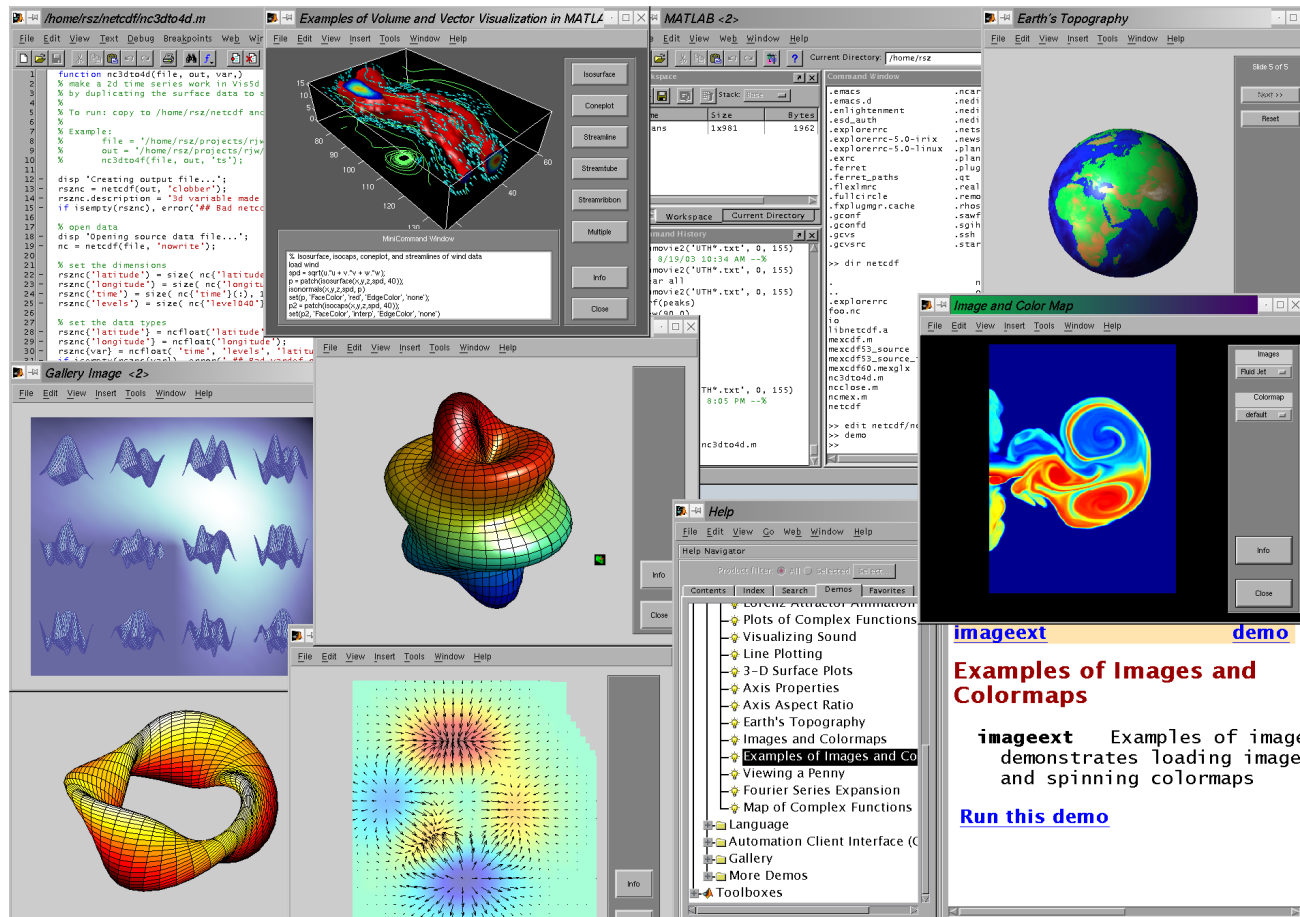


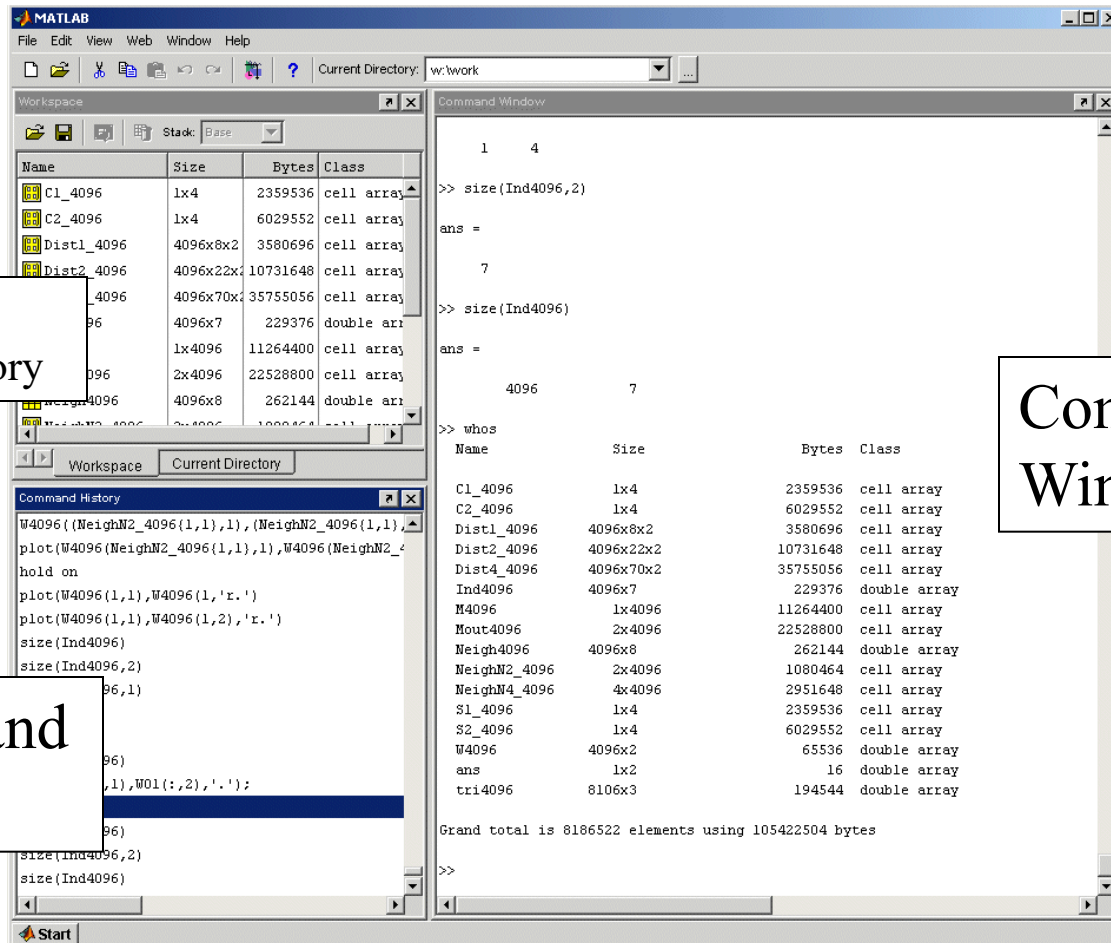
Matlab/Octave Tutorial



Outline

- Basics of Matlab (slides from 張智星教授)
 - <http://mirlab.org/jang/books/matlabProgramming4beginner/>
 - 初探MATLAB
 - 二維平面繪圖
 - 矩陣的處理與運算
 - M檔案
- Symbolic Toolbox in Matlab (slides from Prof. Paul Söderlind)
- My demo code can be downloaded from E3

