

Chapter 6

基本的使用者定義函式

Hung-Yuan Fan (范洪源)

Department of Mathematics,
National Taiwan Normal University, Taiwan

Spring 2019



- 6.1 MATLAB 函式介紹
- 6.2 MATLAB 的變數傳遞方式：按值傳遞
- 6.3 選擇性引數
- 6.4 使用共用記憶體分享資料
- 6.5 函式呼叫間的資料保存
- 6.6 函式握把
- 6.7 匿名函式
- 6.8 MATLAB 內建函式：排序與亂數函式



Section 6.1

MATLAB 函式介紹



儲存 MATLAB 程式碼的檔案稱為 M 檔案 (副檔名是.m) , 包含

① 程序檔案或稱底稿檔案 (script files)

- 它是由一系列宣告式所組成的檔案。
- 執行結果如同把所有的指令直接鍵入指令視窗一樣。
- 任何由程序檔所產生的變數 , 都會繼續存留在工作區內而互相影響。

② MATLAB 函式 (MATLAB functions)

- 一種特殊類型的 M 檔案 , 可在自己專屬的工作區內執行。
- 藉由輸入引數清單 (input argument list) 接受輸入資料。
- 經由輸出引數清單 (output argument list) 傳回計算結果給呼叫的程式。
- 在函式內使用的變數是區域變數 (local variable) , 因此即使工作區內已使用相同名稱的變數 , 它們彼此之間不會混淆。



- 完整的函式包括函式定義列、H1 列 (唸成 H-one line)、函式說明文字區，以及函式的主體四個部分。

```
function [out1,out2,...] = filename(in1,in2,...)
% H1 comment line
% Other comment lines
...
(Executable code)
...
(return) % 可省略此宣告式
end % 有時亦可省略此宣告式
```

- `function` 標示了函式的起點，它指明了函式名稱，以及輸入與輸出引數清單。
- 儲存函式的檔名必須是 `filename.m`，與函式名稱的英文大小寫一致！



- 在函式宣告 `function` 之後的第一列註解，稱為 **H1 註解行 (H1 comment line)**。
- 它必須只包含一行註解文字，用以表明整個函式的目的。
- 此行文字的特別意義，是讓 MATLAB 藉由 `lookfor` 指令搜尋並顯示此函式用途。
 - 語法: `lookfor keyword`
 - MATLAB 會搜尋函式名稱或是 H1 列中是否包含該關鍵字 `keyword`，並將所有搜尋結果顯示於指令視窗。



H1 列和註解說明文字 (1/2)

範例: 函式 `vec_plus.m`

```
function total = vec_plus(x,y)
% vec_plus Sum of two numbers or vectors.
% vec_plus(X,Y) computes X+Y and returns the result,
% where X and Y can be scalars or vectors.
% function's body starts here
total = x + y;
```

```
>> vec_plus(2,ones(2,1))
```

```
ans =
```

```
3
```

```
3
```



```
>> help vec_plus
```

`vec_plus` **Sum** of two numbers or vectors.

`vec_plus(X,Y)` computes $X+Y$ and returns the result, where X and Y can be scalars or vectors.

```
>> lookfor sum
```

[vec_plus](#)

[reciproc_sum](#)

[StatisticsByGroupMapReduceExample](#)

[summer](#)

[uiresume](#)

⋮

- **Sum of two numbers or vectors.**
- Function-valued tensor for $1/(x_{i_1} + \dots + x_{i_d})$
- Compute Summary Statistics by Group Using MapReduce
- Shades of green and yellow colormap
- Resume execution of blocked MATLAB code.



