Chapter 2
MATLAB 基本功能介紹

Hung-Yuan Fan (范洪源)

Department of Mathematics,
National Taiwan Normal University, Taiwan

Spring 2020

Outline

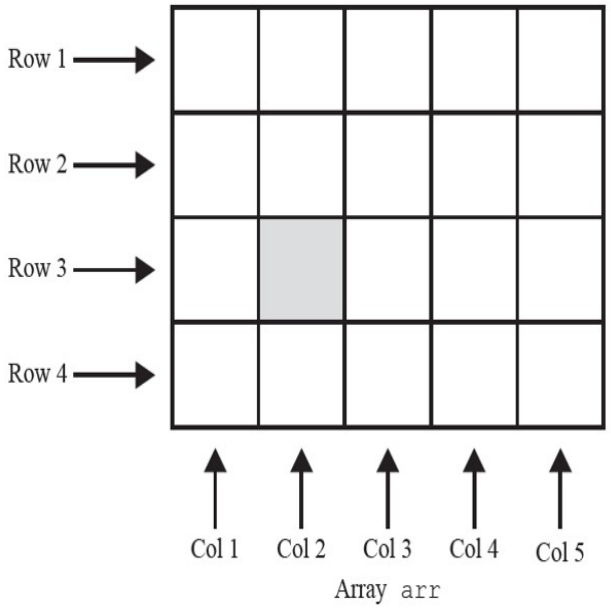
Lecture 1
變數與陣列

陣列 (Array)

- 在 MATLAB 程式裡，資料的基本單位是**陣列 (array)**。

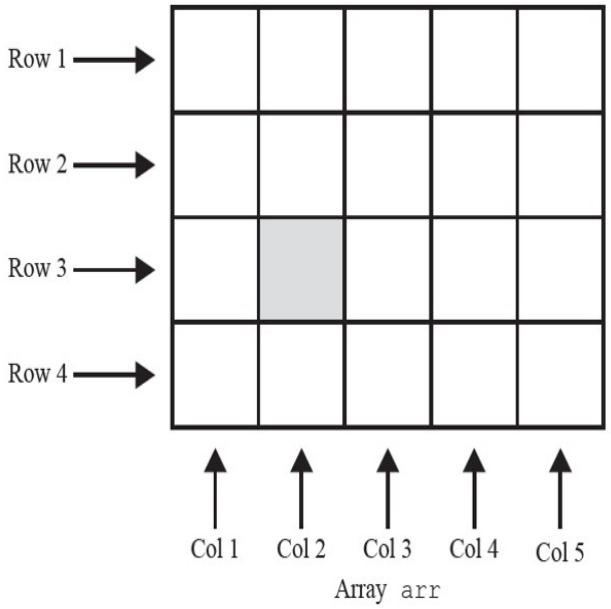
陣列 (Array)

- 在 MATLAB 程式裡，資料的基本單位是**陣列（array）**。
- 陣列是由一群排成行列結構的資料值所組成，並在程式中擁有獨一無二的名稱。



陣列 (Array)

- 在 MATLAB 程式裡，資料的基本單位是**陣列（array）**。
- 陣列是由一群排成行列結構的資料值所組成，並在程式中擁有獨一無二的名稱。



- 陣列可以被歸類為是一種**向量（vectors）**或是**矩陣（matrices）**。
- 陣列大小（array size）是由陣列的行數及列數來決定的。

向量與矩陣

- 向量通常被用來描述成**一維陣列**。
- 矩陣通常被用來描述**二維陣列**。
- 純量 (scalars) 在 MATLAB 中被視為是**一行一列的陣列**。

向量與矩陣

- 向量通常被用來描述成**一維陣列**。
- 矩陣通常被用來描述**二維陣列**。
- 純量 (scalars) 在 MATLAB 中被視為是**一行一列的陣列**。

陣列	大小
$a = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$	這是 3×2 的矩陣，含有 6 個元素。
$b = \begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix}$	這是 1×4 的矩陣，含有 4 個元素，亦算是一個**列向量 (row vector)**。
$c = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$	這是 3×1 的矩陣，含有 3 個元素，亦算是一個**行向量 (column vector)**。

MATLAB 的變數

- 變數 (variables) 是一個**使用者設定名稱的陣列**。

MATLAB 的變數

- 變數 (variables) 是一個**使用者設定名稱的陣列**。
- 實體上，是由一塊記憶體區域所組成。

MATLAB 的變數

- 變數 (variables) 是一個**使用者設定名稱的陣列**。
- 實體上，是由一塊記憶體區域所組成。
- 變數名稱**第一個字必須是文字**, 而其後的字可以使用文字、數字及底線字元 (_)任意組合而成。

MATLAB 的變數

- 變數 (variables) 是一個**使用者設定名稱的陣列**。
- 實體上，是由一塊記憶體區域所組成。
- 變數名稱**第一個字必須是文字**, 而其後的字可以使用文字、數字及底線字元 (_)任意組合而成。
- 變數名稱只有**前 63 個字元**是有意義的，且**大小寫有區分!**

MATLAB 的變數

- 變數 (variables) 是一個**使用者設定名稱的陣列**。
- 實體上，是由一塊記憶體區域所組成。
- 變數名稱**第一個字必須是文字**, 而其後的字可以使用文字、數字及底線字元 (_)任意組合而成。
- 變數名稱只有**前 63 個字元**是有意義的，且**大小寫有區分!**
- 任何時候只要指定數值給變數，便能直接產生變數，而其變數型態是由指定給變數的資料類型所決定。

良好的程式設計習慣

- 請確定您的變數名稱在前 63 個字原是獨一無二的。
- 記得給變數一個**具描述性且容易記憶**的名字。舉例來說，貨幣的匯率可以被命名為 `exchange_rate`。

良好的程式設計習慣

- 請確定您的變數名稱在前 63 個字原是獨一無二的。
- 記得給變數一個**具描述性且容易記憶**的名字。舉例來說，貨幣的匯率可以被命名為 `exchange_rate`。
- 為每個程式加上註解 (data dictionary):
  - 列出程式中每個使用變數的定義，包括變數內容描述及其物理單位。
  - 當您或其他人在日後需要修改程式時，這類註解便會變得十分重要。

良好的程式設計習慣

- 請確定您的變數名稱在前 63 個字原是獨一無二的。
- 記得給變數一個**具描述性且容易記憶**的名字。舉例來說，貨幣的匯率可以被命名為 `exchange_rate`。
- 為每個程式加上註解 (data dictionary):
  - 列出程式中每個使用變數的定義，包括變數內容描述及其物理單位。
  - 當您或其他人在日後需要修改程式時，這類註解便會變得十分重要。
- 請確定每次使用同一個變數時，其名稱所使用字母的大寫或小寫必須完全一致。
- 以小寫字母命名變數名稱是一個很好的程式技巧。

常用變數類型

- double型態的變數:
  - 包含了 **64 位元雙倍精度浮點數**的數字或陣列。

  - 它們能處理**實數、虛數或複數**。
  - 變數 i 和 j 的內建預設值是純虛數 $\sqrt{-1}$。
  - 正數表示範圍從10^{-308} 到 10^{308}，而且具有 **15 到 16 個十進位有效位數**。
  - 範例: >> var1 = 10.5; 或是 >> var2 = 10 + 10i;

常用變數類型

- double型態的變數:
  - 包含了 **64 位元雙倍精度浮點數**的數字或陣列。

  - 它們能處理**實數、虛數或複數**。
  - 變數 i 和 j 的內建預設值是純虛數 $\sqrt{-1}$。
  - 正數表示範圍從10^{-308} 到 10^{308}，而且具有 **15 到 16 個十進位有效位數**。
  - 範例: >> var1 = 10.5; 或是 >> var2 = 10 + 10i;
- char型態的變數:
  - 字元陣列: 每一個陣列元素為一個字元 (character)，且每個字元佔 2 Bytes 的記憶體空間。
  - 這類陣列是用來儲存**字元字串**的資料。
  - 範例: >> date = 'Sep. 24, 2003.'; % 變數 date 是一個 1×14 的字元陣列 (char array)。

MATLAB 的資料型態

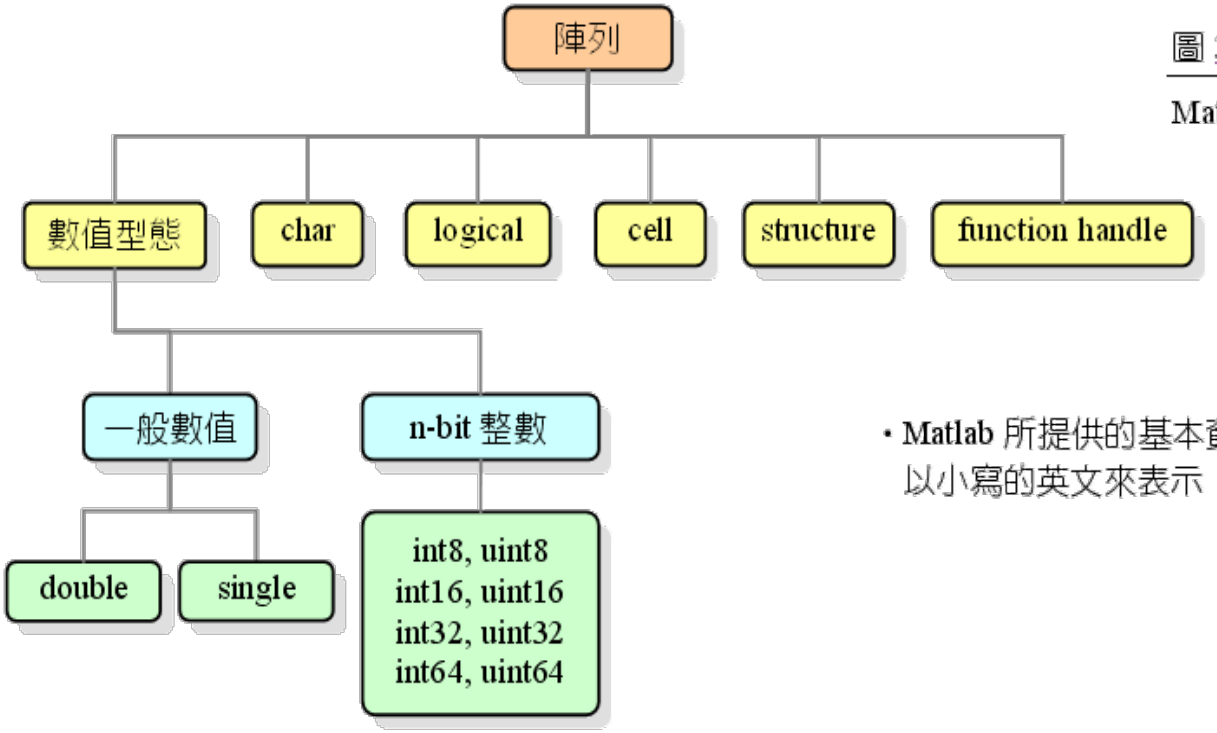


圖 3.1.1

Matlab 提供的資料型態

- Matlab 所提供的基本資料型態以小寫的英文來表示

- **高度類型化 (Strongly-typed) 語言**: C、PASCAL
- **低度類型化 (Weakly-typed) 語言**: MATLAB

Lecture 2
MATLAB 變數的初始化

如何設定變數的內容?