Matlab Desktop



Workspace /
Current Directory

Command
Window

Command
History



Explore the Matlab Desktop



MATLAB 程式設計入門篇

初探MATLAB

張智星

jang@cs.nthu.edu.tw

<http://www.cs.nthu.edu.tw/~jang>

清大資工系 多媒體檢索實驗室

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（**Command Window**）內的提示符號（**>>**）之後輸入運算式，並按入 **Enter** 鍵即可。例如：

```
>> (5*2+3.5)/5
```

```
ans =
```

```
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，例如：

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

變數命名規則與使用

- 第一個字母必需是英文字母。
- 字母間不可留空格。
- 最多只能有 **31** 個字母，**MATLAB** 會忽略多餘字母（在 **MATLAB** 第 4 版，則是 **19** 個字母）。
- **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** 資料型式儲存。

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 $m \times n$ 的矩陣

- 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矩陣的各種處理之一

- `>> 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矩陣的各種處理之二

- `>> 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矩陣的各種處理之三

- `>> 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**：用來查詢已知指令的用法。
- **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**：產生線上支援視窗，其效果和直接點選 **MATLAB** 命令視窗工作列的圖示是一樣的。
- **doc**：產生特定函數的線上支援。

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
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：
`>> path`
- 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，以避免記憶體的空置與浪費
- 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。



MATLAB 程式設計入門篇

二維平面繪圖

張智星

jang@cs.nthu.edu.tw

<http://www.cs.nthu.edu.tw/~jang>

清大資工系 多媒體檢索實驗室

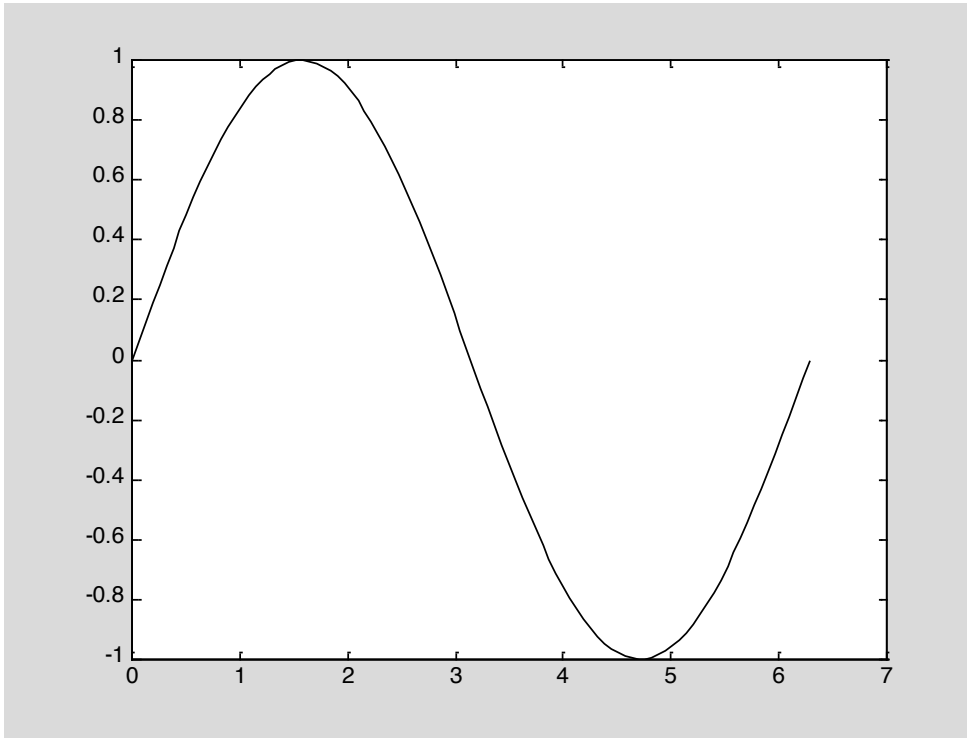
3-1 基本的繪圖指令

- 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 2 $\pi$  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

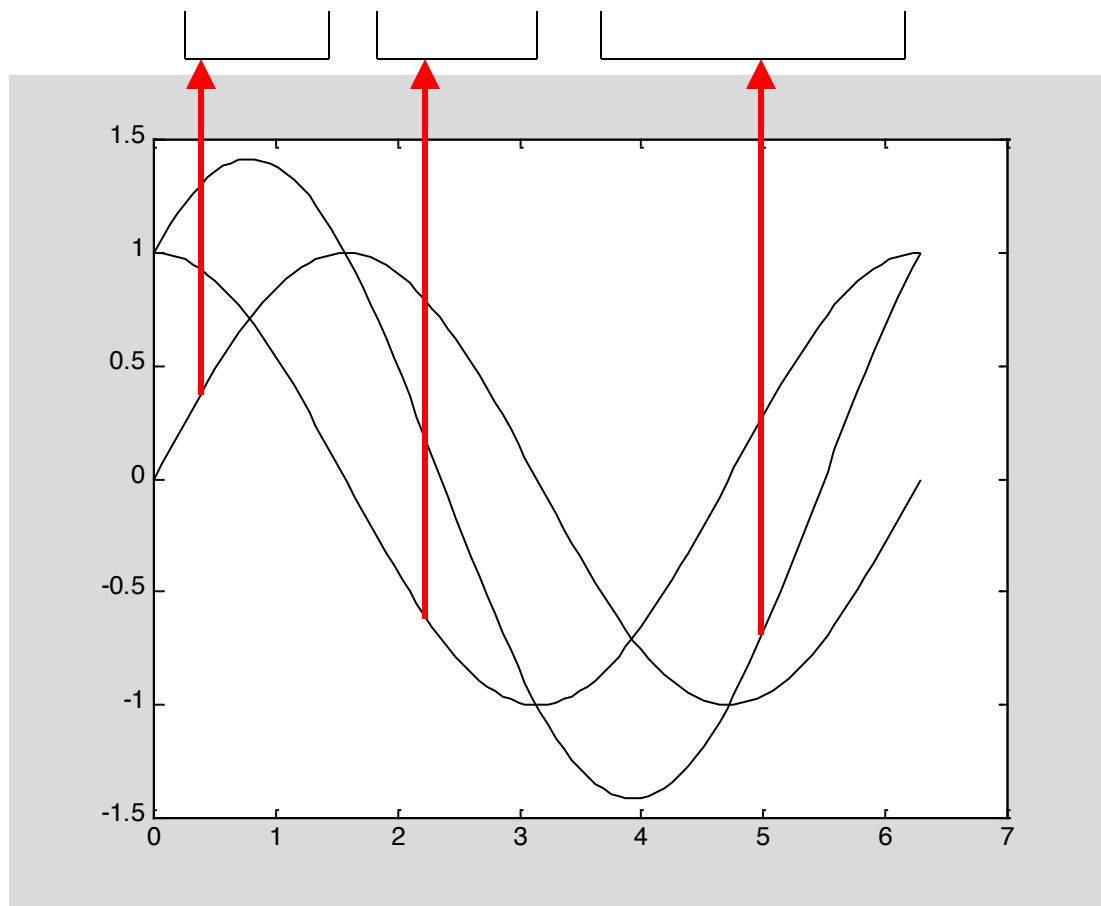
Plot基本繪圖-2 (I)

- 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

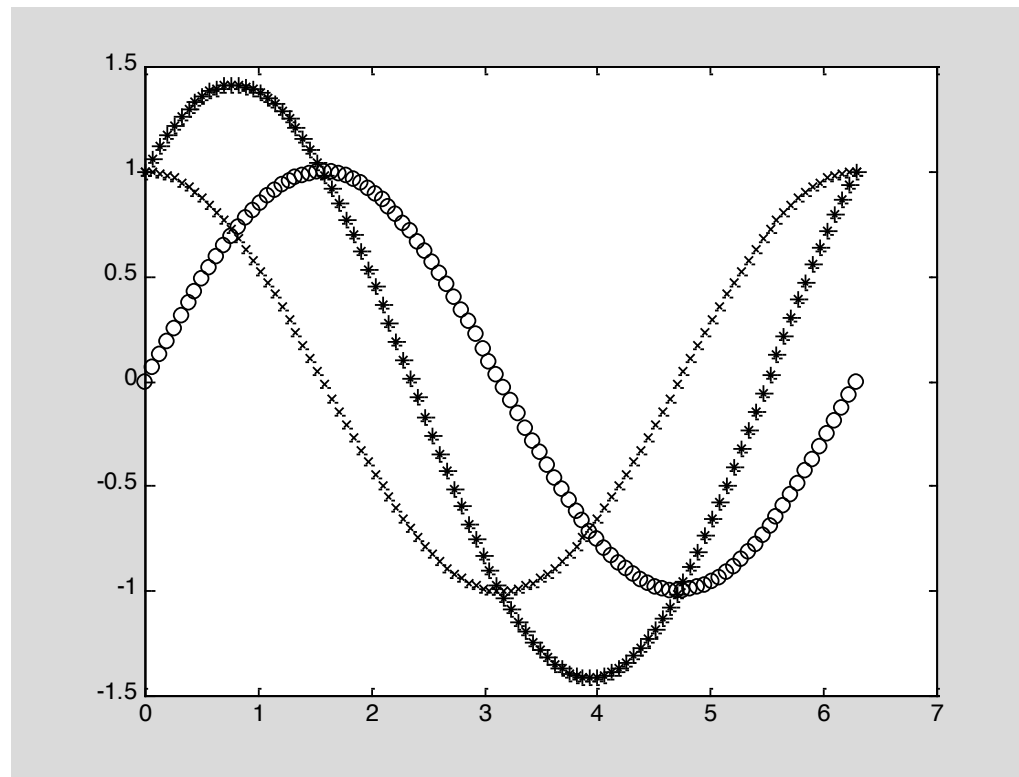
Plot基本繪圖-3 (I)

- 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 2 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](#)

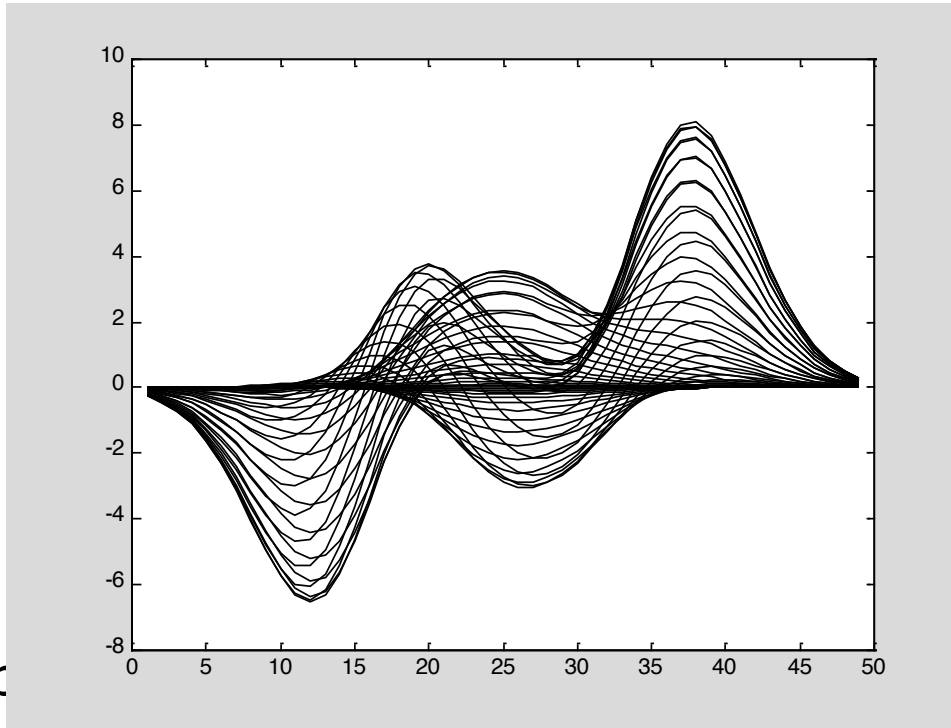
```
y = peaks;
```

