第二章

初探 MATLAB

本章重點

本章介紹 MATLAB 的基本環境與操作，如果您是 MATLAB 的初學者，建議您先熟悉本章各節的內容，可讓您很快地進入情況，立即享用 MATLAB 的簡潔與方便。當然， 如果您是 MATLAB 老手，就可以直接跳到後面各章節，研讀各個相關主題。

2-1 使用變數與基本運算

MATLAB 認識一般常用到的加（+）、減（-）、乘（*）、除（/）、冪次（^）等數學運算符號，因此在 MATLAB 下進行基本運算，最快速簡單的方式是在 上一節介紹過的 — 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
```

MATLAB 會將運算結果直接存入預設變數 ans，代表 MATLAB 運算後的答案（Answer），並在螢幕上顯示其運算結果的數值（在上例中，即為 2.7000）。

提示：MATLAB 命令提示符號

- 在本書中，“ >> ”代表 MATLAB 命令視窗內的提示符號（Command Prompt），使用者可以在其後面輸入任何 MATLAB 語法的運算式，以進行處理。
- “ >> ”是 MATLAB 第五版原始設計的命令提示符號，但由於中英文編碼方式不同，在 Windows 95/98 中文視窗系統下，此命令提示符號常會消失不見，而在 Windows NT/2000/XP 中文視窗系統下，則會改變成「？」，但這並不會影響到 MATLAB 的運算結果。（註：但自從 MATLAB 第六版以後，則這些問題都已經不存在了！）

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;
```

在上例中，由於運算式後面有加入分號，因此 MATLAB 只會將運算結果儲存在預設變數 ans 內，不會顯示於螢幕上；若有需要取用或顯示此運算結果，可直接鍵入變數 ans 即可，例如：

```
>> ans
ans =
    2.7000
```

使用者也可將運算結果儲存於使用者自己設定的變數 x 內，例如：

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

提示：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。

若要加入註解（Comments），可以使用百分比符號（%），MATLAB 會將所有在百分比符號之後的文字視為程式的註解，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
```

在上例中，百分比符號之後的文字都是註解，會被 MATLAB 忽略而不執行，但是註解的使用可提高 MATLAB 程式的可讀性。

MATLAB 可同時執行以逗號（,）或分號（;）隔開的數個運算式，例如：

```
>> x = sin(pi/3); y = x^2; z = y*10
z =
    7.5000
```

若一個數學運算是太長，可用三個句點（...）將其延伸到下一行，例如：

```
>> z = 10*sin(pi/3)*...
>> sin(pi/3);
```

2-2 向量與矩陣的處理

在上一節的各個範例例中，我們使用 MATLAB 的變數來儲存純量（Scalars），其實 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
```

在上例中，MATLAB 使用中括號（[]），來建立一個列向量（Row Vector） [1 3 5 2]，將其儲存在變數 s 中，再對其進行運算產生另一新的列向量 [3 7 11 5]，並將其結果儲存在變數 t 內。

提示：

- s = [1 3 5 2] 與 s = [1, 3, 5, 2] 的效果是一樣的。
- 我們也可以使用 x = 1:n 來產生由 1 到 n 的列向量。
- 一個長度為 n 的列向量也可以看成是大小為 1×n 的矩陣。

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

>> s(2)*3 + t(4)       % 取出 s 的第二個元素和 t 的第四個元素來運算
```

```
ans =
     9

>> t(2:4) - 1          % 取出向量 t 的第二至第四個元素來做運算
ans =
     6     1    -1
```

用類似上述建立向量的方法，使用者也可以直接建立大小為 m×n的矩陣 （m 代表矩陣的橫列數，n 代表矩陣的直行數），但必需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
```

同樣地，我們可以對矩陣進行各種處理，例如：

```
>>  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

>> 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

>> 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
     9    11    12     5
```

善用這幾種矩陣處理的方式，可以產生意想不到的效果，就看各位的巧思與創意。有興趣的讀者，可以參閱本書第九章「矩陣的處理與運算」與本書姊妹作「MATLAB程式設計：進階篇」的第二章「程式碼與記憶體之最佳化」。

提示：MATLAB 內部儲存資料結構

- 在 MATLAB 的內部資料儲存結構中，每一個矩陣都是一個以行為主（Column-oriented）的向量（Vector），因此對於矩陣內各元素的存取，我們可用一維或二維的索引（Index）或下標（Subscript）來定址。舉例來說，在上述矩陣 A 中，若需取用位於第二列、第三行的元素，可採取二維索引的存取方式寫為 A(2, 3)，或可採取一維索引的存取方式寫為 A(6)，此即對矩陣 A 中的所有直行進行堆疊後取用第六個元素。
- A(:) 就是將矩陣 A 每一直行堆疊起來，成為一個行向量，而這也是 MATLAB 變數的內部儲存方式。以前例而言，A(:) 將會產生一個 8×1 的二維矩陣或一內有八個元素的行向量。

