูณ (*) การหาร(/) การยกกำลัง (^) ทรานสโพส (') มีตัวอย่างคำสั่งดังต่อไปนี้

```
>>D=a'                   %การทรานสโพสเมทริกซ์
>>E=a+D; H=a-D;          %การบวกลบเมทริกซ์
>>F=2*E                  %การคูณเมทริกซ์
>>G=E^2                  %การยกกำลังเมทริกซ์ ซึ่งมีค่าเท่ากับ G = E*E
>>E=a/D                  %การหารเมทริกซ์ หรือ E=a*inv(D)
```

จากตัวอย่างตัวดำเนินการเมทริกซ์ เมื่อเครื่องหมายเปอร์เซ็นต์ % ที่ปรากฏต่อท้ายคำสั่งทุกตัวอย่าง เป็นเครื่องหมายไว้ใส่คำอธิบายคำสั่ง (Comment) ข้อความที่อยู่ต่อจากเครื่องหมายนี้จะไม่ถูกประมวลผล คือตัวแปรภาษาจะข้ามข้อความของส่วนนี้

จากตัวอย่างคำสั่งแรก D=a' ตัวแปร a จะต้องถูกกำหนดไว้ก่อน ซึ่งในกรณีนี้ a = [1 2 3;4 5 6;7 8 9]; ผลจากคำสั่งทรานสโพสจะได้เมทริกซ์ D ดังนี้

$$D = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}^T = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 6 \\ 3 & 6 & 9 \end{bmatrix}$$

นอกจากตัวดำเนินการทรานสโพส (') แล้วยังสามารถเรียกใช้ฟังก์ชัน transpose ได้เช่นกัน คำสั่งที่ใช้แทนกันได้คือ D=transpose(a)

ตัวอย่างต่อมา E=a+D; และ H=a-D; ในบรรทัดนี้มีอยู่สองคำสั่ง คือการบวกและลบเมทริกซ์ โดยลำดับ เมื่อ a และ D ถูกกำหนดไว้แล้วในตัวอย่างก่อนหน้านี้ ทั้งนี้การนำตัวแปรที่เป็นเมทริกซ์มาบวกลบกันจะต้องเป็นไปตามกฎการบวกเมทริกซ์ คือตัวโอเพอร์แรน(Operands) ของตัวดำเนินการจะต้องมีขนาดเท่ากัน นอกจากนี้ใน MATLAB ยังมีกฎการแปลงข้อมูล นั่นคือข้อมูลที่ต่างๆชนิดกัน เช่น ถ้า a เป็นชนิด double แต่ D เป็นชนิด uint8 ก็ไม่สามารถนำมาดำเนินการได้ จะต้องแปลงข้อมูลให้เป็นชนิดเดียวกันก่อนจึงจะนำมาประมวลผลได้

คำสั่งต่อไปคือ F=2*E เป็นการคูณเมทริกซ์ด้วยค่าคงที่ สำหรับสองคำสั่งสุดท้ายคือ G=E^2 และ E=a/D สองคำสั่งนี้ถือว่าเป็นตัวดำเนินการคูณเมทริกซ์ทั้งคู่ เพราะ G=E^2 เมทริกซ์ E ยกกำลังสองคือ E*E ส่วน E=a/D เครื่องหมายหารของตัวดำเนินการเมทริกซ์ในเชิงคณิตศาสตร์ไม่มี แต่ในที่นี้ MATLAB จะใช้เครื่องหมายหารเป็น inv(D) คือส่วนกลับหรืออินเวริสเมทริกซ์ (Inverse matrix) ของ D ซึ่ง inv คือฟังก์ชัน inverse ดังนั้น E = a*inv(D) การนำตัวแปรเมทริกซ์มาคูณกันจะต้องเป็นไปตามกฎการคูณเมทริกซ์นั่นคือ

$$C_{m\times n} = A_{m\times r}B_{r\times n}$$

จำนวนคอลัมน์ของเมทริกซ์ A ที่เป็นเมทริกซ์ตัวตั้งจะต้องเท่ากับจำนวนแถวของเมทริกซตัวคูณ ในที่นี้จำนวนคอลัมน์ของ A คือ r ส่วนจำนวนแถวของ B เท่ากับ r เช่นกัน ทำให้เมทริกซ์ผลลัพธ์ C มีขนาด m แถว n คอลัมน์ ตามจำนวนแถวของ A และจำนวนคอลัมน์ของ B ตามลำดับ

