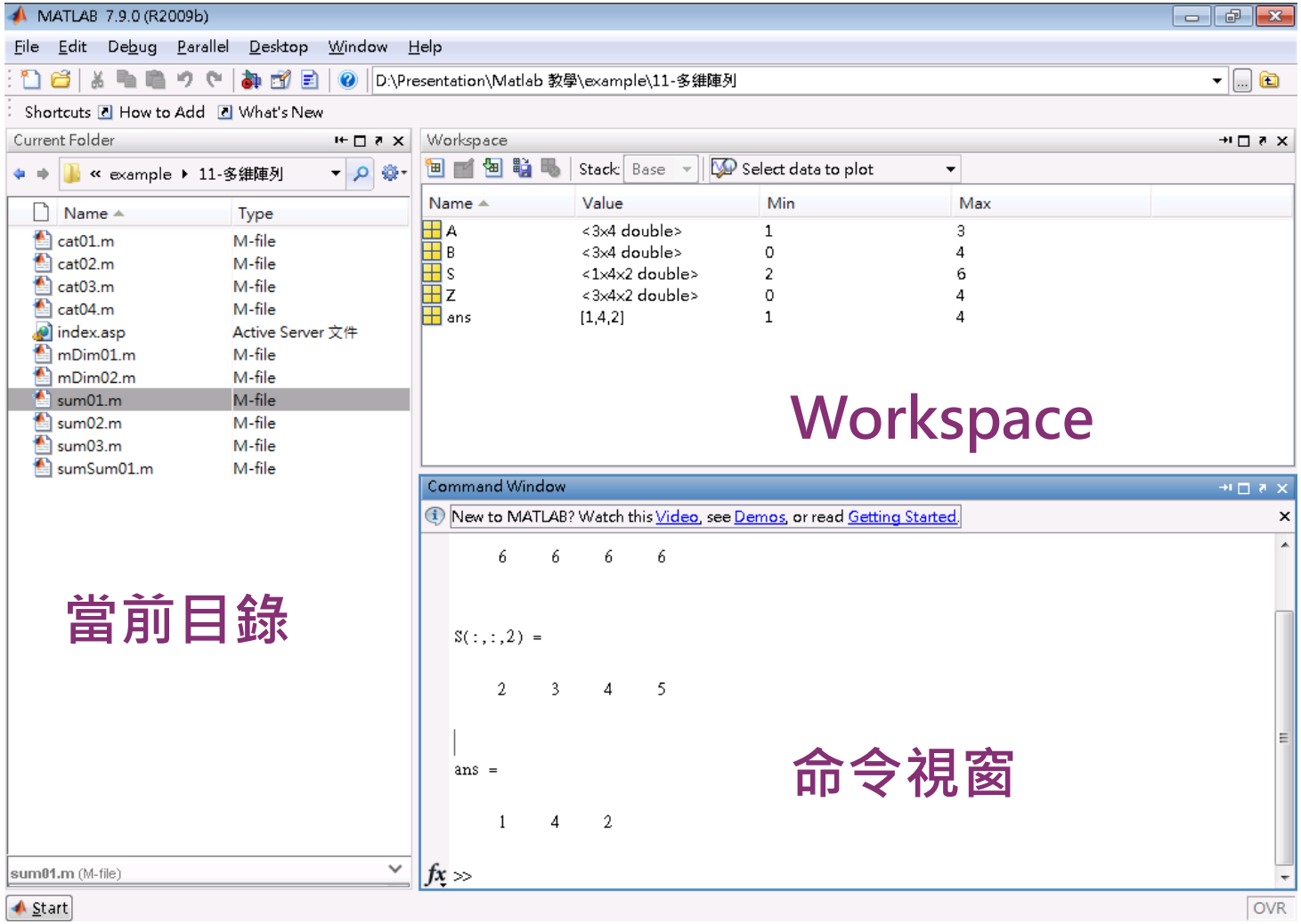
MATLAB 程式設計

江振國 (Adrian Chiang)
清大資工所 電腦視覺實驗室

20120920

MATLAB 環境介面


MATLAB 7.9.0 (R2009b)
File Edit Debug Parallel Desktop Window Help
D:\Presentation\Matlab 教學\example\11-多維陣列
Shortcuts How to Add What's New
Current Folder
example ▸ 11-多維陣列
Name Type
cat01.m M-file
cat02.m M-file
cat03.m M-file
cat04.m M-file
index.asp Active Server 文件
mDim01.m M-file
mDim02.m M-file
sum01.m M-file
sum02.m M-file
sum03.m M-file
sumSum01.m M-file
sum01.m (M-file)
當前目錄
Workspace
Stack: Base
Select data to plot
Name Value Min Max
A <3x4 double> 1 3
B <3x4 double> 0 4
S <1x4x2 double> 2 6
Z <3x4x2 double> 0 4
ans [1,4,2] 1 4
Workspace
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
6 6 6 6
S(:,:,2) =
2 3 4 5
ans =
1 4 2
>>
命令視窗
Start
OVR

有趣的指令

- 安裝MATLAB後，你會想做的事（請直接在命令視窗內的提示符號 “>>” 之後輸入下列指令）：
  - 檢查版本
    - version
    - ver
  - 測試電腦速度：bench

2-1 使用變數與基本運算

- 一般數學符號運算
  - 在MATLAB 命令視窗內的提示符號（ >> ）之後輸入運算式，並按入 Enter 鍵即可：

```
>> (5*2+3.5)/5
ans =
    2.7000
```

其中ans是內建的變數，用於儲存每次的運算結果。

  - 我們也可以將運算結果送到另一個變數：

```
>> x =(5*2+3.5)/5
x =
    2.7000
```

  - 若不想讓 MATLAB 每次都顯示運算結果，只需在運算式最後加上分號即可：

```
>> y=(5*2+3.5)/5;
```

變數命名規則與使用

- 第一個字元必需是英文，後面可以接數字或是底線。
- 最多只能有 31 個字母，MATLAB 會忽略多餘字母。
- MATLAB 在使用變數時，不需預先經過變數宣告（Variable Declaration）的程序，而且所有數值變數均以預設的 double 資料型態（佔用8個bytes）來儲存。

加入註解

- 若要加入註解（Comments），可以使用百分比符號（%）例如：

```
>> y = (5*2+3.5)/5;        % 將運算結果儲存在變數 y，但不用顯示於螢幕
>> z = y^2                 % 將運算結果儲存在變數 z，並顯示於螢幕
z =
    7.2900
```

2-2 向量與矩陣的處理

- MATLAB 中的變數還可用來儲存向量（Vectors）及矩陣（Matrix），以進行各種運算，例如：

```
>> s = [1 3 5 2];        % 注意 [] 的使用，及各數字間的空白間隔
>> t = 2*s+1
t =
     3     7    11     5
```

矩陣的各種處理

- MATLAB 亦可取出向量中的一個元素或一部份來做運算，例如：

```
>> t(3) = 2     % 將向量 t 的第三個元素更改為 2
 t =
      3     7     2     5
>> t(6) = 10                % 在向量 t 加入第六個元素，其值為 10
 t =
     3     7     2     5     0    10

>> t(4) = []                % 將向量 t 的第四個元素刪除，[] 代表空集合
t =
     3     7     2     0    10
```

建立大小為 M×N的矩陣

- 常用名詞：橫列➔row, 直行➔column
- 欲建立矩陣，可在每一橫列結尾加上分號（；），例如：

```
>> A = [1 2 3 4; 5 6 7 8; 9 10 11 12];        % 建立 3×4 的矩陣 A
>> A                                          % 顯示矩陣 A 的內容
A =
     1     2     3     4
     5     6     7     8
     9    10    11    12
```

MXN矩陣的各種處理 (I)

- >> A(2,3) = 5　　% 將矩陣 A 第二列、第三行的元素值，改變為 5

  A =

  1　2　3　4

  5　6　5　8

  9　10　11　12

- >> B = A(2,1:3)　　% 取出矩陣 A 的第二橫列、第一至第三直行，並儲存成矩陣 B

  B =

  5　6　5

MXN矩陣的各種處理 (II)

- >> A = [A B']　% 將矩陣 B 轉置後、再以行向量併入矩陣 A

```
A =
     1     2     3     4     5
     5     6     5     8     6
     9    10    11    12     5
```

- >> A(:, 2) = []　　% 刪除矩陣 A 第二行（ ：代表所有橫列，[]代表空矩陣）

```
A =
     1     3     4     5
     5     5     8     6
     9    11    12     5
```

MXN矩陣的各種處理 (III)

- >> A = [A; 4 3 2 1]　　% 在原矩陣 A 中，加入第四列

  A =

1	3	4	5
5	5	8	6
9	11	12	5
4	3	2	1

- >> A([1 4], :) = []　　% 刪除第一、四列 (：代表所有直行，[]是空矩陣)

  A =

5	5	8	6

2-3 常用數學函數

- MATLAB 是一個科學計算軟體，因此可以支援很多常用到的數學函數
  - y = abs(x) ➔ 取 x 的絕對值
  - y = sin(x) ➔ 取 x 的正弦值
  - y = exp(x) ➔ 自然指數 exp(x)
  - y = log(x) ➔ 自然對數 ln(x)

- MATLAB 也支援複數運算，通常以 i 或 j 代表單位虛數

向量矩陣的運算

- 有一些函數是特別針對向量而設計
  - y = min(x) ➔ 向量 x 的極小值
  - y = max(x) ➔ 向量 x 的極大值
  - y = mean(x) ➔ 向量 x 的平均值
  - y = sum(x) ➔ 向量 x 的總和
  - y = sort(x) ➔ 向量 x 的排序

線上支援

- help：查詢指令的用法（顯示於命令視窗）。
  - 例如：help mean
- doc：查詢指令的用法（顯示於線上支援視窗）
  - 例如：doc mean
- lookfor：用來尋找未知的指令。找到所需的指令後，即可用 help 進一步找出其用法。
- helpwin 或 helpdesk：顯示線上支援視窗（其效果等同於直接點選 MATLAB 命令視窗工作列的圖示）。

2-4 程式流程控制

- MATLAB 提供重複迴圈（Loops）及條件判斷（Conditions）等程式流程控制（Flow Control）的指令
  - for 迴圈是最常用到的重複運算，其中迴圈變數會依次取用每個行向量來進行運算，格式如下：

```
for 變數 = 向量
        運算式;
end
```

流程控制

- while 迴圈（While-loop）

```
while 條件式
        運算式;
end
```

- if – else – end

```
if 條件式
   運算式;
else
   運算式;
end
```

2-5 M 檔案

- 若要一次執行大量的 MATLAB 指令，可將這些指令存放於一個副檔名為 m 的檔案，並在 MATLAB 指令提示號下鍵入此檔案的主檔名即可。

```
>> pwd              % 顯示目前的工作目錄
>> cd d:\matlabBook\MATLAB程式設計：入門篇\02-初探 MATLAB
>> type myTest.m    % 顯示 myTest.m 的內容
>> myTest           % 執行 myTest.m
```

2-6 搜尋路徑

- 若要檢視 MATLAB 已設定的搜尋路徑，鍵入 path 指令即可
- 若只要查詢某一特定指令所在的搜尋路徑，可用 which 指令
- 要將目錄加入 MATLAB 的搜尋路徑，可使用 addpath 指令

2-7 工作空間與變數的儲存及載入

- MATLAB 在進行各種運算時，會將變數儲存在記憶體內，這些儲存變數的記憶體空間稱為基本工作空間（Base Workspace）或簡稱工作空間（Workspace)
  - 若要檢視現存於工作空間（Workspace）的變數，可鍵入 who
  - 若要知道這些變數更詳細的資料，可使用 whos 指令

檢視工作空間變數的其他方式

- 使用 clear 指令來清除或刪除工作空間內的某一特定或所有變數，以避免記憶體的閒置與浪費
- 不加任何選項（Options）時，save 指令會將工作空間內的變數以二進制（Binary）的方式儲存至副檔名為 mat 的檔案
  - save：將工作空間的所有變數儲存到名為 matlab.mat 的 二進制檔案。
  - save filename：將工作空間所有變數儲存到名為 filename.mat 的二進制檔案。
  - save filename x y z：將變數 x、y、z 儲存到名為 filename.mat 的二進制檔案。

2-8 離開 MATLAB

- 在命令視窗內，鍵入 exit 指令。
- 在命令視窗內，鍵入 quit 指令。
- 直接關閉 MATLAB 的命令視窗。

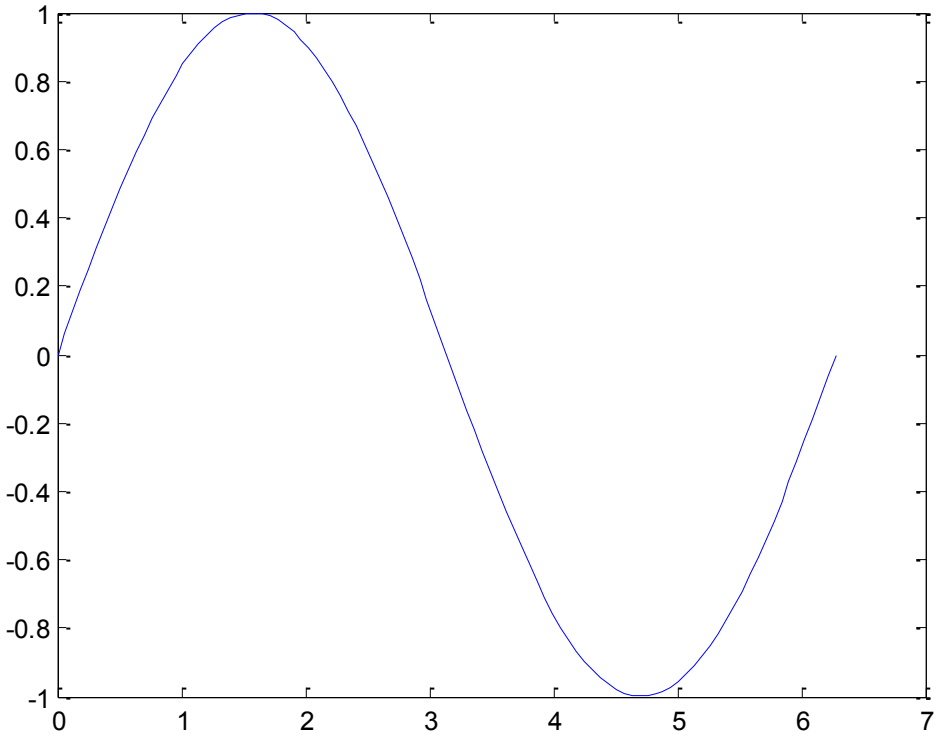
二維平面繪圖

3-1 基本的繪圖指令

- Plot：最基本的繪圖指令
- 對 x 座標及相對應的 y 座標進行作圖
  - 範例3-1：plotxy01.m

```
x = linspace(0, 2*pi);      % 在 0 到 2π 間，等分取 100 個點
y = sin(x);                 % 計算 x 的正弦函數值
plot(x, y);                 % 進行二維平面描點作圖
```

PLOT基本繪圖-1



- linspace(0, 2*pi) 產生從 0 到 2π且長度為 100 (預設值) 的向量 x
- y 是對應的 y 座標

- 只給定一個向量
  - 該向量則對其索引值(Index)作圖
- plot(y)和plot(1:length(y), y)會得到相同的結果

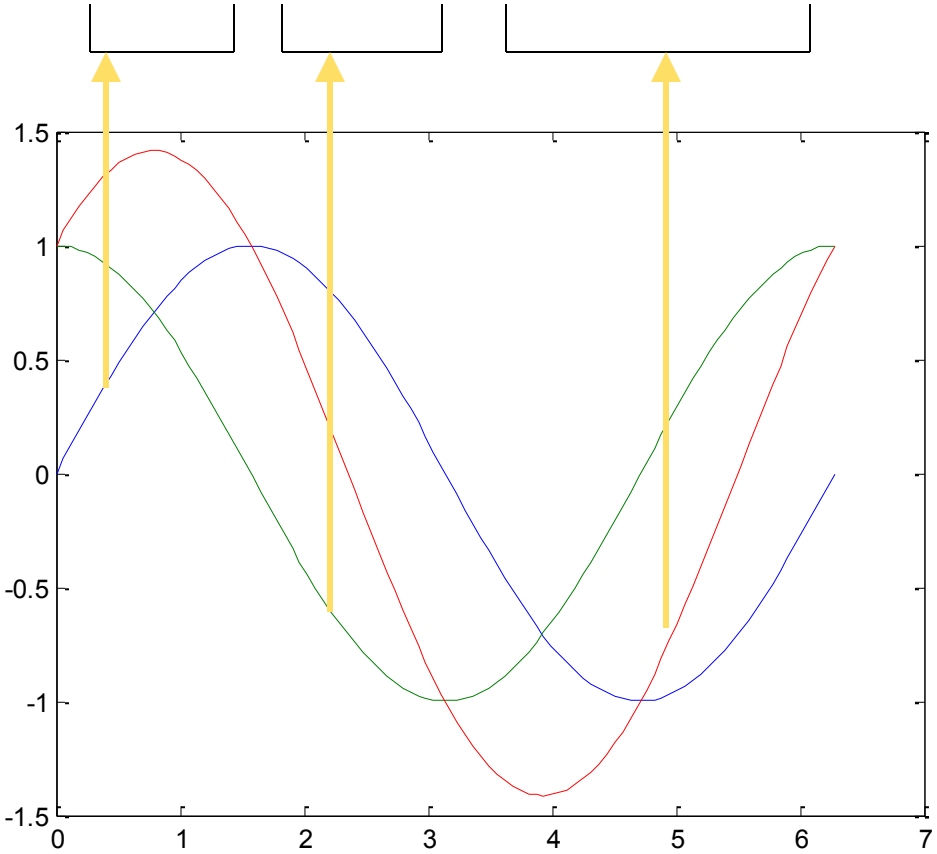
PLOT基本繪圖-2 (I)

- 一次畫出多條曲線
  - 將 x 及 y 座標依次送入plot 指令
  - 範例3-2：plotxy02.m

```
x = linspace(0, 2*pi);                    % 在 0 到 2 間，等分取 100 個點
plot(x, sin(x), x, cos(x), x, sin(x)+cos(x));  % 進行多條曲線描點作圖
```

PLOT基本繪圖-2 (II)

Plot(x,sin(x), x, cos(x), x, sin(x)+cos(x));



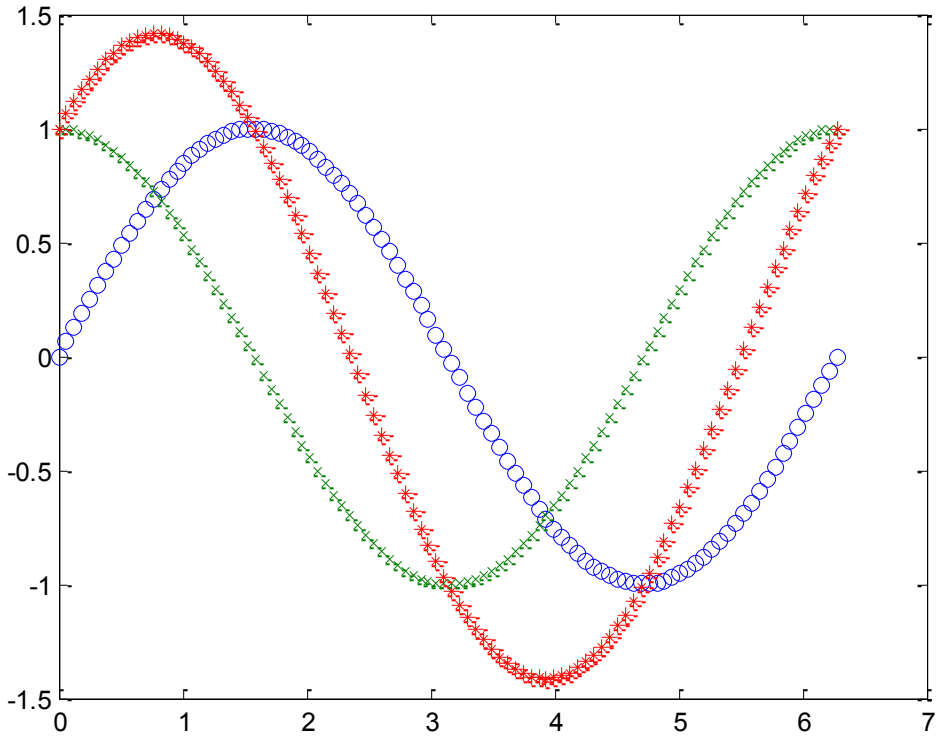
- 畫出多條曲線時，會自動輪換曲線顏色

PLOT基本繪圖-3 (I)

- 若要以不同的線標(Marker)來作圖
  - 範例3-3：plotxy03.m

```
x = linspace(0, 2*pi);        % 在 0 到 2π 間，等分取 100 個點
plot(x, sin(x), 'o', x, cos(x), 'x', x, sin(x)+cos(x), '*');
```

PLOT基本繪圖-3 (II)

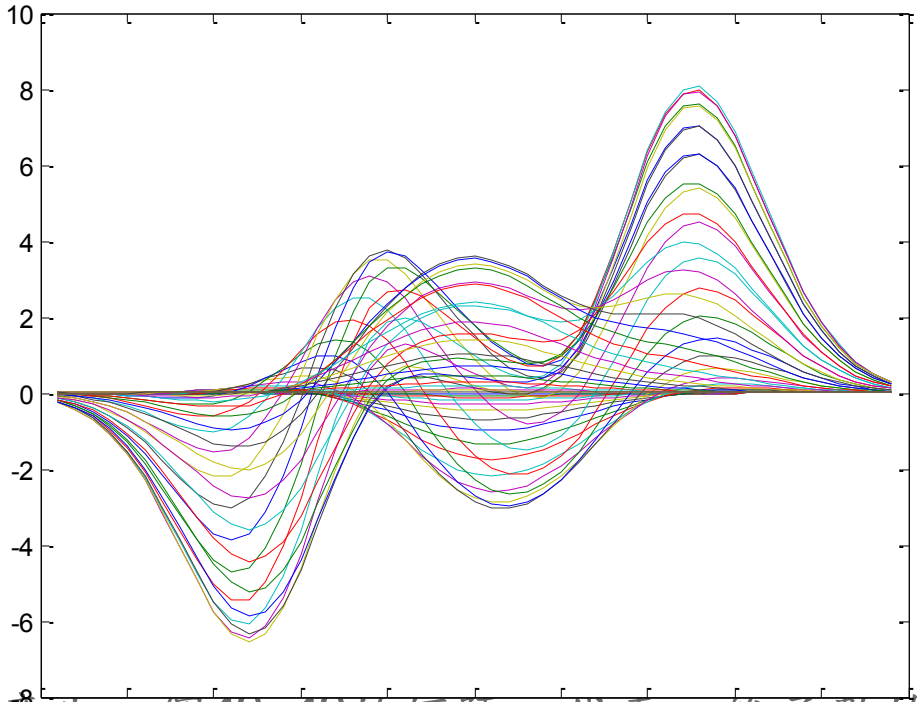


PLOT基本繪圖-4 (I)