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，例如：

```
>> x = 4;
```

```
>> y = abs(x)                    % 取 x 的絕對值
y =
     4

>> y = sin(x)                    % 取 x 的正弦值
y =
   -0.7568

>> y = exp(x)                    % 自然指數 exp(x)
y =
   54.5982

>> y = log(x)                    % 自然對數 ln(x)
y =
    1.3863
```

MATLAB 也支援複數運算，通常以 i 或 j 代表單位虛數 $\sqrt{-1}$，例如：

```
>> z = 5 + 4j                    % 複數 z = 5 + 4√(-1)
z =
   5.0000 + 4.0000i

>> z = 5 + 4i                    % 這也是複數 z = 5 + 4√(-1)
z =
   5.0000 + 4.0000i

>> y = angle(z)                  % 複數 z 的相角
y =
    0.6747

>> y = real(z)                   % 複數 z 的實部
y =
     5

>> y = imag(z)                   % 複數 z 的虛部
y =
```

```
     4

>> y = conj(z)                    % 複數 z 的共軛複數
y =
   5.0000 - 4.0000i

>> y = z'                         % 這也是複數 z 的共軛複數
y =
   5.0000 - 4.0000i

>> y = exp(j*pi/6)                % e^(jθ) = cosθ + j sinθ
y =
   0.8660 + 0.5000i
```

其中 pi 是 MATLAB 的內建常數，代表圓周率，約等於 3.1415926…。

提示：Euler Identity

- 上述範例就是赫赫有名的「尤拉恆等式」（Euler Identity），有修過工程數學的同學應該還記得吧？
- 其實應該翻成「澳衣勒恆等式」才對，因為 Euler 是瑞士數學家，應該是德語發音。

以上這些基本的數學函數，也都通用於向量或矩陣，例如：

```
>> x = [4 2j 9];
>> y = sqrt(x)                    % 對 x 開平方
y =
   2.0000     1.0000 + 1.0000i    3.0000
```

在上例中，sqrt 指令會對 x 的每一個元素（無論是實數或複數）進行開平方的運算。

另外還有一些函數是特別針對向量而設計的，例如：

```
>> x = [1 2 3 0 12];
>> y = min(x)                     % 向量 x 的極小值
y =
```

```
     0

>> y = max(x)                    % 向量 x 的極大值
y =
    12

>> y = mean(x)                   % 向量 x 的平均值
y =
    3.6000
>> y = sum(x)                    % 向量 x 的總和
y =
    18

>> y = sort(x)                   % 向量 x 的排序
y =
     0     1     2     3    12
```

這些函數是針對向量設計的，無論輸入是行向量或列向量，都可傳回正確的結果。若輸入為矩陣時，這些函數將輸入矩陣看成是行向量的集合，並選一對行向量進行運算，例如：

```
>> x = [1 2 3;4 5 6;7 8 9];
>> y = median(x)                 % x 每個行向量的中位數
y =
     4     5     6

>> y = prod(x)                   % x 每個行向量的乘積
y =
    28    80   162
```

在本書的第九章「矩陣的處理與運算」有對這些函數的進一步介紹。

若對 MATLAB 函數用法有疑問，可隨時使用 help 指令來尋求線上支援（On-line Help），例如：

```
>> help sort                     % 查詢 sort 指令的線上使用說明
```

指令執行結果：

```
SORT   Sort in ascending order.
For vectors, SORT(X) sorts the elements of X in ascending
order.
For matrices, SORT(X) sorts each column of X in ascending
order.
For N-D arrays, SORT(X) sorts the along the first non-
singleton dimension of X. When X is a cell array of
strings, SORT(X) sorts the strings in ASCII dictionary
order.

    SORT(X,DIM) sorts along the dimension DIM.

[Y,I] = SORT(X) also returns an index matrix I. If X is
a vector, then Y = X(I).  If X is an m-by-n matrix, then
for j = 1:n, Y(:,j) = X(I(:,j),j); end
When X is complex, the elements are sorted by ABS(X).  Complex
matches are further sorted by ANGLE(X).

    Example: If X = [3 7 5
                     0 4 2]

    then sort(X,1) is [0 4 2  and sort(X,2) is [3 5 7
                       3 7 5]                    0 2 4];

    See also SORTROWS, MIN, MAX, MEAN, MEDIAN.

     Overloaded methods
 help cell/sort.m
```