เมื่อนำตัวเนินการเมทริกซ์มาใช้รวมกับจุด (.) จะมีการดำเนินการต่างจากตัวดำเนินการเมทริกซ์ปกติ เช่น

```
>> K=[1 2 3]; L=[4 5 6];
>> V = K.*L
   V = [4 10 18]
```

เป็นการนำสมาชิกแต่ละตัวของเวกเตอร์ K และ L มาคูณกันโดยตรง ซึ่งผลลัพธ์ที่ได้ที่เก็บอยู่ในตัวแปร V คือ [4 10 18] ที่ได้จาก [1*4, 2*5, 3*6] ตัวดำเนินการจุดสามารถใช้รวมกับตัวดำเนินการคูณ หาร และยกกำลัง ลักษณะการดำเนินการจะเหมือนกับการคูณ คือดำเนินการกับสมาชิกที่ตำแหน่งเดียวกัน และแน่นอนว่าตัวแปรที่ถูกดำเนินการจะต้องมีสมาชิกเท่ากัน ตัวดำเนินการจุดยังสามารถใช้ร่วมกับ การหาร และ ยกกำลัง เช่น

```
>> V=L./K
V = [4 2.5 2]
>> V=L.^K
V = [4 25 216]
```

**การอ้างถึงสมาชิกของเมทริกซ์**

ตัวแปรอะเรย์ที่เก็บข้อมูลเป็นเวกเตอร์หรือเมทริกซ์สามารถอ้างถึงได้หลายวิธีดังนี้

- อ้างถึงสมาชิกแต่ละตัวได้ด้วยดัชนีบอกตำแหน่ง จะต้องอ้างถึงชื่อตัวแปรและบอกตำแหน่งในวงเล็บเป็น (แถว, คอลัมน์) เช่น $A(1,2)$ เป็นการเข้าถึงสมาชิกแถวที่หนึ่งคอลัมน์ที่สองของเมทริกซ์ $A$ การอ้างอิงแบบนี้ส่วนใหญ่จะใช้ในการเข้าถึงสมาชิกแต่ละตัวด้วยการวนซ้ำในการเขียนโปรแกรม

- อ้างถึงสมาชิกได้ที่ละหลายตัวมีสองแบบคือใช้เครื่องหมายโคลอน (Colon, :) และใช้เวกเตอร์บอกตำแหน่งสมาชิกในอะเรย์ รูปแบบในการอ้างอิงตำแหน่งต่างๆ เช่น $A=\begin{bmatrix}1 & 4 & 7\\ 2 & 5 & 8\\ 3 & 6 & 9\end{bmatrix}$

การเข้าถึงสมาชิกแถวที่สองทุกคอลัมน์ของเมทริกซ์ $A$ สามารถกำหนดได้ดังนี้

```
>> C=A(2,:)
 C =  2    5    8
```

การเข้าถึงสมาชิกทุกแถวของคอลัมน์ที่หนึ่งในเมทริกซ์ $A$ สามารถกำหนดได้ดังนี้

```
>>D = A(:,1)
D =     1
        2
        3
```

การเข้าถึงสมาชิกทุกตัวในเมทริกซ์ $A$ สามารถกำหนดได้ดังนี้

```
>>E = A(:);
```

ในที่นี้ $E$ จะเป็นคอลัมน์เวกเตอร์ที่มีค่าเป็น $E=\begin{bmatrix}1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9\end{bmatrix}^T$ จะเห็นว่ามีความแตกต่างจากการกำหนด E = A; ซึ่งจะได้ E เป็นเมทริกซ์ที่มีค่าเท่ากับ A และมีข้อสังเกตจากการใช้เครื่องหมายโคลอนคือ ถ้าใช้ในตำแหน่งของแถว จะได้ผลลัพธ์เป็นคอลัมน์เวกเตอร์ แต่ถ้าใช้โคลอนในตำแหน่งของคอลัมน์จะได้เวกเตอร์ (Row vector)