函式的引數與傳回值

表 8.2.2 函數定義列的幾種範例

函數定義列的格式	說明
<code>function [x,y]=myfun(a)</code>	myfun 有一個輸入引數 a ，有兩個輸出引數 x 與 y
<code>function [x]=myfun(a)</code> <code>function x=myfun(a)</code>	myfun 有一個輸入引數 a ，有一個輸出引數 x
<code>function [x,y]=myfun()</code> <code>function [x,y]=myfun</code>	myfun 沒有輸入引數，但有兩個輸出引數 x 與 y 。在沒有輸入引數的情況下，函數名稱後面的括號可有可無
<code>function []=myfun(a)</code> <code>function myfun(a)</code>	myfun 沒有輸出引數，但有一個輸入引數 a 。當函數沒有輸出引數時，方括號與等號可以省略不寫



多個輸出引數的函式

範例: 函式 `vec_minmax.m`

```
function [mn, mx] = vec_minmax(v)
% vec_minmax 找出向量元素的最小值與最大值。
% 函式 vec_minmax(v) 計算向量 v 所有元素的最小值 mn
% 和最大值 mx，並且傳回這兩個輸出引數。
```

```
mn = min(v);
```

```
mx = max(v);
```

```
>> [min_val, max_val] = vec_minmax([4, -5, 7, 1])
```

```
min_val =
```

```
-5
```

```
max_val =
```

```
7
```



```
>> help vec_minmax
```

`vec_minmax` 找出向量元素的最小值與最大值。

函式 `vec_minmax(v)` 計算向量 `v` 所有元素的最小值 `mn` 和最大值 `mx`，並且傳回這兩個輸出引數。

```
>> lookfor 最小值
```

```
vec\_minmax
```

```
>>
```

- 找出向量元素的**最小值**與**最大值**。



不需要括號的函式呼叫方式

- 函式如果**沒有輸出引數**，就不需要將輸入的引數括起來。
- 例如，如果 `my_func(a, b)` 沒有輸出引數，可用下面兩種方法呼叫：

`my_func(a, b);` % 需要括號的呼叫方式

`my_func a b;` % 不需括號的呼叫: 將 a、b 視為字串

範例: 函式 `axis` 的使用方式

```
axis([-4, 4, 0, 20]); % 輸入引數是 double 型態的向量
```

```
axis on; % 效果和 axis('on') 一樣
```

```
axis off; % 效果和 axis('off') 一樣
```



沒有輸出引數的範例 (檔名: no_outarg.m)

```
function no_outarg(x, y)
fprintf(' 輸入引數分別是%s 和 %s \n',x, y);
```

```
>> no_outarg 2 -10
輸入引數分別是 2 和 -10
```

```
>> no_outarg(2, -10)
輸入引數分別是 和
```

```
>> no_outarg('2', '-10')
輸入引數分別是 2 和 -10
```



Section 6.2

MATLAB 的變數傳遞方式：按值傳遞



按值傳遞與按址傳遞

- MATLAB 程式使用**按值傳遞** (**pass-by-value**) 的方式，進行程式與函式間的溝通。
- C 語言是採取**按址傳遞** (**pass-by-address**) 的方式，進行程式與函式間的溝通。
- 當程式呼叫函式時，MATLAB 便複製**實際引數**，並傳遞這些**實際引數的備份**給函式使用。
- 即使函式更改了輸入引數值，也不會影響到呼叫程式的原始資料。
- 這種特點可防止在函式中因產生某個錯誤，而無意間修改到呼叫程式的原始變數值。



函式 sample.m

```
function out = sample(a,b)
fprintf('In sample: a = %f, b = %f \n',a,b);
a = b(1) + 2*a;
b = a .* b;
out = a + b(1);
fprintf('In sample: a = %f, b = %f \n',a,b);
```

呼叫上述函式的主程式 test_sample.m

```
a = 2; b = [6 4];
fprintf('Before sample: a = %f, b = %f \n',a,b);
out = sample(a,b);
fprintf('After sample: a = %f, b = %f \n',a,b);
fprintf('After sample: out = %f \n',out);
```



主程式的執行結果

```
>> test_sample
```

```
Before sample: a = 2.000000, b = 6.000000 4.000000
```

```
In sample: a = 2.000000, b = 6.000000 4.000000
```

```
In sample: a = 10.000000, b = 60.000000 40.000000
```

```
After sample: a = 2.000000, b = 6.000000 4.000000
```

```
After sample: out = 70.000000
```

```
>>
```