提示：MATLAB 的查詢指令及線上支援

▶▶ help：用來查詢已知指令的用法。例如已知 inv 是用來計算反矩陣，鍵入 help inv 即可得知有關 inv 指令的用法。（鍵入 help help 則顯示 help 的用法，請試看看！）

- lookfor：用來尋找未知的指令。例如要尋找計算反矩陣的指令，可鍵入 lookfor inverse，MATLAB 即會列出所有和關鍵字 inverse 相關的指令。找到所需的指令後，即可用 help 進一步找出其用法。（lookfor 事實上是對所有在搜尋路徑下的 M 檔案進行關鍵字對「第一註解行」的比對，詳見本章第五節或本書第十五章有關於「M 檔案」的說明。）
- 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。（例如：讀者可嘗試輸入指令「doc eig」，來驗證一下。

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，最簡單的程式重複執行指令是 for 迴圈（For-loop），其基本使用語法為：

```
For 變數 = 向量
        運算式;
end
```

其中變數的值會被依次設定為向量的每一個元素值，來重複執行介於 for 和 end 之間的運算式。因此，若無意外情況，運算式執行的次數會等於向量的長度。舉例來說，下列指令會產生一個長度為 6 的調和數列（Harmonic Sequence）：

```
>> x = zeros(1,6);          % 預先配置矩陣 x 為一個維度 1×6 的零矩陣
>> for i = 1:6
>>         x(i) = 1/i;
>> end
>> disp(x)                  % 顯示矩陣 x 的內容
    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（即矩陣 1:6 或 [1 2 3 4 5 6] 中的每一個元素），因此矩陣 x 的第 i 個元素的值依次被設為 1/i 。

另一個常用到的程式重複執行指令是 while 迴圈（While-loop），其基本使用語法為：

```
while 條件式
      運算式;
end
```

也就是說，只要條件式成立，運算式就會一再被重複執行。例如先前用 for 迴圈產生的調和數列的作法，我們可用 while 迴圈改寫如下：

```
>> x = zeros(1,6);          % x 是一個 1X6 的零矩陣
>> i = 1;
>> while i <= 6
>>       x(i) = 1/i;
>>       i = i + 1;
>> end
>> disp(x)                  % 顯示矩陣 x 的內容
    1.0000    0.5000    0.3333    0.2500    0.2000    0.1667
```

提示：預先配置矩陣

上面的幾個例子中，我們使用 zeros 來預先配置（Pre-allocate）了一個適當大小的矩陣。若不預先配置矩陣，程式仍可執行，但此時 MATLAB 需要動態地增加（或減小）矩陣的大小，因而降低程式的執行效率。所以在使用一個矩陣時，若能在事前知道其大小，則最好先使用 zeros 或 ones 等指令來預先配置矩陣所需的記憶體大小，以增進程式的執行效率。

MATLAB 也提供依條件判斷來控制程式流程的指令，最常見為 if – else – end 的指令組合，其基本使用形式為：

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

```
>> if rand(1,1) > 0.5
>>       disp('Given random number is greater than 0.5.');
>> else
```

```
>>       disp('Given random number is smaller than 0.5.');
>> end
Given random number is less than 0.5.
```

有關程式流程控制，在本書的第十六章「程式流程控制」將會有更詳盡的介紹。

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。此種包含 MATLAB 指令的檔案都以 m 為副檔名，因此通稱 M 檔案（M-files）。例如一個名為 myTest.m 的 M 檔案，包含一連串的 MATLAB 指令，那麼只要直接鍵入 myTest，即可執行其所包含的指令：

```
>> pwd          % 顯示目前的工作目錄（pwd = present working directory）
ans =
C:\MATLAB6p5\work
```

接著我們要用 cd 指令來進入我們的範例檔案 myTest.m 所在的目錄：

```
>> cd d:\matlabBook\MATLAB程式設計：入門篇\02-初探MATLAB
```

（如果此目錄路徑包含空格，我們就必須使用英文的單引號來包夾此目錄路徑。）

提示：如何取得本書的範例程式

▶▶ 在進行上述範例時，你必須先從本書所附的範例光碟片中，將所有的範例程式拷貝到「d:\matlabBook\MATLAB 程式設計：入門篇」，詳見光碟內的說明。你也可以由本書作者的網頁（http://www.cs.nthu.edu.tw/~jang）連到本書的網頁，就可以下載相關的範例程式碼。

接著我們可以顯示 myTest.m 的內容，如下：

