Visual Basic

CHAPTER 5

非線性方程式

(Nonlinear Equations)

張 榮 興 博士

豐映科技股份有限公司

E-mail: chang.ronhsin@msa.hinet.net

http://www.resi.com.tw/vb.htm

工程師所面對的真實世界大部分都是非線性問題

計題 D-V 題

流體輸送管路計算

A 及 B 爲兩個石油產品儲存槽，各利用直徑 30 公分的管路輸送至距離 1.5 公里處的配料管制站 D。油料再由 D 點利用內徑 50 公分的管線輸送至距離 750 公尺的第三個儲存槽 C，作爲某一化工程序的進料。系統簡圖如圖 5.1。儲存槽 A 的液位較 C 高 10 公尺，B 的液位較 A 高 6 公尺。假設所有管線均爲鋼管，且其相對粗操度ε / D = 0.0001。液體密度爲 870 kg / m^3，黏度爲 0.7 m N s / m^2。

描述此系統之質量平衡及能量平衡方程式爲[1]：

$$u_1 + u_2 = 2.78\,u_3$$
$$58.9 + 40\left(u_1^2 - u_2^2\right) = 0$$
$$-1.56.0 + 40\left(u_2^2 + 0.3\,u_3^2\right) = 0$$

試求 A 至 D，B 至 D 及 D 至 C 的流速 u_1、u_2 及 u_3。

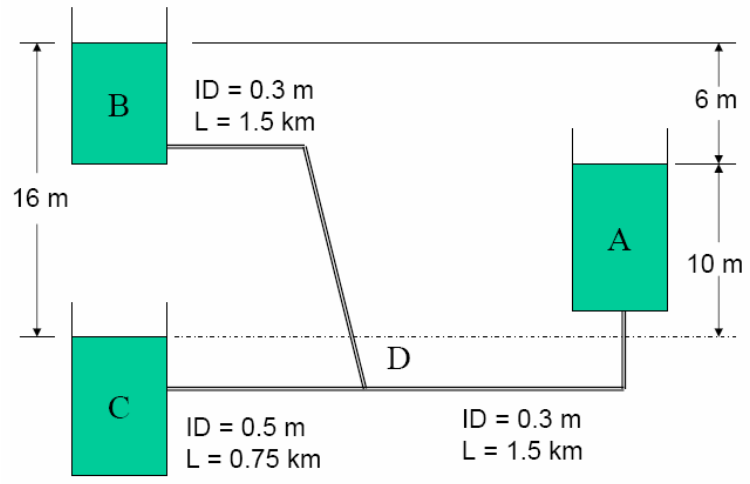


圖 5.1　設計問題 D – V 的儲存槽配置圖

工程師所面對的真實世界，大部分都是非線性問題。本章所要討論的是非線性方程式或非線性聯立方程組的求解方法。非線性方程式可簡單寫成 $\underline{f}(\underline{x})=0$，其中 $\underline{x}=[x_1, x_2, \ldots\ldots, x_n]^T$ ；$\underline{f}=[f_1, f_2, \ldots\ldots, f_n]^T$。當 $n \neq 1$ 時，即稱爲聯立方程組。在工程及科學應用上，非線性聯立方程組的求解，是工程師時常會遇見的問題。例如設計問題 D-V 所表示的流體流速計算，微分方程式所得超越函數 (Transcendental Equation) 的求解，非理想氣體的 P-v-T 關係計算等等，均爲常見的計算問題。

本章將依次探討多項式方程式、一般非線性方程式及非線性聯立方程組的求解方法。

第一節　葛瑞菲根平方法

Visual Basic

在本章所介紹的各種求解方程式的方法中，葛瑞菲根平方法 (Graeffe Root Square Method) 是最能完整地求出一多項式所有根的方法，但其缺點是所求得的根只爲近似解。因此，通常是利用葛瑞菲根平方法，先找出方程式所有根的大略位置，作爲稍後所介紹的他種求根方法的敲門磚，再利用其他方法求出更正確的根。

假設 n 次多項式 $f(x)=0$ 之根 $\alpha_1, \alpha_2, \cdots\cdots, \alpha_n$ 均爲實數，且 $|\alpha_1|>|\alpha_2|>|\alpha_3|>\cdots\cdots>|\alpha_n|$。則多項式函數 $f(x)$可寫成

$$f(x)=a_0x^n+a_1x^{n-1}+\cdots\cdots+a_{n-1}x+a_n=0 \qquad \textbf{(5-1.1)}$$

$$f(x)=\sum_{i=0}^{n}A_ix^{n-i}; \qquad A_i=a_i/a_0 \qquad \textbf{(5-1.2)}$$

或

$$f(x)=(x-\alpha_1)(x-\alpha_2)\cdots\cdots(x-\alpha_n)=\prod_{i=1}^{n}(x-\alpha_i) \qquad \textbf{(5-1.3)}$$

定義輔助函數 $\phi(\underline{x})$ 爲 $f(x)$ 及 $f(-x)$ 之乘積：

$$\phi(x)=(-1)^n f(x)f(-x)=\left[\prod_{i=1}^{n}(x-\alpha_i)\right]\left[\prod_{i=1}^{n}(x+\alpha_i)\right]=\prod_{i=1}^{n}(x^2-\alpha_i^2) \qquad \textbf{(5-1.4)}$$

由於 $\phi(x)$ 爲偶函數多項式，故可定義一新多項式 $f_2(x)$ 爲

$$f_2(x)=\phi(\sqrt{x})=\prod_{i=1}^{n}(x-\alpha_i^2) \qquad \textbf{(5-1.5)}$$

其根均爲原多項式 $f(x)=0$ 之平方。重複此步驟，則我們可得到一序列多項式函數 f_2，f_4, f_8, ……，

$$f_m(x)=\prod_{i=1}^{n}(x-\alpha_i^m) \qquad m=2,4,8,16,\cdots\cdots \qquad \textbf{(5-1.6)}$$

其中 m 爲 2 的整數次方，而且 $f_m(x)$ 之根爲 $\alpha_1^m,\alpha_2^m,\cdots\cdots,\alpha_n^m$。

又由於 $|\alpha_1|>|\alpha_2|>\cdots\cdots|\alpha_n|$，因此，當 m 值足夠大以後，可以使得

$$1>>\left|(\alpha_2/\alpha_1)^m\right|>>\left|(\alpha_3/\alpha_2)^m\right|>>\cdots\cdots>>\left|(\alpha_n/\alpha_{n-1})^m\right| \qquad \textbf{(5-1.7)}$$

將方程式 (5-1.6) 展開，整理後得到

$$\begin{aligned} f_m(x)=x^n+\left(-\sum_{i=1}^{n}\alpha_i^m\right)x^{n-1}+\left(\sum_{i=1}^{n-1}\sum_{j=i}^{n}\alpha_i^m\alpha_j^m\right)x^{n-2}+ \\ +\left(-\sum_{i=1}^{n-2}\sum_{j=i}^{n-1}\sum_{k=j}^{n}\alpha_i^m\alpha_j^m\alpha_k^m\right)x^{n-3}+\cdots \\ +(-1)^m\left(\alpha_1^m\alpha_2^m\cdots\cdots\alpha_n^m\right) \end{aligned} \qquad \textbf{(5-1.8)}$$

將方程式 (5-1.7) 代入上式，簡化後可以得到

$$f_m(x)=x^n-A_1x^{n-1}+A_2x^{n-2}+\cdots\cdots+(-1)^nA_n \qquad \textbf{(5-1.9)}$$

其中

$$A_1\cong\alpha_1^m \quad , \quad A_2\cong\alpha_1^m\,\alpha_2^m \quad , \quad A_j\cong\prod_{k=1}^{j}\alpha_k^m\,; \quad j=1,2,\cdots\cdots,n$$

因此，可得 $f_m(x)$ 之近似解爲

$$\alpha_1^m\cong A_1 \quad , \quad \alpha_2^m\cong A_2/A_1 \quad , \quad \alpha_j^m\cong A_j/A_{j-1}\,; \qquad j=1,2,\cdots\cdots,n \qquad \textbf{(5-1.10)}$$

由所得的第 m 組根，我們即可求得原方程式 $f(x)=0$ 之近似解。但由於這種方法在運作中將方程式的根作平方處理，因此，應特別注意，所得到的近似解的正負符號，仍需代入原方程式檢查才可確定。

例題 5-1　矩陣的特徵值

多項式方程式

$$f(x) = x^4 - 35x^3 + 146x^2 - 100x + 1 \qquad \textbf{(5-1.11)}$$

爲對稱矩陣 $\underline{A}$的特徵方程式，試求 $\underline{A}$ 的特徵值 (Eigenvalue)。

$$\underline{A} = \begin{bmatrix} 10 & 9 & 7 & 5 \\ 9 & 10 & 8 & 6 \\ 7 & 8 & 10 & 7 \\ 5 & 6 & 7 & 5 \end{bmatrix}$$

解：

TOP-DOWN 設計

由方程式 (5-1.4) 得 $\phi(x)$ 之定義爲

$$\phi(x) = (-1)^n \left[\sum_{i=0}^{n} A_i x^{n-1} \right] \left[\sum_{i=0}^{n} A_i (-x)^{n-1} \right] \qquad \textbf{(5-1.12)}$$

