

# MATLAB 的數學實驗

從資料分析學 MATLAB 程式設計

從 MATLAB 程式設計瞭解統計

---

汪群超

updated 2010.2



## 2 MATLAB 的數學實驗

## 目錄

序	4
再序：不可變與不可不變：我的程式寫作觀	7
1 從繪製函式圖認識 MATLAB	9
2 MATLAB的數學運算	25
3 MATLAB的機率分配	33
4 MATLAB的程式檔：以微分的計算為例	47
5 MATLAB 程式的迴圈技巧	59
6 MATLAB 的積分計算與程式設計觀念	71
7 MATLAB的副程式	83

## 4 MATLAB 的數學實驗

## 序

如果學習數學相關的學科是痛苦的，那真是一個天大的誤會。數學長久以來被『妖魔』化了。在許多學生心中有一種無形的恐懼，甚至厭惡。有些人選擇提早擺脫數學的糾纏，但有些人卻一直揮之不去，走到哪裡都會碰到數學，或數學相關的話題或應用的領域。

逃避未必求得正果，逃避只是摀著眼睛假裝看不到，一切的逃避或美其名的以不感興趣迴避，都是錯把數學當成表皮摻有農藥的蘋果。儘管知道裡面好吃，卻不敢去碰。但數學能力對一個人的重要性不會因此消失。因為數學能力的展現不一定用來解決數學問題。或許因為如此隱晦，才會引導學生對數學的學習做出零和的決定：學或不學，而且往往是一輩子的賭注。

這本單元式的講義企圖挽回一般學生對數學莫名的恐懼，進而開始喜歡上它。不管你以前多麼痛恨數學，從此刻起，不計前嫌的再一次面對數學。這一次讓電腦來幫幫忙，透過電腦程式的寫作去了解數學的內涵與精神。數學題材不在深，電腦程式不在精闢，一切都是玩票的。學完後，你不會成為電腦程式專家，更不會變成數學家，但是你可能不再討厭數學，且對電腦程式的運作有些概念。或許不知不覺中，數學與電腦會激盪出你未來求知求學的另一番憧憬。幾句話充作參考

用電腦來解決數學問題，比較輕易的化解對數學的厭惡與對電腦的恐懼。

用電腦來解決數學問題，找不到答案也可以觀察到許多未知的領域。

用電腦來解決數學問題，不知不覺中，觀察、解析問題的能力提昇了。

用電腦來解決數學問題，時間似乎流逝的特別快，你已經浸在裡面了。

用電腦來解決數學問題，看問題的角度變大了、變寬廣些了。

## 6 MATLAB 的數學實驗

對數學的畏懼來自不當的教學或失敗者的恫嚇。不了解其實學習數學是培養各種領域專長的催化劑。數學不見得是第一線的武器，但它永遠是後勤的資源。常常隱而不見，需要時，卻自然流露。不要小看數學的影響力，它無所不在、無孔不入，你只是沒有得到適當的引導！這本講義透過獨立單元介紹一些統計系學生會接觸到的數學，並結合數學軟體MATLAB，將數學的內涵呈現在螢幕上。這本講義的編排方式不是朝向完整教科書的巨細靡遺，僅作為上課練習的腳本與課後作業的參考，上課的過程仍是必須的。部分內容摘自同學的作品。當學生的數學情緒被激發時，我似乎看到潛藏在他們內心理面，受到壓抑的數理能力，他們的發現往往超過我的預期。

汪群超

2002年7月於台北大學

## 再序

# 不可變與不可不變：我的程式寫作觀

教學七年了，這本講義也用了三年。其間經過多次的修改，不管擴編還是刪減，多半是依據上課時學生的反應而來。這本講義其實很多地方寫得不夠詳細，本想進一步將所有細節完整呈現，成為一本書。但幾經思量，仍維持原貌，原因是太詳細的內容會養成學生的依賴心，喪失原先期望學生自己去補足不清楚、不詳細的部分。希望學生藉著這門課拾回過去學得不清不白的微積分、統計學與線性代數。

這本講義企圖將數學原理以電腦數據圖表的方式呈現出來，再要求同學以文字圖案呈現出其間的條理，這樣的訓練是現今大學生十分欠缺的。說穿了就是「表達的能力」的培養。這可不是說、學、逗、唱之類的表達，而是一種試圖將不易說清楚或難以理解的東西，透過文字、圖表或語言將它交代清楚。這樣的能力絕對需要長時間的訓練，有了這項「絕技，」大學畢業生不必急著說自己學非所用。有太多的事實證明，擁有絕佳的表達能力，放諸四海都餓不著肚子。

表達能力的養成必須按部就班，一點都急不得。可惜的是，莘莘學子不是自作聰明，便是固執己見，往往喜歡憑自己過去的經驗來解決未知的問題，缺乏耐心去熟練不熟悉工具，不願將專注力用在問題的觀察。學習過程像極矇著雙眼亂砍亂殺，到頭來學不到東西還怪老師出太多怪怪的功課，既對升學沒有幫助，也無助以後做生意賺大錢，不多久便放棄了，殊是可惜。告訴他這是未來升官發財的利器，他當你在三娘教子，在家裡聽多了。

以寫作程式為例，每一種程式語言都有其語法規範，該怎麼寫怎麼用，一點也馬虎不得，連錯一點點都不行，沒得商量的。初學者往往輕忽之，不喜歡被「規範」束

## 8 MATLAB 的數學實驗

縛，不顧老師一再地提醒，愛怎麼寫就怎麼寫，天才般的自己編撰起語法來了，結果當然是錯誤百出，急得老師在一旁乾著急。更有甚之，錯了還不認帳，直呼語法太不人性化，不能隨意更動，學它何用，便率性的打起電動或MSN來了。

寫程式首要遵守語法教條，待熟悉語法規範之後，才能漸漸懂得運用，透過寫一些不痛不癢的小程式，一方面熟悉語法，一方面體驗其威力。漸熟，才慢慢從觀察別人寫的「模範程式」中，瞭解死的語言原來也能玩出活把戲，這才一步步進入寫作程式的精髓，進一步玩出樂趣。這道理亙古不變，古今達人不管學習琴棋書畫，還是打拿摔跌等武藝，無不遵循這樣的哲理<sup>1</sup>

能力未至不可變也、學識未敷不得變也、功侯未到不能變也。  
學於師已窮其法，不可不變也、友古人已悉其意，不得不變也、  
師造化已盡其理，不能不變也。

從「不可變」、「不得變」、「不能變」，到「不可不變」、「不得不變」、「不能不變」，可以作為寫作程式的養成過程。學習之初應謹慎遵循所有的規範，一絲不苟，不能濫用自己的小聰明亂抄捷徑，要聽話、要服從，將老師的交代與叮嚀當作聖旨般遵循，務必做到。如此這般一段時日之後，犯錯愈少，進步愈多，自然而然當變則變，逐漸形成自己的風格。

不能急，成就總在不知不覺中「赫然」被別人發現，絕非刻意營造而能得。別人眼中看到的成就，對自己而言永遠都是平常事而已，只不過在許多小地方比別人好一點點罷了。但別小看這一點點，許許多多的一點點累積起來，那可有多少啊！

汪群超

2005年2月於台北大學

---

<sup>1</sup> 摘自五絕奇人鄭曼青先生名著「曼髯三論。」



# 第 1 章

## 從繪製函式圖認識 MATLAB

能夠在畫面上看到繪製的數學函式，可以拉進與數學間的疏離感。如果能夠親自動手操作電腦畫出在書本上曾經看過的函數圖形，那會是件令人興奮的事。本單元試圖讓學生了解電腦繪製函式圖形的原理，並藉此熟練 MATLAB 這個數學軟體的特色，特別是它的矩陣表示法。

### 本章將學到關於程式設計

變數的觀念與設定、MATLAB 的四則運算、變數矩陣的索引、函數的計算與繪圖的觀念。另外在 MATLAB 軟體的使用及繪圖功能也有一些簡單的介紹，方便初學者儘早掌握 MATLAB 所提供的優勢。

#### 〈本章關於 MATLAB 的指令與語法〉

操作元 (operators): + - × / . ^ : ;

指令: zeros, ones, eye, diag, plot, polyval, sin, cos, exp, sqrt, nthroot, pi, set, xlabel, ylabel, title, grid, text, gtext, ezplot

語法: 矩陣的建立

## 10 MATLAB 的數學實驗

### 1.1 練習

MATLAB以描點法來繪圖。所以對二維圖形而言, 必須先產生函式上對應的 (x,y) 點。然後將這些點以符號的方式顯示在一定範圍的座標軸上。下面的範例將協助你成功的在 MATLAB 畫出一張簡單的圖。

---

**範例1:** 先練習如何在MATLAB的環境下 (命令視窗) 建立矩陣 (含 scalar, vector and matrix)。分別建立定義如下的 A,B,C,D,E,F,G,H,I,J,K 11個矩陣, 並觀察結果。

輸入方式:

```
A = [1 2; 3 4] B = [1 2 3] C = [5 6 7 8]' D = 1
E = C(1 : 2) F = C(3) G = A(1, 2) H = A(:, 1) I = A(2, :)
J = 0 : 5 K = -5 : 0.5 : 5
```

---

注意:

1. 在 MATLAB 的語法中, 上述代表矩陣的  $A, B, \dots, K$  稱為變數, 其內容 (值) 可以透過等號 (=) 右邊的資料賦予, 且隨時可以被改變。請注意等號右邊的資料可以是數值、向量或矩陣資料, 或者也可以一律視為矩陣, 因不管是數值或是向量都是矩陣的特殊型態。
2. 當變數被賦予內容後, 在命令視窗下輸入變數名稱, 將可顯示出其內容值。
3. 每個指令的最後若加上分號 (;), 則運算的結果將不會顯示在命令視窗上。
4. 請留意等號 (=) 右邊賦予矩陣資料時的符號 (, : ;), 從執行的結果去瞭解其代表的意涵, 並記錄下來。
5. 變數  $E, F, G, H, I$  是取其他變數的部分內容, 請特別留意矩陣的索引方式及結果, 對於往後 MATLAB 的使用非常關鍵。MATLAB 的矩陣索引與理論上一致, 指令  $A(i, j)$  代表第  $i$  (橫) 列, 第  $j$  (直) 行的元素, 如圖 1.1 所示。圖中  $A$  代表一個  $5 \times 5$  的矩陣, 指令  $A(2, 3)$  代表第 2 列第 3 行的值, 也

## 1. 從繪製函式圖認識 MATLAB 11

就是7。MATLAB 除提供與理論上相同的索引方式, 另可採一維的索引法, 圖中每個位置右下角的編號, 代表該矩陣一維的索引值, 所以指令 $A(12)$ 與 $A(2, 3)$ 會得到相同的結果。

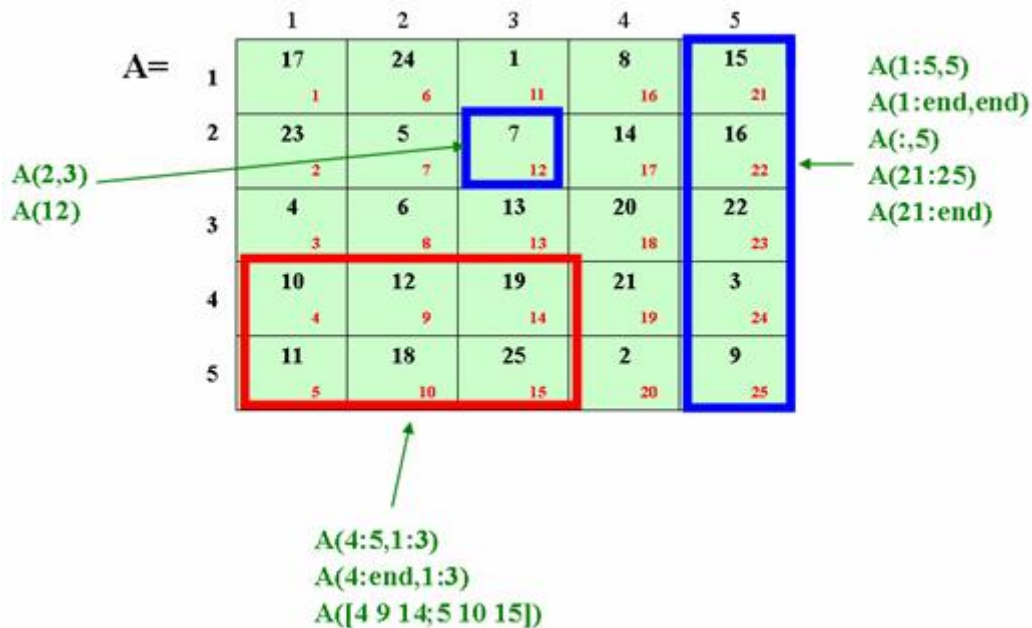


圖 1.1: MATLAB 的矩陣索引

此外, MATLAB 提供更多的索引方式, 方便擷取更多的資料, 圖 1.1 也顯示擷取第5行的5種方式, 其中保留字 'end' 代表該行或該列的最後一個索引位置, 方便下指令時不需要先知道矩陣的長度。圖中也展示如何擷取一個子矩陣。不管是從矩陣中擷取一個值、一個向量或是一個子矩陣, 初次接觸的使用者, 不妨實際在命令視窗上逐一輸入, 並觀察結果, 對照指令, 相信一定可以心領神會, 比起在此說得九彎十八拐要來得有用。練習時, 可以指令  $A = \text{magic}(5)$  來產生圖中的矩陣。

6. 利用 MATLAB 的指令 `whos` 觀察每個矩陣的大小。
7. MATLAB 也提供一些指令可以更迅速的產生特殊的矩陣, 試試看這些指令: `zeros`, `ones`, `eye`, `diag`。使用前可以利用 `help` 的指令了解使用的方式,

## 12 MATLAB 的數學實驗

譬如 help zeros。

---

範例2: Let  $A = \begin{bmatrix} 1 & 2; 3 & 4 \end{bmatrix}$ ,  $B = \begin{bmatrix} 5 & 6; 7 & 8 \end{bmatrix}$ ,  $c = 3$ , 逐一執行以下的運算練習並觀察結果。

$A + B$ ,  $A - B$ ,  $A + c$ ,  $A * B$ ,  $A/B$ ,  $A^c$ ,  $c * A$ ,  $A/c$ ,  
 $A .* B$ ,  $A ./ B$ ,  $A.^c$ ,  $A'$ ,  $B'$ ,  $(A * B)'$ ,  $B' * A'$

---

請注意 MATLAB 所使用的運算元符號, 如  $+$   $-$   $*$   $/$   $^$   $.$   $'$  等所代表的意涵。從執行的結果去觀察並記錄下來。

MATLAB 的數學四則運算與冪方大致符合學理上的矩陣運算, 但 MATLAB 也巧妙的利用矩陣式的資料結構, 設計了一個特殊的運算元 (operator) “.”, 讓多筆資料的運算可以一次完成, 譬如變數  $x, y$  各有 5 個樣本, 其值如下

$$x = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 \end{bmatrix}, \quad y = \begin{bmatrix} y_1 & y_2 & y_3 & y_4 & y_5 \end{bmatrix}$$

要計算兩變數的乘積  $xy$  的 5 樣本值, MATLAB 的指令為  $x .* y$ , 即

$$\underbrace{\begin{pmatrix} x_1 & x_2 & x_3 & x_4 & x_5 \end{pmatrix}}_x \underbrace{.* \begin{pmatrix} y_1 & y_2 & y_3 & y_4 & y_5 \end{pmatrix}}_y = \underbrace{\begin{pmatrix} x_1 y_1 & x_2 y_2 & x_3 y_3 & x_4 y_4 & x_5 y_5 \end{pmatrix}}_{x .* y}$$

一個指令同時計算好所有的樣本資料, 這有別一般正規的向量乘積, 必須注意前後向量的大小才能相乘。而 MATLAB 這種元素對元素的乘法 (element by element), 提供大量資料運算時的方便。再舉矩陣的運算為例來區分  $*$  與  $.*$  的差別:

$$\underbrace{\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}}_A * \underbrace{\begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}}_B = \underbrace{\begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{pmatrix}}_{A*B}$$

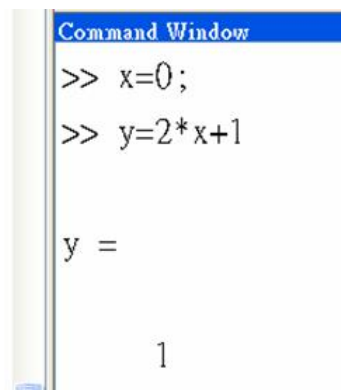
$$\underbrace{\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}}_A .* \underbrace{\begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}}_B = \underbrace{\begin{pmatrix} a_{11}b_{11} & a_{12}b_{12} \\ a_{21}b_{21} & a_{22}b_{22} \end{pmatrix}}_{A.*B}$$

除乘法外，除法（\）與乘冪（^）都可以這樣使用，不管是向量或矩陣，只要使用的時機恰當，都可以達到很好的效果。以下的內容及單元將陸續透過實際的問題介紹這個特殊運算元的好處。

**範例3:** 函式  $y = f(x) = 2x + 1$

1. 當  $x = 0, 1, 2, \dots, 10$ , 分別計算其相對的函數值  $y$ 。
2. 繪製函式圖, 其中  $0 \leq x \leq 10$

利用簡單的四則運算，函數值很容易計算出來，如圖 1.2。



```

Command Window
>> x=0;
>> y=2*x+1

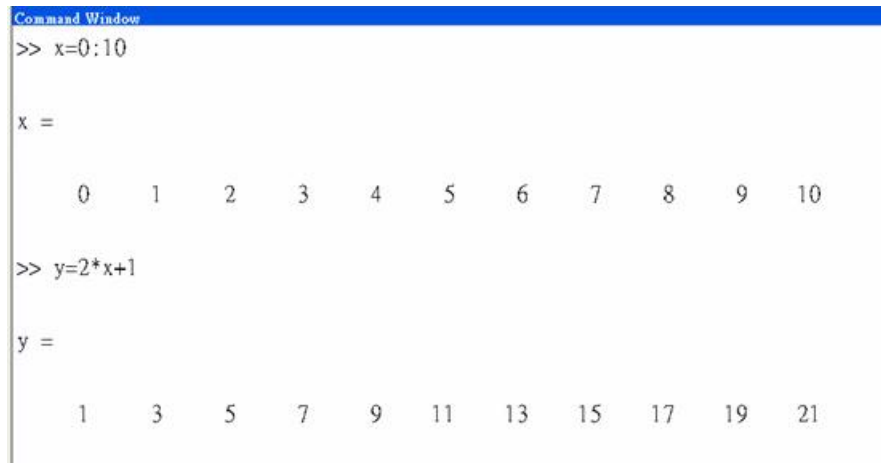
y =

     1
  
```

圖 1.2: 函數運算

## 14 MATLAB 的數學實驗

當重複執行這兩個指令並改變  $x$  值<sup>1</sup>，便可以輕鬆的計算出所有的函數值。不過每次只算一個函數值絕不是強力軟體的作為，MATLAB 利用了矩陣的運算提供方便的計算方式，如圖 1.3 所示



```
Command Window
>> x=0:10

x =

    0    1    2    3    4    5    6    7    8    9   10

>> y=2*x+1

y =

    1    3    5    7    9   11   13   15   17   19   21
```

