淺談數值計算

張　康

1. 導言

什麼叫做數值計算 (Numerical Computation)? 簡單地說, 它就是發展一種計算的方法, 以利用電腦來解數學問題。所以, 它包括了許多層面, 最外一層是應用, 其次是數值方法之建立與數學分析, 再來是該數值方法之電腦程式設計。

舉例來說, 我們談「太空科技」, 您可曾想過太空船爲什麼能登陸月球呢? 如何能正確地抵達目的地呢? 必需先把該飛行的數學模式列出來, 然後求該模式的解。這些繁瑣複雜的式子, 要靠有效率而精確的數值方法, 才能快速而正確地得解, 以即時導航。其他如飛機的設計, 電子電路的設計, 分析地震對橋樑或建築物的影響, 科學氣象預報, 或單純至如何使公司的利潤最大, 使工廠的生產系統最佳化等, 這些都是數不盡的應用。抽絲剝繭後來看, 去掉了最外層的應用, 剩下的就是一個單純的數學模式求解之問題。通常, 這一類的問題, 往往較複雜而不能直接用解析的方式求解, 必需要用數值方法在電腦上解該數學問題。這就是數值計算的領域, 它可以說是應用科學的工具。所以, 不管是唸理、工或商的學生, 都應該學習這門課程, 因爲它結合了應用、數學與電腦。

本文中將數值計算做一個簡要的介紹, 希望能激起大家的興趣, 進而探討此一領域, 共同爲科技發展貢獻心力。另外, 市面上這方面的書在名稱上和性質上大體分爲兩類, 即數值方法 (Numerical Methods) 和數值分析 (Numerical Analysis)。前者較重方法與應用, 後者較多數學推導與分析, 可以依個人之需要來選擇, 本文則以數值計算通稱, 而將一般性數值計算所討論之課題分別介紹於下節中。

2. 數值計算的重要課題

本節中我們利用一些應用問題來介紹數值方法解題之必要性。並將數值計算所包括的幾個重要課題概略討論。

2.1 解方程式的根

科學計算中常會遇到要解方程式的根, 可是又找不到求根的公式, 要如何求呢? 舉個簡單的例子:

例1: 計算行星的軌道時要解 Kepler's

方程式

$$x - a\sin x = b \qquad (1)$$

其中a和b爲已知。

沒辦法寫出根的公式, 怎麼辦呢? 數值方法中倒是有許多方法可以解決它, 譬如牛頓法 (Newton's method) 求$f(x) = 0$時用

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)},\ n = 0, 1, 2, \cdots \quad (2)$$

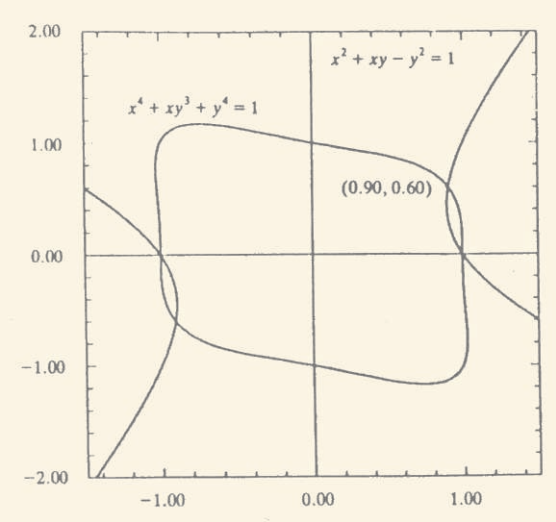
來迭代出一個數列(x_n), 而x_0若選得適當,(x_n)就會逼近$f(x)$之根。這就是一個數值計算的方法, 只要知道題目, 如 (1) 式, 並且會微分就可以代入 (2) 式了。只是選 x_0的學問可不少, 不是三言兩語可以交代, 在這個課題中可以深入討論。

另外, 例1中只有一個未知數, 如果有n個方程式,n個未知數要聯立求解呢? 上面的觀念推廣, 就成爲解非線性聯立方程組的課題了。例如:

例2: 求平面上下列兩個曲線 (見圖一) 的交點

$$\begin{cases} x^4 + xy^3 + y^4 = 1 \\ x^2 + xy - y^2 = 1 \end{cases} \qquad (3)$$

假設已知$(0.9, 0.6)$附近有一個交點, 能否更精確地求出來?



圖一 兩個曲線方程式之圖形

想要直接解 (3) 式並不容易, 但數值方法就可以解決它。例如牛頓法 (2) 之推廣, 把解$f(x) = 0$看成解向量函數

$$\mathbf{F}(\mathbf{x}) = \begin{bmatrix} f_1(x_1, x_2, \ldots, x_n) \\ f_2(x_1, x_2, \ldots, x_n) \\ \vdots \\ f_n(x_1, x_2, \ldots, x_n) \end{bmatrix} = 0 \quad (4)$$

而 (2) 式亦可以推廣, 我們不再深入, 但相信大家已經體會到了用數值方法在電腦中計算求解應是較可行之道。

2.2 數值微分與數值積分

想要求某個可微函數$T(x)$的微分, 只要學過微積分的都會。但是, 有些時候根本沒有$T(x)$函數而只是一些實驗數據, 例如

例3: 一個太陽能加熱系統中, 設x爲到被加熱表面的距離,$T(x)$爲所測得之溫度如下

x (m)	0.2	0.4	0.6	0.8	1.0	1.2
$T(x)(^\circ c)$	20.	21.3	22.5	20.2	18.5	16.