```
>> type myTest.m                    % 顯示 myTest.m 的內容
% myTest: my first test M-file.
```

```
% Roger Jang, March 3, 1997
fprintf('Start of myTest.m!\n');
for i = 1:3
     fprintf('i = %d ---> i^3 = %d\n', i, i^3);
end
fprintf('End of myTest.m!\n');
```

接著我們就可以執行 myTest.m：

```
>> myTest                              % 執行 myTest.m
Start of myTest.m!
i = 1 ---> i^3 = 1
i = 2 ---> i^3 = 8
i = 3 ---> i^3 = 27
End of myTest.m!
```

提示：第一註解行（H1 Help Line）

myTest.m 的前兩行是註解，可以使程式易於瞭解與管理。特別要說明的是，第一註解行通常用來簡短說明此 M 檔案的功能，以便 lookfor 能以關鍵字比對的方式來找出此 M 檔案。舉例來說，myTest.m 的第一註解行包含 myTest 這個字，因此如果鍵入 lookfor myTest，MATLAB 即可列出所有在第一註解行包含 myTest 的 M 檔案，因而 myTest.m 也會被列名在內。

嚴格來說，M 檔案可再細分為底稿 （Scripts） 及函數（Functions）。基本上，底稿（例如前述 myTest.m 檔）的效用和將個別 MATLAB 的指令或運算式，在 MATLAB 的命令視窗內逐一輸入完全一樣，因此在底稿執行時，可以直接使用儲存於工作空間的變數，而且在底稿中設定的變數，也都在工作空間中看得到。至於函數則需要用到輸入引數（Input Arguments）和輸出引數（Output Arguments）來傳遞資訊，這就像是 C 語言的函數，或是 FORTRAN 語言的副程序（Subroutines）。舉例來說，若要計算一個正整數的階乘（Factorial），我們可以寫一個如下的 MATLAB 函數並將之存檔於 fact01.m：

```
>> type fact01.m
function output = fact01(n)
% FACT01 Calculate factorial of a given positive integer (for-loop
version)
output = 1;
```

```
for i = 1:n,
      output = output*i;
end
```

其中 fact01 是函數名稱，n 是輸入引數，output 是輸出引數，而 i 則是此函數用到的暫時變數。要使用此函數，直接鍵入函數名稱及適當輸入引數值即可：

```
>> y = fact01(5)
y =
   120
```

（當然，在執行 fact01 之前，您必需先進入 fact01.m 所在的目錄。）在執行 fact01(5) 時，MATLAB 會跳入一個下層的暫時工作空間（Temperary Workspace），將變數 n 的值設定為 5，然後進行各項函數檔內部的各項運算，在此階段產生的所有變數（包含輸入引數 n、暫時變數 i，以及輸出引數 output）都會儲存在此暫時工作空間中。待運算完畢後，MATLAB 會將最後輸出引數 output 的值設定給上層的變數 y，並將清除此暫時工作空間及其所含的所有變數。換句話說，在呼叫函數時，您只能經由輸入引數來控制函數的輸入，經由輸出引數來得到函數的輸出，但所有的暫時變數都會隨著函數的結束而消失，您並無法得到他們的值。

MATLAB 的函數也可以是遞迴式的（Recursive），也就是說，一個函數可以呼叫他本身。舉例來說，n! = n*(n-1)!，因此前面的階乘函數可以改成遞迴式的寫法：

```
>> type fact02.m
function output = fact02(n)
% FACT02 Calculate factorial of a given positive integer (recursive
version)