- 只給定一個矩陣 y
  - 對矩陣 y 的每一個行向量(Column Vector)作圖
  - 範例3-4：plot04.m

```
y = peaks;                    % 產生一個 49x49 的矩陣
plot(y);                      % 對矩陣 y 的每一個行向量作圖
```

PLOT基本繪圖-4 (II)



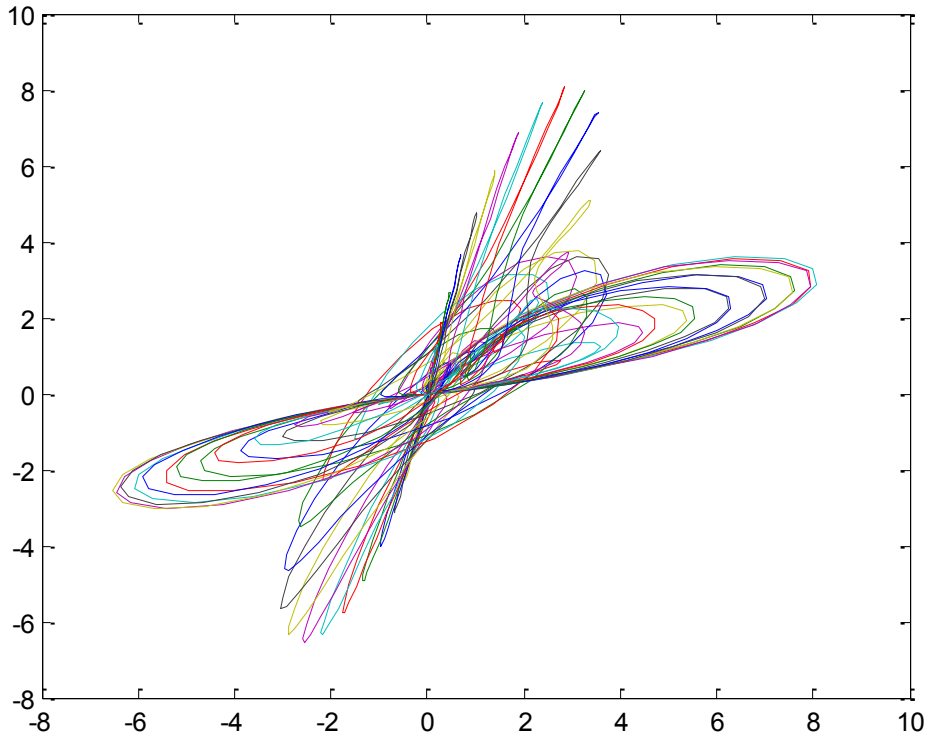
- peaks 指令產生一個49×49的矩陣，代表二維函數的值
- plot(y) 直接畫出 49 條直線
- 類似於從側面觀看 peaks 函數

PLOT基本繪圖-5 (I)

- x 和 y 都是矩陣
- plot(x, y) 會取用 y 的每一個行向量和對應的 x 行向量作圖
  - 範例3-5：plotxy05.m

```
x = peaks;
y = x';          % 求矩陣 x 的轉置矩陣 x'
plot(x, y);      % 取用矩陣 y 的每一行向量，與對應矩陣 x
                 % 的每一個行向量作圖
```

PLOT基本繪圖-5 (II)



提示

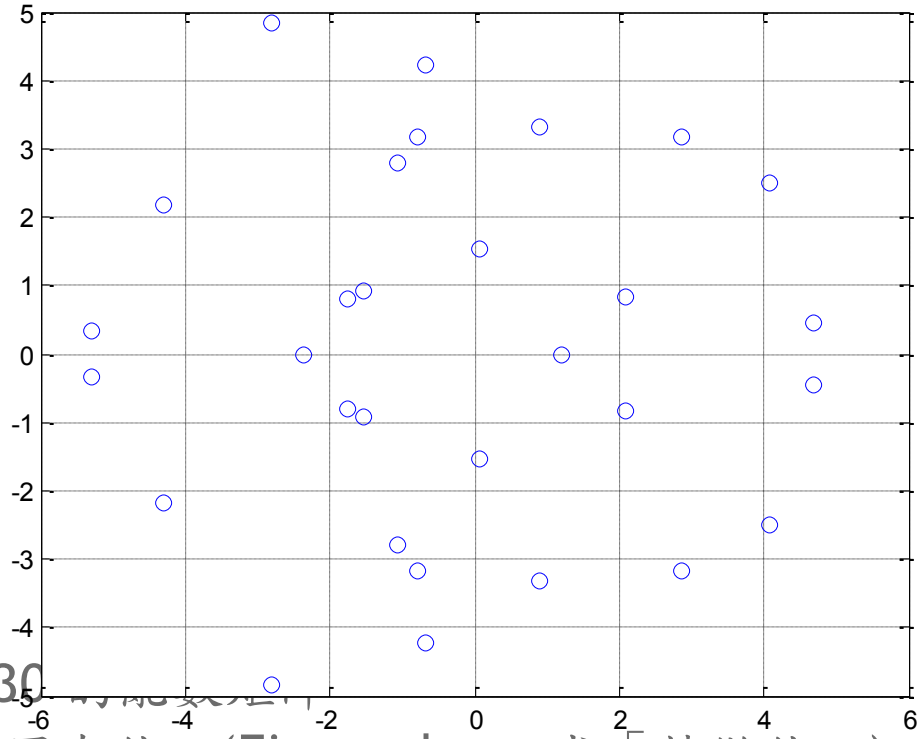
- 一般情況下，MATLAB 將矩陣視為行向量的集合
- 對只能處理向量的函數(Ex：max、min、mean)
  - 給定一個矩陣，函數會對矩陣的行向量一一進行處理或運算

PLOT基本繪圖-6 (I)

- z 是一個複數向量或矩陣
  - plot(z) 將 z 的實部(即 real(z))和虛部(即 imag(z)) 當成 x 座標和 y 座標來作圖，
  - 其效果等於 plot(real(z), imag(z))
  - 範例3-6：plotxy06.m

```
x = randn(30);      % 產生 30x30 的亂數(正規分佈)矩陣
z = eig(x);         % 計算 x 的「固有值」(或稱「特徵值」)
plot(z, 'o')
grid on             % 畫出格線
```

PLOT基本繪圖-6 (II)



- x 是一個 30×30 的亂數矩陣
- z 則是 x 的「固有值」(Eigenvalue，或「特徵值」)
- z 是複數向量，且每一個複數都和其共軛複數同時出現，因此畫出的圖是上下對稱

基本二維繪圖指令

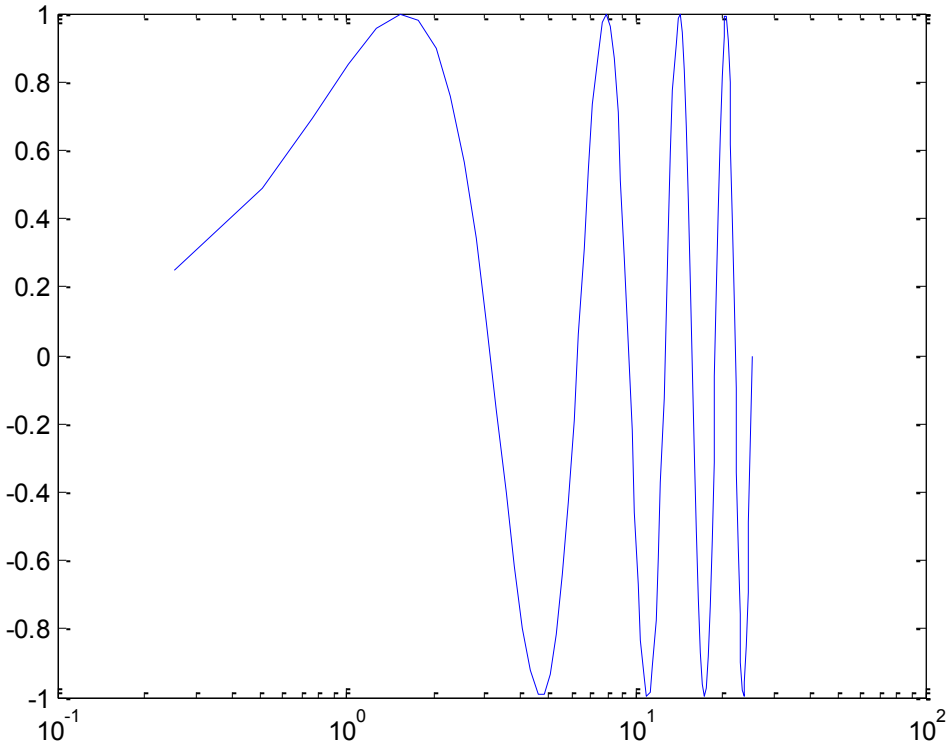
指令	說明
Plot	x 軸和 y 軸均為線性刻度(Linear Scale)
loglog	x 軸和 y 軸均為對數刻度(Logarithmic Scale)
semilogx	x 軸為對數刻度，y 軸為線性刻度
semilogy	x 軸為線性刻度，y 軸為對數刻度
plotyy	畫出兩個刻度不同的y軸

PLOT基本繪圖-7 (I)

- Semilogx指令
  - 使 x 軸為對數刻度，對正弦函數作圖
  - 範例plotxy07.m

```
x = linspace(0, 8*pi);        % 在 0 到 8 間，等分取 100 個點
semilogx(x, sin(x));    % 使 x 軸為對數刻度，並對其正弦函數作圖
```

PLOT基本繪圖-7 (II)



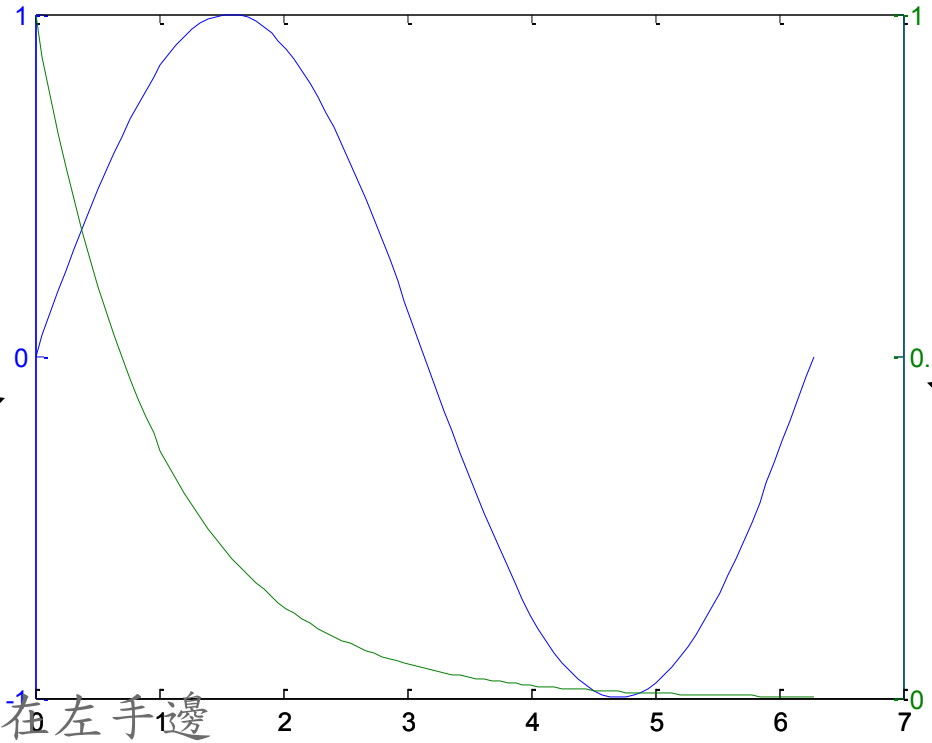
X軸為對數刻度

PLOT基本繪圖-8 (I)

- plotyy 指令
  - 畫出兩個刻度不同的 y 軸
  - 範例3-8：plotxy08.m

```
x = linspace(0, 2*pi);        % 在 0 到 2 間，等分取 100 個點
y1 = sin(x);
y2 = exp(-x);
plotyy(x, y1, x, y2);   % 畫出兩個刻度不同的 y 軸，分別是 y1, y2
```

PLOT基本繪圖-8 (II)



y1的刻度

y2的刻度

- y1 的刻度是在左手邊
- y2 的刻度是在右手邊
- 兩邊的刻度不同

3-2 圖形的控制

- plot 指令，可以接受一個控制字串輸入
  - 用以控制曲線的顏色、格式及線標
  - 使用語法

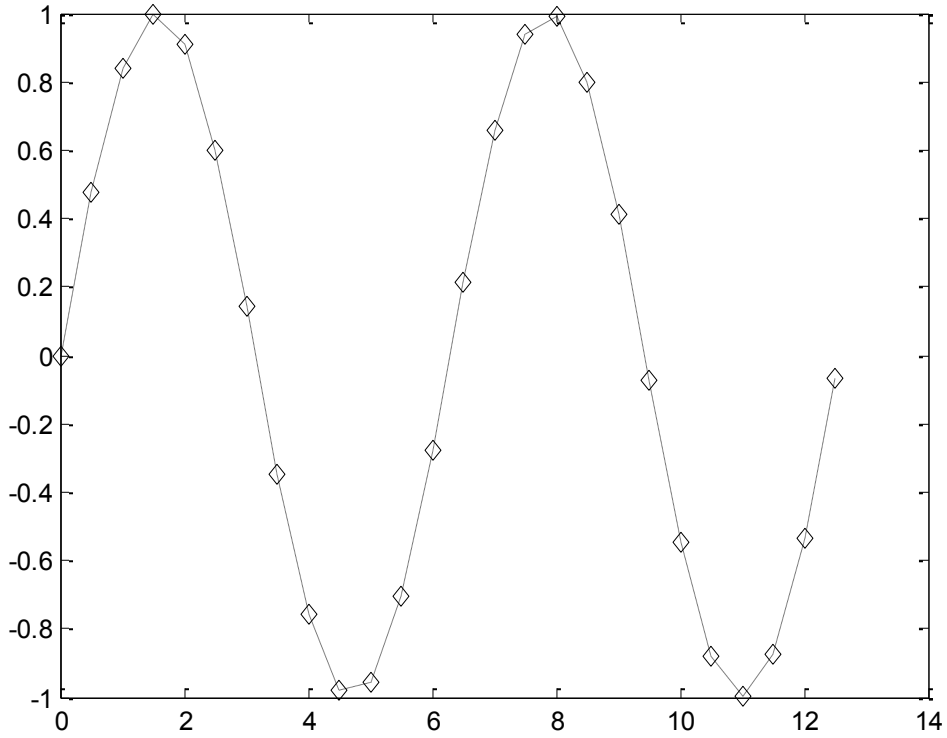
    plot(x, y, ‘CLM‘)
  - C：曲線的顏色 (Colors)
  - L：曲線的樣式 (Line Styles)
  - M：曲線資料點所用的線標 (Markers)

圖形控制範例-1 (I)

- 用黑色點線畫出正弦波
- 每一資料點畫上一個小菱形
- 範例3-9：plotxy09.m

```
x = 0:0.5:4*pi;          % x 向量的起始與結束元素為 0 及 4，
                         % 0.5為各元素相差值
y = sin(x);
plot(x, y,'k:diamond')   % 其中「k」代表黑色，「：」代表點
                         % 線，而「diamond 」則指定菱形為曲
                         % 線的線標
```

圖形控制範例-1 (II)



PLOT 指令的曲線顏色

Plot指令的曲線顏色字串	曲線顏色	RGB值
b	藍色(Blue)	(0,0,1)
c	青藍色(Cyan)	(0,1,1)
g	綠色(Green)	(0,1,0)
k	黑色(Black)	(0,0,0)
m	紫黑色(Magenta)	(1,0,1)
r	紅色(Red)	(1,0,0)
w	白色	(1,1,1)
y	黃色(Yellow)	(1,1,0)

PLOT 指令的曲線格式

plot 指令的曲線樣式字串	曲線樣式
-	實線(預設值)
--	虛線
:	點線
-.	點虛線

PLOT 指令的曲線線標 (I)

plot 指令的曲線線標字串	線標說明
O	圓形
+	加號
X	叉號
*	星號
.	點號
^	朝上三角形
V	朝下三角形

PLOT 指令的曲線線標 (II)

plot 指令的曲線線標字串	線標說明
>	朝右三角形
<	朝左三角形
square	方形
diamond	菱形
pentagram	五角星形
hexagram	六角星形
None	無符號(預設值)

3-3 圖軸的控制

- plot 指令會根據座標點自動決定圖軸範圍
- 也可以使用 axis 指令指定圖軸範圍
  - 使用語法：

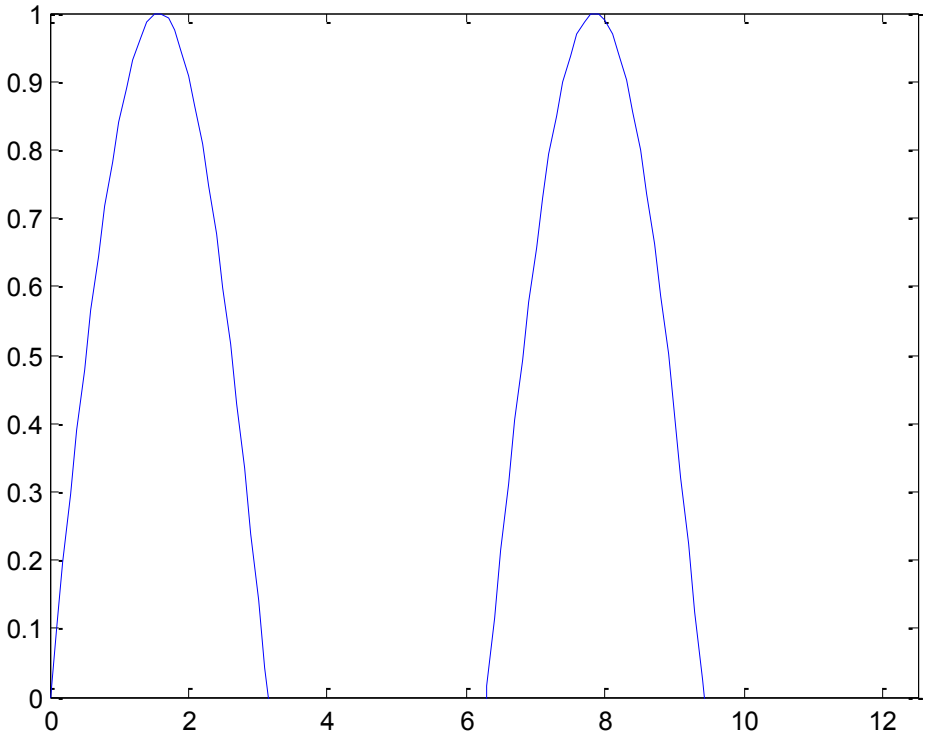
    axis([xmin, xmax, ymin, ymax])
  - xmin , xmax：指定 x 軸的最小和最大值
  - ymin , ymax：指定 y 軸的最小和最大值

圖軸控制範例-1 (I)

- 畫出正弦波在 y 軸介於 0 和 1 的部份
  - 範例3-10：plotxy10.m

```
x = 0:0.1:4*pi;                 % 起始與結束元素為 0 及 4，0.1 為各
                                % 元素相差值
y = sin(x);
plot(x, y);
axis([-inf, inf, 0, 1]);        % 畫出正弦波 y 軸介於 0 和 1 的部份
```

圖軸控制範例-1 (II)



- inf指令：
  - 以資料點(上例： x 軸的資料點)的最小和最大值取代之

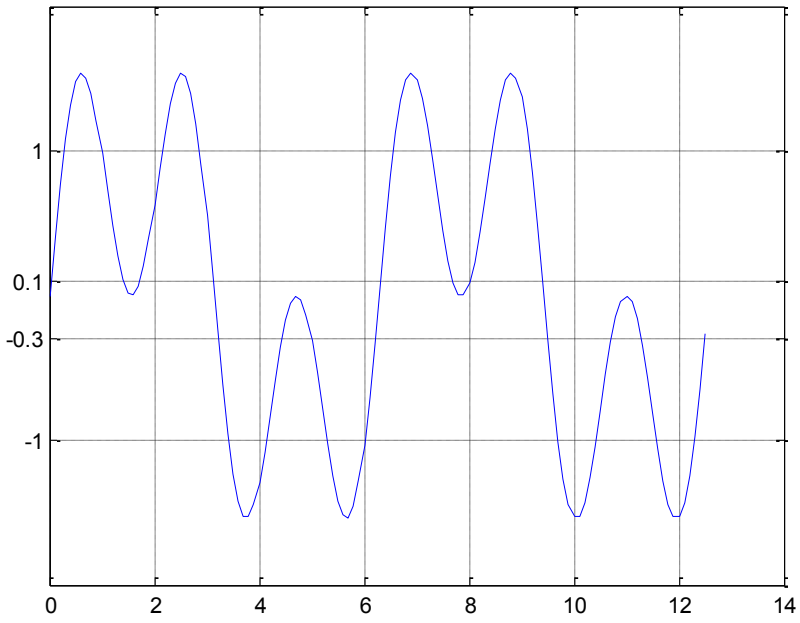
圖軸控制範例-2 (I)

- 指定圖軸上的格線點(Ticks)
  - 範例3-11：plotxy11.m

```
x = 0:0.1:4*pi;
plot(x, sin(x)+sin(3*x))
set(gca, 'ytick', [-1 -0.3 0.1 1]);      % 在 y 軸加上格線點
grid on                                  % 加上格線
```

圖軸控制範例-2 (II)

