Lecture Note in Experimental Physics
Scientific Writing & Programming

Chih-Han Lin 林致翰
clin@ltl.iams.sinica.edu.tw

High-Field Physics and Ultrafast Technology Labotary
Department of National Central University
Institude of Atomic and Molecular Science, Academica Sinica, Taiwan

September 15, 2011

第二版編者序

這份講義最初的版本在 2009 年的夏天完成，除去首章文書編輯軟體的介紹，主要的重點放在如何透過電腦程式的撰寫來幫助我們模擬一些常見的物理現象，透過繪圖的方式來呈現實驗數據並加以分析。

科學計算是現代的實驗物理工作者不可缺乏的工具，使用人工統計大量的實驗數據極困難且浪費時間，透過各式各樣的電腦軟體我們可以有效率地處理這些數據，甚至利用各類演算法來驗證，甚至預先模擬不同情況下可能發生的實驗結果。在許多大型且昂貴的實驗中，我們很難透過試誤法 (trial by error) 收斂出最佳結果，必須先透過電腦先行模擬，在龐大的參數空間過濾出幾個可能的方向再開始實驗。

即便是理論學家，也能透過相當聰明的符號式計算工具 (例如 Maple 與 Mathematica) 來簡化繁雜的符號運算，利用電腦來驗算推導過程並且可以利用數值方式將不可解析的抽象結果畫成具體的圖表來分析其中的趨勢與隱含的物理。

本書預設的讀者爲大學部的物理系學生，大多數範例僅涉及微分方程，需要較多背景知識的數值模擬範例則放到附錄之中，有興趣的同學可以加以閱讀。另一方面，這份講義以大學部必修課程的參考資料爲目標進行撰寫，考慮到軟體建置經費的緣故，筆者以自由軟體爲媒介來實作各種在一般研究中可能遇到的數值問題與數據分析(本書提到的絕大多數軟體都有 Linux 平台的版本可執行，爲了降低入門難度，仍以 Window 版本進行安裝介紹)，或許計算的速度不是最快，卻是以對初學者最友善的方式來講解。本書採用 Scilab 這個與 Matlab 系出同源的直譯式計算軟體作爲講解範例(在 Linux 平台也有類似的 Octave 可用，不過圖形化介面與繪圖介面的整合度不若 Scilab 來的好)，一方面前者指令與 Matlab 這個許多實驗室使用的商用軟體極爲類似，將 Scilab 上學到的概念轉換到 Matlab 上並不困難，一方面 Scilab 這類直譯式軟體對於語法的要求較爲寬鬆，入門難度較低，在繪圖方面的便利度遠勝其他傳統程式語言。

本書初稿第一次用於教學時，學生最大的反應便是爲何不使用 Matlab 這個 user-friedly 作爲計算平台，市面上有關 Scilab 的書太少了。這是無法改變事實，自由軟體的文件支援很難贏過有大型專業團隊維護的商業軟體，在市場熟悉度尚未打開的的情況下，許多優秀的自由軟體缺少足夠的中文化說明文件來吸引更多人使用，Scilab 就是其中的一個例子。但是缺少中文解說不代表沒有出色的英文導覽可以閱讀，細心搜索，還是可以在網路上尋得不少免費且具參考價值的電子書。我個人認爲 scilab 的功能足以涵蓋大學部學生可能遇到的各類數值問題與數據分析，假如有運算速度的瓶頸，它也可以與 c/c++ 和 fortran 等程式語言作連結，只是各位可能不知道或者沒有用到。

目前 Scilab 最大的不足在於圖像處理與繪圖控制器的部份，不比 matlab 直覺，且套件的限制較大。比如說如果你懂 jpg 或 bmp 的編碼格式，依然可以用 scilab 來處理 16bit 的圖檔，只是需要多花時間來讀懂 datasheet 並撰寫中介程式。這就是商用軟體爲何可以賣這麼貴的原因，不是嗎？筆者由衷地希望各位在使用軟體的時候要多些自覺：這些東西並沒有想像中地那麼不值錢。許多人不喜歡自由軟體，那是因爲他沒計算到正版軟體的價格因素，台灣人對於盜版太過習以爲常了。考量到 C/P 值，這些免費的軟體提供了很大程度的折衷，既省了荷包，又可以作到想做的事情，只是犧牲了一些便利性。

爲了彌補 Scilab 在影像處理上的不足，本書新版中加入這套由 美國衛生研究院 (NIF) 開發並公開原始碼的軟體 ImageJ 的介紹，直觀的 GUI 介面與衆多使用者開發的外部套件讓這套軟體可以統計多種格式的圖檔中物件間距、角度等二維資訊，甚至可以批次處理連續影像的分鏡，有效率地完成物體追跡等多維度資訊。

考量到市面上 Scilab 的教材缺乏，本書往後的更新在以 Scilab 作爲基礎範例的段落會追加與 matlab 對應的原始碼與提示，讓各位有機會從類似的指令查找相關的書籍來驗證所學到的概念。同時將靜態資料不容易的呈現的軟體動態操作製作成影片放至 Youtube 上作爲參考資料，最終達成不需要他人講解也能透過閱讀本書與相關資料而有效地學習科學計算。

本書以 XeLateX 編譯完成，感謝中央物理系 102 級的全體學生在本書爲主軸進行教學帶來的回饋與啓發，以及當時的助教群兼摯友：黃揚智、廖孜桓與林昱廷協助筆者進行如此新穎近乎瘋狂的物理教學工作。

September, 2011 林致翰

第一版編者序

第一學期的講義內容由林致翰與林昱廷根據汪治平老師規劃的課程大綱進行編寫。其中部份機械加工的章節，GUI 介面的主要內容與函數繪圖中非線性映射的範例由林昱廷完成，其餘所有章節則由林致翰編寫完成。

機械加工的部份以助教帶領同學實機操作爲主，這份講義僅列出安全規則與一些小訣竅，比較深入的內容請同學參考指定書目的內容來學習。程式寫作課程則考量到時間上的限制與軟體是否容易取得，這門課選擇了 Scilab 來展示電腦如何幫助我們理解物理問題，筆者相信若同學們能充分掌握這份講義中的程式範例與習題，未來在學習更爲抽象的物理概念時也能透過電腦的幫助來加深你的思考深度。

實驗課中安排了一個樹脂翻模的課程，或許這是一般物理系學生絕少會碰觸到的主題；也許這樣的技術所能提供的精度與效率並不能好好地用在實驗室上，但它代表實驗物理的一種精神：自製儀器時中不可避免的反覆檢討，逐漸將成品修改到最佳狀態的一種過程。

這本講義的另外一個特色便是習題衆多，當然預設上並不要求同學們把所有的題目都做完，以三學分的實驗課而言能完成 50% 的內容就算是及格了。設計大量題目的用意在於讓不夠過癮人有更多練習寫程式的機會，眞正的學習應該是 self-learning 才對，而非侷限在上課時台上講解的內容。

September, 2009 林致翰

備註：原書中機械加工與翻模課程的章節在新版中已刪去另成一册。

Contents

Chapter 1

科學文件寫作平台

> Herodotus relates that Xerxes wept at the sight of his army, which stretched further than the eye could reach, in the thought that of all these, after a hundred years, not one would be alive. And in looking over a huge catalogue of new books, one might weep at thinking that, when ten years have passed, not one of them will be heard of.
>
> Arthur Schopenhauer

1.1 OpenOffice

OpenOffice 是一套免費且功能強大的文書編輯軟體，你可以從 http://zh.openoffice.org/new/zh_tw/ 下載此套軟體。之所以建議各位使用 OpenOffice 作爲寫作平台，一方面在於 OpenOffice 有很好的 LaTeX 外掛模組可用(OooLatex)，能夠排版高水準的方程式內容；另一方面 OpenOffice 的檔案可以直接輸出成 pdf 檔（ Portable Document Format ）。pdf 具有字型內嵌，保有原文件格式的優點，不會發生如一般的 MS Word 文檔時常換台電腦開啓，圖片與段落便有很大機會發生錯亂的窘境（有的時候是範本設定的問題，更多情況是使用者不懂得設定格式，總以爲文字圖片隨意拉到的滿意的位置就完成排版的緣故）。除去含特效的投影片檔案，完稿形式的報告電子檔轉存成 pdf 格式流通可以最大程度讓讀者獲取你預設的版面格式而不產生誤解。

免費且好用的 pdf 的閱讀程式，有些甚至支援簡單的修改與註解功能，以下列出幾個常見

且效能極佳之軟體供參考:

Acrobat Reader `http://get.adobe.com/tw/reader/`
Foxit Reader `http://www.foxitsoftware.com/pdf/reader/`
PDF-XChange Viewer `http://pdf-xchange-viewer.en.softonic.com/`

Openoffice 中的 writer 使用起來類似於 MS office 的 Word，編輯文件的時候請善用 格式→ 樣式與格式(或按快捷鍵 [F11])，預先將助教指定的報告格式（標題字內文字體大小）建成標準樣式，往後新增標題與內文可以減少更改字型格式的繁瑣操作。欲排版雙欄格式可以利用 插入→ 區段→ 欄，常用的指令有:

- 要換頁請利用 Ctrl + Enter，別一直按 Enter 換行。
- 修改頁面樣式: 格式→ 頁。
- 加入圖片標題: 選取圖片， 插入→ 標籤。
- 輸出 pdf 檔: 檔案→ 匯出成 pdf

1.2 LaTeX

"It has often been said that a person does not really understand something until he teaches it to someone else. Actually a person does not really understand something until after teaching it to a computer, i.e., express it as an algorithm."

『常言道: 學而能教謂之通。事實上，我認爲能教授電腦做出一樣的事情才意味著眞正的瞭解，這樣的過程也就是演算法的實作。』

Donald Knuth, in "American Mathematical Monthly," 81

TeX 排版系統最早是由 Stanford 大學的教授 Don Knuth (有個正式的中文名字，高德納) 發展。七零年代末， Knuth 著手計劃出版一套電腦科學界的巨著 "The Art Of Computer Programmnig "（此書曾被 "American Scientist"《科學人》選爲與 Dirac 所著之量子力學， Einstein 的相對論並列的二十世紀最佳學術專著），誰知道該系列第二册付梓時，Knuth 便被印刷廠送刷樣本中慘不忍睹的數學式弄得火冒三丈。氣憤之餘，Knuth 乾脆停下手邊的寫作進度，用了近八年的時光開發出一套基於 literate programming 概念，專門用於科學文件排版的系統，也就是 plain TeX，並包含一套名爲 Metafont 的向量字體描述語言。TeX 使用上的便利導致越來越多的使用者加入開發以TeX 爲核心的應用套件，今日我們常用的 LaTeX 系統便是 Lamport 以 TeX 爲基礎，額外加入許多套件而釋出的加強版 TeX。

許多出版社和科學期刊也開發自己專屬的 LaTeX 範本 (比如說美國物理學會 APS 的 RevTeX4)，利用 LaTeX 作爲完稿的排版工具，對於將來想從事學術研究的理工科同學而言學會 LaTeX的操作就像是擁有一個強大的工具，可以不假他人之手，以相對少的時間完成高品質排版效果且滿足格式要求的科學文件。 LaTeX 將排版的行爲簡化成穩定且有規則的指令式排版環境。 事實上，這本講義也是透過 LaTeX 編譯完成的作品。早期 LaTeX對於非英文語系的國家較不友善，需要透過一些非正式的字碼轉換 macro 來達成多國語系文件編纂的目的。近年來 LaTeX已經有相當不錯的 unicode 套件，字型與套件的安裝較以往容易許多，降低入門難度。

1.2.1 Miktex

在 WINDOWS 系統上最常使用的 TeX distribution 是 Miktex ，可以從 `http://miktex.org/` 下載並進行安裝 (目前已更新至 2.9 版，需注意的是新的版本釋出以後，舊版本的套件庫更新會在一段時間後停止支援) 。除了 TeX 主程式以外，通常還需要配合專屬的文字編輯器來使用才能事半功倍。由於 TeX有一定的入門難度，對於圖片的處理也不似一般的文書編輯體直觀，使用 OpenOffice 配合專用外掛來調用 LaTeX 的數學式編輯功能會是讓初學者接受度較高的作法，或者利用一些線上 LaTeX 數學式編輯網頁輸出方程式圖檔轉貼至 office 上。等到熟悉 LaTeX 的操作之後，想要追求更完美的排版效果時再進入純 LaTeX 編輯環境便能架輕就熟，減低學習挫折感。

1.2.2 TeXworks

新版的 MikTeX 中已內建一個優秀的文字編輯器 TeXworks，其圖形化介面已經嵌入數種 LaTeX 編譯器的快捷鍵，不需要額外的設定便能編譯寫好的 LaTeX 文檔。

1.2.3 Online LaTeX Math editor

可輸出各種圖片(png, pdf, svg, emf) 格式的 LaTeX online editor
`http://www.codecogs.com/latex/eqneditor.php`

對於只想使用 LaTeX 編輯數學式，卻不想安裝 LaTeX 系統的使用者而言，利用網際網路上其他人架設好的伺服器來輸出數學式圖片十分方便。若想將方程式圖檔匯入向量編輯軟體如 Inkscape, CorelDraw 等等可選擇下載成 svg 檔，若是希望將數學式匯入 MS powerpoint 作成的投影片或 office 相關的文書軟體則可以轉成 emf 檔。

若想將方程式匯入僅接受點陣圖檔的程式，轉存成 300 dpi 的 png 檔會是不錯的處理方式，或者儲存成 pdf 檔，透過 pdf 閱讀程式的照相機功能存取更高解析度的方程式結圖。以 Acrobat Reader X 爲例，編輯→ 偏好設定→ 一般 中勾取使用固定的解析度來顯示快照工具節舉的影像，調整右欄位的像素/英吋 (即 dpi)，即可使用最高 720 dpi 的解析度將 pdf 檔的內容轉換成點陣圖儲存至剪貼簿中。

1.2.4 OOoLaTeX*

LaTeX 之所以令人愛不釋手，有一半的原因來自於它超凡入聖的方程式排版功能，其完稿呈現美觀之至，絕不是一般的文書編輯軟體可匹敵的。OpenOffice 有個輔助程式，名曰 OOoLaTeX ，能以簡單的方法呼叫 LaTeX 編輯方程式，若有安裝指定的數學字型，還可以輸出向量格式（無失眞）的方程式內容，扣除 compile 造成的少量時間延遲（每個獨立的 equation 都要分別 compile 一次），不失爲揉合 LaTeX 與一般文書編輯軟體便利性的優秀方案。

要利用 OpenOffice 輸出 LaTeX 方程式區塊，你必須安裝這些程式

- OpenOffice
- OOoLaTeX (from `http://ooolatex.sourceforge.net/`)
- Miktex or other distribution
- MSYS (for OOoLaTeX , `http://www.mingw.org/wiki/msys`)
- Ghostscript (from `http://pages.cs.wisc.edu/~ghost/`)
- math fonts (from OOoLaTeX website)

安裝完 OOoLaTeX 後，我們會發現 OpenOffice 工具列左端多出 OOoLaTeX 的 button；要先在 Config 中設定 Latex，Ghostscript 以及 MinSYS (MSYS) 的執行檔位址才能插入方程式。在前述軟體使用預設目錄安裝的情況下應該要填入:

- Latex: `C:\ProgramFiles\MiKTeX2.7\miktex\bin\`
- Ghostscript: `C:\ProgramFiles\gs\gs8.60\bin\`
- MinSYS: `\C:\msys\1.0\bin\`