Section 6.3

選擇性引數



- 許多 MATLAB 函式支援選擇性的輸入引數及輸出引數，例如: `plot`、`max` 等函式。
 - `x_max = max(x);`
 - `[x_max,ii_max] = max(x);`
- MATLAB 有關選擇性引數資訊的函式
 - `nargin`: 傳回用來呼叫函式的實際輸入引數數目，是 number of argument input 的縮寫。
 - `nargout`: 傳回用來呼叫函式的實際輸出引數數目，是 number of argument output 的縮寫。
 - `nargchk`: 假如用來呼叫函式的引數太少或是太多，將傳回一個錯誤訊息。
 - `error`: 顯示錯誤的訊息，並放棄執行產生錯誤的函式。用在當引數產生的錯誤是嚴重的 (`fatal`) 情況。
 - `warning`: 顯示警告的訊息，並繼續執行函式。用在當引數的錯誤並不嚴重，而函式可以繼續執行的情況。
 - `inputname`: 傳回對應輸入引數清單特定次序的實際變數名稱。



● nargchk 語法

- `msg = nargchk(min_args,max_args,num_args);`
- `min_args`: 引數最小數目, `max_args`: 引數最大數目, `num_args`: 實際引數數目。
- 假使引數的數目不在可接受的範圍, 將會產生一個標準的錯誤訊息。
- 假使引數的數目在可接受範圍內, 則函式將會傳回一個空字串。

● error 語法

- `error('msg')`, 其中 `msg` 是包含錯誤訊息的字串。
- `error` 可與 `nargchk` 互相搭配使用。
- 當程式發生錯誤時, 產生一個錯誤訊息並停止執行。
- 如果程式沒有錯誤, 就只輸出一個空字串。



- **warning 語法**

- `warning('msg')`，其中 `msg` 是包含警告訊息的字串。
- 會告知使用者發生問題的函式名稱，以及程式發生問題的位置，但程式仍繼續執行而不會中斷。

- **inputname 語法**

- `name = inputname(argno);`
- 會傳回所使用的真實引數名稱，其中 `argno` 是引數的序數。
- 如果引數是個變數，函式會傳回變數名稱。
- 假如引數是一段敘述，則會傳回一個空字串。



範例 6.3: 使用選擇性引數

- 產生一個函式 `polar_value`，將直角座標 (x, y) 轉換成等價的極座標 (r, θ) 。
- 如果只有輸入一個引數，則函式將會假設 y 值為 0，並繼續完成轉換的計算。
- 正常的情況下，函式將傳回距離 r 與角度 θ (以度為單位) 兩個數值，但如果呼叫此函式的敘述式只有一個輸出引數，它將只會傳回距離值 r 。



MATLAB 程式碼 (檔名: polar_value.m)

```
function [mag,angle] = polar_value(x,y)
msg = nargchk(1,2,nargin);
error(msg);
if nargin < 2
    y = 0;
end
if x == 0 & y == 0
    msg = 'x and y are zero: angle is meaningless!';
    warning(msg);
end
mag = sqrt(x.^2 + y.^2);
if nargout == 2
    angle = atan2(y,x) * 180/pi;
end
```



程式執行結果 (1/2)

```
>> [r, theta] = polar_value(1, -1)
```

```
r =
```

```
1.4142
```

```
theta =
```

```
-45
```

```
>> r = polar_value(1, -1)
```

```
r =
```

```
1.4142
```

```
>> [r, theta] = polar_value(-2)
```

```
r =
```

```
2
```

```
theta =
```

```
180
```



程式執行結果 (2/2)

```
>> [r, theta] = polar_value(3, 4)
```

```
r =
```

```
5
```

```
theta =
```

```
53.1301
```

```
>> [r, theta] = polar_value(0, 0)
```

```
Warning: x and y are zero: angle is meaningless!
```

```
> In polar_value at 32
```

```
r =
```

```
0
```

```
theta =
```

```
0
```



Section 6.4

使用共用記憶體分享資料



共用記憶體與全域變數

- 除了引數清單之外，MATLAB 函式也能在工作區內使用**共用記憶體**，以交換彼此間的資料。
- **共用記憶體 (global memory)** 是一種特別型態的記憶體，能在任何工作區內存取。
- 對於**分享函式間的大量資料特別有用**，因為每次函式被呼叫時，就**不需要複製整個資料集合**。
- **全域變數 (global variables)** 是藉由 `global` 宣告來產生，其宣告形式如下：