使用者加入的格線點和文字



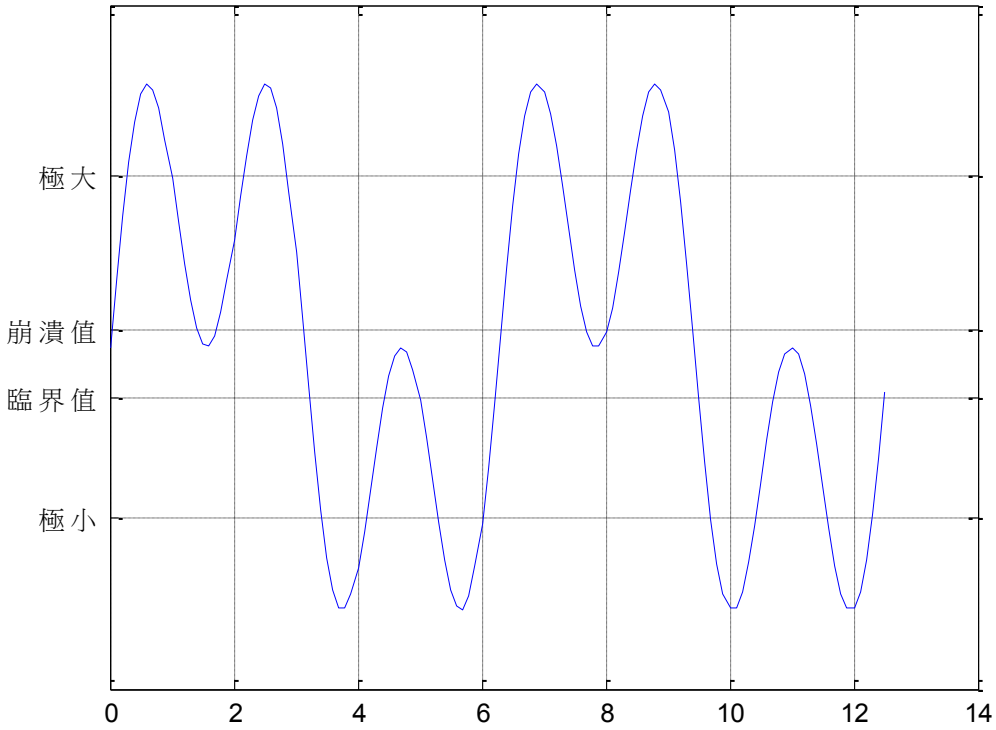
- grid on：加上格線
- gca：
  - get current axis的簡稱
  - 傳回目前使用中的圖軸
  - gca屬Handle Graphics的指令，第七章會有更詳細的說明

圖軸控制範例-3 (I)

- 將格線點的數字改為文字
  - 範例3-12：plotxy12.m

```
x = 0:0.1:4*pi;
plot(x, sin(x)+sin(3*x))
set(gca, 'ytick', [-1 -0.3 0.1 1]);                    % 改變格線點
set(gca, `yticklabel', {`極小' ,`臨界值' ,`崩潰值' ,`極大' });
                                                       % 改變格線點的文字
grid on                                                % 加上格線
```

圖軸控制範例-3 (II)



SUBPLOT

- subplot
  - 在一個視窗產生多個圖形(圖軸)
  - 一般形式為 subplot (m, n, p)
  - 將視窗分為 m ×n 個區域
  - 下一個 plot 指令繪圖於第 p 個區域
  - p 的算法為由左至右，一列一列

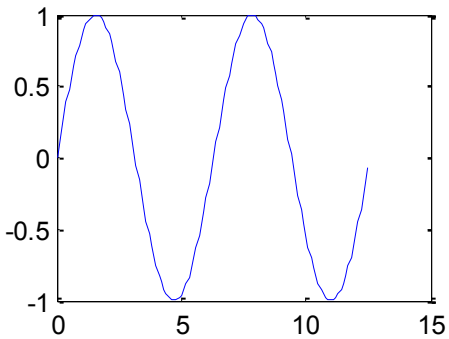
圖軸控制範例-4 (I)

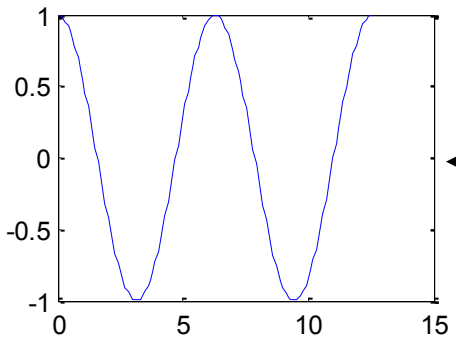
- 同時畫出四個圖於一個視窗中
  - 範例3-13：plotxy13.m

```
x = 0:0.1:4*pi;
subplot(2, 2, 1); plot(x, sin(x));                  % 此為左上角圖形
subplot(2, 2, 2); plot(x, cos(x));                  % 此為右上角圖形
subplot(2, 2, 3); plot(x, sin(x).*exp(-x/5));       % 此為左下角圖形
subplot(2, 2, 4); plot(x, x.^2);                    % 此為右下角圖形
```

圖軸控制範例-4 (II)

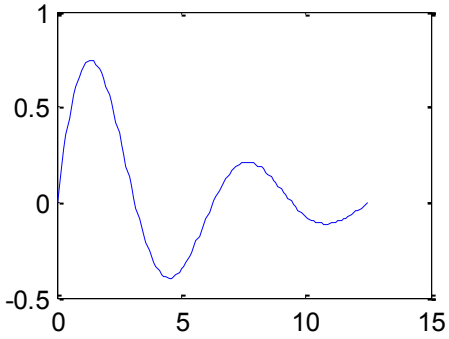
Subplot(2,2,1) →

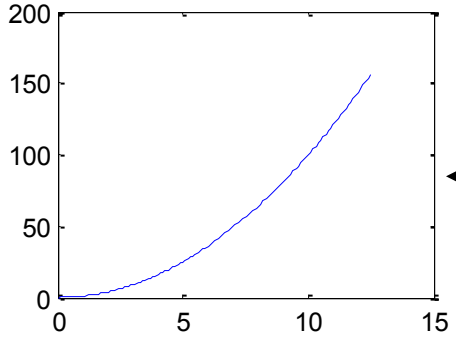




← Subplot(2,2,2)

Subplot(2,2,3) →





← Subplot(2,2,4)

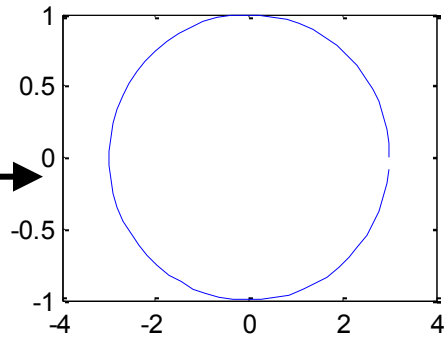
圖軸控制範例-5 (I)

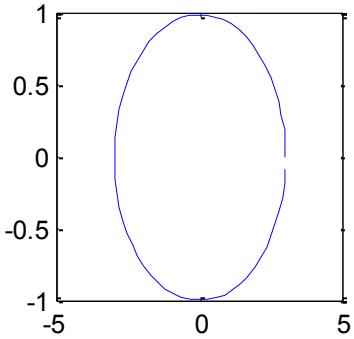
- 長寬比(Aspect Ratio)
  - 一般圖軸長寬比是視窗的長寬比
  - 可在 axis 指令後加不同的字串來修改
  - 範例3-14：plotxy14.m

```
t = 0:0.1:2*pi;
x = 3*cos(t);
y = sin(t);
subplot(2, 2, 1); plot(x, y); axis normal
subplot(2, 2, 2); plot(x, y); axis square
subplot(2, 2, 3); plot(x, y); axis equal
subplot(2, 2, 4); plot(x, y); axis equal tight
```

圖軸控制範例-5 (II)

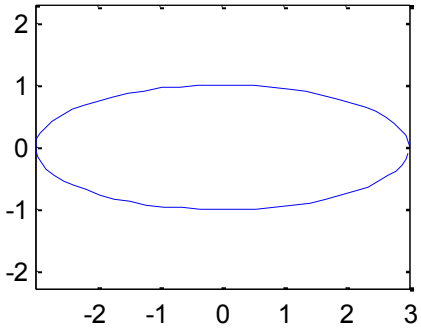
axis normal →

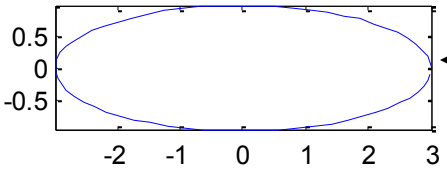




← axis square

axis equal →





← axis square tight

改變圖軸長寬比的指令

- 改變目前圖軸長寬比的指令
- 需在 plot 指令之後呼叫才能發揮效用

指令	說明
axis normal	使用預設長寬比(等於圖形長寬比)
axis square	長寬比例為 1
axis equal	長寬比例不變，但兩軸刻度一致
axis equal tight	兩軸刻度比例一致，且圖軸貼緊圖形
axis image	兩軸刻度比例一致(適用於影像顯示)

改變圖軸背景顏色的指令

- colordef
  - 改變圖軸與視窗之背景顏色
  - 先呼叫 colordef 指令，其後 plot 指令產生的圖形才有效用

指令	說明
colordef white	圖軸背景為白色，視窗背景為淺灰色
colordef black	圖軸背景為黑色，視窗背景為暗灰色
colordef none	圖軸背景為黑色，視窗背景為黑色(這是 MATLAB 第 4 版的預設值)

GRID 和 BOX 指令

- 畫出格線或畫出圖軸外圍的方形

指令	說明
grid on	畫出格線
grid off	取消格線
box on	畫出圖軸的外圍長方形
box off	取消圖軸的外圍長方形

3-4 加入說明文字

- 在圖形或圖軸加入說明文字，增進整體圖形的可讀性

指令	說明
title	圖形的標題
xlabel	x 軸的說明
ylabel	y 軸的說明
zlabel	z 軸的說明(適用於立體繪圖)
legend	多條曲線的說明
text	在圖形中加入文字
gtext	使用滑鼠決定文字的位置

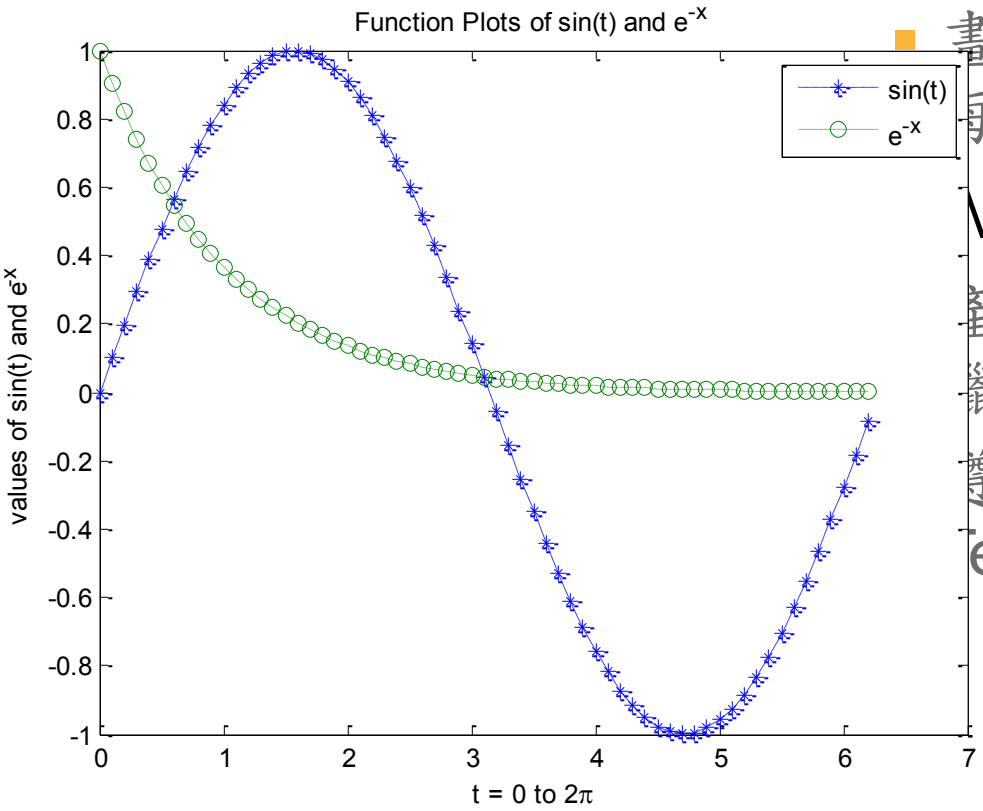
說明文字範例-1 (I)

- 範例3-15：plotxy15.m

```
subplot(1,1,1);
x = 0:0.1:2*pi;
y1 = sin(x);
y2 = exp(-x);
plot(x, y1, '--*', x, y2, ':o');
xlabel('t = 0 to 2\pi');
ylabel('values of sin(t) and e^{-x}')
title('Function Plots of sin(t) and e^{-x}');
legend('sin(t)','e^{-x}');
```

說明文字範例-1 (II)

- legend 指令
  - 畫出一小方塊，包含每條曲線的說明
  - 「\」為特殊符號
  - 產生上標、下標、希臘字母、數學符號等
  - 遵循一般 LaTex 或 TeX數學模式



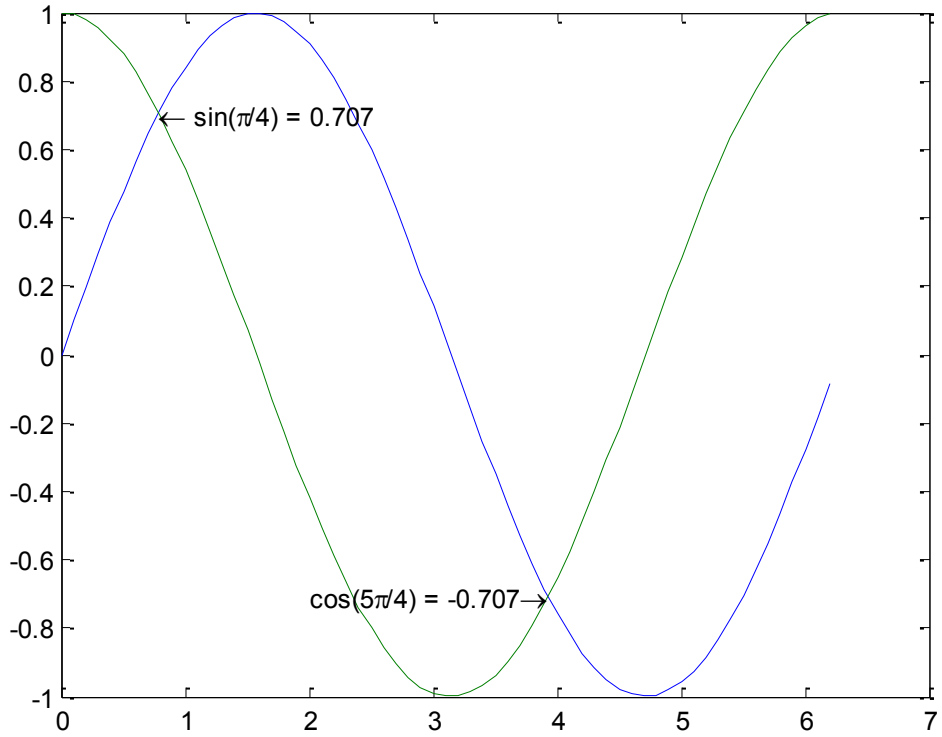
說明文字範例-2 (I)

- text指令
  - 使用語法：

    text(x, y, string)
  - x、y ：文字的起始座標位置
  - string ：代表此文字
  - 範例3-16：plotxy16.m

```
x = 0:0.1:2*pi;
plot(x, sin(x), x, cos(x));
text(pi/4, sin(pi/4),'\leftarrow sin(\pi/4) = 0.707');
text(5*pi/4, cos(5*pi/4),'cos(5\pi/4) = -0.707\rightarrow',
'HorizontalAlignment', 'right');
```

說明文字範例-2 (II)



- 「HorizontalAlignment」及「right」指示 text 指令將文字向右水平靠齊

GTEXT指令

- 使用語法

  gtext(string)
- 在圖上點選一位置後，string 顯示在其上。
- gtext 只能用在二維平面繪圖

3-5 其他平面繪圖指令

- 各種二維繪圖指令

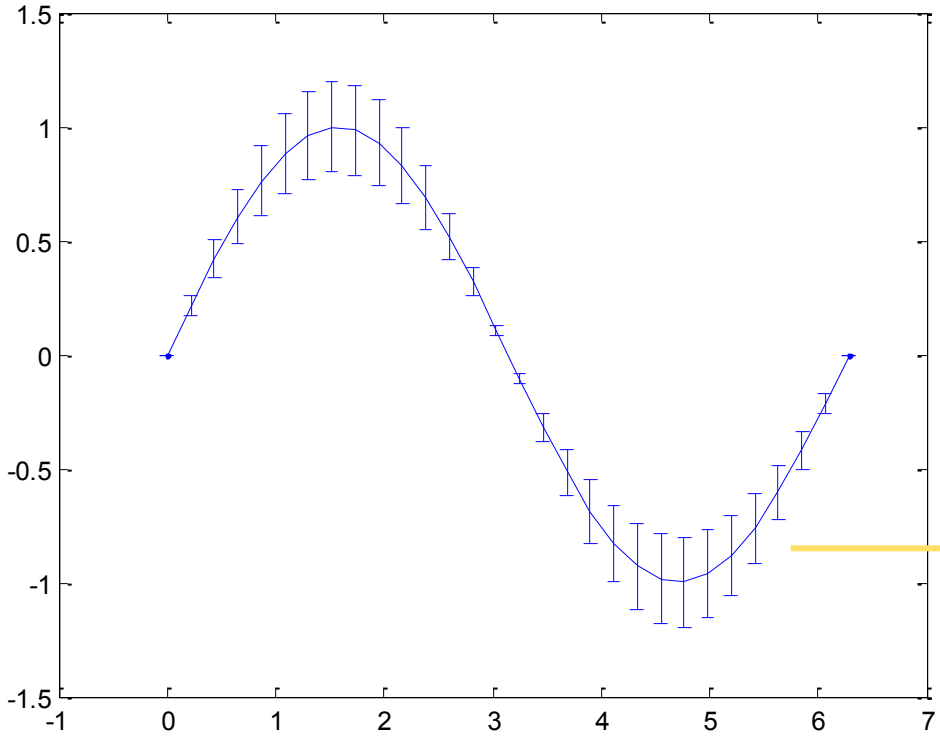
指令	說明
errorbar	在曲線加上誤差範圍
fplot、ezplot	較精確的函數圖形
polar、ezpolar	極座標圖形
hist	直角座標質方圖(累計圖)
rose	極座標質方圖(累計圖)
compass	羅盤圖
feather	羽毛圖
area	面積圖(第五章「特殊圖形」介紹)
stairs	階梯圖(第五章「特殊圖形」介紹)

其他平面繪圖範例-1 (II)

- 已知資料的誤差範圍，用 errorbar 表示
  - 以 y 座標高度 20% 作為做資料的誤差範圍
  - 範例3-17：plotxy17.m

```
x = linspace(0,2*pi,30);    % 在 0 到 2 間，等分取 30 個點
y = sin(x);
e = y*0.2;
errorbar(x,y,e)             % 圖形上加上誤差範圍 e
```

其他平面繪圖範例-1 (II)



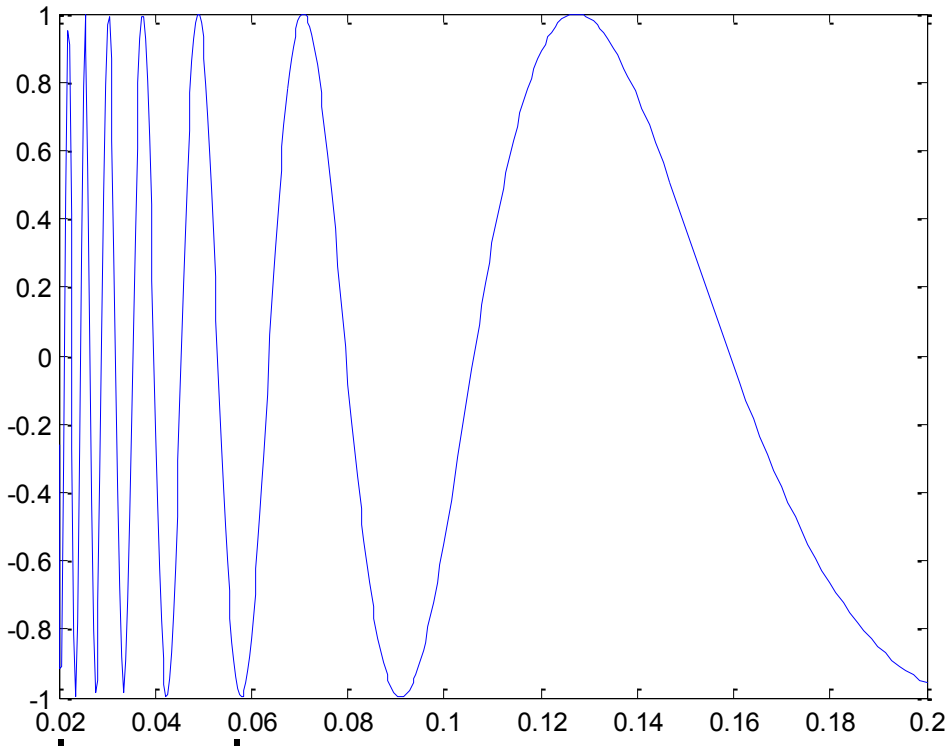
誤差範圍

其他平面繪圖範例-2 (I)

- fplot 指令
  - 對劇烈變化處進行較密集的取樣
  - 範例3-18：plotxy18.m

```
fplot('sin(1/x)', [0.02 0.2]);          % [0.02 0.2]是繪圖範圍
```

其他平面繪圖範例-2 (II)



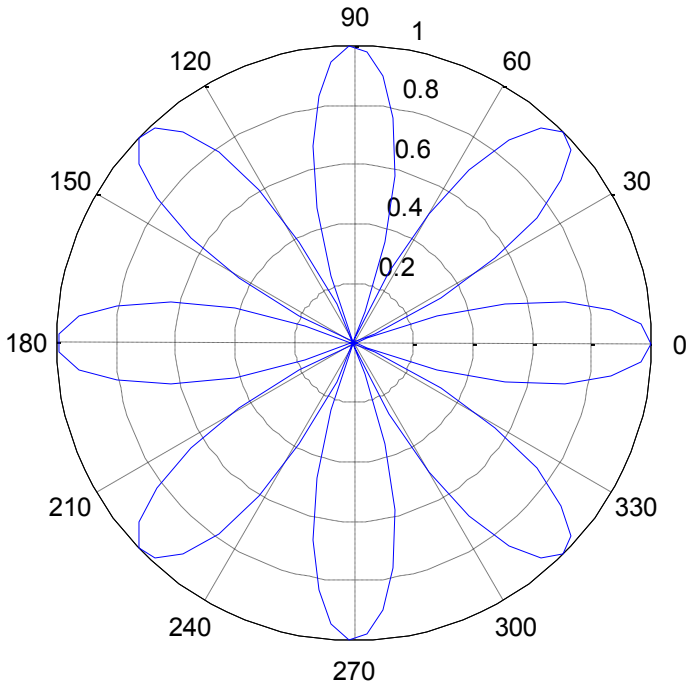
此區作較精確的取點繪圖

其他平面繪圖範例-3 (I)

- polar 指令
  - 產生極座標圖形
  - 範例3-19：plotxy19.m

```
theta = linspace(0, 2*pi);
r = cos(4*theta);
polar(theta, r);                    % 進行極座標繪圖
```

其他平面繪圖範例-3 (II)