圖 1.3: 以矩陣運算計算多個函數值。

從圖 1.3 中  $x, y$  向量很清楚的呈現出其間的函數關係，這些向量的對應數值形成繪圖時的座標點，以下說明繪製函數圖形的步驟：

1. 先依繪圖範圍產生  $x$  方向的所有點，譬如  $x = 0 : 10$
2. 再根據函式的關係計算所有相對應  $y$  方向的座標值，譬如： $y = 2 * x + 1$
3. 再利用指令 `plot` 繪製不同型態的圖形，譬如 `plot(x, y)` 將所有的座標點以直線連接。請特別注意 `plot` 提供的各種繪圖的選項（利用 `help plot` 或 `doc plot`），試著去改變描繪座標點的方式、點的符號、顏色等選項。

---

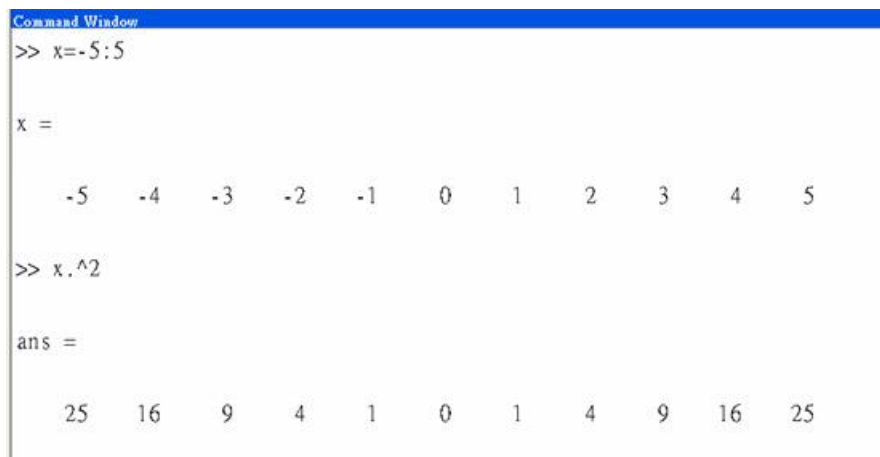
### 範例4: 繪製函式

<sup>1</sup>Tips: 由於必須不斷的帶入不同的  $x$  值，相同的指令必須不斷的被執行，如果要重複的輸入指令，將影響實驗的興致。所幸在 MATLAB 的命令視窗中可以利用鍵盤中的  $\uparrow$  鍵，不斷的將之前執行過的指令秀出來，方便實驗的進行。例如要重複圖 1.2 的指令，可以按兩次  $\uparrow$  鍵回到  $x = 0$ ，將 0 修改為 1，按「Enter」執行該指令，隨後再按兩次  $\uparrow$  鍵回到  $y = 2 * x + 1$ ，直接按「Enter」執行便可以得到答案。另外也可以先輸入欲執行指令的前幾個字，再按  $\uparrow$ ，即可快速找到該指令。

$$1. y = f(x) = x^2 + 3x + 5 \quad -5 \leq x \leq 5$$

$$2. y = f(x) = x^3 - 10x^2 + 29x - 20 \quad 0 \leq x \leq 7$$

前一個範例說明繪製函數圖形的三個步驟，缺一不可，也各有「訣竅」，否則不是畫錯就是畫得不好看，展現不出「一張圖值千字」的價值。對初學者而言，應先學會利用 MATLAB 的特點來計算函數值，特別是 MATLAB 特殊的運算元「 $\cdot$ 」。例如計算第一個函數值時，可以先試試  $x^2$  的計算方式，待結果正確了才加入其他較簡單的項次  $(3x + 5)$ ，圖 1.4 展示平方項的計算。



```

Command Window
>> x=-5:5

x =

    -5    -4    -3    -2    -1     0     1     2     3     4     5

>> x.^2

ans =

    25    16     9     4     1     0     1     4     9    16    25
  
```

圖 1.4: 函數中計算平方項次的方法。

繪圖的方式同前一題，不過須注意

1. 描繪的座標點「數」；前一個範例的函數是一條直線，當 plot 指令將座標點與點間以直線連接時，看起來毫無問題，但是當函數為曲線圖形時，座標點的距離如果不夠近，兩點間的直線連接將會造成圖形的鋸齒狀，不符合函數的特質，因此需要利用更密集的座標點來產生平滑的視覺效果。這需要從 x 軸資料的產生做起，譬如  $x = 0 : 0.1 : 7$ ，從 0 到 7 每隔 0.1 產生一個點，而點與點間 0.1 的間距是否可以造成平滑的曲線端視視覺感受而定，過小並無意義，徒增資料量與計算時間。圖 1.5 展示不同間距的視覺效果。

## 16 MATLAB 的數學實驗

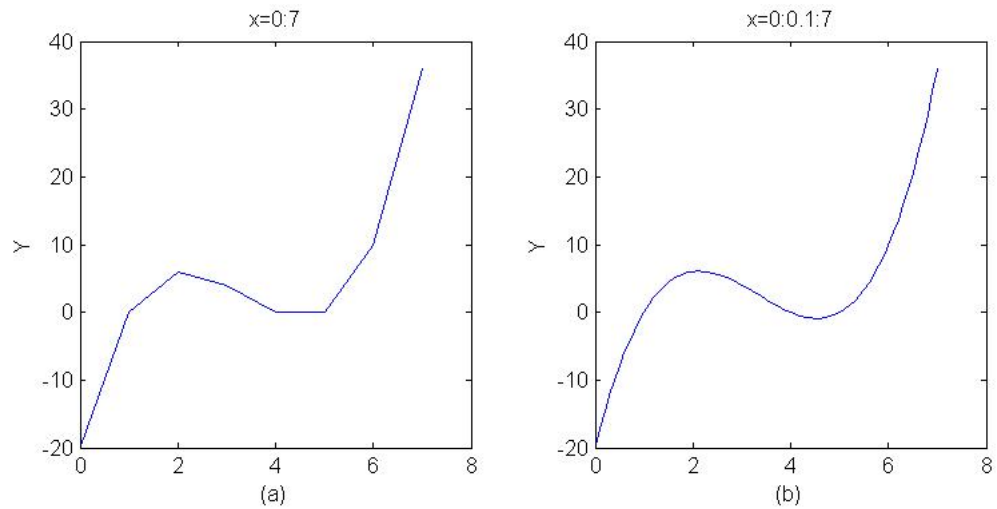


圖 1.5: 座標點的間距

2. 指令 `plot(x, y)` 中，變數  $x$  與  $y$  分別代表函數的  $X, Y$  座標點，其值的產生也不同。變數  $x$  的內容依據繪圖的範圍與座標點的間距設定一組向量，如前述的  $x = 0 : 0.1 : 7$ ，而變數  $y$  的內容則依函數的關係從變數  $x$  計算得來，本題的函數具平方與立方項，可以利用前一個範例陳述的運算元 “ $\cdot$ ” 來做乘冪，一次計算所有  $x$  座標的對應函數值  $y$ 。譬如指令： $y = x.^2 + 3 * x + 5$ ，在此  $x.^2$  的意義請參考範例 1、2 的矩陣運算。如果只是單純的使用  $x^2$ ，MATLAB 將出現錯誤的訊息，告知

```
??? Error using ==> mpower  
Matrix must be square.
```

圖 1.6: MATLAB 指令執行的錯誤訊息

表示只有方陣（含數值）才能做平方，一般向量的平方是不合法的。

3. 當函數是多項式時，MATLAB 提供一個簡便的指令 `polyval`，方便計算多項式的函數值，下列的指令可以取代之前的函數計算：



## 1. 從繪製函式圖認識 MATLAB 17

```
x=-5:0.1:5;  
p=[1 3 5];           多項式函數的係數  
y=polyval(p,x);
```

關於 polyval 的詳細使用方式，請參考線上使用手冊的說明。

為方便指令的下達與圖形變動的觀察，MATLAB 在 7.x 版以後可以將圖形視窗也整合進來，如圖 1.7 所示。圖中的視窗切分成四個子視窗，每個視窗都可以點選其右上角的「箭頭」圖示，成為獨立的視窗。子視窗的位置與配置也可以透過滑鼠「拖曳」互相交換或整併。讀者可以試著擺設所有子視窗到自己最習慣觀察的位置。另外，子視窗也可以含多個不同功能的視窗，譬如左上角的子視窗含「Current Directory」與「Workspace」兩個功能視窗。右上角的子視窗含「Array Editor」與「Figures」兩個視窗。透過滑鼠點選標頭即可選取。

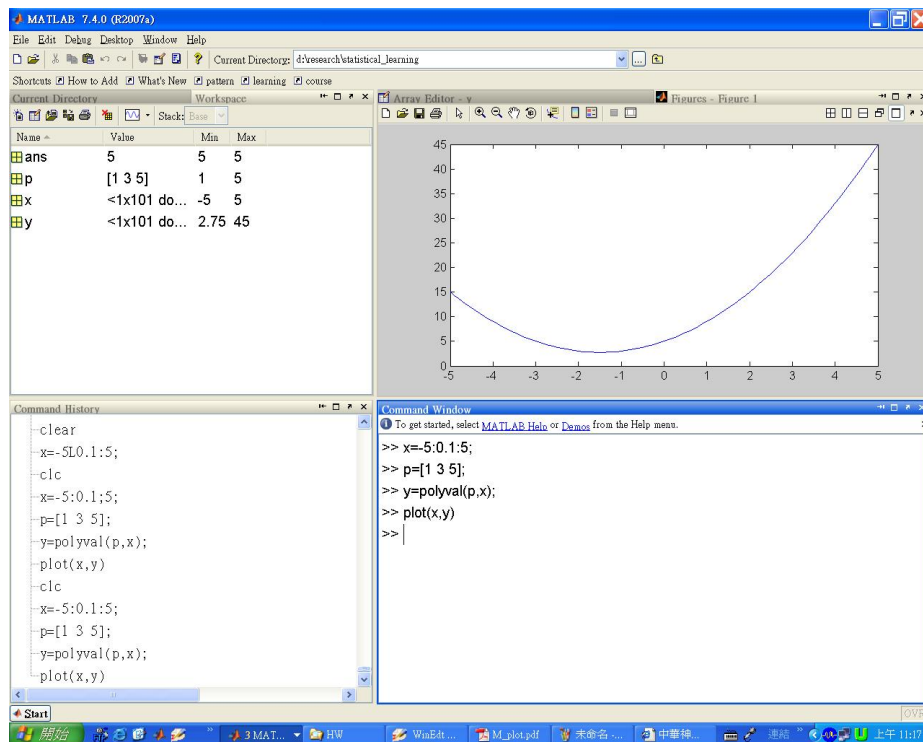


圖 1.7: MATLAB 整合視窗

範例5: 繪製函式  $y = f(x) = x^4 - 8x^3 + 16x^2 - 2x + 8$

請選擇適當的範圍, 務必看到比較完整的圖。

函數圖的繪製範圍的選取非常重要, 太寬、太窄、偏左或偏右都可能看不到該函數的全貌或特質。譬如畫出像圖 1.9 的函數圖, 便因為範圍取得太寬, 以致於看不到該函數「最有價值」的最小值部分, 因此繪圖時必須不斷的調整範圍, 以突顯出函數最值得觀察的部分。

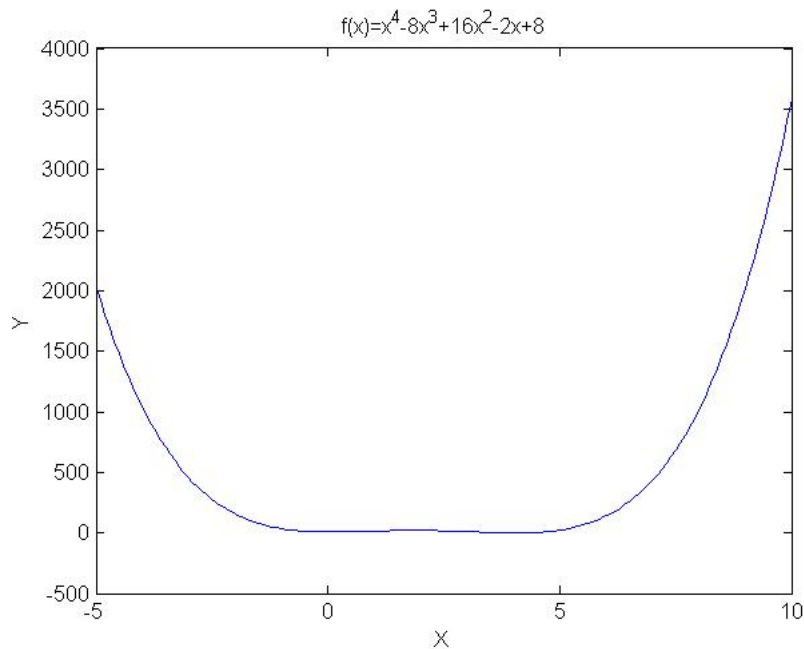


圖 1.8: 函數圖範圍的決定

範例6: 依下列範圍繪製函式  $f(x) = x + \sin(x)$

(1)  $1 \leq x \leq 1000$  (2)  $1 \leq x \leq 500$  (3)  $1 \leq x \leq 100$  (4)  $1 \leq x \leq 10$

這個範例也是說明範圍選擇的重要, 透過不斷的縮小範圍, 才能看出函數的特性。

## 1.2 觀察

1. 前面的範例僅提及如何將資料放入一個變數中，或是如何從一個資料變數中選取部分範圍。由時候應用上卻需要將資料變數的內容變小，譬如，剔除一向量的某一項或某幾項資料，或是剔除一矩陣的某一行（或列）資料，示範如下

```
a=[1 2 3 4 5];
a(3)=[ ];
此時向量 a=[1 2 4 5];
a([1 3])=[ ];
此時向量 a=[2 5];
*****
A=[1 2 3;4 5 6;7 8 9];
A(2,:)= [ ];
此時矩陣 A=

$$\begin{bmatrix} 1 & 2 & 3 \\ 7 & 8 & 9 \end{bmatrix}$$

```

2. 電腦繪圖係以描點的方式畫上去，因此點與點間的距離越近，所呈現出來的圖形越平滑。請觀察在一定的範圍內，點的數量多與少的效果，也就是調整  $x$  的間距。
3. 利用 MATLAB 指令計算所畫上去的點數（即觀察向量變數  $x$  的長度）。譬如 `length`, `size` 等指令。
4. 改變 `plot` 指令的參數，讓畫上去的點做不同的表現，譬如，畫出散佈圖。利用 `help plot` 指令，看看有那些繪圖的技巧可以使用。
5.  $x$  軸的範圍取捨非常重要，除了給定的範圍外，請試試看其他範圍。寬一點或窄一點。

6. 為這些圖加上些裝飾, 如  $x$ ,  $y$  軸的文字、標題或是格線等等, e.g. `xlabel`, `ylabel`, `title`, `grid`。例如圖 1.9 展示 `title` 的使用, 其指令如下

```
title('f(x) = x^4 - 8x^3 + 16x^2 - 2x + 8')
```

請注意 `title` 所顯示的是文字而非函數的計算, 因此與 MATLAB 的運算元無關, 純粹是文字的編輯, 譬如符號  $\wedge$  代表文字的上標。相關的用法可以查閱線上使用手冊。

7. 對一個陌生的函數繪圖, 常要花很長的時間才摸索到適當的範圍, 這時可以使用 MATLAB 指令 `ezplot`, 迅速的得到圖形。做法如下

```
ezplot('x^4 - 8 * x^3 + 16 * x^2 - 2 * x + 8')
```

注意, 引號內的函數輸入方式不同於前面的陳述, 不需要將  $x$  當作向量看待。 `ezplot` 雖然迅速方便, 但畢竟是「電腦選的」範圍, 有時候還是看不到函數的細節。不過不失為一個首先嘗試的指令。 `ezplot` 內放置函數的方式也可以採用內建函數的方式, 譬如上述的指令可以改為

```
ezplot('polyval([1 - 8 16 - 2 8], x)')
```

MATLAB 提供許多常用函數 (如機率密度函數) 的指令, 用 `ezplot` 來繪製相當方便。

8. 練習將結果 (圖) 貼到 WORD 或其他文書編輯工具 (如 Latex 或中文的 `cwTex`) 裡面, 做好如何呈現結果的工作。將 MATLAB 產生的圖形貼到其他地方的方式有兩種; (1) 利用 `File/Export`(7.0 版以前) 或 `Save As`(7.0 版以後) 的方式將圖形以檔案的方式儲存下來。(2) 利用 `Edit/Copy Figure` 的方式將圖形 `copy` 下來, 再到 WORD 或其他類似的軟體, 直接以 `paste` 的方式貼回。

### 1.3 作業

畫出下列函式,  $x$  範圍自訂, 盡量畫出比較完整的函式圖, 並加上必要的裝飾文字。  
未曾使用過的指令, 請自行利用 `help` 或 `doc` 該指令的方式察看其使用說明。

1.  $y = f(x) = \sin(x) + \cos(x)$ , 代表  $\sin, \cos$  的指令分別是 *sin, cos*

2.  $y = f(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}$  代表指數的指令 `exp`

3.  $y = f(x) = \sqrt[3]{\frac{4 - x^3}{1 + x^2}}$

開根號的方式有兩種, 譬如計算  $\sqrt{9}$  的指令可以是: (1) 以冪方的方式  $9^{0.5}$ ,  
(2) 採開平方根的指令 `sqrt(9)`, (3) 採開實數根的指令 `nthroot(9, 2)`。<sup>2</sup>

第 1 與第 3 個方式均適用於本題函數, 當使用冪方時須特別注意 MATLAB 冪方指令的下達方式, 譬如指令  $-8^{(1/3)}$  及  $(-8)^{(1/3)}$  會產生不同的結果, 何者才是正確的, 從結果很容易辨識, 但卻與一般常理的認知相左, 因此建議除平方根外, 還是採 `nthroot` 指令比較安全, 否則一旦錯誤將很難發現。

4.  $y = f(x) = \frac{1}{x - 1}$

5.  $y = f(x) = \frac{1}{2\sqrt{2\pi}} e^{\frac{-(x-1)^2}{8}}$   $\pi$  的指令為 `pi`

6.  $y = f(x) = x^{2/3} = \sqrt[3]{x^2}$

7.  $y = f(x) = 2x^3 - x^4$

8.  $y = f(x) = x\sqrt{4 - x^2}$

9.  $y = f(x) = \frac{\ln x}{x^3}$

10.  $y = f(x) = 3, 1 \leq x \leq 5$

11.  $x^2 + y^2 = 1$

---

<sup>2</sup>這個指令在 7.x 版之後才有。

**補充：**下面的指令可以改善圖形呈現的品質

```
set(gca,'xtick',0:1:5)
```

這個指令可以將 x 軸在 0 到 5 之間的刻度間隔改為 1。當然將 'xtick' 改為 'ytick' 就可以針對 Y。

如果想將線條變粗，使用

```
plot(x,y,'LineWidth',2)
```

裡面的 2 代表線條得寬度，數字越大越寬。如果還想配上顏色，可以這樣做

```
plot(x,y,'LineWidth',2,'color','red')
```