這裡$\frac{dT}{dx}$和熱的傳導率有關, 而$\frac{d^2T}{dx^2}$和能量的得失有關, 所以我們想求各點之$T'(x)$,$T''(x)$值。要怎麼求呢？

上面又是個必需靠數值方法來近似求解的例題, 譬如用中央差分近似法 (Central Difference Approximation) 知

$$f'(x) \approx \frac{f(x+h)-f(x-h)}{2h} \quad (5)$$

$$f''(x) \approx \frac{f(x+h)-2f(x)+f(x-h)}{h^2} \quad (6)$$

我們利用 (5) 式和 (6) 式就不難求出例3中$0.4 \sim 1.0$各點之一次和二次微分之近似值了。當然, 兩個端點之微分值亦可求得, 其他尚有許多不同方式來估計它們。

還有, 學過微積分的同學都知道有許多函數實在無法寫出它的反導函數來, 要怎麼積分呢 ?例如:

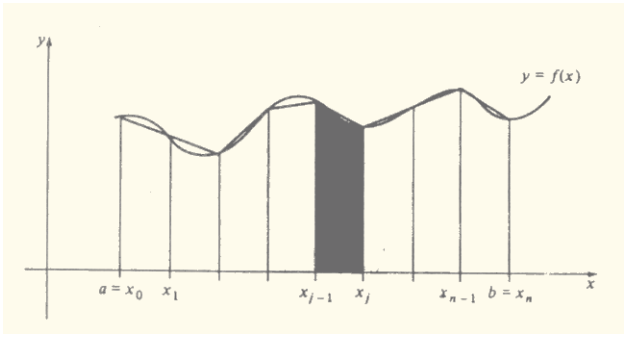
例4: 電場理論中證明過, 一個圓形電線迴路中電流所產生的磁場強度爲

$$H(x) = \frac{4\ell r}{r^2-x^2}\int_0^{\frac{\pi}{2}} \sqrt{1-(\frac{x}{r})^2\sin^2\theta}\,d\theta \quad (7)$$

其中ℓ爲電流,r爲迴路之半徑,x爲所欲測量磁場強度之點到圓心的距離$(0 \le x \le r)$。假設$\ell = 15.3$,$r = 120$和$x = 84$爲已知, 這個積分並不易做。但是, 數值方法可以求得$H = 1.355661135$, 精確到小數點以下九位呢!

求積分就是求面積, 最簡單如組合梯形法 (Composite Trapezoide Rule) 利用電腦的快速大量計算之能力, 把欲求積分的區間$[a,b]$細細地分割, 如圖二所示, 那些梯形面積的和 ,就是積分的近似值了。即

$$\int_a^b f(x)dx \approx \frac{h}{2}[f(a)+2\sum_{j=1}^{N-1} f(a+jh)+f(b)] \quad (8)$$



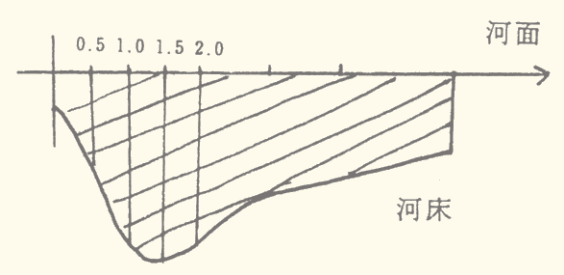
圖二 組合梯形法求$\int_a^b f(x)dx$

所以, 不論$f(x)$有多繁複,(8) 式皆可輕易地算出積分之近似值。更進一步地可 以看出,(8) 式實際上並不需要$f(x)$之公式, 只要知道$f(x_j)$之值就足夠了, 所以利用實驗量得的數據, 仍然可以用來做積分了, 例如:

例5: 水利工程師想知道某河流的橫斷面面積, 雖然只能量出不同點之河床深度爲

x	0	0.5	1.0	1.5	2.0	$\cdots$
水深	0.5	1.2	2.5	2.7	2.5	$\cdots$

如圖三所示:

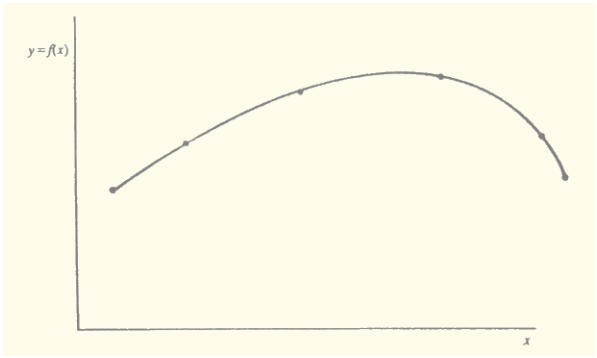


圖三 河流的橫斷面

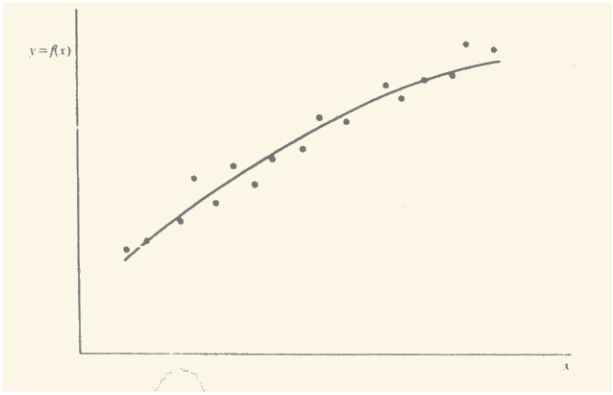
簡易的方法如梯形法和 Simpson 法都是屬於低階的封閉式 Newton-Cotes 公式 (Closed Newton-Cotes Formula), 想提高精確度時可以用較高階的公式。另外, 如果 $f(a)$ 或 $f(b)$ 爲未知或 $\pm\infty$, 則可以選用開放式的 Newton-Cotes 公式 (Open Newton-Cotes Formula) 來逼近。再則, 如圖二中利用組合的觀念, 把區間分割得十分小, 每個小區間用 Newton-Cotes 類公式來逼近, 再全體組合起來, 又可以提高精確度。其他如 Romberg 積分法 (Romberg Integration),Gauss求積法 (Gauss Quadrature) 都是高精確度的近似法。還有, 多重積分亦可層層解決, 只要知道$f(\mathbf{x})$即可。

2.3 內插 (Interpolation) 與曲線撮合 (Curve Fitting)

如果您做完了實驗, 取得了一大堆(x, y)關係的數據, 要如何求其他點之值呢? 如例3中溫度分佈的數據, 怎麼推測$T(0.3)$,$T(0.5)$,$T(1.02)$,$\cdots$各點之值呢? 或是找一個過某些數據點的曲線呢? 如圖四所示。另外, 也許您的實驗數據數量多而又有些許誤差, 找出過這些點的函數反而不應用, 您要的是一個接近這些點的平滑曲線, 如圖五所示。要如何求呢?