การใช้เวกเตอร์ของดัชนีเพื่อเข้าถึงสมาชิก จะมีศักยภาพมากในการเลือกเมทริกซ์ย่อย เช่น

```
>> S = A([1 2],[2 3]) %ในวงเล็บเหลี่ยมคือเวกเตอร์ดัชนีที่ใช้อ้างถึงตำแหน่งในเมทริกซ์ A
S =
   4   7
   5   8
```

เป็นการอ้างถึงเมทริกซ์ย่อยในแต่ละตำแหน่ง นั่นคือ $S=\begin{bmatrix}A_{1,2} & A_{1,3}\\ A_{2,2} & A_{2,3}\end{bmatrix}=\begin{bmatrix}4 & 7\\ 5 & 8\end{bmatrix}$

## 1.5 จำนวนเชิงซ้อน

การคำนวณจำนวนเชิงซ้อนในโปรแกรม MATLAB สามารถกระทำได้โดยตรงโดยการกำหนดตัวแปรเป็นจำนวนเชิงซ้อน โปรแกรม MATLAB เตรียมตัวแปร $i, j=\sqrt{-1}$ ซึ่งเป็นส่วนจินตภาพของจำนวนเชิงซ้อน ดังนั้นจึงควรหลีกเลี่ยงการตั้งชื่อตัวแปรเป็นสองอักษรนี้ ตัวอย่างการป้อนค่าจำนวนเชิงซ้อน

```
>>a=1+j*2; b=3+j*4;
>> a*b
ans =  -5.0000 +10.0000i
>>a=2*exp(j*pi/4)
a =  1.4142 + 1.4142i
```

จากตัวอย่างการกำหนดจำนวนเชิงซ้อนใช้ตัว j แต่ MATLAB จะกำหนดและแสดงผลเป็นตัว i สำหรับแทนส่วนจินตภาพ ดังจะเห็นจากผลลัพธ์ของการคูณจำนวนเชิงซ้อน a*b โปรแกรม MATLAB จะแสดงผลลัพธ์เป็น ans = -5.0000 +10.0000i

และในคำสั่ง a=2*exp(j*pi/4) เป็นการคำนวณค่าเอ็กโพเน็นต์เทียลของผลคูณส่วนจินตภาพ j กับค่าไพน์ (pi หรือ π ที่มีค่าเท่ากับ 22/7) MATLAB ก็แสดงค่าส่วนจินตภาพด้วยค่า i เช่นกัน

MATLAB ยังเตรียมฟังก์ชันพื้นฐานที่เกี่ยวข้องกับจำนวนเชิงซ้อนอีกเช่น

```
>>real(a)      % หาส่วนจำนวนจริงของตัวแปร a
>>imag(a)      % หาส่วนจำนวนจินตภาพของตัวแปร a
>>abs(a)       % หาค่าขนาดของตัวแปร a
>>angle(a)     % หาค่ามุมของตัวแปร a
>>conj(a)      % หาค่าคอนจูเกต(Conjugate) ของตัวแปร a
```

## 1.6 ฟังก์ชันทางคณิตศาสตร์

โปรแกรม MATLAB มีฟังก์ชันทางคณิตศาสตร์มากมายแต่จะยกมาแสดงพอสังเขปได้ดังต่อไปนี้

ฟังก์ชันเกี่ยวกับสเกลา

ฟังก์ชันตรีโกณมิติ sin(x), cos(x), tan(x), asin(x), acos(x), atan(x), atan2(x)

ฟังก์ชันไฮเปอร์โบลิก sinh(x), cosh(x),tanh(x), asinh(x), acosh(x), atanh(x)

ฟังก์ชันเลขยกกำลัง exp(x), log(x), log10(x),

ฟังก์ชันทั่วไป abs(x), sign(x), rem(x), round(x), floor(x), ceil(x), sqrt(x)

ฟังก์ชันเกี่ยวกับเวคเตอร์

max(x), min(x), length(x), mean(x), median(x), std(x), sum(x), sort(x), roots(x)