直方圖及HIST指令

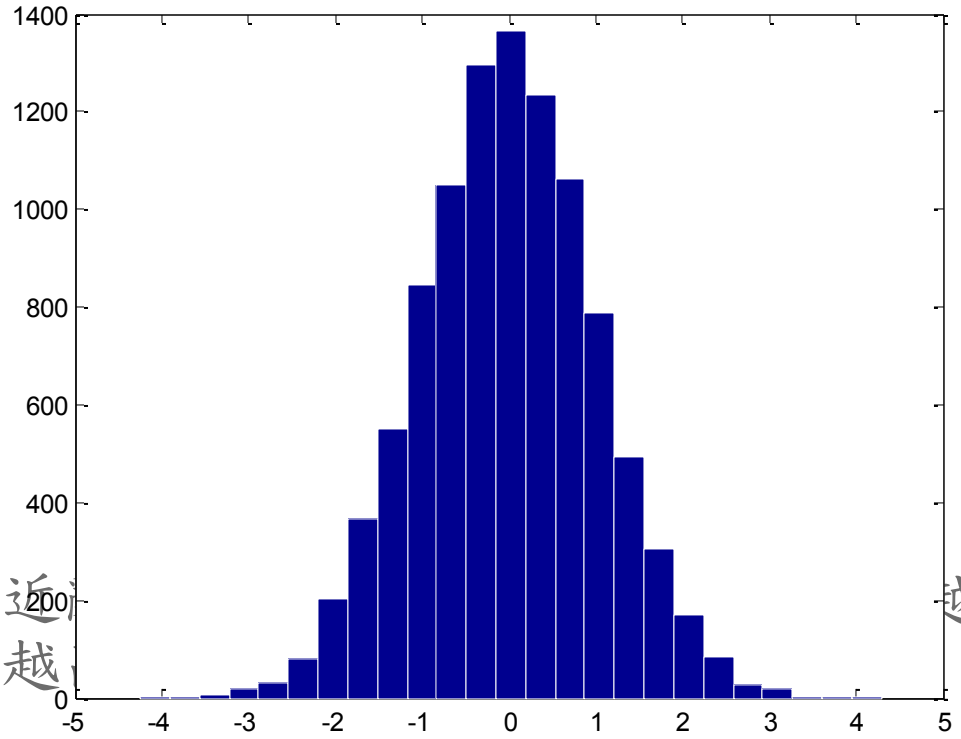
- 直方圖(Histogram)
  - 對大量的資料，顯示資料的分佈情況和統計特性
- hist指令
  - 將資料依大小分成數堆，將每堆的個數畫出

其他平面繪圖範例-4 (I)

- 將 10000 個由 randn 產生的正規分佈之亂數分成 25 堆
- 範例3-20：plotxy20.m

```
x = randn(10000, 1);          % 產生 10000 個正規分佈亂數
hist(x, 25);                  % 繪出直方圖，顯示 x 資料的分佈情
                              %況和統計特性，數字 25 代表資料依
                              %大小分堆的堆數，即是指方圖內長條
                              %的個數
set(findobj(gca, 'type', 'patch'), 'edgecolor', 'w');        % 將長條
                              %圖的邊緣設定成白色
```

其他平面繪圖範例-4 (II)



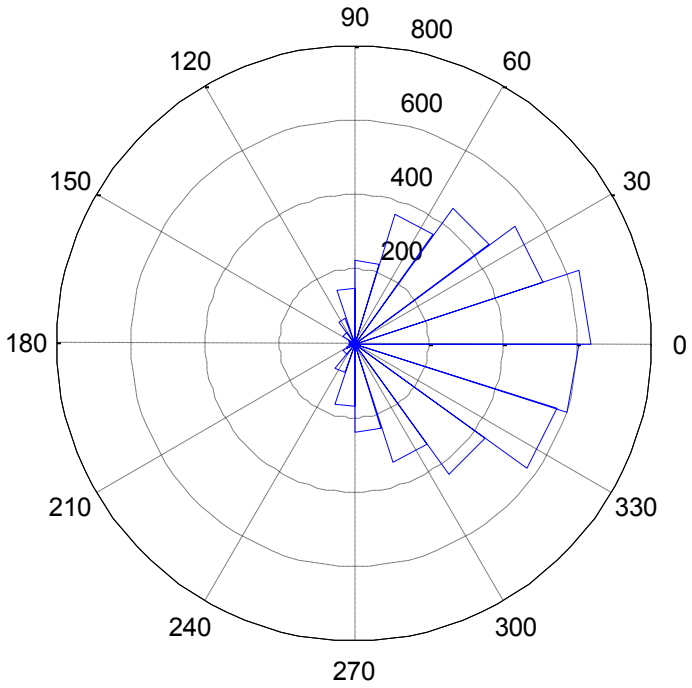
- 質方圖逼近[illegible]越大時，逼近程度越[illegible]

其他平面繪圖範例-5 (I)

- rose指令
  - 角度：資料大小
  - 距離：資料個數
  - 以極座標繪製表示
  - 範例3-21：plotxy21.m

```
x = randn(5000, 1);          % 產生 5000 個正規分佈的亂數
rose(x);                     % x 資料大小為角度，x 資料個數為距離，
                             %以繪製類似玫瑰花瓣的極座標直方圖
```

其他平面繪圖範例-5 (II)

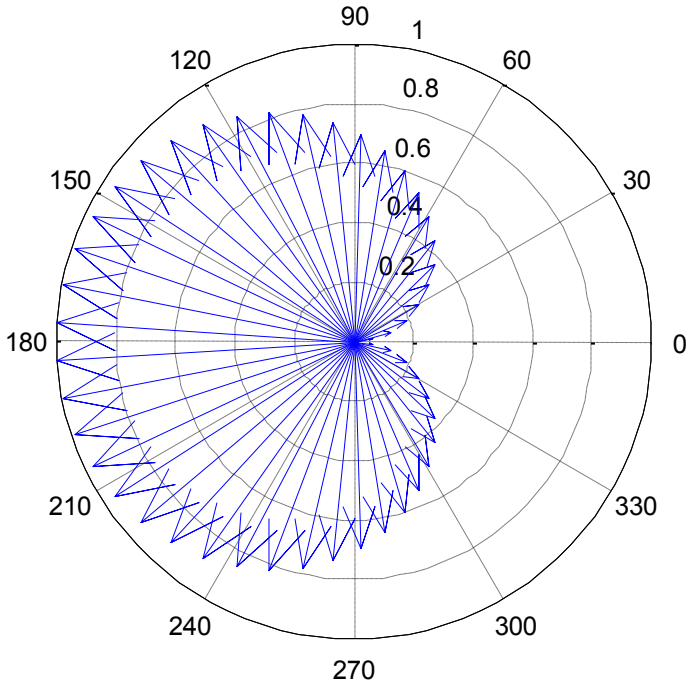


其他平面繪圖範例-6 (I)

- compass 指令
  - 畫出以原點為起始點的向量圖
  - 稱為「羅盤圖」
  - 範例3-22：plotxy22.m

```
theta = linspace(0, 2*pi, 50);
rho = sin(0.5*theta);
[x, y] = pol2cart(theta, rho);        % 由極座標轉換至直角座標
compass(x, y);                        % 畫出以原點為向量起始點
                                      % 的羅盤圖
```

其他平面繪圖範例-6 (II)



- 範例3-23：plotxy23.m

- 若只有一個引數輸入z
  - 將 z 的實部做為 x 座標，將 z 的虛部做為 y 座標，再進行作圖
  - compass(z) 即等效於 compass(real(z),imag(z))
  - 上述四列程式碼可簡化

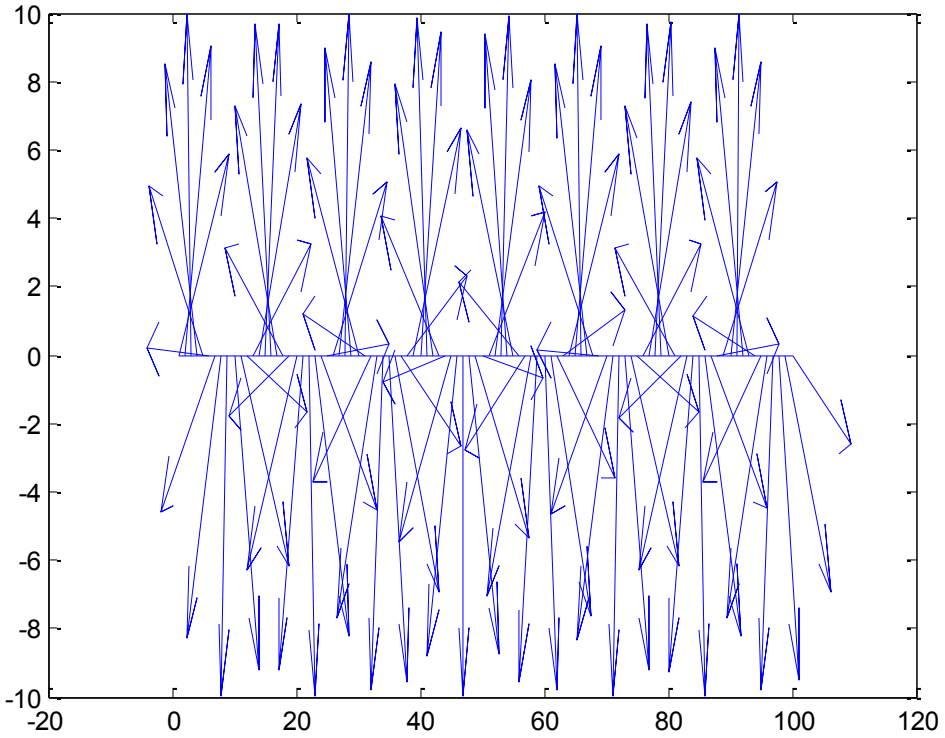
```
theta = linspace(0, 2*pi, 50);

compass(sin(0.5*theta).*exp(j*theta));
```

其他平面繪圖範例-7 (II)

- 羽毛圖
  - 起始點是 (k, 0)，k = 1~n ，其中 n 是向量個數
  - 範例3-24：plotxy24.m

```
theta = linspace(0.2*pi,50);
rho = 10;
[x, y] = pol2cart(theta,rho);          % 由極座標轉換至直角座標
feather(x, y);                         % 繪製羽毛圖
```

其他平面繪圖範例-8 (I)

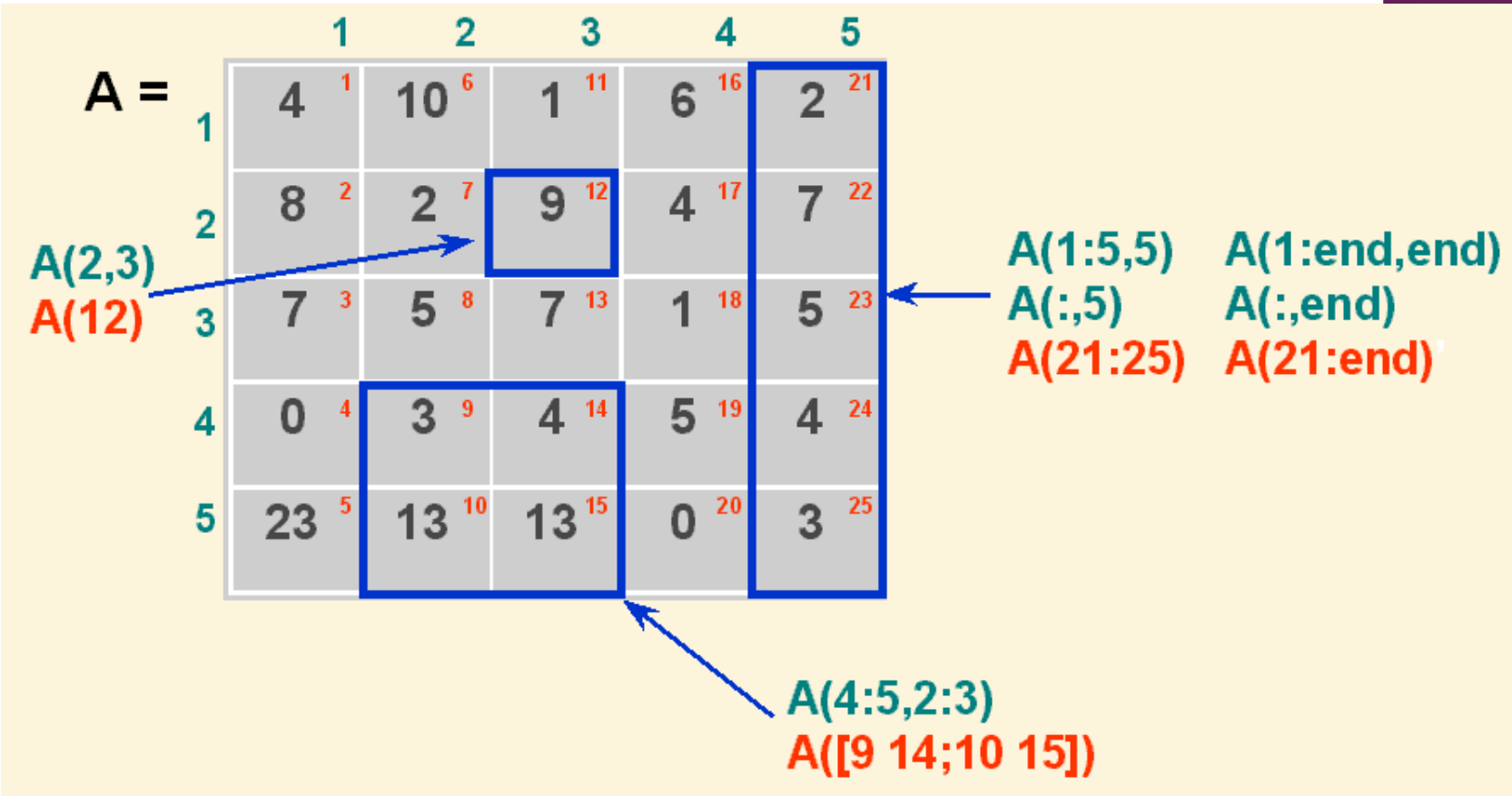


矩陣的處理與運算

9-1 矩陣的索引或下標

- 矩陣 A 中，位於第 i 橫列、第 j 直行的元素可表示為 A(i, j)
  - i 與 j 即是此元素的下標（Subscript）或索引（Index）
- MATLAB 中，所有矩陣的內部表示法都是以直行為主的一維向量
  - A(i, j) 和 A(i+(j-1)*m) 是完全一樣的，其中m為矩陣A的列數
- 我們可以使用一維或二維下標來存取矩陣

矩陣的索引或下標


A =
1 2 3 4 5
1 4 1 10 6 1 11 6 16 2 21
2 8 2 2 7 9 12 4 17 7 22
3 7 3 5 8 7 13 1 18 5 23
4 0 4 3 9 4 14 5 19 4 24
5 23 5 13 10 13 15 0 20 3 25
A(2,3)
A(12)
A(1:5,5)
A(:,5)
A(21:25)
A(1:end,end)
A(:,end)
A(21:end)
A(4:5,2:3)
A([9 14;10 15])

矩陣的索引或下標

- 可以使用矩陣下標來進行矩陣的索引（Indexing）
  - A(4:5,2:3) -取出矩陣 A 的 第四、五 橫列與 二、三 直行所形成的部份矩陣
  - A([9 14; 10 15]) - 用一維下標的方式來達到同樣目的
- 用冒號（:），取出一整列或一整行
  - A(:, 5) -取出矩陣 A 的第五個直行
- 用 end 這個保留字來代表某一維度的最大值
  - A(:, end) - 矩陣 A 的最後一個直行
- 可以直接刪除矩陣的某一整個橫列或直行
  - A(2, :) = [] – 刪除A矩陣的第二列
  - A(:, [2 4 5]) = [] - 刪除 A 矩陣的第二、四、五直行

矩陣的索引或下標

- 可把矩陣 A 和其倒數「並排」起來，得到新矩陣 B
  - B = [A 1./A]　　% 1./A 是矩陣 A 每個元素的倒數
- 用 diag 指令取出矩陣的對角線各元素
  - D = diag(B) % 取出矩陣 B 的對角線元素
  - D = diag(v) % 產生以向量 v 為主對角線的方陣
  - E = A*diag(v)　　% 將矩陣A的每個行向量乘上向量v的元素
  - E = diag(v)*A　　% 將矩陣A的每個列向量乘上向量v的元素
- 用 reshape 指令來改變一個矩陣的維度
  - B = B(1:4, 1:4);
  - C = reshape(B, 2, 8) % 將矩陣 B 排成 2×8 的新矩陣 C
  - MATLAB 會先將矩陣 B 排成一個行向量（即 MATLAB 內部的矩陣表示法），再將此行向量塞成 2×8 的新矩陣

9-2 特殊用途矩陣

- 產生各種特殊用途矩陣的好用指令：

指令	說明
zeros(m, n)	產生維度為 m×n ，構成元素全為 0 的矩陣
ones(m, n)	產生維度為 m×n ，構成元素全為 1 的矩陣
eye(n)	產生維度為 n×n ，對角線的各元素全為 1 ，其他各元素全為 0 的單位矩陣
pascal(m, n)	產生維度為 m×n 的 Pascal 矩陣
vander(v)	產生 Vandermonde 矩陣，其中每一個行向量都是向量 v 的冪次
hilb(n)	產生維度為 n×n 的 Hilbert 矩陣
rand(m, n)	產生均勻分佈於 [0, 1] 的亂數矩陣，其維度為 m×n
randn(m, n)	產生 μ = 0, σ= 1 的正規分佈亂數矩陣，其維度為 m×n
magic(n)	產生維度為 n×n 的魔方陣，其各個直行、橫列及兩對角線的元素和都相等

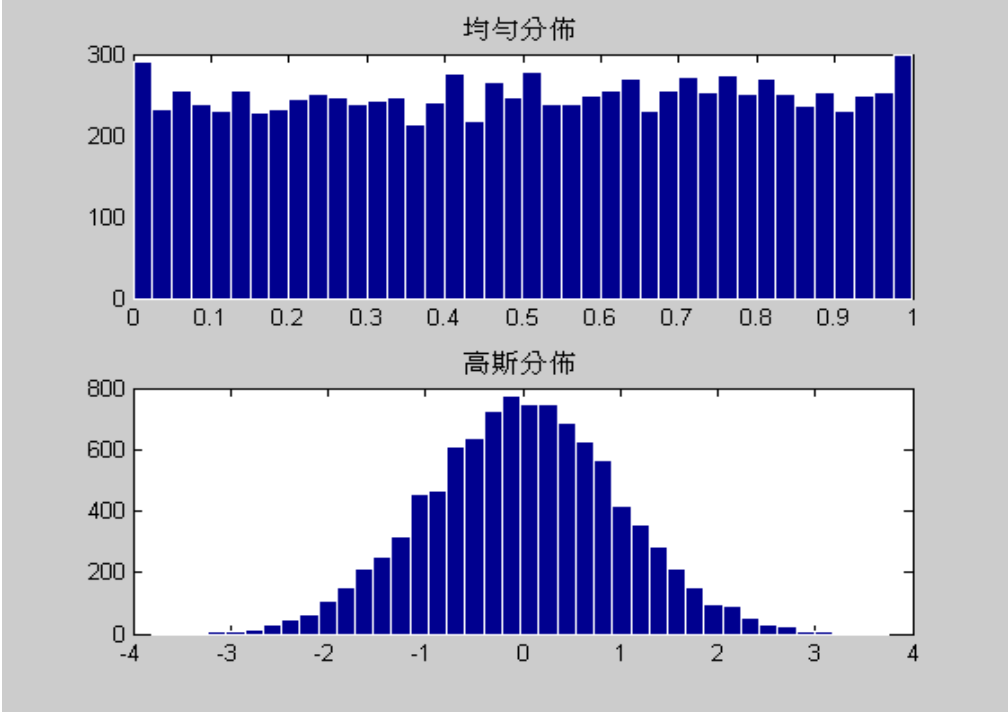
HILBERT矩陣 AND 魔方陣

- hilb(n) 指令可以產生 n×n 的 Hilbert 矩陣
  - Hilbert 矩陣的特性：當矩陣變大時，其反矩陣會接近 Singular（即矩陣的行列式會接近於 0）
  - Hilbert 矩陣常被用來評估各種反矩陣計算方法的穩定性
- magic(n) 可以產生一個 n×n 的魔方陣（Magic Matrix），
  - 其各個直行、橫列及兩對角線的元素值總和都相等

均勻和高斯分布

- rand 指令及 randn 指令則常用於產生亂數矩陣
- 範例9-11: matrix11.m

```
x1 = rand(10000, 1);
x2 = randn(10000, 1);
subplot(2,1,1); hist(x1, 40); title('均勻分佈');
subplot(2,1,2); hist(x2, 40); title('高斯分佈');
set(findobj(gcf, 'type', 'patch'), … 'EdgeColor',
'w');          % 改邊線為白色
```



9-3 矩陣的數學運算

- 矩陣的加減與一般純量（Scalar）的加減類似
- 相加或相減的矩陣必需具有相同的維度
- 範例9-12: matrix12.m

```
A = [12 34 56 20];
B = [1 3 2 4];
C = A + B
```

```
C =
   13   37   58   24
```

- 矩陣與純量可以直接進行加減，MATLAB 會直接將加減應用到每一個元素

```
>> A = [1 2 3 2 1] + 5
A =
    6    7    8    7    6
```

矩陣的乘法與除法

- 純量對矩陣的乘或除，可比照一般寫法

```
>> A = [123 , 442];     >> C = A/3
>> B = 2*A                          C =
B =                                    41.0000  147.3333
   246   884
```

- 欲進行矩陣相乘，必需確認第一個矩陣的直行數目（ Column Dimension ） 必需等於第二個矩陣的橫列數目（ Row Dimension ）
- 範例9-13: matrix12.m

```
A = [1; 2];
B = [3, 4, 5];
C = A*B
```

```
C =
   3    4    5
   6    8   10
```

- 矩陣的除法，常藉由反矩陣或解線性方程式來達成

矩陣的次方運算

- 矩陣的次方運算，可由「^」來達成，但矩陣必需是方陣，其次方運算才有意義
- 範例9-14: matrix14.m

```
A = magic(3);
B = A^2
```

```
B =
    91    67    67
    67    91    67
    67    67    91
```

- 在「*」，「/」及「^」之前加上一個句點，MATLAB 將會執行矩陣內「元素對元素」（Element-by-element） 的運算

```
A = [12; 45];
B = [2; 3];
C = A.*B                % 注意「*」前面的句點
D = A./B                % 注意「/」前面的句點
E = A.^2                % 注意「^」前面的句點
```

轉置和「共軛轉置」矩陣

- 複數矩陣 z，其「共軛轉置」矩陣（Conjugate Transpose）可表示成矩陣 z'
- 範例9-16: conjTranspose01.m

```
i = sqrt(-1);                    % 單位虛數
z = [1+i, 2; 3, 1+2i];
w = z'                           % 共軛轉置（注意 z 後面的單引號）
```

```
w =
   1.0000-1.0000i   3.0000
   2.0000           1.0000-2.0000i
```