表 8.4.2 使用全域變數

語法	說明
<code>global var₁ var₂ ...</code>	宣告全域變數 var_1, var_2, \dots
<code>whos global</code>	查詢工作區內的全域變數
<code>clear global var₁ var₂ ...</code>	刪除全域變數 var_1, var_2, \dots



- 在 MATLAB 函式內部的變數均是**區域變數 (local variables)**。
- 以**大寫字母宣告全域變數**，使其容易與區域變數區別。
- 通常在函式中的使用說明與**第一個可執行的宣告式之間**，宣告所有的全域變數。
- 每個全域變數在函式內第一次使用前，必須先宣告其為全域變數。如果在局部工作區產生一個變數後才宣告其為全域變數，將會產生錯誤。



全域變數的範例

使用 `global` 宣告的函式: `test_global.m`

```
function test_global(num)
```

```
global VAR; % 宣告 VAR 為一個全域變數  
VAR = VAR * num; % 改變 VAR 的內存數值  
fprintf(' 在函式內 · VAR = %g \n', VAR);
```

```
>> global VAR
```

```
>> VAR = 3.14;
```

```
>> test_global(-2)
```

```
在函式內 · VAR = -6.28
```

```
>> num
```

```
Undefined function or variable 'num'.
```

```
>> VAR
```

```
VAR =
```

```
-6.2800
```



Section 6.5

函式呼叫間的資料保存



- 續存記憶體 (**persistent memory**)
 - 一種特別類型的記憶體，只允許在函式內存取。
 - 在函式呼叫之間，其值保持不變。
- 續存變數宣告式

`persistent var1 var2 var3...`

其中 `var1`、`var2`、`var3` 等，是保存在續存記憶體的變數。



範例 6.5: 動態平均值 (1/4)

1. 宣告問題

產生一個函式，當輸入一個新的數值時，即時計算資料的動態平均及標準差。這個函式也需要應使用者的需要，重置動態總和統計值。

2. 定義輸入和輸出

- 兩種類型的輸入值：
 - ① 使用字元字串「reset」重置動態總和統計值歸零。
 - ② 輸入資料集合裡的數值，每呼叫函數一次，便顯示一個數值。
- 輸出值: 提供資料的**動態平均值與標準差**。

3. 設計演算法

- Step 1 檢查合法的引數個數
- Step 2 檢查'reset'，並重置顯示總和值
- Step 3 增加新的數值到動態總和值
- Step 4 如果有足夠多的資料，計算並傳回動態平均值與標準差。
如果資料不夠的話，就傳回零。



範例 6.5: 動態平均值 (2/4)

4. MATLAB 程式碼 (函式檔名: runstats.m)

```
function [ave, std] = runstats(x)
persistent n sum_x sum_x2
msg = nargchk(1,1,nargin); error(msg);
if x == 'reset'
    n = 0; sum_x = 0; sum_x2 = 0;
else
    n = n + 1; sum_x = sum_x + x; sum_x2 = sum_x2 + x^2;
end
if n == 0
    ave = 0; std = 0;
elseif n == 1
    ave = sum_x; std = 0;
else
    ave = sum_x / n;
    std = sqrt((n*sum_x2 - sum_x^2) / (n*(n-1)));
end
```



範例 6.5: 動態平均值 (3/4)

4. MATLAB 程式碼 (主程序檔名: test_runstats.m)

```
% First reset running sums
[ave, std] = runstats('reset');

% Prompt for the number of values in the data set
nvals = input('Enter number of data: ');

% Get input values
for ii = 1:nvals
    % Prompt for next value
    string = ['Enter value ' int2str(ii) ': '];
    x = input(string);
    % Get running statistics
    [ave, std] = runstats(x);
    % Display running statistics
    fprintf('Ave = %8.4f; Std = %8.4f \n',ave,std);
end
```



範例 6.5: 動態平均值 (4/4)