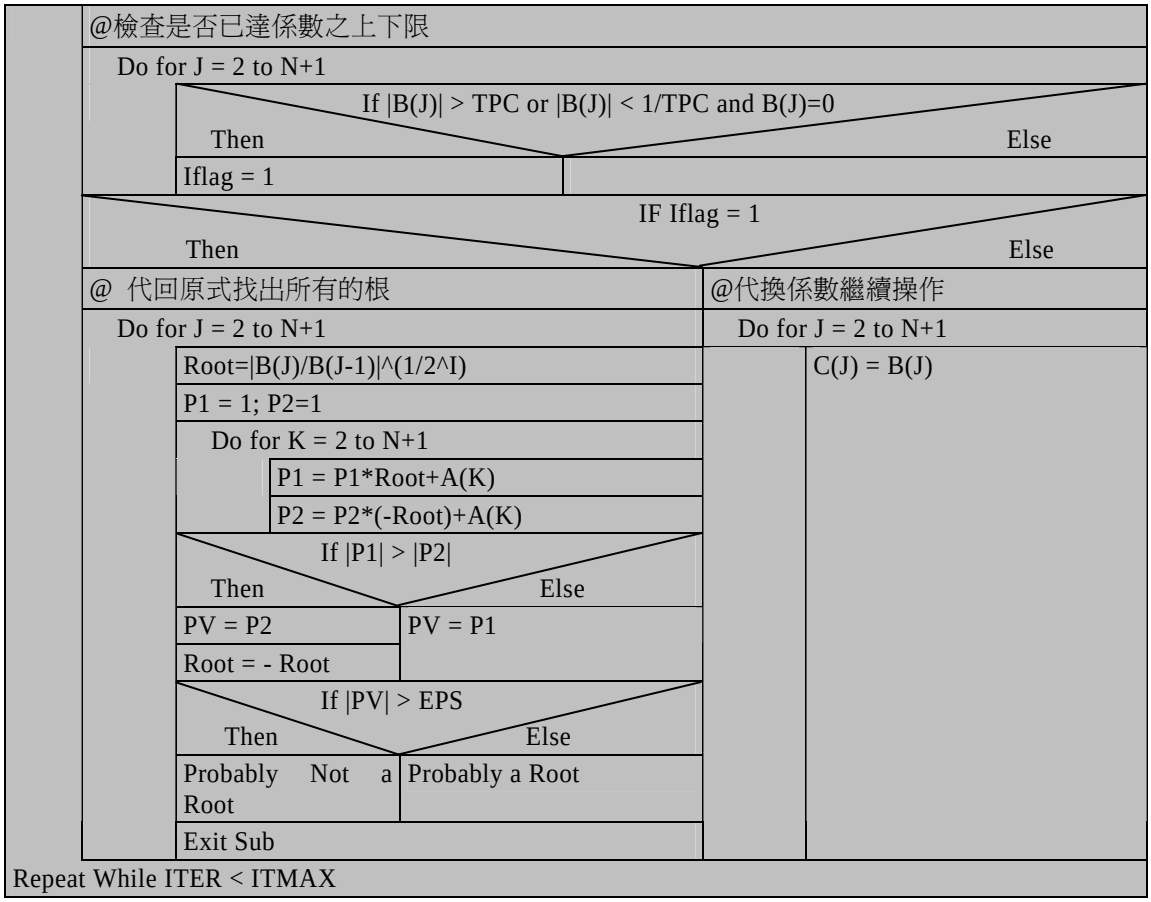
則經過一次這種操作，所得到的新係數由二項式定理得知爲

$$B_j = (-1)^j \left[A_j^{\,2} + 2\sum_{k=1}^{j} (-1)^k A_{j+k} A_{j-k} \right] \quad , \quad j = 1, 2, \cdots\cdots, n \qquad \textbf{(5-1.13)}$$

$$B_0 = 1$$

Sub Iteration(N1, ITMAX, TPC, EPS, PT$, A, B, C)

```
@根平方操作，方程式(5-1.13)
    Do for J = 2 to (N+1)
        B(J) = C(J) * C(J)
        Do for K = 1 to (J-1)
            JPK = J + K
            JMK = J – K
            If (JPK <= (N+1))
            Then                                  Else
            B(J)=B(J)+(-1)^k*2*C(JPK)*C(JMK)
        B(J) = (-1)^(J-1)*B(J)
```



符號說明：

A： 多項式係數，A_i

B： 平方處理後的多項式係數

C： 上一次處理所得多項式係數

EPS： ε，用於判斷所得根是否爲原方程式之根，建議採用 0.1 至 0.001 間。

ITMAX：最大根平方處理次數

N： 多項式最高冪次，n

P1： $f(\alpha_i)$；α 表求出的解。

P2： $f(-\alpha_i)$

PV： $f(\alpha_i)$

ROOT： $f(x) = 0$ 可能的根

TPC： 根平方操作所得係數 B_i 之極限值，建議採用 1.E+32

程式列印：

```
Sub Graeffe(xpos, ypos)
'       ************************************************
'       GRAEFFE'S ROOT SQRUARE METHOD
'       ************************************************
'
'     A: Vector of polynomial coefficients
'     B: Vector of coefficient of the squared polynomial after current iteration
'     C: Vector of coefficient of the squared polynomial prior to current iteration
'
Dim A(100), B(100), C(100)
TPC = 1E+32
EPS = 0.001
ITMAX = 50
PT$ = "N"
Do
      B(1) = 1
      C(1) = 1
      Cls

      Print "Degree of the Polynomial N = ";
      N = Val(InputBox("Degree of the polynomial = ", "", N, xpos, ypos))
Print N

      Print "Upper Limit on the Magnitude of Coefficients"
      Print "Produced by the Root Squaring Process = ";
      TPC = Val(InputBox("Upper limit on the magnitudes of coefficients_
            produced by the root squaring process = ", "", TPC, xpos, ypos))
      Print TPC

      Print "Small Positive Number Used to Test the Existence of a Root = ";
      EPS = Val(InputBox("Small positive number used to test the_
            existence of a root = ", "Small Number", EPS, xpos, ypos))
      Print EPS

      Print "Maximum Number of Iteration Allowed = ";
      ITMAX = Val(InputBox("Maximum number of iterations allowed = ",_
                  "Max. # Iteration", ITMAX, xpos, ypos))
      Print ITMAX

      Print "Print Coefficients After Each Iteration <Y/N> ? ";
      PT$ = InputBox("Print coefficients after each iteration <Y/N>", "", PT$)
      Call ReadCoeff(N, A, xpos, ypos)
      N1 = N + 1
      Call NormCoeff(N1, A, C)
      Call ECHO(N, A)
      Call Iteration(N1, ITMAX, TPC, EPS, PT$, A, B, C)
```

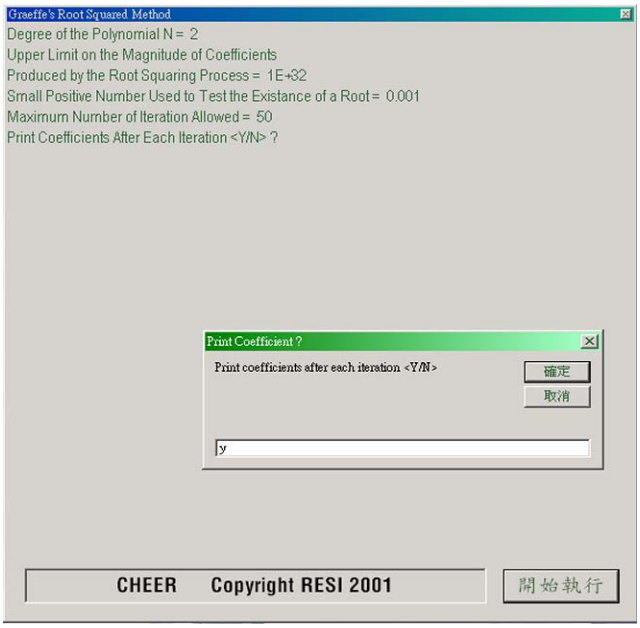
```
        YN$ = InputBox("RUN NEXT JOB <Y/N> ", "", "N", xpos, ypos)
Loop While YN$ <> "N" And YN$ <> "n"
End Sub


Sub ReadCoeff(N, A, xpos, ypos)
'       ---------------------
'           READ COEFFICIENTS
'       ---------------------
Cls
N1 = N + 1
For I = 1 To N1
        Print "Coefficient #"; I; " = ";
        A(I) = Val(InputBox("Enter Coefficient A(I) = ", "", A(I), xpos, ypos))
        Print A(I)
Next I
End Sub


Sub NormCoeff(N1, A, C)
'       -------------------------
'           NORMALIZE COEFFICIENTS
'       -------------------------
For I = 2 To N1
        A(I) = A(I) / A(1)
        C(I) = A(I)
Next I
A(1) = 1
End Sub


Sub ECHO(N, A)
'       --------
'           ECHO
'       --------
Cls
Print "N = "; N
N1 = N + 1
Print "Coefficients:"
For I = 1 To N1
        Print Format(A(I), "    #.####E+");
        If (Int(I / 5) - I / 5) = 0 Then Print
Next I
Print
End Sub


Sub Iteration(N1, ITMAX, TPC, EPS, PT$, A, B, C)
'
'     ------------------
```