1.2.5 LaTeX 參考資料

吳聰敏，吳聰慧教授的 cwTeX 使用手册

http://homepage.ntu.edu.tw/~ntut019/cwtex/cwtex.html

雖然本書主要是介紹吳教授撰寫之中文 TeX 轉換外掛 (以外加轉換字集方式讓原本非 unicode 的 TeX 系統可輸出 Big5 文件，事實上在支援 unicode 的 LaTeX 出現之後，亞洲語系的使用者不再需要這類繁瑣的轉換程式，可以直接編寫中文 TeX 文件，比方說這份講義的初稿就是透過 CJKutf8 package 完成，新版則加入少許指令改用用 XeLaTeX 編譯)，但其中的排版概念也相容於原本的，書中第九章講解數學式繕寫的部份對初學者而言相當親切，廣而足的內容也足以應付勝任方程式輸入的 FAQ，是很出色的參考資料。

美國數學協會 (AMS) 的 Short Math Guide for LaTeX

符號速查表

http://amath.colorado.edu/documentation/LaTeX/Symbols.pdf

colorado's LaTeX online tutorials

http://amath.colorado.edu/documentation/LaTeX/tutorial/

1.3 LaTeX 數學式

1.3.1 基本操作

LaTeX 中所有指令皆以反斜線 \ (通常位於 ⟵ 與 Enter 鍵之間) 開頭，指令區分大小寫，若需輸入參數則於該指令後加入大括號 { } 。比方說在文檔中輸入 \LaTeX{} 會輸出成 LaTeX，要在文中排版數學符號則必須同時使用隨文數式環境: $\sqrt{\pi}$ 會排版成 $\sqrt{\pi}$。有些指令會需要多個參數，比如說分式指令 \frac ，此時搭配的大括號不只一組:

```
$\frac{1}{2}$
```

$\frac{1}{2}$

各個指令可巢狀套用，換言之某個指令大括號內部可以再放入一組新的指令:

```
$\sqrt[3]{\frac{1}{2}}$
```

$\sqrt[3]{\frac{1}{2}}$

有些指令會在大括號加入一組方括號 [] 來標示進階參數設定，通常方括號標示的參數可以不輸入(取用內定值)，比方說上式的根號指令\sqrt 指令可以加入 [<n>] 來改變根號左上方的次數，若不輸入任何則預設爲空白，即一般的平方根。

除了反斜線與大括號以外，LaTeX 還有許多有意義保留字元，這些字元通常需要以指令的方式輸入才能正確顯現(若爲 unicode 中文系統則可直接輸入全形標點):

\%⟶%, 註解字元	\&⟶&, 表格分欄字元
\#⟶#, 巨集指令參數	\$⟶$, 隨文數式提示字元
\verb+~+ ⟶ ~, 加入空白	\verb+^+⟶ ^, 數式上標
_{}⟶_ , 數式下標	

指令名稱的大小寫不能隨變更換，比方說 Ω 會顯示成 Ω，而 ω 會顯示成 ω。LaTeX 文稿會自動計算空格與換行，換行等同於空格，空許多格相當於空一格，換行兩次以上代表另起段落，換兩行以上相當於換兩行，同時我們也可以利用一組雙反斜線 \\ 來代表換行，用波浪號 [~] 來手動加入空格 。 值得注意的地方在於 LaTeX 在數學式環境中不能隨意換行，否則編譯時會出現錯誤。

1.3.2 數式環境

基本的數學指令可以參考吳聰敏，吳聰慧教授的 cwTeX 使用手册第九章。LaTeX 的數學式環境主要可分爲以下三種:

- 隨文數式 (inline): 以一對 [$] ([Shift ⇧] + [4]) 符號夾住數學敍述: $<equation>$，可直接將數學式嵌入文句中，例如 $\sqrt{\pi}$ 會顯示成 $\sqrt{\pi}$。很多時候隨文數式的高度超過行高而自行壓縮符號大小，例如 $\sqrt{1}{2}$ 會顯示成 $\frac{1}{2}$ ，這個時候可以加入 \displaystyle 強迫輸出正常比例的數學式，並自動加大行高，例如 $\displaystyle\frac{1}{2}$ 會顯示成 $\displaystyle\frac{1}{2}$。使用 \displaystyle 雖然可以正確顯示數學式字體比例，更多時候卻會破壞整體段落的美觀，因此對於簡單的分式來說我們會將它改寫成橫式 $1/2$ $1/2$，既不破壞行高 也能正確表達式子。若數學語句太過於複雜而不能改成橫式，我們則傾向將之獨立成行。

- 無編號單行展示數式 (display): 數學式獨立成行，一般而言需要使用 \begin 指令進入數式環境，例如

```
\begin{displaymath}
\sum_{k=0}^{n}\frac{1}{n}=1+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{n}.
\end{displaymath}
```

$$\sum_{k=0}^{n}\frac{1}{n}=1+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{n}.$$

上述數式環境指令可以用 \[<equation> \] 或一對雙錢號 $$ <equation> $$ 的簡化版指令來取代。複雜結構的連分式也同樣可以使用 \displaystyle 來還原結構

```
\begin{displaymath}
\frac{1}{1+\frac{1}{1+\frac{1}{1+\cdots}}}.
\end{displaymath}
```

$$\frac{1}{1+\frac{1}{1+\frac{1}{1+\cdots}}}.$$

```
\begin{displaymath}
\frac{1}{1+\displaystyle\frac{1}{1+\displaystyle\frac{1}{1+\cdots}}}.
\end{displaymath}
```

$$\frac{1}{1+\displaystyle\frac{1}{1+\displaystyle\frac{1}{1+\cdots}}}.$$

- 編號或多行展示數式。通常使用多行編號數式環境 \begin{equation} ... \end{ equation}、多行編號數式環境 \begin{eqnarray}...\end{eqnarray} 或者引用 amsmath 套件的多行數式環境 \begin{align}... \end{align}。原則上建議各位不論排版多行或者單行編號數學式，皆使用 align 環境， eqnarray 雖然是 LaTeX 內建之環境，但是排版時會刻意加寬第一個等號兩邊的距離，有時候並不是非常美觀，比如下面的例子:

```
\begin{eqnarray}
a_1&=&k_1=b_1+c_1,~~~~d_1=e_1+f_1,\\
a_2&=&k_2=b_2+c_2,~~~~d_1=e_1+f_1.
\end{eqnarray}
```

$$a_1 = k_1 = b_1 + c_1, \quad d_1 = e_1 + f_1, \tag{1.1}$$
$$a_2 = k_2 = b_2 + c_2, \quad d_1 = e_1 + f_1. \tag{1.2}$$

eqnarray 採用在需對齊符號的兩邊加上分隔符號 & 來做對齊，加上雙反斜線換行，沒辦法處理單行多條方程式並列的對齊，只能靠手動加空格來對齊。有些時候過長的方程式拆行對齊的時候不見得在等號的位置，這個時候 eqnarray 預留的空白會變得相當難處理。如果將上面的例子改用 align 來做:

```
\begin{align}
a_1&=k_1=b_1+c_1, & d_1&=e_1+f_1,\\
a_2&=k_2=b_2+c_2, & d_1&=e_1+f_1.
\end{align}
```

$$a_1 = k_1 = b_1 + c_1, \qquad d_1 = e_1 + f_1, \tag{1.3}$$
$$a_2 = k_2 = b_2 + c_2, \qquad d_1 = e_1 + f_1. \tag{1.4}$$

要對齊的地方用一個分隔符號 & 分開即可，若有單行多組方程式需對齊，在方程式之間使用 & 隔開即可。若想要讓多行方程式無編號，只要將 align 改爲 align* 即可，若僅有某幾行不需要編號或者另外給定編號，可以使用 \notag 與 \tag 指令:

```
\addtocounter{equation}{+1}
\begin{align}
\notag a_1&=k_1=b_1+c_1, & d_1&=e_1+f_1,\\
a_2&=k_2=b_2+c_2, & d_1&=e_1+f_1.
\tag{\Roman{chapter}.\arabic{equation}a,b}
\end{align}
```

$$a_1 = k_1 = b_1 + c_1, \qquad d_1 = e_1 + f_1,$$
$$a_2 = k_2 = b_2 + c_2, \qquad d_1 = e_1 + f_1. \tag{I.5a,b}$$

使用 \notag 與 \tag 會讓該行方程式不列入編號，換言之紀錄方程式編號的 equation 變數不會增加，所以我們用了 \addtocounter{equation}{+1} 手動讓編號加一；最後 \tag 指令則依序打印章編號、方程式編號以及自行定義的追加內容，\Roman 選擇編號以大寫羅馬數字顯示，\arabic 則是選擇將編號以阿拉伯數字顯示。

Ex 1.3.3

常用字元與分式 \frac 練習

```
\[\nabla^2\Phi=\frac{1}{r^2}\left[\↩
   frac{\partial}{\partial r}\left(↩
   r^2\frac{\partial\Phi}{\partial ↩
   r}\right)+\cdots\right].\]
```

$$\nabla^2\Phi=\frac{1}{r^2}\left[\frac{\partial}{\partial r}\left(r^2\frac{\partial\Phi}{\partial r}\right)+\cdots\right].$$

LaTeX 可以利用 \left 與 right 自動調節括號包含( ), [ ], { } 與 ⟨ ⟩ 的大小，前文出現 \left 則文後必然出現一個 \right 與之配對，否則編譯文檔時會出錯。有些時候我們希望僅顯現單邊的符號時該如何？ 我們可以利用 \left. 或 \right. 來配對，以其替代掉不希望出現的某邊括號，請參考以下範例:

```
\[\left.\frac{d^2x}{dt^2}\right|_{x↩
   =1}\]
```

$$\left.\frac{d^2x}{dt^2}\right|_{x=1}$$

Ex 1.3.4

學習 \array 的用法。

```
\[F_{ij}=\left(
\begin{array}{cccc}
0&B^3&-B^2&cE_1\\
-B^3&0&B^1&cE_2\\
B^2&-B^1&0&cE_3\\
-cE_1&-cE_2&-cE_3&0\end{array}
\right)\]
```

$$F_{ij}=\left(\begin{array}{cccc}0&B^3&-B^2&cE_1\\-B^3&0&B^1&cE_2\\B^2&-B^1&0&cE_3\\-cE_1&-cE_2&-cE_3&0\end{array}\right)$$

或者可以使用 pmatrix 的指令，必須注意的是這個指令必須先在文稿全文設定區加上 \usepackage{amsmath} 引入 amsmath 套件方能使用。將關鍵字 pmatrix 改爲 bmatrix, Bmatrix, vmatrix 或 Vmatrix 可以改變矩陣括號形式，請讀者自行練習。

```
\[F_{ij}=
\begin{pmatrix}
0&B^3&-B^2&cE_1\\
-B^3&0&B^1&cE_2\\
B^2&-B^1&0&cE_3\\
-cE_1&-cE_2&-cE_3&0\end{pmatrix}\]
```

$$F_{ij} = \begin{pmatrix} 0 & B^3 & -B^2 & cE_1 \\ -B^3 & 0 & B^1 & cE_2 \\ B^2 & -B^1 & 0 & cE_3 \\ -cE_1 & -cE_2 & -cE_3 & 0 \end{pmatrix}$$

Ex 1.3.5

以底括號指令 \underbrace 配合 \mbox 插入正體說明文字，以手動字體放大指令 \big 或\Big 手動調整左右括號的大小。

```
\[H'_{ng}(t_1)=-\frac{1}{2}\mu_{ng}\underbrace{\big[E(t_1)e^{-i\omega t_1}\big]}_{\mbox{resonant}}+\underbrace{E^{\ast}(t_1)e^{+i\omega t_1}\big]}_{\mbox{antiresonant}}\]
```

$$H'_{ng}(t_1) = -\frac{1}{2}\mu_{ng}\underbrace{\big[E(t_1)e^{-i\omega t_1}\big]}_{\text{resonant}}+\underbrace{E^{*}(t_1)e^{+i\omega t_1}\big]}_{\text{antiresonant}}$$

Chapter 2

科學數值運算：Scilab 入門

> Beware of bugs in the above code; I have only proved it correct, not tried it.
>
> Donald Knuth

> Computer programming is an art, because it applies accumulated knowledge to the world, because it requires skill and ingenuity, and especially because it produces objects of beauty. A programmer who subconsciously views himself as an artist will enjoy what he does and will do it better.
>
> Donald Knuth

Scilab 是法國 INRIA 與 ENPC 自 90 年代發展起的 open source 科學數值運算軟體，與頗富盛名商業軟體 Matlab 系出同源，Scilab 的開發方向與 Matlab 靠攏，除了提供一系列等效轉換指令與轉換程式外，基於 Scilab 開發的 Scicos 與 Matlab 旗下之 simulink 同爲出色之圖像化模擬器，在自動控制領域擁有衆多使用者。以 Scialb 作爲這門實驗課的平台一方面是鼓勵大家使用不必花錢，功能強大不遜於商業軟體的開放源代碼自由軟體（以教育爲目的初級課程也不需動牛刀），一方面日後同學需要轉換至 Matlab 平台也不會有疏離感，甚至待各位有機會深入學習 C/C++ 與 FORTRAN 等計算較有效率之程式語言時，仍然可以 Scilab/Matlab 作爲後端數據繪圖與分析工具。`testfile`

如果想對程式語言、資料結構或者演算法有通盤的瞭解，僅靠這本書是完全不夠的。正統的學習路線會從 C/C++ 或 Fortran 之類的語言學起，但等你熟悉資料結構，學會如何運用

運算子來處理簡單的物理問題並可以繪製圖形時，恐怕已過數月之久。無奈之餘，由 Scilab 這種入門難度低，有配套 GUI 繪圖部件的直譯式語言作爲學習程式寫做的第一步對於非主修電機資訊的物理系學生而言不失爲折衷辦法。好處是即便一面查語法，一面改範例來寫模擬都不容易出錯，學習上的挫折感較低，壞處則是是隨便寫隨便對的特性會讓人養成不好的程式寫作習慣，一個不小心就會落入 garbage in, garbage out 的窘態。既名曰科學數值計算，除去計算，仍需帶有相同比重的科學(物理)滋味才不致失焦，倘若學了一整年的數值計算後方程式求根、排序搜尋爛熟在心，卻連一個簡單的物理系統都無法模擬，甚至沒辦法將計算結果精練成數幅圖表公開法表，如此的學習方式也稱不上有效。折衷之餘，非正統的程式教學方式主要仍在於增加同學的成就感，提供大量與物理相結合的範例以待各位觸類旁通，而眞正有志於程式寫作的人則可同時閱讀正規之資工程式教材補足基礎方能更進一步。

你可以從 Scilab 的官方網頁下載最新版本的 Scilab: `http://www.scilab.org/`

2.1 Scilab 學習資源

以下依照難度（以星號標定）列出數筆可於網路上合法取得之參考資料供各位參考。

2.1.1 入門參考資料

■ □ □ □ □ □ Finn Haugen, Master Scilab, `http://home.hit.no/~finnh/scilab_scicos/`
此份 Finn Haugen 所撰寫之網頁版簡易指引包含各類基本操作，每個步驟皆有截圖，初學者仿照指引由頭至尾操作一遍約需 1 至 2 個小時，其後基本操作可了然於心。