圖四 過數據點之函數



圖五 近數據點之最佳函數

上面兩個目的用數學表示即爲: 給一組數據

x	x_1	x_2	$\cdots$	x_n
y	y_1	y_2	$\cdots$	y_n

想要得如圖四之函數即是求f, 使得$y_i = f(x_i)$,$i = 1, 2, \cdots, n$。而圖五則可以求f使得

$$\sum_{i=1}^{n}(f(x_i) - y_i)^2 \qquad (9)$$

爲最小, 即稱爲最小平方近似法 (Least Squares Approximation)。上面 (9) 式這個觀念還可以推廣, 視不同的x_i之比重不同,

即插入一個加權函數 (Weight Function) $w(x) \geq 0$, 而來找一個函數f使得

$$\sum_{i=1}^{n}(f(x_i)-y_i)^2 w(x_i) \qquad (10)$$

爲極小。另外, 也不一定要極小化 (9) 式, 也可以要求整體誤差爲極小, 即

$$\min \sum_{i=1}^{n} |f(x_i)-y_i| \qquad (11)$$

或是要求最大誤差爲極小, 即

$$\min \max_{1\leq i\leq m} |f(x_i)-y_i| \qquad (12)$$

稱爲極小極大法 (Minimax Method)。

這個畫曲線的課題再深一層發展可是變化萬千。較理論的如近似理論 (Approximition Theory), 而較應用的如電腦繪圖、電腦輔助設計等等。上自科技應用, 下至生活應用如服裝設計、卡通漫畫, 都是它的觀念之發展。例如:

例6: 圖六中這個女孩子的側面輪廓是利用16個平滑的三次多項式連接而成的。這種曲線稱爲三次樣條函數 (Cubic Spline), 它是先選取17個節點, 然後在每兩點間找出一個三次多項式來, 要求這些多項式彼此之間連續, 而且一次導數和二次導數亦連續, 而求出來的16個三次多項式之圖形。

圖六 三次樣條函數繪出女孩外廓

2.4 數值微分方程

顧名思義, 它包括了解常微分方程式和偏微分方程式。由於工程與科學的應用中, 許多問題之數學模式常常是個解微分方程式的問題。往往有許多微分方程式無法用解析的方法來求解, 或是應用的本身亦不需要函數的顯式 (Explicit Form), 而是要它在某些點之「值」, 所以要借助數值方法來求解了。

例7: 一個半導體之熱傳導函數滿足下列微分方程式

$$\begin{cases} \dfrac{d^2x(t)}{dt^2} = -e^{-x(t)} \\ \\ x(0) = x(1) = 0 \end{cases} \qquad (13)$$

請問$x(0.1)$,$x(0.2)$,$\cdots$,$x(0.9)$各點之值爲何?

由於數值方法是求函數在某些點之值, 通常我們先選一個網格大小 (Mesh Size)h, 而將所欲探討之區間$[0,1]$分格。如例 7 中選$h = 0.1$, 於是$t_1 = 0.1$,$t_2 = 0.2$, ...,$t_9 = 0.9$, 令$x_i = x(t_i)$, 例7即爲求$x_1, x_2, \ldots, x_9$之問題。

解微分方程式就要用到前面數值微分之觀念, 我們必需將微分方程式離散化 (Discre- tize) 而改寫爲差分方程式 (Difference Equation)。例如用右式之中央差分法來近似二次微分, 令$x_i = x(t_i)$, 則例7中的 (13) 式可改爲

$$x_{i-1}-2x_i+x_{i+1} = -e^{-x_i}\cdot h^2, i = 1, 2, \cdots, 9 \tag{14}$$

其邊界值條件即爲

$$x_0 = x_{10} = 0 \tag{15}$$

將 (14) 和 (15) 式仔細列出來, 就成爲解下列非線性聯立方程組的問題:

$$\begin{cases} 2x_1 - x_2 - 0.01e^{-x_1} = 0 \\ -x_1 + 2x_2 - x_3 - 0.01e^{-x_2} = 0 \\ -x_2 + 2x_3 - x_4 - 0.01e^{-x_3} = 0 \\ \vdots \\ -x_7 + 2x_8 - x_9 - 0.01e^{-x_8} = 0 \\ -x_8 + 2x_9 - 0.01e^{-x_9} = 0 \end{cases} \tag{16}$$

所以, 研讀數值微分方程一定要先把基礎的數值計算學好。不同型式的微分、偏微分方程式將應用不同的數值方法, 如何將微分方程式離散化之探討也十分重要, 除了前面所提的有限差分法外, 尚有有限元素法 (Finite Element Method), 有限體積法 (Finite Volume Method), 處處皆學問呢。更重要的, 用數值方法解了題目後答案到底可不可靠呢? 數值方法的穩定性 (Stability) 及計算機之捨入誤差 (Round-off Error) 的擴增都是不能忽略的課題。

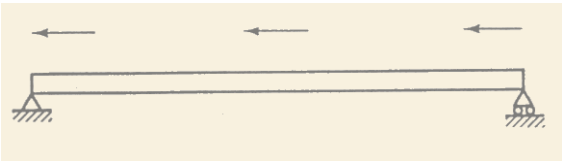
2.5 固有值問題 (Eigenvalue Problem)

給一個方陣A, 若常數λ和向量x滿足

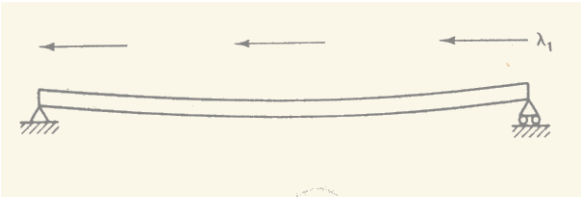
$$Ax = \lambda x \tag{17}$$

則稱λ爲A之固有值 (Eigenvalue), 而x爲對應於λ的固有向量 (Eigenvector)。固有值問題即爲尋求滿足 (17) 式之λ和x的問題。許多應用和固有值有關, 譬如國家音樂廳的管風琴, 琴管的長度是個固有值問題之解呢!吉他絃的調音亦如此, 橋樑的結構也和固有值有關呢!下面舉個例子來說明:

例8: 圖七 (a) 中爲一個簡單支撐的桁樑, 當一個重力加上去後桁樑彎曲如圖七 (b) 所示, 如果重力超過某一臨界值λ_1時它甚至會斷裂。



(a)



(b)