- 想得到任何矩陣 z 的轉置（Transpose），則可表示成矩陣 z.'
- 範例9-17: transpose01.m

```
i = sqrt(-1);                    % 單位虛數
z = [1+i, 2; 3, 1+2i];
w = z.'                          % 單純轉置（注意 z 後面的句點及單引號）
```

```
w =
   1.0000+1.0000i   3.0000
   2.0000           1.0000+2.0000i
```

- 若 z 為實數，則 z' 和 z.' 的結果是一樣的

向量的Lp-NORM

- 一個向量 a 的 Lp-norm 可以定義為

$$\|a\|_p = \left(\sum_i |a_i|^p \right)^{1/p}, p \geq 1$$

  - p=1 ➔ taxicab distance, or Manhattan distance
  - p=2 ➔ Euclidean Length (length of a)
  - P=inf ➔ max. distance
- 欲求一向量的 p-norm，可使用 norm 指令 norm(x,p)
- 範例9-18: normVector01.m

```
a = [3 4];
x = norm(a, 1)          % x = 7
y = norm(a, 2)          % y = 5
z = norm(a, inf)        % z = 4
```

$$norm\ (a,1) = \sum_i |a_i|$$

$$norm\ (a,\inf) = \max_i |a_i|$$

矩陣的Lp-NORM

- 一個矩陣 A 的 Lp-norm 可以定義如下：

$$\|A\|_p = \max_x \frac{\|Ax\|_p}{\|x\|_p}$$

- norm 指令亦可用於計算矩陣的 Lp-norm
- 範例9-19: normMatrix01.m

```
A = [1 2 3; 4 5 6; 7 8 9];
norm(A, 2)                          % ans = 16.8481
```

- MATLAB 有相當完整的數學函數, 三角函數還有計算向量元素統計量的函數(課本 9-15~9-17)

SORT指令

- sort 指令可對向量元素進行排序（Sorting）
- 範例9-20: sort01.m

```
x = [3 5 8 1 4];
[sorted, index] = sort(x)                    % 對矩陣 x 的元素進行排序
```

```
sorted =
    1    3    4    5    8
index =
    4    1    5    2    3
```

- sorted 是排序後的向量，index 則是每個排序後的元素在原向量 x 的位置，因此 x(index) 即等於 sorted 向量。
- 如何使用 sort 指令加上前例中的 sorted 及 index 來求得原先的向量 x？

矩陣的最大元素

- 找出一矩陣最大元素的位置
- 範例9-21: max01.m

```
x = magic(5);
[colMax, colMaxIndex] = max(x)
```

```
colMax =
    23    24    25    21    22
colMaxIndex =
     2     1     5     4     3
```

- colMax 代表每一直行的最大值，colMaxIndex 則是每一直行出現最大值的位置
- 求得 x 的最大元素的位置
- 範例9-22: max02.m

```
x = magic(5);
[colMax, colMaxIndex] = max(x);
[maxValue, maxIndex] = max(colMax);
fprintf('Max value = x(%d, %d) = %d\n', colMaxIndex(maxIndex), maxIndex, maxValue);
```

```
Max value = x(5, 3) = 25
```

- x 的最大元素即是 maxValue，發生位置為 [colMaxIndex(maxIndex), maxIndex] = [5 , 3]
- 若只要找出一矩陣 x 的最大值，可輸入 max(max)或是 max(x(：))

9-4 矩陣的內部資料型態

- 一般矩陣的內部資料型態都是 double（雙精準浮點數），但在 MATLAB 5.3 版之後，也支援不同長度的整數與浮點數資料態

指令	說明
uint8	轉換成帶正負號 、 8 位元的整數，其值域為 [-128,127]
uint16	轉換成帶正負號、16 位元的整數，其值域為 [-32768,32767]
uint32	轉換成帶正負號、32 位元的整數，其值域為 [-2^31,2^31-1]
int8	轉換成不帶正負號、 8 位元的整數，其值域為 [0,255]
int16	轉換成不帶正負號 、16 位元的整數，其值域為 [0,65535]
int32	轉換成不帶正負號 、32 位元的整數，其值域為 [0,2^32-1]
single	轉換成 single（單精準浮點數），佔用 32 位元（4 bytes）
double	轉換成double（雙精準浮點數），佔用 64 位元（8 bytes）
char	轉換成字元或字串，每個字元佔用（16 位元）（2 bytes）

不同資料的儲存

- 我們要節省記憶體空間，可以依矩陣元素值的範圍，選用不同的資料來儲存
- 範例9-23: datatype01.m

```
clear all                                   % 清除所有工作空間的變數
x_double = magic(10);
x_single = single(x_double);
x32 = uint32(x_double);
x16 = uint16(x_double);
x8 = uint8(x_double);
whos
```

```
Name          Size            Bytes  Class
  x16         10x10             200  uint16 array
  x32         10x10             400  uint32 array
  x8          10x10             100  uint8 array
  x_double    10x10             800  double array
  x_single    10x10             400  single array
Grand total is 500 elements using 1900 bytes
```

- uint8 來儲存變數所佔的空間只有 double 的八分之一！

資料儲存的注意事項

- 整數資料型態的範圍有限，若超過此範圍，則超出部分將會被「裁掉」

```
>> uint8(300)                    % uint8 的最大值為 255
ans =
   255
>> int8(-500)                    % int8 的最小值為 -128
ans =
  -128
```

- 整數資料型態可以比較大小，亦可直接進行數學運算，但必須注意其資料型態的自動轉換：

```
>> uint8(20)== 20                % 可比較大小
ans =
     1
>> z=uint8(magic(3))
>> z*2.5
```

(Please try it by yourself to get the conversion rule!)

- 若要進行精準的數學運算，需先用 double 指令將整數型態之變數轉成雙倍精準浮點數

INTERESTING DEMOS BY CLEVE

- Under “cleve” folder of this chapter…
  - eigshow.m
  - svdshow.m
  - vorodrag.m
  - vshow.m

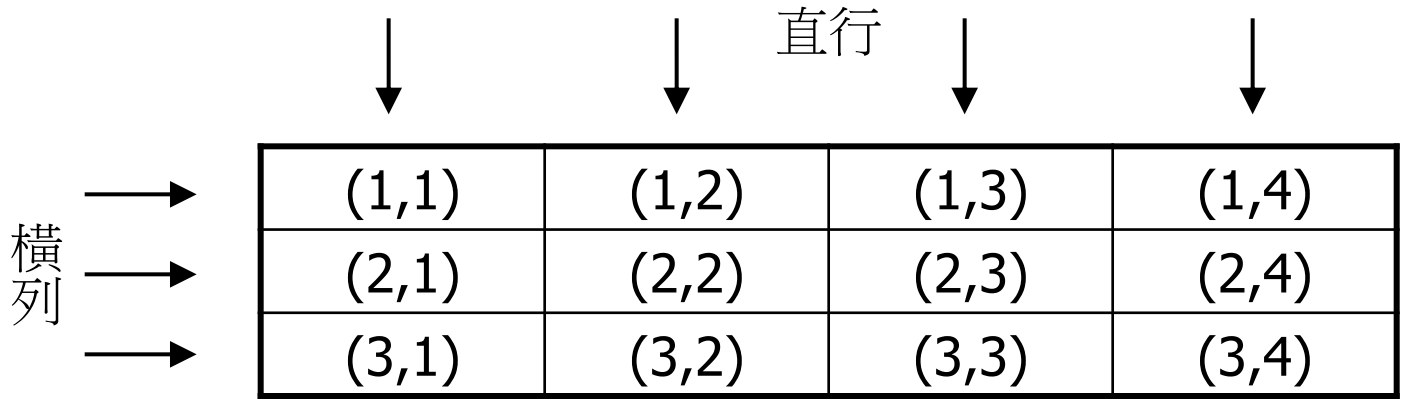
多維陣列

11-1 多維陣列的定義

- 在 MATLAB 的資料型態中：
  - 一維陣列稱為向量（Vectors）
  - 二維陣列稱為矩陣（Matrices）
  - 維度（Dimensions）超過 1 的陣列則均可視為「多維陣列」（Multidimesional Arrays，簡稱 N-D Arrays）。

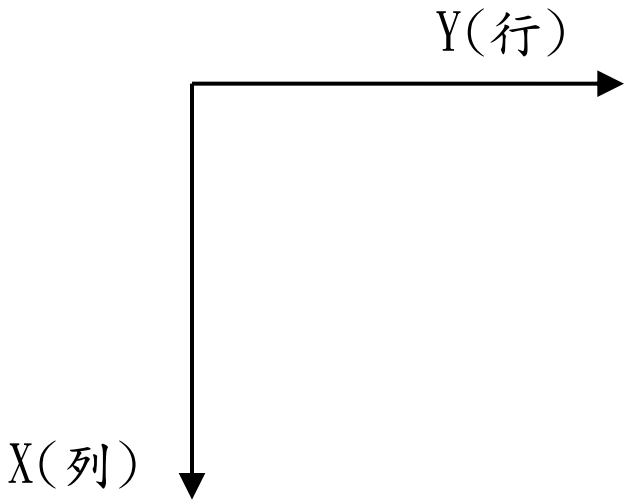
二維陣列 (I)

- 二維陣列又簡稱矩陣，具有兩個維度
  - 橫列（Row）
  - 直行（Column）



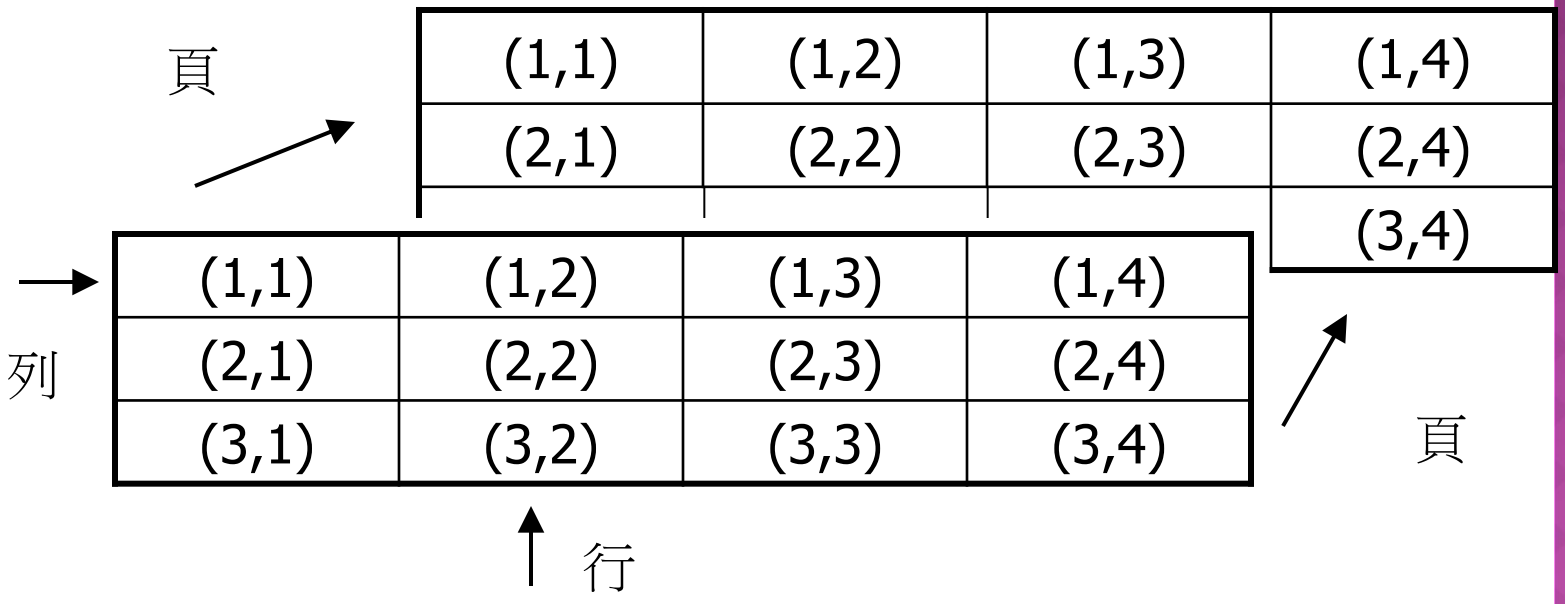
二維陣列 (II)

- 二維陣列可對應至一個 X-Y 二維平面座標，圖示如下：



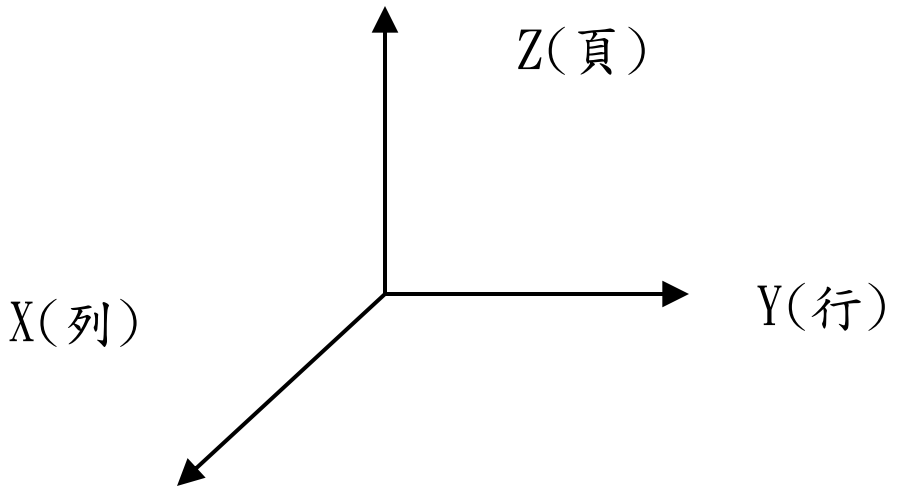
三維陣列 (I)

- 將兩個矩陣疊在一起，就形成第三個維度，此第三個維度可稱為「頁」（Page），圖示如下：



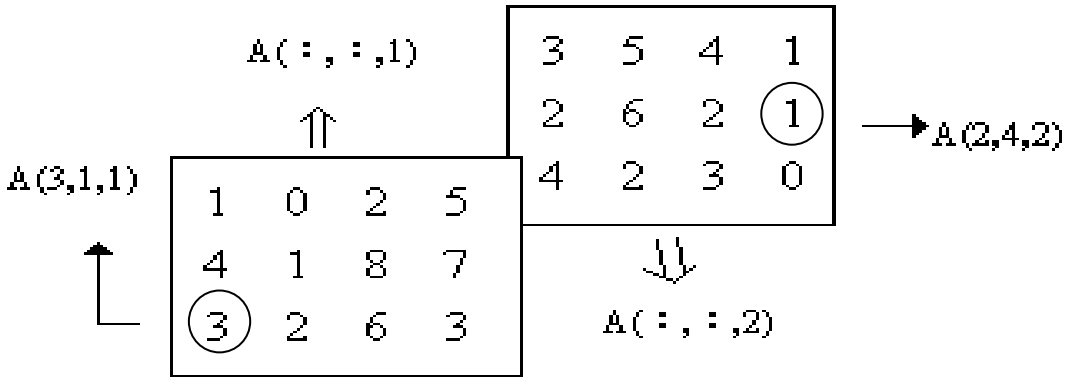
三維陣列 (II)

- 三維陣列可對應至一個 X - Y - Z 三維立體座標，圖示如下：



三維陣列 (III)

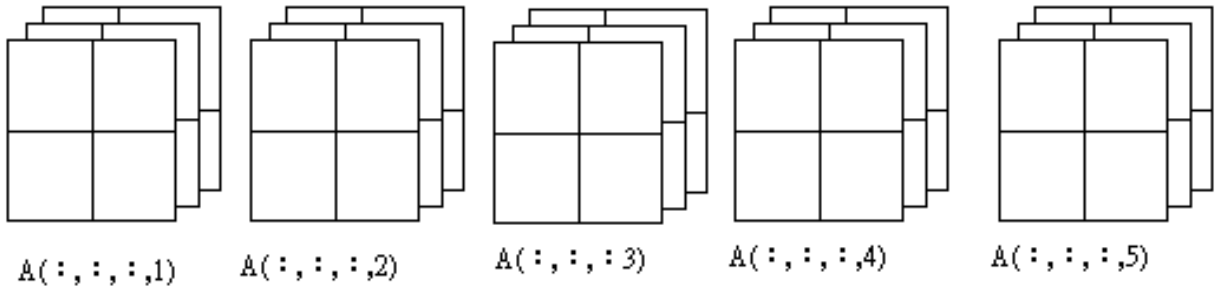
- 三維陣列定址，可以 (列,行,頁) 定之。
- 以維度為 3x4x2 的三維陣列為例，其定址方式可圖示如下：



- 陣列 A 是三維陣列，其中 A(：,：,1)代表第一頁的二維陣列，A(：,：,2)代表第二頁的二維陣列。

四維陣列

- 四維陣列的第四個維度可視為「箱」（Box），而每個箱是由一個三維陣列所組成，其定址方式為(列,行,頁,箱)。
  - 例如：一個 2×2×3×5 的四維陣列，可表示成 5 個箱，每個箱都由一個 2×2×3 的三維陣列所組成，圖示如下：



- 依此可類推至 n 維陣列，n 為任意自然數。

11-2 多維陣列的建立

- 欲建立一個簡單的多維陣列，可直接由 MATLAB 指令視窗內輸入
- 範例11-1：mDim01.m

```
A = [1 0 2 5; 4 1 8 7; 3 2 6 3];
A(:, :, 2) = [3 5 4 1; 2 6 2 1; 4 2 3 0]
```

```
A(:,:,1) =
     1     0     2     5
     4     1     8     7
     3     2     6     3
A(:,:,2) =
     3     5     4     1
     2     6     2     1
     4     2     3     0
```

多維陣列直接設定
範例11-2 (I)

- 上例是先建立一個二維陣列 A ，再輸入第二頁的二維陣列 A(：,：,2) ，如此逐頁輸入二維陣列的內容，即可建立三維陣列。
- 如果直接設定某一個新頁的一個元素值，此時 MATLAB 會將此頁其他未指定之元素直接設定為 0 ，範例如下：

多維陣列直接設定
範例11-2 (II)

- 範例11-2：mDim02.m

```
A = [1 0 2 5; 4 1 8 7; 3 2 6 3];
A(:, :, 2) = [3 5 4 1; 2 6 2 1; 4 2 3 0];
A(2, 1, 3) = 5
```

```
A(:,:,1) =
     1     0     2     5
     4     1     8     7
     3     2     6     3
A(:,:,2) =
     3     5     4     1
     2     6     2     1
     4     2     3     0
A(:,:,3) =
     0     0     0     0
     5     0     0     0
     0     0     0     0
```

純量展開功能 (SCALAR EXPANSION) (I)

- MATLAB 第五版新增純量展開 (Scalar Expansion) 功能，直接設定一個純量給多維陣列的一頁。
- 承接上述範例，若要將陣列 A 的第二頁所有元素設為 7，可輸入如下：

純量展開功能 (SCALAR EXPANSION) (II)

```
>> A(:,:,2) = 7

A(:,:,1) =
     1     0     2     5
     4     1     8     7
     3     2     6     3
A(:,:,2) =
     7     7     7     7
     7     7     7     7
     7     7     7     7
A(:,:,3) =
     0     0     0     0
     5     0     0     0
     0     0     0     0
```

垂直並排多維陣列
範例11-3 (I)

- 對於較複雜的多維陣列，可用 cat 指令來建立，其功能為「並排」數個陣列，並可指定「並排」時所用的維度
  - 其指令格式如下：

    Z = cat(dim, A, B, C...)

  - A、B、C 為陣列
  - dim 是將 A、B、C... 合併時所用到的維度。（亦即在合併後，此維度的大小會改變。）

垂直並排多維陣列
範例11-3 (II)

- 欲將矩陣 A 與 B 上下(垂直)並排
  - 範例11-3：cat01.m

```
A = [1 2; 3 4];
B = [1 0; 0 1];
Z = cat(1, A, B)      % 數字 1 表示將陣列 A 與 B 上下垂直並排
```

```
Z =

     1     2
     3     4
     1     0
     0     1
```

水平並排多維陣列
範例11-4

- 欲將陣列 A 與 B 左右(水平)並排
  - 範例11-4：cat02.m

```
A = [1 2; 3 4];
B = [1 0; 0 1];
Z = cat(2, A, B)  % 數字 2 表示將陣列 A 與 B 左右水平並排
```

```
Z =

     1     2     1     0

     3     4     0     1
```

陣列堆疊 - 範例11-5

- 將陣列 A 與 B 疊起來，得到一個三維陣列
  - 範例11-5：cat03.m

```
A = [1 2; 3 4];
B = [1 0; 0 1];
Z = cat(3, A, B)   % 數字3表示將陣列 A 與 B 重疊排成三維陣列
```

```
Z(:,:,1) =
     1     2
     3     4
Z(:,:,2) =
     1     0
     0     1
```

CAT自動補齊維度 - 範例11-6

- 所設定的 dim 值比陣列 A、B、C...的各自原先的「維度數」(Dimensionality)還要超出 2 或更多，MATLAB 會自動補上大小為 1 的維度
- 此時陣列 Z 的維度變為 2x2x1x2
  - 範例11-6：cat04.m

```
A = [1 2; 3 4];
B = [1 0; 0 1];
Z = cat(4, A, B)      % 數字 4 表示將陣列 A 與 B 放在相鄰的兩「箱」
```

```
Z(:,:,1,1) =
     1     3
     2     4

Z(:,:,1,2) =
     1     2
     1     1
```

亂數陣列 (I)

- MATLAB 可產生特殊用途的多維陣列
- 要產生一個維度是 2x3x5 的亂數陣列，可用 rand 指令

```
>> A = rand(2, 3, 5)
A(:,:,1) =
    0.3412    0.7271    0.8385
    0.5341    0.3093    0.5681
A(:,:,2) =
    0.3704    0.5466    0.6946
    0.7027    0.4449    0.6213
```

亂數陣列 (II)

```
A(:,:,3) =

    0.7948    0.5226    0.1730
    0.9568    0.8801    0.9797

A(:,:,4) =

    0.2714    0.8757    0.1365
    0.2523    0.7373    0.0118

A(:,:,5) =

    0.8939    0.2987    0.2844
    0.1991    0.6614    0.4692
```

- 類似的指令可見本書第九章「矩陣的處理與運算」的第二節「特殊用途矩陣」

11-3 多維陣列的數學運算

- 許多用於向量和矩陣的數學運算，例如 sum、max、min、mean 等，也都可以用在多維陣列。
- 在使用這些指令時，我們必須指定這些指令的操作是在哪一個維度。

多維陣列運算 維度指定
範例11-7 (I)

- 範例11-7：sum01.m

```
A = [1 1 1 1; 2 2 2 2; 3 3 3 3];
B = [0 0 0 0; 1 1 1 1; 1 2 3 4];
Z = cat(3, A, B);          % 將矩陣 A, B 疊成一個三維陣列
S = sum(Z, 1)              % 根據第一維度來對元素進行相加
size(S)
```

```
S(:,:,1) =
     6     6     6     6
S(:,:,2) =
     2     3     4     5
ans =
     1     4     2
```