if n == 1,    % Terminating condition
       output = 1;
       return;
end

output = n*fact02(n-1);
```

提示：有關階乘函數

▶▶ 前面所用到的階乘函數只是純粹用來說明 MATLAB 的函數觀念。若實際要計算一個正整數 n 的階乘（即 n!）時，可直接寫成 prod(1:n)，或是直接呼叫 gamma 函數：gamma(n-1)。

值得注意的是，在寫一個遞迴函數時，一定要包含結束條件（Terminating Condition），否則此函數將會一再呼叫自己，進入無窮迴圈，永遠不會停止，直到電腦的記憶體被耗盡為止。以上例而言，n==1 即滿足結束條件，此時我們直接將 output 設為 1，而不再呼叫此函數本身。（請注意，在呼叫此函數時，如果 n 不是整數，一樣會進入反覆呼叫自己的無窮迴圈中。）

有關 M 檔案的其他細節，讀者可參考本書的第十五章「M 檔案」。

2-6 搜尋路徑

在前一節中，myTest.m 所在的目錄是「d:\matlabBook\MATLAB程式設計：入門篇\02-初探MATLAB」。如果不先進入這個目錄，MATLAB 就找不到您要執行的 M 檔案。如果希望 MATLAB 不論在何處都能執行 myTest.m，那麼就必需將「d:\matlabBook\MATLAB程式設計：入門篇\02-初探MATLAB」加入 MATLAB 的搜尋路徑（Search Path）上。

提示：MATLAB 指令處理程序

▶▶ 每次 MATLAB 遇到一個指令（例如 myTest）時，其處置程序為：

1. 檢查 myTest 是否為使用者定義之變數。若是，則取用之；若不是，進入下個步驟。
2. 檢查 myTest 是否為永久常數。若是，則取用之；若不是，進入下個步驟。
3. 檢查 myTest 是否為目前目錄之的 M 檔案。若是，則取用之；若不是，進入下個步驟。
4. 檢查 myTest 是否為搜尋路徑下的 M 檔案。若是，則取用之；若不是，進入下個步驟。
5. 若不是，則 MATLAB 發出錯誤訊息。

若要檢視 MATLAB 已設定的搜尋路徑，鍵入 path 指令即可：

```
>> path
        MATLABPATH
C:\MATLAB6p5\toolbox\matlab\general
C:\MATLAB6p5\toolbox\matlab\ops
C:\MATLAB6p5\toolbox\matlab\lang
```

```
C:\MATLAB6p5\toolbox\matlab\elmat
C:\MATLAB6p5\toolbox\matlab\elfun
.....
```

當然，在螢幕上看到所有傳回的搜尋路徑內容，會依各別使用者所安裝的工具箱（Toolboxes）不同而有所差異。

若只要查詢某一特定指令所在的搜尋路徑，可用 which 指令，例如：

```
>> which demo
C:\MATLAB6p5\toolbox\matlab\demos\demo.m

>> cd(matlabroot);          % 跳到 MATLAB 的根目錄
>> which fact01
fact01 not found.
```

很顯然「d:\matlabBook\MATLAB程式設計：入門篇\02-初探MATLAB」並不在 MATLAB 的搜尋路徑中，因此 MATLAB 找不到 fact01.m 這個 M 檔案：

要將目錄「d:\matlabBook\MATLAB程式設計：入門篇\02-初探MATLAB」加入 MATLAB 的搜尋路徑，可使用 addpath 指令：

```
>> addpath('d:\matlabBook\MATLAB程式設計：入門篇\02-初探MATLAB');
```

此時目錄「d:\matlabBook\MATLAB程式設計：入門篇\02-初探MATLAB」已加入 MATLAB 搜尋路徑（請讀者自己鍵入 path 指令試看看！），因此 MATLAB 已經「看」得到 fact01.m：

```
>> which fact01
d:\matlabBook\MATLAB程式設計：入門篇\02-初探MATLAB\fact01.m
```

現在我們就可以直接呼叫 fact01 這個函數，例如要計算 10!，可直接鍵入 fact01(10)，而不必先進入 fact01.m 所在的目錄。

提示：設定 MATLAB 搜尋路徑

- 如果在每一次啟動 MATLAB 後，都要設定所需的搜尋路徑，將是一件很麻煩的事。有兩種方法，可以使 MATLAB 啟動後，即可載入使用者定義的搜尋路徑：

1. MATLAB 的預設搜尋路徑是定義在 matlabrc.m（在 MATLAB 的安裝目錄之下，依版本不同，也可能還放在子目錄之下，可用微軟檔案總管的搜尋功能來找此檔案），MATLAB 每次啟動後，即自動執行此檔案。因此您可以直接修改 matlabrc.m，以加入新的目錄於搜尋路徑之中。
2. MATLAB 在執行 matlabrc.m 時，同時也會在預設搜尋路徑中尋找 startup.m，若此檔案存在，則執行其所含的指令。因此我們可將所有 MATLAB 啟動時必需執行的指令（包含更改搜尋路徑的指令），放在此檔案中。

在 MATLAB 命令視窗下，輸入「command(‘ string ’)」和「command string」是完全等效的，因此「addpath('d:\matlabBook')」可以寫成「addpath d:\matlabBook」。此外，addpath 指令通常將目錄附加至搜尋路徑之前。若要將目錄附加於搜尋目錄之後，可使用「addpath d:\matlabBook -end」。若要從搜尋路徑中移除目錄，可用 rmpath 指令，例如：「rmpath d:\matlabBook」。

除了使用命令列來增刪路徑外，您也可以由「路徑瀏覽器」（可由 MATLAB 的 pathtool 指令叫出）來進行搜尋路徑的新增或移除。

提示：目錄操作相關的指令

- pwd 指令可傳回目前工作目錄（pwd = present working directory）。
- cd 指令可改變目錄。
- dir 指令可顯示（或傳回）目前工作目錄下的內容。
- 欲呼叫 OS 的命令，可在 MATLAB 命令視窗下輸入驚嘆號及 OS 的命令。例如在微軟視窗作業系統下，「!dir」可將 DOS 命令「dir」的結果秀在 MATLAB 命令視窗內。

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）。當每次 MATLAB 呼叫執行某一函數時，即進入該函數的暫時工作空間（或可視為是相對於基本工作空間的下層暫時工作空間），函數可在此暫時工作間內產生各種變數並進行運算，而不會影響到基本工間空間內的變數。當函數執行結束時，MATLAB 會同時刪除函數的暫時工

作空間（當然也會刪除儲存於其內的所有變數），並回到 MATLAB 的基本工作空間。以下簡述幾個有關工作空間的指令。

若要檢視現存於工作空間（Workspace）的變數，可鍵入 who：