圖七 簡單支撐的桁樑

令$x(t)$表示t點位置桁樑彎曲之位移, 則λ_1爲滿足下列微分方程式之最小值

$$\frac{-d^2x(t)}{dt^2} = \lambda x(t),\ 0 \le t \le 1 \qquad (18)$$

而$x(0) = x(1) = 0$。

上面 (18) 式是個微分方程式, 若用 (6) 式之中央差分法來轉換可得

$$h^{-2}(-x_{i-1} + 2x_i - x_{i+1}) = \lambda x_i,$$
$$i = 1, 2, \cdots, N-1 \qquad (19)$$

及邊界值條件

$$x_0 = x_N = 0 \qquad (20)$$

其中$h = 1/N$, 則$x_1, x_2, \ldots, x_{N-1}$爲未知數。將 (19),(20) 式列爲矩陣向量形式即得

$$N^2 \begin{bmatrix} 2 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & \ddots & & & & \\ & & & -1 & 2 & -1 \\ & & & & -1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \\ x_{N-1} \end{bmatrix}$$
$$= \lambda \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \\ x_{N-1} \end{bmatrix} \qquad (21)$$

原來它是一個解固有值問題。而且N愈大時 (21) 式中之最小固有值才愈接近原始題目 (18) 式所要找的λ_1。所以我們要求當N很大時 (21) 式之最小固有值, 那只有靠在電腦上用數值方法來求得了。

解 (21) 式常用的方法如QR方法, 令$A_0 = A$, 將A_0分解爲

$$A_0 = Q_0 R_0 \qquad (22)$$

其中Q_0爲正交矩陣而R_0爲上三角矩陣。則A_0有個相似 (Similar) 矩陣爲

$$\begin{aligned} A_1 &= Q_0^{-1} A_0 Q_0 \\ &= R_0 Q_0 \end{aligned} \qquad (23)$$

所以我們得到了一個迭代方式來把矩陣A不斷地做相似轉換

$$\begin{aligned} A_k &= Q_k R_k \\ A_{k+1} &= R_k Q_k \end{aligned}$$
$$k = 0, 1, 2, \cdots \qquad (24)$$

漸漸地A_k會變成很易求固有值的矩陣, 這就是QR方法之精神。

另外, 除了 (17) 式之固有值問題外, 尚有廣義的固有值問題 (Generalized Eigenvalue Problem)

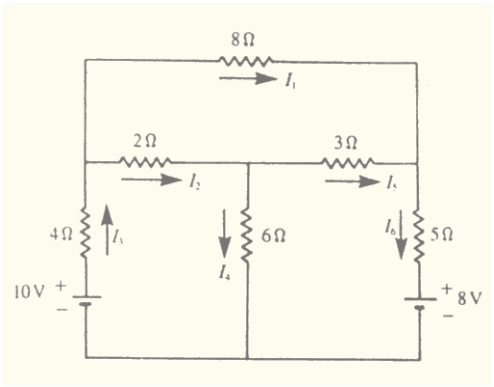
$$Ax = \lambda Bx \qquad (25)$$

奇異值分解 (Singular Value Decomposition) 等之探討, 都屬於這個課題。

2.6 **解線性系統** (Linear System)

我們將這個課題放在本節最後介紹, 主要因爲數值計算函蓋的範圍太廣, 要考慮的細節太多, 無法一一細談, 而我們選這部分在下一節中做較詳細介紹, 主要因爲它的應用實在太多了, 舉例來說:

例9: 圖八是由六個電阻所組成的電路, 請問通過每個電阻之電流爲多少?



圖八 六個電阻組成的電路

解: 由Kirchhoff 定律知道進入某點之電流等於離開該點之電流, 設各電阻之電流分別爲$x_1, x_2, \ldots, x_6$則

$$\begin{aligned} x_1 + x_2 - x_3 &= 0 \\ x_2 - x_4 - x_5 &= 0 \\ x_1 + x_5 - x_6 &= 0 \end{aligned} \tag{26}$$

倘若考慮其他點之電流, 所得到的皆是 (26) 式中方程式的線性組合, 所以只列這三個就足夠了。另外, 一個迴路之電壓和爲零, 故

$$\begin{aligned} &2x_2 + 4x_3 + 6x_4 - 10 = 0 \\ &-6x_4 + 3x_5 + 5x_6 + 8 = 0 \\ &8x_1 - 2x_2 - 3x_5 = 0 \end{aligned} \tag{27}$$

於是 (26) 和 (27) 中共6個線性方程式可用來決定$x_1, x_2, \ldots, x_6$。也就是說, 這就是一個解線性系統

$$\begin{bmatrix} 1 & 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & -1 & 0 \\ 1 & 0 & 0 & 0 & 1 & -1 \\ 0 & 2 & 4 & 6 & 0 & 0 \\ 0 & 0 & 0 & -6 & 3 & 5 \\ 8 & -2 & 0 & 0 & -3 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 10 \\ -8 \\ 0 \end{bmatrix} \tag{28}$$

之問題。

另外, 第 2.1 節中 (2) 之牛頓法若推廣爲解 (4) 式之非線性聯立方程組時, $f'(x_n)^{-1}$就要改爲$J(\mathbf{x}_n)^{-1}$了, 而$J(x_n)$爲x_n點之 Jacobi矩陣。換句話說公式爲

$$\mathbf{x}_{n+1} = \mathbf{x}_n - J(\mathbf{x}_n)^{-1}\mathbf{F}(\mathbf{x}_n), n = 0, 1, 2, \cdots \tag{29}$$

我們可以看出來 (29) 式中每一步要要解一個線性系統

$$J(\mathbf{x}_n)\mathbf{y} = -\mathbf{F}(\mathbf{x}_n) \tag{30}$$

於是

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \mathbf{y} \tag{31}$$

求得新的近似解。其他的數值解法中, 也常需要在每一迭代內解一個線性系統。

還有, 例6中介紹的由16個三次多項式所組成的三次樣條函數, 想要算出它來, 一定要先解一個15 × 15的線性系統求出這個三次樣條函數在當中的15個節點之二次微分值。再看看例7, 如果 (13) 式的等號右邊是$x(t)$和$x'(t)$的線性組合, 化出來的就不是 (16) 式, 而是$x_1, x_2, \ldots, x_9$的線性聯立方程式了。

3. 線性系統的數值解法

爲了讓大家更進一步了解數值計算的堂奧之秘, 以激起學習的興趣, 我們選了線性系統這個重要領域來進一步介紹。

在實際應用的問題中, 我們常會遇到要解線性聯立方程組:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n \end{cases} \tag{32}$$