```
% 產生一個 49×49 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)



■ `plot`

■ `plot(y)` 直接畫出 49 條直線

■ 類似於從側面觀看 `peaks` 函數

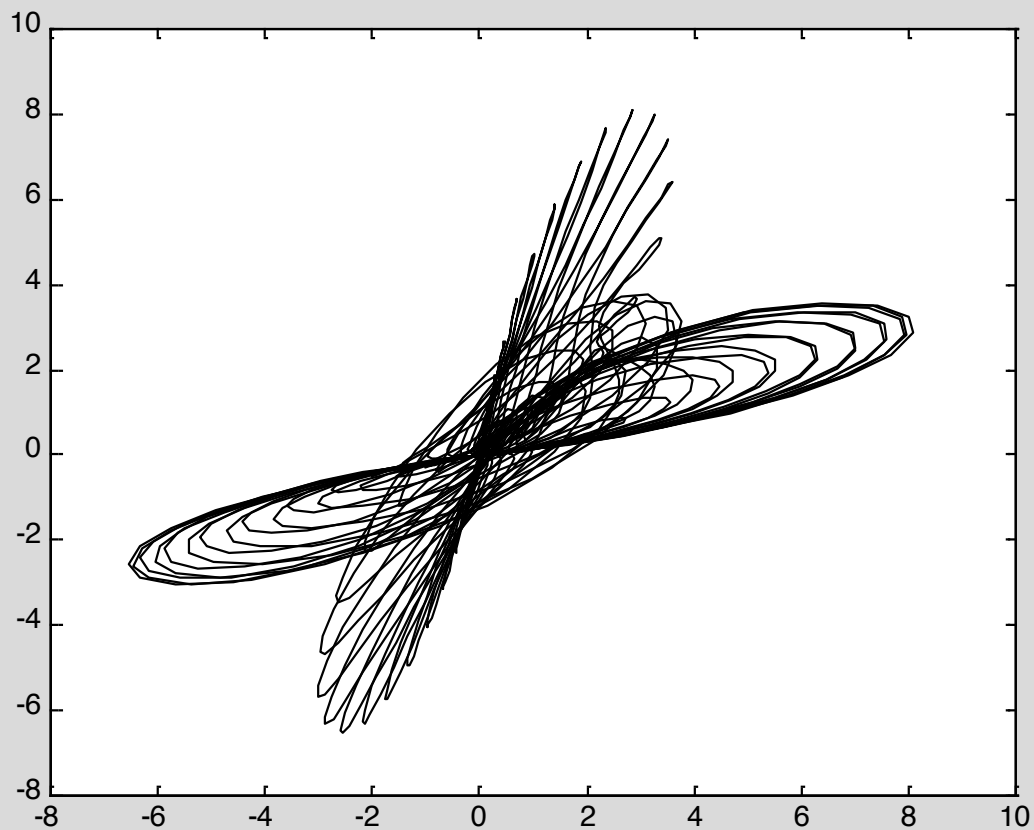
二維函數的值

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

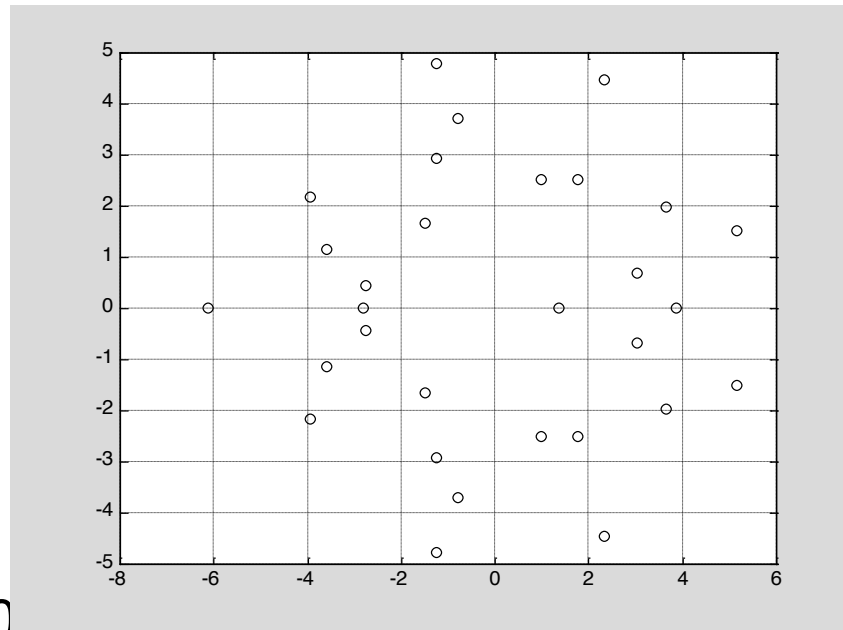
- 一般情況下，MATLAB 將矩陣視為行向量的集合
- 對只能處理向量的函數(Ex：max、min、mean)
 - 給定一個矩陣，函數會對矩陣的行向量一一進行處理或運算

Plot基本繪圖-6 (I)