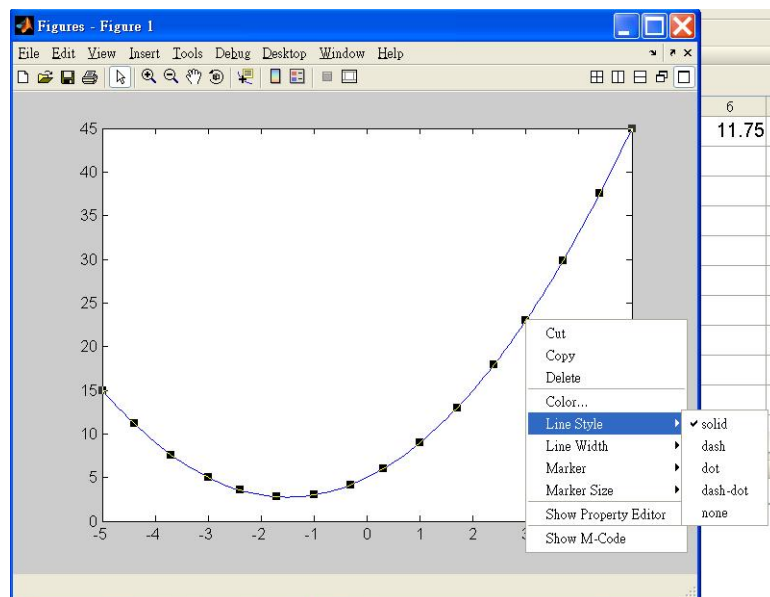


圖 1.9: 圖形編輯器

MATLAB 的繪圖指令非常豐富，變化很多，可以從線上使用手冊查詢到。不過對初學者而言，實無須浪費過多時間在這方面，看到一個學一個，夠用就好，不急於一時，隨著時間的累積，將會非常可觀。另一個簡便的方式是直接在繪圖區作圖形的編輯，如圖 1.9，步驟如下：

1. 點選功能選單「Tools → Edit Plot」或直接以滑鼠選取左上方的「箭頭」工具圖示。
2. 選取欲編輯的物件（線條、符號、或是區域，）按滑鼠右鍵或雙擊左鍵拉出編輯盤。
3. 選取編輯項目，進行修改。

利用圖形編輯器的好處是，不需要知道指令，憑著編輯盤的選項即可隨意改變圖形。但缺點是動作多，速度慢，僅適合一次或不常用的編輯需求。有一個折衷的辦法既可以符合編輯上的需求，也可以趕上指令編輯的速度。在圖形視窗的選單上找到「File → Generate M-File」，要求 MATLAB 將目前的圖形轉換成指令。圖 1.10 中橢圓形圈出的部分即畫出圖中線條的指令。複製這個指令就可以立即產生相同的線條。

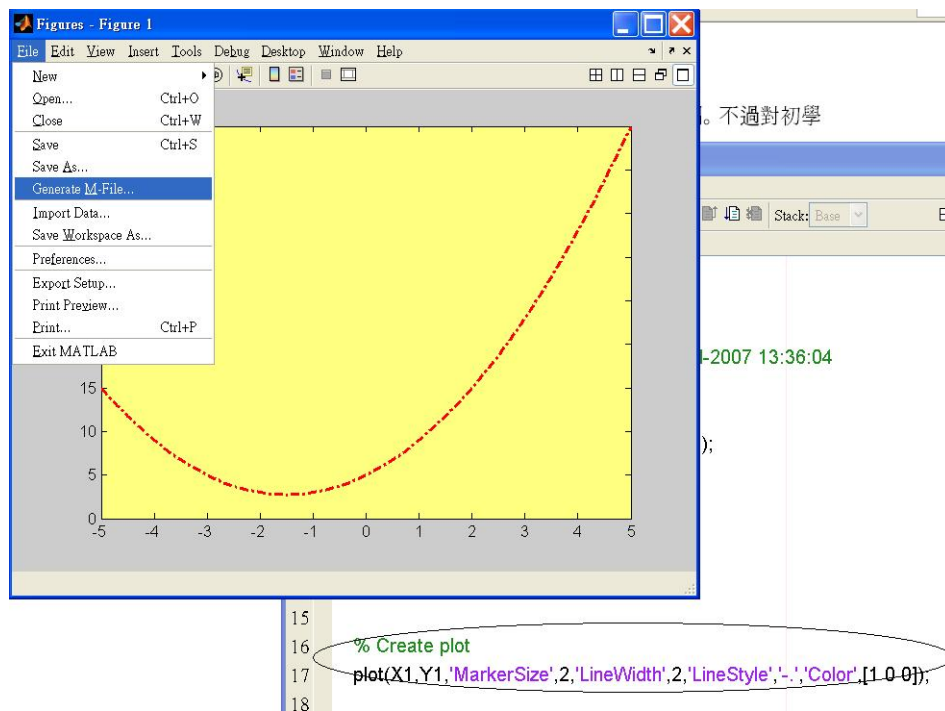


圖 1.10: 圖形編輯器產生指令的功能

另外，如果想在 MATLAB 所做的圖上放上比較複雜的數學式，可以套用  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$

的指令，作法如圖 1.11 所示，結果如圖 1.12。圖 1.11 的 `text` 指令中的「...」用在指令太長時作為斷句使用，兩個「\$」號中間放入一般  $\text{\LaTeX}$  的數式指令。令外一個比較輕便的做法如下，其中「...」的部分填入  $\text{\LaTeX}$  的數式指令。

```
gtext('$ ... $','interpreter','latex')
```

```
>>
>> text('interpreter','latex',...
        'String','$\frac{1}{\sqrt{2\pi}}e^{\frac{-x^2}{2}}$',...
        'Position',[2 .35],...
        'FontSize',12)
```

圖 1.11:  $\text{\LaTeX}$  指令的連結。

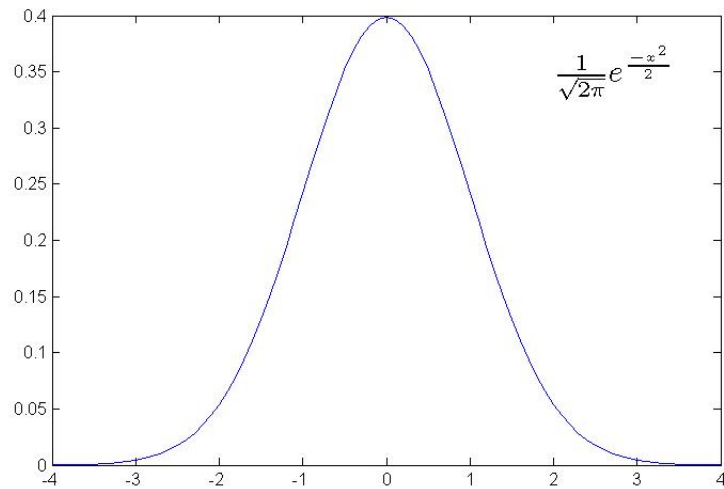


圖 1.12:  $\text{\LaTeX}$  指令連結範例。



## 第 2 章

# MATLAB 的數學運算

解決數學問題常需直接面對數學公式，基礎的訓練是以紙筆解題。本章做為以電腦解題的開端，了解 MATLAB 語言處理數學公式計算的原理，並藉此進一步熟練 MATLAB 的矩陣表示法及其計算方式與技巧，提供學生學習在面對數學公式時，如何利用電腦語言有效率的去計算公式的值。試著從對電腦語言的探索，去發覺解決問題的方法及其差異性。

**本章將學到關於程式設計**

變數的運用、MATLAB 計算公式的技巧與變化。

〈本章關於 MATLAB 的指令與語法〉

操作元 (operators): .\*

指令: sum, mean, var, std, length, zscore, scatter, corr, corrcoef, load, axis, xlim, ylim, xlsread, importdata

語法: 資料檔的讀取: 檔案格式 MAT, TXT, XLS

## 2.1 練習

假設量測自變數  $X$  與應變數  $Y$  所得的資料為

X	6	3	5	8	7
Y	2	4	3	7	6

資料準備：依前一個練習的方式，在 MATLAB 的命令視窗裡將這些數值建立成  $x, y$  矩陣 (向量)，如 <sup>1</sup>

$$x = [6 \ 3 \ 5 \ 8 \ 7]$$

---

**範例1:** 依序利用下列的 MATLAB 指令，計算變數  $X$  的 sample mean:

$$\mu_x = \frac{\sum_{k=1}^N x_k}{N}, \text{ where } N = 5$$

1.  $\mu = (x(1) + x(2) + x(3) + x(4) + x(5))/5$
2.  $\mu = \text{sum}(x)/5$
3.  $\mu = \text{mean}(x)$

---

其中  $\text{sum}$ ,  $\text{mean}$  為 MATLAB 提供的指令，使用前請先利用 `help` 瞭解其使用方式。請注意分母的 5 代表樣本數  $N$ ，在程式設計的技巧上，會先設定一變數，譬如  $N = 5$ ，代表樣本數，之後的指令便以  $N$  取代固定的常數 5。當樣本數很大時，則可以利用 MATLAB 指令直接計算向量的長度，其指令如下

---

<sup>1</sup>請注意程式設計中變數的命名不可輕忽，最好能賦予有意義且相關的名稱，有利程式的可讀性，在未來除錯或維護上都會很方便，無形中節省許多不必要的時間浪費。在此以小寫的  $x$  代表變數  $X$  的樣本資料，使用小寫是確知這是個向量不是矩陣，來符合以小寫字母代表向量或常數，大寫代表矩陣的慣用語法。當然習慣隨人，只要保持一致即可。

```
N=length(x)
```

**範例2:** 依序利用下列的MATLAB 指令, 計算變數 X 的 sample variance:

$$S_x^2 = \frac{\sum_{k=1}^N (x_k - \mu_x)^2}{N - 1}$$

1. 模仿範例1之1的作法, 非常繁瑣, 但請耐心的完成, 其中 $\mu_x$ 可以直接利用範例1的結果, 即以變數 *mu* 代表  $\mu_x$  的值。

2. 模仿範例1之2的作法, 利用向量的運算並加入 sum 指令以減少繁雜程度, 即

$$\sum_{k=1}^N x_k^2 = \text{sum}(x .* x)$$

3. 利用矩陣運算的技巧, 將平方與累加合併計算, 進一步精簡指令, 即

$$\sum_{k=1}^N x_k^2 = x * x'$$

4. 利用 MATLAB 內建函式 var。

上述不同的過程應該得到相同的結果, 只是繁瑣與精簡之差別。程式設計當然力求精簡, 除便利可讀性外, 往往效率也會比較高。本練習的目的在讓初學者體驗 MATLAB 矩陣運算的效率與技巧, 當數學式本身非常繁雜時會需要這些技巧。初學者應具耐心細心的輸入與比較才能漸得 MATLAB 程式設計之精髓。

**範例3:** 計算每一個 $x$ 值的 z-score:  $z = \frac{x - \mu_x}{S_x}$

參考指令: zscore

**範例4:** 計算  $S_{xy} = \sum_{k=1}^N (x - \mu_x)(y - \mu_y)$ , 你可以用多少種方法來計算這個值

呢？哪一種方法最乾淨俐落？

## 2.2 觀察

1. 觀察常數在矩陣運算中的角色與結果。
2. 觀察累加 $\Sigma$ 與矩陣的關係，並且想像上述的問題如果資料量大的時候，計算上，表達上有何困擾？這是為什麼 MATLAB 強調避免直接使用加法，而以矩陣的運算取代。
3. 變數的命名必須非常謹慎小心，不可與 MATLAB 內建的指令相同，如此將混淆該指令的功能，使之失效。譬如範例2的 sample variance，一般直接會取用 `var` 當作變數名稱，但卻與 MATLAB 內建的指令相同，會使隨後使用的 `var` 功能失效。

## 2.3 作業

兩變數間相關性的觀察與相關係數的計算（注意繪圖時  $x, y$  軸的 scale）

1. 寫下上述範例4的各種計算方式，每一種方式都盡量用一條指令完成。
2. 根據網頁上提供的資料 (<http://web.ntpu.edu.tw/~ccw/statmath/book.htm>):  
資料1,
  - 先畫出  $x, y$  的散佈圖並以目視約略判斷  $x, y$  的相關性。
  - 計算變數  $x, y$  的相關係數 (the coefficient of correlation):

$$r = \frac{S_{xy}}{\sqrt{S_{xx}S_{yy}}}$$

其中 $S_{xx}$ 及 $S_{yy}$ 分別是變數  $x$  與  $y$  的變異，與之前練習所計算的變異數關係為 $S_{xx} = (N - 1)S_x^2$ ,  $S_{yy} = (N - 1)S_y^2$ 。觀察這個  $r$  值與散佈圖呈現的樣子是否吻合？

- MATLAB 也提供了計算相關係數的指令, 例如 `corr`、`corrcoef`, 請自行了解其使用方式, 最後並驗證自己的答案。
  - 利用指令 `text` 將  $r$  值寫在散佈圖上 (如圖2.1), 並寫下計算  $r$  值的所有指令。
  - MATLAB 也提供簡便的散佈圖指令, `scatter`, 讀者不妨參考線上手冊的說明, 自行找出使用的方式。
3. 同2, 但使用資料2。觀察這個  $r$  值與散佈圖呈現的樣子是否吻合?
4. 同2, 但使用資料3。觀察這個  $r$  值與散佈圖呈現的樣子是否吻合?

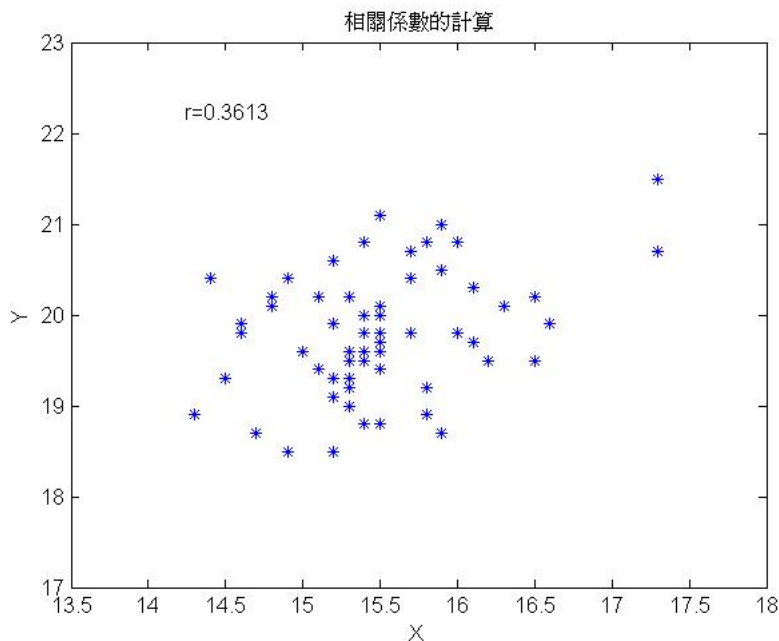


圖 2.1: 散佈圖與相關係數的計算

**補充1:**

作業所採用的資料來自檔案, 如 `data1.txt`。在 MATLAB 中讀取檔案的方式為:

```
load data1.txt
```

### 30 MATLAB 的數學實驗

從 MATLAB 讀取外部資料檔時，必須注意兩件事：

1. 資料檔案的位置：當開啓 MATLAB 軟體時，MATLAB 主動設定其工作區域在預設的目錄（隨版本與安裝的目錄相關），執行上述指令時，資料檔案所在的位置必須與工作目錄相同。如果不同，作法有二（1）將檔案轉移至預設的工作目錄（2）更改工作目錄至檔案所在位置。第二種方式比較常見，更改目錄的作法如圖 2.2 所示。

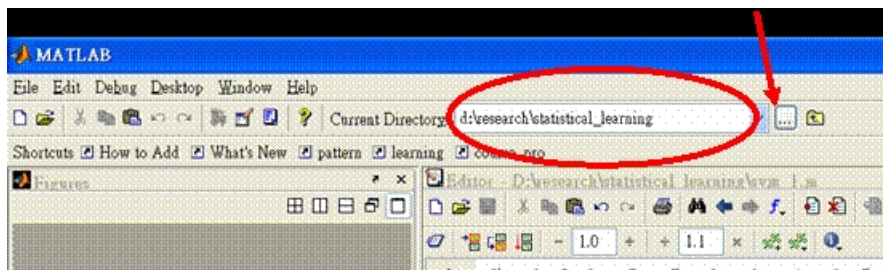


圖 2.2: 更改工作區域目錄

2. 檔案的格式：MATLAB 支援幾種資料檔案格式，常見的有文字檔 (\*.txt) 及 EXCEL 檔，另一種為 MATLAB 專用的 binary 檔 (\*.mat)，本章暫不討論。其中文字檔需以類似 EXCEL 的行列方式編輯，讀入後將被視為矩陣的行列。

當使用 `whos` 指令察看命令視窗的變數時，將發現新的變數 `data1`，代表下載的資料檔案 `data1.txt` 的內容，其大小為  $N \times 2$ ，其中第一行代表變數  $X$ ，第二列代表變數  $Y$ 。通常資料讀入後，會依實際情況將內容指定給特定變數，方便後續的運算，譬如

```
x=data1(:,1):  
y=data1(:,2):
```

檔名直接轉為變數名稱是 MATLAB 內定的作法，如果不喜歡或嫌麻煩，當然可以換，譬如，`Y=data1`。不過也可以一氣呵成，如，

```
Y=load('data1.txt');
```

讀取 EXCEL 檔的指令與 TXT 檔不同, 如 (先下載資料檔 data1.xls)

```
Y=xlsread('data1');
```

另外, 指令 `importdata` 可以讀取多種不同型態的檔案, 譬如, TXT, MAT, XLS, WAV, AU, JPG...。做法如下

```
Y=importdata('data1.xls');
x=Y.data(:,1);
y=Y.data(:,2);
```

對大部分檔案格式而言, MATLAB 會以一個結構式的變數存放資料及附加的訊息。結構式的變數是一種多層結構的變數, 第一層是變數名稱, 第二層才開始存放資料或其他相關訊息。取用的方式如上第二行, 以 `'.'` 的方式連結兩層變數名稱。使用前可以先在命令視窗將變數叫出來看看。詳細的用法無法在此一一說明, 有興趣的讀者不妨到線上手冊察看, 然後試讀不同的檔案, 相信很快便能上手。

#### 補充2:

作業中要求將計算出來的  $r$  值寫在散佈圖上, 這需要用到指令 `text` 或 `gtext`, 其差別在 `text` 必須指定  $(x,y)$  座標, 而 `gtext` 則可以使用滑鼠選定適當的位置, 詳細的使用方式請自行參考使用手冊。勤用 `help` 是必要的, 畢竟每個指令都有不同的參數給定方式, 簡單的 `help` 一下, 立刻便能查知, 無須記憶。

#### 補充3:

散佈圖的觀察會隨著  $X, Y$  軸的座標範圍而在目視上有所不同, 適當的 `zoom-in` 或 `zoom-out` 可以讓圖形的表現更貼切, 如圖2.1並非畫出來就長這樣。畫完圖之後座標軸可以利用指令 `axis` 加以改變, 如

```
axis([13.5 18 17 23])
```

這四個數字的意義不難從圖中窺知, 但詳細的介紹與使用方式請利用線上使用手冊或輸入 `help axis` 指令。如果只想改變  $X$  或  $Y$  軸的範圍, 則可以使用指令 `xlim` 或 `ylim`, 使用前請參考手冊上的說明。

---





## 第 3 章

# MATLAB 的機率分配

機率相關的問題對於統計學門不僅是基礎的理論,更是應用上不可或缺的工具。對多數初學機率的學生而言,機率是有點抽象的,需要帶點想像的,學習的障礙不小,不過有了電腦工具(如 MATLAB)的輔助,那些想像與抽象的部分會變的比較具體些。本章旨在熟悉 MATLAB 處理機率問題的指令及其應用,包括分配函數的繪製與亂數的產生。