■ ■ □ □ □ □ 中文 Scilab 資源, `http://science.openfoundry.org/ade/scilab/`
非到萬不得已，請不要依賴中文 manual。並不是所有的軟體都有人幫你寫好中文說明。此網站蒐羅不少中文化的手册條目，以及獨立章節的幾類常見方程的Scilab 求解步驟。

■ ■ □ □ □ □ Introduction to Scilab `http://www.scilab.org/support/documentation/tutorials`
這是 Scilab 官方撰寫的 Scilab 入門教材，很平衡地介軟體的各種功能，扣除安裝指南與圖形化介面指令介紹，大約有 50 頁的內容講解基本操作，包含十餘道含解答流程的習題。

■ ■ ■ □ □ □ Gilberto E. Urroz 的 lecture note,
`http://www.neng.usu.edu/cee/faculty/gurro/Scilab.html`

在國外仍有大學數值計算的課程採用 Scilab 作爲實作平台，此份由 Utah State University 的 Gilberto E. Urroz 發展的課程網頁包含許多細部講解，以及大量範例可供參考。

■ ■ ■ □ □ □ Eike Rietsch, An Introduction to Scilab - from a Matlab User's Point of View
http://wiki.scilab.org/Tutorials?action=AttachFile&do=get&target=Scilab4Matlab.pdf 這本書很完整地介紹各種基本指令的用法以及語法規則，同時列出 Scilab 與 Matlab 中用法相近的函數對照表。想要透過 Scilab 嚴謹地學習程式寫作概念可以從這本百來頁的著作開始。

2.1.2 進階參考資料

■ ■ ■ ■ □ □ Lydia E. van Dijk and Christoph L. Spiel, Scilab Bag Of Tricks, http://kiwi.emse.fr/SCILAB/sci-bot/book1.htm
這是一份相當實用的 IAQ (Infrequently Asked Questions)，遇到一些比較 tricky 的問題時可以來查找這份文件。此書有 pdf 版本: http://kiwi.emse.fr/SCILAB/sci-bot/sci-bot.pdf

■ ■ ■ □ □ □ William Hayt and John Buck, Scilab Code for Engineering Electromagnetics,
urlhttp://scilab.in/files/textbooks/ProfSenthikumar/EM.pdf
此書羅列數十筆以 Scilab 撰寫之範例求解工程電磁學中常遇到的簡易計算問題，需參照 William Hayt and John Buck 所著之 Engineering Electromagnetics中各章節例題。

■ ■ ■ ■ ■ □ Optimization in Scilab http://www.scilab.org/support/documentation/tutorials
同樣是 Scilab 官方釋出的教學文件，主要介紹如何以 Scilab 來求解極值問題，比如模擬退火法 (Simulated Anealing) 與基因演算法 (Genetic algorithm) 皆有獨立章節介紹。

■ ■ ■ ■ □ □ Scilab manual, http://www.scilab.org/download/5.2.2/manual_scilab-5.2.2_en_US.pdf
嚴格說起來，這本三千多頁的巨著幾乎就等同於 Scilab 內建說明的 pdf 版，你也可以查找網頁版 http://help.scilab.org/docs/5.3.2/en_US/。原則上這本書作爲查找函數定義的字典使用。

2.1.3 如何寫好 Scilab

1. 官方的語法說明是最好的參考書，
2. 或者閱讀坊間大量 MatLab 相關書籍，兩者寫作邏輯相通。
3. 把 Scilab 本身提供的 demostrations 執行過一遍，對照著程式碼研究。
4. 多方嘗試，練習不同的程式題目來訓練熟習度。
5. 建立自己的程式庫，往後遇到相同的狀況就能直接改編寫過的 code 來用。
6. 遇到問題可以和別人請教討論，可以到官方討論區留言求解。

2.2 基本運算

2.2.1 命令列 (console)

安裝好 Scilab，點擊捷徑後會看到白底黑字的命令列視窗。在這個模式下直接鍵入算式後按 [Enter] 可以執行運算，此時操作模式與一般掌上型科學計算機雷同。要清除 commond history 請按 [F2]。

```
--> 1+2*3/(5-2)    Enter      ans = 3.
--> ans*2          Enter      ans = 6.
```

要指定變數也很簡單

```
--> a=3            Enter      a = 3.
--> a              Enter      ans = 3.
--> a^2-1          Enter      ans = 8.
--> exp(a+3*%i)    Enter      ans = - 19.884531+2.844711 i
```

Scilab 指令爲關鍵字後加上一組圓括號，括號內爲函數的輸入值，上面的例子 sin() 即 Scilab 內建的數學函數。%i 爲系統內建的常數，也就是虛數單位，前綴字元 % 與關鍵字結合即呼叫該函數值，如圓周率 π 即 %pi，自然常數 e 即 %e。常用的數學函數請參考 Help 中 Elementary Function 章節。要查詢特定函數（或指令）的用法可以使用 help <function name>。

2.2.2 矩陣運算

Scilab 的一大特色便是矩陣運算，以下介紹陣列與矩陣的宣告與運算。

```
--> a1= [1 0 1 0 0 1]      Enter      a1 = 1.  0.  1.  0.  0.  1.

--> a1(3)                  Enter      ans = 1.
--> a2 = [1,2;3,4]         Enter      a2 = 1.  2.
                                           3.  4.

--> a3 = 1:4               Enter      a3 = 1.  2.  3.  4.
--> a4 = 0:0.2:1           Enter      a4 = 0.  0.2  0.4  0.6  0.8  1.0
--> a5 = [1:3;2:4]         Enter      a5 = 1.  2.  3.
                                           2.  3.  4.

--> row1 = a5(2,:)         Enter      row1 = 2.  3.  4.
--> column3 = a5(:,3)      Enter      column =  3.
                                              4.

-->a2*a5     Enter      ans =  5.   8.   11.
                               11.   18.   25.


-->a2^2      Enter      ans = 7.    10.
                              15.    22.

-->a.^2      Enter      ans = 1.     4.
                               9.    36.
```

+ - * / 加上前綴字元 . 代表矩陣元素的個別運算。: (colon) 是 Scilab 中相當常用的一個運算子，可以用來宣告等比數列，請特別注意它的用法。宣告矩陣元素時可以， (comma) 區隔，或者直接加上一格空白。要換到下一列的時候則加上 ; (semicolon) 標記。

2.2.3 運算次序

```
--> 2^2*3^2      Enter      ans = 36.
--> a=[1 2;3 4]; b=[1 2;1 2]; a^2*b^2      Enter      ans = 51.   102.
                                                        111.   222.
--> a.^2.*b.^2      Enter      ans = 5.   20.
                                     25.   100.

--> (a.^2).*(b.^2)      Enter      ans = 1.   16.
                                       9.   24.
```

倒數第二條敍述可能會被誤解爲與最末條敍述等義，事實上 Scilab 會把第二個句點判別爲指數 2 之小數點：在不確定運算優先序的情況下，還是不厭其煩地使用括號來輔助比較保險。一個敍述若是以；（semicolon）作結尾，計算結果不會輸出在螢幕上：若以（comma）作結尾則相反。

2.2.4 字串 string

Scilab 可以處理字串類型的資料，宣告時以單引號或是雙引號標示，要將數值類型的變數轉換成字串可以使用 `string` 指令，反之想將字串類型的傳換成數值可以使用 `eval` 指令。字串當然也可以宣告成矩陣的形式，但是字串與數值變數不允許混用。

```
--> a=[ 'what is your name?' , 'how old are you ?' ]   Enter
a = !what is your name?  how old are you ?  !

-->name = input( a(1),'s' )   Enter
what is your name?   Type   clin   Enter   name = clin

--> age = input( a(2) )   Enter
how old are you ?   Type   21   Enter   age=21.

printf('You are %s. Next year you will be %d. ', name, age+1)   Enter
You are clin. Next year you will be 22.
```

printf 顧名思義即 Scilab 模擬 C language 列印輸出的函數，用法與 C language 的相同，範例中的 <%+specifer> 可以看成不同資料型態的變數，%s %d 依序對應到後頭以 comma 分開的變數 name 與 age-1；不同的 specifer 代表不同的 data type，比方說 s 代表字串，d 代表十進位制的數值變數，詳細的用法請自行查找 Scilab 的說明。

2.2.5 Script

利用指令 scipad 或點擊工具列上的快捷鍵可以進入 script 編寫環境，與直譯式的程式語言相近，可以撰寫內容較長的程式碼來完成比較複雜的計算，以後你們繳交報告的時候也要附上相關的 .sce 檔供助教檢閱。由 scipad 工具列 Execute→ Load in scilab 或按下組合鍵 Ctrl + i 可以執行 script 程式碼，或者在命令列中輸入該檔案名稱並按 Enter 執行。

編輯較大型的程式請遵守以下原則:

- 不希望將計算結果即時顯示在命令列上的中間計算指令請記得在該句末加上分號;
- 加上適當的註解 (使用連續兩個斜線 //}(double-slash)
- 變數名稱要設的易懂且有意義，常用的命名方式有兩種:一種使用縮寫加手自大寫，如 DisplsmtOfArCrft， 一種使用縮寫加底線，如 Displsmt_of_ar_crft。
- 有意義的常數值要設成變數以方便可能的修正。
- 程式落入無窮迴圈想終止時在命令列視窗按 Ctrl + c

以下提供一個簡單的自由落體程式碼作爲範例，計算的結果以 plot 指令將物體速度與位移函數繪製成 Figure. 2.2.5。

```
//---------script demo--------

// constant table
grvy = 9.8;              // m/s, acceleration of gravity
m = 1;            // kg, mass of the ball
time = 2E2;       // total simulation time, sec
t_step = 1E-3;        // ms
x0=0;              // initial displacement, meter
v0=0;             // initial velocity, m/s

// 將計算中會用到的常數值與初始條件放至文件開頭
```

```
// 在註解中紀錄各常數之單位
// 大數值使用科學記號 aEb=a*10^b 可讓版面更簡潔

//----------parameter----------
total_step = roundtime/t_step;      // total simulation step
t_scale = (1:total_step)*t_step ;    // create corresponding t-axis

// 此模擬預先設定總模擬時間與單步間距，總步數與對應的時間軸由其導出

// ---------free falling---------
displacement = x0+0.5*grvy*t_scale.^2;
vlcty = v0+grvy*t_scale;

// 完成參數設定便進行計算
// 此範例有解析解，直接帶入時間軸陣列即可推算出位移與速度

// ----------plot----------------

// subplot 可設定子圖顯示，subplot(klm) 代表有 k 列 x l 欄 = m 個子圖，
// m 爲當前子圖編號，由第一列左至右依序填入圖塊，第一列填滿後再填入下一列
// ex:  | subplot(231) | subplot(232) | subplot(233) |
//      | subplot(234) | subplot(235) | subplot(236) |

subplot(121)
plot( t_scale,displacement,'x');
xtitle('dispalcement of free falling','10^(-1) ms','meter');

// plot(a(1:n),b(1:n)) 代表以 (x1,y1)=(a(1,1),b(1,1)),(x2,y2)=(a(1,2),b(1,2)) 等等
//  n 個數據點進行二維繪圖，plot(a,b,'s') 中的 's' 爲繪圖控制字串，'x' 代表以 x 標記數據點

subplot(122)
plot(t_scale,vlcty,'o');
xtitle('velocity of free falling','10^(-1) ms','meter');
```

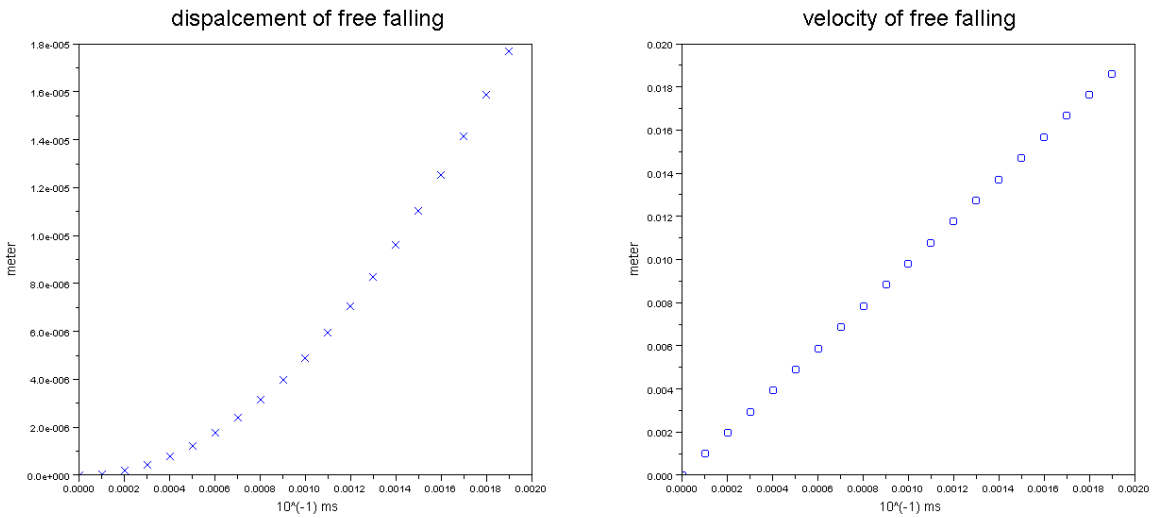


Figure 2.1: script demo 輸出圖形，左圖爲位移隨時間之變化，右圖爲速度隨時間之變化

在編輯大型程式的時候，有幾點要特別注意:

寫程式一方面是要簡化繁複的運算，建立穩定可靠、具備特殊用途的個人化計算器，一方面也要有可維護、可更新的彈性，讓我們可以透過少許修改來擴充用途。適當的命名變數與註解讓程式碼不再是艱澀難解，也不會有看不懂自己寫的程式的窘境（事實上，這是極有可能發生的慘劇）。對於剛開始學習撰寫程式的初學者而言，不大會碰到動輒數百行的龐大程式，但是良好的程式寫作習慣必須趁早培養：就 team work 的觀點，程式寫的再好，別人看不懂也是沒有用。

2.2.6 資料存取

Scilab 可以利用 `save` 與 `load` 存取 binary data，也可以使用 `file` 與 `write read` 將檔案存取成 txt 檔。前者用法較爲簡單，不做贅述。資料的存取主要在 Scilab 的 work directory 下進行，在命令列鍵入 `cd` 可以得知當前的工作目錄，你可以點選工具列 File→ 改變目前目錄 來變更檔案存放的目錄，不過請記得儲存路徑儘量不要包含中文檔名 (比如說中桌面，我的資料夾等等)，否則 Scialb 執行時可能發生錯誤。以下先示範如何使用 `save` 與 `load` 來存取變數資料:

```
--> x=1:0.5:3      Enter        x = 1.   1.5   2   2.5   3.
--> y=0:-0.5:-2    Enter        y = 0.  -0.5  -1  -1.5  -2.
--> save('dataXY.dat',x,y)     Enter        --> clear     Enter

-->listvarinfile('data.dat')    Enter
Name                        Type            Size            Bytes
-------------------------------------------------------------------
x                           constant        1 by 5          56
y                           constant        1 by 5          56

--> load('data.dat','y')    Enter
--> x     Enter     !--error 4   未定義變數
--> y     Enter     y = 0.  -0.5  -1  -1.5  -2.

--> clear     Enter     --> load('data.dat')     Enter
--> x     Enter     x = 1.   1.5   2   2.5   3.
--> y     Enter     y = 0.  -0.5  -1  -1.5  -2.
```