三種用來初始化變數的方式

1. 利用宣告的方式指定資料給變數。

   var = expression;

2. 從鍵盤輸入資料給變數。

   var = input('文字提示字串');

3. 從檔案讀取資料。(Lecture 6 將會討論)

宣告式初始化變數

宣告式一般的形式為：

```
var = expression;
```

宣告式初始化變數

宣告式一般的形式為：

```
var = expression;
```

- var 是變數的名稱。

宣告式初始化變數

宣告式一般的形式為：

```
var = expression;
```

- var 是變數的名稱。
- expression可以是一個純量常數、陣列、常數組合、其他變數及數學運算公式。

宣告式初始化變數

宣告式一般的形式為：

```
var = expression;
```

- var 是變數的名稱。
- expression可以是一個純量常數、陣列、常數組合、其他變數及數學運算公式。
- 宣告式尾端的分號 “;” 也可移除。

宣告式初始化變數

宣告式一般的形式為：

var = expression;

- var 是變數的名稱。
- expression可以是一個純量常數、陣列、常數組合、其他變數及數學運算公式。
- 宣告式尾端的分號 “;” 也可移除。
- 等號 “=” 可視為一個指派算子 (assignment operator)。
- 範例: var = 40i; var2 = var/5;
  x = 1; y = 2;
  array = [1 2 3 4];

MATLAB 的陣列敘述式

`[3 . 4]`	這敘述會產生 1×1 的陣列（純量），含有數值 3.4。在此情況下，方括號並不是必須的。
`[1.0  2.0  3.0]`	這敘述會產生 1×3 的陣列，含有列向量 [1 2 3]。
`[1.0; 2.0; 3.0]`	這敘述會產生 3×1 的陣列，含有行向量 $\begin{bmatrix} 1 \\ 2 \end{bmatrix}$。
`[1, 2, 3; 4, 5, 6]`	這敘述會產生 2×3 的陣列，含有矩陣 $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$。
`[1, 2, 3` `4, 5, 6]`	這敘述會產生 2×3 的陣列，含有矩陣 $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$。 第一指令列的尾端結束了矩陣第一列的輸入。
`[]`	這敘述會產生**空陣列（empty array）**，不含任何列或行（注意這與一個只含零的陣列完全不同）。

Remarks

- 在陣列裡的每列元素個數必須相同，而且每欄的元素個數也必須相同。
- 定義一個違反此項規定的陣列，都會導致執行上的錯誤。

Remarks

- 在陣列裡的每列元素個數必須相同，而且每欄的元素個數也必須相同。
- 定義一個違反此項規定的陣列，都會導致執行上的錯誤。
- **錯誤的範例**: >> [1 2; 3 4 5]

Remarks

- 在陣列裡的每列元素個數必須相同，而且每欄的元素個數也必須相同。
- 定義一個違反此項規定的陣列，都會導致執行上的錯誤。
- **錯誤的範例**: >> [1 2; 3 4 5]
- 在 MATLAB 宣告式的結尾加上分號，可停止在指令視窗中產生數值結果的回應，這將會大大地加快程式的執行速度。

Remarks

- 在陣列裡的每列元素個數必須相同，而且每欄的元素個數也必須相同。
- 定義一個違反此項規定的陣列，都會導致執行上的錯誤。
- **錯誤的範例**: >> [1 2; 3 4 5]
- 在 MATLAB 宣告式的結尾加上分號，可停止在指令視窗中產生數值結果的回應，這將會大大地加快程式的執行速度。
- **如果要除錯並檢查一個宣告式的執行結果，可將其尾端的分號拿掉，以便顯示執行結果在指令視窗上。**

以快捷敘述式初始化變數

- **冒號算子（colon operator）**可藉著指定數列中的第一個數值、遞增 (減) 值及最後數值，來產生一整串數列。

  `first:incr:last`

以快捷敘述式初始化變數

- **冒號算子 (colon operator)** 可藉著指定數列中的第一個數值、遞增 (減) 值及最後數值，來產生一整串數列。
  `first:incr:last`
  - 範例: >> a = 1 : 4
    a =
    1 2 3 4
  - 範例: >> b = 5 : −1 : 1
    b =
    5 4 3 2 1

以快捷敘述式初始化變數

- **冒號算子 (colon operator)** 可藉著指定數列中的第一個數值、遞增 (減) 值及最後數值，來產生一整串數列。

  `first:incr:last`

  - 範例: `>> a = 1 : 4`

    ```
    a =
       1   2   3   4
    ```

  - 範例: `>> b = 5 : -1 : 1`

    ```
    b =
       5   4   3   2   1
    ```

- **轉置算子 (transpose operator)** (') 對陣列作用的結果，會造成行與列的互換。

  - 範例: `>> c = [a'  2*a' ]`

    ```
    c =
       1   2
       2   4
       3   6
       4   8
    ```

以內建函式初始化變數

函式	目的
zeros(n)	產生一個 n × n 的零矩陣。
zeros(m,n)	產生一個 m × n 的零矩陣。
zeros(size(arr))	產生一個與 arr 大小相同的零矩陣。
ones(n)	產生一個 n × n 的全 1 矩陣。
ones(m,n)	產生一個 m × n 的全 1 矩陣。
ones(size (arr))	產生一個與 arr 大小相同的全 1 矩陣。
eye(n)	產生一個 n × n 的單位矩陣。
eye(m,n)	產生一個 m × n 的單位矩陣。
length(arr)	傳回向量的長度，或者是一個陣列中各維度長度之最大值。
size(arr)	傳回 arr 陣列的列個數及行個數值。

範例

例如，使用函式 zeros 來初始化變數：

$$\begin{aligned}
&\mathtt{a} = \mathtt{zeros}(2);\\
&\mathtt{b} = \mathtt{zeros}(2,3);\\
&\mathtt{c} = [1\ \ 2; 3\ \ 4];\\
&\mathtt{d} = \mathtt{zeros}(\mathtt{size}(\mathtt{c}));
\end{aligned}$$

範例

例如，使用函式 zeros 來初始化變數：

$$\begin{aligned}
&\mathtt{a} = \mathtt{zeros}(2);\\
&\mathtt{b} = \mathtt{zeros}(2,3);\\
&\mathtt{c} = [1\ \ 2; 3\ \ 4];\\
&\mathtt{d} = \mathtt{zeros}(\mathtt{size}(\mathtt{c}));
\end{aligned}$$

這些宣告式會產生下列矩陣:

$$\mathtt{a} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \quad \mathtt{b} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

$$\mathtt{c} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \quad \mathtt{d} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

常用的陣列建立函式 (1/2)

表 4.3.1　常用的陣列建立函數

函 數	說 明
zeros(n)	建立一個 $n \times n$ 的全零矩陣
zeros($m,n,\ldots,p$)	建立一個 $m \times n \times \cdots \times p$ 的全零矩陣
ones(n)	建立一個 $n \times n$ 的全 1 矩陣
ones($m,n,\ldots,p$)	建立一個 $m \times n \times \cdots \times p$ 的全 1 矩陣
eye(n)	建立一個 $n \times n$ 的單位矩陣（對角線元素為 1，其它元素為 0）
eye(m,n)	建立一個 $m \times n$ ，且對角線為 1，其它元素為 0 的矩陣
diag(v)	以向量 v 為對角元素，建立一個矩陣
magic(n)	建立一個 $n \times n$ 的魔術方陣（magic square）

常用的陣列建立函式 (2/2)

表 4.3.2　以亂數來建立陣列之函數

函 數	說 明
randi(*imax*, *n*)	建立 $n \times n$ 個 1 到 *imax* 之間均勻分佈的整數亂數
randi(*imax*, [*m*, *n*, ⋯, *p*])	建立 $m \times n \times \cdots \times p$ 個1到 *imax* 之間均勻分佈的整數亂數
randi([*imin*, *imax*] , [*m*, *n*, ⋯, *p*])	同上，但整數亂數的範圍為 *imin* 到 *imax*
rand()	建立一個 0 到 1 之間均勻分佈的亂數
rand(*n*)	建立 $n \times n$ 個 0 到 1 之間均勻分佈的亂數
rand(*m*, *n*, ⋯, *p*)	建立 $m \times n \times \cdots \times p$ 個 0 到 1 之間均勻分佈的亂數
randn()	建立一個平均值為 0，標準差為 1 的常態分佈亂數
randn(*n*)	建立 $n \times n$ 個 0 到 1 之間常態分佈的亂數
randn(*m*, *n*, ⋯, *p*)	建立 $m \times n \times \cdots \times p$ 個 0 到 1 之間常態分佈的亂數
rng(*seed*)	設定亂數種子為 *seed*

以鍵盤輸入初始化變數

範例

-
```
>> in1 = input('Enter data: ');
Enter data:
```

以鍵盤輸入初始化變數

範例

- `>> in1 = input('Enter data: ');`
  `Enter data:` **1.23**

以鍵盤輸入初始化變數

範例

- `>> in1 = input('Enter data: ');`
  `Enter data:` **1.23**
- `>> in2 = input('Enter data: ', 's');`
  `Enter data:`

以鍵盤輸入初始化變數

範例

- `>> in1 = input('Enter data: ');`
  `Enter data:` **1.23**
- `>> in2 = input('Enter data: ', 's');`
  `Enter data:` **1.23**
- 變數 `in1` 儲存浮點數，但變數 `in2` 儲存字元字串。
  `>> whos`

以鍵盤輸入初始化變數

範例

- `>> in1 = input('Enter data: ');`
  `Enter data:` **1.23**
- `>> in2 = input('Enter data: ', 's');`
  `Enter data:` **1.23**
- 變數 `in1` 儲存浮點數，但變數 `in2` 儲存字元字串。
  `>> whos`

Name	Size	Bytes	Class
in1	1×1	8	double array
in2	1×4	8	char array

Lecture 3
子陣列

- 可選擇並使用 MATLAB 陣列的子集合，就像把它們當成個別的陣列使用一樣。

- 可選擇並使用 MATLAB 陣列的子集合，就像把它們當成個別的陣列使用一樣。
- 若想要選擇子陣列，只要在陣列名稱的後面加上括號，並在括號內填上所想要選擇的元素範圍。

範例

$>> \ \ \mathtt{arr2} = [1\ 2\ 3; -2\ \ -3\ \ -4; 3\ 4\ 5]$

範例

```
>> arr2 = [1 2 3; -2 -3 -4; 3 4 5]

arr2 =

   1   2   3
  -2  -3  -4
   3   4   5

>> arr2(1,:)    % 選取陣列 arr2 的第一列
```

範例

```
>> arr2 = [1 2 3; -2 -3 -4; 3 4 5]

arr2 =

   1   2   3
  -2  -3  -4
   3   4   5

>> arr2(1,:)    % 選取陣列 arr2 的第一列

ans =

   1   2   3
```

範例

```
>> arr2 = [1 2 3; -2 -3 -4; 3 4 5]

arr2 =

   1   2   3
  -2  -3  -4
   3   4   5

>> arr2(:, 1:2:3)   % 選取陣列 arr2 的第一行與第三行，
                    % 其結果和指令 arr2(:, [1 3]) 相同
```

範例

```
>> arr2 = [1 2 3; -2 -3 -4; 3 4 5]

arr2 =

   1    2    3
  -2   -3   -4
   3    4    5

>> arr2(:, 1:2:3)   % 選取陣列 arr2 的第一行與第三行，
                    % 其結果和指令 arr2(:, [1 3]) 相同

ans =

   1    3
  -2   -4
   3    5
```

使用 end 函式 (1/2)

當函式 end 應用在陣列的下標時，它會傳回該下標的**最大值**。

使用 end 函式 (1/2)

當函式 end 應用在陣列的下標時，它會傳回該下標的**最大值**。

一維陣列的範例

```
>> arr3 = [1 2 3 4 5 6 7 8];
>> arr3(5 : end)
```

使用 end 函式 (1/2)

當函式 end 應用在陣列的下標時，它會傳回該下標的**最大值**。

一維陣列的範例

```
>> arr3 = [1 2 3 4 5 6 7 8];
>> arr3(5 : end)

ans =

    5  6  7  8

>> arr3(end)
```

使用 end 函式 (1/2)

當函式 end 應用在陣列的下標時，它會傳回該下標的**最大值**。

一維陣列的範例

```
>> arr3 = [1 2 3 4 5 6 7 8];
>> arr3(5 : end)

ans =

    5  6  7  8

>> arr3(end)

ans =

    8
```

使用 end 函式 (2/2)

二維陣列的範例

$>>$ arr4 $= [1\ 2\ 3\ 4; 5\ 6\ 7\ 8; 9\ 10\ 11\ 12]$

使用 end 函式 (2/2)

二維陣列的範例

```
>> arr4 = [1 2 3 4; 5 6 7 8; 9 10 11 12]
arr4 =
   1   2   3   4
   5   6   7   8
   9  10  11  12
>> arr4(2:end,2:end)   % 其結果與arr4(2:3,2:4)相同
```

使用 end 函式 (2/2)

二維陣列的範例

```
>> arr4 = [1 2 3 4; 5 6 7 8; 9 10 11 12]
arr4 =
   1   2   3   4
   5   6   7   8
   9  10  11  12
>> arr4(2:end,2:end)   % 其結果與arr4(2:3,2:4)相同
ans =
   6   7   8
  10  11  12
```