### 本章將學到關於程式設計

圖形的表現技巧與變化、線上使用手冊的運用。

#### 〈本章關於 MATLAB 的指令與語法〉

指令:hold, normpdf, binopdf, stem, stairs, bar, normrnd, hist, subplot, copularnd, normplot, qqplot, cdfplot, ecdf, normspec, boxplot, type

### 3.1 練習

各種分配函數的「長相」最好能深植腦海，當面對不同的資料時，才能迅速的找到對應的母體。MATLAB提供哪些 pdf 及 cdf 函式呢？你可不可從 MATLAB 提供的查詢功能查到呢？你必須練習找找看。

---

**範例1:** 先來畫常態分配 $N(\mu, \sigma^2)$  的 pdf 及 cdf 圖，其中的參數 $\mu = 0, \sigma = 1$ 。請注意調整 x 軸的範圍，方便看到最完整的分配圖形。計算常態分配的機率密度函數的指令為 `normpdf`，請自行以 `help normpdf` 或 `doc normpdf` 查詢其使用方式。其他分配的相關指令也可以在使用手冊中查詢到。

- 練習給予不同的 $\mu$ 及 $\sigma$ 值，看看圖形的變化。
- 固定 $\mu$ 值，改變 $\sigma$ 值，練習將每一張圖都疊在畫面上（如圖3.1所示）。
- 試試看其他的分配？找出相對的指令來，把圖畫出來，並使用不同的參數。

---

請注意，只要是繪圖就是描點法，要將每一個點的 x, y 值都事先給定或計算好。MATLAB 中 pdf 及 cdf 函式都是用來計算 y 值的。另外本練習為觀察參數對於一個分配函數的影響，要求將不同參數的分配圖疊在一起，方便觀察。MATLAB 利用 `hold on` 指令將目前的圖形保留，隨後畫上的圖會直接疊在上面，再利用 `hold off` 取消保留的設定。譬如將三張圖疊在一起的作法：

```
plot(x1,y1)
hold on
plot(x2,y2)
plot(x3,y3)
hold off
```

另外，不連續型分配（如二項分配）的繪製要特別謹慎，要注意 x 軸範圍的限制與間距的特性，不能以一般繪製連續函數的方式大刺刺的一筆帶過，畫出來的圖必

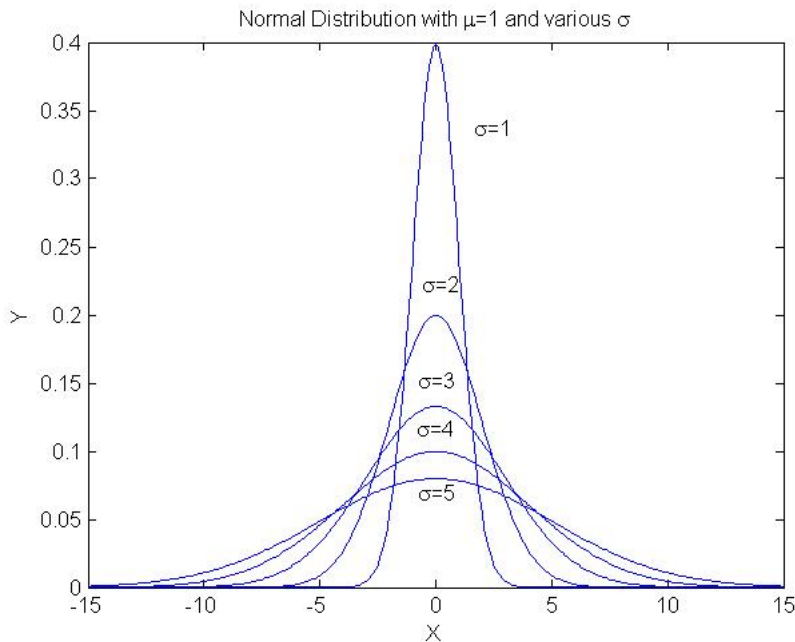


圖 3.1: 常態分配的機率密度函數 (pdf)

須合乎學理，不是畫出圖來便是對的。以二項分配為例， $x$  座標的選定與母體的選擇息息相關，假設母體為  $B(20, 0.2)$ ，其 pdf 的繪製可以這樣做

```
x = 0 : 20;
y = binopdf(x, 20, 0.2);
stem(x, y)
```

其結果如圖3.2(a) 所示， $x$  的範圍與間距準確的呈現出母體的所有可能性，太大的範圍沒意義 (如  $x = 0 : 50$ )，太小的範圍 (如  $x = 0 : 15$ ) 與非整數的間距 (如  $x = 0 : 0.1 : 20$ ) 則是對二項分配的不瞭解，都應該很警覺的避免。MATLAB指令 `stem` 可以畫出漂亮的間距圖形，也有些變化可以應用，如

```
stem(x, y, 'filled')
```

試看看畫出什麼不一樣的圖形。`stem` 不是唯一的選擇，圖3.2(b) 利用寬度比較窄的長條圖 `bar`，也可以畫出類似的效果，甚至變化更豐富，譬如指令的第三個參數用來指定長條的寬度

```
bar(x,y,0.5,'r')
```

其中第四個參數決定顏色。至於 cdf 圖，可以採用 stairs 的指令，畫出如階梯般的機率累積圖。stairs指令怎麼用呢？其實繪圖指令用多了，猜猜看往往八九不離十，學習程式語言要有此本事才會越學越輕鬆愉快。

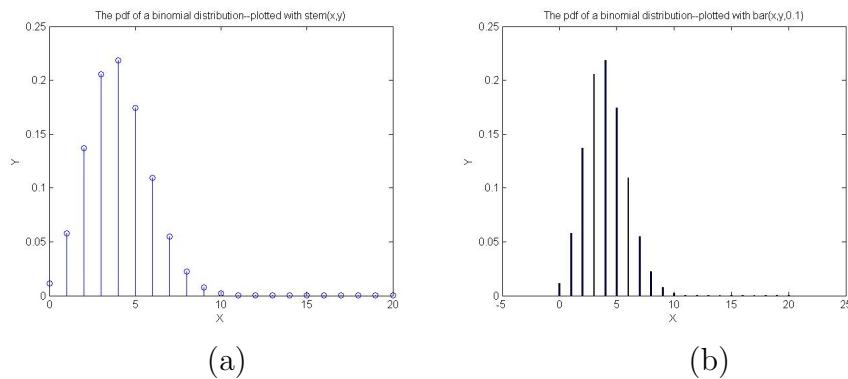


圖 3.2: 二項分配的機率密度函數 (pdf), 以指令 (a)stem(b)bar 繪製的不連續函數圖。

**範例2:** 如何從MATLAB 產生亂數 (random numbers)? 照例可以從 Help 裡面查到亂數相關的指令及其使用方式。其指令參數的給定不外乎是該分配的參數與欲產生亂數的個數。譬如下列指令產生  $M \times N$  個具常態分配的亂數 (樣本), 放在矩陣  $A$  裡:

```
A=normrnd(mu,sigma,M,N);
```

第三及第四個參數決定亂數的個數與排列的方式 (依實際需要排成矩陣或向量), 不妨多試幾個不同的數字便一目了然。

- 隨意產生 100 個具常態分配的亂數。你如何確定這些亂數值具備常態分配的特性？用 plot 畫出這些值，看看長什麼樣子？再試試用直方圖去畫。

```
x=normrnd(0,1,1,100);%標準常態
hist(x)
```

- 重複上個練習，但試著改變所產生的亂數個數（變多或少），與不同的分配。觀察畫出來的直方圖有何不同？跟你原本的認知有什麼差別？

直方圖常用來觀察資料的分佈情形（頻率、落點分佈），MATLAB 對應的指令為 `hist`。繪製直方圖需要很有「感覺」，否則容易畫出連自己都不易看懂的圖，其中樣本數的多寡牽涉到直方圖選定的組界數（bins），尤其困擾初學者，不妨多練習、不斷的變更組界數並觀察圖形的變化，在 `hist` 指令中，組界數在第二個參數，不指定時組界數內定為 10。圖 3.3 展示不同組界範圍的視覺效果。

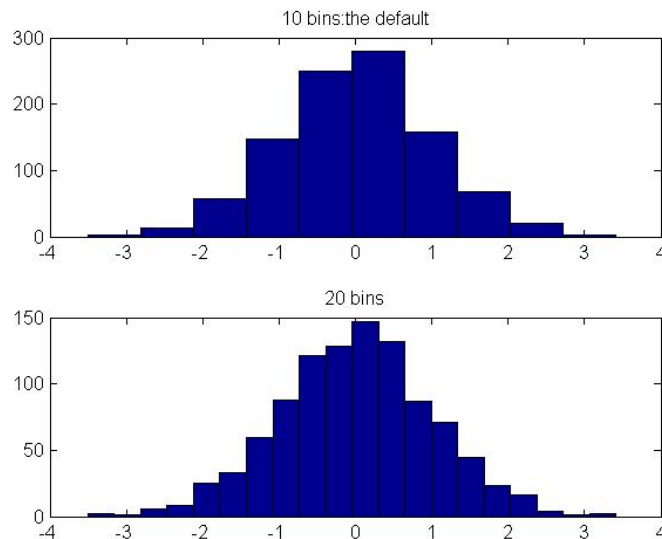


圖 3.3: 相同的 100 個樣本畫出不同組界數的直方圖，上圖 `hist(x)`，下圖 `hist(x,20)`

從直方圖只能大概判斷樣本是否來自常態的母體，若能為直方圖疊上接近的常態分配的機率密度圖，可以提供更精準的觀察。`histfit` 指令提供這樣的功能，

```
x=normrnd(0,1,1,1000);
histfit(x)
```

結果如圖 3.4 左圖所示。而右圖則展示 MATLAB 指令 `cdfplot` 畫出從樣本資料計算的 Empirical CDF (直線部分)，虛線是理論的 CDF 函數圖。兩者幾乎重

疊，幾可判斷樣本資料來自常態分配。cdfplot 指令若無法從 help 指令得到其使用方式，表示使用的是早期的版本，屬於未公開的指令，但仍可透過在命令視窗輸入 type cdfplot 看到程式的原貌及並從裡面的結構與說明探知其使用方式與限制。另外一個繪製 Empirical CDF 的指令是 ecdf，這個指令還可以取得一組累積機率值與對應的 X 值，這一組資料可以用來以 X 值查詢該分配的累積機率值。

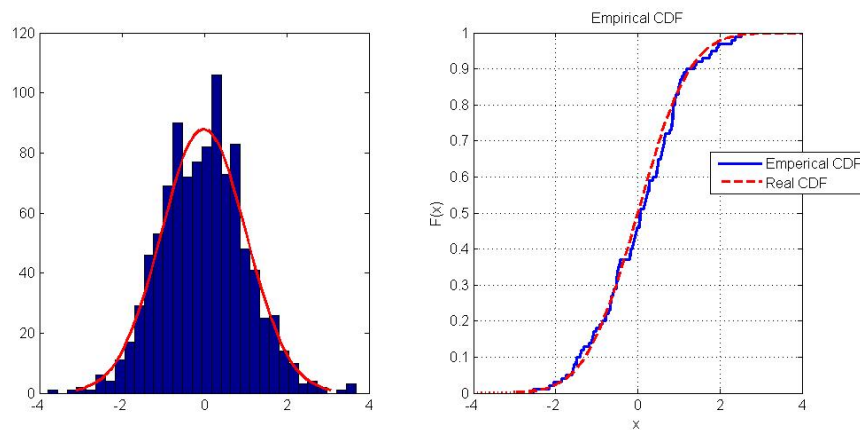


圖 3.4: 直方圖配上估計的常態分配。

判定樣本是否來自常態分配的母體，還可以使用指令 normplot 繪製常態機率圖 (Normal Probability Plot)，方式如

```
x=normrnd(0,1,1,100);
normplot(x)
```

圖 3.5 顯示常態機率圖。當圖中的 '+' 號越趨近線性 (即與虛線越貼近) 時，表示樣本來自常態的可能性越高。其繪圖原理請參考手冊上的說明。讀者不妨試試其他分配的樣本，觀察 '+' 的分佈情況。其他指令如 qqplot 也提供類似的功能，做法也相同。

### 範例3:

- 產生兩組資料，每組各 100 個樣本。這兩組資料來自兩個不相關的變數 (uncorrelated variables)  $y_1$  及  $y_2$  (可以先假設變數都具常態分配)。畫一個散佈圖來呈現資料間的不相關性  $y_1$  vs.  $y_2$ 。並且計算出兩者間的相關係數  $r$ 。

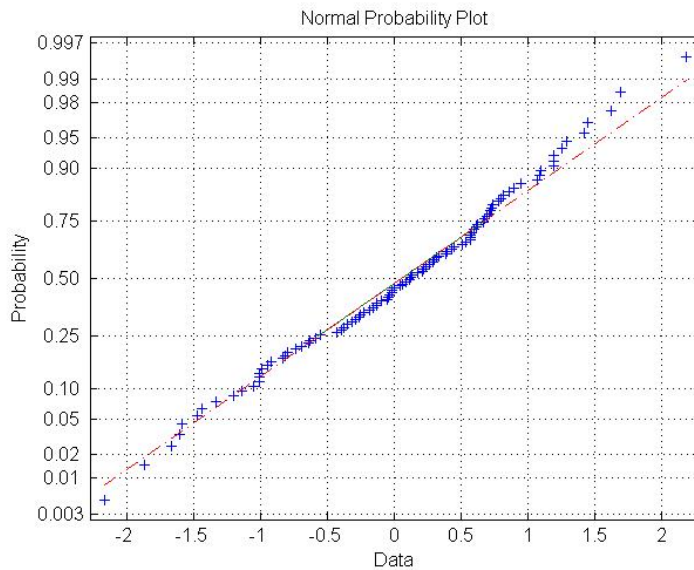


圖 3.5: 常態機率圖

- 同上, 但資料來自兩個完全相關 (completely correlated) 的變數並觀察相關係數, 例如:  $y_1 = cy_2$ ,  $c$ 代表一個常數。
- 同上, 但資料來自兩個部分相關 (partially correlated) 的變數並觀察相關係數。如何模擬「部分相關」的兩組資料呢? 動動腦筋!

本範例假設學習者對於兩變數間的相關性及其散佈圖的模樣已經有相當的概念, 方能藉此逐步模擬出自己所認知的相關性, 特別是模擬部分相關的資料。此外, 當畫出的幾張圖形具有比較意義時, 可以利用 MATLAB 提供的畫面切割技術, 將幾個圖分別呈現在畫面的不同位置, 這個指令是 `subplot`。用來作為繪圖前位置的指定, 使用方式如下 (圖 3.3展示了下列切圖的方式)

```
subplot(2,1,1),plot(x1,y1)
subplot(2,1,2),plot(x2,y2)
```

至於 `subplot` 的參數代表的意義, 在命令視窗上輸入 `help subplot`, 即可一目了然, 當然親自動手畫圖的時候便可以領會, 無須在此贅述。寫得太仔細會把學的人

教笨了, 豈不罪過! 圖 3.6 展示 subplot 的  $2 \times 2$  切圖方式及不同相關性變數的散佈圖, 其中的  $\rho$  代表相關係數。

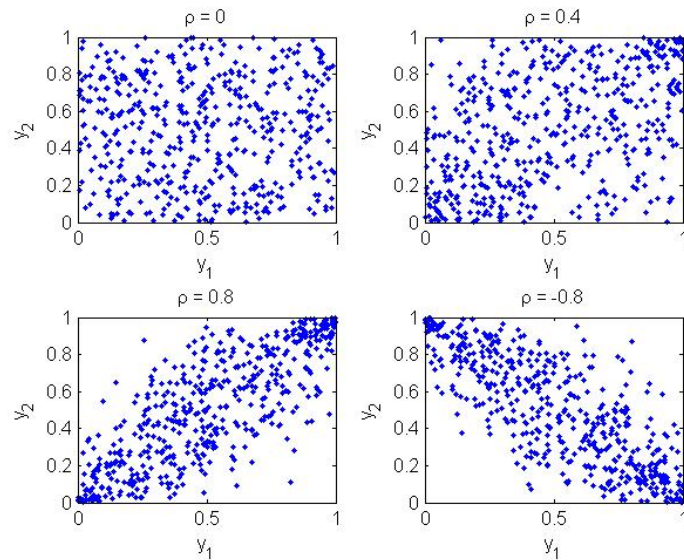


圖 3.6: subplot 的  $2 \times 2$  切圖方式及不同相關性變數的散佈圖。

### 3.2 觀察

1. 本單元繪製機率密度函數時, 也可以嘗試使用 `ezplot` 指令, 譬如繪製標準常態分配的 pdf 圖,

```
ezplot('normpdf(x, 0, 1)')
```

建議經常使用, 相當方便。不過當需要疊圖時, `ezplot` 似乎不太好搞, 不妨試試看。

2. MATLAB 關於機率方面的指令不少, `normspec` 也是一個常用來表達機率概念的圖, 其使用方式如下, 結果如圖 3.7。



```
normspec([-5 5],0,3)
```

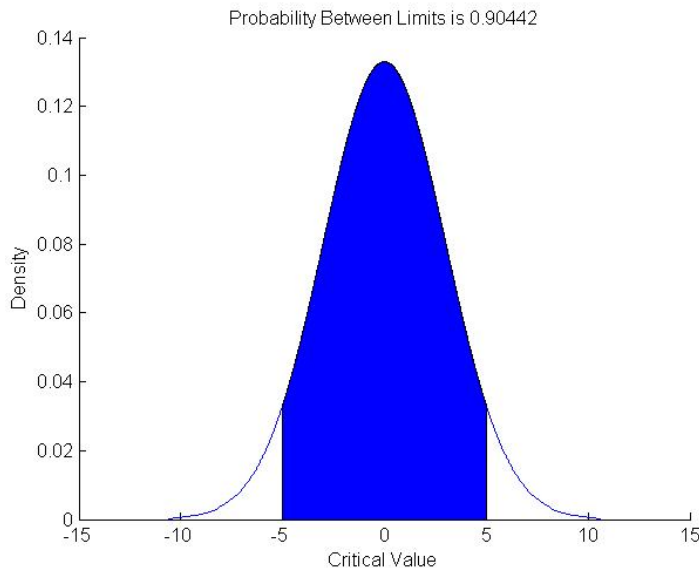


圖 3.7: 常態分配函數的部分機率圖。

其中第一個參數指出涵蓋的範圍，第二與第三個參數則是常態分配的參數。`normspec` 指令除畫出覆蓋的面積圖之外，也計算出這個面積並展示在 `title`。

3. 畫出來的 pdf 與 cdf 圖是否都符合你的預期呢？如果不是，去翻翻統計學、機率論的書驗證一下。
4. 釐清楚機率分配的 pdf 圖與依亂數值產生的直方圖。不要搞混了！兩者間有什麼異同？
5. 畫出來的直方圖若與自己認定的不符時，請特別注意是畫圖技巧不好，還是指令的操作錯誤或是觀念的錯誤。通常直方圖的畫圖技巧必須注意樣本數及範圍間距的選擇。
6. 透過這個單元的練習，當給予一組隨機資料時，你有多少把握知道其原始的

分配是什麼？這與資料量的多寡有關嗎？除了畫直方圖之外，還有沒有其他方式可以提供更多的參考訊息呢？

7. Boxplot 是一種常用的統計圖，可以立刻看出其代表變數的位置及其分佈狀況，也可以呈現出資料分佈的對稱性或偏斜情況。另外最大、最小值及 outliers 也都可以清楚的看出來，方便迅速的比較並得到粗淺的認識。MATLAB 的 boxplot 指令的幾種做法如下：

```
%先模擬兩組資料
x1=normrnd(0,1,100,1);
x2=normrnd(1,2,100,1);
boxplot([x1 x2])
```