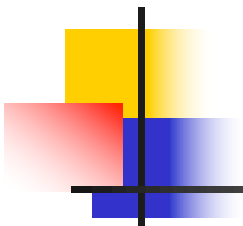
- z 是一個複數向量或矩陣
 - `plot(z)` 將 z 的實部(即 `real(z)`)和虛部(即 `imag(z)`)當成 x 座標和 y 座標來作圖，
 - 其效果等於 `plot(real(z), imag(z))`
 - 範例3-6：[plotxy06.m](#)

```
x = randn(30);    % 產生 30×30 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 的隨機矩陣
- 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

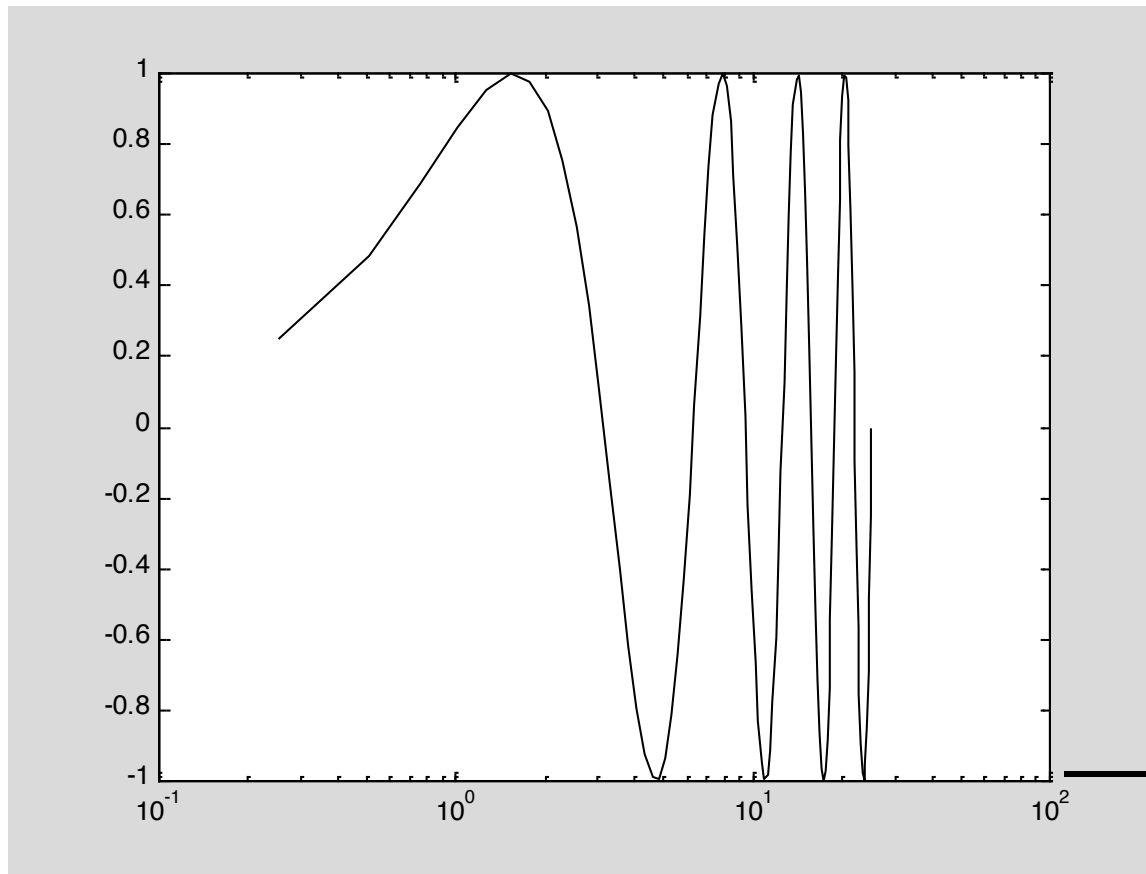
Plot基本繪圖-7 (I)

■ 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

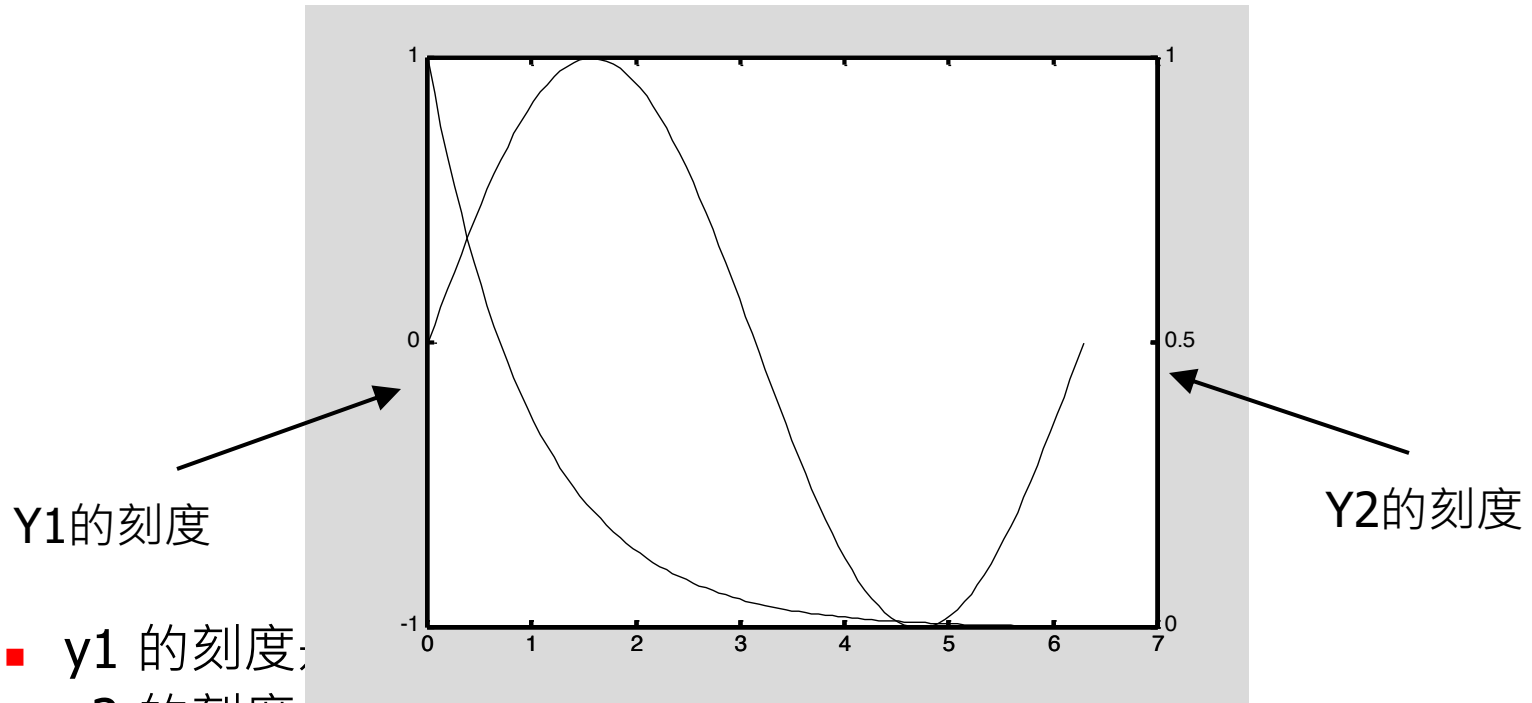
Plot基本繪圖-8 (I)

■ 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 的刻度是在右邊
- 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')`

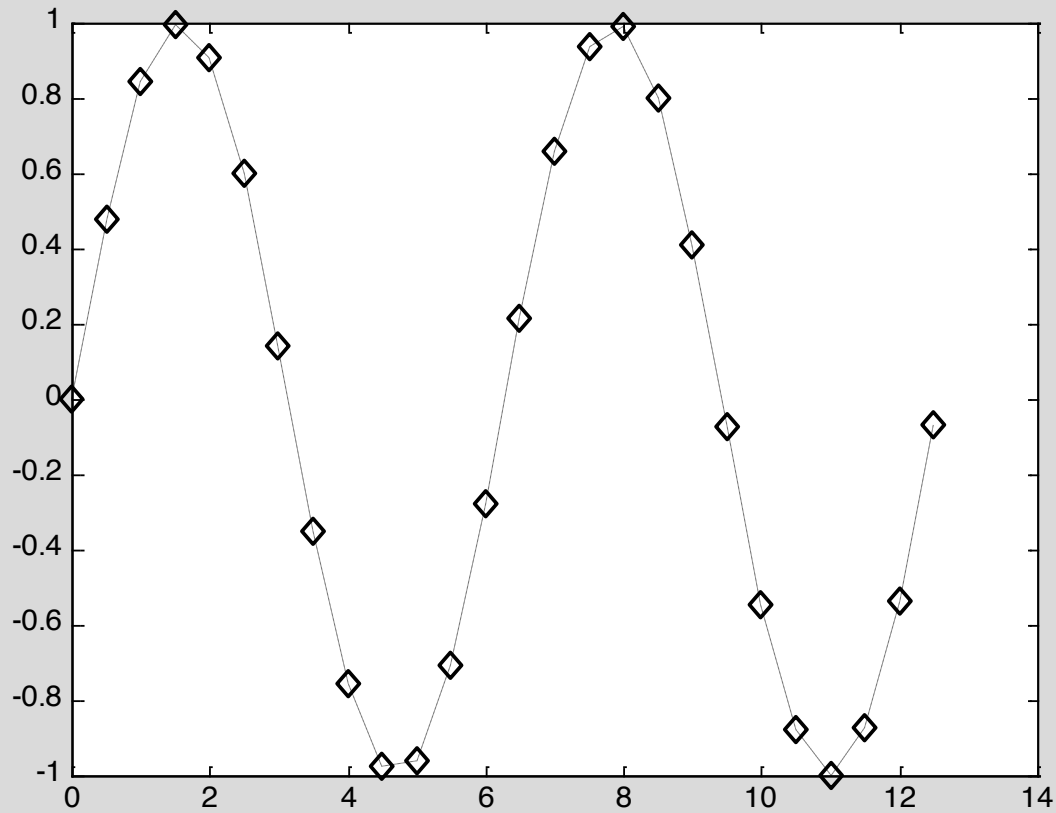
- **C**：曲線的顏色(Colors)
- **L**：曲線的格式(Line Styles)
- **M**：曲線所用的線標(Markers)

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
```

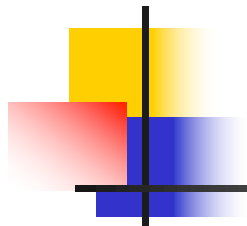
圖形控制範例-1 (II)





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)

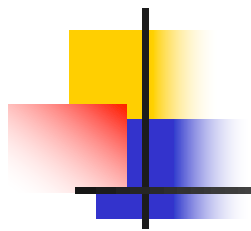


plot 指令的曲線格式

plot 指令的曲線格式字串	曲線格式
-	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	曲線符號符號
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	曲線符號符號
>	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

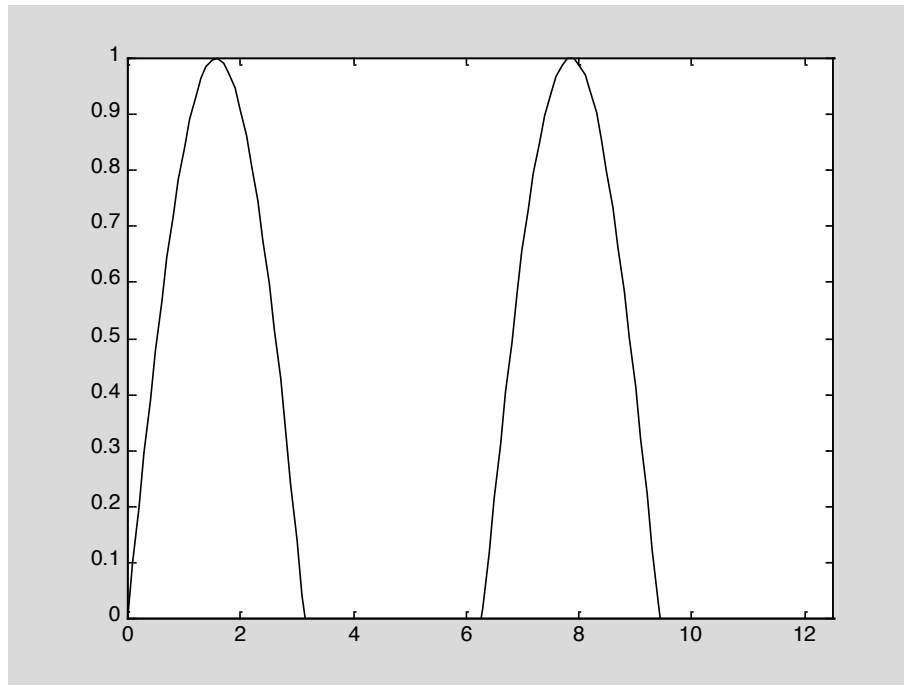
- **plot** 指令會根據座標點自動決定圖軸範圍
- 也可以使用 **axis** 指令指定圖軸範圍
 - 使用語法：
`axis([xmin, xmax, ymin, ymax])`
 - **xmin , xmax**：指定 x 軸的最小和最大值
 - **ymin , ymax**：指定 y 軸的最小和最大值

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

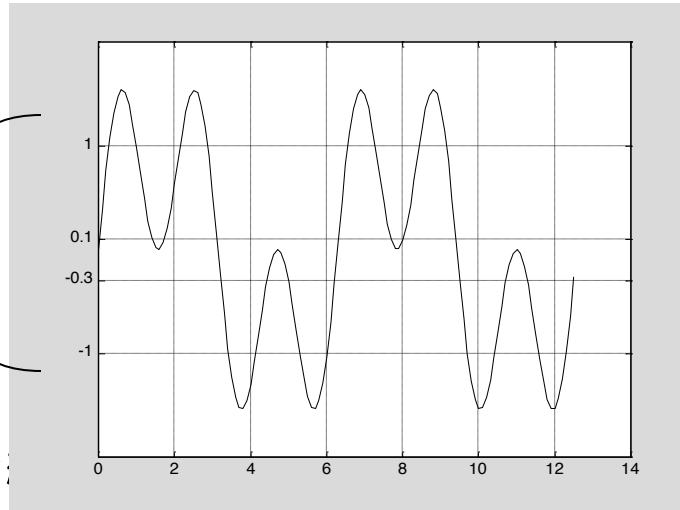
圖軸控制範例-2 (I)