在宣告式左邊使用子陣列

二維陣列的範例 (承上頁)

```
>> arr4 = [1 2 3 4; 5 6 7 8; 9 10 11 12]
arr4 =
   1   2   3   4
   5   6   7   8
   9  10  11  12
>> arr4(1:2, [1 4]) = [20 21; 22 23]
```

在宣告式左邊使用子陣列

二維陣列的範例 (承上頁)

```
>> arr4 = [1 2 3 4; 5 6 7 8; 9 10 11 12]
arr4 =
   1   2   3   4
   5   6   7   8
   9  10  11  12
>> arr4(1:2, [1 4]) = [20 21; 22 23]
arr4 =
  20   2   3  21
  22   6   7  23
   9  10  11  12
```

二維陣列的範例 (承上頁)

$>>$ arr4 $=$ [20 21; 22 23] % arr4 的內容被 2×2 矩陣覆蓋

二維陣列的範例 (承上頁)

$>>$ arr4 $=$ [20 21; 22 23] % arr4 的內容被 2×2 矩陣覆蓋

arr4 $=$

20	21
22	23

Remarks

- 當宣告式左邊包含子陣列時，在等號兩邊的子陣列形狀必須相同; 否則，MATLAB 將會產生錯誤的訊息。

二維陣列的範例 (承上頁)

```
>> arr4 = [20 21; 22 23]    % arr4 的內容被 2 × 2 矩陣覆蓋
arr4 =
   20   21
   22   23
```

Remarks

- 當宣告式左邊包含子陣列時，在等號兩邊的子陣列形狀必須相同; 否則，MATLAB 將會產生錯誤的訊息。
- 請弄清楚指定數值給一個子陣列，與指定數值給一個陣列之間的區別。MATLAB 對這兩種情況的處理方法是不同的。

二維陣列的範例

```
>> arr4 = [1 2 3 4; 5 6 7 8; 9 10 11 12]
arr4 =
   1   2   3   4
   5   6   7   8
   9  10  11  12
>> arr4(1:2, 1:2) = 1   % 左上角 2×2 子陣列變成全 1 矩陣
```

分配純量給子陣列

二維陣列的範例

```
>> arr4 = [1 2 3 4; 5 6 7 8; 9 10 11 12]
arr4 =
   1   2   3   4
   5   6   7   8
   9  10  11  12
>> arr4(1:2, 1:2) = 1   % 左上角 2×2 子陣列變成全 1 矩陣
arr4 =
   1   1   3   4
   1   1   7   8
   9  10  11  12
```

擴增陣列的行或列 (1/2)

增加二維陣列的行數 (承上頁)

```
>> arr4 = [1 2 3 4; 5 6 7 8; 9 10 11 12]
arr4 =
   1    2    3    4
   5    6    7    8
   9   10   11   12
>> arr5 = [arr4 [1 1 1]']   % 全 1 行向量增至 arr5 的第 5 行
```

擴增陣列的行或列 (1/2)

增加二維陣列的行數 (承上頁)

```
>> arr4 = [1 2 3 4; 5 6 7 8; 9 10 11 12]
arr4 =
   1    2    3    4
   5    6    7    8
   9   10   11   12
>> arr5 = [arr4 [1 1 1]']   % 全 1 行向量增至 arr5 的第 5 行
arr5 =
   1    2    3    4    1
   5    6    7    8    1
   9   10   11   12    1
```

擴增陣列的行或列 (2/2)

增加二維陣列的列數 (承上頁)

```
>> arr4 = [1 2 3 4; 5 6 7 8; 9 10 11 12]
arr4 =
   1    2    3    4
   5    6    7    8
   9   10   11   12
>> arr6 = [arr4; [1 1 1 1]] % 全 1 列向量增至 arr6 的第 4 列
```

擴增陣列的行或列 (2/2)

增加二維陣列的列數 (承上頁)

```
>> arr4 = [1 2 3 4; 5 6 7 8; 9 10 11 12]
arr4 =
   1   2   3   4
   5   6   7   8
   9  10  11  12
>> arr6 = [arr4; [1 1 1 1]] % 全 1 列向量增至 arr6 的第 4 列
arr6 =
   1   2   3   4
   5   6   7   8
   9  10  11  12
   1   1   1   1
```

陣列的合併

表 4.4.3　陣列轉換函數

函 數	說 明
[A, B]	將陣列 A, B 橫向併排，組合成一個新的陣列
[A; B]	將陣列 A, B 垂直併排，組合成一個新的陣列
cat(dim, A, B, ...)	依 dim 所指定的方向合併（concatenate）陣列 A, B,...

以下宣告式的執行結果和前述方法相同:

- `>> arr5 = cat(2, arr4, ones(3,1))`
- `>> arr6 = cat(1, arr4, ones(1,4))`

刪除陣列的行或列 (1/2)

刪減二維陣列的行數 (承上頁)

```
>> arr4 = [1 2 3 4; 5 6 7 8; 9 10 11 12]
arr4 =
   1    2    3    4
   5    6    7    8
   9   10   11   12
>> arr4(:,3) = [ ]   % 刪除 arr4 的第 3 行
```

刪除陣列的行或列 (1/2)

刪減二維陣列的行數 (承上頁)

```
>> arr4 = [1 2 3 4; 5 6 7 8; 9 10 11 12]
arr4 =
   1    2    3    4
   5    6    7    8
   9   10   11   12
>> arr4(:,3) = [ ]   % 刪除 arr4 的第 3 行
arr4 =
   1    2    4
   5    6    8
   9   10   12
```

刪除陣列的行或列 (2/2)

刪減二維陣列的列數 (承上頁)

```
>> arr4 = [1 2 3 4; 5 6 7 8; 9 10 11 12]
arr4 =
   1    2    3    4
   5    6    7    8
   9   10   11   12
>> arr4(2, :) = [ ]   % 刪除 arr4 的第 2 列
```

刪除陣列的行或列 (2/2)

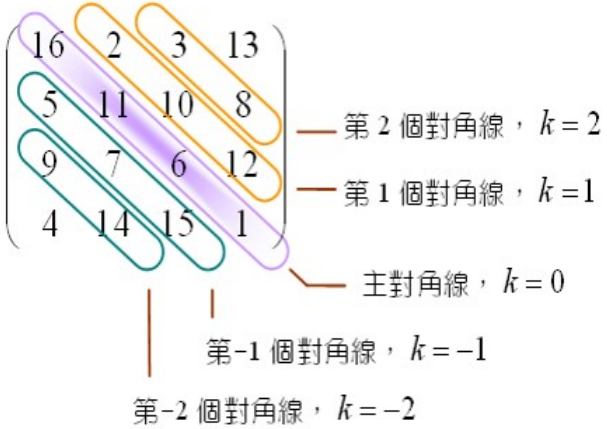
刪減二維陣列的列數 (承上頁)

```
>> arr4 = [1 2 3 4; 5 6 7 8; 9 10 11 12]
arr4 =
   1    2    3    4
   5    6    7    8
   9   10   11   12
>> arr4(2, :) = [ ]   % 刪除 arr4 的第 2 列
arr4 =
   1    2    3    4
   9   10   11   12
```

提取陣列的對角、右上或左下部分元素

表 4.4.1　陣列元素的提取函數

函　數	說　明
diag(*A*)	取出陣列 *A* 的主對角線（**main diagonal**）元素
diag(*A*,*k*)	取出陣列 *A* 的第 *k* 個對角線元素
triu(*A*)	取出陣列 *A* 之主對角線以上之元素，其它元素則設為 0（即上三角矩陣，**upper triangular matrix**）
triu(*A*,*k*)	取出陣列 *A* 之第 *k* 個對角線以上之元素，其它元素則設為 0
tril(*A*)	取出陣列 *A* 之主對角線以下之元素，其它元素則設為 0（即下三角矩陣，**lower triangular matrix**）
tril(*A*,*k*)	取出陣列 *A* 之第 *k* 個對角線以下之元素，其它元素則設為 0



對調或重排陣列的元素

表 4.4.2　陣列轉換函數

函　數	說　明
fliplr(A)	將陣列 A 的元素左右翻轉（flip left/right）
flipud(A)	將陣列 A 的元素上下翻轉（flip up/down）
flipdim(A,n)	將陣列 A 的元素依第 n 個維度翻轉
reshape(A,m,n,...,p)	將陣列 A 的元素依由上到下，由左到右的次序重新排列成一個 $m \times n \times \cdots \times p$ 的矩陣
repmat(A,m,n,...,p)	以陣列 A 為單位，將陣列 A 以類似排列磁磚的方式排成 $m \times n \times \cdots \times p$ 個陣列 A
rot90(A)	將陣列 A 逆時針旋轉 90°
rot90(A,k)	將陣列 A 逆時針旋轉 $k \times 90°$，k 為整數

函式 reshape 的範例 (1/2)

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

A =

     1     4     7
     2     5     8
     3     6     9
```

函式 reshape 的範例 (1/2)

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

A =

    1    4    7
    2    5    8
    3    6    9

>> vec = A(:) % 將陣列 A 行行相接!

vec =

    1
    2
    3
    4
    5
    6
    7
    8
    9
```

函式 reshape 的範例 (2/2)

```
>> B = reshape(vec,3,3)
B =
    1   4   7
    2   5   8
    3   6   9
```

Note: 將向量 vec 重排為 3×3 陣列 B，結果得到 B = A!

Lecture 4
特殊的數值

特殊的 MATLAB 保留字 (1/2)

變數和檔案名稱請盡量不要使用下列保留字:

函式	目的
pi	代表 π，有效數字 15 位。
i, j	代表 $i(\sqrt{-1})$ 的值。
Inf	這符號代表無窮大。通常是由於除以 0 所產生的結果。
NaN	這符號代表「不是數字」（Not-a-Number）。它是由未定義的數學運算所產生的，像 0 除以 0 就沒有定義。

特殊的 MATLAB 保留字 (2/2)

函式	目的
clock	這特殊的變數代表現在的日期與時間，是一個 6 個元素的列向量，分別是年、月、日、時、分、秒。
date	以字元字串代表現在的日期，例如：24-Nov-1998 的值。
eps	這變數是 “epsilon” 的縮寫，它代表在電腦上兩個數字間可表示的最小差異。
ans	如果一個表示式的運算結果沒有明確指定儲存在某個變數中，這個特殊變數是用來儲存此運算結果。

Lecture 5
顯示輸出資料

浮點數的顯示格式

MATLAB 的預設格式，是在小數點之後**顯示四位數字**。

```
>> x = 100.11

x =

    100.1100
```

浮點數的顯示格式

MATLAB 的預設格式，是在小數點之後**顯示四位數字**。

```
>> x = 100.11

x =

    100.1100

>> format short e
```

浮點數的顯示格式

MATLAB 的預設格式，是在小數點之後**顯示四位數字**。

```
>> x = 100.11

x =

   100.1100

>> format short e
>> y = 1001.1

y =

   1.0011e+003
```

浮點數的顯示格式

MATLAB 的預設格式，是在小數點之後**顯示四位數字**。

```
>> x = 100.11

x =

   100.1100

>> format short e
>> y = 1001.1

y =

   1.0011e+003

>> z = 0.00010011

z =

   1.0011e-004
```

format 指令的輸出顯示格式

指令格式	結果	舉例（註[1]）
format short	顯示 4 位小數（預設值）	12.3457
format long	顯示 14 位小數	12.34567890123457
format short e	顯示 5 個數字加冪次方	1.2346e+001
format short g	總共顯示 5 個數字（可加或不加冪次方）	12.346
format long e	顯示 15 位小數加冪次方	1.2345678901234 57e+001
format long g	總共顯示 15 個數字，加或不加冪次方	12.3456789012346
format bank	貨幣的格式	12.35
format hex	16 位元進位格式	4028b0fcd32f707a
format rat	顯示最接近的整數比例	1000/81
format compact	關閉額外換行功能	
format loose	恢復額外換行功能	
format +	只印出數值之正負數	+