```
>> who
Your variables are:

A         ans       s         x         z
B         i         t         y
```

在上例中，只顯這些是由使用者定義的變數的名稱。若要知道這些變數更詳細的資料（例如：變數占用的記憶體大小、及其儲存的資料型態別），則可使用 whos 指令，例如：

```
>> whos
  Name      Size          Bytes  Class

  A         2x4              64  double array
  B         1x3              24  double array
  ans       1x18             36  char array
  i         1x1               8  double array
  s         1x4              32  double array
  t         1x5              40  double array
  x         1x6              48  double array
  y         1x1               8  double array
  z         1x1              16  double array (complex)

Grand total is 47 elements using 276 bytes
```

提示：檢視工作空間變數的其他方式

▶▶ 在 MATLAB 命令視窗內直接輸入 workspace 指令，亦可達到檢視工作空間的變數效果。

同時也可以使用 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：

```
>> clear A          % 刪除工作空間內的變數 A
```

```
>> clear all        % 刪除工作空間內的所有變數
```

值得注意的是，MATLAB 有些永久常數（Permanent Constants），雖然在工作空間中看不 到，但使用者可直接取用，例如：

```
>> pi
ans =
    3.1416
```

整理：MATLAB 的永久常數

常數	說明
i 或 j	基本虛數單位（即 $\sqrt{-1}$）
eps	系統的浮點（Floating-point）精確度
inf	無限大， 例如：1/0
nan 或 NaN	非數值（Not A Number），例如：0/0
pi	圓周率（= 3.1415926...）
realmax	系統所能表示的最大數值
realmin	系統所能表示的最小數值
nargin	函數的輸入引數個數
nargout	函數的輸出引數個數

在大量長時間執行 MATLAB 運算中，使用者通常希望能將計算所得各變數內容儲存在檔案中（一般會暫時儲存在工作空間內），以便將來可重新載入以進行其他處理。此時，使用者可用 save 指令儲存變數內容到檔案，而後再用 load 指令將檔案的內容載入到工作空間以進行其他處理，以下依次分述之。

在不加任何選項（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。

首先我們先跳到本章的範例目錄：

```
>> cd 'd:\matlabBook\MATLAB程式設計：入門篇\02-初探MATLAB'
```

以下為使用 save 指令的一個例子：

```
>> clear all                          % 清除工作空間內的所有變數
>> x = 10; y = [1, 2, 3];             % x, y 為新變數
>> whos                               % 列出工作空間內所有變數
  Name      Size         Bytes  Class
  x         1x1              8  double array
  y         1x3             24  double array

Grand total is 4 elements using 32 bytes

>> save test x y            % 將變數 x 與 y 儲存至 test.mat
>> dir *.mat                % 列出目前工作目錄中的所有 MAT 檔案
test.mat