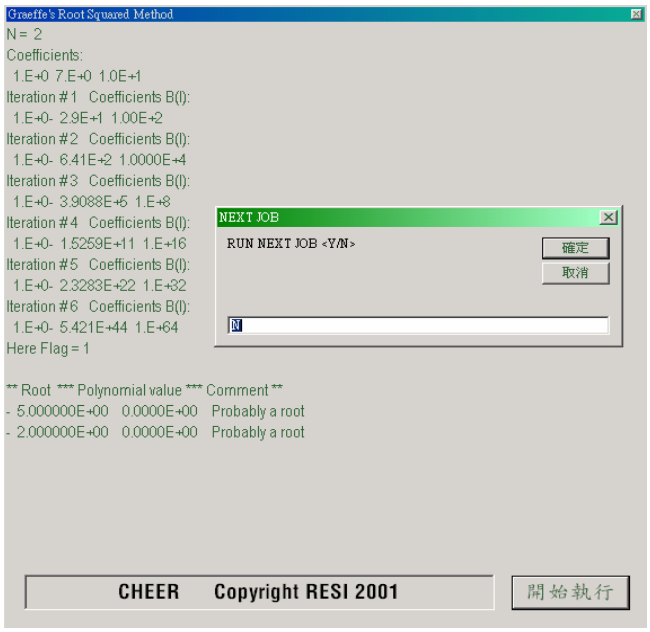
```
'         START ITERATION
'     ------------------
'
Flag% = 0
For I = 1 To ITMAX
        For J = 2 To N1
                B(J) = C(J) * C(J)
                JM1 = J - 1
                For K = 1 To JM1
                        JPK = J + K
                        JMK = J - K
                        If (JPK > N1) Then
                                Exit For
                        Else
                                B(J) = B(J) + (-1) ^ K * 2! * C(JPK) * C(JMK)
                        End If
                Next K
                B(J) = (-1) ^ JM1 * B(J)
        Next J

        If PT$ <> "N" And PT$ <> "n" Then
                Print "Iteration #"; I;
                Print "    Coefficients B(I):"
                For J = 1 To N1
                        Print Format(B(J), "    #.####E+");
                        If (Int(J / 5) - J / 5) = 0 Then Print
                Next J
                Print
        End If
'     ---------------
'         CHECK LIMIT
'     ---------------
        For J = 2 To N1
                If ((Abs(B(J)) > TPC) Or (Abs(B(J)) < 1 / TPC) And (B(J) = 0)) Then
                        Flag% = 1
                        Print "Here Flag = 1"
                        Iter = I
                        Exit For
                End If
        Next J

        If Flag% = 1 Then
                Exit For
        Else
'       SHIFT B TO C
                For J = 2 To N1
                        C(J) = B(J)
                Next J
        End If
Next I
```

```
If Flag% <> 1 Then
        Print "Iteration number exceeds ITMAX     ---- CALCULATION CONTINUES"
        I = ITMAX
End If

Call PossibleRoots(N1, A, B, Iter, EPS)
End Sub


Sub PossibleRoots(N1, A, B, Iter, EPS)
'      -------------------------
'       COMPUTE POSSIBLE ROOTS
'      -------------------------
'
Print
Print "** Root     *** Polynomial value *** Comment **"
For J = 2 To N1
        ROOT = Abs(B(J) / B(J - 1)) ^ (1 / 2 ^ Iter)
'
'    EVALUATE POLYNOMIAL VALUE AT ROOT AND -ROOT
'
P1 = 1
        P2 = 1
        For K = 2 To N1
                P1 = P1 * ROOT + A(K)
                P2 = P2 * (-ROOT) + A(K)
        Next K
'
'    CHOOSE LIKELY ROOTS
'
        If (Abs(P1) > Abs(P2)) Then
                PValue = P2
                ROOT = -ROOT
        Else
                PValue = P1
        End If
        Print Format(ROOT, "    0.000000E+00    ");
        Print Format(PValue, "    0.0000E+00    ");
        Print "    ";
        If (Abs(PValue) > EPS) Then
                Print "Probably NOT a root"
        Else
                Print "Probably a root"
        End If
Next J
End Sub
```

測試數據及結果討論：

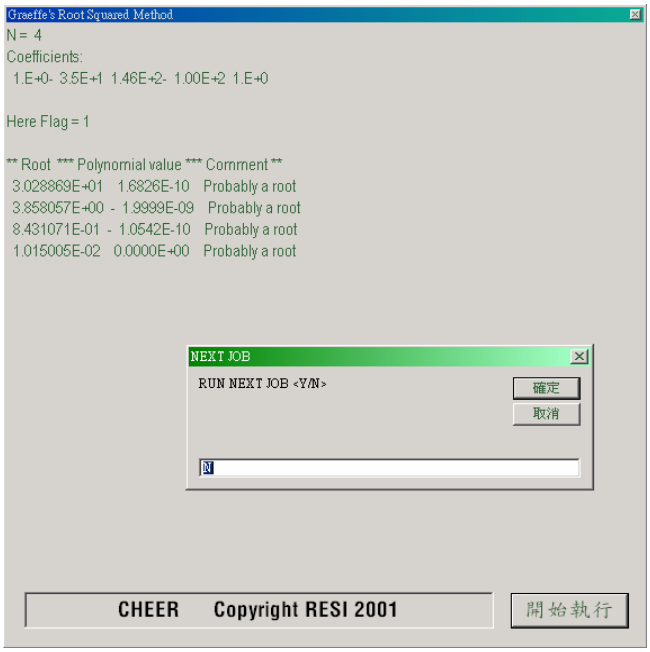
1. $f(x) = x^2 + 7x + 10 = 0$





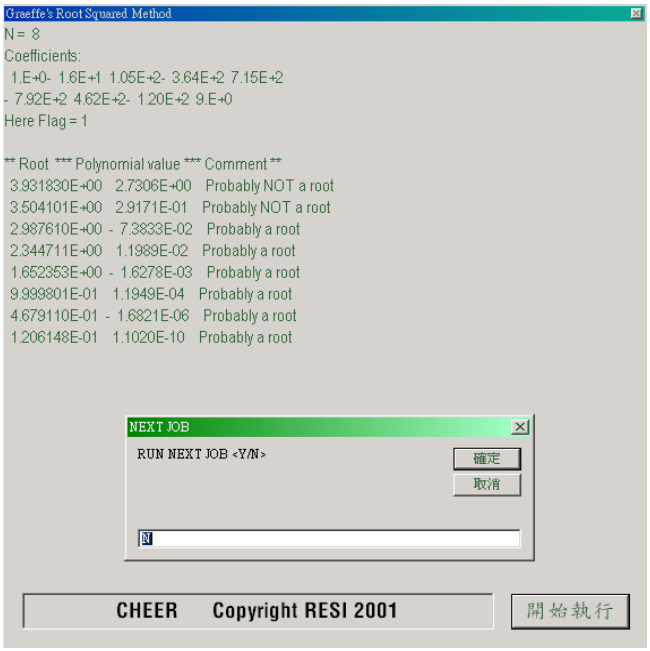
計算所得的根	真正的根
–5.0	–5.0
–2.0	–2.0

2. $f(x) = x^4 - 35x^3 + 146x^2 - 100x + 1 = 0$



計算所得的根	真正的根
30.289	30.289
3.858	3.858
0.843	0.843
0.010	0.010

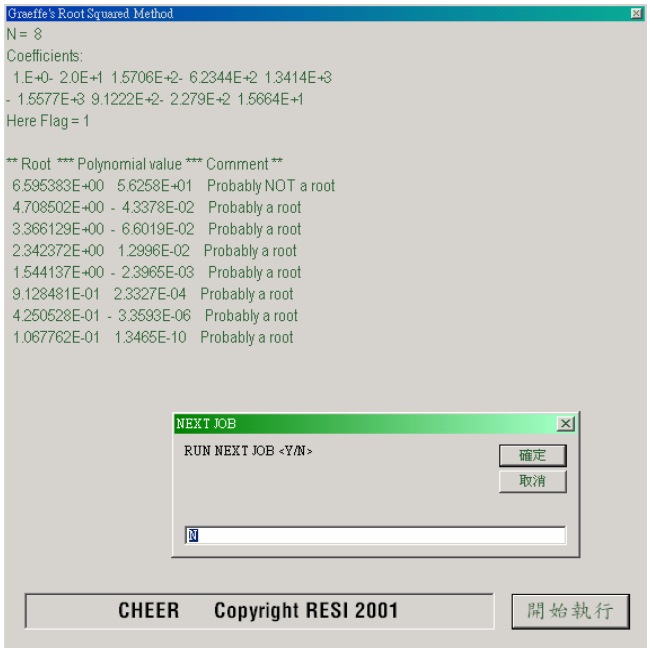
3. $f(x) = x^8 - 16x^7 + 105x^6 - 364x^5 + 715x^4 - 792x^3 + 462x^2 - 120x + 9 = 0$



計算所得的根	真正的根
3.9318	3.8794
3.5041	3.5321
2.9876	3.0000
2.3447	2.3473
1.6524	1.6527
1.0000	1.0000
0.4679	0.4679
0.1206	0.1206

4. $f(x) = x^8 - 20x^7 + 157.0625x^6 - 623.4374x^5 + 1341.445x^4$

$$-1557.656x^3 + 912.2167x^2 - 227.9003x + 15.6643 = 0$$



計算所得的根	真正的根
6.5954	6.5935
4.7085	4.7084
3.3661	3.3665
2.3424	2.3426
1.5441	1.5442
0.9129	0.9129
0.4251	0.4251
0.1068	0.1068

由本例可發現，利用葛瑞菲根平方法只需極少次的迭代即可求得所有根的近似值，故可作爲其他求根方法的最佳啓始值，以供求得更準確的方程式解。

第二節　假位法

Visual Basic

假位法 (False Position Method) 又稱 Regula Falsi，是一種最簡單的求解方程式根的方法，屬於較古老且低效率的一種方法。但是說明這種求解方程式的方法有助於了