5. 測試程式

試輸入五筆資料 **3, 2, 3, 4, 2.8**，並計算其移動平均值與標準差。

```
>> test_runstats
```

```
Enter number of data: 5
```

```
Enter value 1: 3
```

```
Ave = 3.0000; Std = 0.0000
```

```
Enter value 2: 2
```

```
Ave = 2.5000; Std = 0.7071
```

```
Enter value 3: 3
```

```
Ave = 2.6667; Std = 0.5774
```

```
Enter value 4: 4
```

```
Ave = 3.0000; Std = 0.8165
```

```
Enter value 5: 2.8
```

```
Ave = 2.9600; Std = 0.7127
```



Section 6.6

函式握把



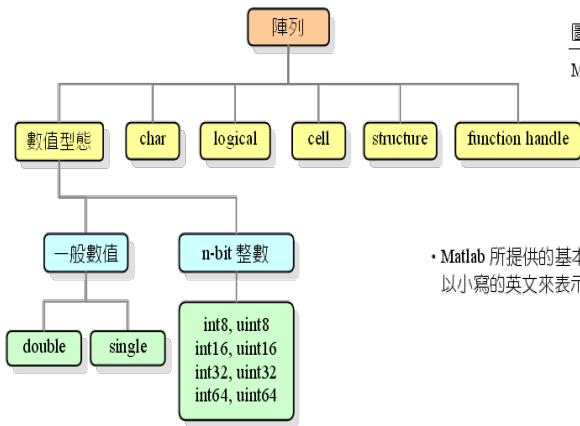


圖 3.1.1

Matlab 提供的資料型態

- Matlab 所提供的的基本資料型態以
小寫的英文來表示



何謂函式握把?

- 函式握把 (**function handle**) 是 MATLAB 的一種資料型態，它保有呼叫一個函式所需的資訊。
- MATLAB 提供兩種方法來產生函式握把：
 - ① 使用 **@** 運算子來產生函式握把時，只需要將運算子放在函式名稱前面即可。
`fh = @函式名稱;`
 - ② 使用 **str2func** 產生函式握把時，必須將函式的名稱當成輸入引數的字串。
`fh = str2func('函式名稱');`



函式握把的範例 (1/2)

使用者定義的函式: my_func.m

```
function y = my_func(x)
```

```
y = x.^2 - 2*x + 1;
```

```
>> fh1 = @my_func; % 宣告 fh1 為一個函式握把
```

```
>> [fh1(4), my_func(4)]
```

```
ans =
```

```
9    9
```

```
>> fh2 = @randn;
```

```
>> fh2() % 若沒有輸入引數時，必須加上圓括弧 ()
```

```
ans =
```

```
-0.1241
```



函式握把的範例 (2/2)

```
>> whos
```

Name	Size	Bytes	Class
ans	1 × 1	8	double
fh1	1 × 1	32	funciton_handle
fh2	1 × 1	32	funciton_handle

```
>>> feval(fh1, 4) % 效果與 fh1(4) 相同
```

```
ans =
```

```
9
```

```
>> func2str(fh1) % 找回函式握把 fh1 的原函式名稱
```

```
ans =
```

```
my_func
```



處理函式握把的 MATLAB 函式

函式	描述
@	產生函式握把。
feval	使用函式握把來求取函式的數值。
func2str	找回一個函式握把之函式名稱。
functions	由函式握把找回其各種資訊，並以結構形式傳回資料。
str2func	從特定的字串產生一個函式握把。



Section 6.7

匿名函式



- 匿名函式 (**anonymous function**) 是一個「沒有名稱」的函式。
- 它是用單一行 MATLAB 敘述所宣告的函式，而且會回傳一個函式握把，然後可以用此握把來執行此函式。
- 可以在指令視窗裡直接定義一個匿名函式，而不用把函式寫在 M 檔案裡。
- 一般形式為

表 8.6.1 匿名函數的定義

指令	說明
$fname=@(arg_list) \ expr$	定義匿名函數，函數名稱為 $fname$ ，輸入引數為 arg_list ，函數的內容則定義在 $expr$ 的位置