使用儲存指令 save(<FileName>,variable1[,variable2,...]) 時第一個參數要放入檔案名稱 (可包含副檔名)，檔案名稱必須是字串變數 (加上單引號) 如 'data.dat'，後面的參數則填入要儲存的變數名稱，變數名不需要加引號。假如不記得當初儲存了哪些變數在檔案中，我們可以使用 listvarinfile 來詢問儲存變數的名稱、資料型態與總長度。當我們需要將資料載入記憶體時，使用讀取指令 load('<FileName>'[,variable1, variable2, ...]) 時第一個參數要放入檔案名稱，後面的參數則填入要載入的變數名稱 (若不填參數則預設爲該檔案中的變數全部載入)。要特別注意的地方在於: load 指令的 variable1 等變數名必須使用字串的型態呼叫， 與 save 指令不太一樣。文後描述指令規則時若以 括號夾註如 <FileName> 代表必須以字串型態輸入，不以括號夾住如 variable1 代表直接以變數名稱輸入即可，方括號內以逗點隔開變數的標示則代表可選擇不輸入的參數。

使用 save 與 load 指令只能儲存 scilab 能識別的資料型態，若資料數量不大，可以使用 file 相關指令將變數儲存成 txt 檔以使用者方便調閱。相關的語法如下:

```
fileID = file ( <action>, <FileName>, <status> )
```

- <action>: 'open' 開啓檔案、'close' 關閉檔案、'rewind' 將指標切回檔案最前端

- <status>:目錄中尚未存在 <FileName> 這個檔案時使用'new'; 目錄中已存在 <FileName> 這個檔案，開啓時使用'old' ; 不確定要開啓或關閉檔案的狀態時使用 'unknown'。

使用 flie 創造或開啓檔案之後，可以使用 write (<FileName>, [var1, var2, ...], <format>) 指令將變數資料寫進檔案中。其中 <format> 定義了變數要以何種型態存進檔案中，基本規則與 FORTRAN 語言相同，整數與浮點數十進位的表達方式請看 Figure. 2.2。我們可以先利用指令列模式來熟悉格式操作，若以 %io(2) 作爲 <FileName> 可以將結果直接顯示在螢幕上:

```
--> y=%pi        Enter        y  =  3.1415927
// Scilab 預設的輸出格式設爲十進位，有效數字 10 位
--> format('v',20), y    Enter      y  =  3.14159265358979312
// 利用 format 指令將變數輸出格式設爲十進位，有效數字 20 位

--> write(%io(2),y,'(F6.4)')          Enter      3.1416
--> write(%io(2),y,'(F8.4)')          Enter        3.1416
--> write(%io(2),y,'(F8.6)')          Enter      3.141593
--> write(%io(2),y,'(F8.6)')          Enter      3.141593
--> write(%io(2),[y,y],'(F8.4,F8.4))')     Enter        3.1416  3.1416
--> write(%io(2),[y,y],'(2(F8.4))')        Enter        3.1416  3.1416
// 連續印出兩個變數，第二個參數必須是 1 x 2 的陣列
// F8.4 改爲 2(F8.4) 代表連續印出兩個這種格式的變數
// 當然 2(F8.4)與 F8.4,F8.4 的效果相同，只是語法更爲簡潔

--> format('v',10), ystring=string(y);       Enter
--> write(%io(2),['pi=',ystring],'(A,A)')    Enter      pi=
                                                        3.1415927
--> write(%io(2),['pi='+ ystring],'(A)')     Enter      pi=3.1415927
// write 指令不能混合輸出不同資料型態的變數，假如我想要顯示字串與數值變數，
// 必須先用 string 將數值變數改爲字串，才能配合文字輸出。write 指令在輸出時
// 字串會自動換行，要避免換行可以使用字串的 + 運算將欲呈現的字串群合成單一字串

--> write(%io(2),'pi=')                     Enter      pi=
--> write(%io(2),%pi,'(''pi='',F8.4)')      Enter      pi=  3.1416
```

- w : 總顯示位數（若取值大於該變數包含位數，補空白或零）
- m : 最小顯示位數（當 w 大於變數包含位數時，補零位數與變數位數的總和）
- d : 小數點右邊要顯示的位數（小數點本身算一位）

欲儲存變數資料型態	語法	變數內容	格式指令	結果
整數 (Integer)	Iw	1234	I3	
			I4	1234
			I5	1234
			I6	1234
整數 (Integer)	I$w.m$	1234	I5.4	1234
			I5.5	01234
			I6.6	001234
			I7.6	001234
浮點數十進位 (decimal)	F$w.d$	123.456	F4.0	123.
			F5.0	123.
			F5.1	123.
			F6.0	123.
			F6.1	123.5
			F6.2	123.46
			F7.3	123.456
			F8.4	123.4560

Figure 2.2: 整數與浮點數十進位的規則

```
// 事實上 write 的格式敍述仍然允許你直接輸入字串混和數值變數
// 這個時候需要將代表字串的提示字元單引號'改成兩個單引號''，並以逗號與數值格式隔開
```

除了整數格式 F 、十進位浮點數 F 以外，我們也很常使用指數格式E ES 來顯示資料，其中 E 與 ES 的差別在於 E 整數部位僅顯示小數，相關的語法規則請參考 Figure. 2.3 與以下範例:

```
--> foramt('v',10), k=2^20      Enter      k = 1048576
--> format('e',10), k           Enter      k = 1.049D+06
--> 1/k  Enter      k = 9.537D-07
// 利用 format 指令將變數輸出格式設爲指數形式，k=1.049 x 10^6
```

- w : 總顯示位數（若取值大於該變數包含位數，補空白或零）
- m : 小數顯示位數（當 w 大於變數包含位數時，補零位數與變數位數的總和）
- e : 指數部份右邊要顯示的位數（小數點本身算一位）

欲儲存變數資料型態	語法	變數內容	格式指令	結果
指數 (Exponential)	Ew	1.049D+6	E8.0	
			E8.1	0.1E+07
			E8.2	0.10E+07
			E8.3	.105E+07
指數 (Exponential)	Ew.mEe	1.049D+6	E8.1E1	0.1E+7
			E8.1E2	0.1E+07
			E8.1E3	0.1E+007
指數 (Scientific)	ESw	1.049D+6	E8.0	1.E+06
			ES8.1	1.0E+07
			ES8.2	1.05E+07
			ES8.3	
指數 (Scientific)	ESw.mEe	1.049D+6	ES8.1E1	1.0E+6
			ES8.1E2	1.0E+06
			ES8.1E3	1.0E+006

Figure 2.3: 浮點數指數表達的規則

```
--> write(%io(2),k,'(E8.2E2)')      Enter       0.10E+07
--> write(%io(2),k,'(ES8.1E3)')     Enter       1.05E+06
--> write(%io(2),k,'(ES8.1E3)')     Enter       1.05E+06

-->write(%io(2),[k,k],'(f8.0,ES8.1E3)')  Enter     1048576.1.0E+006
-->write(%io(2),[k,k],'(f8.0,3x,ES8.1E3)')  Enter     1048576.    1.0E+006
// 在表達式中加入 x 可以加入空格，3x 代表加 3 個空格
```

代表第一個變數使用指數表示，總長度十位，小數點後保留五位；第二個變數代表十進位表示，小數點後留四位。若 file-name 設爲 %io(2) 則代表輸出結果顯示在螢幕上。

```
--> format('v',20)  //            20
--> a=1234567890.123456 ;
--> write ( %io(2), [a a a ], '(f15.4, f15.3, f15.2)' ) [Enter]
1234567890.1235 1234567890.123  1234567890.12

--> write ( %io(2), [a a a ], '(f15.4,3x, e15.8,3x, d15.8)' ) [Enter]
1234567890.1235    0.12345679E+10     0.12345679D+10

--> write  ( %io(2), a, '(f15.5)') [Enter]
***************  //           format   f16.5

-->str1 = 'alpha' ; str2 = 'beta' ;
-->write ( %io(2), [str1 str2; str1 str2] ,'(''str='',2x,a5)' ) [Enter]
str=  alpha        //   format
str=  alpha        //    write
str=   beta         //  x    2x
str=   beta


read

data1 = read ( file-name, row , column, <format> )
```

當你知道讀取的資料行數的時候，row 設爲行數，不曉得行數時，row 設爲 -1 可以確保讀取所有資料。

2.2.7 I/O 程式碼範例

寫入 txt 檔

```
time = 1:0.2:2; //
data_sin = sin(a);
data_cos = cos(a);

title_str=['time','data_sin','data_cos'];
title_name = input('title name?','s'); //
title name? IOdemofile [Enter]

file_name = title_name+'.txt';
f1=file('open',file_name,'new'); //

write(f1,title_str,'(a)'); //

for i=1:size(time,2),  //
write(f1,[time(i),data_sin(i),data_cos(i)],'(d,5x,e,e)');
end;

file('close',f1); //
```

讀取 txt 檔資料

一旦知道資料排列格式，很容易就能利用 read 指令取出之前存入的變數。

```
//----------script file-------------
f3 = file ('open','demofinal.txt','unknown');
file ('rewind',f3);
title1 = read(f3,3,1,'(a)');
```

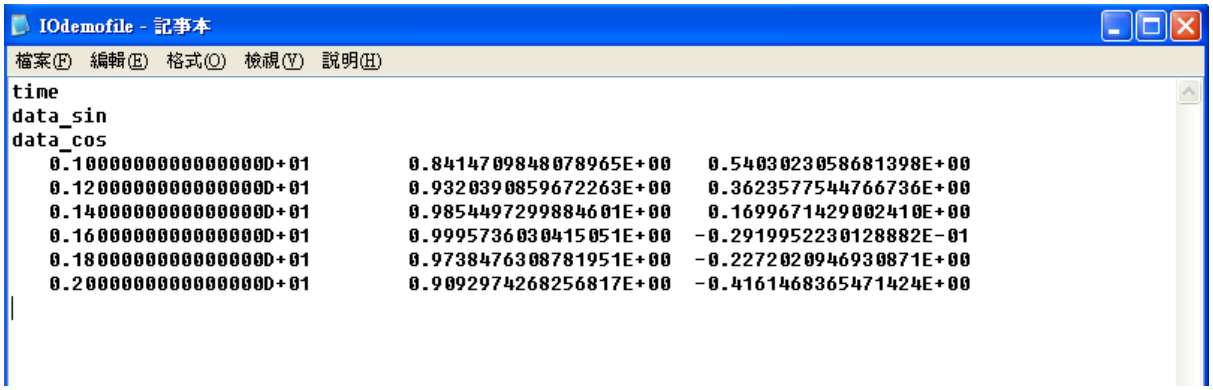


Figure 2.4: IOdemofile.txt 的內容

```
data = read(f3,6,3,'(d,5x,e,e)')
write (%io(2),title1(1),'(''column1 = '',a)');
write (%io(2),title1(2),'(''column2 = '',a)');
write (%io(2),title1(3),'(''column3 = '',a)');

for i= 1:6,
write (%io(2),[data(i,1),data(i,2),data(i,3)],'(d,e,e)');
end;

//----------screen---------------

column1=time
column2=data_sin
column3=data_cos
   0.1000000000000000D+01   0.8414709848078965E+00   0.5403023058681398E
+00
   0.1200000000000000D+01   0.9320390859672263E+00   0.3623577544766736E
+00
   0.1400000000000000D+01   0.9854497299884601E+00   0.1699671429002410E
+00
  0.1600000000000000D+01  0.9995736030415051E+00 -0.2919952230128882E-01
   0.1800000000000000D+01   0.9738476308781951E+00  -0.2272020946930871E
+00
   0.2000000000000000D+01   0.9092974268256817E+00  -0.4161468365471424E
```

+00

2.2.8 實習報告

使用任何程式語言（除了助教講解所使用 Scilab，你也可以利用C，C++ 或 java，Python 等各類你熟悉的程式語言）做字串的輸入（從鍵盤或檔案）與輸出（到螢幕或檔案）與檔案合併。題目如下

資料輸入

以鍵盤鍵入 10 ~ 100 個英文字，輸出成 txt 檔，使用者可以決定檔案名稱。 如果你使用 Scilab 作爲工具，可以利用`input` 或是 GUI 函數 `x_dialog`，`x_mdialog` 來完成這個問題。

資料讀入

由助教給予包含特定資訊的 txt 檔（可能是利用某台儀器量取到的電壓電流值，某一行第一個數字代表電壓值，加一個空格後第二個數字代表電流值），利用程式將 txt 檔中的資料擷取成矩陣或陣列的資料格式，並繪製成圖檔輸出。

檔案合併

根據助教提供的數個以自然數編號的 txt 文字檔，依照順序將這幾個檔案合併成一個較大的 txt 檔輸出。

Chapter 3

曲線套適

本章介紹如何使用 Scilab 進行曲線套適 (curve fitting)。一般而言， Excel 之類試算表軟體僅能處理多項式形態的函數，遇到非線性的情況（正弦函數，高斯分佈等等）就不容易處理了，這個時候使用 Scilab 自行撰寫函數，並使用內建的演算法來處理這些計算便能很快取得我們需要的結果。除了Scilab/Matlab/R language 等在數據分析統計上可以靈活撰寫的程式語言外，也有專門針對這類功能開發的軟體，比方說許多實驗室採用的商用軟體 ORIGIN，還有功能和介面與之相當類似的自由軟體 LabPlot。這些專門用於數據處理的軟體通常擁有很直觀的圖形化介面，允許使用者直接於表格中輸入數據，透過簡單的步驟便可即時輸出各類符合期刊與著作規範的圖表，也能處理各類曲線套適問題。

以下章節將輔以實際的例子講解，除了實際演練如何計算實驗測量結果的誤差量，繪製出帶有 error bar 的數據套適圖以外，也引入了誤差傳播分析。最後則介紹如何使用 Scilab 內建的非線性套適指令來加速特殊情形的套適工作。

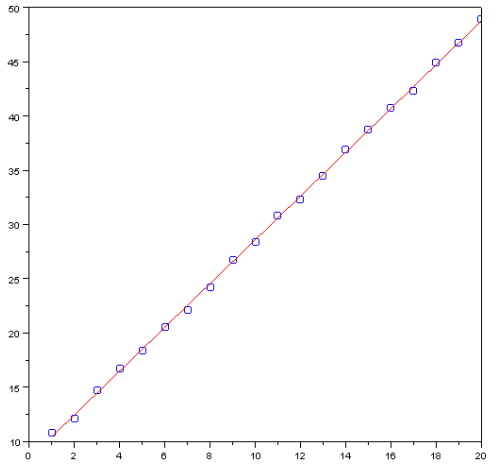
3.1 線性回歸分析

針對線性數據的回歸分析 Scilab 中有個簡單好用的指令 `reglin`:

`[a,b,sig]=reglin(x,y)` 其中 `sig` 定義成 $\sqrt{\frac{\sum(y(x_i)-y_i)^2}{N}}$

`x` 與 `y` 爲原始數據，`a` 與 `b` 爲滿足 $y = ax + b$ 結果之待定參數。

假如我們使用這個函式來計算同一個物理量 y 的多次測量，則 $a \to 0,\ y(x_i) \backsimeq b = \bar{y}$ ，而 sig 就會是測量值 y 的標準差。