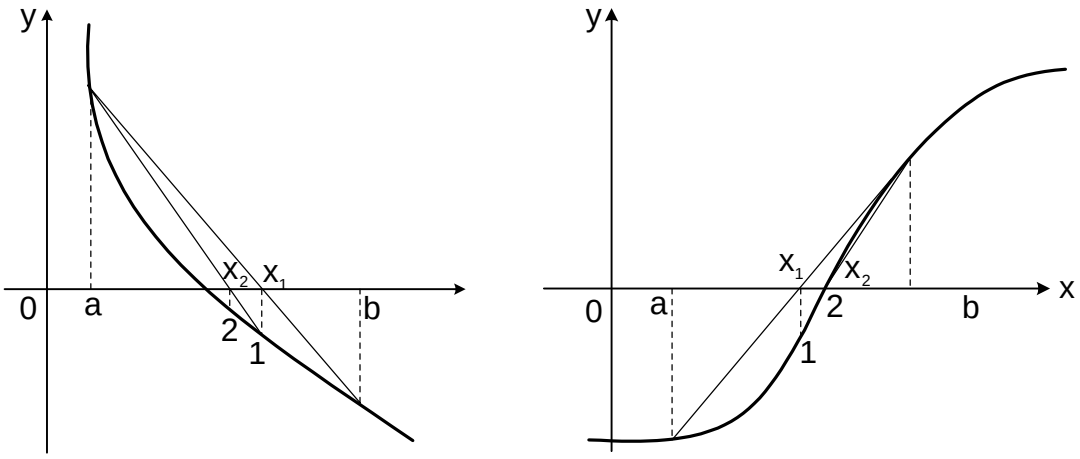
解求根的基本策略，因此，本節仍略加以說明。

假設已知方程式 $f(x)=0$，在 a 與 b 之間存在一個根 r，其中 $a < r < b$，則 $f(a)f(b)<0$。如圖 5.2 (a)所示，連接 $(a, f(a))$ 及 $(b, f(b))$ 之直線必交 x 軸於 (a, b) 間之一點 x_1。由直線方程式得

$$x_1 = \frac{a\ f(b) - b\ f(a)}{f(b) - f(a)} \qquad \textbf{(5-2.1)}$$

此時，由於點 1 與 $(b, f(b))$ 同側，因此，以 $(x_1, f(x_1))$ 取代 $(b, f(b))$ 繼續利用方程式 (5-2.1) 求得下一近似值。

如圖 5.2 (b)之情況，於第一次求解後以 $(x_1, f(x_1))$ 取代 $(b, f(b))$，所得第二次近似根變成與 $(a, f(a))$ 同側，因此，改用 $(x_2, f(x_2))$ 取代 $(a, f(a))$，再繼續進行求解過程。



(a) 近似根位於根之同側　　(b) 近似根位於根之異側

圖 5.2　利用假位法求解方程式的根

第三節　割線法

Visual Basic

割線法 (Secant method) 之迭代式與假位法相當類似，但其收斂速度較快。割線法是在根的附近利用直線來模擬原函數曲線。若方程式有實根，則直線將交 x 軸於該根附近，見圖 5.3。x_1 及 x_2 爲迭代序列的兩個最初值，此二值可利用適當判斷方法選

擇，或隨便猜測決定。

x_1 及 x_2 點之函數值分別爲 $f(x_1)$ 及 $f(x_2)$，則通過該二點之直線方程式爲

$$\frac{f(x)-f(x_1)}{x-x_1}=\frac{f(x_1)-f(x_2)}{x_1-x_2} \qquad \textbf{(5-2.2)}$$

直線與 x 軸之交點即爲根之近似值，故假設該點之函數值 $f(x)=0$，由上式得根之近似值爲

$$x=\frac{x_1 f(x_2)-x_2 f(x_1)}{f(x_2)-f(x_1)} \qquad \textbf{(5-2.3)}$$

所得的 x 值即圖 5.3 中的 x_3；再以 $(x_2, f(x_2))$ 及 $(x_3, f(x_3))$ 兩點作一新的直線，與 x 軸交點爲 x_4。重複此程序，直到達到所要求的準確度或迭代至一定次數而未收斂時才停止。

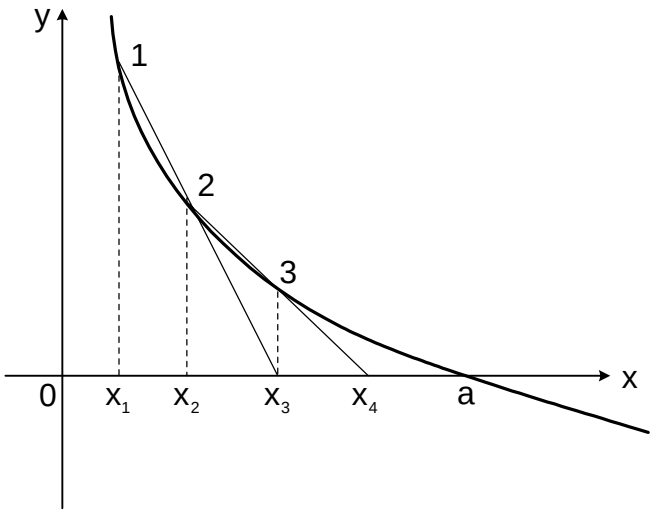


圖 5.3　割線法求解方程式的根

割線法的一般式爲：

$$x_{k+1}=\left[x_{k-1}f(x_k)-x_k f(x_{k-1})\right]/\left[f(x_k)-f(x_{k-1})\right] \qquad \textbf{(5-2.4)}$$

割線法由於不必像假位法測試近似值之相對位置，因此，其執行速度較快；但其缺點則是必須有理想的最初猜測值，否則可能無法收斂。

例題 5-2　方程式之根

試利用割線法求方程式 $f(x) = x^5 - x^3 - x - 14 = 0$ 之根。

解：

TOP-DOWN 設計：

工程及科學應用上，很少盲目地尋求任一方程式之根。較常見的情況是希望找出一方程式在某一範圍內可能存在的根，因此，在設計本程式時，我們爲尋根範圍作了上下限的限制。此外，利用割線法求根，若最初猜測值不佳，迭代結果可能不收斂，因此，程式設計時亦應作最大迭代次數之限制。

主　程　式

輸入兩個最初猜測值 XST 及 XND
輸入 X 之上下限　XL 及 XH
輸入容許誤差值 EPS
輸入最大迭代次數 ITMAX
執行 SECANT 副程式
END

副程式 ***Secant(XST, XND, XL, XH, EPS, ITMAX)***

@程式參數歸零		
ChangeBoundFlag% = 0　,　ConvergentFlag% = 0		
K = 0		
H = XST – XND		
@開始執行迭代		
	X(1) = XST	
	X(2) = XND	
	R(2) = f(XND)	
	II = 0	
		II = II+1
		R(1) = f(X(1))
		Z = X(1) – (R(1)*(X(1) – X(2))/(R(1) – R(2))
		@檢驗下限
		IF Z <= XL Then　　Else

<table>
<tr><td rowspan="8"></td><td rowspan="8"></td><td rowspan="2">Call Bounds</td><td colspan="3">@檢驗上限</td></tr>
<tr><td colspan="3">If Z >= XH
Then Else</td></tr>
<tr><td rowspan="2">XST = XH</td><td>Call Bounds</td><td colspan="2">E = |Z-X(1)/Z|</td></tr>
<tr><td>XST = XL</td><td colspan="2">@檢驗是否收斂</td></tr>
<tr><td>XND = XH + H</td><td>XND = XL + H</td><td colspan="2">If E>EPS
Then Else</td></tr>
<tr><td>K = 1</td><td>K = 1</td><td>X(2) = X(1)</td><td>Z = X(1)</td></tr>
<tr><td rowspan="2">CheckBoundFlag% = 1</td><td rowspan="2">CheckBoundFlag% = 1</td><td rowspan="2">X(1) = Z</td><td>RES = R(1)</td></tr>
<tr><td>Print Results</td></tr>
<tr><td rowspan="2">Exit Do</td><td rowspan="2">Exit Do</td><td rowspan="2">R(2) = R(1)</td><td>ConvergentFlag%=1</td></tr>
<tr><td>Exit Do</td></tr>
<tr><td colspan="5">Loop While II < ITMAX</td></tr>
<tr><td colspan="5">Loop While CheckBoundsFlag% = 1</td></tr>
</table>

符號說明：

FCT： 函數 f(x)