若用矩陣向量來表示, 即爲解

$$Ax = b \tag{33}$$

其中$A = (a_{ij})$爲$n \times n$矩陣和$b = (b_i)$爲n維向量是已知, 而$x = (x_i)$爲所欲求之n維向量。

要如何解 (33) 式呢? 解題的方法可以分爲兩大類型, 即直接解法 (Direct Methods) 和迭代解法 (Iterative Methods)。在介紹他們之前, 我們先提電腦計算的誤差問題, 這就是計算值與理論值不同的原因之一, 不能不知。

3.1 **捨入誤差** (Round-off Error)

我們都知道實數有無窮多個, 它們是用什麼方式儲存在電腦記憶體中呢? 它們是用固定個數的一串0與1來表示, 稱爲浮點數 (Floating Point Numbers)。由於硬體設計的限制, 電腦只能精確地表示出有限個實數來, 它們稱爲機器數 (Machine Numbers)。而大多數的實數存入電腦都有誤差, 因爲只能用最接近它的機器數 (若是用捨入的設計法) 或是小於它的最大機器數 (若是用截尾的方式設計) 來表示這個實數, 統稱爲捨入誤差。我們用最小的正數ε使得

$$1 + \varepsilon > 1 \tag{34}$$

來觀察捨入誤差, 這個ε稱爲機器精準度 (Machine Precision), 它依電腦的硬體組織不同而異, 表1中我們列下幾個常用電腦之機器精準度。

電腦	位元數	精準度
SUN PC IBM RISC System/6000	32	$2^{-23} \approx 0.12 \times 10^{-6}$
DEC VAX	32	$2^{-24} \approx 0.60 \times 10^{-7}$
CDC Cyber 205	64	$2^{-46} \approx 0.14 \times 10^{-13}$
Cray X-MP	64	$2^{-47} \approx 0.71 \times 10^{-14}$

表一 幾個不同電腦之精準度

從表一中得知, 在計算機上 (34) 式並不恆成立, 誤差也就產生了。除了受硬體設計影響外, 軟體之處理方式也會使這個精準度改觀呢!下面用一個有趣的例子來解釋, 有興趣者, 不妨在您的電腦上試一試。

例10: $(1. - (1./41.)41.) \times 10^{18} = ?$

解: 這是個簡單的式子, 您一定一眼看出它應該是0., 因爲$(1./41.)41. = 1.$。但是, 我們試著在 PC,SUN 工作站, 和 VAX 等電腦上用不同的語言如 FORTRAN,PASCAL, C語言, 和 C++ 來做這個簡單的式子, 由於捨入誤差的不同, 除了0. 外, 還有可能得到 5.4210108×10^{-2},5.9604644×10^{10}, 和5.9604648×10^{10}這些大得驚人的答案呢!

3.2 **直接解法**

直接解法的特性是在有限步驟內得到 (33) 式之數值解, 而且若無捨入誤差的話, 此數值解即 (33) 式之眞解。最常見的即高斯消去法 (Gaussian Elimination), 它的觀念是 (32) 式中

(i) 兩個方程式相加, 解不變。

(ii) 一個方程是乘以c倍,$c \neq 0$, 解不變。

把 (i) 和 (ii) 合併, 即

(iii) 方程式j乘以c倍,$c \neq 0$, 加到方程式i上, 解不變。

這種觀念換成矩陣向量來說, 就是 (33) 式中的擴增矩陣$[A|b]$用基本列運算

(i) 兩列相加;

(ii) 某列乘以c,$c \neq 0$;

(iii) 第j列乘以c,$c \neq 0$, 加到第i列上;

來轉化。我們的目標是將$[A|b]$轉化爲$[U|c]$的形式, 而U是一個上三角矩陣。其主要意義在於 (33) 式之解和下式相同

$$Ux = c \tag{35}$$

而 (34) 式之解十分易求。因爲

$$\begin{aligned} x_n &= c_n/u_{nn} \\ x_i &= (c_i - \sum_{j=i+1}^{n} u_{ij}x_j)/u_{ii}, \qquad (36) \\ & \qquad i = n-1, n-2, \cdots, 1 \end{aligned}$$

先求出x_n, 代入求x_{n-1}, 再將x_n,x_{n-1}代入求x_{n-2}, 依此類推, (36) 這個方法稱爲倒回置換法 (Back Substitution)。至於高斯消去法的演算法 (Algorithm), 我們列在下面以供讀者參考:

```
For k = 1 to n − 1
  For i = k + 1 to n
    α ← a_ik / a_kk
    For j = k + 1 to n
      a_ij ← a_ij − α a_kj        (37)
    end
    b_i ← b_i − α b_k
  end
end
```

上述方法簡易, 但是要深一層探討的問題可不少呢? 列舉一些在下面:

(1) 什麼情況下高斯消去法可以進行無礙呢?

由 (37) 中可以看出, 在中途若某個$a_{kk} = 0$, 這個方法就中斷無法繼續下去了。數學推導可以告訴您,A如果具備什麼樣的條件, 您就可以高枕無憂, 保證不會中斷, 也就是說 (33) 式一定有解。

(2) 高斯消去法中斷, 代表無解嗎?

不一定。下面例子可以舉證有解仍會中斷。

例11: 解

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \tag{38}$$

大家都知道答案是$x_1 = 2$,$x_2 = 1$。但是, 您若想用 (37) 之演算法,$a_{11} = 0$, 第一步就做不了。

(3) 高斯消去法做得下去, 答案一定精確嗎?

不一定。要解釋這個觀念, 就牽涉到前面所提過的捨入誤差了。通常數值計算的第一章都要介紹這個捨入誤差的概念, 讓學習者了解理論數學與計算數學之間是有差距的。

從例10中不難想像更複雜的數值方法會使捨入誤差累積。若是起始誤差很小, 所得答案卻差得十分遠, 這種方法就稱爲不穩定的方法 (Unstable Algorithm), 要避免採用。下面我們用一部精準度爲0.5×10^{-2}之假想電腦來計算, 這種機器上的計算又稱 2位浮點運算 (2-digit Floating Point Arithmetic), 好讓各位以小窺大, 了解到我們爲什麼說高斯消去法成功完成但答案不見得正確。