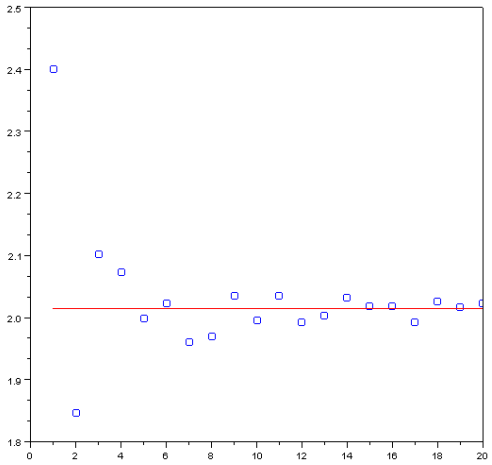


Figure 3.1: 圖左: `plot(x,y,'bo');` `:plot(x,a*x+b,'r');`

3.1.1 線性回歸

假設我們要對以下數據做線性回歸:

```
x=1:20;
y=2*x+8+5*rand(1,20);
```

我們使用 $dy/dx = 2$，截距爲 8 再加上 `rand()` 生成的雜訊來模擬成原始資料。接著可以使用 `reglin` 來進行線性回歸分析:

```
[a,b,sig]=reglin(x,y);
printf('a=%f,b=%f,stdev=%f',a,b,sig);
```

Enter

```
a = 2.014705,b=8.435809,stdev=0.241168
```

顯然 `reglin` 成功地抓出這組的回歸參數。我們可以作圖觀察回歸分析的結果，Figure. 3.1 左邊中紅線爲回歸分析之結果，藍色圓圈則是原始數據; 右邊則爲相對殘差（ $(y - \bar{y})/x$ ）之作圖。我們可以觀察到當 x 越小時，數據相對偏離回歸曲線較遠，這個結果也反映生成測試數據時所引入的誤差不隨 x 增加而增大的特性。

3.1.2 加入 error bar

假如對每個 x 點我們都取了多筆數據（也許是幾十筆，比方說從製造商處可得知某把游標尺的測量的標準誤差爲 ± 0.05 mm，我們可將其視爲已知標準差）我們可以針對每一個單一的數據點找出其 error bar，在已知標準差(這是相當重要的假設!)且保證測量不發生任何系統誤差的情況下， error bar 的量可取爲母體標準差 σ（population standard deviation）再除上 $\sqrt{N}$, 意即 $err = \sigma/\sqrt{N}$，其中 N 爲樣本數（該 x 點測量的次數），$\sigma = \sqrt{\sum(x_i - \bar{x})^2/N}$ 。此 error bar 之意義在於我們確信測量值的分佈呈常態分佈（即高斯分佈）時，樣本平均值 $\bar{x}$ 落在 $\mu - err$ 與 $\mu + err$ 之間的機率爲 68.3%， 換句話說我們有 68.3% 的信心水準推斷母體（population，即觀察對象總體眞實的分佈，我們所有的測量都可視爲對母體的取樣）分佈的平均值會落在此信賴區間（confidence intervals，即由母體平均值 μ 往兩邊擴展的區間 $(\mu - err, \mu + err)$）。從這個關係式我們也可以發現當測量次數越多，信賴區間越小，換言之我們越能確定期望值落在何處。多次測量能夠有效地降低期望值的標準差。

以上的討論成立的先決條件在於我們得到測量值（抽樣）必須是統計獨立的隨機變數（以游標尺測量爲例，你必須保證測量時不至於發生扭曲夾爪，導致後來的測量失準，或者這把尺一拉開就無法復原等等影響測試獨立性的問題），且抽樣數必須大於 30 才行(實用上的一個分界點) 。中央極限定理保證不論原先測量誤差分佈爲何，絕大多數的情況下大量取樣後必然呈現常態分佈，也因此可透過實驗值推估的方式訂出信賴區間。

實際應用的時候，我們其實無法得知測量對象的標準差爲何，通常是透過較少量的數據取樣來推估整體標準差，這個時候 error bar 就要修正成以樣本標準差（sample standard deviation）爲基礎的形式:

$$err = \frac{S_x}{\sqrt{N}} = \sqrt{\frac{\sum(x_i - \bar{x})^2}{N-1}}\frac{1}{\sqrt{N}} = \sqrt{\frac{\sum(x_i - \bar{x})^2}{N(N-1)}}$$

其中分子的 N 改爲 $N-1$ 的變化來自於取期望值運算的結果，當 N 極大的時候樣本標準差會趨近眞實標準差。當然在 $N > 30$ 的情況下兩者其實相去不遠，N 值不大的時候基本上有更好的方式來推估信賴區間。無論如何，以後者作爲 N 值不大時的近似計算效果仍好些。假如有一筆原始資料:

```
for i=1:5, x(i,:)=1:6; end
y=4*x+8+6*rand(5,6);
```

```
--> disp(x)   Enter
```

```
1.    2.    3.    4.    5.    6.
1.    2.    3.    4.    5.    6.
1.    2.    3.    4.    5.    6.
1.    2.    3.    4.    5.    6.
1.    2.    3.    4.    5.    6.

-->disp(y)  Enter
15.701337    20.676435    20.582262    26.541694    28.865982    32.102974
16.362677    17.116509    22.627503    28.335808    30.547077    34.265607
13.607524    20.854629    24.707176    27.604018    32.226605    36.985531
17.418075    18.333678    20.944104    28.545069    32.498039    33.270383
14.494213    16.295701    22.950088    24.876884    32.877474    37.979044
```

y 的 column 爲同一個 x 值下不同的測量結果，我們可以利用 `variance` 搭配 `sqrt` 指令來算出樣本標準差，最後以 `errbar` 這個指令來加上 error bar。。注意到 `variance` 必須接受 n×1 的行向量作爲輸入，因此原本的資料陣列必須先經過轉置才行。

```
for i=1:6,
err(1,i)=sqrt(variance(y(:,i)','c')/(5-1));
// 也可以直接用內建樣本標準差指令 err(1,i)=stdev(y(:,i)')
ybar(1,i)=sum(y(:,i))/5; end
[a,b,sig]=reglin(x,ybar);


subplot(121)
plot(x(1,:),ybar,'bo'); errbar(x(1,:),ybar,err,err);
plot(x(1,:),a*x+b,'r');
subplot(122)
plot(x(1,:),(ybar-b)./x(1,:),'bo');
errbar(x(1,:),(ybar-b)./x(1,:),err./x(1,:),err./x(1,:));
plot(x(1,:),a*ones(1,size(x,2)),'r'); isoview(0.5,6.5,2,6)
```

同樣地我們可以 rescale 座標軸來觀察相對 error bar ，繪製結果如 Figure. 3.2。

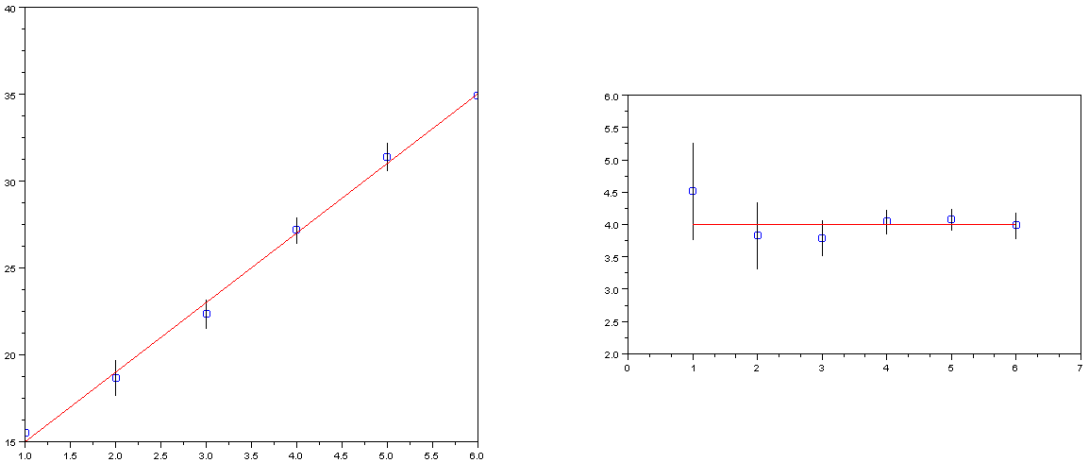


Figure 3.2: 加上 error bar 的回歸分析圖表

3.1.3 Student's t-test

事實上，當 N 小於 30 的時候 (一般的小型實驗)，或者我們無法事先得知母體標準差之值，在推估母體參數時必須將標準差視爲隨機變數去分析，將樣本標準差代之以 $N-1$ 個自由度的 χ^2 分佈，最後可以發現平均值的信賴區間由 student's t 分佈來決定。

$$P(\mu - t_\xi \frac{S_x}{\sqrt{N}} < \bar{x} < \mu - t_\xi \frac{S_x}{\sqrt{N}}) = \xi, \quad \frac{S_x}{\sqrt{N}} = \sqrt{\frac{\sum (x_i - \bar{x})^2}{N(N-1)}}$$

此式的意味著：平均值落於區間 $(\mu - t_\xi S_x/\sqrt{N}, \mu + t_\xi S_x/\sqrt{N})$ 的機率爲 ξ，同時信賴區間的大小又由 ξ 和取樣次數 N 所決定。t_ξ 可以透過查表的方式來求出。與前述基於樣本標準差的方法不同之處在於此法信賴區間之取法不僅僅受信心水準影響（前一個例子中若要改變信心水準可以透過 error funtion 查表來轉換對應的信賴區間），取樣次數的多寡（自由度）也會大幅影響取值。整體而言 t 分佈較高斯分佈來的扁平些，若在取樣數不高的情況下使用高斯分佈來推估參數很可能會誇大實驗的準確度。以下將採用 t 分佈來重新處理上個例子的數據。

採用 0.683 的信心水準（少樣本的推估在使用未知變異數分析時通常會選取超過 0.9 的信心水準以增加準確度，此例中爲了與前例比較故選取 0.683），樣本數 $N = 5$，透過查表可得 $t_{0.683} = 1.11$。使用 t 分佈推估出來的 error bar 如 Figure. 3.4 所示。

```
tx=1.11;  // 設定 error bar 縮放 factor
plot(x(1,:),(ybar-b)./x(1,:),'bo');
```

N-1	1	2	3	4	5	6	7	8	10	20	40	∞
$t_{0.50}$	1.00	0.82	0.77	0.74	0.73	0.72	0.71	0.71	0.70	0.69	0.68	0.67
$t_{0.683}$	1.84	1.32	1.20	1.14	1.11	1.09	1.08	1.07	1.05	1.03	1.01	1.00
$t_{0.80}$	3.08	1.89	1.64	1.53	1.48	1.44	1.42	1.40	1.38	1.33	1.30	1.28
$t_{0.90}$	6.31	2.92	2.35	2.13	2.02	1.94	1.90	1.86	1.83	1.73	1.68	1.65
$t_{0.95}$	12.7	4.30	3.18	2.78	2.57	2.45	2.37	2.31	2.23	2.09	2.02	1.96
$t_{0.98}$	31.8	6.97	4.54	3.75	3.37	3.14	3.00	2.90	2.76	2.53	2.42	2.33
$t_{0.99}$	63.7	9.93	5.84	4.60	4.03	3.71	3.50	3.36	3.17	2.85	2.70	2.58
$t_{0.995}$	127	14.1	7.45	5.60	4.77	4.32	4.03	3.83	3.58	3.15	2.98	2.81
$t_{0.999}$	637	31.6	12.9	8.61	6.87	5.96	5.41	5.04	4.59	3.85	3.55	3.29

Figure 3.3: 不同信賴水準的 t_ξ 隨 N 值的變化

```
errbar(x(1,:),(ybar-b)./x(1,:),tx*err./x(1,:),tx*err./x(1,:));
plot(x(1,:),a*ones(1,size(x,2)),'r');  isoview(0.5,6.5,2,6)
```

3.1.5 which is better?

在前面兩個多次量測的例子中，我們採用先求出自變數 x 相同時應變數 y 的平均值作爲迴歸分析對象。這樣的作法是否合理？我們可以將之前的結果與直接對所有數據進行迴歸分析做比較:

```
[aBar,bBar,sigBar]=reglin(x(1,:),ybar);

xAll=[x(1,:),x(1,:),x(1,:),x(1,:),x(1,:)];
yAll=[y(1,:),y(1,:),y(1,:),y(1,:),y(1,:)];
// 將五列資料併爲一列輸入 reglin 函式運算
[aAll,bAll,sigAll]=reglin(xAll,yAll);

printf('ybar  fit: a=%f, b=%f, variance=%f \n',aBar,bBar,sigBar);
printf('all y fit: a=%f, b=%f, variance=%f \n',aAll,bAll,sigAll);

----result-------
ybar  fit: a=4.002318, b=10.998358, variance=0.406937
all y fit: a=3.215322, b=12.824822, variance=1.035531
```

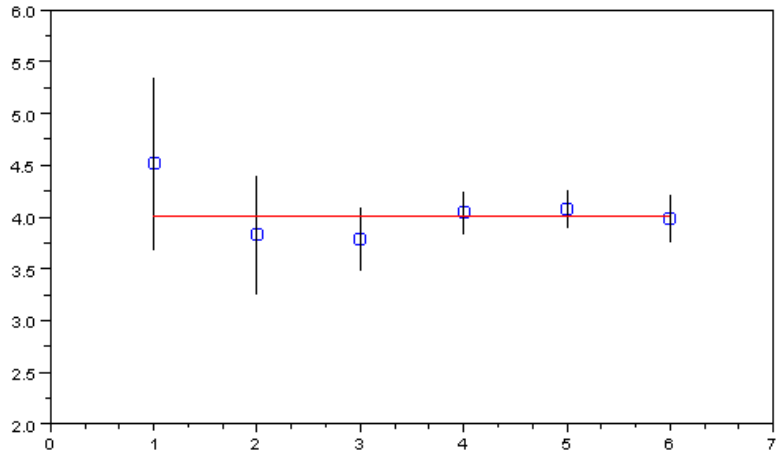


Figure 3.4: 少樣本推估所得的 error bar 較前例大了 11%

我們可以發現先前採用的方法可以求得比較接近眞實狀況的參數，且標準差較小。前者先取平均值的作法相當於使用已知的 error bar 來進行曲線套適，估計到參數 a, b 之變異數會比較小。詳細的 linear regression 理論不在本文的介紹範圍，有興趣者可以參考 mathematical statistics 領域的書籍或是 Supplementary Notes on General Physics, Jyhpyng Wang 附錄中以最大概似法（maximum likelihood estimation）進行套適理論分析的段落。Scilab 的 `reglin` 函式僅使用最小平方法進行分析，欲獲得更好的統計結果勢必得自行撰寫所需要的 script code 才行。

更進一步來思考，如果每個數據點的取樣數目不同導致各點的誤差區差異極大的時候，進行回歸分析的時候究竟要如何找出更接近眞實情況的套適參數? 這個時候 `reglin` 指令已不敷使用，必須選擇可加入權重因子 (weight) 的 `datafit` 指令。