H： XST-XND

II： 迭代次數

K： 超限旗幟

R： 函數值

RES： 函數殘值

X： X(1)及 X(2)表 X 之左、右側值

XH , XL： X 之上下限

XST , XND： 最初猜測值

Z： X 之下一次迭代值

程式列印：

```
'     ******************************************************
'     * Secant Method for Solving Single Equation *
'     ******************************************************
'
Sub SecantMethod(xpos, ypos)
Do
      Cls
```

```
        Print "Enter two initial guesses: "

        X1st = Val(InputBox("1st guess = ", "First Guess", X1st, xpos, ypos))
        Print "First Guess X1 = "; X1st

        X2nd = Val(InputBox("2nd guess = ", "Second Guess", X2nd, xpos, ypos))
        Print "Second Guess X2 = "; X2nd
        Print

        Print "Upper and Lower Bounds of X:"
        Xlow = Val(InputBox("Lower bound of X = ", "", Xlow, xpos, ypos))
        Print "Lower Bound of X = "; Xlow
        Xhigh = Val(InputBox("Higher bound of X = ", "", Xhigh, xpos, ypos))
        Print "Upper Bound of X = "; Xhigh
        Print

        EPS = Val(InputBox("Error Tolerance EPS = ", "", EPS, xpos, ypos))
        Print "Error Tolerance EPS = "; EPS

        ITMAX = Val(InputBox("Maximum Iteration ITMAX = ", "", ITMAX, xpos, ypos))
        Print "Maximum Iteration ITMAX = "; ITMAX

        Call Secant(X1st, X2nd, Xlow, Xhigh, EPS, ITMAX)
        RepeatYN$ = InputBox("Try Next Solution", "NEXT TRIAL", "Y", xpos, ypos)
Loop While RepeatYN$ = "Y" Or RepeatYN$ = "y"
End Sub


Function FCT(Z) As Double
'       -------------------
'     DEFINE FUNCTION
'       -------------------
FCT = Z ^ 5 - Z ^ 3 - Z - 14
End Function


Sub Secant(XST, XND, XL, XH, EPS, ITMAX)
'         *************************
'         * SECANT METHOD *
'         *************************
'   X1ST = 1ST TRIAL VALUE
'   X2ND = 2ND TRIAL VALUE
'         X = LEFT OR RIGHT X
'     FCT = FUNCTION TO BE SOLUED
'         R = FUNCTION VALUES
'     RES = RESIDUE
```

```
'         K = BOUNDS INDEX
'
Dim X(2), R(2)
ChangeBoundFlag% = 0
ConvergentFlag% = 0
K = 0
H = XST - XND
'
'    -- ITERATION
'
Do
        X(1) = XST
        X(2) = XND
        R(2) = FCT(XND)
        II = 0
        Do
                II = II + 1
                R(1) = FCT(X(1))
                Z = X(1) - R(1) * (X(1) - X(2)) / (R(1) - R(2))
'
'    -- CHECK LOWER BOUND
'
                If Z <= XL Then
                        Call Bounds(K)
                        XST = XH
                        XND = XH + H
                        K = 1
                        ChangBoundFlag% = 1
                        Exit Do
                End If
'
'    - CHECK UPPER BOUND
'
                If Z >= XH Then
                        Call Bounds(K)
                        XST = XL
                        XND = XST + H
                        K = 1
                        ChangeBoundFlag% = 1
                        Exit Do
                End If
'
'    - CHECK CONVERGENCE
'
                E = Abs((Z - X(1)) / Z)
                If E <= EPS Then
```

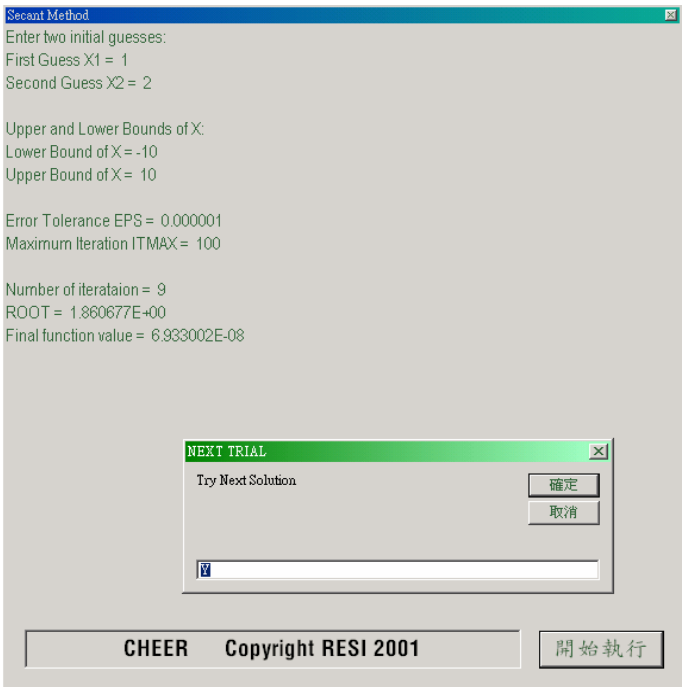
```
                    Z = X(1)
                    RES = R(1)
'
'    -- O.K. PRINT RESULT
'
                    Print
                    Print "Number of iterataion = "; II
                    Print "ROOT = ";
                    Print Format(Z, " 0.000000E+00")
                    Print "Final function value = ";
                    Print Format(RES, " 0.000000E+00")
                    ConvergentFlag% = 1
                    Exit Do
                Else
'
'    - INSIDE BOUNDS
'       NOT CONVERGENT
'
                    X(2) = X(1)
                    X(1) = Z
                    R(2) = R(1)
                End If
            Loop While II < ITMAX
Loop While ChangeBoundFlag% = 1
'
'    -- REPAET WHILE II < ITMAX
'
If ConvergentFlag% <> 1 Then
        Print "** SECANT METHOD NOT CONVERGED ** too many iteration is required"
End If
'-- CHEER by Ron Hsin Chang, Copyright 2001
End Sub


Sub Bounds(K)
'
'     ---------------------
'        SUBROUTINE BOUNDS
'     ---------------------
'
If K = 1 Then
        Print "** SECANT METHOD NOT CONVERGED ** root ;outside bounds"
        MsgBox ("Secant Method NOT Convergent")
End If
End Sub
```

副程式使用說明：

副程式 Sub Secant(XST, XND, XL, XH, EPS, ITMAX) 使用方法說明如下：

1. 設定第一個猜測值 XST 及第二個猜測值 XND。
2. 設定下限 XL、上限 XH、誤差容忍度 EPS 及迭代次數上限 ITMAX。
3. 將所要求解的函數設定在 ***Function FCT(Z) As Double*** 中。
4. 呼叫副程式 Call Secant(XST, XND, XL, XH, EPS, ITMAX)。
5. 程式會列印所得結果。

測試結果：



第四節　韋格斯坦法

Visual Basic

韋格斯坦法 (Wegstein Method) 採用投射迭代技巧。首先將方程式 $f(x) = 0$ 改寫成 $x = g(x)$ 的形式，再由 $g(x)$ 曲線上的兩已知點投射求得第三點；然後連續利用最後所得的兩個 x 值，重複此投射步驟，求得新的嘗試值。如圖 5.4，實線表函數值 F =

$g(x)$，虛線表函數值 $F = x$，由於我們已將原方程式改寫成 $x = g(x)$ 的形式，因此，點線與曲線的交點即表示方程式 $f(x) = 0$ 之解。

由於韋格斯坦法係利用兩點投射，求得第三點，因此，可先利用直接代入法求得最初兩點，再進行投射。首先，由嘗試值 x_1 開始，求得 $F_1 = g(x_1)$；再利用虛線函數求得 $x_2 = F_1$，建立最初的兩個據點 (x_1 , F_1) 及 (x_2 , F_2)。1→2 兩點所連成的直線方程式爲：

$$1 \to 2 : \frac{F - F_1}{x - x_1} = \frac{F_1 - F_2}{x_1 - x_2} \qquad \textbf{(5-3.1)}$$

與直線 $F = x$ 之交點，可將 F 以 x 取代，代入上式，直接求得新的 x 值爲

$$x_3 = \frac{x_1 F_2 - F_1 x_2}{x_1 - x_2 - F_1 + F_2} \qquad \textbf{(5-3.2)}$$

投射所得 x_3 值，再利用 $F = g(x)$，求得第 3 點之座標爲 (x_3 , F_3)。然後再利用第 2 點及第 3 點重複此步驟，求出第四點座標。餘此類推，直到達到滿意的收斂程度爲止。

韋格斯坦法收斂速度相當快，且在任何情況下均可收斂。而較簡單的直接迭代法及假位法 [2 , 3] 由於有不收斂及收斂速度慢的問題，因此，使用程度不如韋格斯坦法普遍。

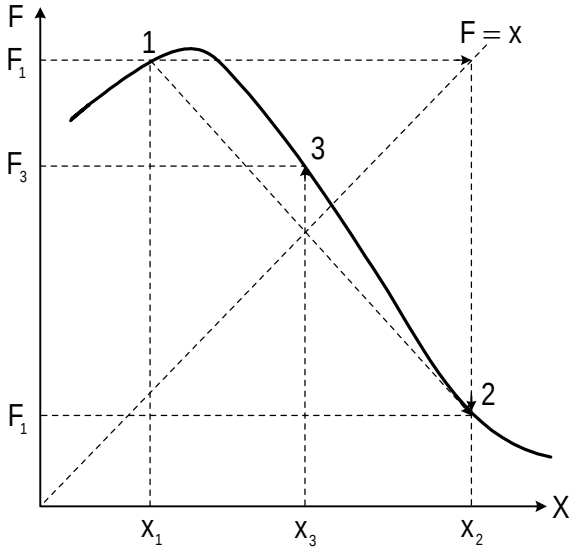


圖 5.4　韋格斯坦法求解方程式的根

例題 5-3 異丁烷之莫耳體積

Beattie-Bridgeman 狀態方程式可寫成

$$V = (R\,T + \frac{\beta}{V} + \frac{\gamma}{V^2} + \frac{\delta}{V^3})\,\frac{1}{P} \qquad \textbf{(5-3.3)}$$

其中

$$\beta = R\,T\,B_0 - A_0 - R\,C/T^2$$
$$\gamma = -R\,T\,B_0\,b + a\,A_0 - R\,B_0\,C/T^2$$
$$\delta = R\,B_0\,b\,c/T^2$$

R = 氣體定律常數 (atm – ℓ/°Kg – mole)(= 0.08206)

T = 溫度 (°K)

P = 壓力 (atm)

V = 莫耳體積 (ℓ / g – mole)

許多常見氣體的常數 A_0、B_0、a、b、c 均可由文獻 [4] 找到。異丁烷的常數值爲

$$A_0 = 16.6037$$
$$B_0 = 0.2354$$
$$a = 0.11171$$
$$b = 0.07697$$
$$c = 3.00\times10^6$$