多維陣列運算 維度指定
範例11-7 (II)

- 上述範例，矩陣 Z 的大小是 3x4x2，sum(Z, 1) 是根據第一個維度來進行相加，因此第一個維度值就會被被壓成是 1，因此 size(S) 所傳回的值是 [1, 4, 2]，代表矩陣 S 的大小是 1x4x2
- sum(Z) 的預設相加維度即是 1，因此 sum(Z) 和 sum(Z, 1) 所得到的結果是一樣的。

多維陣列運算 維度指定
範例11-8 (I)

- 對第二個維度進行相加，可見下列範例。
  - 範例11-8：sum02.m

```
A = [1 1 1 1; 2 2 2 2; 3 3 3 3];
B = [0 0 0 0; 1 1 1 1; 1 2 3 4];
Z = cat(3, A, B);                % 將矩陣 A, B 疊成一個三維陣列
S = sum(Z, 2)                    % 根據第二維度來對元素進行相加
size(S)
```

```
S(:,:,1) =
     4
     8
    12
```

多維陣列運算 維度指定
範例11-8 (II)

```
S(:,:,2) =
     0
     4
    10
```

- 在上述範例中，sum(Z, 2)是對第二個維度進行相加運算，因此所傳回的矩陣 S 的維度是 3×1×2。

SUM的累加－範例11-9（I）

- 執行sum(Z)時，如果預設要相加的維度只有單一維度，那麼 sum 指令會對下一個維度進行相加的動作，
- sum(sum(Z)) 將會得到三維陣列 Z 的每一頁的總和。
- sum(sum(sum(Z))) 或 sum(Z(:)) 可得到三維陣列 Z 的總和。

SUM的累加－範例11-9（II）

- 範例11-9：sumSum01.m

```
A = [1 1 1 1; 2 2 2 2; 3 3 3 3];
B = [0 0 0 0; 1 1 1 1; 1 2 3 4];
Z = cat(3, A, B);          % 將矩陣 A, B 疊成一個三維陣列
S = sum(sum(Z))
```

```
S(:,:,1) =

    24

S(:,:,2) =

    14
```

- 與 sum 類似的指令還有 max、min、mean、median、mode、std、diff、sort 等，可以參考第九章的第三小節「矩陣的數學運算」。

5-1 長條圖之繪製

- 長條圖 （Bar Graphs）特別適用於少量且離散的資料。欲畫出垂直長條圖，可用 bar 指令。

- 範例5-1：bar01.m

```
x = [1 3 4 5 2];
bar(x);
```

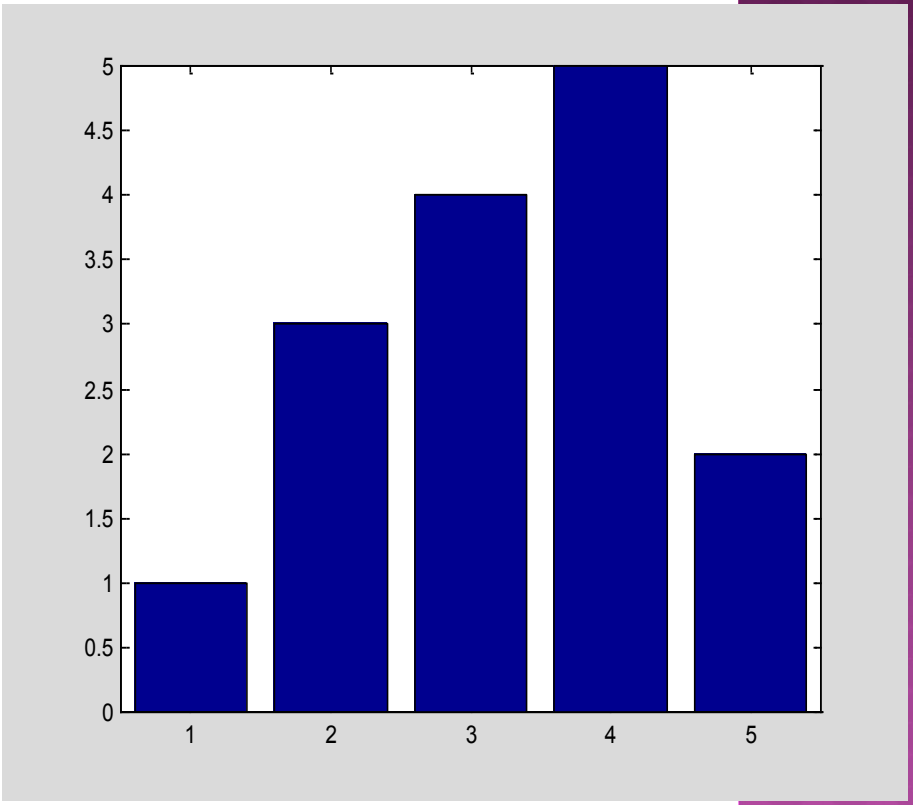


Fig. 5-1

5-1 長條圖之繪製(CONT.)

- bar 指令也可接受矩陣輸入，它會將同一橫列的資料聚集在一起。
- barh指令則可畫出水平的長條圖。

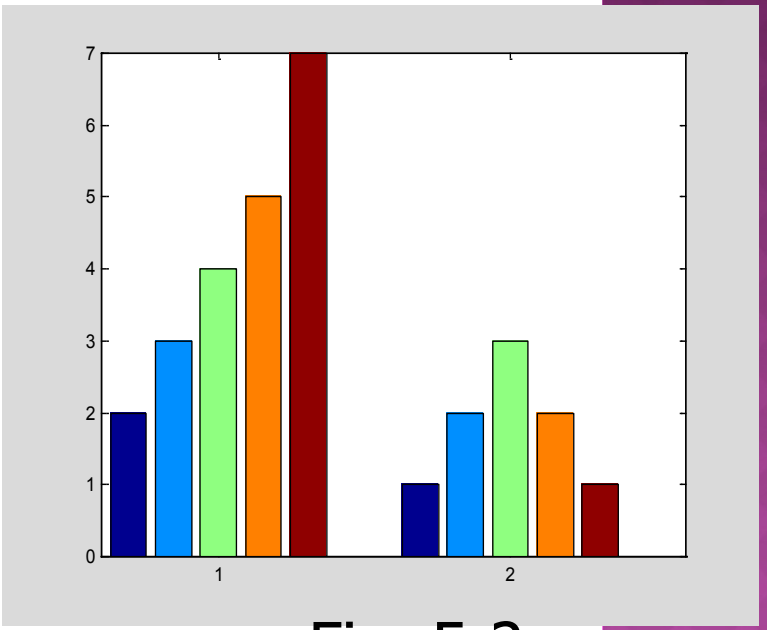


Fig. 5-2

- 範例5-2：bar02.m

```
x = [2 3 4 5 7; 1 2 3 2 1];
bar(x);
```

5-1 長條圖之繪製(CONT.)

- bar 及 barh 指令還有一項特異功能，就是可以將同一橫列的資料以堆疊（Stack）方式來顯示。

- 範例5-3：bar03.m

```
x = [2 3 4 5 7; 1 2 3 2 1];
bar(x,'stack')
```

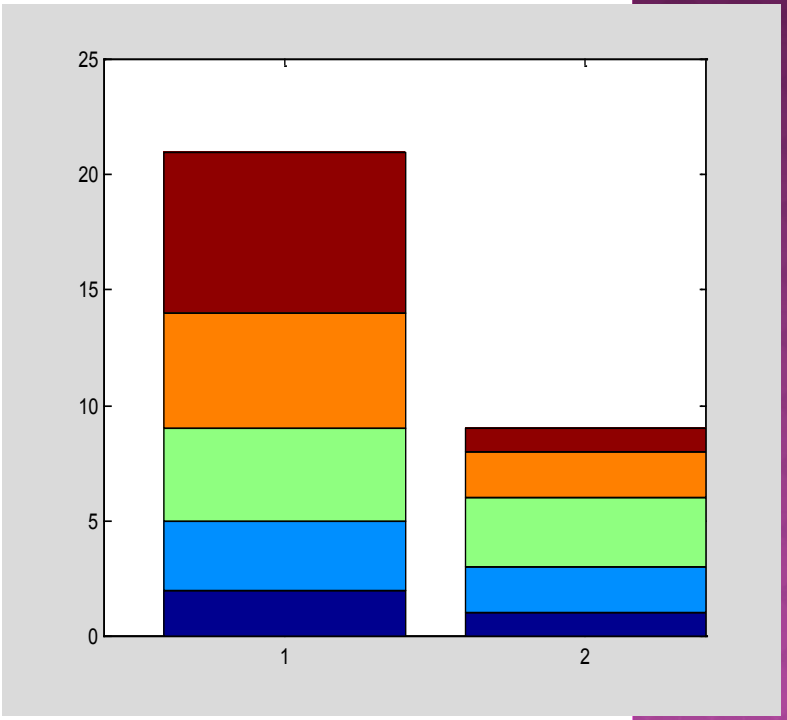


Fig. 5-3

5-1 長條圖之繪製(CONT.)

- 除了平面長條圖之外，MATLAB 亦可使用 bar3 指令來畫出立體長條圖。

- 範例5-4：bar04.m

```
x = [2 3 4 5 7; 1 2 3 2 1];
bar3(x)
```

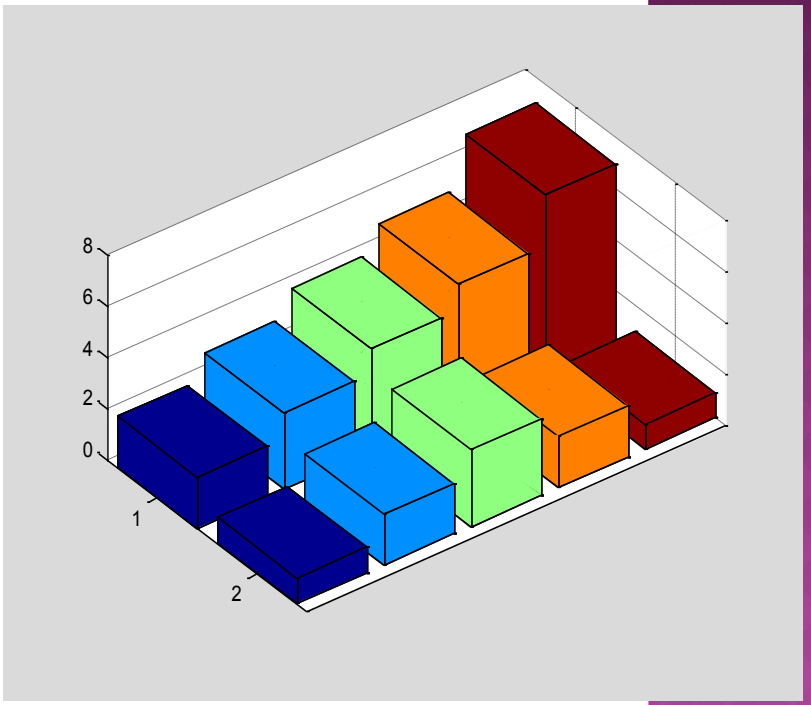


Fig. 5-4

5-1 長條圖之繪製(CONT.)

- bar3 指令還可以使用群組（Group）方式來呈現長條圖

- 範例5-5：bar05.m

```
x = [2 3 4 5 7; 1 2 3 2 1];
bar3(x, 'group')
```

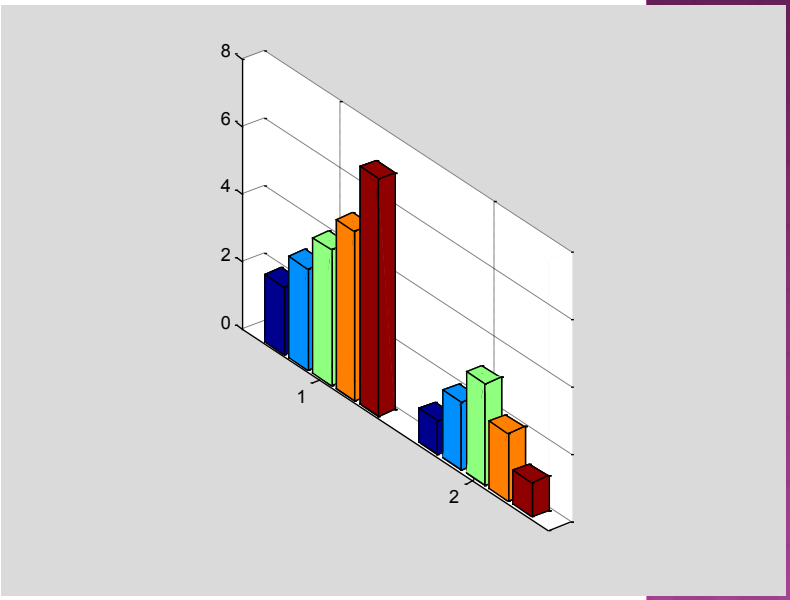


Fig. 5-5

5-1 長條圖之繪製(CONT.)

- 長條圖的指令和類別：

	垂直長條圖	水平長條圖
平面	bar	barh
立體	bar3	bar3h

5-1 長條圖之繪製(CONT.)

- 若要指定長條圖的 x 座標，可使用兩個輸入向量給 bar 指令。假設新竹的月平均溫度如下：

- 範例5-6：bar06.m

```
x = 1:6;                          % 月份
y = 35*rand(1, 6);                % 溫度值（假設是介於 0～35 的亂數）
bar(x, y);
xlabel('月份');                    % x 軸的說明文字
ylabel('平均溫度 (^{o}c)');        % y 軸的說明文字
% 下列指令將 x 軸的數字改成月數
set(gca, 'xticklabel', {'一月','二月','三月', '四月', '五月', '六月'});
```

5-1 長條圖之繪製(CONT.)

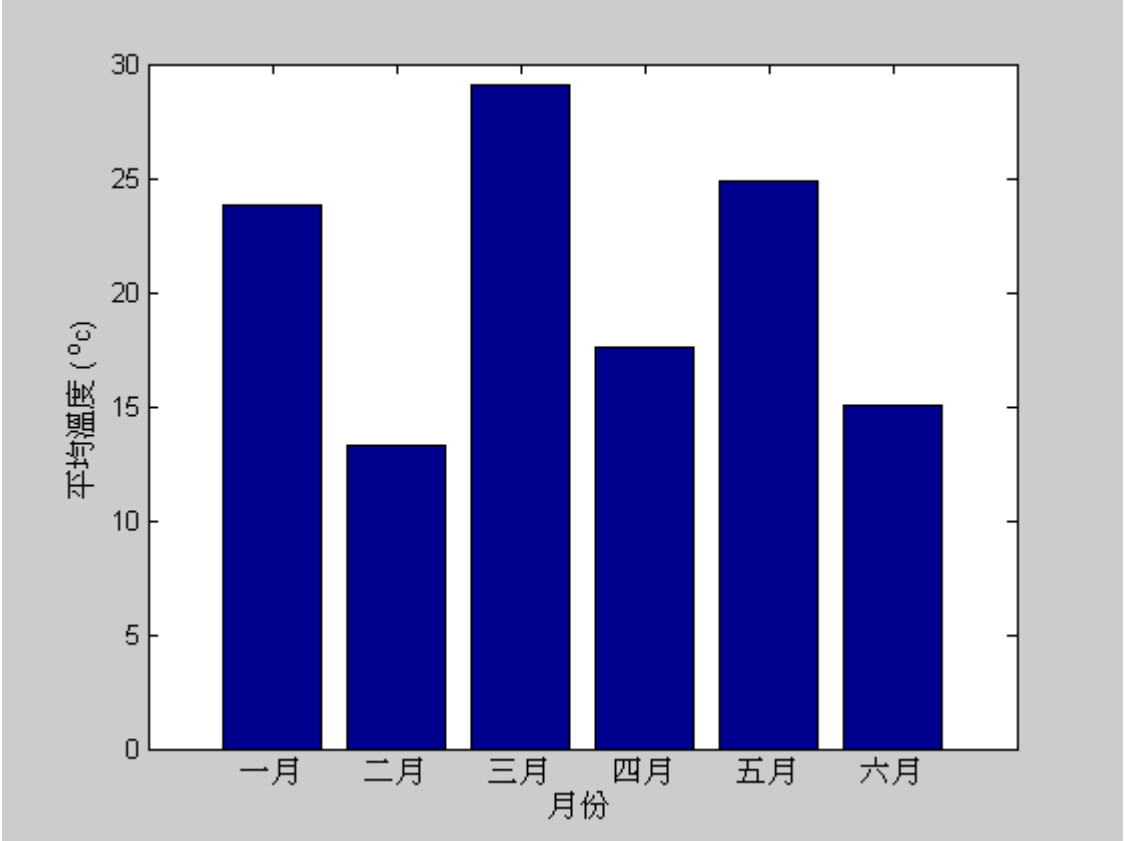


Fig. 5-6

M檔案

15-0 簡介

- M檔案（附檔名為m的檔案）可分兩類：
  - 底稿（Scripts）
  - 函數（Functions）

15-1 底稿

- 底稿(Script)
  - 副檔名為m的檔案，包含 MATLAB各種指令
  - 在MATLAB指令視窗直接輸入檔名，即逐一執行檔案內的指令，如同在MATLAB命令列逐一執行各列指令一般。

M檔案的顯示

- 在目前目錄下有一個M檔案 “script01.m”，可用 type 指令顯示其內容：

```
>> cd 'd:\ matlabBook\MATLAB程式設計：入門篇\15-M檔案'
>> type script01.m

clear all                          % 清除所有變數
x = [1 4 -2 3 -1 -5];
for i = 1:length(x),
   if x(i)>0,
        fprintf('x(%g) = %g is positive\n', i, x(i));
   else
        fprintf('x(%g) = %g is negative or zero\n', i, x(i));
   end
end
```

M檔案的執行

- 欲執行 script01.m，
  - 在指令視窗下輸入 script01 即可

```
>> script01
x(1) = 1 is positive
x(2) = 4 is positive
x(3) = -2 is negative or zero
x(4) = 3 is positive
x(5) = -1 is negative or zero
x(6) = -5 is negative or zero
```

M檔案的執行效應

- 執行程式底稿的效應，相當直接在指令視窗下下達 script01.m 裡的每一列指令
- 所產生的變數也都存放在 MATLAB 的基本工作空間（Base Workspace），可驗證如下：

```
>> whos
  Name      Size            Bytes  Class

  i         1x1                 8  double array
  x         1x6                48  double array

Grand total is 7 elements using 56 bytes
```

提示

- 可在函數中呼叫一程式底稿
  - 產生的變數會放在該函數的工作空間中

底稿的優缺點

- 優點
  - 適用於簡單但重複性高的程式碼
  - 產生的變數保留在基本工作空間中
    - 變數檢視及除錯容易
- 缺點
  - 不支援輸入及輸出引數（Input/Output Arguments）
  - 產生的變數保留在基本工作空間中
    - 變數互相覆蓋而造成程式錯誤

M 檔案編輯器 (I)

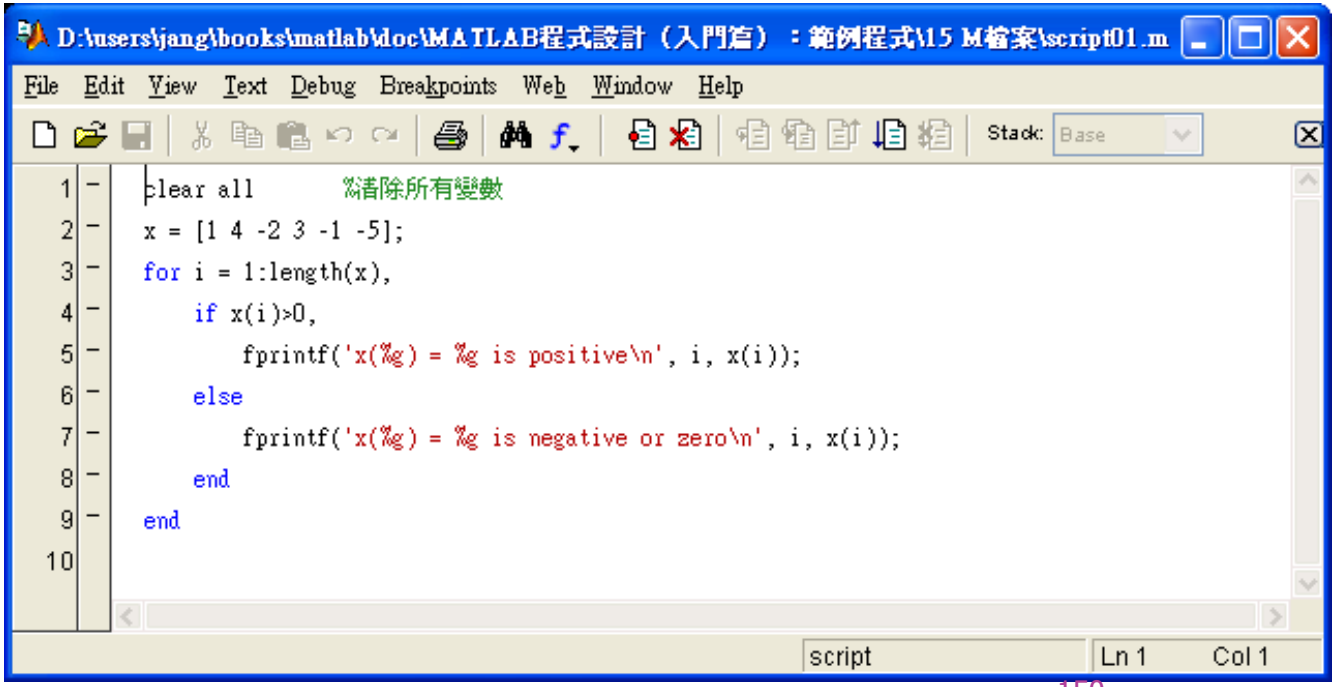
- M 檔案是文字檔
  - 可以用各種文字編輯器修改
  - 儲存時，需以文字模式儲存
- MATLAB在 Windows 及 Mac 平台上，提供了內建的「M 檔案編輯器」（M-File Editor）
  - 點選指令視窗的 file/open 下拉式選單，開啟 M 檔案編輯器
  - 或在指令視窗直接鍵入「edit filename.m」或「open filename.m」

M 檔案編輯器 (II)

- 開啟 Script01.m，可輸入

  >> **edit script01.m**

- 即可開啟 M 檔案編輯器：



提示

- M 檔案編輯器以不同的顏色顯示註解、關鍵字、字串、及一般程式碼
- M 檔案編輯器也是一個除錯器（Debugger）
  - 欲使用其除錯功能，詳見第十七章「程式除錯」

15-2 函數

- 函數
  - 也是 M 檔案的一種
  - 可接受輸入變數，並將結果送至輸出變數
  - 運算過程產生的變數都存放在函數本身的工作空間
    - 不會和 MATLAB 基本工作空間的變數相互覆蓋
  - 函數適用於大型程式碼
    - 使程式碼模組化（Modularized）並易於維護與改進