單自變量函數的範例

```
>> f = @(x) cos(x) - x; % 宣告 f 是一個函式握把
```

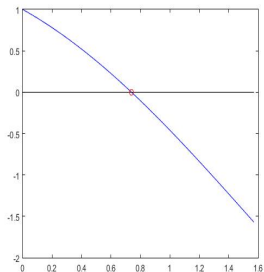
```
>> z = fzero(f,[0,pi/2]) % 求出函數 f 的零根
```

```
z =
```

```
0.7391
```

```
>> x = linspace(0,pi/2); y = f(x); fz = f(z);
```

```
>> plot(x,y,'b-',x,zeros(size(x)),'k-',z,fz,'ro');
```



雙自變量函數的範例

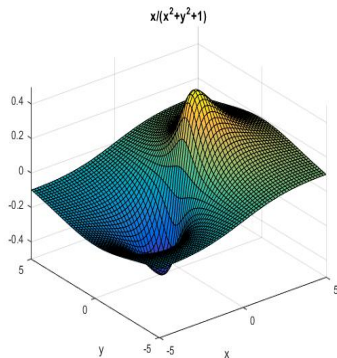
```
>> g = @(x,y) x./(x.^2+y.^2+1);
```

```
>> g(1,1)
```

```
ans =
```

```
0.3333
```

```
>> ezsurf(g, [-5,5,-5,5]);
```



Section 6.8

MATLAB 內建函式: 排序與亂數涵式



- `sort` 函式可將一組資料排序成遞增 (ascending order) 或遞減順序 (descending order)。
- 如果這組資料是一個行或列向量，整組資料會進行排序。
- 如果這組資料是一個二維矩陣，則矩陣的行向量 (column vectors) 會被分別排序。
- 宣告形式如下：

```
res = sort(a);           % 排成遞增順序  
res = sort(a, 'ascend'); % 排成遞增順序  
res = sort(a, 'descend'); % 排成遞減順序
```



函式 sort 的使用範例

```
>> a = [1 4 5 2 8];
```

```
>> sort(a,'descend')
```

```
ans =
```

```
8 5 4 2 1
```

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

```
>> b >> sort(b)
```

```
b = ans =
```

```
1 5 2 1 4 2
```

```
9 7 3 8 5 3
```

```
8 4 6 9 7 6
```



- `sortrows` 函式是 `sort array rows` 之意。
- 將一個矩陣的資料依照**某一或某些特定欄位 (columns)** 的資料進行**遞增或遞減**排序。
- 宣告形式如下：

```
res = sortrows(a);           % 第 1 欄資料排成遞增順序  
res = sortrows(a,n);       % 第 n 欄資料排成遞增順序  
res = sortrows(a,-n);      % 第 n 欄資料排成遞減順序
```

- 宣告式 `res = sortrows(a, [m n])`
 - 主要先針對第 m 個欄位的列資料進行排序。
 - 在第 m 個欄位中，若有兩個或兩個以上的元素相同，則再依據第 n 個欄位相對應的列資料進行排序。



函式 sortrows 的使用範例

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

```
>> c
```

```
c =
```

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

```
>> sortrows(c,-1)
```

```
ans =
```

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

```
>> sortrows(c,[2 3])
```

```
ans =
```

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



常用的內建亂數函式

下表為函式 `rand` 和 `randn` 的宣告形式：

宣告形式	說明
<code>rand</code> 或 <code>rand()</code>	產生一個在 $[0, 1)$ 區間內均勻分布的亂數。
<code>rand(n)</code>	產生元素在 $[0, 1)$ 區間內均勻分布的 $n \times n$ 亂數矩陣。
<code>rand(m,n)</code>	產生元素在 $[0, 1)$ 區間內均勻分布的 $m \times n$ 亂數矩陣。
<code>randn</code> 或 <code>randn()</code>	產生一個標準常態分布的亂數。
<code>randn(n)</code>	產生元素為標準常態分布的 $n \times n$ 亂數矩陣。
<code>randn(m,n)</code>	產生元素為標準常態分布的 $m \times n$ 亂數矩陣。



Thank you for your attention!