3.2 誤差傳播

當我們想知道的物理量只能透過間接計算其他物理量來獲得的時候，我們可以對相關函數進行泰勒展開來估算目標物理量的變異數。

假設目標物理量爲 N 個直接觀測量的函數 $y = y(\boldsymbol{x}) = y(x_1, x_2, \cdots, x_N)$，我們可以在

直接觀測量 $\bar{\boldsymbol{x}} = (\bar{x_1}, \bar{x_2}, \cdots, \bar{x_N})$ 的期望值附近做泰勒展開:

$$\begin{aligned} y(\boldsymbol{x}) &= y(\bar{\boldsymbol{x}}) + \sum_i (x_i - \bar{x_i}) \frac{\partial y}{\partial x_i}\bigg|_{\boldsymbol{x}=\bar{\boldsymbol{x}}} + \frac{1}{2!}\sum_{i,j}(x_i - \bar{x_i})(x_j - \bar{x_j}) \frac{\partial y}{\partial x_i}\frac{\partial y}{\partial x_j}\bigg|_{\boldsymbol{x}=\bar{\boldsymbol{x}}} + \mathcal{O}(\boldsymbol{x}^3) \\ &\simeq y(\bar{\boldsymbol{x}}) + \sum_i (x_i - \bar{x_i}) \frac{\partial y}{\partial x_i}\bigg|_{\boldsymbol{x}=\bar{\boldsymbol{x}}} \end{aligned}$$

因此我們可求得 y 之變異數:

$$\begin{aligned} \sigma^2(y) &= (y - y(\bar{\boldsymbol{x}}))^2 = \sum_{i,j}(x_i - \bar{x_i})(x_j - \bar{x_j}) \frac{\partial y}{\partial x_i}\frac{\partial y}{\partial x_j}\bigg|_{\boldsymbol{x}=\bar{\boldsymbol{x}}} \\ &= \mathrm{Cov}(x_i, x_j) \sum_{i,j} \frac{\partial y}{\partial x_i}\frac{\partial y}{\partial x_j}\bigg|_{\boldsymbol{x}=\bar{\boldsymbol{x}}} \\ &= \sum_i \left(\frac{\partial y}{\partial x_i}\right)^2\bigg|_{\boldsymbol{x}=\bar{\boldsymbol{x}}} \sigma^2(x_i) + 2\sum_{i<j} \frac{\partial y}{\partial x_i}\frac{\partial y}{\partial x_j}\bigg|_{\boldsymbol{x}=\bar{\boldsymbol{x}}} \mathrm{Cov}(x_i, x_j) \end{aligned}$$

假設各個直接觀測量彼此無關，換言之 $\mathrm{Cov}(x_i, x_j) = 0$，我們可以省略末式的第二項。

3.2.2 基本運算的誤差傳遞

若 $y = \sum_i x_i$，則 $\sigma^2(y) = \sum_i \sigma^2(x_i)$。 若 $y = \Pi_i x_i = x_1 \times x_2 \times \cdots \times x_N$，則

$$\sigma^2(y) = \sum_i \left(\frac{\bar{x_1}\bar{x_2}\cdots\bar{x_N}}{\bar{x_i}}\right)^2 \sigma^2(x_i) = \bar{y}^2 \sum_i \frac{\sigma^2(x_i)}{\bar{x_i}^2}, \quad \left(\frac{\sigma(y)}{\bar{y}}\right)^2 = \sum_i \left(\frac{\sigma(x_i)}{\bar{x_i}}\right)^2$$

當目標物理量爲直接觀測值之積時，目標量的相對標準差的平方會是觀測值相對標準差的平方和。 要特別注意的地方在於以上所作近似要在每一個直接觀測量都滿足下式才成立:

$$\left|\frac{\partial y}{\partial x_i}\right|_{\boldsymbol{x}=\bar{\boldsymbol{x}}} \gg \left|\frac{\partial^2 y}{\partial x_i^2}\sigma(x_i) + \frac{\partial^3 y}{\partial x_i^3}\sigma^2(x_i) + \cdots.\right|_{\boldsymbol{x}=\bar{\boldsymbol{x}}}$$

3.2.3 RC 電路範例

RC 串聯電路在外加定電壓 V_0 充放電的時候電容的電壓 V_c 會呈現指數式增加或衰減。移除外加電壓，電容放電使 V_c 衰減至最大值 V_0 的 $1/e$ 倍的時間 τ 稱作 RC 時間常數。由描述 RC

電路的一階微分方程式我們知道 $\tau = 1/RC$ ，時間常數可以透過在 RC 電路上施加方波電壓源，透過示波器觀察電容電壓 V_c 的變化，直接讀出 τ 值。

一般來說，電容的性質容易受溫度與信號頻率影響，測量電容的儀器遠較測量電阻的儀器來的貴。假如我們想透過測量 RC 時間常數的方式反推電容值（由示波器讀取刻度的誤差約爲 0.01 ms，電錶量測電阻值精準位數有三到四位），可以嘗試不同的 V_0 來得到一系列的 τ 測量值，配合電阻量測數據即能透過誤差傳遞的計算推估電容值。若實驗數據如下:

	1	2	3	4	5	6	7	8	9	10
R (MΩ)	5.067	5.050	5.034	5.069	5.057	5.070	5.042	5.069	5.054	5.031
τ (ms)	9.77	9.75	9.83	9.76	9.78	9.78	9.78	9.84	9.84	9.79

$$\begin{aligned}\sigma^2(C) &= \left(\frac{\partial}{\partial R}\frac{1}{R\tau}\right)^2\Big|_{\bar{\tau},\bar{R}}\sigma^2(R) + \left(\frac{\partial}{\partial \tau}\frac{1}{R\tau}\right)^2\Big|_{\bar{\tau},\bar{R}}\sigma^2(\tau) + \text{Covariance term} \\ &\simeq -\frac{1}{\bar{\tau}^2\bar{R}^2}\left[\frac{S_R^2}{\bar{R}^2} + \frac{S_\tau^2}{\bar{\tau}^2}\right] + \text{Covariance term}\end{aligned}$$

假設一系列的測量值已存進 `rMOhm` 與 `tauTime` 兩個 1×10 的陣列

```
rMOhmBar=sum(rMOhm)/10;  Sr=stdev(rMOhm'); // Meg ohm
tauTimeBar=sum(tauTime)/10;  Stau=stdev(tauTime'); // msec


cBar=1/(rMOhmBar*tauTimeBar);  //(1E-3*1E6)^-1= mF
varSum=(Sr^2/rMOhmBar^2+Stau^2/tauTimeBar^2);
Sc=sqrt(1/(rMOhmBar*tauTimeBar)^2*varSum); // mF
printf('The capacitor is %f uF with %f uF err.',cBar*10^3,Sc*10^3 );

------result------
The capacitor is 20.205406 uF with 0.090107 uF err.
```

求得之電容值爲 20.2 μF，相對誤差約 0.46 %。 一般而言電容上標示的製造誤差爲 1 至 10% ，且對於溫度極爲敏感，利用 RC 電路來量測電容值可輕易獲得準確的測量值。附帶一提，市售簡易的數位電容測量電路多半是配合已知電感器，使用 LC 振盪電路與數位合成之內部頻率比較來量出電容值，這類測量電路的精準度則受作爲基準的內部電感所限制。至於 covariant term 請讀者試著算算看，基本上應該遠小於 0.46 %。

3.3 Scilab 的曲線套適指令

Scilab 中最簡單的套適指令爲 fit_dat 。由 manual 可查到此指令的用法 (可視爲 datafit 這個指令的精簡版，本文討論僅限於 fit_dat 此指令):

```
[p,err]=fit_dat(G,p0,Z [,W] [,pmin,pmax] [,DG])
```

G 爲數據與套適函數的誤差項，要寫成 function 的形式， G 的自變數必須是待定參數陣列 p 與實驗數據 Z=[X;Y]，其中 X, Y 分別爲 1×n 的陣列。以線性套適 $y = ax + b$ 爲例， G 必須寫成 G(p,Z)，其中 p=[a;b] 。p0 則是事先必須給定的待定參數起始值，一般可以透過預先畫出數據點分佈圖來估計，若代入過度誇張猜測值很可能發生無效的套適結果。W 則是可以自行設定權重參數，預設值爲 1。 pmin, pmax 則是事先規範的停止條件，可以規避異常的套適結果。

3.3.1 以內建函數線性套適

若要將之前 reglin 的範例改用 fit_dat 來處理必須這麼寫:

```
x2=1:20;  y2=2*x2+8+5*rand(1,20);

function y=linearfit(x,p)
  y=p(1)*x+p(2);  endfunction

function e=G(p,Z)
  x=Z(1); y=Z(2);  e=y-linearfit(x,p); endfunction

[p,err]=fit_dat(G,[1;1],[x2;y2]);
[a2,b2,sig]=reglin(x2,y2);

printf('fit_data predict:  y= %f x+%f, err=%f\n',p(1),p(2),err);
printf('reglin predict:  a=%f,b=%f,stdev=%f',a2,b2,sig);

------result------
fit_data predict:  y= 2.061891 x+9.675984, err=45.111641
```

```
reglin predict:  a=2.063188,b=9.655670, stdev=1.501826
// 這裡所給出的 err ~= stdev^2*N
```

可以看到 fit_dat 與 reglin 得到的結果相仿。事實上 fit_dat 每次計算出來的結果都不太一樣，如果想要讓計算誤差收斂到比較好的結果通常可以這麼做:

```
p0=[200;200];  [p,err]=fit_dat(G,p0,[x2;y2]);  k=5;

for i=1:10,
[p,err]=fit_dat(G,p+k/i*rand(2,1),[x2;y2]);
printf('a=%f,b=%f, err=%f \n',p(1),p(2),err); end

------result------
a=2.053018, b=9.789958, err=45.193536
a=2.058689, b=9.719334, err=45.128482
a=2.063254, b=9.656360, err=45.109667
a=2.062373, b=9.665001, err=45.110080
a=2.064033, b=9.639521, err=45.111159
a=2.066919, b=9.604201, err=45.121905
a=2.062108, b=9.671400, err=45.110787
a=2.061047, b=9.686570, err=45.114092
a=2.057866, b=9.735685, err=45.140110
a=2.062128, b=9.669790, err=45.110552
```

程式碼中的 k 可微調，加入 rand 的目的在於不讓計算落入局部極值，本例中可能看不出影響（讀者可以試著把 p+k/i*rand(2,1) 改成 p 並觀察有何不同 ），分析大量的數據時可這樣的小技巧便能派上用場。以上即透過反覆迭代算出的 p0 以追求套適效果的一種嘗試。

3.3.2 高斯函數套適

高斯分佈又稱常態分佈，包含兩個參數: 期望值 μ 與標準差 σ。

$$P(x,\mu,\sigma)=\frac{1}{\sigma\sqrt{2\pi}}\exp\left[\frac{1}{2}(\frac{x-\mu}{\sigma})^2\right],\qquad x=(-\infty,\infty) \tag{3.1}$$

一維的隨機漫步 (random walks) 在移動步數極大的時候，最終落點與原點的可能距離會呈現高斯分佈，我們可以利用 scilab 模擬 1000 步的隨機漫步 1000 次來考察距離分佈的可能情況。

```
x=zeros(1,1000); id1=waitbar('cal...'); // 新增進度條監控

for num=1:1000,
waitbar(num/1000,id1);  // 在最外層迴圈中控制進度條
  for i=1:1000,
    if rand()>0.5, x(1,num)=x(1,num)+1;
    else  x(1,num)=x(1,num)-1;  end
  end
end

close(id1); data=nfreq(x);
```

nfreq 這個指令可以用來統計陣列中各個整數元素出現的次數，模擬結果計次後存入 data 中。接著我們便能使用高斯函數進行套適:

```
function y=gaussian(x)
  y=amp/sqrt(2*%pi)/sig*exp(-((x-mu)/sig)^2/2);  endfunction

function e=G(p,Z)
  x=Z(1);y=Z(2);  mu=p(1);sig=p(2);amp=p(3);
  e=y-gaussian(x);  endfunction

[pf,err]=fit_dat(G,[0;10;1000],data);
printf('mu= %f, sig= %f, amp= %f ',pf(1),pf(2),pf(3));

------result------
mu= -2.257571, sig= 21.999998, amp= 2003.557963
```

其中 amp=p(3) 爲縮放之因子。另外這裡的套適函數將 p 當成 global 變數來存取也是合法的寫法（注意到在之前線性套適的例子中 p 爲套適目標函數 linearft(x,p) 的輸入參數）。

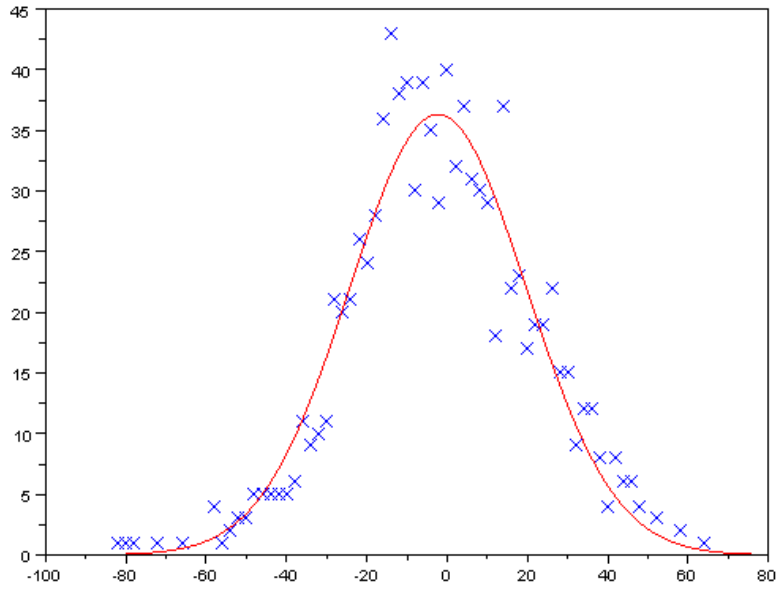


Figure 3.5: 藍點爲模擬數據，紅線爲高斯套適結果

```
plot(data(1,:),data(2,:),'bx');
t=-80:0.1:80;  mu=pf(1); sig=pf(2);amp=pf(3);  plot(t,gaussian(t),'r');
```

最後我們將套適結果與原始數據繪製成 Figure. 3.5

3.3.3 特殊非線性套適

一些比較複雜的非線性函數在套適時受到起始條件（p0）的影響相當大，爲了能有效地掃描參數可能的區域，我們必須採用一些更有效率的演算法來處理這些非線性套適的問題。在 Scilab 中，使用 `leastsq` 可以更快地獲得我們需要的結果。假設我們想要處理形如 $y = p_1 \sin(p_2 x) + p_2 \sin(p_1 x)$ 的非線性函數套適問題:

```
function y=nonlinear(x,p)
  y=p(1)*sin(p(2)*x)+p(2)*sin(p(1)*x); endfunction

t=1:0.1:20;  y1=nonlinear(t,[2.42,3.73])+rand(1,size(t,2));
```