例12: 在2位浮點運算的電腦上用高斯消去法解下列線性系統

$$\begin{cases} 0.001x_1 + x_2 = 1 \\ x_1 + x_2 = 2 \end{cases} \tag{39}$$

解: 用R_1,R_2分別表示第一列及第二列, 則

$$\begin{bmatrix} 0.001 & 1 & 1 \\ 1 & 1 & 2 \end{bmatrix} \xrightarrow{R_2-R_1\times1000} \begin{bmatrix} 0.0010 & 1.0 & 1.0 \\ 0 & -1000 & -1000 \end{bmatrix}$$

故

$$\begin{aligned} -1000x_2 &= -1000 \\ x_2 &= 1.0 \end{aligned} \tag{40}$$

代入

$$0.0010x_1 + x_2 = 1.0 \text{ 得 } x_1 = 0.0 \tag{41}$$

但若用精確算術來算則爲

$$\begin{bmatrix} 0.001 & 1 & 1 \\ 1 & 1 & 2 \end{bmatrix} \xrightarrow{R_2-R_1\times1000} \begin{bmatrix} 0.0010 & 1 & 1 \\ 0 & -999 & -998 \end{bmatrix}$$

$$\begin{aligned} x_2 &= 998/999 \approx 0.999 \\ x_1 &= (1 - 998/999)/0.001 \approx 1.001 \end{aligned}$$

可見 (40) 中x_2尚十分精確, 但 (41) 式$x_1 = 0.0$則有100%的誤差了!

由於第 (2) 項和第 (3) 項兩個不肯定的答案, 產生了許多改進的技巧, 如: 部分軸元高斯消去法 (Gaussian Elimination with Partial Pivoting), 全軸元高斯消去法 (Gaussian Elimination with Complete Pivoting), 比例列軸元高斯消去法 (Gaussian Elimination with Scaled Row Pivoting)。其他尚有列均衡 (Row Equilibration), 行均衡 (Column Equilibration) 及迭代改進 (Iterative Refinement) 之方法等, 都可以提高解的精確度。

(4) 誤差分析(Error Analysis)

能否由演算法推論出捨入誤差之上限是多少?(33) 式中A有誤差, 對所求解之影響如何 ?b有誤差對所求解之影響如何?A和b都有誤差呢? 深入討論的話, 也是不少學問呢 !

(5) 寫程式時要如何設計才節省佔用電腦記憶體之空間?

觀察 (36) 和 (37) 式可以發現$[v|c]$如果壓在$[A|b]$上最節省空間了, 而且 (37) 不受影響。

(6) 如何讓程式更有效率?

這還要看您是使用那一種電腦語言來寫這個方法。若是用 PASCAL 或 C 語言, 它們存放矩陣時是列存法 (Rowwise), 即先存第一列再存第二列$\cdots$, 則 (22),(23) 十分好用。若是用 FORTRAN來寫, 它是行存法 (Columnwise) 的語言, 也就是說存放A時先存第一行, 再存第二行,$\cdots$。一個思維細密的程式設計師, 會體會到如果A的維數十分大時, 用 FORTRAN寫 (37) 效率就不好了。整個演算法要重寫, 針對行向量的運算來改寫, 就可以大大提昇效率。近年來, 向量電腦 (Vector Computer) 與平行電腦 (Parallel Computer) 日漸普及, 若要用它們來做計算, 演算法與程式設計都要跟著改寫。

(7) 如(33)式的線性系統, 假設A不變, 而有許許多多個不同的b, b_i,$i = 1, 2, \cdots, m$, 想求這m個系統的解怎麼辦?

談到效率問題, 就要做運算計數 (Operation Count) 來了解一個方法所需的時間。高斯消去法 (23) 式要$O(\frac{n^3}{3})$個運算, 而 (22) 式則只要$O(\frac{n^2}{2})$個運算。如果有m個系統要解, 而A又全部一樣, 我們從頭算m次可就大大浪費時間了。我們可以證得高斯消去法實際上是將A做分解, 即

$$A = LU \tag{42}$$

而L爲單位下三角矩陣, 它就是 (37) 中α所形成的矩陣, 只要將 (37) 中每一步驟之α存在a_{ik}位置, 既不影響後面之運算又記錄下 (42) 式中的L。有了這個認知後, 若要解10個A相同之線性系統, 只要先做高斯消去法, 留下那些α得 (42) 式。那麼, 我們就改爲考慮解

$$LU\, x_i = b_i, \quad i = 1, 2, \cdots, 10 \tag{43}$$

然後再對$i = 1, 2, \cdots, 10$做

$$\begin{aligned} &\text{(i) 解 } L\, z_i = b_i \\ &\text{(ii) 解 } U\, x_i = z_i \end{aligned} \tag{44}$$

就可以快速解得這10個系統了。

(8) A做LU分解的觀念可以推廣嗎?

是的, 只要將A表示爲 (42) 式, 則 (33) 式之求解就可以化爲如 (44) 式的兩個步驟, 其中 (ii) 爲倒回置換法,(i) 則稱爲向前置換法 (Forward Substitution), 都只要$O(\frac{n^2}{2})$個運算即可。其它尚有 Dolittle 分解法,Crout 分解法,Cholesky 分解法$\cdots$等, 它們都是用直接分解的觀念而不是用消去的觀念所推導出來的方法。

(9) A如果不是方陣呢?

假設A爲$m \times n$矩陣, 則表示有m個條件數n個未知數。如果$m > n$, 表示條件數多於未知數之系統, 這種情形大多數無解。在應用中, 往往也只是想求一個最佳解, 即找 x使得

$$\min_x \|b - Ax\|_2 \tag{45}$$

成立, 也就是找出使餘向量$b - Ax$爲最短之解。同樣地, 當$m < n$時, 通常有無窮多解滿

足 (33) 式, 我們要從中選一個最佳解, 即令

$$\min_{\substack{x \\ Ax=b}} \|x\|_2 \qquad (46)$$

成立, 也就是在所有解中找出長度最短的那一個。上面 (45) 和 (46) 式都稱爲最小平方解。

3.3 迭代解法

在許多科學應用中, 如石油工程、核子反應、氣象預測、太空科技等等, 常常需要解偏微分方程式。這些偏微分方程式離散化後, 就是一個解線性系統 (33) 的問題, 而且A的維數十分大, 譬如美國西屋電力公司的 Bettis Atomic Power Laboratory 早在二十多年前就每天要解二維的 Laplace 問題, 化成的矩陣維數超過 20000, 有時尙要解三維的問題, 而矩陣維數大到108000呢!別說二十多年前的電腦, 現在的電腦也無法把這麼大的矩陣整個放入記憶體內。要怎麼解決呢? 這就是迭代法佔優勢的地方, 由於這一類的問題,A 不但大而且大部分元素爲零, 稱爲稀疏矩陣 (Sparse Matrix)。一個1000×1000 的矩陣, 裡面非零元素不到5000個, 才0.5%呢! 迭代法通常採用特殊儲存格式, 只存非零元素, 一則大大地節省空間, 二則大大地節省運算時間, 因爲所有乘以0的不必要運算全部可以免了。