函數顯示及內容

- func1.m 可算出一向量的平均值
- 用 type 指令顯示其內容：

```
>> type func1.m

function average = func1(vector)
average = sum(vector)/length(vector); % 計算平均值
```

- 第一列為函數定義列（Function Definition Line）
  - 定義函數名稱（func1，最好和檔案的檔名相同）
  - 輸入引數（vector）
  - 輸出引數（average）
  - function為關鍵字
- 第二列為函數主體（Function Body）
  - 規範函數運算過程，並指定輸出引數的值

呼叫函數

- 呼叫此函數，可輸入：

```
>> vec = [1 5 3];
>> ave = func1(vec)
ave =
     3
```

函數線上輔助說明 (I)

- 加上函數「線上輔助說明」（On-line Help）
  - 在函數定義列下直接加入註解

>> **type func2.m**

```
function average = func(vector)
% FUNC2 A simple function with a single help line.
%
% Usage of this function:
% output = func2(input)
% "output" is the average of the input vector "input".

% Roger Jang, 19991123.

average = sum(vector)/length(vector);   % 計算平均值
```

函數線上輔助說明 (II)

- 函數定義列之後的連續註解（以「%」開頭），即為函數的線上輔助說明
- 輸入「help 函數主檔名」，即可看到這些輔助說明

```
>> help func2
  FUNC2 A simple function with a single help line.

    Usage of this function:
    output = func2(input)
    "output" is the average of the input vector "input".
```

H1輔助說明

- 函數線上輔助說明，最重要的就是第一列
  - 又稱為「H1 輔助說明」（H1 Help）
- 使用lookfor keyword查詢 MATLAB 指令
  - 對所給的關鍵字和搜尋路徑上所有函數的「H1 輔助說明」一一比對

```
>> lookfor 'help line'
FUNC2 A simple function with a single help line.
```

提示

- 安裝很多工具箱，或搜尋路徑很長時，lookfor 指令的執行時間可能會較長

函數的目錄 (I)

- 使MATLAB 在任何目錄內，均可執行其他目錄內的函數，但須先將此目錄放在MATLAB的搜尋路徑上：
  - 將和某應用相關的函數，存放於一子目錄內
  - 將此目錄加入搜尋路徑
  - 加入路徑 => 使用 addpath 指令
  - 移除路徑 => 使用 rmpath指令

函數的目錄 (II)

- 建立目錄的線上輔助說明
  - 在此目錄下加入特定檔案 Contents.m
  - 此檔案只能包含輔助說明文字
  - 每列均需以「%」開頭
  - 輸入「help 目錄名稱」時，顯示在「目錄名稱」下 Contents.m 的輔助說明

函數命名的限制

- 函數名稱和變數名稱有相同的限制
  - 只接受前 31 個字母（MATLAB 5.x）或前 63 個字母（MATLAB 6.x 和 7.x）
  - 以英文字母作為開頭
- 函數名稱和檔案名稱不同
  - 仍可依檔案名稱呼叫檔案
  - 函數名稱將被忽略

函數的輸入和輸出

- 一個函數可以有多輸入及輸出
- func3.m 可接受兩個輸入並產生兩個輸出

```
>> type func3.m
function [ave1, ave2] = func3(vector1, vector2);
ave1 = sum(vector1)/length(vector1);
ave2 = sum(vector2)/length(vector2);
```

- func3.m 的呼叫方式

```
>> [a, b] = func3([1 2 3], [4 5 6 7 8])
a =
     2
b =
     6
```

輸出入變數的個數 (I)

- 決定函數實際輸入輸出變數的個數
  - 使用內建變數 nargin 及 nargout
  - 主要功能
    - 設定未被指定之輸入引數的預設值
    - 避免計算未被用到的輸出引數，以節省計算時間

輸出入變數的個數 (II)

- 上述函數 func3.m 可改寫成 func4.m

```
>> type func4.m
function [ave1, ave2] = func4(vector1, vector2)

if nargin == 1,                % 只有一個輸入變數
   ave1 = sum(vector1)/length(vector1);
end

if nargout == 2,                          % 有兩個輸出變數
   ave1 = sum(vector1)/length(vector1);
   ave2 = sum(vector2)/length(vector2);
end
```

輸出入變數的個數 (III)

- func4.m 可以接受一個或兩個輸入變數：

```
>> [a, b] = func4([1 2 3], [4 5 6 7 8])
a =     2
b =     6
>> c = func4([1 3 5 7 9])
c =     5
```

- MATLAB 函數亦可傳送不定數目的輸入引數和輸出引數

提示

- 從外表來看，MATLAB 函數的變數傳遞方法是 “Call by Value”
  - 函數的工作空間中，所有的輸入變數均是父工作空間（Parent Workspace）的一份拷貝
  - 在函數中更改這些輸入變數，並不會影響原先父工作空間的變數
- 實際運作上
  - 若輸入變數未被修改，MATLAB 採用 “Call by Reference”
  - 否則，則採用 “Call by Value”

15-3 次函數與私有化目錄

- 一個 M 檔案可以包含一個以上的函數
  - 一個主函數（Primary Function）
  - 其他則為次函數（Subfunctions）
  - 次函數只能被同檔案中的函數（主函數或次函數）呼叫，但不可被不同檔案的其他函數呼叫
- 主函數與次函數的位置
  - 主函數必需出現在最上方
  - 其後接上任意數目的次函數
  - 次函數的次序並無任何限制

主函數與次函數範例

- func5.m 包含一個主函數及一個次函數
- 次函數的功能是計算倒數向量

```
>> type func5.m
function out = func5(x)
recip = reciproc(x);
out = sum(recip);

% Definition for subfunctions
function output = reciproc(input)
output = 1./input;
```

- 呼叫此函數

```
>> func5([1 2 3])
ans =
    1.8333
```

私有化目錄

- 私有化目錄（Private Directory）
  - 在目錄中建立名稱為 private 的私有化目錄
  - 存放與這目錄相關的函數
  - 目錄 private 之下的函數，只能被其父目錄函數所呼叫，不能被其他目錄的函數來呼叫

函數搜尋次序

- 從 M 檔案呼叫一個函數時，MATLAB 搜尋函數的次序：
  - 檢查此函數是否為次函數
  - 檢查此函數是否為私有化目錄的函數
  - 從系統所設定的搜尋路徑找尋此函數
- MATLAB 找到第一個檔名相符的函數，即會立即取用

15-4 區域變數與全域變數

- 區域變數（Local Variables）
  - 每一個函數在運算時，均佔用個別的記憶體
  - 此工作空間和 MATLAB 的基本工作空間或是其他函數的工作空間是互相獨立的
  - 不同空間的變數是完全獨立，不會相互影響
  - 不同工作空間的變數，稱為「區域變數」

全域變數的使用 (I)

- 若要減少變數的傳遞，可用「全域變數」（Global Variables）
- 使用全域變數前，需先進行變數宣告

```
>> type func6.m
function func6
global X                              % 全域變數宣告
X = X + 2;
fprintf('The value of X in "func6" is %g.\n', X);
```

全域變數的使用 (II)

- Func6.m沒有輸出和輸入，只宣告全域變數 X，將 X 的值加 2，並印出其值
- 測試

```
>> global X                    % 在基本工作空間進行全域變數 x 的宣告
>> X = 2;
>> fprintf('The value of X in the base workspace is %g.\n', X);
The value of X in the base workspace is 2.

>> func6;
The value of X in "func6" is 4.

>> fprintf('The value of X in the base workspace is %g.\n', X);
The value of X in the base workspace is 4.
```

全域變數的使用原則

- 盡量少用全域變數
  - 全域變數使程式的流程不透明，造成程式除錯或維護的困難
- 使用全域變數，請遵循下列兩原則
  - 在各個工作空間使用前，一定要個別宣告
  - 使用全部大寫或較長的變數名稱，以資區別
- 檢視工作空間的變數，輸入whos global
- 清除所有工作空間的全域變數 X，需使用 clear global X

15-5 程式碼保護：P-CODE

- p-code
  - 一般的 M 檔案都是文字檔
  - 所有的 MATLAB 原始程式碼都看得到
  - 讓別人使用您的程式碼，又不想被看到程式碼的內容，使用 pcode 指令將底稿或函數轉成 p-code（即Pseudo-Code）

    pcode filename.m

P-CODE的使用

- 將函數 func5.m 轉成 p-code

```
>> pcode func5.m
>> dir *.p
func5.p
```

- 檢視func5，以p-code的程式碼為優先

```
>> which func5
D:\matlabBook\MATLAB程式設計：入門篇\15-M檔案\func5.p
```

- 呼叫 p-code 的函數和一般函數並無不同

```
>> func5([2 4 8])
ans =
    0.8750
```

P-CODE提高效率

- 一函數被呼叫時，MATLAB 會載入並剖析（Parse）此函數
  - 剖析結果存放置在記憶體內
  - 下次再呼叫此函數，可以省下剖析所花的時間
- pcode 的作用是將程式碼剖析後的結果儲存
- 程式碼牽涉到很多 M 檔案時
  - 將程式碼轉成 p-code，可節省剖析的時間
  - 但由於目前電腦速度太快，節省的時間並不顯著

程式流程控制

16-1 迴圈指令

- MATLAB 提供兩種迴圈指令
  - for 迴圈（For Loop）
  - while 迴圈（While Loop）

FORMATS OF FOR LOOPS

- Format 1:

```
for   變數 = 向量
          運算式
end
```

- 在上述語法中，變數的值會被依次設定為向量的每一個元素值，來執行介於 for和 end 之間的運算式。

- Format 2:

```
for   變數 = 矩陣
          運算式
end
```

- 在上述語法中，變數的值會被依次設定為矩陣的每一個直行，來執行介於 for 和 end 之間的運算式。

程式流程控制之範例一

- 下列 for 迴圈會產生一個長度為 6 的調和數列（Harmonic Sequence）：
- 範例16-1：forLoop01.m

```
x = zeros(1,6);                    % 變數 x 是一個 1×6 大小的零矩陣
for i = 1:6
        x(i) = 1/i;
end
x                                  % 顯示 x
x =
    1.0000    0.5000    0.3333    0.2500    0.2000    0.1667
```

- 在上例中，矩陣 x 最初是一個 1×6 大小的零矩陣，在 for 迴圈中，變數 i 的值依次是 1 到 6，因此矩陣 x 的第 i 個元素的值依次被設為 1/i。
- 我們接著可用分數形式來顯示此數列：

```
>> format rat% 使用分數形式來顯式數值
>> disp(x)
     1       1/2       1/3       1/4       1/5       1/6
```

程式流程控制之範例二

- for 迴圈可以是多層或巢狀式（Nested）的，在下例中即產生一個 6×6 的Hilbert 矩陣 h，其中為於第 i 列、第 j 行的元素為 ：

$$h_{i,j} = \frac{1}{i + j - 1}$$

- 範例16-2：forLoop02.m

```
h = zeros(6);                    % 變數 x 是一個 6×6 大小的零矩陣
for i = 1:6
        for j = 1:6
                h(i,j) = 1/(i+j-1);
        end
end
format rat                       % 使用分數形式來顯式所有數值
h                                % 顯示 h
h =
     1     1/2     1/3     1/4     1/5     1/6
    1/2    1/3     1/4     1/5     1/6     1/7
    1/3    1/4     1/5     1/6     1/7     1/8
    1/4    1/5     1/6     1/7     1/8     1/9
    1/5    1/6     1/7     1/8     1/9     1/10
    1/6    1/7     1/8     1/9     1/10    1/11
```

程式流程控制之範例三

- 在下例中，for 迴圈列出先前產生的 Hilbert 矩陣的每一直行的平方和：
- 範例16-3：forLoop01.m

```
format short                    % 回到預設形式來顯式所有數值
for i = h
        disp(norm(i)^2);        % 印出每一行的平方和
End
```

```
1.4914
0.5118
0.2774
0.1787
0.1262
0.0944
```

- 在上例中，由於 h 是一個矩陣，因此每一次 i 的值就是矩陣 h 的一直行的內容。

程式流程控制之範例四

- 若要跳出 for 迴圈，可用 break 指令。例如，若要找出最小的 n 值，滿足 n! > 1e100，可輸入如下：
- 範例16-4：break01.m

```
for i = 1:1000
    if prod(1:i) > 1e100
        fprintf('%g! = %e > 1e100\n', i, prod(1:i));
        break;                    % 跳出 for 迴圈
    end
end
```

70! = 1.197857e+100 > 1e100

程式流程控制之範例五

- 在一個迴圈內若要直接跳至到此迴圈下一回合的執行，可使用 continue 指令。
- 範例16-5：continue01.m

```
x = [1 -2 3 -4 5];
posTotal = 0;
for i = 1:length(x)
        if x(i)<0, continue; end   % 若 x(i) 小於零，跳到此迴圈的下一回合
        posTotal=posTotal+x(i);
end
posTotal                                                    % 顯示 posTotal 的值
posTotal =
     9
```

- 上述範例中，我們計算向量 x 的正元素的總和，因此只要遇到 x(i) 是負數，即可使用 continue 指令來直接跳到此迴圈的下一個回合來繼續執行。Continue 指令從MATLAB 6.x 才開始支援，若是使用 MATLAB 5.x，可用 if–else 來達到相同的功能。

FORMAT OF WHILE LOOPS

- While loop 的格式如下：

```
while  條件式
        運算式；
end
```

- 在上述語法中，只要條件式為真，運算式就會一再被執行

程式流程控制之範例六

- 先前產生調和數列的例子，亦可用 while 迴圈改寫如下：
- 範例16-6：while01.m

```
x = zeros(1,6);
i = 1;
while i<=6
      x(i) = 1/i;
      i = i+1;
end
x                          % 顯示 x
```

```
x =
    1.0000    0.5000    0.3333    0.2500    0.2000
0.1667
```

程式流程控制之範例七

- 若要用 while 指令找出最小的 n 值，使得 n! > 1e100 ，可輸入如下：
- 範例16-7：while02.m

```
n = 1;
while prod(1:n) < 1e100
        n = n+1
end
fprintf('%g! = %e > 1e100\n', n, prod(1:n));
```

```
70! = 1.197857e+100 > 1e100
```

- 與前述的 for 迴圈相同，在任何時刻若要跳出 while 迴圈，亦可使用 break 指令；若要跳到下一回合的 while 迴圈，也可以使用 continue 指令。
- 無論是 for 或 while 迴圈，均會降低 MATLAB 的執行速度，因此盡量使用向量化的運算（Vectorized Operations）而盡量少用迴圈。
- break 指令若用在多重迴圈中，每次只跳出包含break指令的最內部迴圈。

16-2 條件指令

- MATLAB 支援二種條件指令（Branching Command，或中譯成「分支指令」）
  - if-else 條件指令
  - switch-case-otherwise 條件指令（ MATLAB 在第五版之後開始支援）

IF-ELSE 條件指令

- 最常用的條件指令是 if-else，其使用語法為

```
if 條件式
        運算式一;
else
        運算式二;
end
```

- 在上述語法中，當條件式成立時，MATLAB 將執行運算式一，否則，就執行運算式二。若不需使用運算式二，則可直接省略 else 和運算式二。

程式流程控制之範例八

- 在數值運算的過程中，若變數值為 NaN（即 Not A Number ）時，我們要立刻印出警告訊息，可輸入如下例：
- 範例16-8：if01.m

```
x = 0/0;
if isnan(x)
        disp('Warning: NaN detected!');
end
```

```
Warning: Divide by zero.
...
Warning: NaN detected!
```

- 在上例中，第一個警告訊息是 MATLAB 自動產生的，第二個警告訊息則是我們的程式碼產生的，其中 isnan(x) 可用於判斷 x 是否為 NaN，若是，則傳回 1（真），否則即傳回 0（偽）。

程式流程控制之範例九

- 在下例中，我們可根據向量 y 的元素值為奇數或偶數，來顯示不同的訊息：
- 範例16-9：if02.m

```
y = [0 3 4 1 6];
for i = 1:length(y)
        if rem(y(i), 2)==0
                fprintf('y(%g) = %g is even.\n', i, y(i));
        else
                fprintf('y(%g) = %g is odd.\n', i, y(i));
        end
end
```

```
y(1) = 0 is even.
y(2) = 3 is odd.
y(3) = 4 is even.
y(4) = 1 is odd.
y(5) = 6 is even.
```

- 上述的 if-else 為雙向條件，亦即程式只會執行「運算式一」或「運算式 二」，不會有第三種可能。

程式流程控制之範例十

- MATLAB 亦可執行多向條件，若要進行更多向的條件，只需一再重覆 elseif 即可。例如，欲判斷 y 向量之元素是屬於 3n、3n + 1、 或 3n+2，可輸入如下：
- 範例16-10：if03.m

```
y = [3 4 5 9 2];
for i = 1:length(y)
        if rem(y(i),3)==0
                fprintf('y(%g)=%g is 3n.\n', i, y(i));
        elseif rem(y(i), 3)==1
                fprintf('y(%g)=%g is 3n+1.\n', i , y(i));
        else
                fprintf('y(%g)=%g is 3n+2.\n', i , y(i));
        end
end
```

```
y(1)=3 is 3n.
y(2)=4 is 3n+1.
y(3)=5 is 3n+2.
y(4)=9 is 3n.
y(5)=2 is 3n+2.
```

SWITCH-CASE-OTHERWISE 條件指令

- MATLAB 在第五版開始支援 switch-case-otherwise 的多向條件指令，其使用語法如下：

```
switch    expression
        case   value(1)
                  statement(1)
        case   value(2)
                  statement(2)
        …
        case   value(n-1)
                  statement(n-1)
        otherwise
                  statement(n)
end
```

- 在上述語法中，expression 為一數值或字串，當其值和 value(k) 相等時，MATLAB 即執行 statement(k) 並跳出 switch 指令。若 expression 不等於 value(k)，k=1, 2, …, n-1，則 MATLAB 會執行 statement(n) 並跳出 switch 指令。

程式流程控制之範例十一

- 欲根據月份來判斷其季別，可輸入如下：
- 範例16-11：switch01.m

```
for month = 1:12
        switch month
                case {3,4,5}
                        season = 'Spring';
                case {6,7,8}
                        season = 'Summer';
                case {9,10,11}
                        season = 'Autumn';
                case {12,1,2}
                        season = 'Winter';
        end
        fprintf('Month %d ===> %s.\n', month, season);
end
Month 1 ===> Winter.
  .                       .
  .                       .
  .                       .
Month 12 ===> Winter.
```

程式流程控制之範例十二

- 如果 expression 是字串，那麼若要在 case 之後比對多個字串，就必需使用字串的異值陣列（Cell Arrey of Strings）：
- 範例16-12：switch02.m

```
month = {'Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep'};
for i = 1:length(month)
        switch month{i}
                case {'Mar','Apr','May'}
                        season = 'Spring';
                case {'Jun','Jul','Aug'}
                        season = 'Summer';
                case {'Sep','Oct','Nov'}
                        season = 'Autumn';
                case {'Dec','Jan','Feb'}
                        season = 'Winter';
        end
        fprintf('%s is %s.\n', month{i}, season);
end
```

程式流程控制之範例十二

- 上述範例output如下:

  Jan is Winter.
  Feb is Winter.
  Mar is Spring.
  Apr is Spring.
  May is Spring.
  Jun is Summer.
  Jul is Summer.
  Aug is Summer.
  Sep is Autumn.
- MATLAB 的 switch 指令和 C 語言的 switch 指令略有差別：在 C 語言的 switch 敘述內，每個 case 敘述需加上 break 以跳出該 switch 敘述，而在 MATLAB 則不必多此一舉。
- 一般而言，switch–case–otherwise 的執行效率優於 if–else 。

影像顯示與讀寫

19-1 MATLAB的影像格式

- MATLAB 最常處理的影像格式為索引影像（Indexed Images）
- 顯示此類型影像的語法如下：

```
image(X)
colormap(map)
```

其中X為影像的資料矩陣，map為色盤矩陣。

- 色盤矩陣的大小為K×3，每個橫列由三個元素所組成，分別是R（紅）、G（綠）、B（藍），每個元素的範圍為0~1
- X的值為1~K，也就是當X(i, j)的值為p，則像素點(i, j)的顏色為map(p, :)這一列的向量所決定。

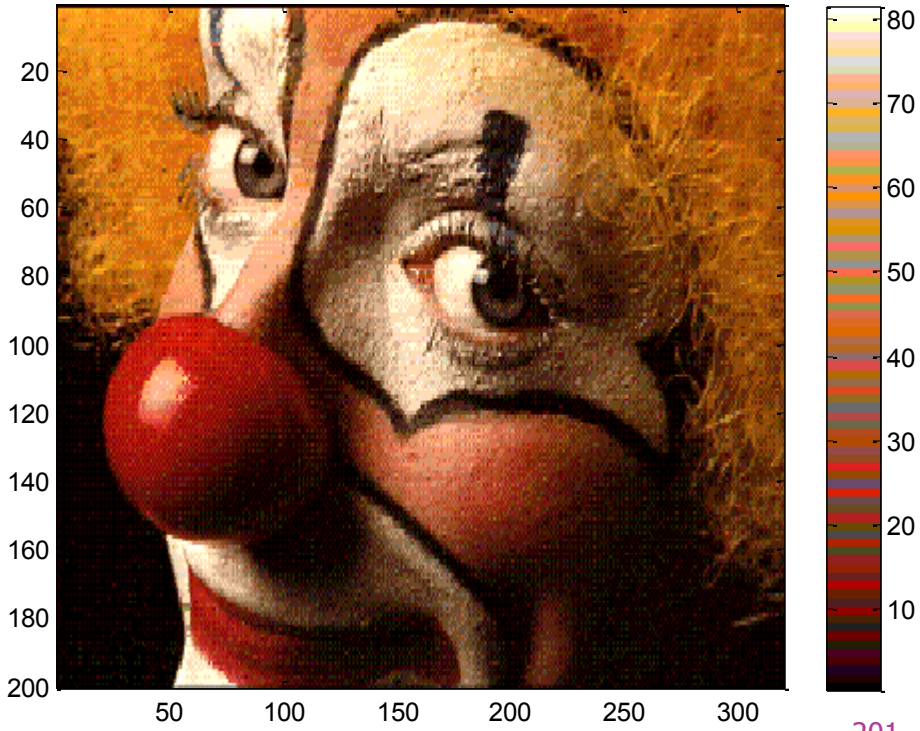
索引影像：顯示

- 使用MATLAB顯示內建的小丑圖。
- 範例19-1：image01.m

```
load clown.mat                    % 載入小丑影像資料，含變數 X 和 map
image(X);                         % 顯示影像
colormap(map)                     % 取用色盤矩陣
```