註 [1] 表內之原始資料值為 12.345678901234567。

disp 函式

宣告式 disp(X) 只顯示變數 X 的內容，但不顯示變數名稱。

```
>> disp(pi)
```

disp 函式

宣告式 disp(X) 只顯示變數 X 的內容，但不顯示變數名稱。

```
>> disp(pi)
 3.1416
```

```
>> disp(-3+5i)
```

disp 函式

宣告式 disp(X) 只顯示變數 X 的內容，但不顯示變數名稱。

```
>> disp(pi)
 3.1416
```

```
>> disp(-3+5i)
 -3.0000 + 5.0000i
```

```
>> disp('This is a test!')
```

disp 函式

宣告式 disp(X) 只顯示變數 X 的內容，但不顯示變數名稱。

```
>> disp(pi)
 3.1416
```

```
>>  disp(-3 + 5i)
 -3.0000 + 5.0000i
```

```
>> disp('This is a test!')
This is a test!
```

```
>> disp(['The value of pi is ' num2str(pi)])
```

disp 函式

宣告式 disp(X) 只顯示變數 X 的內容，但不顯示變數名稱。

```
>> disp(pi)
 3.1416
```

```
>>  disp(-3 + 5i)
 -3.0000 + 5.0000i
```

```
>> disp('This is a test!')
This is a test!
```

```
>> disp(['The value of pi is ' num2str(pi)])
The value of pi is 3.1416
```

使用 fprintf 函式做格式化輸出

函式 fprintf 的一般型式

```
fprintf(format ,data)
```

- format: **格式字串**用以描述輸出資料的方式。
- data: 顯示一個或多個純量或陣列。

使用 fprintf 函式做格式化輸出

函式 fprintf 的一般型式

```
fprintf(format ,data)
```

- format: **格式字串**用以描述輸出資料的方式。
- data: 顯示一個或多個純量或陣列。

```
>> fprintf('The value of pi is %f \n', pi)
```

使用 fprintf 函式做格式化輸出

函式 fprintf 的一般型式

```
fprintf(format ,data)
```

- format: **格式字串**用以描述輸出資料的方式。
- data: 顯示一個或多個純量或陣列。

```
>> fprintf('The value of pi is %f \n', pi)
The value of pi is 3.141593
```

- 字元%f稱為**轉換字元 (conversion characters)**。
- 字元\n稱為**逸出字元 (escape characters)**。

函式 fprintf 格式字串中常用的轉換字元:

格式字串	結果
%d	以整數格式顯示數值。
%e	以指數格式顯示數值。
%f	以浮點數格式顯示數值。
%g	以浮點或指數格式來顯示數值，由何者較短為優先顯示。
\n	跳到新的一行。

傳換字串或格式碼的完整版語法:

表 3.3.1　格式化列印函數 fprintf() 的語法

函數	說 明
fprintf('*str*', e_1, e_2, ⋯)	依格式字串 *str* 所記載的格式碼，依序將 e_1, e_2 填入 *str* 中列印出來。下面列出了格式字串裡常用的格式碼： **%c**：列印字元 **%s**：列印字串 **%*m*d**：以 *m* 個欄位的寬度列印整數，若省略 *m*，則以最精簡的格式來列印 **%*m.n*f**：以 *n* 個小數位數，總共 *m* 個欄位的寬度列印數值，若省略 *m.n*，則以 6 個位數的小數來列印 **%*m.n*e**：同上，但以指數型式來列印數值 **%*m.n*g**：以 *m* 個欄位，*n* 個有效位數來列印數值。如果省略 *m.n*，則以最精簡的格式來列印

表 3.3.2　用於 fprintf() 裡的特殊字元

特殊字元	說 明
\n	換行
\t	跳格
' '	印出單引號
\\	印出反斜線
%%	印出百分比符號

格式碼 %m.nf 的範例

```
a=22;
fprintf('a=%6.3f\n',a);
```

a	=	2	2	.	0	0	0	\n

%6.3f，佔 6 個欄位，小數點以下 3 位

```
b=3.14159;
fprintf('b=%5.2f\n',b);
```

b	=		3	.	1	4	\n

%5.2f，佔 5 個欄位，小數點以下 2 位

範例

```
>> fprintf('The value of pi is %6.2f \n', pi)
The value of pi is   3.14
```

範例

```
>> fprintf('The value of pi is %6.2f \n', pi)
The value of pi is   3.14

>> x = 2*(1-2*i)^3;
>> disp(x)
-22.0000 + 4.0000i
```

範例

```
>> fprintf('The value of pi is %6.2f \n', pi)
The value of pi is   3.14

>> x = 2*(1-2*i)^3;
>> disp(x)
-22.0000 + 4.0000i
>> fprintf(' x = %8.4f \n', x)
x = -22.0000
```

函式 fprintf 無法顯示變數 x 的虛部!

Lecture 6
資料檔案

使用 save 指令儲存變數內容

- save filename: 將工作區裡所有的變數儲存至一個預設副檔名為 **MAT** 的檔案，其檔案名稱為filename.mat。

使用 save 指令儲存變數內容

- save filename: 將工作區裡所有的變數儲存至一個預設副檔名為 **MAT** 的檔案，其檔案名稱為filename.mat。
- save filename var1 var2 var3: 將變數 var1、var2、var3 儲存至檔案 filename.mat。
- 亦可使用 save('filename','var1','var2','var3')。

使用 save 指令儲存變數內容

- save filename: 將工作區裡所有的變數儲存至一個預設副檔名為 **MAT** 的檔案，其檔案名稱為filename.mat。
- save filename var1 var2 var3: 將變數 var1、var2、var3 儲存至檔案 filename.mat。
- 亦可使用 save('filename','var1','var2','var3')。
- save filename.txt var -**ascii**: 將變數 var 儲存至 **ASCII** 檔案 filename.txt。

使用 save 指令儲存變數內容

- save filename: 將工作區裡所有的變數儲存至一個預設副檔名為 **MAT** 的檔案，其檔案名稱為filename.mat。
- save filename var1 var2 var3: 將變數 var1、var2、var3 儲存至檔案 filename.mat。
- 亦可使用 save('filename','var1','var2','var3')。
- save filename.txt var -**ascii**: 將變數 var 儲存至 **ASCII** 檔案 filename.txt。
- 若使用 ASCII 編碼儲存，檔案名稱通常與變數名稱一致，而且也可取 filename.dat 作為檔名。

使用 load 指令載入變數內容

- `load filename`或是 `load filename.mat`: 將 **MAT** 檔案裡的所有變數都回復到檔案儲存前在工作區的狀態。

使用 load 指令載入變數內容

- `load filename`或是 `load filename.mat`: 將 **MAT** 檔案裡的所有變數都回復到檔案儲存前在工作區的狀態。
- `load filename.txt` 或是 `load filename.dat`: 將 **ASCII** 檔案裡的所有數據資料，以變數名稱`filename`儲存在 MATLAB 工作區內。

範例

```
>> x = 0:0.1:6;
>> y = sin(x);
>> save xy_points x y % 將變數 x 和 y 儲存在 MAT 檔案
```

範例

```
>> x = 0:0.1:6;
>> y = sin(x);
>> save xy_points x y % 將變數 x 和 y 儲存在 MAT 檔案
>> clear all % 將工作區的變數全部刪除
```

範例

```
>> x = 0:0.1:6;
>> y = sin(x);
>> save xy_points x y % 將變數 x 和 y 儲存在 MAT 檔案
>> clear all % 將工作區的變數全部刪除
>> whos % 查詢變數的狀態
>>      % 工作區內空無一物!
```

範例

```
>> x = 0:0.1:6;
>> y = sin(x);
>> save xy_points x y % 將變數 x 和 y 儲存在 MAT 檔案
>> clear all % 將工作區的變數全部刪除
>> whos % 查詢變數的狀態
>>      % 工作區內空無一物!
>> load xy_points % 將 x 和 y 的內容重新載入至工作區
>> whos % 查詢變數是否回復原來的狀態
```

範例

```
>> x = 0:0.1:6;
>> y = sin(x);
>> save xy_points x y % 將變數 x 和 y 儲存在 MAT 檔案
>> clear all % 將工作區的變數全部刪除
>> whos % 查詢變數的狀態
>>      % 工作區內空無一物!
>> load xy_points % 將 x 和 y 的內容重新載入至工作區
>> whos % 查詢變數是否回復原來的狀態
```

Name	Size	Bytes	Class
x	1×61	488	double array
y	1×61	488	double array

Lecture 7
純量與陣列運算

兩純量間的標準算術運算

實數與複數均採用下列純量運算:

運算方法	代數的形式	MATLAB 的形式
加法	$a+b$	a + b
減法	$a-b$	a - b
乘法	$a \times b$	a * b
除法	$\frac{a}{b}$	a / b
取冪次方	a^b	a ^ b

純量運算的範例

- 括號可以視需要用來將算式中的相關項次分組。
- 從最內層的括號算起，括號中的敘述式將會被**優先計算**。
- `2^((8+2) / 5) = 2^(10/5) = 2^2 = 4`

陣列與矩陣運算

- **陣列運算**: 依據元素對元素方式執行運算。
  - 兩陣列間的列數目與行數目必須完全相同。
  - 運算後的陣列維度和原來的陣列維度相同。

陣列與矩陣運算

- **陣列運算**: 依據元素對元素方式執行運算。
  - 兩陣列間的列數目與行數目必須完全相同。
  - 運算後的陣列維度和原來的陣列維度相同。
- **矩陣運算**: 依據線性代數運算規則來計算。
  - 矩陣 A 和 B 的維度相同才能相加減。
  - 若矩陣乘法 C = A*B 要能夠正確執行，則 A 的**行數**必須等於 B 的**列數**。

$$C_{ij} = [A_{i1}, A_{i2}, \cdots, A_{in}] \begin{bmatrix} B_{1j} \\ B_{2j} \\ \vdots \\ B_{nj} \end{bmatrix} = \sum_{k=1}^{n} A_{ik} B_{kj}$$

  - 純量乘法 $(k * A)_{ij} = k * A_{ij}$，其中 k 為一純量。

重要的陣列運算 (1/2)

表 4.5.3　陣列的數學運算

指 令	說 明
A.*B	將矩陣 A 內的元素乘上矩陣 B 內相同位置的元素
A.^n	計算矩陣 A 內，個別元素的 n 次方
A.'	計算矩陣 A 的轉置（transpose）矩陣
A./B	將 A 裡面的每一個元素除以 B 裡面每一個相對應的元素
A.\B	將 B 裡面的每一個元素除以 A 裡面每一個相對應的元素

重要的陣列運算 2/2)

運算	MATLAB 形式	註解
陣列左除法	a \ b	矩陣除法由 inv(a)*b 來定義，其中 inv(a) 為 a 的反矩陣。
陣列冪次	a .^ b	a 與 b 的元素對元素冪次方法運算。a(i,j)^b(i,j)。兩陣列必須形狀相同，或其中一個為純量。

陣列運算的範例

```
>> A = [1 2 3; 4 5 6];  B = 2*ones(2,3);
```

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}, \quad B = \begin{bmatrix} 2 & 2 & 2 \\ 2 & 2 & 2 \end{bmatrix}$$

陣列運算的範例

```
>> A = [1 2 3; 4 5 6];  B = 2*ones(2,3);
```

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}, \quad B = \begin{bmatrix} 2 & 2 & 2 \\ 2 & 2 & 2 \end{bmatrix}$$

```
>> A .* B
ans =

 2   4   6
 8  10  12
```

```
>> A ./ B
ans =

 0.5000  1.0000  1.5000
 2.0000  2.5000  3.0000
```

陣列運算的範例

```
>> A = [1 2 3; 4 5 6];  B = 2*ones(2,3);
```

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}, \quad B = \begin{bmatrix} 2 & 2 & 2 \\ 2 & 2 & 2 \end{bmatrix}$$

```
>> A .* B
ans =

 2   4   6
 8  10  12
```

```
>> A ./ B
ans =

 0.5000   1.0000   1.5000
 2.0000   2.5000   3.0000
```

```
>> A .^ B
ans =

 1   4   9
16  25  36
```

>> **A + 2** % 相當於 $A_{ij} + 2$

```
ans =

 3  4  5
 6  7  8
```

基本的矩陣運算

表 4.5.1　矩陣的數學運算

矩陣的運算	說 明
$A+B$	矩陣 A 加上矩陣 B
$A-B$	矩陣 A 減去矩陣 B
$A*B$	矩陣 A 乘上矩陣 B
A^n	矩陣 A 的 n 次方，即矩陣 A 連乘 n 次，A 必須為方陣
A'	計算矩陣 A 的共軛轉置（conjugate transpose）。如果矩陣 A 的所有元素都是實數，則 A' 相當於是 A 的轉置矩陣
inv(A)	計算矩陣 A 的反矩陣（inverse）
det(A)	計算矩陣 A 的行列式（determinate）
expm(A)	計算矩陣 A 的指數（matrix exponential）
logm(A)	計算矩陣 A 的對數（matrix logarithm）
sqrtm(A)	計算矩陣 A 的平方根