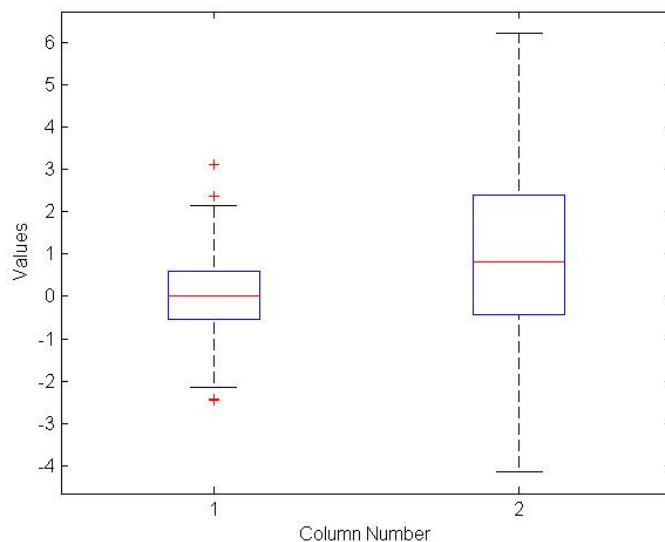


圖 3.8: 兩組資料的 Boxplot 圖。

圖 3.8 顯示了兩組資料的 Boxplot，每個「盒子」裡面有三條水平線，由上而下分別代表 75,50,25 百分位的位置。「盒子」上下個有一條垂直虛線，頂端分別代表這組資料（扣除 outliers 後）的最大值與最小值。「+v」代表 outliers，垂直虛線的長度是 interquartile(75 百分位- 25 之百分位的距

離) 的 1.5 倍 (可以調整)。boxplot 指令為方便比較, 有可以增加如下的選項:

%先模擬兩組資料

```
boxplot([x1 x2], 'notch', 'on', 'labels', {'Group A', 'Group B'})
```

圖 3.9 將中位數的上下切出 95% 的信賴區間的缺口, 方便兩組並列時比較中位數的差異。另外也加入自訂的組別名稱。圖 3.8 與圖 3.9 的資料來自對稱的常態分配, 不妨試著拿偏斜分配的資料畫幾張 Boxplot, 才能對 Boxplot 的特質有所掌握。

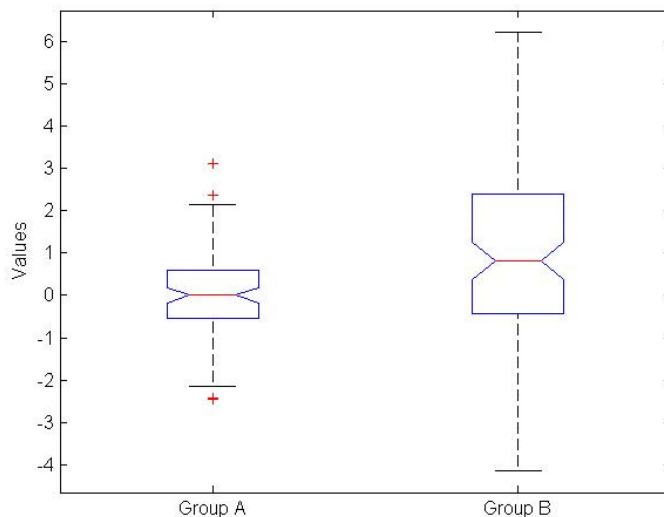


圖 3.9: 兩組資料的 Boxplot 圖 (加選項)。

8. 觀察兩組資料的相關性, 相當具實用價值。本練習進而去模擬產生具某種相關性的兩組實驗資料。產生模擬資料對研究工作而言, 往往是必備的基本動作。
9. 兩變數的相關性不一定是線性的, 這裡只是給予線性的訓練。你也可以試試產生非線性相關的資料, 再去計算  $r$ , 看看發生什麼事了?

10. 請觀察變數資料的產生與關係的形成，及最後散佈圖的樣子，要牢牢的將這些東西連結在一起。加強對資料的感覺，培養與資料間的感情，即所謂的『資料感』。

### 3.3 作業

1. 畫出下列分配的 pdf 及 cdf 圖，並觀察參數的改變與圖形的關係。將相關的圖疊在一起（至少 5 張圖疊成一張）。所有的參數資料盡量寫在圖形的空白處。

- Normal Distribution: (1) 固定  $\mu$  改變  $\sigma$ , (2) 固定  $\sigma$  改變  $\mu$
- Chi Square Distribution: 觀察在不同自由度 ( $\nu < 30$ ) 的變化情形。另外可以觀察當自由度很大時，卡方分配的長相？譬如，畫一張圖將自由度 1000 的卡方分配與常態  $N(1000, 45)$  的 pdf 畫在一起。
- Binomial Distribution: 自行調整參數。
- F Distribution: 當兩個自由度的參數  $\nu_1, \nu_2$  由小變大時，F 分配的樣子會如何改變呢？有其極限嗎？請仔細觀察。畫出有代表性的分配圖。
- T Distribution: 觀察當自由度由小變大時，圖形的變化情形。當自由度很大時是否接近標準常態？請以圖形表達出來。
- $\beta$  Distribution: 這個分配很豐富，千變萬化，非常精采。當兩個參數  $a, b$  大小不同時，分配的樣子會如何改變？譬如，觀察當  $b > a$  (固定  $a$ , 調整  $b$ ) 時、當  $a > b$  (固定  $b$ , 調整  $a$ ) 時及當  $a = b$  時。
- Exponential Distribution: 自行調整參數。

其他如幾何分配、卜瓦松分配都可以嘗試。

2. 產生 5 組亂數 (來自 5 個不同的分配)，其個數與分配自行決定，分別畫出直

方圖。觀察畫出來的圖是否符合預期呢？你必須確定這一點。不能畫了就算！

3. 分別產生具左偏與右偏分配的資料各一組，畫出 Boxplot。
4. 假設  $x$  為一服從標準常態分配的變數，令變數  $y = x^2$ 。利用適當的亂數指令產生 1000 個變數  $y$  的樣本並繪製其直方圖。類似這樣的抽樣分配在機率課本找到很多，不妨多做幾個。
5. MATLAB 的 statistics toolbox 也提供一組叫做「Copulas」的指令，可依指定的相關係數產生兩個（或以上）變數的亂數，其使用方式與範例可以在線上使用手冊以「Copulas」為關鍵字查詢到，譬如

```
n = 500;  
U = copularnd('Gaussian',[1 0.8; 0.8 1], n);  
plot(U(:,1),U(:,2),'.');
```

其中第二個參數為相關係數矩陣。這裡產生的亂數具均等分配，其值在  $(0, 1)$  之間。如果你使用的 MATLAB 找不到 copularnd 這個指令，表示版本較早。



## 第 4 章

# MATLAB的程式檔：以微分的計算為例

MATLAB作為一個數學計算的工具，不只是提供許多計算的指令而已，它還是一個電腦語言（接近 C 語言），不但充分運用了一般電腦語言的邏輯能力，還結合數學相關的指令成為一個具高度計算能力的程式語言，可以解決複雜程度較高的數學問題。本章以微分的計算初窺 MATLAB 解決數學問題的典型與能力，也是從學習 MATLAB 的指令跳脫到 MATLAB 程式的第一步。

### 本章將學到關於程式設計

MATLAB的程式結構、程式的邏輯概念與程式檔案的管理。

〈本章關於 MATLAB 的指令與語法〉

操作元 (operators): `.^` `./`

指令: `polyval`, `format long`, `semilogx`, `gtext`, `diff`, `polyder`, `log`, `fprintf`,  
`input`

語法：註解的使用時機與方式。

## 4.1 背景介紹

微分根據其定義：

$$f'(x) = \frac{df(x)}{dx} = \lim_{h \rightarrow 0} \left( \frac{f(x+h) - f(x)}{h} \right) \quad (4.1)$$

代表函數在某個位置 ( $x$ ) 的變化情形 (速率), 是陡峭的還是平緩的, 在不同的應用領域上有其特殊的意義。這種瞬間的變化在函數上以極限 (lim) 的觀念來表達; 當函數從  $x$  移動到  $x+h$  時, 函數值的變化從  $f(x)$  到  $f(x+h)$ , 這個變化速率表示為

$$\frac{f(x+h) - f(x)}{(x+h) - x} \quad (4.2)$$

$h$  愈小代表愈短距離的速率, 當  $h$  小到幾乎等於 0 時, 稱為函數在  $x$  的瞬間變化率 (一般也稱為「斜率」), 代表函數在這個點上的動態表現。圖 4.1 表示出函數在兩個點上的變化情形。

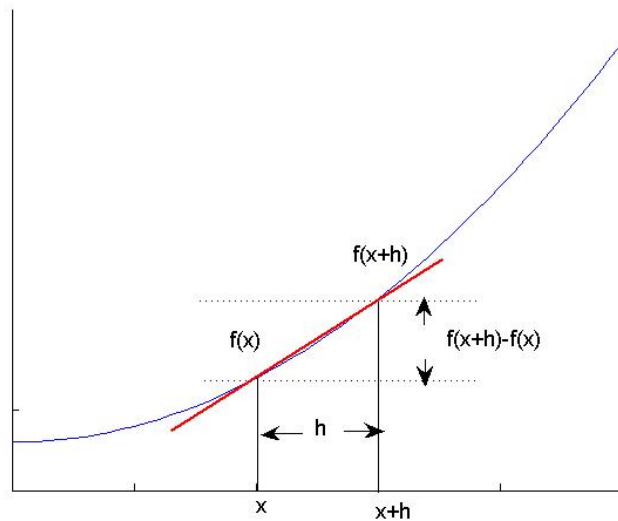


圖 4.1: 微分示意圖

配合函數的型態, 已推展出許多計算極限的微分技巧, 這些技巧為人所熟悉, 但是電腦可沒這麼聰明, 雖有人工智慧的輔助, 還是不及人腦來得靈光。而且技巧



總是有限，碰到一些「不友善」的函數時，往往不夠用。這時電腦可派上用場了，除了可以迅速並大量計算微分問題外，函數的「長相」通常它是認不得的。撇開像 Mathematica 或 Maple 及 MATLAB Simulink(骨子裡也是 Maple) 這類可以根據數學符號就能計算微分的軟體，以電腦程式來計算數值微分，通常會從上述的定義來下手。以下列舉幾個範例來練習，看看電腦如何計算函數  $f(x)$  在任一點的導數 (或稱切線斜率)。<sup>1</sup>

## 4.2 練習

---

範例1: 假設  $f(x) = x^2 + 3x + 1$ ，求  $f'(3)$ 。

- 定義上的極限  $\lim$  在數值計算上如何表示？
- $h$  值該給多少才算趨近於0？試試看當  $h = 0.1, 0.01, 0.001, 0.0001$  時，導數值分別為多少？與實際值相差多少？哪一個選擇比較好？導數值由下列公式計算：

$$f'(3) = \frac{f(3+h) - f(3)}{h}$$

上述的公式需要計算兩個多項函數值，輸入時稍嫌麻煩，此時可以使用 MATLAB 提供計算多項式函數的指令 *polyval*，可以讓整體指令「乾淨」許多，譬如計算  $f(3)$  可以依序輸入下列指令

```
p=[1 3 1] %函數的係數
f=polyval(p,3)
```

MATLAB的命令視窗將指令執行的結果顯示出來，為避免畫面的紊亂，僅顯示到小數第四位，若要觀察更多的小數位數，可以執行以下指令延長到14位。

```
format long
```

---

<sup>1</sup>有一門領域叫做「數值分析」，專門研究如何利用電腦來解決數學計算上問題。關於「微分」當然也發展出許多的演算法 (Algorithm)，克服各式各樣可能在電腦上產生的問題，本文僅就最簡單的方式著手，避免涉及太多細節，以免干擾本文真正的目的：介紹 MATLAB 的程式功能。

如要恢復原來的四位數，執行指令 `format` 即可。

---

**範例2:** 從上一個練習得知，計算數值導數時， $h$  值的選擇會影響導數值。對於這個函數你能找出一個最好的  $h$  值嗎？畫一張圖看看  $h$  值的選擇對計算導數的影響。畫圖時  $x$  軸的選擇很重要，可以採用 `semilogx` 指令取代 `plot` 可以看得更清楚。

---

MATLAB的特點是矩陣的運算，本練習要觀察不同的  $h$  值計算出的導數值，便可以利用這項特質。譬如建立一個變數  $h$ ，其內容涵蓋欲觀察的  $h$  值，如

$$h = \left[ 10^{-4} \quad 10^{-5} \quad 10^{-6} \quad 10^{-7} \quad 10^{-8} \quad 10^{-9} \quad 10^{-10} \quad 10^{-11} \quad 10^{-12} \right]$$

建立的方式可以如上式一一輸入，但最快也最能掌握 MATLAB 精髓的方式為：

$$h = 10.^{-4:-1:-12} \text{ 或 } h = 1./10.^{4:12}$$

利用 MATLAB 提供的運算元  $(\cdot)$ ，可以同時針對矩陣或向量的每一個元素做計算，依序執行下列指令，並逐一觀察每個指令執行的結果，將對 MATLAB 的使用有進一步的瞭解。值得注意的是，第二個指令的分母不需刮號，理由是指數運算元的計算順序優於其他。

```
p=[1 3 1];           %函數的係數
h=10.^(-4:-1:-12); %列出所有 h 值
x=3;
f1=polyval(p,x);      %計算 f(3)
f2=polyval(p,x+h);    %計算不同 h 值下的函數值 f(3+h)
fp=(f2-f1)./h;        %計算不同 h 值下的導數值
semilogx(h,fp)
```

圖 4.2 展示上述指令的執行結果。透過同時計算比較大範圍的  $h$  選擇，從圖形可以清楚的看出  $h$  的選擇對導數的影響，也看出過小的  $h$  值產生電腦計算上困擾，

如圖形的左邊，可見在電腦上極限值的選擇仍有其侷限，並非無限制的趨近 0。圖 4.2 也同時展示 MATLAB 在繪圖區強大的編輯功能，幾乎所有顯示在上面的元件（如座標軸、背景、刻度、線條顏色粗細樣式...）都可以透過繪圖區的編輯工具進行調整。當然所有的編輯也都可以經由適當的 MATLAB 指令，不過用滑鼠編輯還是比用鍵盤簡單許多，也比較好玩。

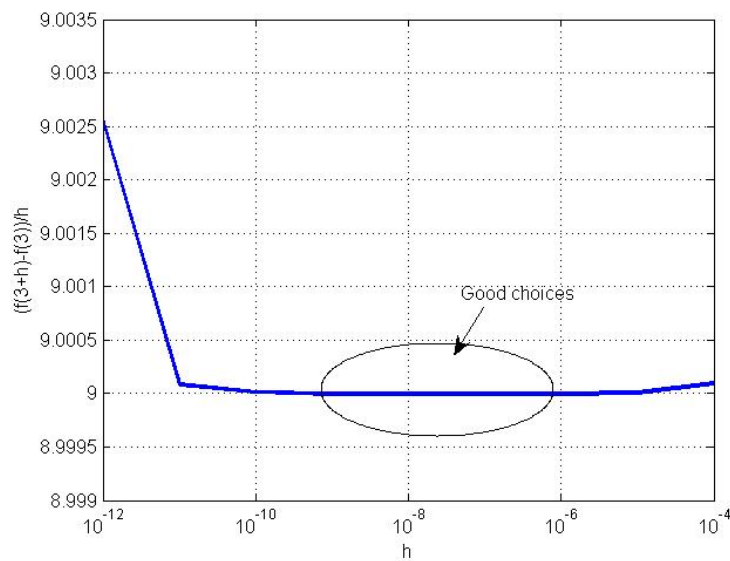


圖 4.2:  $h$  值的選擇對導數計算的影響

**範例3:** 前述的練習需要幾個指令來完成導數的計算，如果每計算一次就要重複執行相同的指令，代入不同的數值，那將會非常沒有效率，更何況面對複雜的數學問題可能要花數十，甚至數百條指令來處理。在這個情況下，MATLAB 允許將這些指令集中起來，存放在一個檔案裡面，藉由在命令視窗輸入檔名，MATLAB 會循序執行檔案裡面的每一條指令。這個檔案被稱為 MATLAB 的程式。請將上述範例計算導數的指令集中起來，存成一個檔案（副檔名自動設定為 .m），<sup>2</sup> 譬如，derivative.m

<sup>2</sup>請注意檔名的選擇，不能與 MATLAB 內建的指令名稱相同，那將影響該指令的功能。

請注意要 MATLAB 正確的執行程式檔，必須符合下面兩個條件之一：

- 「命令視窗」所在的位置必須相同於程式檔。這可藉由變更命令視窗的「current directory」達到。
- 將程式檔所在的目錄設為 MATLAB 可以自動參考的路徑。這可藉由「File」→「Set Path」→「Add Folder」達到。

將所有必要的指令放在一起，不但方便執行，也容易管理與維護，更提供程式寫作許多可以運用的技巧。

---

**範例4:** 範例2針對特定函數  $f(x) = x^2 + 3x + 1$  與特定的  $x$  值，試圖找出計算數值導數 (4.2) 時適當的  $h$  值。在這個範例裡，我們想知道對於特定的  $h$  值，譬如  $h = 0.1$ ，數值導數 (4.2) 與實際的導函數  $f'(x) = 2x + 3$  相差多少？

---

圖 4.3 清楚的展示不當的  $h$  值 ( $h = 1$ ) 計算出來的導數與實際值的差距。當  $h = 0.1$  時，數值導數與實際導數非常貼近，但  $h$  究竟該多小才是恰當需依實際情況而定。圖 4.3 選用了繪圖區的「Legend」工具鍵來區分不同線條的意義，提供 `gtext` 或 `text` 指令外的另一種選擇。

---

**範例5:** MATLAB 提供什麼指令計算數值導數呢?(hint: `diff`, `polyder`)。試著用 MATLAB 指令的方式來計算導數值。觀察看看，與你前面所計算的值相差多少？哪一個比較準？

---

像 MATLAB 這種等級的數學軟體當然會提供計算導數的功能指令，究竟 MATLAB 計算導數的指令與自己根據定義所寫的程式有何不同？這是令人好奇也值得一探的。如果發現 MATLAB 的指令執行結果令人滿意，未來程式有需要自然可以直接引用，不一定要使用自己寫的，畢竟專業的程式設計還是比較縝密，使用上也比較方便。

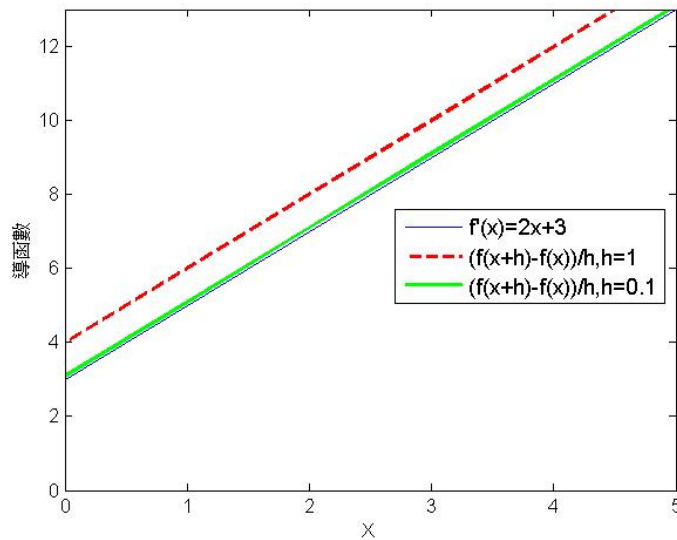


圖 4.3: 實際的導函數與數值導數在  $h = 1, 0.1$  的比較

### 4.3 觀察

寫程式解決數學問題常會觀察到一些平常看不到的「內涵。」透過這些觀察與反覆思考查證，往往會對理論有進一步的體會，那可是平常人無法直接從書本獲得的。以下列舉些觀察點供參考。