- 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



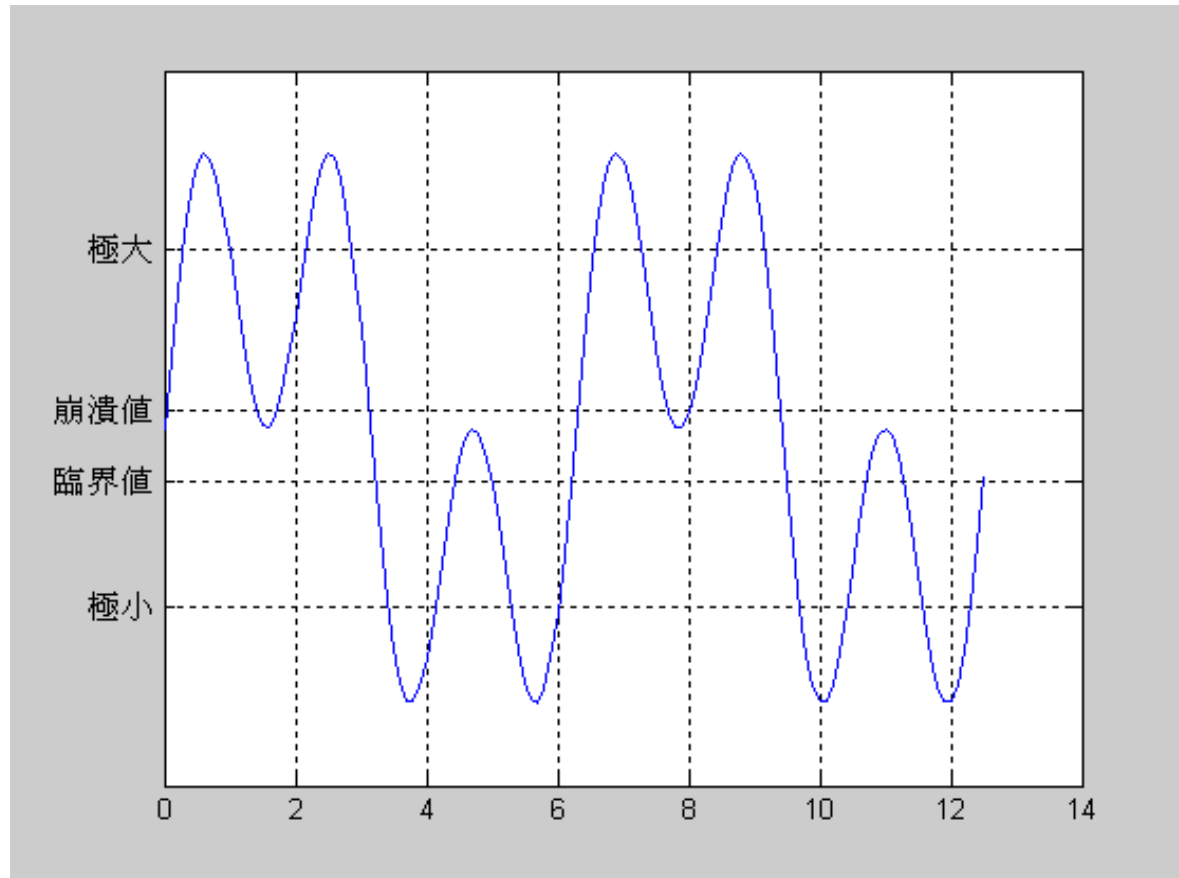
- `grid on`：加上格線
- `gca`：
 - `get current axis`的簡稱
 - 傳回目前使用中的圖軸
 - `gca`屬Handle Graphics的指令，第七章會有更詳細的說明

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 $m \times n$ 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，一系列

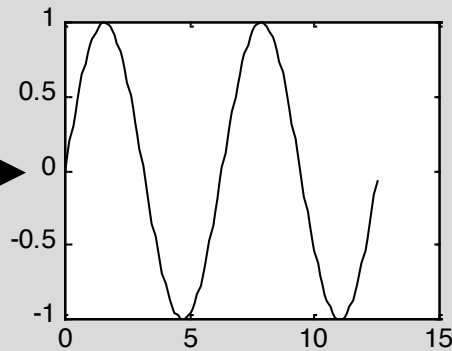
圖軸控制範例-4 (I)

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

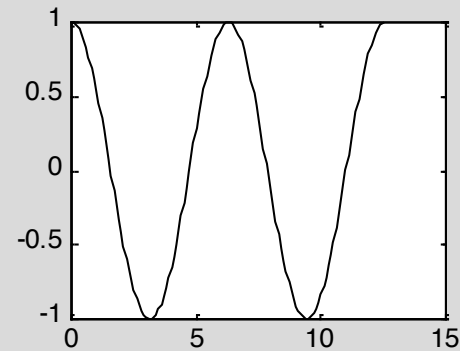
```
x = 0:0.1:4*pi;  
subplot(2, 2, 1); plot(x, sin(x));           % 此為左上角圖形  
subplot(2, 2, 2); plot(x, cos(x));           % 此為右上角圖形  
subplot(2, 2, 3); plot(x, sin(x).*exp(-x/5)); % 此為左下角圖形  
subplot(2, 2, 4); plot(x, x.^2);             % 此為右下角圖形
```

圖軸控制範例-4 (II)

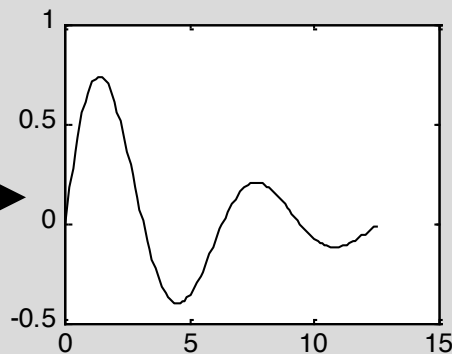
Subplot(2,2,1)



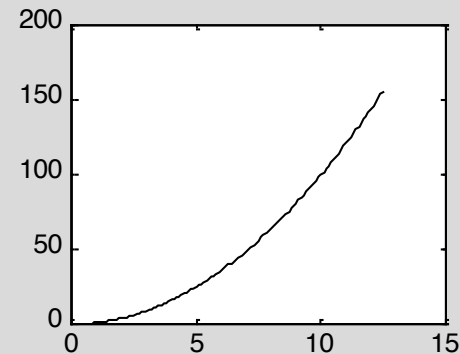
Subplot(2,2,2)



Subplot(2,2,3)



Subplot(2,2,4)



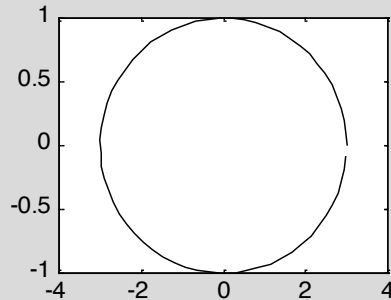
圖軸控制範例-5 (I)

- 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](#)

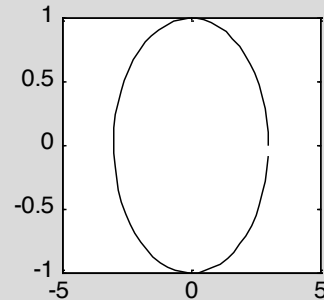
```
t = 0:0.1:2*pi;  
x = 3*cos(t);  
y = sin(t);  
subplot(2, 2, 1); plot(x, y); axis normal  
subplot(2, 2, 2); plot(x, y); axis square  
subplot(2, 2, 3); plot(x, y); axis equal  
subplot(2, 2, 4); plot(x, y); axis equal tight
```

圖軸控制範例-5 (II)

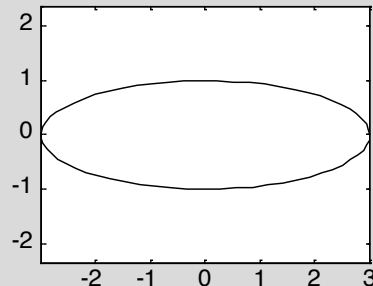
axis normal



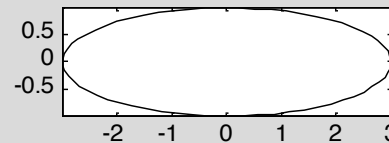
axis square



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
<code>axis normal</code>	使用預設長寬比(等於圖形長寬比)
<code>axis square</code>	長寬比例為 1
<code>axis equal</code>	長寬比例不變，但兩軸刻度一致
<code>axis equal tight</code>	兩軸刻度比例一致，且圖軸貼緊圖形
<code>axis image</code>	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
<code>colordef white</code>	圖軸背景為白色，視窗背景為淺灰色
<code>colordef black</code>	圖軸背景為黑色，視窗背景為暗灰色
<code>colordef none</code>	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