取 $p_1 = 2.42,\ p_2 = 3.73$ 並加上雜訊作爲模擬測量資料。 如果使用 fit_dat 指令來處理這個問題，在條件不足的情況下僅知 $0 < p_1, p_2 < 5$ ，必須掃描所有區域才能找到最佳套適結果:

```
function e=Gn(p,Z)
  x=Z(1);y=Z(2);  e=y-nonlinear(x,p);  endfunction

cputime=0;
[pnf,errf]=fit_dat(Gn,rand(2,1),[t;y1]);

for i=1:5,
   for j=i:5,
      tic();  [pn,err]=fit_dat(Gn,[i;j],[t;y1]);
      tocc=toc();  cputime=cputime+tocc;
      // 紀錄個別套適耗時後並加總
          printf('p0=[%f;%f], err=%f ,cputime=%f \n',pn(1),pn(2),err,cputime);
      if err<errf,  // 紀錄參數掃描過程中 err 最小之點
      pnf=pn;  errf=err; end
   end
end

printf('p0=[%f;%f],err=%f \n',pnf(1),pnf(2),errf);
printf('total time=%f \n',cputime)
```

```
------result------
p0=[1.124922;2.028892], err=2350.596687 ,cputime=3.906000
p0=[1.113312;3.063340], err=2958.974389 ,cputime=9.703000
p0=[1.117388;4.085589], err=3620.006218 ,cputime=14.078000
...  ...   ...
p0=[4.125646;4.362520], err=4393.954481 ,cputime=65.030000
p0=[1.471315;-0.648141], err=2060.494570 ,cputime=67.670000

bestp0=[0.345196;0.467195], err=1951.564543
```

```
total cputime=67.670000

bestp0=[5.027308;2.396391], err=1266.069413
total time=53.126000
```

若是採用 leastsq 來計算，必須注意 leastsq 的參數要使用 list 指令將誤差函數與資料陣列連結，且資料陣列必須是行向量的形式（n×1）。

```
function e=Gn2(p,t,y)
  e=y-nonlinear(t,p);  endfunction

cputime=0;  [pnf2,errf]=fit_dat(Gn,rand(2,1),[t;y1]);

for i=1:5;
   for j=i:5;
     tic();  [f,popt,gr]=leastsq(list(Gn2,t',y1'),[i;j]);
     tocc=toc();  cputime=cputime+tocc;
     printf('p0=[%f;%f], err=%f ,cputime=%f\n',popt(1),popt(2),f,cputime);
     if f<errf,  pnf2=popt;  errf=f;  end
   end
end

------result------
p0=[0.000000;0.000000], err=1906.190047 ,cputime=0.000000
p0=[1.130642;2.028590], err=2349.876706 ,cputime=0.062000
p0=[1.120227;3.061375], err=2957.700769 ,cputime=0.124000
...  ...  ...
p0=[4.326925;3.744290], err=2922.883920 ,cputime=0.705000
p0=[1.921212;-2.183884], err=1852.675872 ,cputime=0.783000

bestp0=[1.921212;-2.183884],err=1852.675872
total cputime=0.783000
```

若以上述迴圈變化參數起始值，會得到如 Figure. 3.6 的結果，事實上兩種方法的結果都不算準確，但我們可以觀察到 leastsq 計算耗用的時間幾乎不到 fit_dat 的百分之三，若將

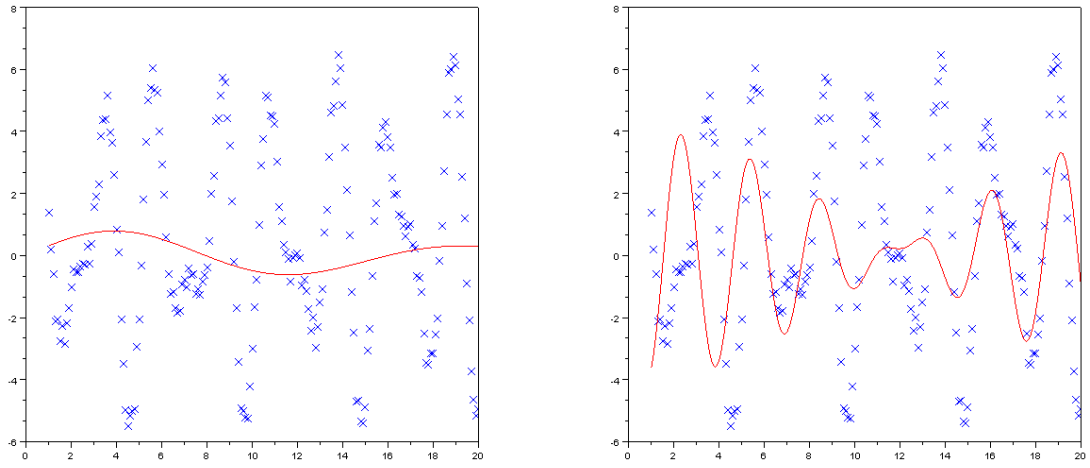


Figure 3.6: 左圖使用 `fit_dat` 套適，耗時 53 秒；右圖使用 `leastsq` 套適，耗時 0.8 秒

前者外層迴圈的 increament 降爲 0.1，則可得到:

```
for i=1:0.1:5,
   for j=1:5,
    *the same* end
end
*the same*

------result------
...  ...  ...
bestp0=[2.418844;3.729368],err=64.048530
total cputime=6.112000
```

稍加修改後便能得到相當準確的結果 (見 Figure. 3.7)，相較於 `fit_dat` ，改用 `leastsq` 可以大幅提昇非線性套適的效率。

實際應用此法來分析數據的範例可以參考 Section. ??。

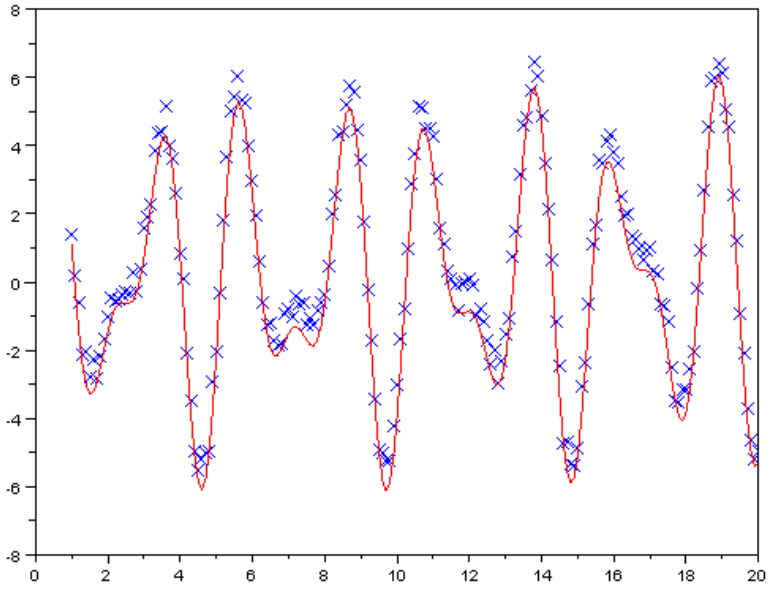


Figure 3.7: 縮減迴圈增量，僅耗時 6 s 即得準確之套適結果

3.4 題目演練

3.4.1 多項式套適

仿照線性套適的例子，我們只要擴展 `p` 的分量即可實現一般的多項式套適 (比方說 `y=p(1)+p(2)*x+p(3)*x^2+...`)。 假設有下列五筆資料:

x	8	19	9	16	13	7	24	22	21
y	631	725	821	1671	1565	461	-5481	-2085	-883

請分別使用二次、三次與四次多項式來進行套適，獲得如 Figure. 3.8 之結果，不同套適分析所得到 err 有何不同，判斷何者爲眞正的分佈。 若使用更高次的多項式進行套適是否可獲得更好的結果？ 是否有意義？

3.4.2 誤差的分佈

請進行一個簡單的實驗：手持碼表，按下開始鍵，默數五秒後再停止計時。重複進行這個實驗一百次後將結果存進一個 1×100 陣列 `data`，利用 scilab 的計次繪圖指令 `histplot(10,data)`

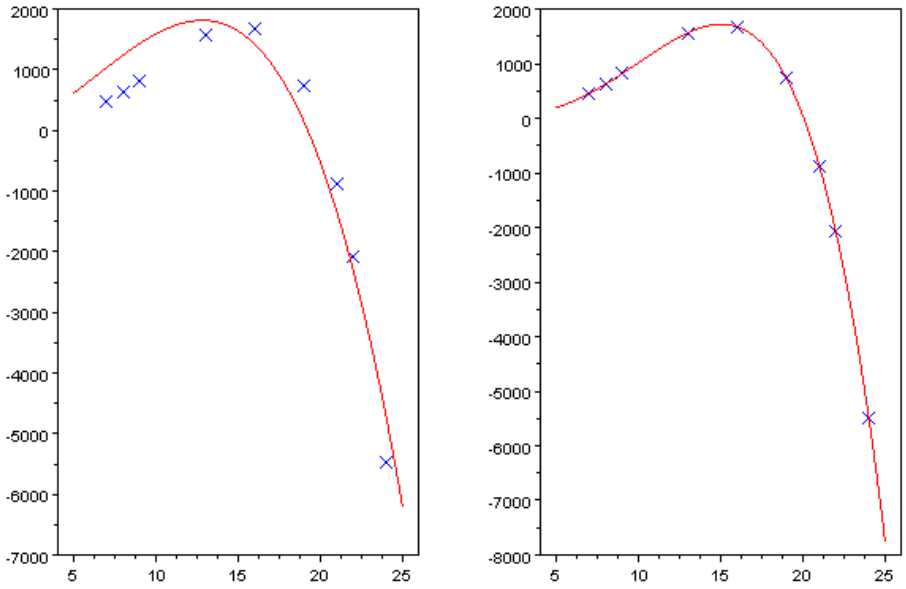


Figure 3.8: 左圖使用三次多項式套適，右圖使用四次多項式套適

來觀察實驗結果是否呈現以五秒爲平均值的高斯分佈。你可以利用下述寫好的 Scilab 小程式來紀錄:

```
global count; count=1;  global data; data=zeros(1,100);

fig=figure('position',[50 50 400 400]);

function trigger1()
global count;  ticc=tic();
time1=uicontrol(fig,'style','edit','string','set to zero',...
'position',[10 10 300 50],'fontsize',15);
count=count+1;  endfunction

function dat=trigger2()
global data count;
tocc=toc();  data(1,count)=tocc;
time1=uicontrol(fig,'style','edit','string',...
'No. '+string(count)+' test:'+string(data(1,count))+'sec',...
```

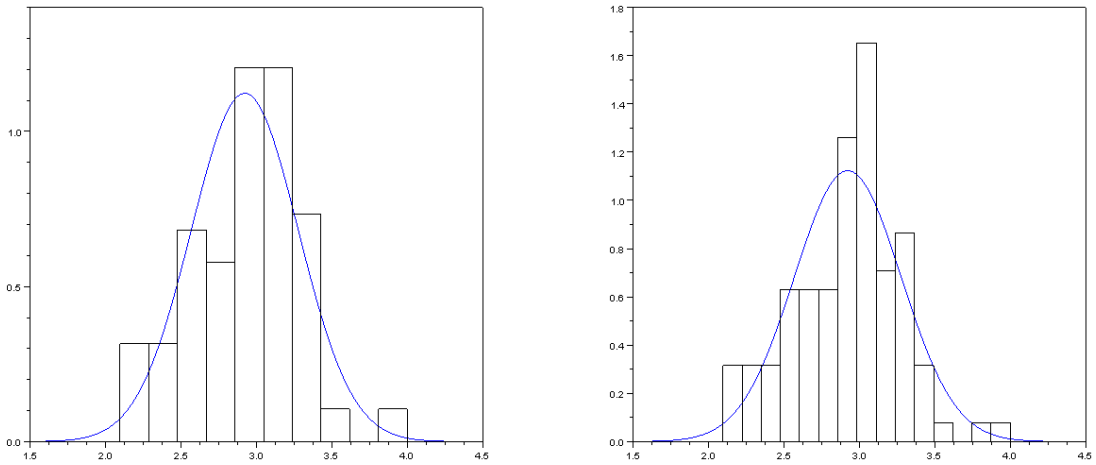


Figure 3.9: 左圖爲 `histplot(10,data)`，右圖爲 `histplot(15,data)`

```
'position',[10 10 300 50],'fontsize',15);  endfunction

uicontrol(fig,'style','pushbutton','position',[100 100 100 50],...
'string','start','callback','trigger1','fontsize',20);
uicontrol(fig,'style','pushbutton','position',[200 100 100 50],...
'string','stop&save' ,'callback','trigger2','fontsize',20);
```

最後利用 `sum` 與 `stdev` 指令算出數據的平均值與標準差，代入高斯函數並與數據累計圖繪至同一張圖上，判斷誤差分佈是否趨近於常態分佈，若多進行實驗是否可以減小標準差? Figure. 3.9 是默數三秒測試一百次的實驗結果。

3.4.3 測量 g 值

如果我想測量中大科四館三樓實驗室該處的重力加速度 g，我可以利用一組間隔已知，置放在眞空腔體中的光學偵測器，配合鋼珠的自由落體來進行實驗。假設這組偵測器的距離爲 0.1 m $\pm$ 0.05 mm (0.05 mm 即給定的標準差，偵測器鎖在同一根鋼柱上，兩偵測器的距離可以由上面的刻度讀出)。由於無法確定鋼珠掉落軌跡是否平行於偵測器之軸，我們還需要測量兩路徑的角度差異 θ 。g 值可透過鋼珠行經 L (1 m) 距離所需的時間 t 與 θ 求得:

$$\frac{1}{2}gt^2 = L\cos\theta \simeq L(1-\frac{\theta^2}{2}), \qquad g = \frac{2L[1-\cos(\theta^2/2)]}{t^2}.$$

角度極小的時候 $\cos\theta$ 可近似成 $1-\theta^2/2$ 。現有實驗數據如下:

	1	2	3	4	5	6	7
t (ms)	1405.835	1405.864	1405.623	1405.226	1404.829	1405.136	1404.224
θ (mrad)	45.2	18.7	6.2	2.9	7.8	21.7	0.5

請利用誤差傳播分析估計實驗得到的 g 值。θ 與 L 何者對 g 的誤差影響較大? 若不採用 $1-\theta^2/2$ ，使用原始未簡化的 $\cos\theta$ 來計算誤差傳播結果有何改變?

3.4.4 常數的測量

如果實驗物理課要求所有的同學進行上個例子中的 g 值測量實驗，課堂結束以後助教蒐集了各個實驗組的成果想要整理一份結果對教授報告，那麼助教該如何統計這些具有不同信賴區間的測量結果得出一個可靠的實驗推估值 g ?