索引影像：顯示

- 欲顯示對應的色盤，可再執行 colorbar，結果如下：



索引影像：驗證索引範圍

- 由於由X 是索引影像，因此其最小值是 1 ，最大值會等於 map 的列數 (即「可顯示之顏色數目」) ，可驗証如下：
- 範例19-2 ：image02.m

```
load clown.mat                    % 載入小丑影像資料，
含變數 X 和 map
fprintf('min(min(X)) = %d\n', min(min(X)));
fprintf('max(max(X)) = %d\n', max(max(X)));
fprintf('size(map, 1) = %d\n', size(map, 1));
```

索引影像：驗證索引範圍

min(min(X)) = 1

max(max(X)) = 81

size(map, 1) = 81

由範例可知，此小丑影像共含有 81 種不同的顏色。

索引影像：驗證數值

- 索引影像的數值可以驗證如下。
- 範例：indexedImage01.m

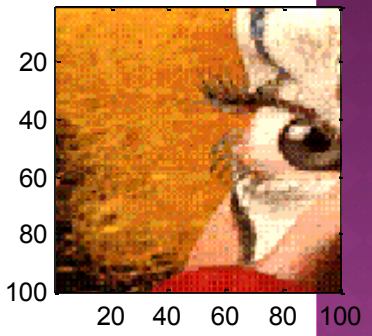
```
load clown.mat                    % 載入小丑影像資料，
含變數 X 和 map
subplot(2,2,1); image(X); axis image
subplot(2,2,2); image(X(1:100, 1:100)); axis
image
subplot(2,2,3); image(X(1:3, 1:3)); axis image
```

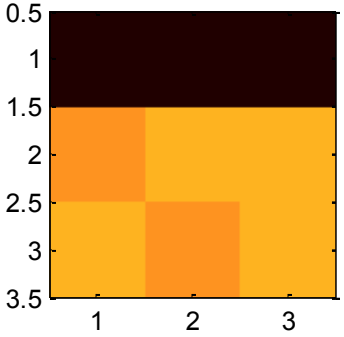
X(1:3, 1:3)

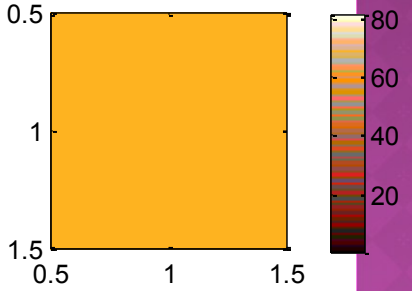
索引影像：驗證數值

- 結果：

```
ans =
     2     2     2
    61    69
69
    69    61
69
```







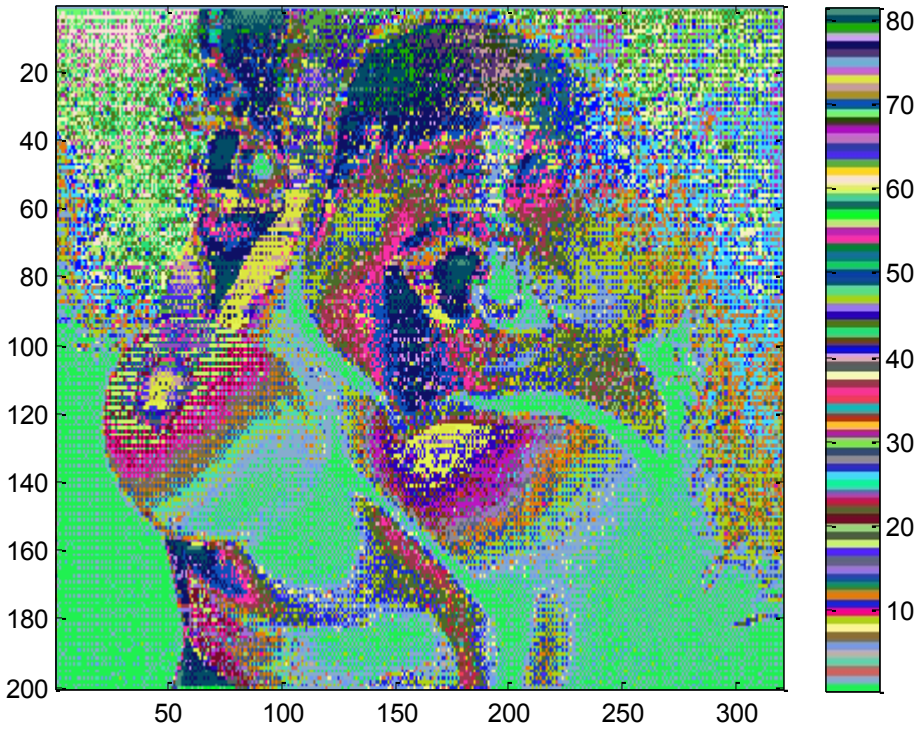
索引影像：亂數色盤

- 要正確地顯示索引影像則需要正確的色盤，以上面的小丑影像為例，如果使用亂數產生的色盤則會產生下面的結果：
- 範例19-3：image03.m

```
load clown.mat                    % 載入小丑影像資料，
含變數 X 和 map
newmap = rand(size(map));
image(X);
colormap(newmap) ;
```

索引影像：亂數色盤

- 由於色盤是亂數產生，所以每次結果都不一樣：



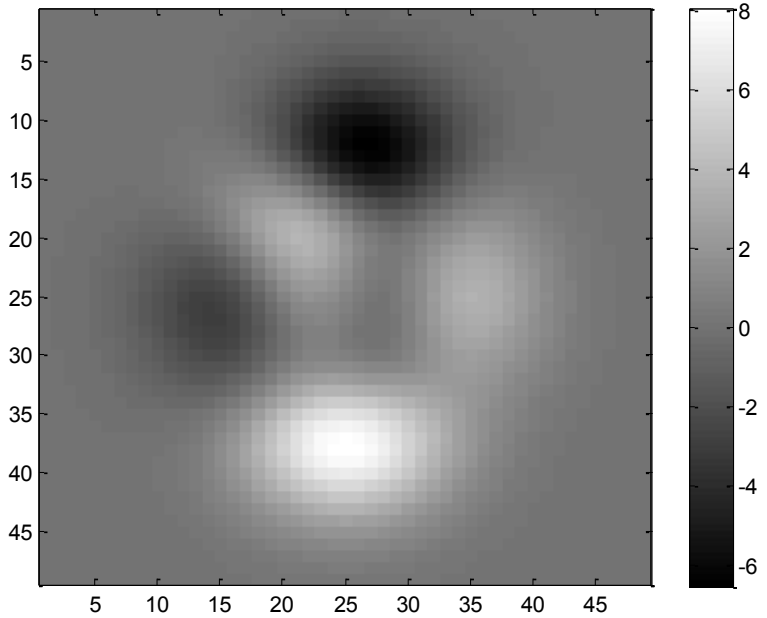
索引影像：強度影像

- 如果我們的色盤矩陣只有 K 個橫列，但是 X 的某些元素值小於 1 或大於 K，則我們可以使用 imagesc指令將 X 的最小值轉換成 1，最大值轉成 K，其他中間值則依線性關係轉換成介於 1 與 K 的值，舉例如下：
- 範例19-4：imagesc01.m

```
X = peaks;
imagesc(X);
colormap(gray);
colorbar;
min(min(X))                % 顯示 X 的最小值
```

索引影像：強度影像

```
ans =
   -6.5466
ans =
    8.0752
```



- 具有上述特性的影像資料稱為強度影像（Intensity Images），一般經由數值運算產生的矩陣均屬此類，因此均可由 imagesc 來顯示。

全彩影像：顯示

- image 指令亦接受全彩影像（Truecolor Images）。全彩影像可以表示成一個 m×n×3 的矩陣 X，其中 X (:, :, 1) 代表R（紅色）的強度。X (:, :, 2) 代表G（綠色）的強度，X (:, :, 3) 則代表B（藍色）的強度。
- X的值的範圍可以是下列兩種：
  - 介於0~1的浮點數
  - 0~255的uint8（詳見本章第三節）。
- 範例19-5：image04.m

```
X = imread('annie19980405.jpg');
image(X);
size(X)
```

全彩影像：顯示

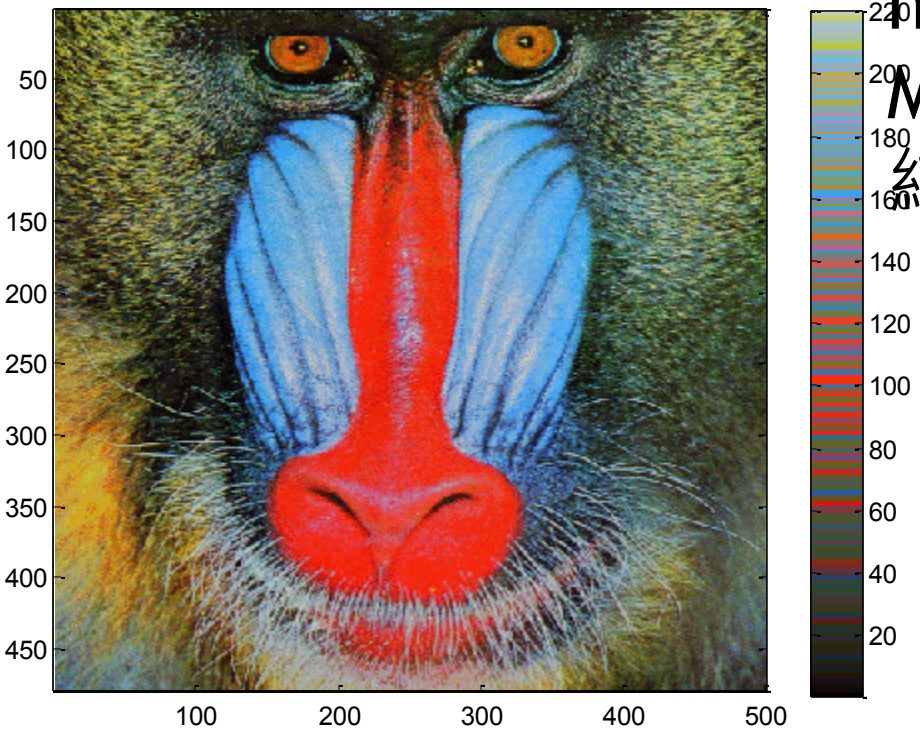
- ans =
  - 480　640　　3
- 此時若再下達 colorbar 指令，只會顯示內定的色盤，和圖形顯示沒有關係。

19-2 影像的顯示與列印

- MATLAB 在顯示影像時，會將之置於預設的圖軸之中，並以此圖軸的長寬比來成像，因而造成影像的失真。若要以影像本身的長寬比來成像，可加入 axis image，如下:
- 範例19-6：image05.m

```
load mandrill.mat
image(X);
colormap(map);
axis image
```

以原影像長寬比例顯示範例



- 亦可下達「axis normal」來觀看MATLAB的預設顯示結果。

將影像對應到螢幕上的點的範例

- 若要使影像資料的每一點對應至螢幕上的一個像素（Pixel），可輸入如下:
- 範例19-7：image06.m

```
load mandrill.mat
[m, n] = size(X);
figure ('unit', 'pixel', 'position', [200, 200, n, m]);
image(X);
colormap(map);
set(gca, 'position', [0, 0, 1, 1]);
```

範例：將影像對應到螢幕上的點

- 此範例產生圖形如同前一個範例，如果你的螢幕解析度較低，圖形會變大。
- 上述範例程式碼中， figure 的 ‘position’ 性質為 [200, 200, n, m]，代表視窗的左下角位置是 [200, 200] (以 pixel 為單位) ，而視窗的寬度為 n，高度為 m，正好可以符合影像的大小。
- gca 傳回使用中的圖軸，最後一個敘述將圖軸的位置設為整個視窗的大小，使用了正規化的單位 。

影像的列印

- 在列印影像時，MATLAB 會根據視窗的 Paper position 性質來調整圖形的長寬比，使得印出的影像再度變形。欲防止情況，可用下列指令：

  >>set(gcf, 'PaperPositionMode', 'auto')

- 若要使 Paper Position Mode 的預設值就是“auto”，可在 startup.m 檔案中加入下一行:

  set(0, 'DefaultFigurePaperPositionMode', 'auto')

19-3 8-BIT影像

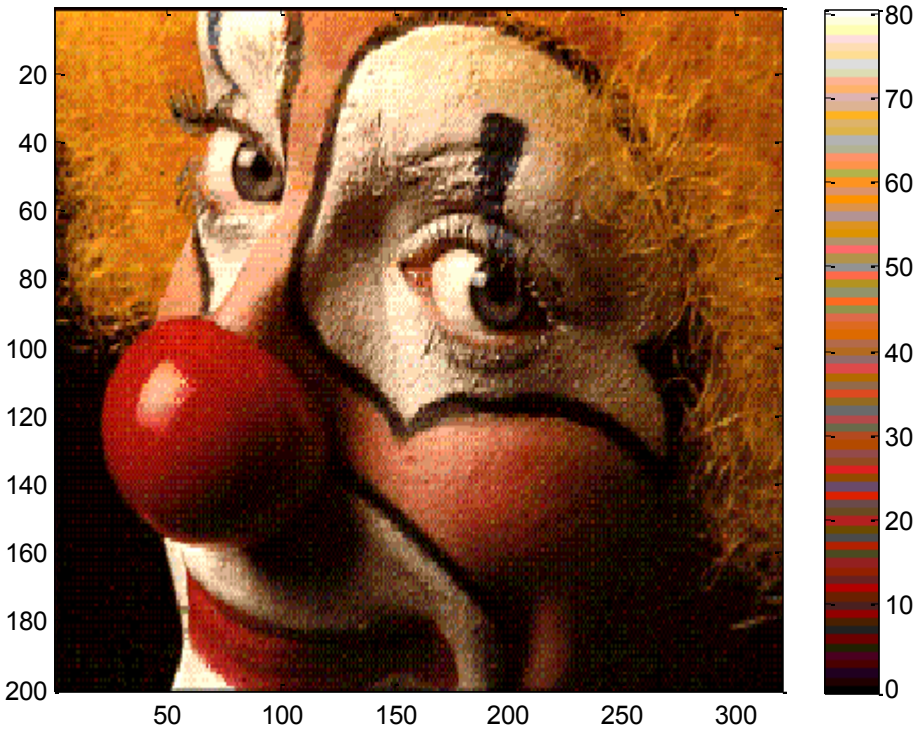
- 在 MATLAB 第 5 版之後，提供了 uint8 的資料型態。
- 由於 uint8 只有 8 個位元，所以能表示的數值範圍為 0 至 255 (=2^8-1) 之間的整數 。

8-BIT影像範例

- 由於 8-bit 影像資料的最小值為 0，和一般的雙精準索引影像資料相差 1，因此在兩種資料相互轉換時，要特別小心。例如：
- 範例19-8：uint801.m

```
load clown.mat
Z8 = uint8(X-1);                % 將 X-1 轉成 uint8 的
資料型態
close all                       % 關掉所有的圖形視窗
image(Z8);
```

8-BIT影像範例


20
40
60
80
100
120
140
160
180
200
50
100
150
200
250
300
80
70
60
50
40
30
20
10
0

8-BIT ⇐⇒ DOUBLE

- 若要將 8-bit 影像轉回雙精準影像，可輸入如下：

  >> Z64 = double(Z8)+1;

- uint8 資料型態亦可用於全彩影像資料，此時每一像素的原色（R，G 或 B）範圍為 0 至 255 間的整數，而不再是 0 至 1 的實數。

8-BIT ⇐⇒ DOUBLE

- 欲將雙精準的全彩影像轉作 uint8 資料型態，可輸入如下：

  >> RGB8 = uint8(round(RGB64*255));

- 其中 RGB64 為雙精準的全彩影像資料，而 RGB8 則是 unit8 的 8-bit 影像資料。反之，若欲進行反轉換，可輸入如下：

  >> RGB64 = double(RGB8)/255;

- 關於影像類別及其資料型態的關係，可見下表：

影像類別及型態關係表

	資料型態	
影像類別	雙精準（Double）	uint8
索引影像 (Indexed Images）	影像矩陣大小：mxn	影像矩陣大小：mxn
	影像資料範圍：介於 [1, k] 的整數	影像資料範圍：介於 [0, k-1] 的整數
	色盤矩陣大小：kx3	色盤矩陣大小：kx3
	色盤資料範圍：介於 [0, 1] 的實數	色盤資料範圍：介於 [0, 1] 的實數
	影像顯示指令：image	影像顯示指令：image （註：k 的值不大於 256）
強度影像 (Intensity Images)	影像矩陣大小: mxn	影像矩陣大小: mxn
	影像資料範圍: 任意實數(但通常是[0,1])	影像資料範圍：介於 [0, 255] 的整數
	色盤矩陣大小：kx3	色盤矩陣大小：kx3
	色盤資料範圍：介於 [0, 1] 的實數	色盤資料範圍：介於 [0, 1] 的實數
	影像顯示指令：imagesc （色盤通常是灰階）	影像顯示指令：imagesc （色盤通常是灰階）
全彩影像 (Truecolor Images)	影像矩陣大小: mxnx3	影像矩陣大小: mxnx3
	影像資料範圍：介於 [0，1] 的實數	影像資料範圍：介於 [0, 255] 的整數
	影像顯示指令：image	影像顯示指令：image

19-4 影像檔案的讀取與寫入

- imread 指令可用於讀取影像檔案。
- imwrite 則可用於寫入影像檔案。
- 這兩個指令可以處理的影像格式有下列幾種：

IMREAD及IMWRITE支援的格式

影像檔案格式	副檔名	相關字串
微軟視窗的 Bitmap	bmp	‘bmp’
階層式資料格式 （Hierarchical Data Format）	hdf	‘hdf’
Joint Photographic Expert Group	jpg 或 jpeg	‘jpg’ 或 ‘jpeg’
微軟視窗的 Paintbrush	pcx	‘pcx’
可攜式網路圖形 （Portable Network Graphics）	png	‘png’
標記式影像檔案格式 （Tagged Image File Format）	tiff	‘tif’ 或 ‘tiff’
X視窗傾印 （X Windows Dump）	xwd	‘xwd’
圖形交換格式 （Graphic Interchange Format） （第六版才支援）	gif	‘gif’

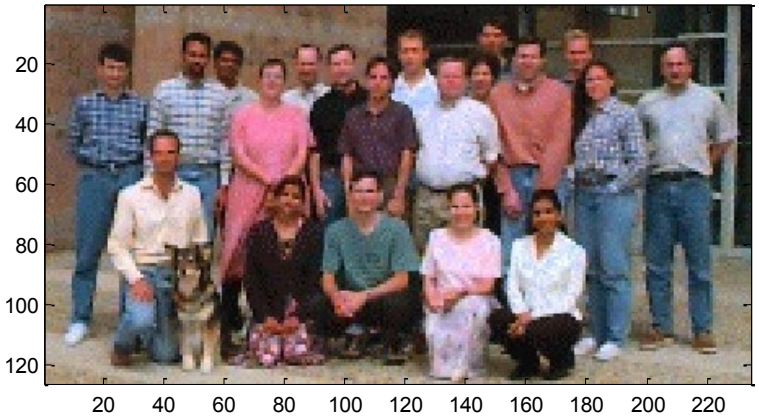
IMREAD 指令

- imread 指令可以讀取上述格式的影像檔案，並進行必要之轉換，如下：
  - 對於強度影像，imread 將資料以 uint8 的矩陣（大小為 m×n）傳回。
  - 對於索引影像，imread 將資料以 uint8 的矩陣（大小為 m×n）傳回，並同時傳回一個雙精準的色盤矩陣，其每個元素值介於[0,1]。
  - 對於全彩矩陣，imread 將資料以 uint8 的矩陣（大小為 m×n×3）傳回。

使用IMREAD讀取全彩JPEG影像

- imread 可讀出下列全彩影像：
- 範例19-9：imread01.m

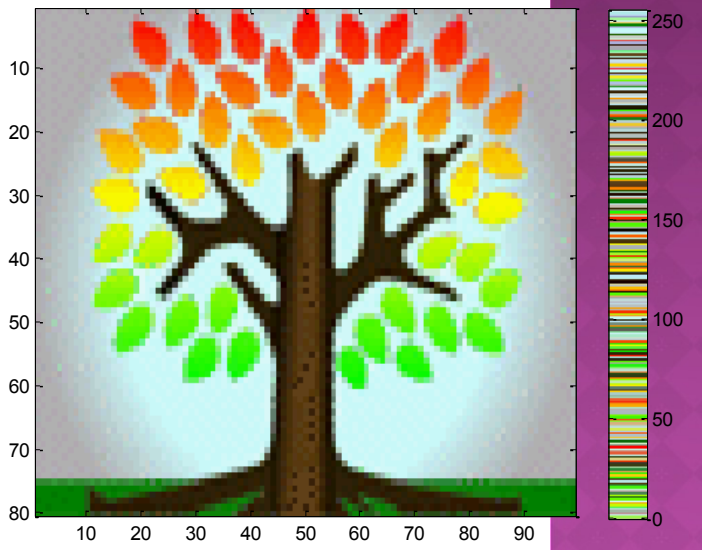
```
RGB = imread('simulinkteam.jpg');
image(RGB) ;
axis image;
class(RGB)
```



使用IMREAD讀取索引影像

- imread 可讀出下列索引影像：
- imread02.m

```
[X, map] =
imread('sbtree.gif');
image(X);
colormap(map);
colorbar;
```



影像檔案寫入範例

- imwrite 指令可將資料寫成影像檔如下：
- 範例19-10：imwrite01.m

```
load clown.mat
imwrite(X, map, 'myClown.jpg');
!start myClown.jpg
```

- 上述最後一列敘述將會呼叫 Windows 作業系統下的應用程式來開啟 myClown.jpg 檔案。

IMFINFO指令

- imfinfo 指令可傳回影像檔案的各項資訊，例如：
  - info1=imfinfo('simulinkteam.jpg')
  - info2=imfinfo('sbtree.gif')
- 對於不同的檔案格式，imfinfo 傳回的資訊項目可能有所不同。

IMFINFO執行結果

- info1=imfinfo(‘simulinkteam.jpg')

```
info1 =
          Filename: 'simulinkteam.jpg'
       FileModDate: '28-三月-2000 17:30:36'
          FileSize: 24071
            Format: 'jpg'
     FormatVersion: ''
             Width: 234
            Height: 126
          BitDepth: 24
         ColorType: 'truecolor'
   FormatSignature: ''
   NumberOfSamples: 3
     CodingMethod: 'Huffman'
    CodingProcess: 'Sequential'
          Comment: {[1x70 char]}
```

- info2=imfinfo('sbtree.gif')

```
info2 =
          Filename: 'sbtree.gif'
       FileModDate: '10-九月-1997 14:53:14'
          FileSize: 7121
            Format: 'GIF'
     FormatVersion: '87a'
             Width: 99
            Height: 80
          BitDepth: 8
         ColorType: 'indexed'
   FormatSignature: 'GIF87a'
   BackgroundColor: 0
       AspectRatio: 0
        ColorTable: [256x3 double]
        Interlaced: 'no'
```

QUIZ

- 那些影像檔案格式支援全彩影像？哪一些支援索引影像？