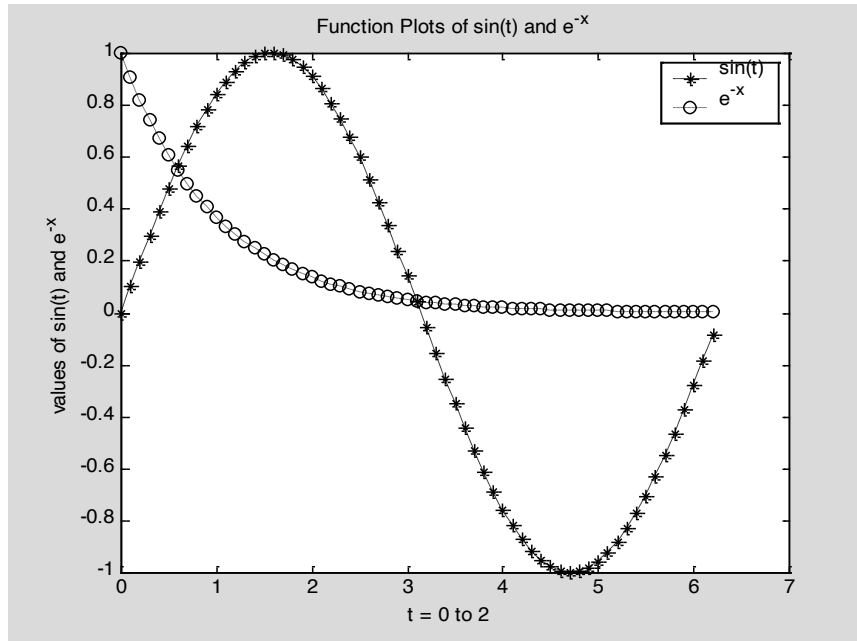
指令	說明
title	圖形的標題
xlabel	x 軸的說明
ylabel	y 軸的說明
zlabel	z 軸的說明(適用於立體繪圖)
legend	多條曲線的說明
text	在圖形中加入文字
gtext	使用滑鼠決定文字的位置

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：

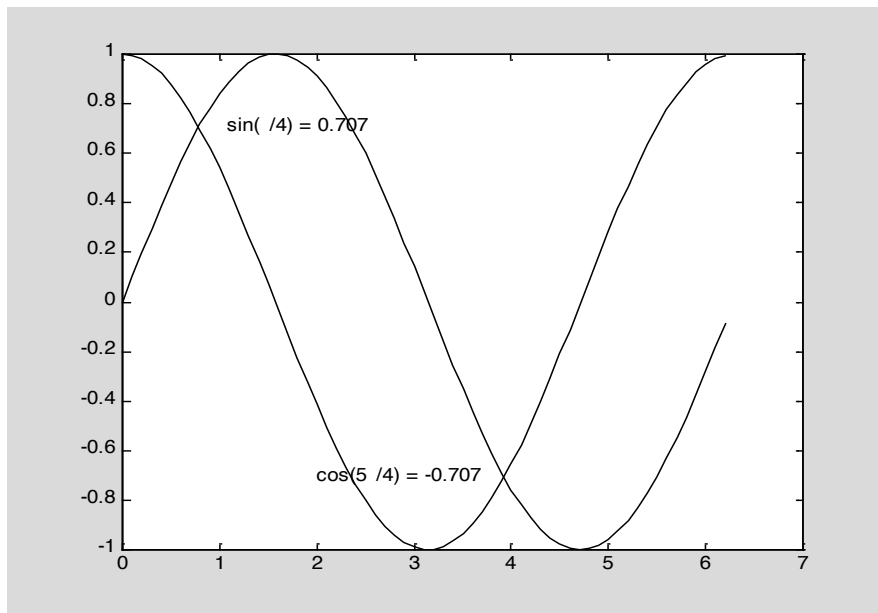
`text(x, y, string)`

- `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)



第九章: 矩陣的處理與運算

張智星

jang@cs.nthu.edu.tw

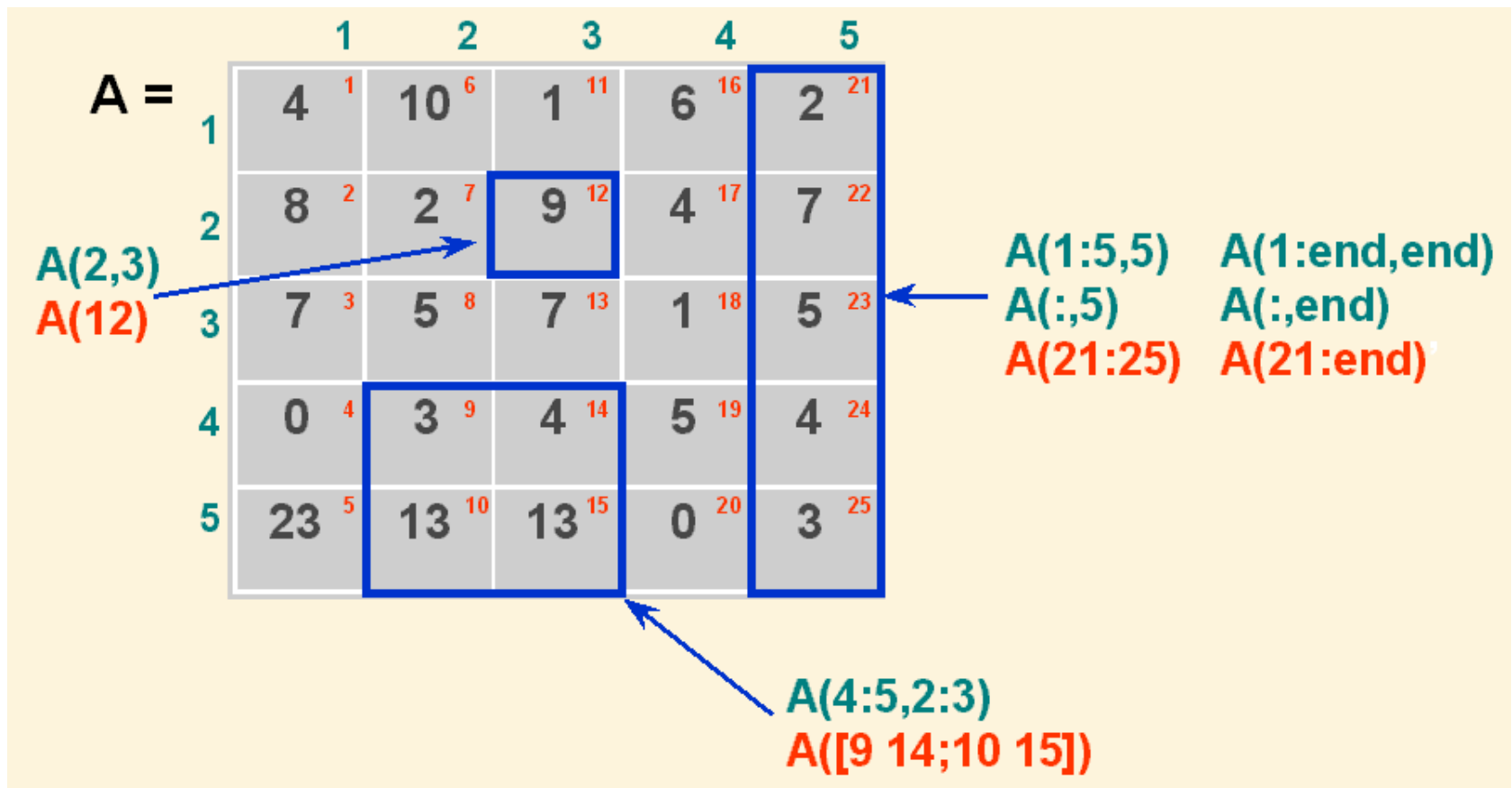
<http://www.cs.nthu.edu.tw/~jang>

清大資工系 多媒體檢索實驗室

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~ 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



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
 - $A(:, \text{end})$ - 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

- 可依次把矩陣 **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
 - $d = \text{diag}(B)$ - 取出矩陣 **B** 的對角線元素
- 用 **reshape** 指令來改變一個矩陣的維度
 - $C = \text{reshape}(B, 2, 8)$ - 將矩陣 **B** 排成 2×8 的新矩陣 **C**
- 注意!! **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
<code>zeros(m, n)</code>	產生維度為 $m \times n$ ，構成元素全為 0 的矩陣
<code>ones(m, n)</code>	產生維度為 $m \times n$ ，構成元素全為 1 的矩陣
<code>eye(n)</code>	產生維度為 $n \times n$ ，對角線的各元素全為 1，其他各元素全為 0 的單位矩陣
<code>pascal(m, n)</code>	產生維度為 $m \times n$ 的 Pascal 矩陣
<code>vander(m, n)</code>	產生維度為 $m \times n$ 的 Vandermonde 矩陣
<code>hilb(n)</code>	產生維度為 $n \times n$ 的 Hilbert 矩陣
<code>rand(m, n)</code>	產生 $[0, 1]$ 均勻分佈的亂數矩陣，其維度為 $m \times n$
<code>randn(m, n)</code>	產生 $\mu = 0, \sigma = 1$ 的正規分佈亂數矩陣，其維度為 $m \times n$
<code>magic(n)</code>	產生維度為 $n \times n$ 的魔方陣，其各個直行、橫列及兩對角線的元素和都相等

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
```

```
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];
```

```
>> B = 2*A
```

```
B =  
246 884
```

```
>> C = A/3
```

```
C =
```

```
41.0000 147.3333
```

- 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  
D = A./B  
E = A.^2
```

% 注意「*」前面的句點

% 注意「/」前面的句點

% 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p-norm

- 一個向量 a 的 p -norm 可以定義為

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

$p=2$ 時，此即為向量 a 的長度，或稱歐氏長度 (Euclidean Length)

- 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
```

$$\begin{aligned} \text{norm}(a,1) &= \sum_i |a_i| \\ \text{norm}(a,\text{inf}) &= \max_i |a_i| \end{aligned}$$

矩陣的p-norm

- 一個矩陣 A 的 p -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 p -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始向量 **x** 的位置
- **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 [-231,231-1]
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 [0,232-1]
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，但不可直接進行數學運算