試求溫度爲 408°K ，壓力爲 36 atm 時，異丁烷的莫耳體積爲何？

解：

TOP-DOWN 設計

利用韋格斯坦法解方程式可分成兩大步驟：

1. 利用直接替代法找出第 1 及第 2 點。
2. 重複利用式 (5-3.2) 求出下一投射位置。

$$x_3 = \frac{x_{i-2}F_{i-1} - F_{i-2}x_{i-1}}{x_{i-2} - x_{i-1} - F_{i-2} + F_{i-1}} \qquad \textbf{(5-3.4)}$$

主　程　式

輸入參數 A_0，B_0，A，B，C	
計算 BT，GM，DET	
設定 V 起始值	
	*VC = (R * T + BT / V + GM / V / V + DET / V / V / V) / P*
	I = I+1
	列印 V, VC
	Call Wegstein
Repeat While NC>1	
END	

Sub Wegstein(X, XC, RTL, NR, XA, YA, ConvergentFlag%)

@檢驗是否收斂		
If |(X-XC)/(X+XC)|<RTL Then		Else
X = XC	If NC = 1 Then	Else
	XA(NR) = X	XT=(XA(NR)*XC-YA(NR)*X)/(XA(NR)-X+XC-YA(NR)
NC = 1 (已收斂)	YA(NR) = XC	XA(NR)=X
	X = XC	YA(NR)=XC
	NC = 2	X=XT
Return		

符號說明：

A_0，B_0，A，B，C：分別表 A_0、B_0、a、b、c

BT，GM，DET：分別表 β，γ，δ

I：　迭代次數

NC：　收斂指標，NC = 0 執行直接替代

NC = 1 表示已收斂

NC = 2 重複投射

副程式中使用 ConvergentFlag%

NR：　用於解多個方程式，標示方程式用

P：　壓力

R：　氣體定律常數

T：　溫度

V，VC：莫耳體積及其投射計算值

X：　　未知數 x

XA：　x 之前一次計算保留值

XC：　x 之下一次計算值

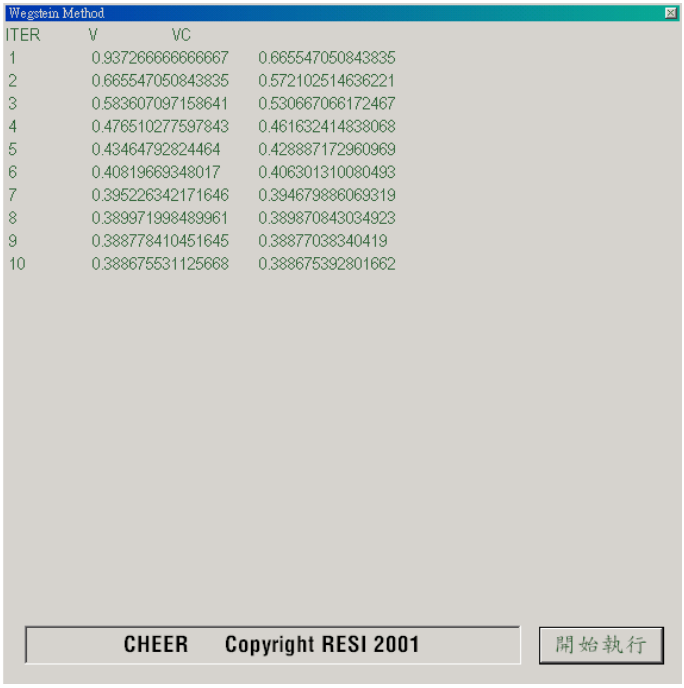
YA：　XC 之前一次計算保留值

程式列印：

```
'       ********************************
'         WEGSTEIN METHOD
'       ********************************

Sub WegsteinMethod(Xpos, Ypos)

Cls
Dim XA(10), YA(10)
'       ----------------------
'         DEFINE THE PROBLEM
'       ----------------------
A = 0.11171
B = 0.07697
C = 3000000#
A0 = 16.6037
B0 = 0.2354

R = 0.0827
T = 408#
P = 36#

'     PRELILMINARY CALCULATION
BT = R * T * B0 - A0 - R * C / T / T
GM = -R * T * B0 * B + A * A0 - R * B0 * C / T / T
DET = R * B0 * B * C / T / T
V = R * T / P
I = 0
'
' START ITERATION
Print "ITER", "V", "VC"
'
RTL = 0.000001: 'RELATIVE ERROR TOLERANCE
'
'       -------------
'         ITERATION
'       -------------
```

```
NC% = 0
Do
        VC = (R * T + BT / V + GM / V / V + DET / V / V / V) / P
        XC = VC
        NR = 1
        I = I + 1
        Print I, V, VC
        Call Wegstein(X, XC, RTL, NR, XA, YA, NC)
        V = X
Loop While NC > 1
End Sub


Sub Wegstein(X, XC, RTL, NR, XA, YA, ConvergentFlag%)
'
'     ------------------------------------------------
'          SUBROUTINE WEGSTEIN
'     ------------------------------------------------
'
'    XA,YA    = RESERVED VALUES FOR MULTIFUNCTION CALLS
'    RTL      = REL. TOLERANCE
'    ConvergentFlag%
'               = CONVERGE INDEX
'                    1. CONVERGENT
'                    2. NOT CONVERGENT
'     X        = TRIAL VALUE
'     XC       = CALCULATED VALAUE
'     NR       = ROUTINE CALL NUMBER FOR MULTIPLE EQNS
'
If (Abs((X - XC) / (X + XC)) < RTL) Then
        X = XC
        ConvergentFlag% = 1
ElseIf ConvergentFlag% <= 1 Then
        XA(NR) = X
        YA(NR) = XC
        X = XC
        ConvergentFlag% = 2
Else
        XT = (XA(NR) * XC - YA(NR) * X) / (XA(NR) - X + XC - YA(NR))
        XA(NR) = X
        YA(NR) = XC
        X = XT
End If
End Sub
```

測試結果：


Wegstein Method
ITER V VC
1 0.937266666666667 0.665547050843835
2 0.665547050843835 0.572102514636221
3 0.583607097158641 0.530667066172467
4 0.476510277597843 0.461632414838068
5 0.43464792824464 0.428887172960969
6 0.40819669348017 0.406301310080493
7 0.395226342171646 0.394679886069319
8 0.389971998489961 0.389870843034923
9 0.388778410451645 0.38877038340419
10 0.388675531125668 0.388675392801662
CHEER Copyright RESI 2001
開始執行

結果討論：

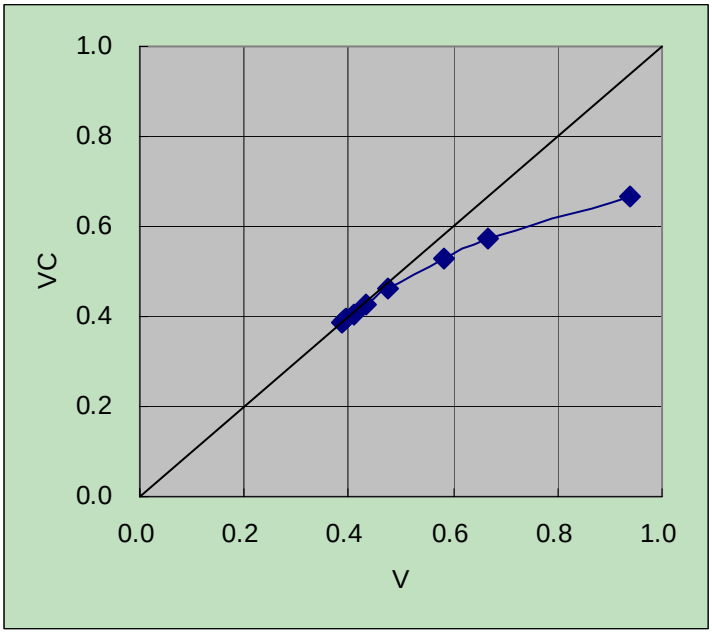

1.0
0.8
0.6
0.4
0.2
0.0
VC
0.0
0.2
0.4
0.6
0.8
1.0
V

圖 5.5　韋格斯坦法之收斂性

本程式利用理想氣體方程式求出異丁烷的莫耳體積，作爲最初值。呼叫 Wegstein 副程式，進行所有必要的迭代計算，迭代過程如圖 5.5 所示，由圖亦可見韋格斯坦法收斂速度相當快速。

第五節　牛頓法

Visual Basic

設方程式 $f(x) = 0$ 之準確根爲 a，x_1 表示 a 之近似解，則函數 $f(x)$ 對 $x = x_1$ 之泰勒級數展開式爲：

$$f(x) = f(x_1) + (x - x_1) f'(x_1) + \frac{(x - x_1)^2}{2!} f''(x_1) + \cdots\cdots = 0 \qquad \textbf{(5-4.1)}$$

設 x_1 與 a 之差爲 h，即 $a = x_1 + h$；代入上式可以得到

$$f(x_1 + h) = f(x_1) + hf'(x_1) + \frac{h^2}{2} f''(x) + \cdots\cdots = 0 \qquad \textbf{(5-4.2)}$$

捨去 h 之二次方以上各項，且令二次方以上各項接近於 0，則可以得到

$$f(x_1 + h) \cong f(x_1) + hf'(x_1) \cong 0 \qquad \textbf{(5-4.3)}$$

解此方程式得到 h 的值爲

$$h = -\frac{f(x_1)}{f'(x_1)} \qquad \textbf{(5-4.4)}$$

由於方程式 (5-4.2) 中，h 的二次方以上各項都被我們捨去，因此，$x_1 + h$ 值可能不等於準確根 a，但其值能夠較 x_1 更爲接近準確的根 a，仿照這種方法，可類推得到迭代近似值的表示式爲