下面我們舉一個例子來說明矩陣的形態, 以幫助了解。

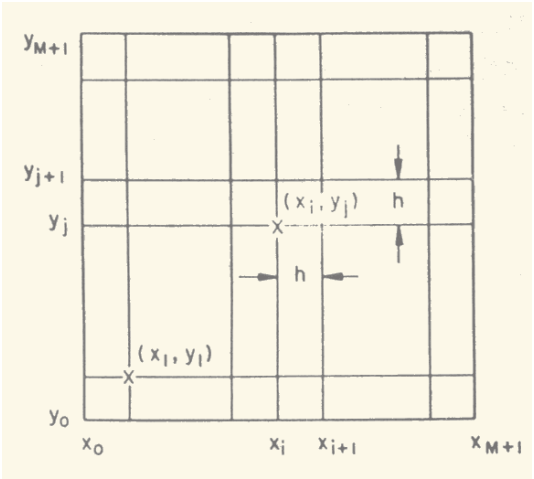
例13: 解下列 Poisson 方程式於單位正方$R=[0,1]\times[0,1]$內

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}=G(x,y),\ 0<x,y<1$$
$$u(x,y)=g(x,y),\ (x,y)\in\partial R \qquad (47)$$

解: 偏微分方程中是找出$u(x,y)$函數來, 但數值計算則是找R內某些點之值。譬如取網格大小爲h, 則單位正方R被分成圖九的樣子, 每個網格點(x_i,y_j) 即爲所欲求值之未知點。若用中央差分法 (6) 式來取代$\frac{\partial^2 u}{\partial x^2}$和$\frac{\partial^2 u}{\partial y^2}$, 則 (47) 式改寫爲

$$4u_{i,j}-u_{i+1,j}-u_{i-1,j}-u_{i,j+1}-u_{i,j-1}$$
$$=-h^2G(x_i,y_j) \qquad (48)$$

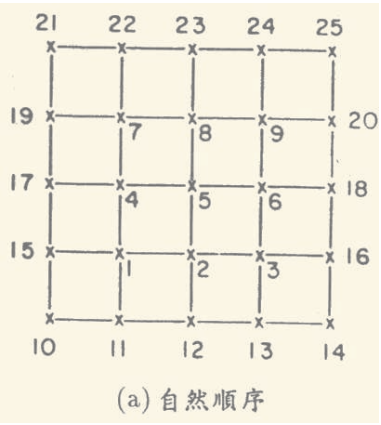
其中$u_{i,j} = u(x_i,y_j)$。接下來, 我們要將u_{ij}定個順序, 寫成一個一維向量 u來,(48) 式才可改寫爲 (33) 式的形式。

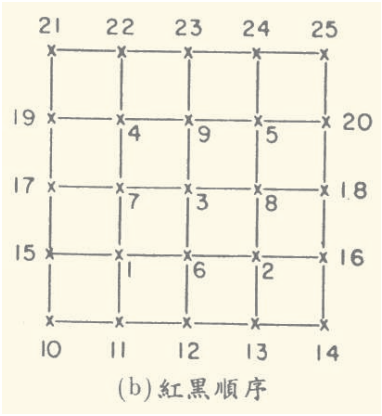


圖九 用網格來分劃出R內未知點

定順序的方式許多, 常見的有自然順序 (Natural Ordering) 和紅黑順序

(Red/Black Ordering), 如圖十所示。用不同的順序來定未知點, 所畫出來的A型態亦不同, 圖十是以$h = 1/4$爲例來分網格;





圖十

自然順序所得的線性系統爲

$$\begin{bmatrix} 4 & -1 & 0 & -1 & & & & & \\ -1 & 4 & -1 & 0 & -1 & & & \mathrm{O} & \\ 0 & -1 & 4 & 0 & 0 & -1 & & & \\ -1 & 0 & 0 & 4 & -1 & 0 & -1 & & \\ & -1 & 0 & -1 & 4 & -1 & 0 & -1 & \\ & & -1 & 0 & -1 & 4 & 0 & 0 & -1 \\ & & & -1 & 0 & 0 & 4 & -1 & 0 \\ & \mathrm{O} & & & -1 & 0 & -1 & 4 & -1 \\ & & & & & -1 & 0 & -1 & 4 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \\ u_9 \end{bmatrix} = \begin{bmatrix} g_{11} + g_{15} - h^2 G_1 \\ g_{12} \qquad - h^2 G_2 \\ g_{13} + g_{16} - h^2 G_3 \\ g_{17} - h^2 G_4 \\ -h^2 G_5 \\ g_{18} - h^2 G_6 \\ g_{19} + g_{22} - h^2 G_7 \\ g_{23} - h^2 G_8 \\ g_{20} + g_{24} - h^2 G_9 \end{bmatrix} \tag{49}$$

而紅黑順序則得

$$\begin{bmatrix} 4 & & & & & -1 & -1 & 0 & 0 \\ & 4 & & \mathrm{O} & & -1 & 0 & -1 & 0 \\ & & 4 & & & -1 & -1 & -1 & -1 \\ & \mathrm{O} & & 4 & & 0 & -1 & 0 & -1 \\ & & & & 4 & 0 & 0 & -1 & -1 \\ -1 & -1 & -1 & 0 & 0 & 4 & & \mathrm{O} & \\ -1 & 0 & -1 & -1 & 0 & & 4 & & \\ 0 & -1 & -1 & 0 & -1 & & & 4 & \\ 0 & 0 & -1 & -1 & -1 & & \mathrm{O} & & 4 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \\ u_9 \end{bmatrix} = \begin{bmatrix} g_{11}+g_{15}-h^2G_1 \\ g_{13}+g_{16}-h^2G_2 \\ -h^2G_3 \\ g_{19}+g_{22}-h^2G_4 \\ g_{20}+g_{24}-h^2G_5 \\ g_{12}-h^2G_6 \\ g_{17}-h^2G_7 \\ g_{18}-h^2G_8 \\ g_{23}-h^2G_9 \end{bmatrix} \tag{50}$$