ฟังก์ชันเกี่ยวกับการสร้างเมตริกซ์

zeros(n,m) สร้างเมตริกซ์ที่มีสมาชิกเป็น 0 ทั้งหมดขนาด n × m

ones(n,m) สร้างเมตริกซ์ที่มีสมาชิกเป็น 1 ทั้งหมดขนาด n × m

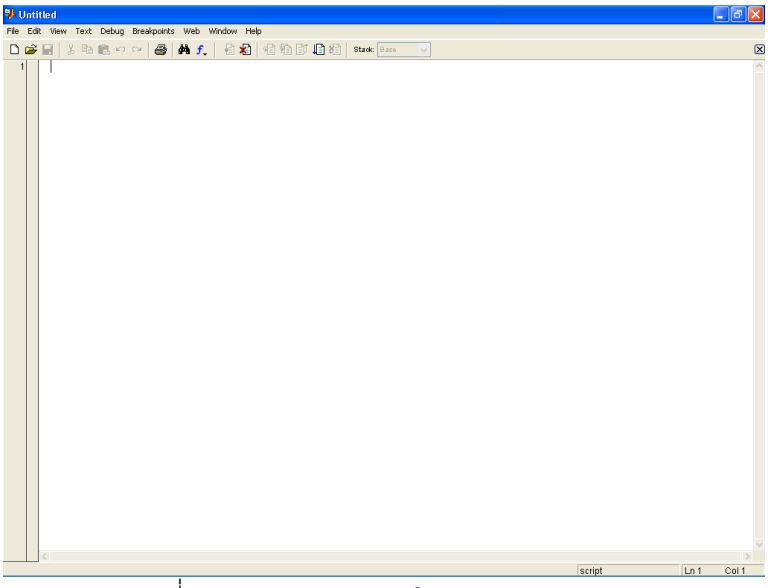
ฟังก์ชันเกี่ยวกับเมตริกซ์

size(x), eig(x), det(x)

ซึ่งรายละเอียดในการใช้คำสั่งที่กล่าวมาข้างต้นสามารถหารายละเอียดได้จากคำสั่ง help command

# 2 พื้นฐานการเขียนสคริปและฟังก์ชัน

การป้อนคำสั่งในการเรียกใช้ command line ที่ละคำสั่งสามารถรวบรวมและสั่งใช้งานหลายคำสั่งโดยการเขียนเป็นลักษณะ m-file ซึ่งภายในโปรแกรม MATLAB editor เป็นตัวสร้างและแก้ไข โดยการใช้จากเมนู ซึ่งเลือก file new m-file จากเมนู จะปรากฏ editor ดังรูปที่ 2




รูปที่ 3 แสดง Editor ของโปรแกรม MATLAB

**การเขียนสคริป**

โปรแกรมของ MATLAB ถูกเรียกว่าสคริปต์(Script) การเขียนสคริปเป็นการเรียกใช้งานคำสั่งที่มีลักษณะเหมือนกับการเขียนโปรแกรมภาษาคอมพิวเตอร์โดยทั่วไป แล้วเก็บเป็นไฟล์ให้มีนามสกุลเป็น .m จึงมักเรียกโปรแกรมหรือสคริปต์ว่า “m-file” ไฟล์นี้สามารถสั่งรันได้จาก editor ของโปรแกรม MATLAB ซึ่งมีตัว debug ได้ด้วย หรือจะเรียกจาก Command line โดยการเรียกชื่อไฟล์เป็นคำสั่ง เครื่องหมายเปอร์เซ็นต์ % จะใช้ในการเขียนคำอธิบายคำสั่ง (Comment) หรือตัวแปร

การสร้างสคริปมีโครงสร้างคำสั่งที่ควบคุมทิศทางของโปรแกรมคล้ายกับทางภาษาสูงโดยทั่วไปในการเขียนโปรแกรม คือ

คำสั่ง if, elseและ elseif ใช้ในการตัดสินใจทางเลือก รูปแบบในการใช้งานคือ

```
if <เงื่อนไขตัดสินใจ>
        <ประโยค>
elseif  <เงื่อนไขตัดสินใจ>
        <ประโยค>
else
        <ประโยค>
end
```