矩陣轉置與共軛轉置

```
>> A = [1+i 2+2i; 3+3i 4+4i]
```

矩陣轉置與共軛轉置

```
>> A = [1+i 2+2i; 3+3i 4+4i]

A =

   1.0000 + 1.0000i   2.0000 + 2.0000i
   3.0000 + 3.0000i   4.0000 + 4.0000i
```

矩陣轉置與共軛轉置

```
>> A = [1+i 2+2i; 3+3i 4+4i]

A =

  1.0000 + 1.0000i  2.0000 + 2.0000i
  3.0000 + 3.0000i  4.0000 + 4.0000i

>> A'   % 取矩陣 A 的共軛轉置

ans =

  1.0000 - 1.0000i  3.0000 - 3.0000i
  2.0000 - 2.0000i  4.0000 - 4.0000i
```

矩陣轉置與共軛轉置

```
>> A = [1+i 2+2i; 3+3i 4+4i]

A =

  1.0000 + 1.0000i  2.0000 + 2.0000i
  3.0000 + 3.0000i  4.0000 + 4.0000i

>> A'   % 取矩陣 A 的共軛轉置

ans =

  1.0000 - 1.0000i  3.0000 - 3.0000i
  2.0000 - 2.0000i  4.0000 - 4.0000i

>> A.'   % 取矩陣 A 的轉置

ans =

  1.0000 + 1.0000i  3.0000 + 3.0000i
  2.0000 + 2.0000i  4.0000 + 4.0000i
```

若 A 為實數矩陣，則 A' 和 A.' 的結果一樣。

MATLAB 特有的矩陣除法

- **矩陣左除法** `A \ B`:
  - 左除法由 `inv(A)*B` 來定義，其中 `inv(A)` 為 `A` 的反矩陣。
  - 此運算等價於求解線性系統 (或是聯立方程組) `A * X = B`。

MATLAB 特有的矩陣除法

- **矩陣左除法** `A \ B`:
  - 左除法由 `inv(A)*B` 來定義，其中 `inv(A)` 為 `A` 的反矩陣。
  - 此運算等價於求解線性系統 (或是聯立方程組) `A * X = B`。
- **矩陣右除法** `B / A`:
  - 矩陣除法由 `B*inv(A)` 來定義。
  - 此運算等價於求解線性系統 `X * A = B`。
- 在上述除法中，我們假設 `A` 為一個可逆的 (invertible) 或是非奇異的 (nonsingular) 方陣。

矩陣左除法的範例

試用M ATLAB 求解線性系統 $\begin{bmatrix} 1 & 0 & 2 \\ 0 & 4 & 3 \\ 3 & 6 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 9 \\ 1 \\ 0 \end{bmatrix}$ 。

```
>> A = [1 0 2; 0 4 3; 3 6 0];  B = [9; 1; 0];
```

矩陣左除法的範例

試用M ATLAB 求解線性系統 $\begin{bmatrix} 1 & 0 & 2 \\ 0 & 4 & 3 \\ 3 & 6 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 9 \\ 1 \\ 0 \end{bmatrix}$ 。

```
>> A = [1 0 2; 0 4 3; 3 6 0];  B = [9; 1; 0];
 >> X = A\B          >> inv(A)*B

 X =                 ans =

   3.5714              3.5714
  -1.7857             -1.7857
   2.7143              2.7143
```

矩陣左除法的範例

試用M ATLAB 求解線性系統 $\begin{bmatrix} 1 & 0 & 2 \\ 0 & 4 & 3 \\ 3 & 6 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 9 \\ 1 \\ 0 \end{bmatrix}$ 。

```
>> A = [1 0 2; 0 4 3; 3 6 0];  B = [9; 1; 0];
 >> X = A\B          >> inv(A)*B

 X =                 ans =

   3.5714              3.5714
  -1.7857             -1.7857
   2.7143              2.7143

>> A*X   % 查看計算解 X 的正確性

ans =
 9.0000
 1.0000
    0
```

Lecture 8
運算的順序

MATLAB 算術運算的順序:

先後順序	運算
1	從最內的括號向外依序求解，直到求出所有括號中內容的值。
2	從左到右求出所有指數的值。
3	從左到右計算所有的乘法與除法。
4	從左到右計算所有的加法與減法。

Remark

使用必要的括弧，將使你的運算式更為清楚易懂。例如: 運算式 $\frac{n}{n+1}$ 應輸入為 `n / (n + 1)`，而非 `n / n + 1`。

Lecture 9
內建的 MATLAB 函式

數學函數與 MATLAB 函式 (1/2)

- 在數學定義裡，**函數（function）**是含有一個或一個以上變數的敘述式，而且這些變數會對應到**單一的結果**。

數學函數與 MATLAB 函式 (1/2)

- 在數學定義裡，**函數（function）**是含有一個或一個以上變數的敘述式，而且這些變數會對應到**單一的結果**。
- 不像數學函數的定義，MATLAB 函式能將一個以上的結果，傳回所呼叫的程式中。

數學函數與 MATLAB 函式 (1/2)

- 在數學定義裡，**函數（function）**是含有一個或一個以上變數的敘述式，而且這些變數會對應到**單一的結果**。
- 不像數學函數的定義，MATLAB 函式能將一個以上的結果，傳回所呼叫的程式中。

```
>> maxval = max([1 -5 6 -3])
```

數學函數與 MATLAB 函式 (1/2)

- 在數學定義裡，**函數（function）**是含有一個或一個以上變數的敘述式，而且這些變數會對應到**單一的結果**。
- 不像數學函數的定義，MATLAB 函式能將一個以上的結果，傳回所呼叫的程式中。

```
>> maxval = max([1 -5 6 -3])

maxval =

  6

>> [maxval, index] = max([1 -5 6 -3]);
```

數學函數與 MATLAB 函式 (1/2)

- 在數學定義裡，**函數（function）**是含有一個或一個以上變數的敘述式，而且這些變數會對應到**單一的結果**。
- 不像數學函數的定義，MATLAB 函式能將一個以上的結果，傳回所呼叫的程式中。

```
>> maxval = max([1 -5 6 -3])

maxval =

  6

>> [maxval, index] = max([1 -5 6 -3]);
>> [maxval, index]

ans =

  6     3
```

數學函數與 MATLAB 函式 (2/2)

- MATLAB 函式的輸入引數 (input arguments) 或是輸出引數 (output arguments)，其元素可以包含**複數**。

數學函數與 MATLAB 函式 (2/2)

- MATLAB 函式的輸入引數 (input arguments) 或是輸出引數 (output arguments)，其元素可以包含**複數**。
- 一個純虛數或是複數的資料型態，MATLAB 是以具有 **16 bytes** 記憶體配置的變數儲存之。

```
>> z = sqrt(-2) % 兩個 double 型態的記憶體配置
```

數學函數與 MATLAB 函式 (2/2)

- MATLAB 函式的輸入引數 (input arguments) 或是輸出引數 (output arguments)，其元素可以包含**複數**。
- 一個純虛數或是複數的資料型態，MATLAB 是以具有 **16 bytes** 記憶體配置的變數儲存之。

```
>> z = sqrt(-2) % 兩個 double 型態的記憶體配置

z =

  0.0000 + 1.4142i
```

常用的 MATLAB 函式 (1/4)

函式	說明
	數學函式（Mathematical functions）
abs(x)	計算 $\lvert x \rvert$。
acos(x)	計算 $\cos^{-1}x$（結果以弧度表示）。
acosd(x)	計算 $\cos^{-1}x$（結果以度表示）。
angle(x)	傳回複數 x 的相位角（以弧度表示）。
asin(x)	計算 $\sin^{-1}x$（結果以弧度表示）。
asind(x)	計算 $\sin^{-1}x$（結果以度表示）。
atan(x)	計算 $\tan^{-1}x$（結果以弧度表示）。
atand(x)	計算 $\tan^{-1}x$（結果以度表示）。
atan2(y, x)	在圓的四個象限內計算 $\theta = \tan^{-1}\frac{y}{x}$（結果介於 $-\pi$ 與 π 之間，以弧度表示）。
atan2d(y,x)	在圓的四個象限內計算 $\theta = \tan^{-1}\frac{y}{x}$（結果介於 $-180°$ 與 $180°$ 之間，以度表示）。

常用的 MATLAB 函式 (2/4)

cos(x)	計算 $\cos x$，x 以弧度表示。
cosd(x)	計算 $\cos x$，x 以度表示。
exp(x)	計算 e^x。
log(x)	計算自然對數 $\log_e x$。
[value, index] = max(x)	傳回向量 x 的最大值，可選擇傳回最大值的位置。
[value, index] = min(x)	傳回向量 x 的最小值，可選擇傳回最小值的位置。
mod(x, y)	傳回餘數或稱為模數（modulo）函式。
sin(x)	計算 $\sin x$，x 以弧度表示。
sind(x)	計算 $\sin x$，x 以度表示。
sqrt(x)	計算 x 的平方根。
tan(x)	計算 $\tan x$，x 以弧度表示。
tand(x)	計算 $\tan x$，x 以度表示。

常用的 MATLAB 函式 (3/4)

捨位函式（Rounding functions）	
ceil(x)	向正無限大的方向，對 x 取最近的整數： ceil(3.1)= 4，而 ceil(-3.1) = -3。
fix(x)	向 0 的方向，對 x 取最接近的整數： fix(3.1)= 3，而 fix(-3.1)= -3。
floor(x)	向負無限大的方向，對 x 取最近的整數： floor(3.1)= 3，而 floor(-3.1) = -4。
round(x)	對 x 取四捨五入的整數值。

常用的 MATLAB 函式 (4/4)

字串轉換函式（String conversion functions）	
char(x)	將一個數字矩陣，轉換成一個字元字串。對 ASCII 字元來說，這矩陣的數字需要 ≤ 127。
double(x)	將一個字元字串，轉換成一個數字矩陣。
int2str(x)	將 x 轉換成一個整數字元字串。
num2str(x)	將 x 轉換成一個字元字串。
str2num(s)	將字元字串 s，轉換成一個數字陣列。

Lecture 10
繪圖功能簡介

MATLAB 的基本繪圖指令

- MATLAB與輸出裝置無關的強大繪圖功能，可將任何資料瞬間繪製成圖形。

MATLAB 的基本繪圖指令

- MATLAB與輸出裝置無關的強大繪圖功能，可將任何資料瞬間繪製成圖形。
- 如果想要畫出一組數據資料的**二維圖形**，只要產生兩個各含 x、y 值的向量，並使用plot函式即可。

以 plot 指令苗繪函數圖形

% 描繪函數 $y = f(x) = x^2 - 10x + 15$ 在區間 $[0, 10]$ 上的圖形。

```
x = 0:0.1:10;
y = x.^2 - 10.*x + 15;
plot(x,y);
```