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \qquad \textbf{(5-4.5)}$$

利用方程式 (5-4.5) 求得方程式 $f(x)$ 之近似根的方法稱爲牛頓法 (Newton's Method)。由方程式 (5-4.2) 知牛頓法的捨去誤差爲 h^2 項，表示第 n 個近似根的誤差爲 0.1 時，第 $n + 1$ 個近似根的誤差約爲$(0.1)^2 = 0.01$，因此，收斂速度相當快。但使用牛頓法解方程式時，應該注意這種方法有兩個缺點：(一) 起始值若不在根之附近，利用

牛頓法時常無法收斂。因此，實際應用時，最好能配合他種數值方法先找出根之大略位置，再利用牛頓法求得根之近似解。(二) 利用牛頓法進行計算時，必須使用一次導函數 *f'(x)*，對於較複雜的方程式，使用者必須小心縝導求出 *f'(x)*。此外，若在根的附近 *f'(x)* 很小時，*f(x)*的計算誤差會在求 *f(x)* / *f'(x)* 時被放大，使得所得到的根準確度變差。

就計算效率的觀點而言，割線法每一次迭代只需計算一個函數值，而牛頓法則需計算 *f(x)* 及 *f'(x)* 兩個函數值，計算效率略差，因此，一般性的計算機程式大部分採用割線法來設計。

例題 5-4 多成分溶液系統之沸點

一混合溶液系統中，含有 *n* 種成分，其莫耳分率分別爲 x_1，x_2，……，x_n。當此系統達平衡時，氣相莫耳分率 y_i *(i = 1 , 2 , …… , n)*可利用 Antoine 方法計算之。

$$y_i = \exp\left[A_i + \frac{B_i}{T + C_i}\right] \cdot \frac{\gamma_i x_i}{P} \qquad \textbf{(5-4.6)}$$

已知一混合溶液中含有苯、甲苯及對二甲苯三種成分，試作圖表示此混合液沸點 T 與系統壓力 P 之關係。

數　據：

成　　分		X_i	Antoine 常數			活性係數
			A_i	B_i	C_i	γ_i
1	苯	0.5	16.07982	–2871.05	224.02	1.0
2	甲苯	0.3	15.98639	–3080.01	218.69	1.0
3	對二甲苯	0.2	16.57200	–3621.00	226.41	1.0

解：

TOP-DOWN 設計：

系統達沸點時，蒸氣相莫耳分率和爲 1，即 $\sum_{i=1}^{N} y_i = 1$，故可令函數 *f(T)* 爲

$$f(T)=\sum_{i=1}^{N} y_i - 1 \qquad \textbf{(5-4.7)}$$

其導函數爲

$$\begin{aligned} f'(T) &= \sum_{i=1}^{N}\left\{\frac{-B_i}{(T+C_i)^2}\cdot\exp\left[A_i+\frac{B_i}{T+C_i}\right]\cdot\frac{\gamma_i x_i}{P}\right\} \\ &\equiv -\sum_{i=1}^{N}\frac{B_i y_i}{(T+C_i)^2} \end{aligned} \qquad \textbf{(5-4.8)}$$

由於牛頓法要求有良好的起始值，因此，假設沸點之起始值是在壓力 P 時各成分的沸點 (T_i) 的液相莫耳分率加權平均值，亦即

$$T_0=\sum_{i=1}^{N} x_i T_i \qquad \textbf{(5-4.9)}$$

其中 T_i 可由方程式 (5-4.6)，設 $x_i=1$, $y_i=1$, $\gamma_i=1$ 求得

$$T_i=\frac{B_i}{Ln(P)-A_i}-C_i \qquad \textbf{(5-4.10)}$$

主　程　式

```
輸入參數
    輸入自變數 P
    Call Newton
    列印結果
重複執行
```

副程式 Newton(N, F, DF, T, P, A, B, C, X, Gama, RelErr)

```
@利用方程式(5-4.9)計算起始值
Call InitialGuess(N, T, P, A, B, C, X)
    T1 = T
    @計算 f(x)及 f'(x)
    Call FunctionDefine(N, F, DF, T, P, A, B, C, X, Gama)
    @利用方程式(5-4.5)計算下一個近似值
    T = T – F/DF
若 |(T1-T)/T|>10^-4 則重複執行
RETURN
```

副程式 **InitialGuess(N, T, P, A, B, C, X)**

T = 0
Do for I = 1 to N
TI = $(B_i/(Ln(P) - A_i)) - C_i$
T = T + x_i * TI
RETURN

副程式 **FunctionDefine(N, F, DF, T, P, A, B, C, X, Gama)**

F = -1
DF = 0
Do for I = 1 to N
@利用方程式(5-4.6)、(5-4.7)、(5-4.8)計算 F 及 DF
Y(I) = $Exp(A_i + B_i/(T+C_i)) * \gamma_i \ x_i / P$
F = F + Y(I)
DF = $DF - B_i Y(I)/(T + C_i)^2$
RETURN

符號說明：

A，B，C：Antoine 常數

DF：　$f'(T)$

F：　$f(T)$

GAMA：　γ_i，活性係數

P：　壓力

T：　溫度

X：　液相莫耳分率

Y：　蒸氣相莫耳分率

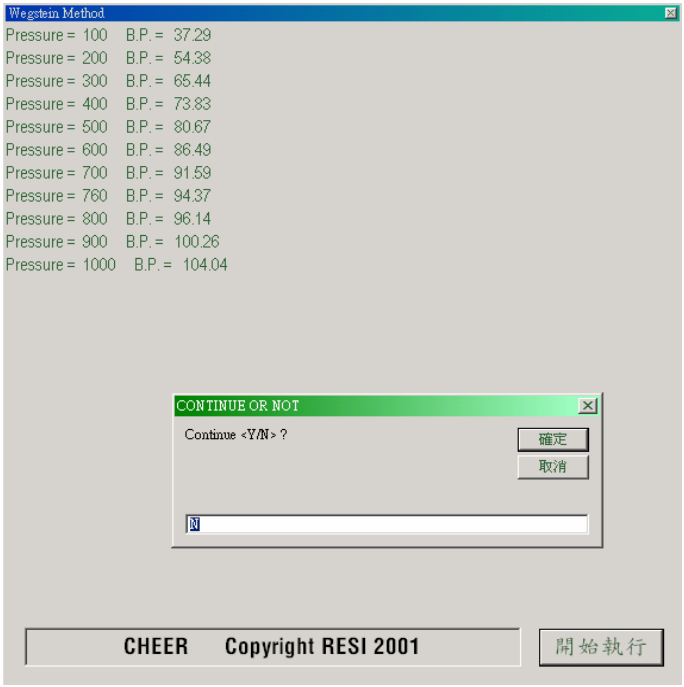
程式列印：

```
'       ********************************************************
'       *          NEWTON'S          METHOD          *
'       ********************************************************
'
Sub NewtonMethod(xpos, ypos)
'     A,B,C = ANTOINE CONST.
'             X = MOLE FRACTION
'             Y = VAP. MOLE FRACTION
```

```
'             T = BOILING POINT
'             P = PRESSURE    (mmHg)
'       GAMA = ACTIVITY COEFF.
'
'     F(X)    = FUNCTION TO BE SOLUED
'     DF(X)   = 1st DERIVATIVE OF F(X)
Dim A(3), B(3), C(3), Gama(3), X(3), Y(3)
'       --------------
'          DATA BLOCK
'       --------------
X(1) = 0.5
X(2) = 0.3
X(3) = 0.2
A(1) = 16.07382
A(2) = 15.98639
A(3) = 16.572
B(1) = -2871.05
B(2) = -3080.01
B(3) = -3621
C(1) = 224.02
C(2) = 218.69
C(3) = 226.41
Gama(1) = 1
Gama(2) = 1
Gama(3) = 1
RelErr = 0.00001
N = 3
'       ---------------
'          MAIN PROGRAM
'       ---------------
Cls
Do
       RelErr = Val(InputBox("Enter Releative Error =", "", RelErr, xpos, ypos))
       P = Val(InputBox("PRESSURE = ", "Pressure", P, xpos, ypos))
       Call Newton(N, F, DF, T, P, A, B, C, X, Gama, RelErr)
       Print "Pressure = "; P;
       Print "        B.P. = ";
       Print Format(T, "    ####.##")
       ContinueYN$ = InputBox("Continue <Y/N> ?", "", "N", xpos, ypos)
Loop While ContinueYN$ = "Y" Or ContinueYN$ = "y"
End Sub


Sub Newton(N, F, DF, T, P, A, B, C, X, Gama, RelErr)
'
'     ---------------------
```

```
'          SUBROUTINE NEWTON
'     ---------------------
'
Call InitialGuess(N, T, P, A, B, C, X)
Do
       T1 = T
       Call FunctionDefine(N, F, DF, T, P, A, B, C, X, Gama)
       T = T - F / DF
Loop While Abs((T - T1) / T) >= RelErr
End Sub


Sub FunctionDefine(N, F, DF, T, P, A, B, C, X, Gama)
Dim Y(3)
'     ---------------------
'          DEFINE      FUNCTION
'     ---------------------
F = -1
DF = 0!
For I = 1 To N
       Y(I) = Exp(A(I) + B(I) / (T + C(I))) * Gama(I) * X(I) / P
       F = F + Y(I)
       DF = DF - B(I) * Y(I) / (T + C(I)) ^ 2
Next I
End Sub


Sub InitialGuess(N, T, P, A, B, C, X)
'     ----------------------
'          INITIAL CONDITIONS
'     ----------------------
T = 0
For I = 1 To N
       TI = B(I) / (Log(P) - A(I)) - C(I)
       T = T + X(I) * TI
Next I
End Sub

Private Sub Start_Click()
xpos = 8500
ypos = 6000
Call NewtonMethod(xpos, ypos)
End Sub
```