คำสั่ง while ใช้ในวนรอบโดยมีการตรวจสอบเงื่อนไข รูปแบบในการใช้งานคือ

```
while <เงื่อนไขตรวจสอบ>
        <ประโยค>
end
```

คำสั่ง switch ใช้ในการตัดสินใจทางเลือกในกรณีที่มีทางเลือกมากกว่าสามรูปแบบการใช้งานคือ

```
switch ( ตัวแปร)   <scalar or string>
```

case เงื่อนไขที่ 1

<ประโยค> % Executes if expression is value1

case เงื่อนไขที่ 2

<ประโยค> % Executes if expression is value2

.

.

otherwise

<ประโยค>% Executes if expression does not

% does not match any case

end

คำสั่ง for ใช้ในวนรอบ รูปแบบในการใช้งานคือ

for ตัวแปร= ค่าเริ่ม:เพิ่มที่ละ:ค่าสุดท้าย

<ประโยค>

end

*สรุปเกี่ยวกับการเขียนสคริป*

คือการนำคำสั่งมาเขียนเรียงต่อกันแล้วเก็บอยู่ในนามสกุล .m ซึ่งสามารถเรียกใช้งานภายหลังทำให้สะดวกในการใช้งานซ้ำ แต่ข้อจำกัดในการเรียกใช้งานได้หลายครั้งแต่ไม่สามารถรับค่าและคืนค่าตัวแปรได้ ถ้าต้องการรับส่งค่าต้องเขียนในรูปแบบของฟังก์ชันแทน

**การเขียนฟังก์ชัน**

การเขียนฟังก์ชันมีลักษณะคล้ายกับการเขียนสคริปแต่สามารถรับส่งค่าไปมาได้ แต่ในการเรียกใช้งานตัวแปรที่ใช้อยู่ในฟังก์ชันนั้นจะไม่คงอยู่หลังจากการเรียกใช้ฟังก์ชันผ่านไป ซึ่งรูปแบบในการเขียนเป็นฟังก์ชันกำหนดได้ดังนี้

function ชื่อฟังก์ชัน(อินพุต1,อินพุต2)

ประโยค

....

ชื่อฟังก์ชัน=ค่าที่จะคืนกลับ

end

**ตัวอย่างเรียกใช้งานสคริปและฟังก์ชัน**

ตัวอย่างในการคำนวณเกี่ยวกับค่าเฉลี่ยเลขคณิตโดยการเขียนเป็นฟังก์ชันแล้วเรียกใช้โดยการสร้างสคริปมาใช้งานดังนี้

1.สร้างฟังก์ชันดังตัวอย่างข้างล่างแล้ว save ชื่อ average.m

2.สร้างสคริปดังตัวอย่างข้างล่างแล้ว save ชื่อ testmath.m

3.ทดสอบใช้งานโดยพิมพ์ testmath แล้วตามด้วย enter ที่ command line

ตัวอย่างฟังก์ชันที่รับค่าเวคเตอร์แล้วคืนค่าเป็นค่าเฉลี่ยเลขคณิต average.m

function y = average(x)

```
% AVERAGE Mean of vector elements.
% AVERAGE(X), where X is a vector, is the mean of vector elements.
% Nonvector input results in an error.
[m,n] = size(x);
if (~((m == 1) | (n == 1)) | (m == 1 & n == 1))
   error('Input must be a vector')
end
y = sum(x)/length(x);     % Actual computation
```

ตัวอย่างสคริปการเรียกใช้ฟังก์ชัน testmath.m

```
z = 1:99;
average(z)
```

**พื้นฐานคำสั่งเกี่ยวกับกราฟ**

**3.1 Plot** เป็นคำสั่งที่ใช้ในการวาดกราฟในระนาบ XY ซึ่งรูปแบบในการใช้งานมีสามแบบดังนี้

plot(ข้อมูลแกนนอน,ข้อมูลแกนตั้ง)

plot(ข้อมูลแกนตั้ง,ข้อมูลแกนนอน,รูปแบบของเส้นกราฟ)

plot(ข้อมูลแกนตั้ง1,ข้อมูลแกนนอน1,รูปแบบของเส้นกราฟ1,ข้อมูลแกนตั้ง2,ข้อมูลแกนนอน2,รูปแบบของเส้นกราฟ2,...)