- $h$  值是影響數值計算的重要因素，是否在所有的練習中可以歸納出一個適當的  $h$  值，對所有的函式都適用？對於  $h$  值的選擇，你有哪些觀察呢？
- MATLAB 利用單一指令解決單純問題。當一個指令無法解決問題時，可藉由指令的「接力」執行，共同合作完成任務。多個指令組合（或稱程式）並且一起被執行，方便解決較複雜的問題。但當指令被組合起來時，必須注意其前後的合作關係，如何安排變數、如何決定運算的次序、如何適當的採用 MATLAB 的內建功能，讓程式執行順暢、有效率，而且易懂、容易變更、擴充，都是非常重要的程式寫作技巧。初學者應細心體會其中的奧妙。多看別人的程式，比較自己的程式，從中獲得真正的寫作技巧。
- 程式的執行遵循從上而下的規則，但程式的寫作可不一定要從第一行寫起。

可以從做容易下手 (最直覺) 的地方著手, 再依程式的需要補足前後面的指令, 這樣反而比較順手, 速度比較快。因為程式語言的邏輯與人腦還是不同的, 因此程式寫作的過程常常是上上下下穿插的寫, 並且不斷的執行部分過程, 以確定指令與邏輯的正確, 千萬不可妄想一次寫完再執行, 往往會連錯在哪裡都不知道, 徒增除錯的困難。

- 本單元著重在多個指令的組合, 共同完成一件工作。不過在程式的寫作過程中或是最後完成的作品, 常需要能看得到程式執行的結果, 這時候一些文字輸出的指令顯得重要, 譬如 `fprintf` 的使用更為常見, 以多項式函數的微分為例, 假設我們選擇  $h = 10^{-8}$  做為  $h$  趨近於零的替代, 下列程式可以直接在命令視窗顯示  $f'(3)$  的結果, 其中多項式  $f(x) = x^2 + 3x + 1$ 。

```
p=[1 3 1];           %函數的係數
h=10-8;             %適當的 h 值
x=3
f1=polyval(p,x);      %計算 f(3)
f2=polyval(p,x+h);    %計算不同 h 值下的函數值 f(3+h)
fp=(f2-f1)/h;         %計算數值導數
fprintf('The derivative of f(x) at x=%7.2f is %10.4f \n',x,fp)
```

程式語言都有指令將文數字 `print` 到螢幕上 (console), MATLAB 的指令 `fprintf` 就是做這件工作。<sup>3</sup> 上述的 `fprintf` 指令將  $x$  值與  $f'(x)$  輸出到命令視窗, 且為了方便察看, 在第一個格式參數 (format) 裡面放了必要的文字。格式參數是 `fprintf` 使用最主要的部分, 在單引號裡面包含兩個部分, 一是文字, 另一個是格式, 穿插放置, 將變數的結果與文字一同呈現出來。格式部分只要是將變數 (自第二個參數起, 如  $x, fp$ ) 以適當的格式呈現出來, 譬如格式 `%7.2f`, 「f」代表 fixed-point, 用來呈現數值, 前方的數字 (如 7.2) 代表欲呈現出來的實數數字長度, 其中小數點後的數字 2 代表小數位數, 7

<sup>3</sup>`fprintf` 不只可以將結果「寫」到命令視窗, 也可以寫到檔案裡面, 詳細的使用方式可以在線上手冊上看到。

代表「整數位數 + 小數位數 + 1」，換句話說，整數位數是 4。最後的 `\n` 會造成換行的效果，否則下一個 `fprintf` 會連續在後面。MATLAB 也提供除了 `f` 以外的特殊格式，詳見線上手冊。

這支程式看以來還不夠好，如果需要測試在不同  $x$  值的導數，程式必須不斷修改與存檔，略顯麻煩。有個指令 `input` 提供程式執行時，從鍵盤直接輸入，使用方式如（取代上述程式第三行）

```
x=input('Enter x =');
```

#### 4.4 作業

1. 將數值導數的定義改為

$$f'(x) = \frac{df(x)}{dx} = \lim_{h \rightarrow 0} \left( \frac{f(x+h) - f(x-h)}{2h} \right)$$

與之前的定義做比較，哪一個定義比較準確？舉範例1的函數，取幾個不同的  $h$  值，譬如： $h = 10^{-2}, 10^{-3}, \dots, 10^{-9}$ ，等，畫一張圖紀錄在每一個  $h$  值，不同導數公式的誤差（誤差=|數值導數 - 實際導數|）。即畫一張圖顯示兩條來自不同的導數定義的誤差線。

2. 從上題的結果，你可以判斷在數值計算上哪一個定義比較好嗎？下面這個練習可以幫助釐清一些觀念：

令函數為  $f(x) = x^2 + 3x + 1$ ，將此函數分別代入兩個定義中，經過一番化簡之後，你得到什麼？這個結果可否幫助你判斷在數值計算上哪一個定義比較好？

3. 同範例4，畫一張圖比較函數  $f(x) = -x^3 + 6x^2$  的數值導數與理論導數的差別。所謂理論導數，即  $f'(x)$ ；而數值導數則是利用數值方法（例如前述的練習）以電腦程式計算的函數值。換句話說，想比較下列兩個函數的差

異:

$$\begin{aligned} f'(x) &= -3x^2 + 12x \\ f'_{app}(x) &= \frac{f(x+h) - f(x)}{h} \end{aligned}$$

請自行選擇適當的  $h$  值與  $x$  的間距範圍; 將理論值的圖與數值計算的圖畫在一起。請注意

- 你可以選擇不同的  $h$  值, 藉以凸顯不當的  $h$  值會影響數值導數的精確度。
  - $x$  軸範圍的選擇頗為關鍵。範圍的大小與區域的選擇不當往往會造成視覺上的錯覺, 導致錯估兩者間的差距。不妨多觀察幾個不同的區域與範圍, 甚至可以去計算誤差的大小。
4. 同上題, 但是函式為:  $f(x) = \frac{\ln x}{x^3}$ , 其導函數  $f'(x) = \frac{1-3\ln x}{x^4}$ 。選擇  $x > 0$  的適當範圍。<sup>4</sup>
5. 前面兩題在凸顯  $h$  值的角色, 如果要訂出一個準則來決定到底多小的  $h$  才是恰當, Root Mean Squared Error(RMSE) 倒是不錯的選擇, RMSE 定義如下:

$$RMSE = \sqrt{\frac{\sum_{k=1}^N (f_h(x_k) - f(x_k))^2}{N}}$$

其中  $f_h(x)$  代表數值導數,  $x_k$  是選取某個範圍內的  $x$  值, 共選取  $N$  個。我們可以指定當所選擇的  $h$  值使得  $RMSE < 0.1$  即可。請為前面兩支程式加入 RMSE 的計算, 為每一個所選擇的  $h$  值計算出相對的 RMSE, 並藉此決定適當的  $h$  值。

6. 利用上一節「觀察」提供的指令 `input`, `fprintf` 及 計算多項式函數一次微分後的係數 `polyder`, <sup>5</sup> 寫一支可以計算任何多項式函數導數的程式, 執行

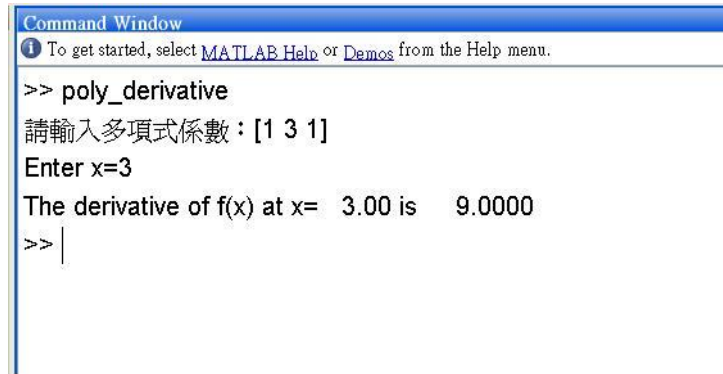
<sup>4</sup>計算對數函數  $\ln x$  的指令為 `log(x)`

<sup>5</sup>接觸一個新的指令, 最好先看看線上手冊的說明, 在自己試試幾個簡單的東西, 多靠自己摸索, 程式功力才能躍進



#### 4. MATLAB的程式檔：以微分的計算為例 57

時要求輸入多項式係數 (例如輸入  $[1\ 3\ 1]$  代表多項式  $f(x) = x^2 + 3x + 1$ ) 及  $x$  值, 程式執行後輸出  $f'(x)$  的結果。下圖是執行的過程與結果:



```
Command Window
To get started, select MATLAB Help or Demos from the Help menu.

>> poly_derivative
請輸入多項式係數: [1 3 1]
Enter x=3
The derivative of f(x) at x= 3.00 is 9.0000
>> |
```

圖 4.4: 多項式導數計算



## 第 5 章

# MATLAB 程式的迴圈技巧

迴圈技巧是程式語言的生命，有了它程式才顯示出價值。電腦之應用貴在於能大量且快速的執行相同的計算，而迴圈便是這項功能的靈魂。MATLAB 程式的迴圈技巧與其他語言類似，甚至相同，具備其他語言基礎的可以很快的進入狀況。

### 本章將學到關於程式設計

MATLAB的程式結構、程式的邏輯概念與程式檔案的管理。

〈本章關於 MATLAB 的指令與語法〉

操作元 (operators): `.\` `./`

指令: `for`, `factorial`, `pause`

語法: `for` 迴圈的建立與技巧。

## 5.1 背景介紹

一支程式由數個指令組成，執行時是一根腸子通到底，由上而下逐步執行，一個指令執行一個動作。當完成一件事情（計算）需要龐大的運算動作時，有限的指令是行不通的。譬如，計算泰勒展開式

$$f(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \cdots + \frac{f^{(n)}(a)}{n!}(x-a)^n + \cdots$$

當展開的項次  $n$  很大或不確定大小時，固定指令的寫法並不可行。從上式累加的項目中發現，這些項目具規則性的結構，程式語言中的迴圈技巧 (for loop) 正可以輕巧的解決這個問題。以下列出一些簡單的範例，說明迴圈技巧的使用方式與時機，最後利用泰勒展開式與迴圈技巧計算無理函數  $\exp(x)$ 。

## 5.2 練習

---

**範例1:** 寫一支迴圈程式計算1加到100，即計算  $\sum_{k=1}^{100} k$ 。程式執行時要在每個迴圈中印出累加值及迴圈數。

---

迴圈的指令為 for，請利用『help for』找出迴圈的使用方法，指令如下。

clear	註：程式開始前，先清除命令視窗的所有變數
total=0;	註：必須先將「總和變數」total 的值預設為0
for i=1:100	註：迴圈開始
total=total+i;	註：開始累加，變數 total 值每次都不同
end	註：迴圈結束
total	註：印出結果

上述程式第三行代表迴圈的開始，第五行表示迴圈結束。當程式開始執行時，由上(第一行)逐步往下，直到進入迴圈後，將重複執行迴圈內的所有指令若干次，次數由指令迴圈索引變數  $i$  所設定的向量決定。譬如， $i=1:100$ ，迴圈內的指令將被執

行 100 次, 每次索引變數  $i$  代表向量中的每個數, 由第一個 ( $i=1$ ) 到最後一個 ( $i=100$ )。

第四行運用了常用的累加技巧, 這裡的等號並非數學裡的相等之意, 而是將右邊所計算的結果指定給左邊的變數。於是變數 `total` 的值在迴圈重複的過程中每次都不同, 因而產生累加的效果。在上述的範例中嘗試將第二行拿掉, 看看執行結果有何差別。將一行程式暫時拿掉的方式, 是將其變為註解行, 即在最前面加上百分比符號<sup>1</sup>。另外, 寫迴圈程式有個慣例, 在迴圈開始的第一行與最後一行間的程式碼做適當的縮排。這樣做的原因除方便除錯外, 也增加程式的可讀性。

Tips: 學習程式寫作最好的方法是參考正確的示範程式, 在瞭解語意後, 多模仿, 久之, 便能得心應手。在模仿上述程式片段前, 初學者最好先找出迴圈指令 `for` 的正確語法, 再逐行解讀示範程式, 完全瞭解後, 最後執行示範程式並觀察結果 (最好逐行觀察中間的結果)。

累加技巧屬於基本動作, 應多做練習, 以下練習一個稍複雜些的問題。

---

**範例2:** 利用累加技巧計算樣本變異數  $\hat{\sigma}^2 = \sum_{i=1}^N (x_i - \hat{\mu}_x)^2 / (N - 1)$ , 其中的樣本  $x_i$  自行以亂數產生器 `normrnd(0,1,1,N)` 產生, 樣本數  $N$  設為 30。

---

樣本變異數的計算有很多方式, 在前面的單元也介紹過, MATLAB 也提供計算樣本變異數的指令 `var`, 這裡純粹是為練習迴圈的累加技巧, 刻意利用這個簡單的問題。問題雖然簡單, 但是模式是固定, 將來遇到複雜的問題可以根據本題的模式推展。請讀者先試著以上一題的架構, 寫出本題的程式, 對錯與否可以 `var` 指令驗證。

---

**範例3:** 在同一張圖上畫出斜率不同的直線函數  $y = mx$ , 斜率  $m = 1, 2, \dots, 50$ , 如圖 5.1。

---

<sup>1</sup>註解指令較快速的方式是先將滑鼠游標指到該行, 再按「Ctrl+r」即可, 復原時再按「Ctrl+t」, 且不管單行或多行都可以。

## 62 MATLAB 的數學實驗

要將 50 個斜率不同的直線畫在同一張圖上，當然不能手動一張張疊上去，迴圈技巧是最好的選擇。以對迴圈技巧最初淺的想法可以寫成

```
clear
x=-5:5;
for i=1:50
    y=i*x;
    plot(x,y)
end
```

上面的程式藉由迴圈中索引變數  $i$  的改變，執行迴圈中的兩行指令 50 次，變數  $i$  的值從 1 到 50。雖然執行的指令不變，但是透過  $i$  的改變，變數  $y$  的值（向量）也變了，當然作圖時也會產生變化，於是斜率不同的直線便一一產生。上述程式還不能產生圖 5.1，需要處理一些細節的問題，譬如疊圖。圖 5.1 便是由下列程式產生的。

```
clear
x=-5:5;
y=x;
plot(x,y)      註：先畫第一條線，方便做 hold on
ylim([-250 250])  註：預先調整 Y 軸範圍
hold on
for i=2:50      從 m=2 畫起
    y=i*x;
    plot(x,y)
    pause(0.1)  製作成動畫，方便觀察繪圖情況。
end
hold off
```

這裡利用指令 `pause(0.1)` 製造程式執行暫留 0.1 秒，造成繪圖過程中的視覺現象，看起來像動畫，一來增加程式樂趣，再則將迴圈執行的過程依序呈現出來，常用來檢查程式的邏輯是否錯誤。

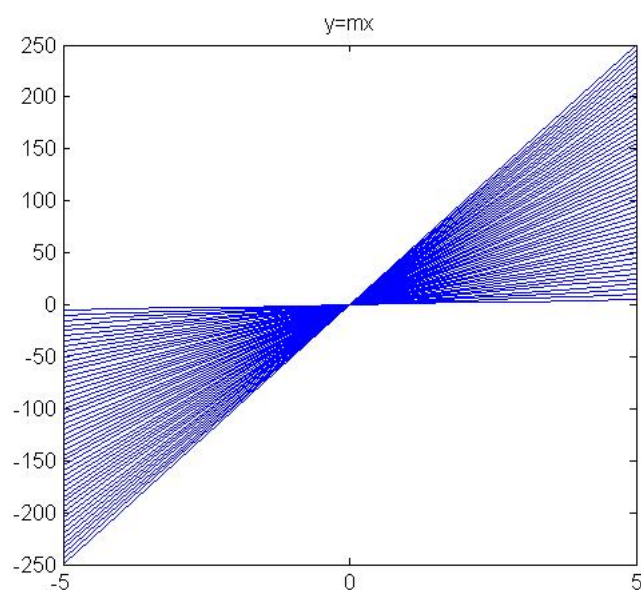


圖 5.1: 利用迴圈技巧畫圖

通常迴圈的索引變數值習慣引用整數值, 如上述的  $i=2:50$ 。如果本題的斜率  $m=0.5, 1, 1.5, \dots, 20$ , 索引變數  $i$  固然可以設為  $i=0.5:0.5:20$ , 但這並非好習慣, 非整數的設定有些時候會出錯的。類似的情況非常普遍, 解決的方式如下

```

...           註: 同前
m=0.5:0.5:20;  註: 預設所有的斜率值在一向量裡
n=length(m);   註: 計算斜率個數
for i=1:n
    y=m(i)*x;   註: 依序取得斜率值
    plot(x,y)
    ...         註: 同前
end
...           註: 同前

```

這裡的  $i$  變成名符其實的索引值, 從向量  $m$  中逐一挑出預設的斜率值, 共有  $n$  個。這個技巧非常好用, 請記得常回來看看。

---

範例3: 承上題, 但函數圖改爲

1.  $y = ie^x, -5 \leq i \leq 5$
2.  $y = i \sin(x), -5 \leq i \leq 5$
3.  $y = i(x - 3)^2 + 2, -10 \leq i \leq 10$

其中  $i$  皆爲整數。

---

自己可以隨意找些函數來玩玩, 只要更動某個參數值常常可以形成很動人的圖案, 配合動畫的效果, 滿有娛樂效果。

---

範例4: 如果想看到機率密度函數的「長相」與其參數間的關係, 可以利用迴圈的技巧來改變參數值, 再將函數一個個畫上去。瞬間便可以看許多密度函數的改變, 或是利用時間的暫停慢慢欣賞密度函數隨參數改變的變化情況。試著利用迴圈技巧畫出卡方分配的機率密度函數隨自由度改變, 譬如從  $\nu = 3, 4, \dots, 20$  的變化情形。

---

依前述範例, 很容易利用迴圈中索引值做出如圖 5.2 所展示的, 在不同自由度下的卡方密度函數。

不過如果希望在時間暫留的動態呈現下, 也可以將參數值送上畫面, 將更有利密度函數改變的觀察。此時牽涉到將不同文字動態呈現出來的技巧, 如下列片段程式

```
...
for i=4:20
    ...
    plot(x,y)
    str=strcat('\nu =',num2str(i));
    title(str);
end
```