函數 $y = f(x)$ 的二維圖形

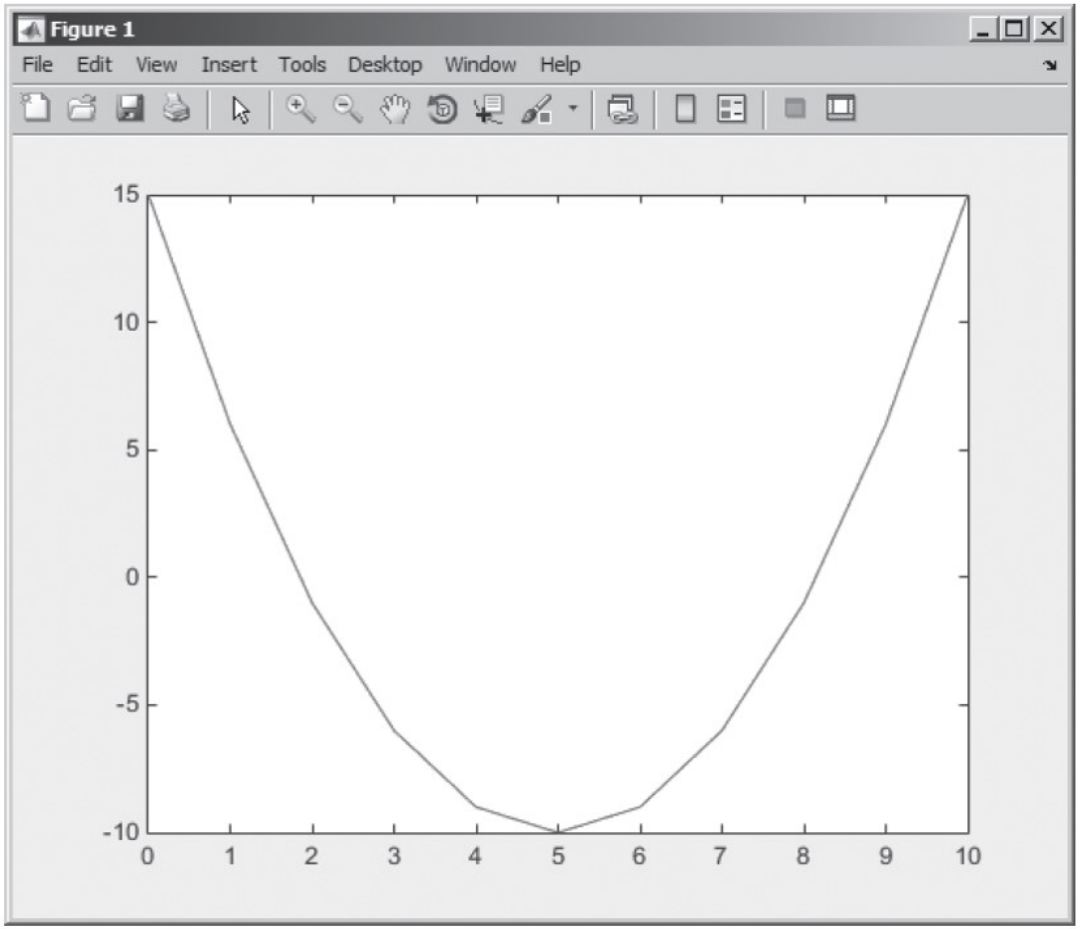


Figure 1
File Edit View Insert Tools Desktop Window Help
15
10
5
0
-5
-10
0 1 2 3 4 5 6 7 8 9 10

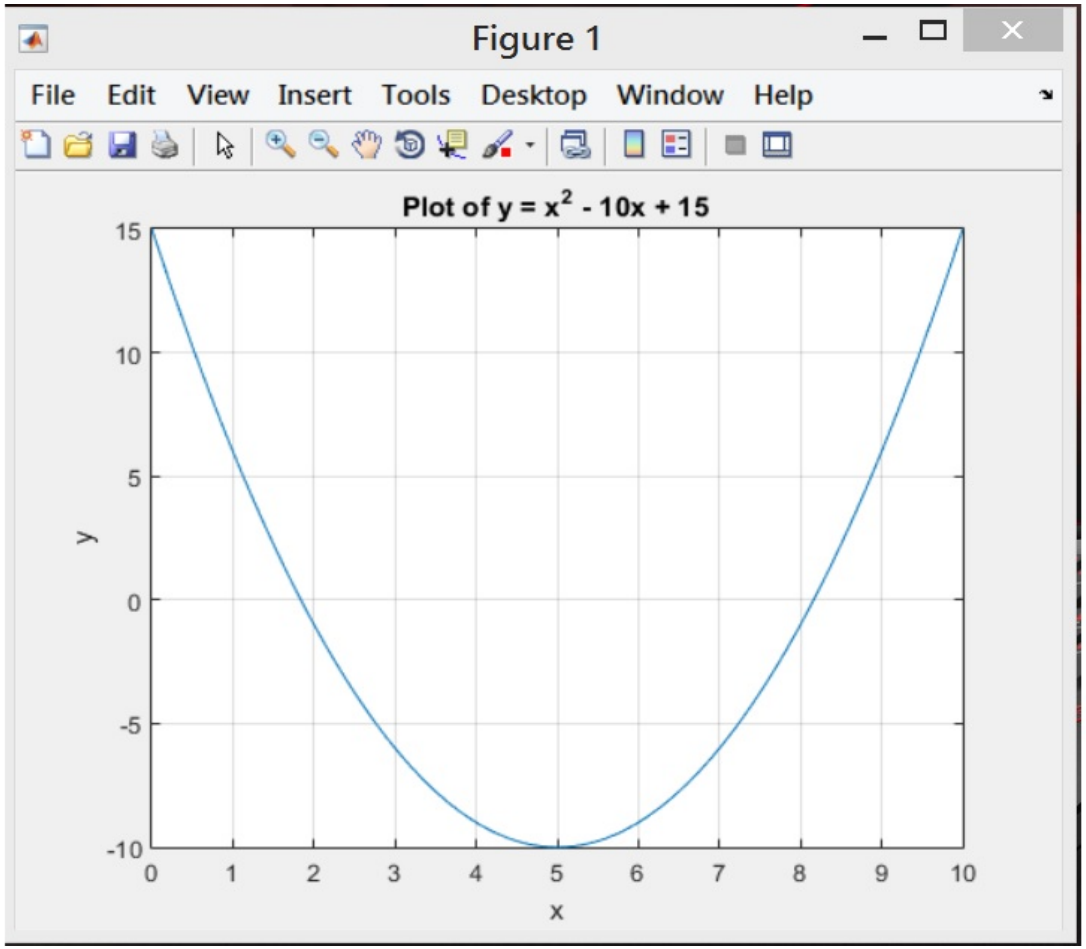
增加圖形標題與 x-y 軸說明 (1/2)

裝飾函數圖形 (承上例)

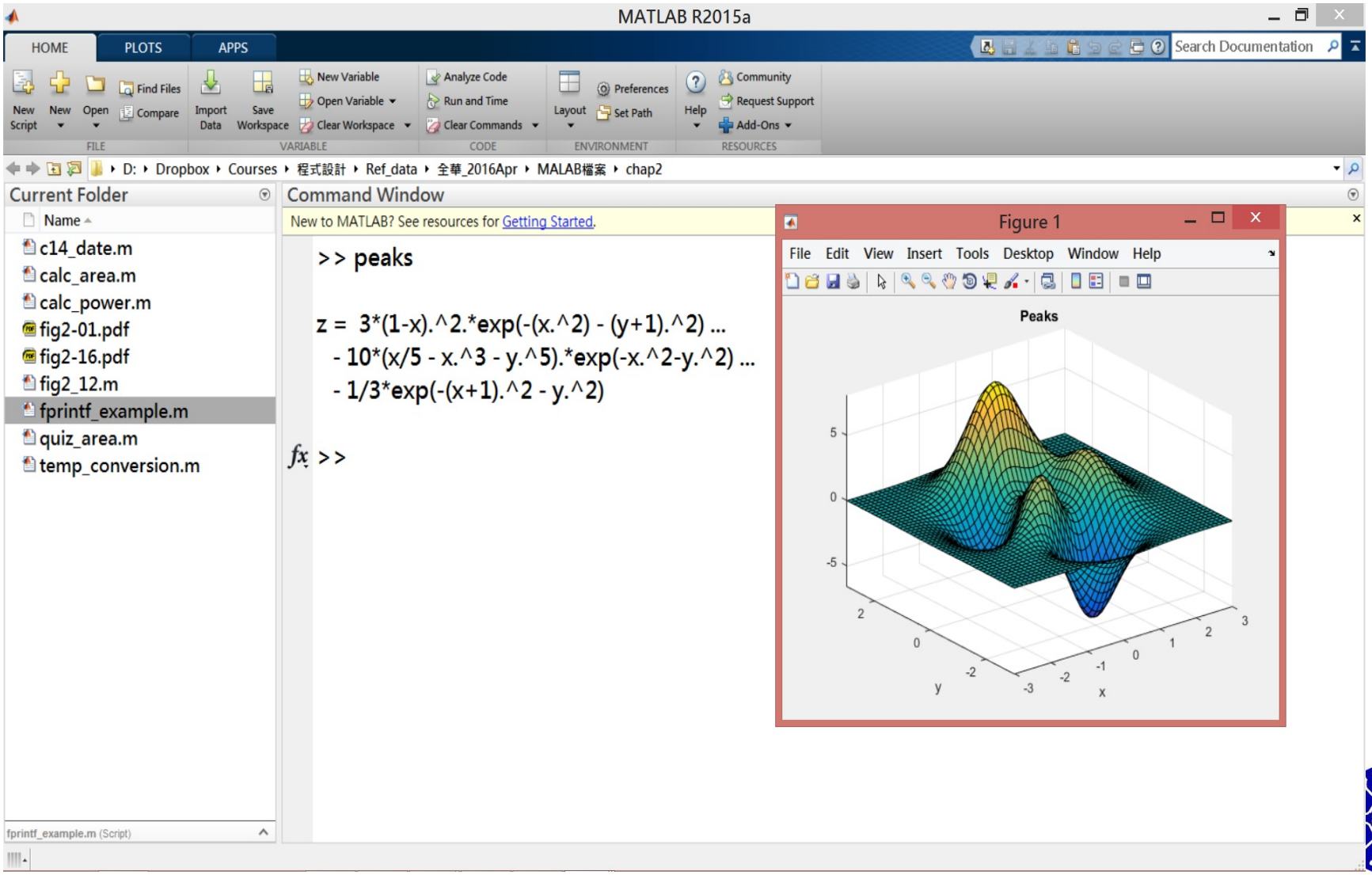
% 描繪函數 $y = f(x) = x^2 - 10x + 15$ 在區間 $[0, 10]$ 上的圖形。

```
x = 0:0.1:10;
y = x.^2 - 10.*x + 15;
plot(x,y);
title('Plot of y = x^2 - 10x + 15');
xlabel('x');
ylabel('y');
grid on;  % 預設值是 grid off。
```

增加圖形標題與 x-y 軸說明 (2/2)



函數 $z = f(x, y)$ 的三維圖形



圖形化影像輸出 (1/2)

- 指令 `print` 將繪圖結果，藉由指定**選項及檔名**，以圖形化影像方式來儲存。
- 指令 `print` 的形式：

```
print  <option>  <filename>
```

圖形化影像輸出 (1/2)

- 指令 `print` 將繪圖結果，藉由指定**選項及檔名**，以圖形化影像方式來儲存。
- 指令 `print` 的形式：

```
print  <option>  <filename>
```

- `print -dtiff my_image.tif`: 將目前指定的圖形，產生 **TIFF 格式**的影像檔，並以`my_image.tif`的檔名儲存。

圖形化影像輸出 (1/2)