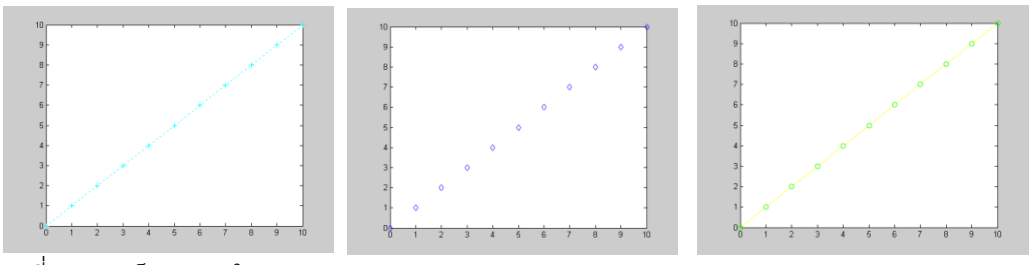
*ข้อกำหนดรูปแบบของเส้นกราฟ*

| สี | | ลักษณะมาร์ค | | ลักษณะเส้น | |
|---|---|---|---|---|---|
| b | blue | . | point | - | solid |
| g | green | o | circle | : | dotted |
| r | red | x | x-mark | -. | dashdot |
| c | cyan | + | plus | -- | dashed |
| m | magenta | * | star | | |
| y | yellow | s | square | | |
| k | black | d | diamond | | |
| | | v | triangle (down) | | |
| | | ^ | triangle (up) | | |
| | | < | triangle (left) | | |
| | | > | triangle (right) | | |
| | | p | pentagram | | |
| | | h | hexagram | | |

ตัวอย่างการเรียกใช้งานในการพล็อตกราฟ

```
>>X=[0:1:10];Y=[0:1:10];
>>plot(X,Y,'c+:')
>>plot(X,Y,'bd')
>>plot(X,Y,'y-',X,Y,'go')
```

ผลการใช้คำสั่งในการพล็อตกราฟข้างต้น



รูปที่ 4 การพล็อตกราฟในรูปแบบลักษณะต่าง ๆ

**Axis** เป็นคำสั่งในการกำหนดขนาดเกี่ยวกับแกนตั้งและแกนนอนที่ปรากฏในกราฟ ซึ่งรูปแบบในการใช้งานรูปแบบดังนี้

| | |
|---|---|
| AXIS ([XMIN XMAX YMIN YMAX]) | การกำหนดแนวแกนตั้งและแกนนอน |
| V = AXIS | เป็นคำสั่งอ่านค่าแนวแกนตั้งและแกนนอน |
| AXIS AUTO | กำหนดแกนแบบอัตโนมัติ |
| AXIS EQUAL | กำหนดแกนทั้งสองให้เท่ากัน |
| AXIS OFF | ไม่ให้แสดงแกนแนวตั้งและแนวนอน |
| AXIS ON | ให้แสดงแนวแกนตั้งและแนวแกนนอน |

**Subplot**

เป็นคำสั่งในการแบ่งรูปย่อยในหน้าต่างเดียวกันให้มี n×m รูปย่อยและโฟกัสไปที่รูปย่อยที่ p ซึ่งรูปแบบคำสั่งดังนี้

SUBPLOT(n,m,p)

**Lable**

xlable('ข้อความที่ต้องการให้ปรากฏบนแกน x')

ylable('ข้อความที่ต้องการให้ปรากฏบนแกน y')

title('ข้อความที่ต้องการให้ปรากฏชื่อกราฟ')

**hold on และ hold off**

hold on เป็นคำสั่งในการโฟกัสกราฟรูปที่ต้องการไว้เมื่อมีคำสั่งเกี่ยวกับกราฟจะกระทำที่กราฟรูปเดิมจนกระทั่งใช้คำสั่ง hold off เมื่อกระทำกับกราฟจะกระทำกับกราฟรูปใหม่