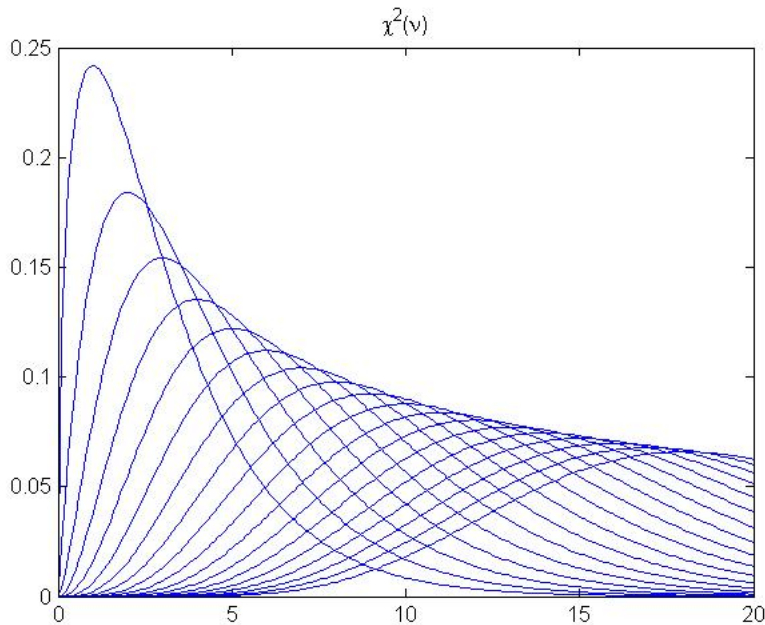


圖 5.2: 不同自由度下的卡方密度函數

其中指令 `strcat` 常用來將不同來源的文字串連起來，裡面的逗號可以一直加下去，一次串連多組文字。這裡串連了兩組文字，一組是固定文字 `'\nu ='`，另一組是動態文字，由迴圈的索引值經過指令 `num2str` 轉來。索引值 `i` 是數字，按 MATLAB 的語法，不得與文字串連，所以需經過數字 (num) 到文字 (str) 的轉化 (2, 音同 to)。因為 `i` 的改變，造成串連出來的字串每次都不同，適合做動態的呈現。這裡將這組動態的文字當做 `title` 展示出來。

---

**範例 5:** 設計一個射飛靶的遊戲。首先畫一個正圓在座標中央，利用適當的亂數產生器給出兩個數值，分別代表座標軸上的 `x` 與 `y` 座標。接著在座標軸上的 `(x,y)` 座標點畫上任一個符號，譬如「o」。利用迴圈重複執行隨機座標點的產生與作圖，配合暫停指令 `pause`，完成一個射飛鏢的簡單遊戲。如圖 5.3 (a)。

---

這個遊戲用到迴圈技術，但是並沒有使用到索引變數 `i` 的值，純粹重複固定的指令若干次。此外 `pasue` 指令可以加入暫停秒數，如 `pause(1)` 表示暫停 1 秒，在

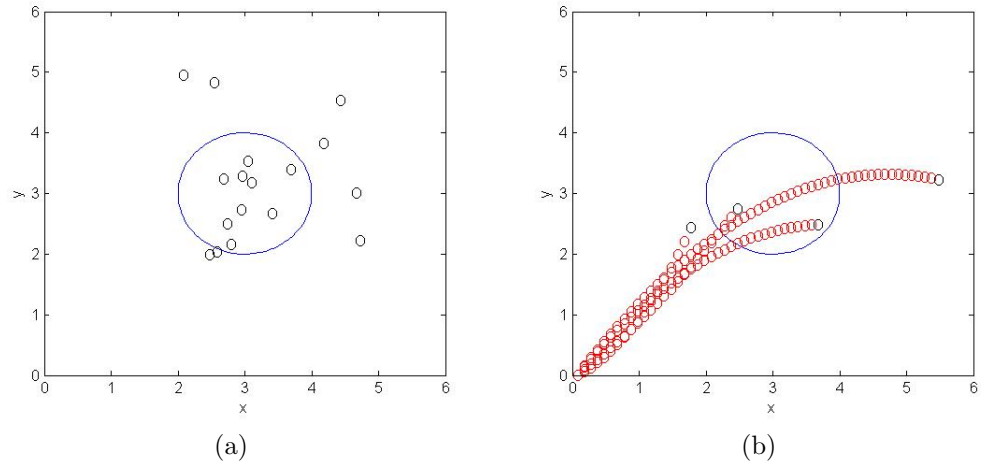


圖 5.3: 射飛標遊戲

這裡不加分秒數代表無限期暫停，等待使用這按任意鍵停止暫停，這個按鍵的動作感覺上就好像發射飛鏢。此時若要停止程式，可在命令視窗按「Ctrl+c」，否則將繼續執行到迴圈走完。

```
clear
ezplot('(x - 3)^2 + (y - 3)^2 = 1') 畫圓
axis([0 6 0 6])                    限制範圍，讓飛靶在中間
axis square                         將座標圖設定為正方形，讓飛靶看起來是正圓
for i=1:100                          設定玩飛靶的次數
    x=normrnd(3,1);                  以常態分配產生隨機亂數，當做 X 座標
    y=normrnd(3,1);                  以常態分配產生隨機亂數，當做 Y 座標
    text(x,y,'O')                    標示字串 O 在 (x,y) 座標點上
    pause                            程式執行無限制暫停，直到按鍵盤任意鍵
end
```

此外，上述程式採常態分配的隨機亂數當做每次射出飛鏢的位置，設計者可以依自己的喜好或遊戲的規則使用不同的方式取得落點。甚至為了增添遊戲的樂趣與畫面的可看性，可以加入不同的元素，譬如，畫出飛鏢射出去的軌跡，如圖 5.3 (b) 展

示的紅色軌跡圖。如果再配合動畫的呈現，便栩栩如生，像看大聯盟的線上轉播一樣，看得到投手投出去的球飛行的軌跡。軌跡的部份可以在飛鏢跑到定點時抹去，才不會讓畫面越來越混亂。至於軌跡繪製方式更是千變萬化，設計者可以憑自己的想像來製作。如圖 5.3 (b) 的軌跡，是以一個二次函數  $y = ax^2 + bx + c$  的曲線圖畫出來的，其中  $a, b, c$  由三個座標點來決定，分別是  $(0, 0), (x, y), (x/2, x/2)$ 。也就是先以隨機亂數取得落點  $(x, y)$ ，並假設發射起點為  $(0, 0)$ ，在這兩點間取一點，譬如  $(x/2, x/2)$ ，便可以計算出該二次函數的係數，再利用 `text` 指令配合另一個迴圈「沿路」寫上符號即可。

---

**範例 6:** 假設隨機變數  $X$  服從均等分配  $\text{Unif}(0,1)$ ，隨機產生  $n$  個樣本  $x_1, x_2, \dots, x_n$ ，並計算其樣本平均數  $\bar{x} = (\sum_{i=1}^n x_i)/n$ 。如果想知道  $\bar{x}$  的抽樣分配長什麼樣子，可以利用迴圈技巧進行  $N$  次的隨機抽樣，每次取  $n$  個樣本並計算樣本平均數，共得  $N$  個樣本平均數  $\bar{x}_1, \bar{x}_2, \dots, \bar{x}_N$ ，最後畫出這  $N$  個平均數的直方圖。本題請先設定  $n = 30, N = 500$ 。

---

利用迴圈技巧解決這個問題有別於前面的範例，過程中每一次迴圈所產生的結果  $\bar{x}_k$  必需被保留，才能在迴圈結束後，將所有計算所得的結果拿來畫直方圖。保留迴圈過程所產生的結果也是迴圈技巧的基本動作，下列程式可供參考。而執行結果如圖 5.4 所示。

```
clear
n=30;N=500;           設定參數值，可以並列
X=zeros(1,N);         設定一個 1xN 的零向量，準備置入 N 個樣本平均數
for i=1:N
    x=unifrnd(0,1,1,n); 產生樣本
    X(i)=mean(x);       計算樣本平均並依序置入 X 向量空間
end
hist(X)                直方圖
ecdf(X)                Empirical CDF
```

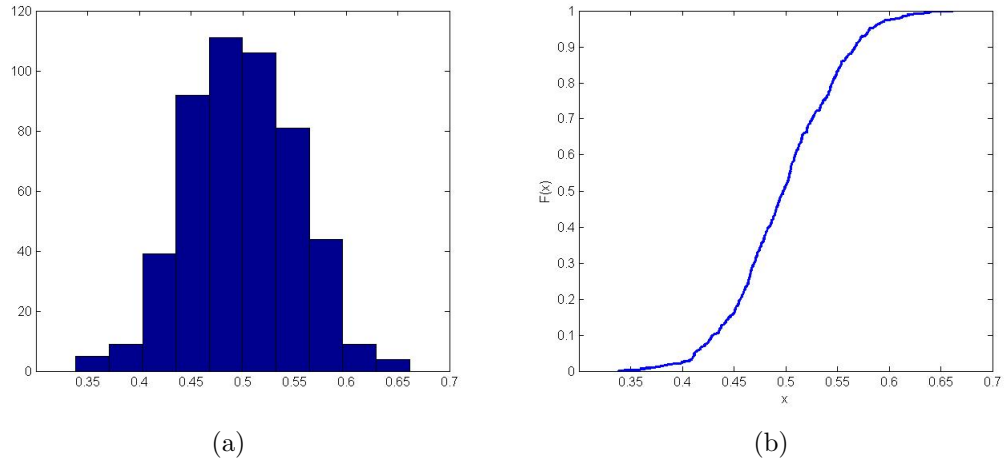


圖 5.4: (a) 直方圖 (b) Empirical CDF。

程式第六行利用預先設定的零向量依序置入樣本平均數，等迴圈跑完，這些平均數仍存在向量  $X$  中，不會消失。在迴圈中執行向量置入的動作前，最好先將這個向量準備好，即第三行所為。就程式執行面而言，是先向記憶體要一個固定的區塊，將來置入數值時，速度會比較快。如果省略這個預設的動作，程式仍會執行，但是會慢很多，<sup>2</sup>尤其是迴圈數多的時候更是明顯。

其實從 MATLAB 程式設計的角度來解決這個問題，並不需要用到迴圈技巧，簡單的矩陣運算便可以解決，請在作業完成這項工作。

---

**範例 7:** 以泰勒展開式表示指數函數  $e^x$ ，寫成

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots + \frac{x^n}{n!} + \cdots$$

寫一支迴圈程式計算  $e^2$ 。

---

類似泰勒展開式的函數有兩個特點，(一) 無限項次 (越後面的項次其值越小)，(二) 有規律的項次。迴圈技巧正適合解決這樣的問題。程式設計不可能執行無限項次的加法，必須以有限的迴圈數計算到最精確的數值。下列程式可供參考。

---

<sup>2</sup>沒有事先向記憶體預定空間，每次儲存時， $X$  向量的大小會逐漸擴增，造成更多記憶體存取的动作 (更換位置)，程式執行上會很明顯的感覺變慢。所以 MATLAB 建議先設定好固定大小的空間。

```

clear
x=2;
N=20;                                迴圈數 (累加項次)
fval=0;                              累加函數值
for i=0:N
    fval=fval + x^i/factorial(i);    進行累加
end
fprintf('The exp(x) evaluated at x=%7.2f is %10.5f ',...
        x, fval)                    結果呈現

```

上述程式設定迴圈數  $N=20$ ，這個作法並不精確，不過以目前學習到的程式技巧，暫時可用。正確的作法是設定更多的迴圈數，設法加入更多的項次，直到預設的精確值到達為止。這牽涉到終止迴圈的技巧，將於後面的單元說明。關於上述程式的結果，可以比較 MATLAB 提供的  $\exp(2)$ 。

上述程式最後一行利用列印指令 `fprintf` 將結果呈現出來，當指令太長時，如果不希望超出目前的畫面，可以強制斷行。只要在切斷處補上三個點「 $\cdots$ 」即可，後面的剩餘部分便可以移至下一行。

### 5.3 觀察

1. 迴圈技巧並非萬靈丹，也不一定是解決問題的首選。有些情形簡單的矩陣便可以輕鬆解決，譬如，計算  $\sum_{k=1}^n x_k y_k$ ，只需將  $x = [x_1 x_2 \cdots x_n]$  向量乘上  $y = [y_1 y_2 \cdots y_n]$ ，即  $xy'$ ，不必一看到  $\sum$  只會想到迴圈。MATLAB 是矩陣運算的高手，寫 MATLAB 程式盡量從矩陣運算的角度來看會比較順手。

### 5.4 作業

1. 利用迴圈技巧設計一個如圖 5.3 的遊戲。
2. 模仿範例 4，畫出 T 分配的密度函數與自由度的變化圖。

## 70 MATLAB 的數學實驗

3. 利用矩陣運算的方式計算範例 6 的  $N$  個平均數。
4. 改寫範例 7 的程式，儲存每個迴圈的累加結果在一個向量變數裡，最後列出該向量，藉此可以協助判斷累加項次  $N=20$  是否合適。

## 第 6 章

# MATLAB 的積分計算與程式設計觀念

積分的數值計算經常出現在數學（統計）問題上。單純的積分問題只需一兩個指令便可輕易解決，複雜的可能需要動用到程式設計的技巧。本單元藉由處理與積分計算相關的問題，帶出一些程式設計相關的觀念與技巧，做為未來解決更複雜問題的基礎。

### 本章將學到關於程式設計

MATLAB的程式結構、程式的邏輯概念與程式檔案的管理。

〈本章關於 MATLAB 的指令與語法〉

操作元 (operators): `.^` `./`

指令: `for`, `tic`, `toc`, `area`, `alpha`, `factorial`, `trapz`, `quad`, `pause`, `inline`

語法: `for` 迴圈的建立與技巧, 匿名函數的介紹 `f=@(x)`。

## 6.1 背景介紹

函數的定積分  $\int_a^b f(x)dx$  可以解釋為函數圖形在直線  $x = a$  及  $x = b$  之間與  $X$  軸所圍的面積, 如圖 6.1 所示的陰影面積。

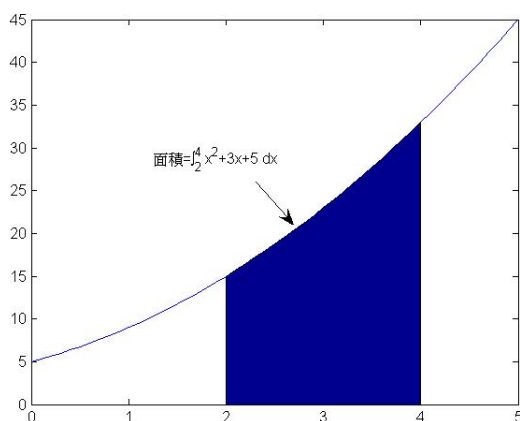


圖 6.1: 積分的幾何意義

這個不規則形的面積也可以所謂的黎曼積分來定義, 適用於數值積分的計算:

**黎曼積分(Riemann Integral) 的定義:**

將一個閉區間  $[a, b]$  任意分割成  $n$  個小區間, 其分割點為  $a < x_1 < x_2 < \cdots < x_{n-1} < b$ , 這些小區間的寬度表示為  $\Delta x_1, \Delta x_2, \cdots, \Delta x_n$ 。這個數值

$$\sum_{k=1}^n f(x_k) \Delta x_k$$

稱為函數  $f(x)$  在這些區間的黎曼和 (Riemann Sum)。當區間寬度  $\Delta x_k \rightarrow 0$  時

$$\lim_{n \rightarrow \infty} \sum_{k=1}^n f(x_k) \Delta x_k$$

稱為函數  $f(x)$  在區間  $[a, b]$  的黎曼積分。



一般定積分的數值計算上多應用黎曼積分的觀念，本單元的探討以此為基礎。為方便說明本單元內容，以下均假設所有小區間的寬度 $\Delta x_k$ 一致，即

$$h = \Delta x_k = \frac{b-a}{n}$$

黎曼和寫成

$$\sum_{k=1}^n f(x_k)h = \sum_{k=1}^n f(a + kh)h \quad (6.1)$$

代表  $n$  個長方形的面積和。圖 6.2 是這黎曼和的示意圖。當  $h$  越小時 ( $n$  越大)，陰影部分的長方形越窄，也因此上緣越貼近函數的曲線。當  $h$  趨近於 0 時 ( $n \rightarrow \infty$ )，這無限多個長方形的面積和即為黎曼積分。

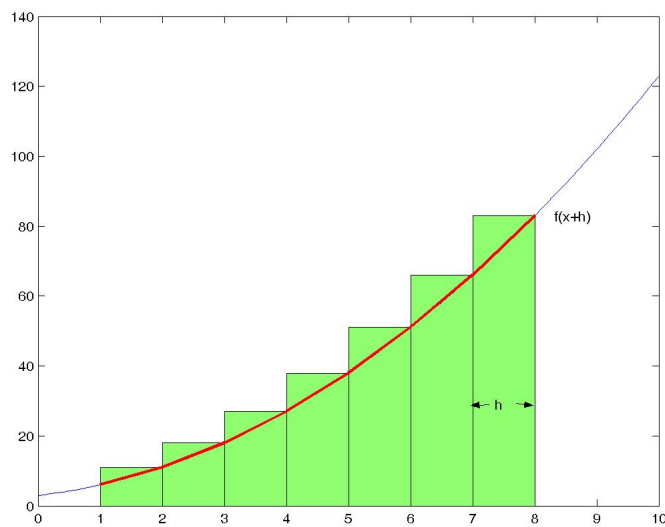


圖 6.2: 積分的定義：黎曼和

## 6.2 練習

---

範例1: 利用上述黎曼和公式(6.1)，寫一支程式計算

$$\int_0^3 x + 3 \, dx$$

其中設  $h = 1$ 。請比較黎曼和與實際的積分值。如果設  $h = 0.5$ ，結果又如何？當  $h = 0.25$ 是不是更接近了呢？

當  $h$  值愈趨近 0 時，數值計算的黎曼和將與實際值愈接近，不過程式的寫作越麻煩，因為要加入的長方形面積越多。此時上個單元介紹的迴圈技巧將可派上用場。

**範例2:** 利用迴圈技術計算範例1的數值積分值，假設  $h=0.01, 0.001$  時。

```
p=[1 3]; % 多項式函數的係數
h=0.01;
x=h:h:3; %所有的 x 值
n=length(x);
A=0; % 黎曼和, 先預設為 0
for i=1:n
    A=A+polyval(p,x(i))*h; %面積累加
end
```

上述程式比較關鍵的地方在  $x$  值的設定，它決定了長方形位置與大小。這些設定與積分的上下限有關，因此比較好的程式寫法，應該納入上下限的變數，上述的程式可以修改為

```
p=[1 3]; h=0.01;
a=0; b=3; % 積分的下限與上限
x=a+h:h:b;
n=length(x);
A=0; % 黎曼和, 先預設為 0
for i=1:n
    A=A+polyval(p,x(i))*h; %面積累加
end
```

為節省空間方便觀看，將一些參數的指令集中在同一行。這個數值積分的方法即俗稱的長方形法。當  $x$  值略作調整，可以做出不同位置的長方形。譬如  $x=a:h:b-h$  或  $x=(a+h/2):h:(b-h/2)$ 。不同的設定是否得到不一樣的結果呢？讀者不妨自己試試看。

利用迴圈的方式，可以順利解決累加的問題，但你是否發現當  $h$  愈小時，所花費的計算時間愈多。別忘了 MATLAB 的專長，為避免過多的迴圈造成較差的執行效率，請利用矩陣的運算解決函數計算與累加的問題，並比較與迴圈技術在計算時間上的差別（ $h$  愈小差別愈明顯）。計算程式執行時間，參考時間指令：`tic`, `toc`，例如

```
範例： 計算  $\sum_{x=1}^5 (x+3)$ 
程式： tic
        x=1:5;
        answer=sum(x+3)
        t=toc %顯示自 tic 後的執行時間
```

Tips: 程式語言的邏輯與數學或一般生活邏輯有些差異，因此寫程式時，千萬別一口氣寫完才執行，這樣會造成除錯上的困難，浪費不必要的時間。對初學者而言，也不需要從第一行開始寫起，可以先寫中間的主體部分或甚至後面的結果，再逐一根據指令或變數的需求穿插補足。過程中，可以隨時儲存程式並執行，確定每個指令與變數的使用都是正確的。