>> delete test.mat     % 刪除檔案 test.mat
```

以二進制的方式儲存工作空間的變數，通常檔案會比較小，而且在載入時速度較快，缺點是無法用普通的文書軟體（例如：MS Word 或記事本）來檢視檔案內容。

若想看到檔案內容，則必需以 ASCII 的檔案格式來儲存工作空間的變數：

- save filename x -ascii：將工作空間中的變數 x 以八位元組大小（8 bytes）儲存到名為 filename 的 ASCII 檔案。
- save filename x -ascii -double：將工作空間中的變數 x 以十六位元組大小（16 bytes）儲存到名為 filename 的 ASCII 檔案（注意：此時並不會自動加上副檔名）。
- save filename -ascii -tabs：將工作空間中的所有變數以八位元組大小（8 bytes）儲存到名為 filename 的 ASCII 檔案，並將同一列相鄰的變數以定位鍵（即 Tab 鍵）隔開。

提示：二進制和 ASCII 檔案的比較

▶▶ 在 save 指令使用 –ascii 選項後，會有下列現象：

1. save 指令就不會在檔案名稱後加上 mat 的副檔名。因此以副檔名 mat 結尾的檔案通常是 MATLAB 的二進位資料檔。
2. 通常只儲存一個變數。若在 save 指令列中加入多個變數，仍可執行，但所產生的檔案則無法以簡單的 load 指令載入。有關 load 指令的用法，詳見下述。
3. 原有的變數名稱消失。因此在將檔案以 load 載入時，會取用檔案名稱為變數名稱。
4. 對於複數，只能儲存其實部，而虛部則會消失。
5. 對於儲存相同的變數，ASCII 檔案通常比二進制檔案大。

由上表可知，若非有特殊需求，我們應該盡量以二進制方式儲存資料。

在將工作空間的變數儲存於檔案後，使用 load 指令可將該檔案所儲存之變數載入工作空間，其格式如下：

- load filename：load 指令會尋找名稱為 filename.mat 的檔案，並以二進制格式載入。若找不到 filename.mat，則尋找名稱為 filename 的檔案，並以 ASCII 格式載入。
- load filename -ascii：load 指令會尋找名稱為 filename 的檔案，並以 ASCII 格式載入。

若以 ASCII 格式載入，則變數名稱即為檔案名稱（但不包含副檔名）。若以二進制載入，則可保留原有的變數名稱，例如：

```
>> clear all;                          % 清除工作空間中的變數
>> x = 1:10;
>> save testfile.dat x -ascii          % 將 x 以 ASCII 格式存至名為
                                       % testfile.dat 的檔案
>> load testfile.dat                   % 載入 testfile.dat
>> who                                 % 列出工作空間中的變數
Your variables are:

testfile    x
```

注意在上述過程中，由於是以 ASCII 格式儲存與載入，所以產生了一個與檔案名稱相同的變數 testfile，此變數的值和原變數 x 完全相同。

提示：MATLAB 第六版的「載入精靈」

▶▶ MATLAB 在第六版提供了「載入精靈」（Load Wizard），可讓讀者由圖形介面載入檔案。讀者可由 MATLAB 桌面的「File/Import Data...」試用之。

有關 MATLAB 對於檔案的讀寫功能，讀者可詳見第十八章「檔案輸出及輸入」。

2-8 離開 MATLAB

若要結束離開 MATLAB 的環境，只要選擇下列三種方式中之一進行即可：

1. 在命令視窗內，鍵入 exit 指令。
2. 在命令視窗內，鍵入 quit 指令。
3. 直接關閉 MATLAB 的命令視窗。

習 題

1. 請在 MATLAB 直接輸入下列常數，看它們的值是多少：

   (a) i
   (b) j
   (c) eps
   (d) inf
   (e) nan
   (f) pi
   (g) realmax
   (h) realmin

2. 請使用 lookfor 指令，找出具有下列功能的 MATLAB 指令。（每一項只需找出一個相關度最高的 MATLAB 指令。）

   (a) 找出矩陣的大小（即行維度和列維度）
   (b) 改變矩陣的大小（例如將 4×6 的矩陣改成 12×2）
   (c) 將矩陣左右翻轉（Left-right flip）
   (d) 將矩陣上下翻轉（Up-down flip）

(e) 找出矩陣每一直行的最大值
(f) 對矩陣的每一直行進行排序
(g) 矩陣的旋轉（Rotate）
(h) 反矩陣（Inverse matrix）的計算
(i) 求矩陣的 rank
(j) 計算矩陣的 reduced row echelon form
(k) 計算矩陣的 null space
(l) 計算矩陣的固有值（Eigenvalues）與固有向量（Eigenvectors
(m) 計算矩陣的 QR 分解（QR Decomposition）
(n) 計算矩陣的 LU 分解（LU Decomposition）
(o) 計算矩陣的奇異值分解（Singular Value Decomposition）
(p) 對向量進行快速傅立葉轉換（Fast Fourier Transform）
(q) 直角座標轉成極座標
(r) 極座標轉成直角座標

3. 寫一個 MATLAB 小程式 findN01.m，求出最小的 n 值，使得 n! > realmax。請問 n 的值是多少？此時 (n-1)! 的值又是多少？

4. MATLAB 的 sqrt 指令可對任一數值進行開平方的運算。用此指令求出下列各數的平方根，並驗算之：

(a) π
(b) 2*i
(c) -5+12*i
其中 i 是單位虛數。

5. 寫一個 MATLAB 函數 myFun01.m 來計算下列方程式：