從例13 中我們知道如果$h = 1/101$,$(x_i,\ y_j)$就一共有10000點了, 這個矩陣就是10000 ×10000呢!所以不難得到一個大矩陣A。再由 (48) 式觀察, 就知道A的每一列最多5個非零元素, 型態仍如 (49) 和 (50) 式所表。

迭代法解 (33) 式的主要觀念是它產生一系列的向量$x^{(0)}$,$x^{(1)}$,$x^{(2)}$, $\cdots$, 它們是眞解x的近似解, 而且越來越逼近x, 即

$$\lim_{n\to\infty} x^{(n)} = x \tag{51}$$

它建構的方法是選一個矩陣Q, 稱爲分裂矩陣 (splitting matrix), 得$A = Q - (Q - A)$, 於是 (33) 式可改寫爲

$$Qx = (Q - A)x + b \tag{52}$$

於是產了一個迭代法

$$Qx^{(n+1)} = (Q-A)x^{(n)}+b,\ n = 0, 1, 2, \cdots \tag{53}$$

它相當於

$$x^{(n+1)} = (I - Q^{-1}A)x^{(n)} + Q^{-1}b,$$

$$n = 0, 1, 2, \cdots \tag{54}$$

這就是基本迭代法的形式。深一層再想, 一連串的問題就來了:

(1) 要如何選分裂矩陣Q呢?

從 (53) 和 (54) 式知道Q一定要是一個非奇異方陣,Q^{-1}才存在。還有(54) 式雖爲迭代法$x^{(n+1)}$向量之形式, 但實際計算時$x^{(n+1)}$要由 (53) 這個線性系統解得。換句話說, 每一次迭代都包含解一個線性系統 (53), 即解

$$Qx = y \tag{55}$$

故Q一定要選得讓 (55) 式十分易解。例如,$Q = I$即爲 RF 方法,$Q = D = \mathrm{diag}(A)$即爲 Jacobi 法。我們若將A表示爲

$$A = D - C_L - C_U \tag{56}$$

其中C_L爲嚴格下三角方陣, 而C_U嚴格上三角方陣。Gauss-seided 法中 $Q = D - C_L$,SOR法則選$Q = w^{-1}D - C_L$,w爲一個鬆弛參數 (Relaxation Parameter)。常用的尙有$Q = LU$而$LU \sim A$, 即LU爲A的一個

近似分解, 所以它比直接解法中的 (42) 式簡易。直接解法中若A爲稀疏矩陣,L和U 卻不盡然, 愈分解愈繁, 反而不利運算, 所以這裡我們要求L,U亦爲稀疏型態, 只要近似分解就好了, 這種方法稱爲不完全LU分解法 (Incomplete LU Factorization)。其他還有許多不同方法。

(2) 向量序列$(x^{(n)})$會收斂嗎?A和Q滿足什麼條件才收斂呢? 是不是對任意$x^{(0)}$皆收斂? 皆收斂到同一向量$\overline{x}$? 此$\overline{x}$ 是否即 (33) 式之眞解x?

(3) 反覆迭代要如何停? 用什麼當停止判斷? 這個判斷方式可保證解的精確度嗎?

(4) A有誤差或/和b有誤差, 對解的影響如何? 這和A的性質十分有關? 假如一點點誤差就使解差得十分大, 我們稱這個問題爲病態問題 (Ill-Conditioned Problem)。這種問題一定要預先條件化(Preconditioned)後才可再 求解。

(5) 如何用理論推演來比較不同選代法的收斂速率? 如何來加速它的收斂呢? 最有名的兩種加速方法爲 Chebyshev 加速法和共軛梯度加速法(Conjugate Gradient Acceleration)。

其它尙有許許多多要探討的課題, 我們不再深入, 只讓大家對迭代法有個概念。

4. 結語

一下子無法將數值計算的全貌展現, 本文想強調的是有許多問題, 分析的方式無法求解, 但可以靠數值方法來得近似解。不過, 使用數值方法並不能拿來就用, 要了解到問題本身的性質、數值方法的優劣、及電腦的精確度等等, 都會影響答案的準確性。

最後, 再舉一個有趣的數據來說明理論數學與實際應用之差距, 突顯數值計算之必要性。念過矩陣向量的朋友可能想, 爲什麼不用 Cramer's 法則直接求解 呢? 也就是說 (33) 式之解爲

$$x_i = \frac{\det A_i}{\det A},\ i = 1, 2, \cdots, n \qquad (57)$$

其中$\det A$表示A之行列式值, 而A_i表示將矩陣A之第i行改以 $(b_1, b_2, \ldots, b_n)^T$來代替而得的矩陣。

由 (57) 式知 Cramer's 法則解線性系統, 一共要求$n+1$個行列式值, 假設不考慮特殊處理技巧它大約需要 $(n^2-1)n!$個運算。這個方法效率好嗎? 可行性高嗎? 我們以$n = 25$爲例, 一共求 26 個行列式值, 共用$(624)25!$個運算, 即9.68×10^{27}個運算。也許您還體會不出這個數目有多大, 我們假設用一台每秒做一千萬個運算的電腦來執行, 這就要 9.68×10^{20}秒, 而一天有86400秒, 所以共要1.12×10^{16}天, 即這台電腦不停地跑, 要跑三千億個世紀才能算完!但是若用比例列軸元高斯消去法來解, 前面提過它的運算個數爲$O(\frac{n^3}{3})$, 即不到一秒鐘可解出答案了。

參考資料

1. Buchanan, J. L., and P. R. Turner, Numerical Methods and Analysis, McGraw-Hill, New York, 1992.

2. Burden, R. L., and J. D. Faires, Numerical Analysis, 4th ed., PWS-Kent, Boston, 1989.
3. Cheney, W., and D. Kincaid, Numerical Mathematics and Computing , 2nd ed., Brooks/Cole, Pacific Grove, Calif., 1985.
4. Hageman, L. A., and D. M. Young, Applied Iterative Methods, Academic Press, New York, 1981.
5. Hager, W. W., Applied Numerical Linear Algebra, Prentice-Hall, Englewood Cliffs, New Jersey, 1988.
6. Kincaid, D., and W. Cheney, Numerical Analysis, Brooks / Cole, Pacific Grove, Calif., 1991.

—本文作者任教於輔仁大學數學系—