```
>> uint8(20) == 20      % 可比較大小
```

```
ans =  
     1
```

```
>> uint8(20)-20         % 無法進行數學運算
```

```
??? Error using ==> -
```

```
Function '-' not defined for variables of class 'uint8'.
```

- 若要進行數學運算，需先用 **double** 指令將之轉成雙精準浮點數才能進行



MATLAB 程式設計入門篇 M檔案

張智星

jang@cs.nthu.edu.tw

<http://www.cs.nthu.edu.tw/~jang>

清大資工系 多媒體檢索實驗室

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

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  
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

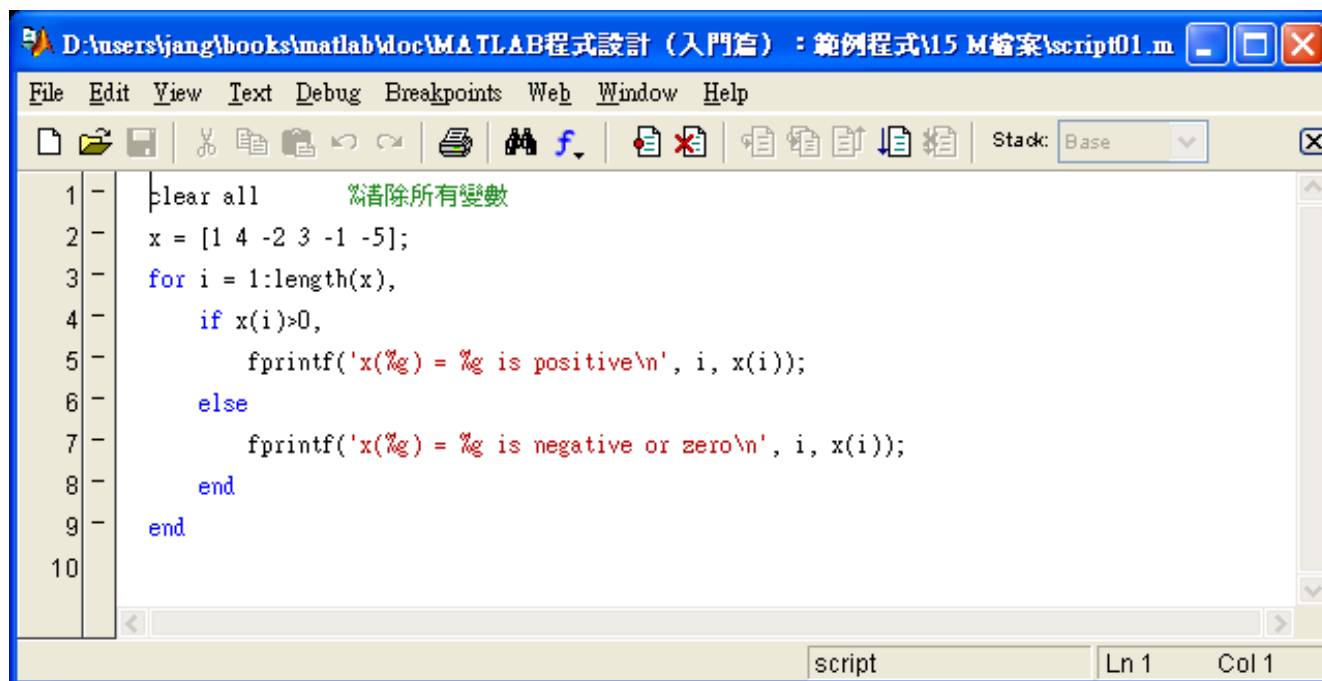
# M 檔案編輯器 (I)

- 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：



```
1 - clear all %清除所有變數
2 - x = [1 4 -2 3 -1 -5];
3 - for i = 1:length(x),
4 - if x(i)>0,
5 - fprintf('x(%g) = %g is positive\n', i, x(i));
6 - else
7 - fprintf('x(%g) = %g is negative or zero\n', i, x(i));
8 - end
9 - end
10 -
```

# 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」

功能鍵**F12**: 設定或解除中斷點  
功能鍵**F5**: 除錯模式下執行程式

## 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，均可執行某目錄內的函數
  - 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 nargsout == 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」（**Local Variables**）

# 全域變數的使用 (I)

- 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
  - 使用前一定要宣告
  - 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**，節省剖析的時間

The following slides are from  
Prof. Paul Söderlind

University of St. Gallen, Switzerland

<http://home.tiscalinet.ch/paulsoderlind/>

# Lecture S1

## Symbolic Math Using MATLAB

### Part 1: Algebra and calculus

See the demos in MATLAB's help, the MATLAB [Symbolic Toolbox instructions](#) , and [Clarkson's on-line tutorial](#).

Based on the [Maple](#) symbolic computational engine ("kernel"), although with fewer commands and slightly more complex syntax.

# Some contents of MATLAB's symbolic toolbox

- Enter expressions in symbolic form
- Expand or simplify expressions
- Find symbolic roots, limits, minima, maxima
- Differentiate and integrate
- Generate Taylor series of functions
- Solve equations symbolically
- Solve simultaneous equations, even when non-linear.
- Variable precision arithmetic
- Plotting and more

Must first tell MATLAB that the variables to be used are to be symbolic rather than numeric (double) or characters. The easiest method is to use the `syms` command (see `>> help syms`).

```
>> syms x y a b c f g
```

(Alternately, the `sym` command can be used, e.g. `x=sym('x')`, `y=sym('y')`, `a=sym('a')`, `b=sym('b')`, etc.)

Notice how these variables are now designated in the Workspace window (or `>> whos`), as “sym” rather than “double” or “char,” with values of `<1x1 sym>`.



# Differentiation

See `>>help sym/diff`

Example: find  $\text{diff\_f} = \frac{df}{dx}$ , where  $f = e^{-ax}x^{3b}\sin(cx)$

with a, b and c being unspecified constants:

```
>> f=exp(-a*x)*x^(3*b)*sin(c*x), diff_f=diff(f,x)
```

- You can use the `simple` (see `>>help sym/simple`) command to tell MATLAB to generate other forms of this expression. MATLAB selects the shortest form.

```
>> simple(diff_f)
```

- Do you agree with MATLAB's choice (ans)?

```
>> f=exp(-a*x)*x^(3*b)*sin(c*x); diff_f=diff(f,x), simple(diff_f)
diff_f =
-a*exp(-a*x)*x^(3*b)*sin(c*x)+3*exp(-a*x)*x^(3*b)*b/x*sin(c*x)+exp(-a*x)*
x^(3*b)*cos(c*x)*c
simplify:
-exp(-a*x)*x^(3*b-1)*(a*sin(c*x)*x-3*b*sin(c*x)-cos(c*x)*c*x)
radsimp:
(-a*sin(c*x)*x+3*b*sin(c*x)+cos(c*x)*c*x)*exp(-a*x)*x^(3*b)/x
combine(trig):
(-a*exp(-a*x)*x^(3*b)*sin(c*x)*x+3*exp(-a*x)*x^(3*b)*b*sin(c*x)+exp(-a*x)
```

## Command Window

File Edit Debug Desktop Window Help

```
convert(tan):
-2*a*exp(-a*x)*x^(3*b)*tan(1/2*c*x)/(1+tan(1/2*c*x)^2)+6*exp(-a*x)*x^(3*b)
)*b/x*tan(1/2*c*x)/(1+tan(1/2*c*x)^2)+exp(-a*x)*x^(3*b)*(1-tan(1/2*c*x)^2
)/(1+tan(1/2*c*x)^2)*c
collect(x):
-a*exp(-a*x)*x^(3*b)*sin(c*x)+3*exp(-a*x)*x^(3*b)*b/x*sin(c*x)+exp(-a*x)*
x^(3*b)*cos(c*x)*c
mwcos2sin:
-a*exp(-a*x)*x^(3*b)*sin(c*x)+3*exp(-a*x)*x^(3*b)*b/x*sin(c*x)+exp(-a*x)*
x^(3*b)*cos(c*x)*c
ans =
-exp(-a*x)*x^(3*b-1)*(a*sin(c*x)*x-3*b*sin(c*x)-cos(c*x)*c*x)
```

# Multiple differentiation

To find  $\text{diff2}_f = \frac{d^2 f}{dx^2}$  we do `>> diff2_f=diff(f,x,2)`

Exercise: Determine the simplest form of  $\frac{d^2 g}{dy^2}$ ,

where  $g = \frac{1 - 3y^{2.1}}{(\sin^2 y)(4 + y)}$