```
y = 0.5*exp(x/3)-x*x*sin(x)
```

其中 x 是函數的輸入，y 是函數的輸出。你的函數必須能夠處理當 x 是純量或是向量的兩種情況。此外，請利用下述兩列程式碼來畫出此函數的圖形：

```
x=0:0.1:10;
plot(x, myFun01(x));
```

6. (a) 寫一個 MATLAB 函數 piFun01.m 來計算下列級數：

```
f(n) = 4*(1 - 1/3 + 1/5 - 1/7 + 1/9 - 1/11 + ...)
```

其中 n 為函數的輸入，代表上述級數的項數，級數和 f(n) 則是函數的輸出。

(b) 使用 tic 和 toc 指令來測量 piFun01(100000) 的計算時間。如果你不知道如何使用這兩個指令，請使用 help tic 及 help toc 來查出它們的用法。我的舊電腦是 Pentium 450MHz，所得的計算時間約為 2 秒。請說明你的電腦規格以及其計算時間。

7. (a) 寫一個 MATLAB 的遞迴函數 fibo.m 來計算 Fibonacci 數列，其定義如下：

```
fibo(n+2) = fibo(n+1)+fibo(n)
```

此數列的啟始條件如下：

fibo(1) = 0, fibo(2) = 1.

(b) fibo(25) 的回傳值是多少？

(c) 使用 tic 和 toc 指令來測量 fibo(25) 的計算時間。如果你不知道如何使用這兩個指令，請使用 help tic 及 help toc 來查出它們的用法。我的電腦是 Pentium 2GHz，所得的計算時間約為 3.35 秒。請說明你的電腦規格以及其計算時間。

(d) 如果你修過離散數學，就會知道 Fibonacci 數列的第 n 項可以直接表示成

```
fibo(n)={[(1+a)/2]^(n-1)-[(1-a)/2]^(n-1)}/a
```

其中 a 是 5 的平方根。寫利用上式一個 MATLAB 的非遞迴函數 fibo2.m 來計算 Fibonacci 數列。

(e) fibo2(25) 的值為何？是否和 fibo(25) 相同？

(f) 請計算 fibo2(25) 的計算時間，並和 fibo(25) 比較。

(g) 請比較並說明使用 fibo.m 和 fibo2.m 來計算 Fibonacci 數列的優缺點。

8. (a) 請寫一個函數 minxy.m，其功能是由一個二維矩陣中找出小元素，用法如下：

```
[minValue, minIndex] = minxy(matrix)
```

其中 matrix 是一個二維矩陣，minValue 則是其元素的最小值，而 minIndex 是一個長度為 2 的正整數向量，代表最小值的索引。（換句話說，matrix(minIndex(1), minIndex(2)) 的值即是 minValue。）

(b) 請測試 [minValue, minIndex] = minxy(magic(20)) 所傳回來的 minValue 和 minIndex 各是多少？

提示：請盡量使用 min 指令。

9. 如果 a = [92, 95, 58, 75, 69, 82]，但我們執行下列 sort 指令：

```
[b, index] = sort(a)
```

會得到 b = [58, 69, 75, 82, 92, 95] 及 index = [3, 5, 4, 6, 1, 2]，其中 index 的每個元素代表 b 的每個元素在 a 的位置，滿足 b 等於 a(index)。請寫一個函數 sort01.m，當輸入為 a 時，可傳回 index2，滿足 a 等於 b(index2)。以上述 a 向量為例，傳回的 index2 應該是 [5, 6, 1, 3, 2, 4]。

提示：你可以使用迴圈（如 for-loop 和 while-loop 等）完成此作業，但是程式碼會比較凌亂，執行效率也會變差。所以請盡量利用 sort 指令，而不要用到迴圈。

10. 假設在期中考後，我們用一個向量 x 來儲存每個人的考試成績。請寫一個函數 ranking01.m，輸入為成績向量 x，輸出則是此成績的排名。例如，當 x = [92, 95, 58, 75, 69, 82] 時，ranking(x) 回傳的排名向量則是 [2, 1, 6, 4, 5, 3]，代表 92 分是排名第 1，95 分是排名第 2，58 分是排名第 6，等等。

提示：你可以使用迴圈（如 for-loop 和 while-loop 等）完成此作業，但是程式碼會比較凌亂，執行效率也會變差。所以請盡量利用 sort 指令，而不要用到迴圈。