- 指令 print 將繪圖結果，藉由指定**選項及檔名**，以圖形化影像方式來儲存。
- 指令 print 的形式：

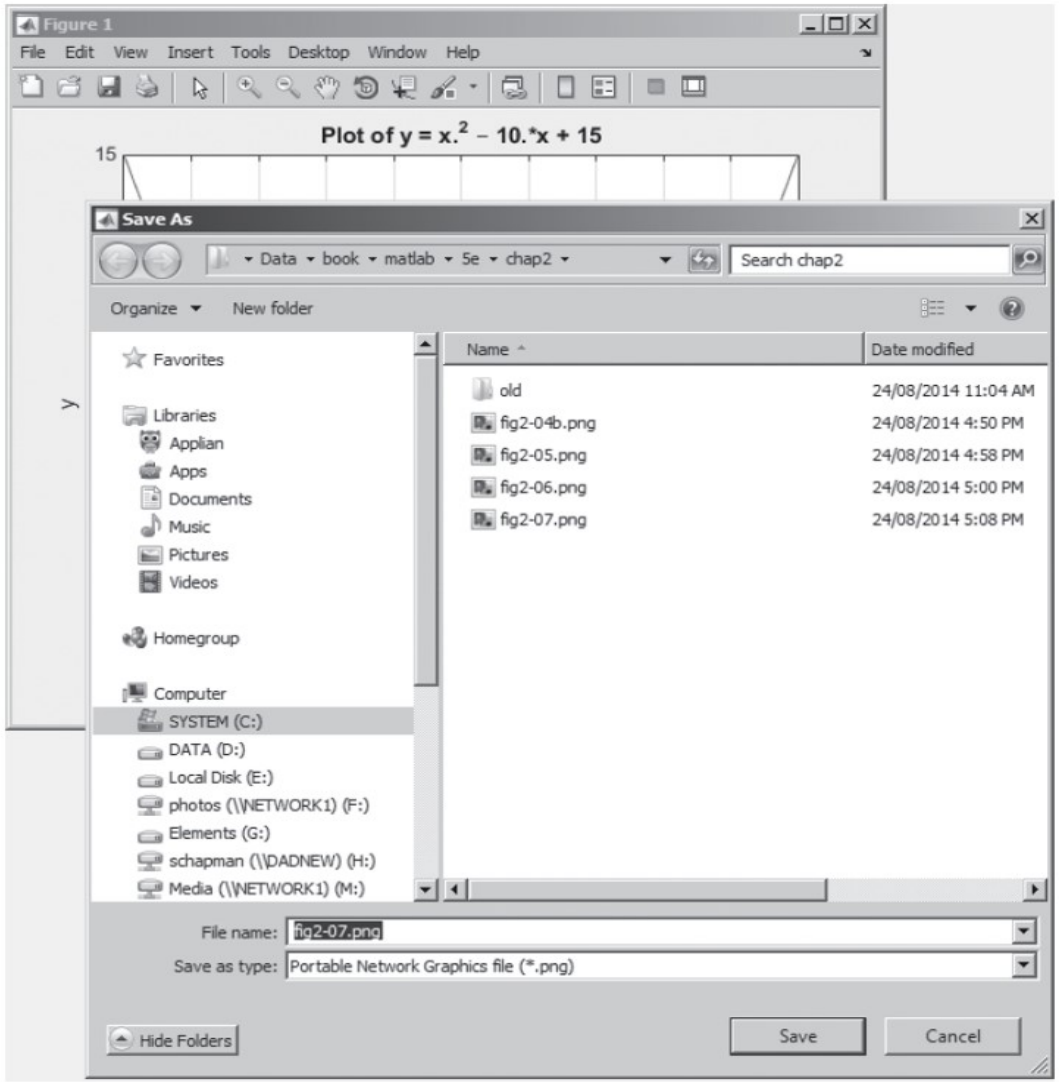
```
print <option> <filename>
```

- print -dtiff my_image.tif: 將目前指定的圖形，產生 **TIFF 格式**的影像檔，並以my_image.tif的檔名儲存。
- 其他檔案格式的選項 <option>:

選項	功能敘述
-deps	產生一個灰階的 EPS 圖形
-depsc	產生一個彩色的 EPS 圖形
-djpeg	產生一個 JPEG 圖形
-dpng	產生一個 PNG 圖形
-dtiff	產生一個壓縮的 TIFF 圖形

圖形化影像輸出 (2/2)

- 在圖形視窗中，使用 **“File/Save As”** 功能表輸出圖檔:



多重線條繪圖

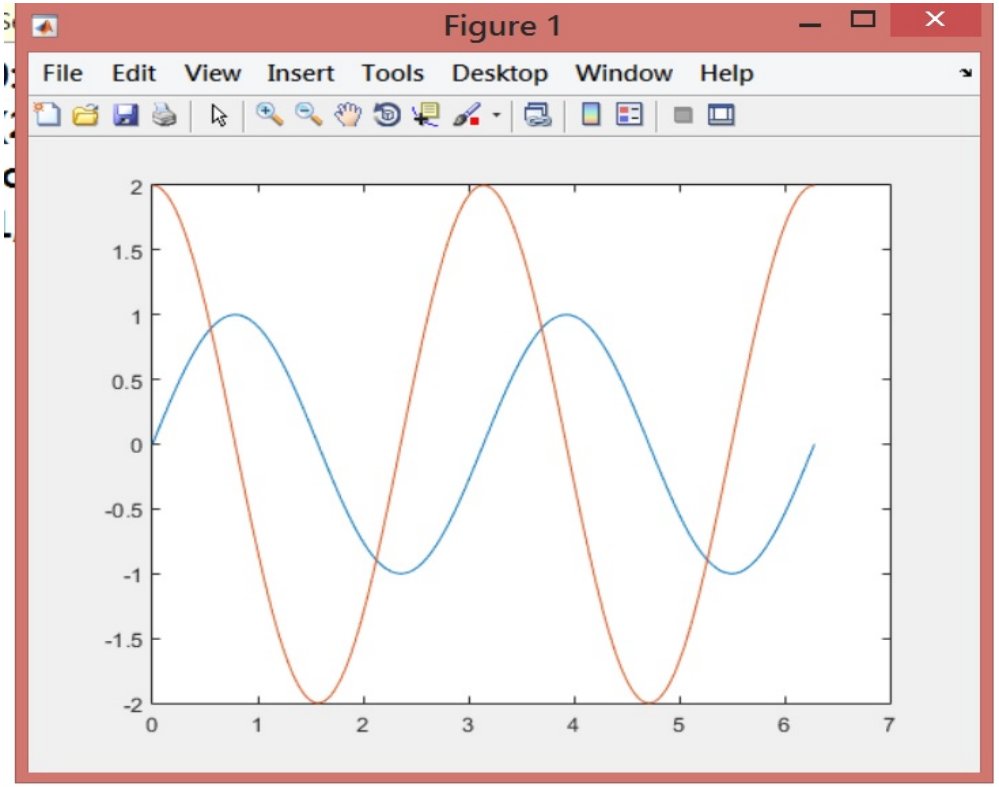
在相同的軸線上的兩個函數圖形

```
x = 0:pi/100:2*pi;
y1 = sin(2*x); % y_1 = f(x) = sin(2x)
y2 = 2*cos(2*x); % y_2 = f'(x) = 2 cos(2x)
plot(x,y1,x,y2);
```

多重線條繪圖

在相同的軸線上的兩個函數圖形

```
x = 0:pi/100:2*pi;
y1 = sin(2*x); % y1 = f(x) = sin(2x)
y2 = 2*cos(2*x); % y2 = f'(x) = 2cos(2x)
plot(x,y1,x,y2);
```



線條顏色、資料標記形式及線條樣式 (1/2)

範例

% 描繪函數 $y = f(x) = x^2 - 10x + 15$ 在區間 $[0, 10]$ 上的圖形。

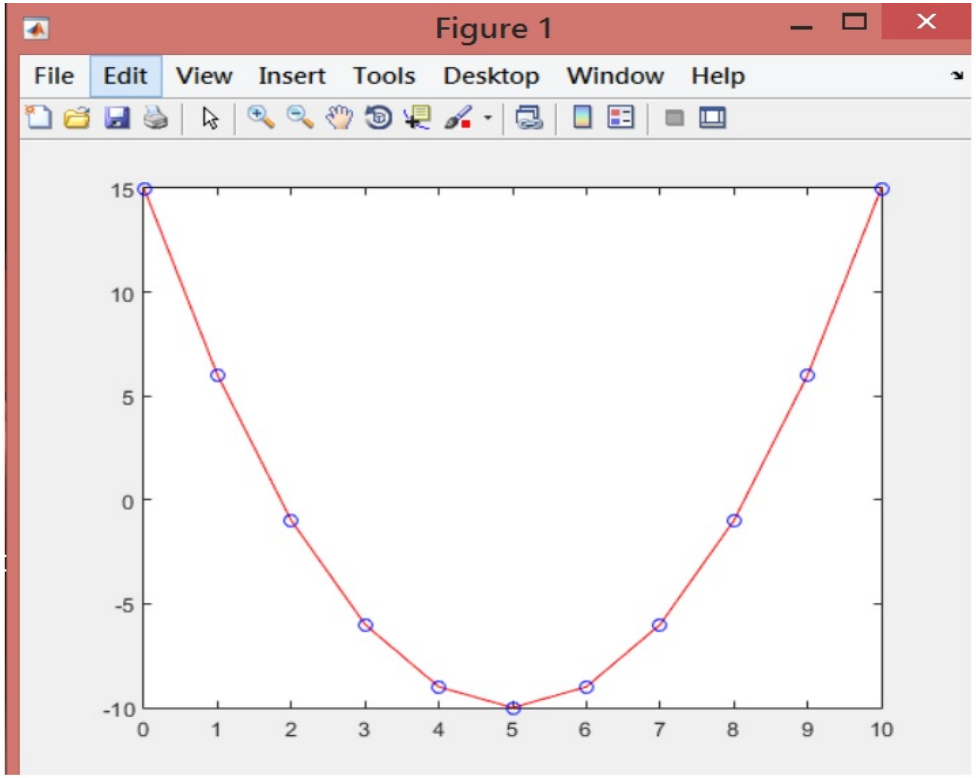
```
x = 0:10;
y = x.^2 - 10.*x + 15;
plot(x,y,'r-',x,y,'bo');
```

線條顏色、資料標記形式及線條樣式 (1/2)

範例

% 描繪函數 $y = f(x) = x^2 - 10x + 15$ 在區間 $[0, 10]$ 上的圖形。

```
x = 0:10;
y = x.^2 - 10.*x + 15;
plot(x,y,'r-',x,y,'bo');
```



線條顏色、資料標記形式及線條樣式 (2/2)

顏色		標記形式		線條樣式	
y	黃色（yellow）	.	點	-	實線
m	紫紅色（magenta）	o	圓	:	點線
c	青色（cyan）	x	x- 記號	-.	點折線
r	紅色（red）	+	加號	--	虛線
g	綠色（green）	*	星號	<none>	不畫線
b	藍色（blue）	s	方塊		
w	白色（white）	d	鑽石形		
k	黑色（black）	v	三角形（向下）		
		^	三角形（向上）		
		<	三角形（向左）		
		>	三角形（向右）		
		p	五角星號		
		h	六角星形		
		<none>	不畫記號		

圖形說明 (legends)

圖形說明指令 `legend` 的基本形式為

```
legend('string1','string2',…,'Location',pos);
```

其中 `string1`、`string2` 等，是所畫線條的相關標示，而pos是指定圖形說明在圖形視窗位置的**一組字串** 。

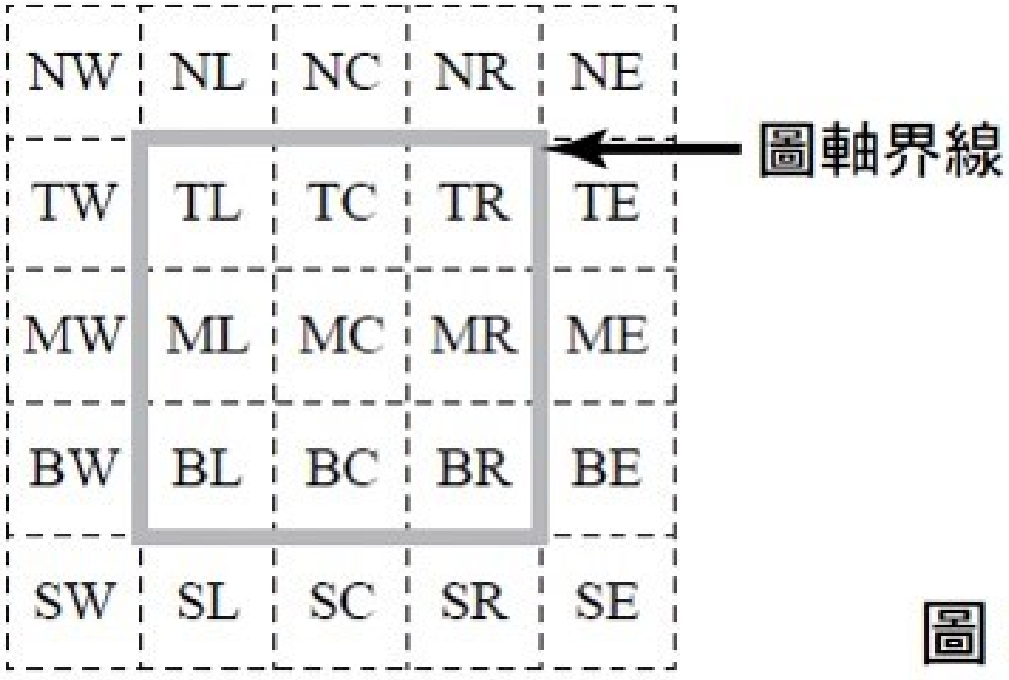


圖 2

指令 legend 的範例

函數 $f(x)$ 及其導函數的圖形說明

```
x = 0:pi/100:2*pi;
y1 = sin(2*x);
y2 = 2*cos(2*x);
plot(x,y1,'k-',x,y2,'b--');
title('Plot of f(x) = sin(2x) and its derivative');
xlable('x');
ylable('y');
legend('f(x)','d/dx f(x)','Location','NW');
grid on;
```