---

### 範例3: 計算定積分

$$\int_{-5}^5 x^2 + 3x + 5 \, dx$$

的黎曼和。建議先畫出函數圖形，再利用迴圈法與矩陣法分別計算積分值，並比較當  $h=0.0001$  時，兩者運算時間的差異。

---

所謂矩陣法就是不使用迴圈的方式，利用向量矩陣的方式直接計算每塊長方形的面積，並加總得到面積和，即

$$\begin{aligned}
 A &= f(-5+h)h + f(-5+2h)h + \cdots + f(5)h \\
 &= (f(-5+h) + f(-5+2h) + \cdots + f(5))h
 \end{aligned}$$

這個面積和  $A$  相當於函數和乘上  $h$ , 以 MATLAB 指令來表示, 可以寫成

```
p=[1 3 5]; h=0.01;
a=-5; b=5;
x=a+h:h:b;
f=polyval(p,x);
A=sum(f)*h;
```

---

**範例4:** 如果能畫出積分計算過程的長方形, 會不會對積分計算的瞭解有幫助呢? 試試看能不能利用迴圈的技巧, 沿著函數的曲線, 畫滿一排長方形。至於長方形的寬度可以從程式來控制。

---

為 MATLAB 圖上的曲線塗上顏色的方式如下:

```
x=[4 6];          註: 準備兩個點的值畫一條直線
y=[5 5];
area(x,y)         註: area 與 plot 的不同在於塗色
axis([0 10 0 10]) 註: 必要改變座標範圍方便觀察
```

最後一個指令 `axis` 用來改變圖形的座標範圍, MATLAB 自行決定範圍有時並不符實際的要求, 矩陣裡的四個數字分別代表 `[xmin xmax ymin ymax]`, 其意義如字面所示。當利用迴圈技術計算面積並繪出面積圖時, 由於電腦速度的關係, 感覺上好像每塊面積是同時畫上去的, 實際上並非如此。此時可以利用 `pause` 指令來緩和速度, 強迫電腦停止運作或停止一定的時間, 譬如在 `area(x,y)` 後面加上

```
pause(1);
```

強迫電腦停止 1 秒鐘，在迴圈裡面呈現出來的效果有如動畫般的神奇。不妨試著結合前一個範例積分程式，配合塗上顏色與時間暫停，玩玩動畫。試試看下面這段程式碼的表現

```
x=0:0.1:7;
f=@(x) polyval([1 3 5],x);%匿名函數的定義
plot(x,f(x))
hold on
for i=1:5
    area([i i+1], [f(i+1) f(i+1)])
end
alpha(0.2)
hold off
```

這裡介紹了一個指令 `alpha`，讓圖形具穿透性，讀者只要將這個指令拿掉，讓程式跑一次，觀察兩者的不同，便瞭解「穿透性」的意思。`alpha` 指令裡面的數字代表「穿透程度」，什麼意思呢？改改它（0 到 1 之間），不就知道了嘛！

上述程式引進了匿名函數的做法，匿名函數可以當作是函數型的變數，經過定義後（如第二行），直接以函數表示的方式計算函數值，如第三與第六行的做法。匿名函數為程式帶來兩個方便：其一，程式看起來更乾淨俐落；其二，程式的表達更接近紙筆的方式。更多關於匿名函數的使用可以參考線上手冊或多留意他人的作品，常有出乎自己意料之外的做法，在寫作程式之餘，憑添樂趣。

---

**範例5：**如上題，將每個小面積從長方形改為梯形，如圖 6.3，並改寫黎曼和的數值定義 (6.1)，重新計算一次。另一方面，請嘗試逐漸降低  $h$  值，觀察計算的結果，與前面長方形面積有何不同？兩個公式孰優孰劣？

---

數值積分的相關計算方法很多，其中的「辛普森法則 Simpson's Rule」比梯形法更為準確。讀者有空不妨去找找看什麼是「辛普森法則」，它的原理及最後的公

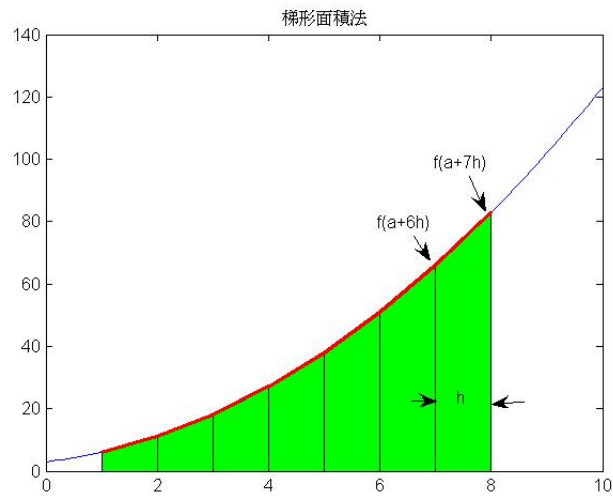


圖 6.3: 梯形面積的黎曼和

式。從長方形或梯形法程式中，經過適當的修改，你也可以輕易的寫成利用辛普森法計算的積分程式。

另外，MATLAB 當然也提供了計算積分的指令，如 `trapz` 及 `quad`，從字面上大約可以猜出是採用了梯形法與辛普森法的積分指令。使用方式如下列程式所示

```
clear
p=[1 3 5]; h=0.01; a=-5; b=5;
x=a:h:b;
y=polyval(p,x);
A=trapz(x,y);
```

```
a=-5; b=5;% 積分下限與上限
f=@(x) x.^2 + 3 * x + 5;
A=quad(f,a,b);
```

這裡的匿名函數將  $x$  當作向量看待，因為後面使用匿名函數計算時，需要給予向量型態的  $x$  值。積分指令 `quad` 根據所定義的函數變數及積分的上下限做積分的計算。

依作者的經驗，在絕大部分的情況下，quad 的表現稱職，所以在積分的計算上，quad 是首選。不過，quad 畢竟是內建的指令，並不操之在我，當這個指令行不通的時候，<sup>1</sup>我們必須有 B 計畫，甚至 C 計畫。積分計算的 B 計畫，trapz 梯形法是不錯的選擇，這個方法不較不受限於函數，是很好的備胎。

---

**範例6:** 計算下列標準常態分配的累積密度函數 CDF

$$P(x) = \int_{-\infty}^x \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}} dz$$

在  $x = -5 : 0.1 : 5$ ，繪製  $P(x)$  的階梯圖 (stairs plot)。

---

這個問題牽涉到多個積分值的計算，因此要考慮迴圈技巧的幫忙。在迴圈過程中，必須儲存每個積分值，留待迴圈結束後畫圖。另一方面，每個積分式的下限是  $-\infty$ ，在實務上必須找一個夠小的替代值，至於多小的值才適合，得要花點時間做些測試，實務面的東西，操作時要靈活些，不能太拘泥，畢竟解決問題才是最終的目標。建議讀者先試著寫寫看，再參考下列程式。程式執行結果如圖 6.4 所示。本題可與指令 normcdf 比較。

```
clear
f=@(x) normpdf(x,0,1);    %定義積分式
x=-5:0.1:5;
n=length(x);
P=zeros(1,n);
for i=1:n
    P(i)=quad(f,-10,x(i)); %以 -10 取代  $-\infty$ 
end
stairs(x,P)                %也可以試著採用不同的圖形，如 plot
ylim([0 1.2])
grid
```

---

<sup>1</sup>內建指令通常無法考慮到所有可能的情況，或許對某些特殊的函數出現計算上的問題，學習程式設計就是要能補這方面的不足，才不會被一個軟體綁架。

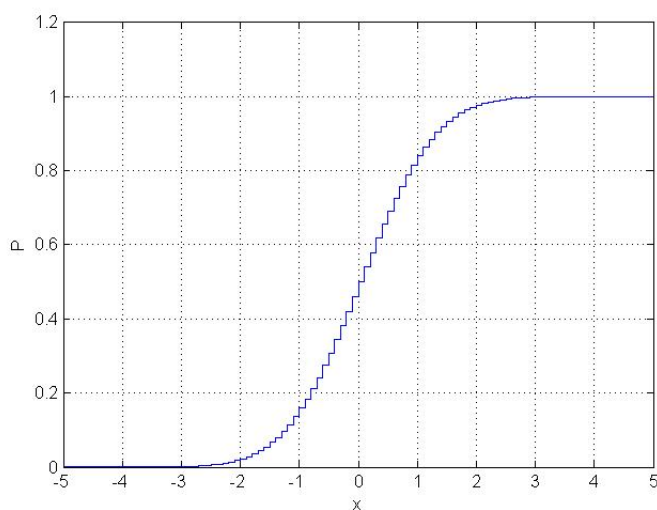


圖 6.4: 積分計算的應用

### 6.3 觀察

1. 數值積分與數值導數一樣都面臨一個  $h$  值的選擇, 但兩者間還是有所不同。  
 $h$  值對數值積分的影響的關鍵因素在哪兒? 與數值導數有很大的不同喔!
2. 長方形面積的黎曼和有三種選擇方式,
  - 其一, 長方形範圍超過曲線, 面積總合將超積分值。
  - 其二, 長方形範圍在曲線下面, 面積和低於積分值。
  - 其三, 長方形範圍介於上述兩種之間。這個方法與梯形法的面積是否較為接近?

### 6.4 作業

1. 改寫黎曼和定義 (6.1), 成為梯形的面積和。
2. 利用長方形面積的黎曼和計算  $\int_{-5}^5 (x^3 + 2x^2 + 3x + 4) dx$  並畫出長方形面積,  $h$  值設定為 0.5。



3. 畫一張圖比較『長方形』法與『梯形』法的準確度。任選一個積分函數與範圍。要怎麼畫？怎麼比較請自行決定。
4. 寫一支程式採梯形法計算下列的積分：程式盡量具彈性，可以讓使用者自行決定上、下限。最好先將圖形畫出來。

- $\int_0^5 100e^x dx$  (ans: 14741.32)
- $\int_0^7 \frac{x^2}{2} e^{-x} dx$  (ans: 1.9407), Gamma(3,1) 分配
- $\int_{-1.96}^{1.96} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx$

5. 已知  $p(\mathbf{y}|\theta)$  為一似然函數，其中  $\mathbf{y}$  及  $\theta$  分別為樣本與未知參數。假設

$$\begin{aligned} p(\mathbf{y}|\theta) &= (2 + \theta)^{125} (1 - \theta)^{38} \theta^{34} \\ p(\theta) &= 1, \quad 0 \leq \theta \leq 1 \end{aligned}$$

計算後驗機率的期望值 (即  $\theta$  的貝氏估計)  $E(\theta|\mathbf{y})$ ，即計算

$$E(\theta|\mathbf{y}) = \frac{\int_0^1 (2 + \theta)^{125} (1 - \theta)^{38} \theta^{34} \theta d\theta}{\int_0^1 (2 + \theta)^{125} (1 - \theta)^{38} \theta^{34} d\theta}$$

6. 計算積分還有一種方式叫做「Monte Carlo Integration,」其原理簡單敘述如后，

$$\int_D f(x) dx = \int_D \frac{f(x)}{p(x)} p(x) dx = E \left[ \frac{f(x)}{p(x)} \right] \approx \frac{1}{N} \sum_{k=1}^N \frac{f(x_k)}{p(x_k)}$$

其中  $p(x)$  被當作機率密度函數 (pdf)，而  $x_k$  是從具  $p(x)$  分配的母體中抽出的樣本值。積分的問題於是可寫成函數的期望值。在決定 pdf 函數  $p(x)$  後，可以利用抽樣的方式，取出相當數量的樣本，再以樣本計算函數值  $f(x)/p(x)$  及其平均數來近似期望值。其準確度與  $p(x)$  的選擇、樣本數大小都有關係。當然，當樣本數趨近無限大時，不論選擇的  $p(x)$  為何，總能逼近完美的積分值。但適當的選擇卻可以以較少的樣本得到最準確的結果。原則上  $p(x)$  的選擇與接近函數  $f(x)$  的外形為佳。

## 82 MATLAB 的數學實驗

利用此法, 重做作业4的積分問題, 需說明使用的  $p(x)$  與樣本數, 譬如計算  $\int_0^7 f(x)dx$ , 當選擇均等分配為其 pdf 函數時, 問題變成

$$\int_0^7 f(x)dx = E \left[ \frac{f(x)}{\frac{1}{7}} \right] = 7E[f(x)] \approx \frac{7}{N} \sum_{k=1}^N f(x_k)$$

其中  $x_k$  為樣本,  $N$  為樣本數。

## 第 7 章

# MATLAB的副程式

副程式 (function) 的應用在一般程式語言裡非常的普遍, 幾乎是必備的技巧, 只要稍具複雜度的程式多半會使用到副程式。當程式出現重複的程式片段, 或是該程式片段過長, 佔據過多篇幅, 妨礙了程式的流暢與可讀性時, 可以將該程式片段移出, 另置他處 (在同一程式檔案或其他檔案), 稱為副程式或子程式 (subroutine)。副程式除了可以減少主程式的程式碼, 提高主程式的可讀性外, 有時候也可以供其他程式呼叫, 提高程式寫作的效率與一致性。另外 MATLAB 在 7.0 版以後新增「匿名函數」功能, 減輕副程式的帶來的檔案管理問題, 對於小型的副程式提供更簡潔的作法。

### 本章將學到關於程式設計

MATLAB的程副式與匿名函數的結構與應用。

〈[本章關於 MATLAB 的指令與語法](#)〉

指令: function

## 7.1 背景介紹

主程式與副程式間的「溝通」靠參數 (arguments) 的傳遞來聯繫。<sup>1</sup>主程式將一些給定的參數值傳給副程式作為程式執行所需, 副程式執行完畢, 將結果以另一組參數傳回給主程式。副程式的角色相當於主程式的「協力廠商,」主程式將部分工作「outsource」出去給副程式做。而副程式扮演「協力廠商」的角色, 通常只做些單純的工作, 因此可以設計成為更「通用」, 為更多不同的「主程式」服務。

副程式的結構隨程式語言的不同有些差異, 請參考 MATLAB 對於副程式的定義, 以下舉個簡單的範例說明 MATLAB 副程式的結構與使用方式:

副程式:stats.m

```
function [mu,sigma]=stats(x)
mu = mean(x);           %計算平均值並且放入輸出變數 mu
sigma = std(x);          %計算標準差並且放入輸出變數 sigma
```

主程式:main.m

```
x=chi2rnd(2,1,100);      %產生任意的資料
[mu, sigma] = stats(x);  %呼叫副程式 stats, 並且傳入一個資料參數 x,
                          %傳回結果放在 mu 及 sigma 的變數裡面。
                          %供後續程式使用。
```

通常副程式的函數名稱 (function name) 與檔名相同。一般而言一個副程式檔案內含一個副程式, 但必要的時候, 也可以超過一個副程式, 副程式之間彼此呼叫。此外, 副程式與主程式間溝通的管道靠輸入與輸出變數, 其個數視需求而定, 可以超過一個, 當然個數太多時, 寫起來很冗長不方便, 可以考慮採結構性 (structure) 的變數, 彈性也更大。關於結構性的變數如何設定與使用, 請參考手冊。

另外, 副程式的執行過程與主程式完全不相干, 變數名稱也可以不同 (雖然上述的範例中主、副程式所使用的參數名稱相同), 輸入與輸出變數依出現的次序為對應

<sup>1</sup>作為主、副程式聯絡的變數一般稱為參數, 分為輸入參數 (主程式傳遞給副程式) 及輸出參數 (副程式傳遞給主程式)。實質上這些參數在主、副程式裡面就是變數。

關係，並非依變數名稱。在副程式內的任何變數也是獨立的，與主程式無關，並不影響主程式的變數或命令視窗的變數。

## 7.2 練習

---

**範例1:** 寫一支程式分別自「常態」、「卡方」與「二項」分配的母體中抽取100樣本，並分別計算其「樣本均數」、「標準差」、「中位數」、「最大值」及「最小值」。其中計算的部分以副程式執行（將上述的副程式擴充即可）。

---

上述的範例是副程式使用的典型，其優點讓主程式看起來比較「乾淨清爽」，也可以提供不同的程式呼叫。另一種副程式的使用時機如範例2的積分函數所示：

---

**範例2:** 使用 MATLAB 提供計算數值積分的指令 *quad*，計算  $\int_0^2 \sin(x)dx$ 。

---

其使用方式有以下三種典型：

方式1	方式2	方式3
<code>quad('sin', 0, 2)</code>	<code>F = inline('sin(x)'); quad(F, 0, 2);</code>	<code>quad(@my_int_fun, 0, 2);</code>

其中方式3,@之後的`my_int_fun`代表一個副程式的名稱。該副程式寫成：

```
function y=my_int_fun(x)
y=sin(x);
```

上述三個方式中，方式1與2適用於當欲積分的函數很單純時。若函數複雜，以副程式的方式來描述積分函數是比較恰當，甚至是唯一的方式。MATLAB 在7.0版以後，提出「匿名函數」(anonymous function) 的方式，免除副程式需要建立一個檔案的麻煩，卻可以提供重複使用的方便性，類似上述方式2的作法，但更具彈性。以上述的積分為例，寫成

```
f=@(x)sin(x);
quad(f,0,2);
```

f 代表 sin 函數，也可以拿來計算函數值，譬如，f(pi/2) 計算  $\sin(\pi/2)$ 。

試試看以下稍複雜些的積分式。

---

範例3: 模仿範例2的三種方式分別計算:

- $\int_0^5 100e^x dx$  (ans: 14741.32)
- $\int_0^7 x^2 e^{-x} dx$  (ans::1.94)
- $\int_{-1.96}^{1.96} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx$

---

匿名函數或 inline 式的副程式還可以提供額外的參數，增加函數使用的方便性。

舉多項式函數的計算為例:

```
f=inline('polyval(p,x)');
p=[1 -3 2];
f(p,3);
pp=polyder(p);
f(pp,3);
```

上述的範例分別計算  $f(3)$  與  $f'(3)$ ，其中  $f(x) = x^2 - 3x + 2$ 。inline 指令中放進了一個參數  $p$ ，隨著  $p$  的不同， $f$ 代表著不同的多項式函數。

### 7.3 觀察

1. 副程式的使用除了解決程式碼重複的問題外，也提供一個可重複使用 (re-useable) 的程式，提供給別的程式呼叫。但副程式的本身也製造出管理的問題，太多的副程式往往讓程式設計師本身迷航了。所以什麼樣的程式碼適合編成獨立的副程式，必須認真思考，包括長度，可重複使用的頻率。如果不合乎這些法則，寧願還是以程式碼的方式存在原程式中。

2. MATLAB 自 6.5 版後加入「程式加速」的機制，在合乎某些條件下，即使使用多層的迴圈 (for loop)，MATLAB 的加速機制依舊可以適當的轉換程式碼，加速程式的執行。不過副程式的呼叫卻違背這些條件，讓加速機制失效。因此，當程式的執行時間很關鍵，或是程式耗費太多時間，都要考慮限制副程式的使用。

## 7.4 作業

1. 將之前寫過計算多項式微分的程式改成副程式的結構，使其使用上像呼叫 MATLAB 的指令。
2. 將之前寫過多項式積分的程式改成副程式的結構，使其使用上像呼叫 MATLAB 的指令。
3. 寫一個標準的副程式，專用來計算多項式的函數值 (功能像 *polyval*)。輸入參數定義為多項式的係數。