```
>> g=(1-3*y^2.1)/(sin(y)^2*(4+y))
```


```
>> diff2_g=diff(g,y,2)
```

```
diff2_g =
```

```
-
```

```
693/100*y^(1/10)/sin(y)^2/(4+y)+126/5*y^(11/10)/sin(y)^3/(4+y)*cos(y)+63/5*y^(11/10)/sin(y)^2/(4+y)^2+6*(1-3*y^(21/10))/sin(y)^4/(4+y)*cos(y)^2+4*(1-3*y^(21/10))/sin(y)^3/(4+y)^2*cos(y)+2*(1-3*y^(21/10))/sin(y)^2/(4+y)+2*(1-3*y^(21/10))/sin(y)^2/(4+y)^3
```

Does simple simplify in this case?

 **Command Window**  
File Edit Debug Desktop Window Help  
ans =  
$$-693/100*y^{(1/10)}/\sin(y)^2/(4+y)+126/5*y^{(11/10)}/\sin(y)^3/(4+y)*\cos(y)+63/5*y^{(11/10)}/\sin(y)^2/(4+y)^2+6*(1-3*y^{(21/10)})/\sin(y)^4/(4+y)*\cos(y)^2+4*(1-3*y^{(21/10)})/\sin(y)^3/(4+y)^2*\cos(y)+2*(1-3*y^{(21/10)})/\sin(y)^2/(4+y)+2*(1-3*y^{(21/10)})/\sin(y)^2/(4+y)^3$$
  
>> diff2\_g  
diff2\_g =  
$$-693/100*y^{(1/10)}/\sin(y)^2/(4+y)+126/5*y^{(11/10)}/\sin(y)^3/(4+y)*\cos(y)+63/5*y^{(11/10)}/\sin(y)^2/(4+y)^2+6*(1-3*y^{(21/10)})/\sin(y)^4/(4+y)*\cos(y)^2+4*(1-3*y^{(21/10)})/\sin(y)^3/(4+y)^2*\cos(y)+2*(1-3*y^{(21/10)})/\sin(y)^2/(4+y)+2*(1-3*y^{(21/10)})/\sin(y)^2/(4+y)^3$$
  
>> |

- Suppose you want the value of  $d^2g/dy^2$  for  $y = 2$ .
- What does the following give you?  
`>> y = 2, diff2_g`
- So use subs command (see `>>help sym/subs`).  
`>> diff2_g2=subs(diff2_g,2)`
- If you have more than one variable, you must use the following syntax:  
`subs(expression,{var1,var2 ...},{value 1, value 2, ..})`
- Exercise: find diff\_f for  $x = 1$ ,  $a = 2$ ,  $b = 3$ ,  $c = 4$   
(remember  $f = e^{-ax}x^{3b}\sin(cx)$ ,  $\text{diff\_f}=df/dx$ )

- If you want diff\_f to retain its original form:

```
>> subs(diff_f,{x,a,b,c},{1,2,3,4})
```

```
ans = -1.0708
```

```
>> diff_f
```

```
diff_f = -exp(-a*x)*x^(3*b-1)*(a*sin(c*x)*x-
3*b*sin(c*x)-cos(c*x)*c*x)
```

- If you want diff\_f to include these substitutions:

```
>> diff_f=subs(diff_f,{x,a,b,c},{1,2,3,4}), diff_f
```

```
diff_f = -1.0708
```

```
diff_f = -1.0708
```

# Indefinite Integrals

See >> help sym/int

- Find the indefinite integral  $\text{int\_g} = \int g dx$  ,  
where  $g = e^{-ax}\sin(cx)$  .  
`>> g = exp(-a*x)*sin(c*x), int_g=int(g,x)`
- Check by differentiating: `>> diff(int_g,x)`  
Does this return g?
- `>> simple(diff(int_g,x))`
- Repeat, adding a constant of integration A to the integral.



```
>> clear, syms a c x g A
```

```
>> g = exp(-a*x)*sin(c*x), int_g=int(g,x)+A
```

```
g =
exp(-a*x)*sin(c*x)
int_g =
-c/(a^2+c^2)*exp(-a*x)*cos(c*x)-a/(a^2+c^2)*exp(-
a*x)*sin(c*x)+A
```

```
>>gid=simple(diff(int_g,x))
```

```
gid =
exp(-a*x)*sin(c*x)
```

# Definite integrals

Suppose we want  $\int_{-\pi}^{+\pi} g dx$  with  $g$  defined as before.

The syntax is:

```
>> int_def_g=int(g,x,-pi,pi)
int_def_g =
-(c*cos(pi*c)+a*sin(pi*c)exp(pi*a)^2*c*cos(pi*c)
+exp(pi*a)^2*a*sin(pi*c))/exp(pi*a)/(a^2+c^2)
```

Now determine the numerical value of `int_def_g` for  $a=2$  and  $c=4$ .

- `>> subs(int_def_g,{a,c},{2,4})`  
`ans =107.0980`
- Sometimes MATLAB is unable to find an analytical solution for a definite integral, e.g.:  
`>> int(exp(sin(x)),x,0,1)`
- In such cases, the numerical procedure `quad` may be used, e.g.:  
`>> quad('exp(sin(x))',0,1)`
- `quad` is **not** part of the symbolic toolbox

# Limits

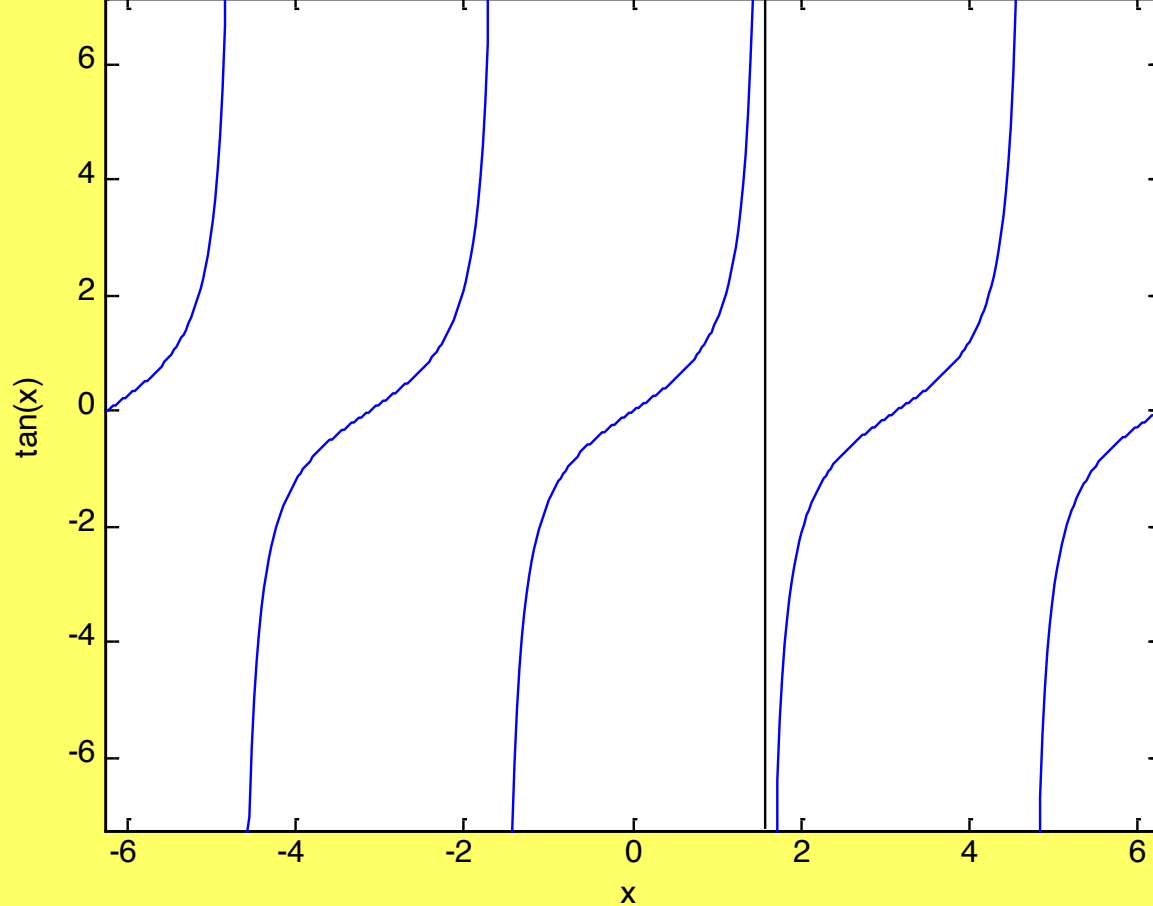
The symbolic toolbox also contains a command for finding limits of expressions, e.g.:

$$\lim_{x \rightarrow 0} \left( \frac{\sin(ax)}{x} \right)$$

```
>> clear, syms x a, value=limit(sin(a*x)/x,x,0)
```

There are cases where the limit depends on the direction from which it is approached. Consider the plot of  $\tan(x)$  versus  $x$  generated by

```
>> ezplot('tan(x)')
```



What is the value of  $\tan(x)$  at  $x = \pi/2$ ? Using MATLAB's numeric engine:

```
>> tan(pi/2)
```

```
ans =
```

```
1.6331e+016
```

# Approach when limit depends on direction

- `>> value_left = limit(tan(x),x,pi/2,'left'),`
- `>> value_right = limit(tan(x),x,pi/2,'right')`
- Conclusions?
- MATLAB's numeric engine gives a finite, albeit very large, value. Big mistakes could result if this were used in subsequent calculations.
- The symbolic engine gives both values correctly, but to get them you must add the additional argument telling MATLAB from which direction to approach the limit ('left' beginning with smaller values of  $x$  and increasing to the limit, and 'right' beginning with larger values of  $x$  and decreasing to the limit.

# Review Questions

- What command do you use to tell MATLAB that  $r$ ,  $s$  and  $t$  are to be symbolic?
- `>> syms r s t`
- What command do you use to differentiate  $\sin^2 t$  ?
- `>> diff(sin(t)^2,t)`
- What command do you use to integrate  $e^s$  from 0 to  $r$ ?
- `>> int(exp(s),s,0,r)`
- What command do you use to evaluate the above integral for  $r = \pi/2$ ?
- `>> subs(int(exp(s),s,0,r),r,pi/2)`

- How do you find  $\cos(\alpha\beta)/\beta$  for  $\beta = \pi/2$ ?
- `>> syms alpha beta`  
`>> limit(cos(alpha*beta)/beta,beta,pi/2)`
- What command do you use to find other forms of  $(3+4x)^4+(2-x)^3+(x/2+10)^2+(3-x)$  ?
- `>> simple((3+4*x)^4+(2-x)^3+(x/2+10)^2+(3-x))`