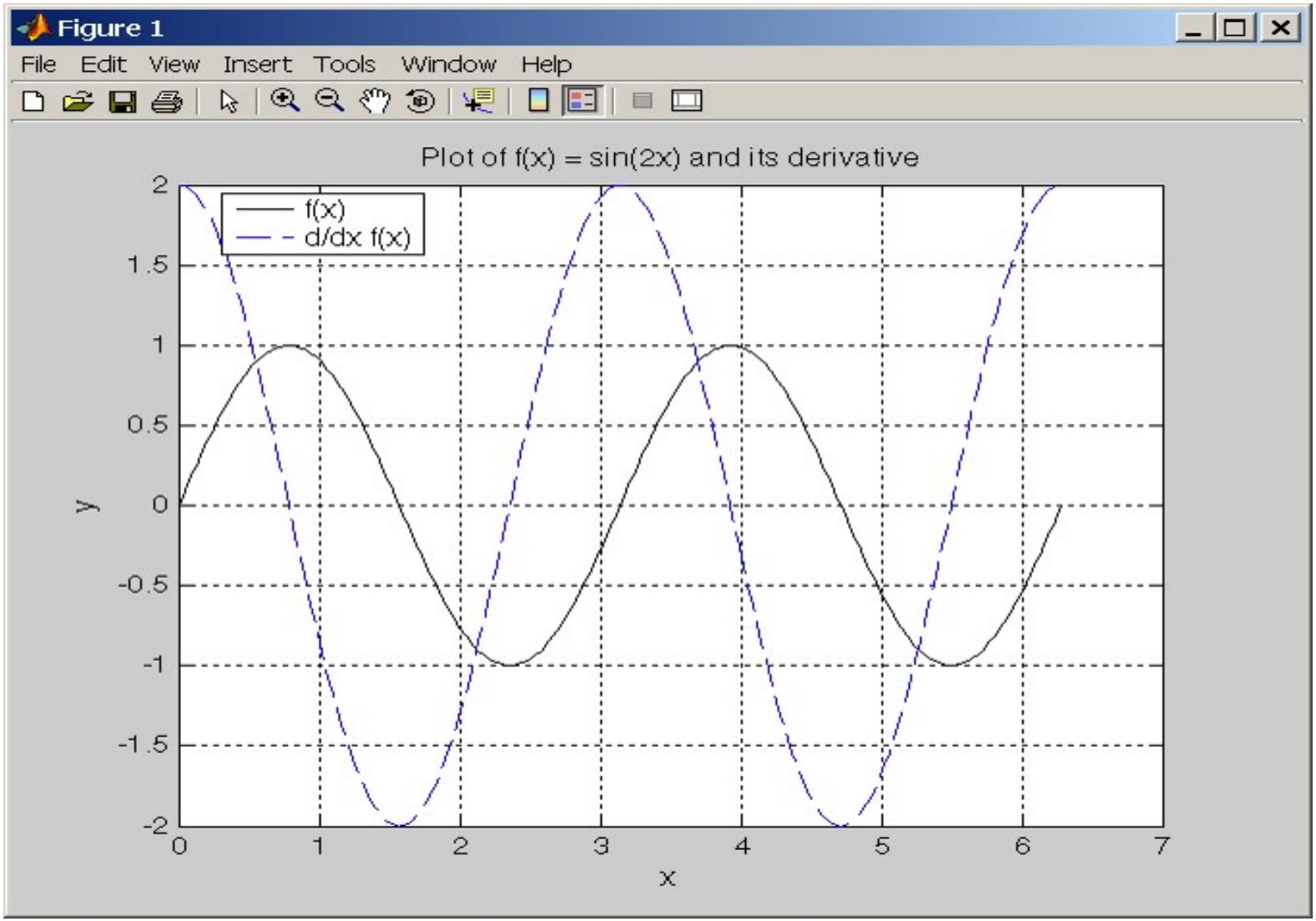


Figure 1
File Edit View Insert Tools Window Help
Plot of f(x) = sin(2x) and its derivative
f(x)
d/dx f(x)
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
y
0
1
2
3
4
5
6
7
x

對數座標的圖形

除了線性座標，也可把資料畫在對數座標上。

- `plot` 函式將資料畫在線性軸上。
- `semilogx` 函式將 x 資料畫在對數軸上，y 資料畫在線性軸上。
- `semilogy` 函式將 x 資料畫在線性軸上，y 資料畫在對數軸上。
- `loglog` 函式將 x y 資料都畫在對數軸上。

對數座標的範例

```
x = 0:0.1:10; y = x.^2 - 10.*x + 26;
```

對數座標的範例

```
x = 0:0.1:10; y = x.^2 - 10.*x + 26;
subplot(2,2,1);
plot(x,y); title('Linear Plot');
xlabel('x'); ylabel('y'); grid on;
```

對數座標的範例

```
x = 0:0.1:10; y = x.^2 - 10.*x + 26;
subplot(2,2,1);
plot(x,y); title('Linear Plot');
xlabel('x'); ylabel('y'); grid on;
subplot(2,2,2);
semilogx(x,y); title('Semilog x Plot');
xlabel('x'); ylabel('y'); grid on;
```

對數座標的範例

```
x = 0:0.1:10; y = x.^2 - 10.*x + 26;
subplot(2,2,1);
plot(x,y); title('Linear Plot');
xlabel('x'); ylabel('y'); grid on;
subplot(2,2,2);
semilogx(x,y); title('Semilog x Plot');
xlabel('x'); ylabel('y'); grid on;
subplot(2,2,3);
semilogy(x,y); title('Semilog y Plot');
xlabel('x'); ylabel('y'); grid on;
```

對數座標的範例

```
x = 0:0.1:10; y = x.^2 - 10.*x + 26;
subplot(2,2,1);
plot(x,y); title('Linear Plot');
xlabel('x'); ylabel('y'); grid on;
subplot(2,2,2);
semilogx(x,y); title('Semilog x Plot');
xlabel('x'); ylabel('y'); grid on;
subplot(2,2,3);
semilogy(x,y); title('Semilog y Plot');
xlabel('x'); ylabel('y'); grid on;
subplot(2,2,4);
loglog(x,y); title('Loglog Plot');
xlabel('x'); ylabel('y'); grid on;
```

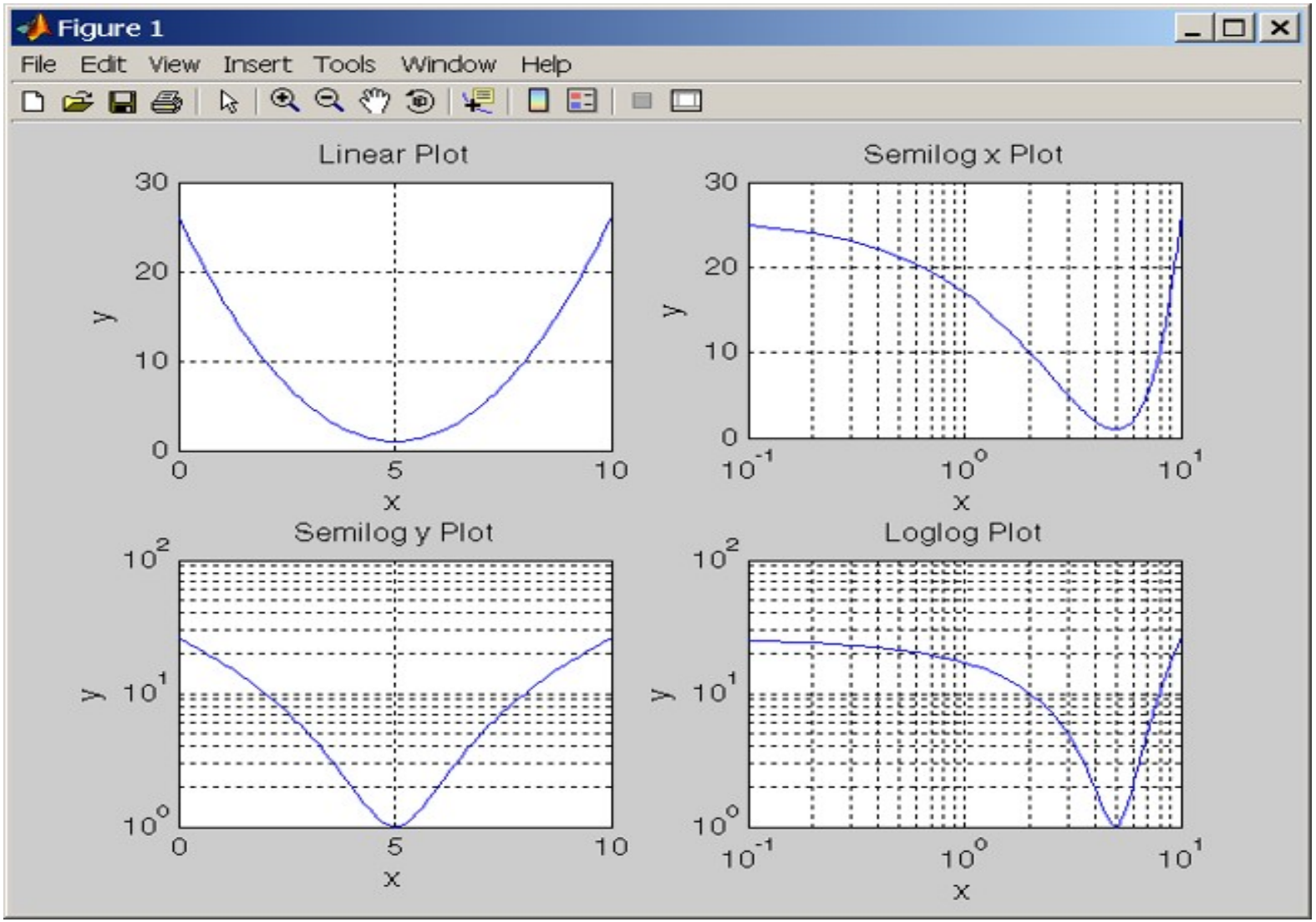


Figure 1
File Edit View Insert Tools Window Help
Linear Plot
Semilog x Plot
Semilog y Plot
Loglog Plot
x
y

Lecture 11
MATLAB 程式除錯

MATLAB 的程式有三種類型的錯誤。

1. **語法錯誤（syntax error）**：
   - 在 MATLAB 敘述式裡的錯誤，如拼字錯誤或是標點錯誤。
   - 當第一次執行 M 檔案時，MATLAB 編譯器將會檢測到這類錯誤。

MATLAB 的程式有三種類型的錯誤。

1. **語法錯誤 (syntax error)**:
   - 在 MATLAB 敘述式裡的錯誤，如拼字錯誤或是標點錯誤。
   - 當第一次執行 M 檔案時，MATLAB 編譯器將會檢測到這類錯誤。
2. **執行時的錯誤 (run-time error)**:
   - 當程式嘗試執行一個不合法的數學運算 (如除以 0)。
   - 這些錯誤將使程式回應 `Inf` 或 `NaN`，造成無效的計算結果。

MATLAB 的程式有三種類型的錯誤。

1. **語法錯誤（syntax error）**：
   - 在 MATLAB 敘述式裡的錯誤，如拼字錯誤或是標點錯誤。
   - 當第一次執行 M 檔案時，MATLAB 編譯器將會檢測到這類錯誤。
2. **執行時的錯誤（run-time error）**：
   - 當程式嘗試執行一個不合法的數學運算 (如除以 0)。
   - 這些錯誤將使程式回應 `Inf` 或 `NaN`，造成無效的計算結果。
3. **邏輯錯誤（logical error）**：
   - 程式已編譯完成，而且已執行完畢，卻得到錯誤的答案。

程式除錯的小叮嚀

1. 若宣告式過於冗長，請將原宣告式改成幾個較短的宣告。
2. 檢查每個宣告式中的**括弧位置**。
3. 建議在主檔案 (main file) 的第一行加入`clc,clear all;`等指令，並確認正確地初始化所有的變數。

程式除錯的小叮嚀

1. 若宣告式過於冗長，請將原宣告式改成幾個較短的宣告。
2. 檢查每個宣告式中的**括弧位置**。
3. 建議在主檔案 (main file) 的第一行加入`clc,clear all;`等指令，並確認正確地初始化所有的變數。
4. 確認在每個函式裡都使用了正確的單位。
5. 在程式中增加一些輸出宣告或將宣告尾端 “;” 移除，作為計算過程中的檢驗點。例如，可將宣告式 `var = expression;`改為 `var = expression,pause`。
6. 請他人幫忙檢查程式碼，可減少除錯盲點。

Thank you for your attention!