問題在於我們如何將具備不同精度的觀測樣本結合在一起? 不同實驗組觀測樣本的系統誤差可能很大（真空腔的壓力不同，使用的儀器也有個別差異），事實上我們有很多統計手法來處理這類問題。許多物理常數或是粒子物理中基本粒子性質的測量都必須綜合世界上多個國家級實驗室的研究結果，經過統計處理才發表成公認的常數表供所有實驗物理學家使用。這些常數的估計並不是單純將所有數據加起來平均即可，採用不同的誤差分佈假設與權重（如果每個人都宣稱他在車庫裡量出了萬有引力常數 G，難道我們都要把他的實驗值放到我們的數據庫一同分析嗎？）也會有不同結果。

Appendices

Appendix A

四階 Runge-Kutta 法的證明

A.1 推導過程

參考《數值方法與程式》林聰悟 、林佳慧一書中第十二章有關常微分方程數值解法分析，利用泰勒展開式即可導出 Runge-Kutta 法之係數。假設足碼代表偏微分，$f_t = \partial f/\partial t$，$f_{yy} = \partial^2 f/\partial y^2$ ， 全微分 (對 t) 以 f' 表示。已知 $dy/dt = f(t,y)$ ，

$$y_{i+1} = y_i + hf_i + h^2\frac{f'}{2!} + h^3\frac{f''}{3!} + h^4 f''4! + O(h^5) \tag{A.1}$$

其中

$$\begin{aligned} f' &= f_t + ff_y \\ f'' &= f_{tt} + 2ff_{ty} + f^2 f_{yy} + f_t f_y + ff_y^2 \\ f''' &= (f_{ttt} + 3ff_{tty} + 3f^2 f_{tyy} + f^3 f_{yyy}) + f_y(f_{tt} + 2ff_{ty} + f_{yy}^f) \\ &+ 3(f_{tfty} + ff_y f_{ty} + ff_t f_{yy} + f^2 f_y f_{yy}) + (f_t f_y^2 + f_{fy}^3) \end{aligned} \tag{A.2}$$

我們將使用 $y_{i+1} = y_i + h(\gamma_0 f_i + \gamma_1 f_j^a + \gamma_2 f_j^b + \gamma_3 f_j^c)$ 來計算下個 step 的函數值 ，其中

$$
\begin{aligned}
f_i &= f(t_i, y_i) \\
f_i^a &= f(t_i + \alpha_1 h, y_i + \beta_1 f_i h) \\
f_i^b &= f(t_i + \alpha_2 h, y_i + \beta_2 f_i h + \beta_3 f_i^a h) \\
f_i^c &= f(t_i + \alpha_3 h, y_i + \beta_4 f_i h + \beta_5 f_i^a h + \beta_6 f_i^b h)
\end{aligned} \tag{A.3}
$$

我們同樣可以將 f_i^a, f_i^b, f_i^c 展開成泰勒級數，取四階近次似後再代回 Eq. (A.1)，透過比較係數找出適合的未定係數 $\alpha_{i,j}$ 與 γ_j 之值。

$$
\begin{aligned}
f_i^a &= f^a(0) + h f^{a\prime}(0) + h^2 \frac{f^{a\prime\prime}(0)}{2!} + h^3 \frac{f^{a\prime\prime\prime}(0)}{3!} + o(h^4) \\
f^a(0) &= f \\
f^{a\prime}(0) &= \alpha_1 f_t +_1 f f_y \\
f^{a\prime\prime}(0) &= \alpha_1^2 f_{tt} + 2\alpha_1\beta_1 f f_{ty} + \beta_1^2 f^2 f_{yy} \\
f^{a\prime\prime\prime}(0) &= \alpha_1^3 f_{ttt} + 3\alpha_1^2\beta_1 f f_{tty} + 3\alpha_1\beta_1^2 f^2 f_{tyy} + \beta_1^3 f^3 f_{yyy}
\end{aligned} \tag{A.4}
$$

f_i^b 與 f_j^c 類推。比較包含未定係數的計算式與其泰勒展開式同階次的係數，可以得到 11 條等式。由於總共只有 9 個變數，四階精度的 Rouge-Kutta 法有多種係數組合可選擇。

$$
\begin{aligned}
\alpha_1 &= \beta_1 \\
\alpha_2 &= \beta_2 + \beta_3 \\
\alpha_3 &= \beta_4 + \beta_5 + \beta_6 \\
f : 1 &= \gamma_0\gamma_1 + \gamma_2 + \gamma_3 \\
f_t + f f_y : \frac{1}{2} &= \gamma_1\alpha_1 + \gamma_2\alpha_2 + \gamma_3\alpha_3 \\
f_t t + 2 f f_{ty} + f^2 f_{yy} : \frac{1}{3} &= \gamma_1\alpha_1^2 + \gamma_2\alpha_2^2 + \gamma_3\alpha_3^2 \\
f_{ttt} + 3 f f_{ftty} + 3 f^2 f_{tyy} + f^3 f_{yyy} : \frac{1}{4} &= \gamma_2\alpha_1\beta_3 + \gamma_3(\alpha_1\beta_5 + \alpha_{25}) \\
f_y(f_{tt} + 2 f f_{ty} + f^2) f_{yy} : \frac{1}{12} &= \gamma_2\alpha_1^2\beta_3 + \gamma_3(\alpha_1^2\beta_5 + \alpha_{25}^2) \\
f_{ty}(f_t + f f_y) + f_{yy}(f f_t + f^2 f_y) : \frac{1}{8} &= \gamma_2\alpha_1\alpha_1\beta_3 + \gamma_3\alpha_3(\alpha_1\beta_5 + \alpha_{25}) \\
f_t f_y^2 + f f_y^3 : \frac{1}{24} &= \gamma_3\alpha_1\beta_3\beta_6
\end{aligned}
$$

(A.5)

A.2 思考問題

1. 試著自己推導出完整的計算過程
2. 試著推導五階 Runge-Kutta 法的係數

Appendix B

套用 LaTeX 範本- RevTeX

RevTeX 是 American Physics Society 所開發的 LaTeX 範本(https://authors.aps.org/revtex4/)，該學會旗下著名的期刊包含 Physical Review 系列以及 AIP 的 Applied physical letter, Journal of Applied Physics 等皆接受此格式之稿件。正式出版的期刊文章絕大多數以 two-column 形式的刊行，RevTeX提供的格式緊湊而美觀，十分適合用來排版短篇的科學報告。

Windows 上的 MikTeX 有收錄最新版的 RevTeX範本，可以從 Maintenance → Package Manager 中使用 revtex 作爲關鍵字搜尋並安裝使用。以下則提供引用該範本之程式碼(純英文的稿件，故使用 pdfLatex 來編譯)，編譯完成的結果請參照 Figure. B.1 與 B.2:

```
\documentclasspra,twocolumn,reprint{revtex4}
% 取用 Physical Review A, 雙欄位，預印本的格式
\usepackage{amsmath}
\usepackage{graphicx}
\usepackage{natbib}

\begin{document}

\begin{abstract}
You should put abstract here. Briefly describe what you've done and
the main idea of your project in ``abstract''. Generally the abstract
contains 60 to 100 words. Using 2 or 3 sentences to clealy define your
work and point out the main idea in it.
```

```
\end{abstract}
% abstract 環境中的段落會被放到雙欄位上方與作者區塊下方的位置


\title{Ex. Student project: Refractive index measurement of glass and polymer}
\author{Chih-Han Lin, Chengfred and Yuhung Lai}
\affiliation{Experimental Physics Laboratory, Class A, Group 13}
\affiliation{Department of Physics, National Central University}

\date{8 December 2010}

\maketitle

% title 形式與預設雷同，多了 affilation 這個項目，此處文字會以斜體置中
% 排入作者下方之欄位
\section{introduction}
You should introduce the whole background and make short discuss in this
section just like pre-report in previous laboratory experiment. There may
be a lot of method to practice your idea, you have to tell me why choosing
one of them but not others. Collect and read journals or reliable reference
articles to help you listing the advantage and disadvantage of these method.

Here is an example of citation\cite{prl1,prl2,prl3} and book citation
\cite{book1}.

\section{theory or method}
You can express the main idea of your project in this section. Use
{\verb $...$ } to present inline math equation. For example,  \verb+
$a_1^2+a_2^2=c_1^2$+ will show $a_1^2+a_2^2=c_1^2$ in paragraph. If
you want to type numerated independent equations, you can use \verb+
eqnarray+ environments.

\begin{verbatim}
\begin{eqnarray}\label{example-eq}
```

```
a_1^2+b_1^2&=&c_1^2 \\
a_2^2+b_2^2&=&c_2^2.
\end{eqnarray}
\end{verbatim}

will display follows:

\begin{eqnarray}
a_1^2+b_1^2=c_1^2 \\
a_2^2+b_2^2=c_2^2.
\end{eqnarray}

\section{Characteristic of the sensor}
You need to specify the sensor used in your own experiment during 3-weeks
project.

\section{Experimental setup}

\subsection{refractive index of glass}
Using \verb+\subsection+ to generate a subsection.

\subsection{refractive index of polymer}
Use \verb+figure+ environment to include figures in \LaTeX ~documents.
It better for you to edit your data plot or picture with Inkscape or other
vector graphic editor and save them as \verb+.png+ \verb+.eps+ or
\verb+.pdf+ file. For example, following codes shows FIG.~ \ref{test-pic}.

\begin{verbatim}
\begin{figure}
\centering
\includegraphicswidth=0.4\textwidth
{test-pic.pdf}
\caption{This is test-pic picture}
\label{test-pic}
```

```
\end{figure}
\end{verbatim}

Commonds \verb+\label+ and \verb+\ref+ are used to extract figure or
equation number automatically  generated by \LaTeX. \begin{verbatim}FIG.~
\ref{test-pic}\end{verbatim} will show FIG.~ \ref{test-pic}.



\begin{figure}
\centering
\includegraphicswidth=0.4\textwidth{test-pic.pdf}
\caption{This is test-pic picture}
\label{test-pic}
\end{figure}




\section{result}
Express your result and data plot here.

\section{conclusions}
Discuss and make conclusions here.

\appendix
\section{Arduino code}
Command {\verb \appendix } signals that all following sections are
appendices. For example, the section ``Arduino code" after
\verb+\appendix+ has prefix ``Appendix''. Use \verb+verbatim+
environvemt to list your computing code as follows:

\begin{verbatim}

const int buttonPin = 2;
```

```
const int ledPin =  13;

// Variables will change:
int ledState = HIGH;
int buttonState;
int lastButtonState = LOW;



long lastDebounceTime = 0;
long debounceDelay = 50;

void setup() {
  pinMode(buttonPin, INPUT);
  pinMode(ledPin, OUTPUT);
}

void loop() {
  // read the state of the switch into
  //a local variable:
  int reading = digitalRead(buttonPin);

  // check to see if you just pressed
  //the button (i.e. the input went from
  //LOW to HIGH), and you've waited
  //long enough since the last press to
  //ignore any noise:

  // If the switch changed, due to noise
  //or pressing:
  if (reading != lastButtonState) {
    // reset the debouncing timer
    lastDebounceTime = millis();
  }
```

```
  if ((millis() - lastDebounceTime)
     > debounceDelay) {
    // whatever the reading is at, it's been
    //there for longer than the debounce delay,
    // so take it as the actual current state:
    buttonState = reading;
  }

  // set the LED using the state of the button:
  digitalWrite(ledPin, buttonState);

  // save the reading.  Next time through
  //the loop, it'll be the lastButtonState:
  lastButtonState = reading;
}
\end{verbatim}


\begin{acknowledgements}
If you want to acknowledge someone, put the paragraph in
\verb+acknowledgements+ environment. Ex. Chih-Han Lin acknowledges
fruitful discussion and patient debugging with Chengfred, and
partial support by department of Physics of NCU.
\end{acknowledgements}

\bibliography{testbib}
%save your bib database in testbib.bib
\end{document}
```

Ex. Student project: Refractive index measurement of glass and polymer

Chih-Han Lin, Chengfred and Yuhung Lai
Experimental Physics Laboratory, Class A, Group 13 and
Department of Physics, National Central University
(Dated: 8 December 2010)

You should put abstract here. Briefly describe what you've done and the main idea of your project in "abstract". Generally the abstract contains 60 to 100 words. Using 2 or 3 sentences to clealy define your work and point out the main idea in it.

I. INTRODUCTION

You should introduce the whole background and make short discuss in this section just like pre-report in previous laboratory experiment. There may be a lot of method to practice your idea, you have to tell me why choosing one of them but not others. Collect and read journals or reliable reference articles to help you listing the advantage and disadvantage of these method.

Here is an example of citation[1–3] and book citation [4].

II. THEORY OR METHOD

You can express the main idea of your project in this section. Use `$...$` to present inline math equation. For example, `$a_1^2+a_2^2=c_1^2$` will show $a_1^2+a_2^2=c_1^2$ in paragraph. If you want to type numerated independent equations, you can use `eqnarray` environments.

```
\begin{eqnarray}\label{example-eq}
a_1^2+b_1^2&=&c_1^2 \\
a_2^2+b_2^2&=&c_2^2.
\end{eqnarray}
```

will display follows:

$$a_1^2 + b_1^2 = c_1^2 \tag{1}$$
$$a_2^2 + b_2^2 = c_2^2. \tag{2}$$

III. CHARACTERISTIC OF THE SENSOR

You need to specify the sensor used in your own experiment during 3-weeks project.

IV. EXPERIMENTAL SETUP

A. refractive index of glass

Using `\subsection` to generate a subsection.

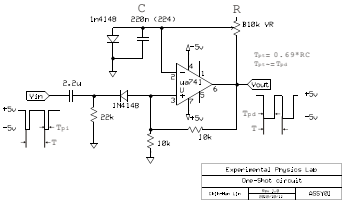


FIG. 1: This is test-pic picture

B. refractive index of polymer

Use `figure` environment to include figures in LaTeX documents. It better for you to edit your data plot or picture with Inkscape or other vector graphic editor and save them as `.png .eps` or `.pdf` file. For example, following codes shows FIG. 1.

```
\begin{figure}
\centering
\includegraphics[width=0.4\textwidth]
{test-pic.pdf}
\caption{This is test-pic picture}
\label{test-pic}
\end{figure}
```

Commonds `\label` and `\ref` are used to extract figure or equation number automatically generated by LaTeX.

```
FIG.~ \ref{test-pic}
```

will show FIG. 1.

V. RESULT

Express your result and data plot here.

VI. CONCLUSIONS

Discuss and make conclusions here.

Figure B.1: 以 ReVTEX 爲範本編譯而成的文章第一頁

APPENDIX A: ARDUINO CODE

Command `\appendix` signals that all following sections are appendices. For example, the section "Arduino code" after `\appendix` has prefix "Appendix". Use `verbatim` environvemt to list your computing code as follows:

```
const int buttonPin = 2;
const int ledPin =  13;

// Variables will change:
int ledState = HIGH;
int buttonState;
int lastButtonState = LOW;


long lastDebounceTime = 0;
long debounceDelay = 50;

void setup() {
  pinMode(buttonPin, INPUT);
  pinMode(ledPin, OUTPUT);
}

void loop() {
  // read the state of the switch into
  //a local variable:
  int reading = digitalRead(buttonPin);

  // check to see if you just pressed
  //the button (i.e. the input went from
  //LOW to HIGH), and you've waited
  //long enough since the last press to
  //ignore any noise:

  // If the switch changed, due to noise
  //or pressing:
  if (reading != lastButtonState) {
    // reset the debouncing timer
    lastDebounceTime = millis();
  }

  if ((millis() - lastDebounceTime)
     > debounceDelay) {
    // whatever the reading is at, it's been
    //there for longer than the debounce delay,
    // so take it as the actual current state:
    buttonState = reading;
  }

  // set the LED using the state of the button:
  digitalWrite(ledPin, buttonState);

  // save the reading.  Next time through
  //the loop, it'll be the lastButtonState:
  lastButtonState = reading;
}
```

ACKNOWLEDGMENTS

If you want to acknowledge someone, put the paragraph in `acknowledgements` environment. Ex. Chih-Han Lin acknowledges fruitful discussion and patient debugging with Chengfred, and partial support by department of Physics of NCU.

[1] P. Wölfle, Y. Dubi, and A. V. Balatsky, Phys. Rev. Lett. **105**, 246401 (2010).
[2] C. Recher, M. Kieburg, and T. Guhr, Phys. Rev. Lett. **105**, 244101 (2010).
[3] W. Bao, Z. Zhao, H. Zhang, G. Liu, P. Kratz, L. Jing, J. Velasco, D. Smirnov, and C. N. Lau, Phys. Rev. Lett. **105**, 246601 (2010).
[4] A. Yariv, *Quantum Electronics* (Wiley, 1989).

Figure B.2: 以 ReVTEX 爲範本編譯而成的文章第二頁