測試結果：



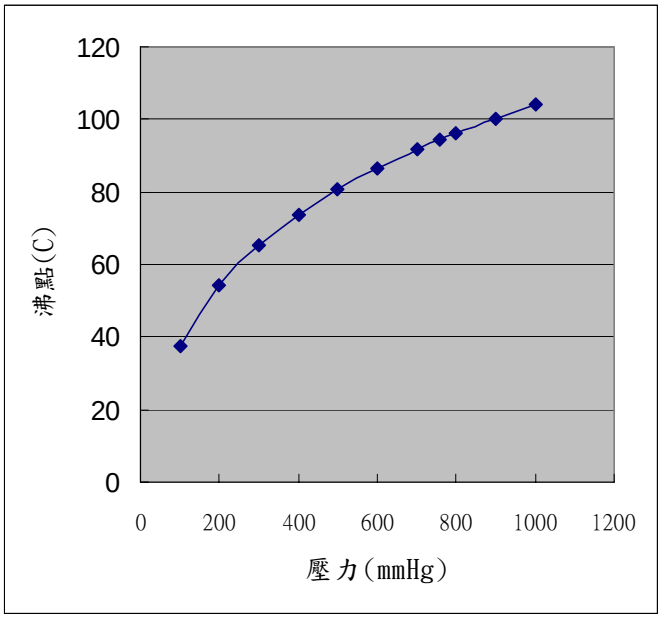


圖 5.6　50% 苯，30% 甲苯，20% 對二甲苯溶液之沸點

第六節　非線性聯立方程式

Visual Basic

本書第四章中曾探討了線性聯立方程式的求解方法，比較起來非線性聯立方程式的求解就可能略爲困難，其原因有二：(1) 非線性聯立方程式通常無法求得真正的解；(2) 非線性聯立方程式可能有一組以上的解。因此，通常需利用電子計算機以迭代法來尋求方程組的解；但迭代法只能找出一組可能的解；而無法一次找出所有可能的解。因此，要找出一組非線性聯立方程式的所有可能解，就必須由不同的起始值開始進行迭代，這項缺點相當不易克服。所幸在大部分工程或科學系統中，通常只有一組解具有實際的物理意義，其他的解可能都存在於無意義的範圍 (例如壓力或濃度爲負值)。

解非線性聯立方成組的方法，常見的方法有三種：

1. 連續取代法 (Successive substitution)

2. 牛頓拉福森法 (Newton-Raphson method)

3. 函數極小化法 (Function minimization)

第七節中，我們將介紹連續取代法；第八節中介紹牛頓拉福森法；第九節中介紹利用數值方法求斜率，以改良牛頓拉福森法所得的割線法。函數極小化法由於牽涉較複雜的數學處理，因此，不列入本書討論範圍，有興趣讀者請參考文獻 [5 , 6]。

第七節　連續取代法

Visual Basic

連續取代法解單一方程式或聯立方程組基本原理完全相同，唯一的差別是後者每一次迭代需重新估計一個以上的變數。假設我們要解以下的聯立方程組：

$$f_1(x_1, x_2, x_3, x_4, x_5) = 0 \qquad \textbf{(5-6.1)}$$

$$f_2(x_1, x_2, x_3, x_4, x_5) = 0 \qquad \textbf{(5-6.2)}$$

$$f_3(x_1, x_2, x_3, x_4, x_5) = 0 \qquad \textbf{(5-6.3)}$$

$$f_4(x_1, x_2, x_3, x_4, x_5) = 0 \qquad \textbf{(5-6.4)}$$

$$f_5(x_1, x_2, x_3, x_4, x_5) = 0 \qquad \textbf{(5-6.5)}$$

首先，可先將原方程組 $\underline{f}(\underline{x})=0$ 改寫成 $\underline{x}=g(\underline{x})$ 的型式，如

$$x_1 = g_1(\underline{x}) \quad \textbf{(5-6.6)}$$

$$x_2 = g_2(\underline{x}) \quad \textbf{(5-6.7)}$$

$$x_3 = g_3(\underline{x}) \quad \textbf{(5-6.8)}$$

$$x_4 = g_4(\underline{x}) \quad \textbf{(5-6.9)}$$

$$x_5 = g_5(\underline{x}) \quad \textbf{(5-6.10)}$$

我們若提供一組最初假設值，代入方程式 (5-6.6) 至 (5-6.10)，即可求得一組新的近似值，重複此步驟直到收斂爲止。利用上述策略有時可能不穩定，且會發散至無窮大而無法收斂至一組定值。雖然，利用其導函數可判斷系統是否會穩定，但由於求導函數過程相當繁瑣，少有人真正願意去測試，較常見的做法倒是試著解解看，如果不收斂，再更換一組 $\underline{x}=\underline{g}(\underline{x})$ 的轉換方式，直到能求得收斂的結果爲止。

利用導函數判斷系統 $\underline{x}=\underline{g}(\underline{x})$ 迭代方式爲收斂的準則爲

$$\sum_{i=1}^{N}\left|\frac{\partial g_i(\underline{x})}{\partial x_i}\right|<1\ ;\qquad i=1,2,\cdots\cdots,N \quad \textbf{(5-6.11)}$$

例題 5-5　設計問題 D-V

利用連續迭代法解設計問題 D-V 之聯立方程組

$$u_1+u_2=2.78u_3 \quad \textbf{(5-6.12)}$$

$$58.9+40(u_1{}^2-u_2{}^2)=0 \quad \textbf{(5-6.13)}$$

$$-156.0+40(u_2{}^2+0.3u_3{}^2)=0 \quad \textbf{(5-6.14)}$$

解：

TOP-DOWN 設計：

將方程式 (5-6.12) 至 (5-6.14) 改寫成 $\underline{x}=\underline{g}(\underline{x})$ 的型式：

$$u_2=\sqrt{3.9-0.3u_3{}^2} \quad \textbf{(5-6.15)}$$

$$u_1=\sqrt{u_2{}^2-\frac{58.9}{40}} \quad \textbf{(5-6.16)}$$

$$u_3 = (u_1 + u_2)/2.78 \qquad \textbf{(5-6.17)}$$

則只需給一個 u_3 的猜測值，即可由 (5-6.15) 求得 u_2，由 (5-6.16) 求得 u_1，再代入 (5-6.17) 修正 u_3 值，重複此步驟直到收斂爲止。

一般而言，利用連續取代法時，需重複迭代的變數 (如上述作法中的 u_3) 應儘量減少，使我們必須供應的最初猜測值數目可減少。

主 程 式

@輸入起始值		
Input U(3)		
UOLD = U(3)		
	@利用方程式 (5-6.15)、(5-6.16)、(5-6.17) 計算下一近似值	
	$u_2 = \sqrt{3.9 - 0.3u_3^{\,2}}$	
	$u_1 = \sqrt{u_2^{\,2} - \frac{58.9}{40}}$	
	$u_3 = (u_1 + u_2)/2.78$	
	If Abs((U(3) - UOLD) / UOLD) > 0.0001	
	Then	Else
	UOLD = U(3)	ConvergentFlag =0
	ConvergentFlag =1	
Loop While ConvergentFlag =1		
列印結果		

符號說明：

U： 速度(m/s)

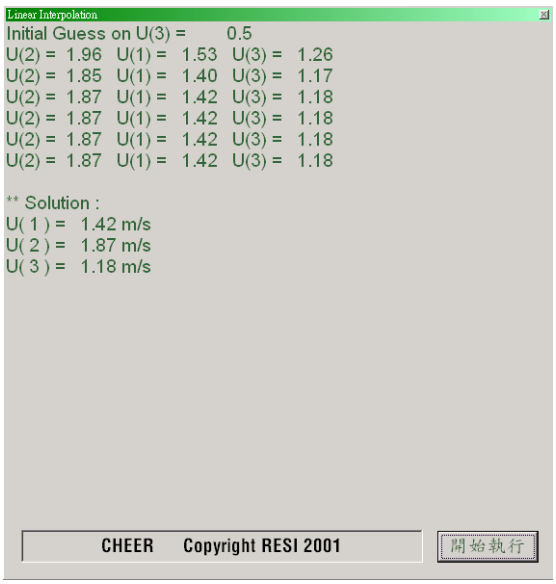
UOLD：上一次迭代所得 U(3) 值

程式列印：

```
'       ************************************************
'       *     SUCCESSIVE     SUBSTITUTION
'       ************************************************
'
Sub Substitution(XPos, Ypos)
'       U = VELOCITY ARRAY
Dim U(3)
Cls
U(3) = Val(InputBox("U(3) Guess = ", "GUESS", U(3)))
Print "Initial Guess on U(3) = ", U(3)
```

```
    UOLD = U(3)
    Do
        U(2) = Sqr(3.9 - 0.3 * U(3) ^ 2)
        Print "U(2) =";
        Print Format(U(2), "    ##.00");

        U(1) = Sqr(U(2) ^ 2 - 58.9 / 40)
        Print "      U(1) = ";
        Print Format(U(1), "    ##.00");

        U(3) = (U(1) + U(2)) / 2.78
        Print "      U(3) = ";
        Print Format(U(3), "    ##.00")

        If Abs((U(3) - UOLD) / UOLD) > 0.0001 Then
                UOLD = U(3)
                ConvergentFlag = 1
            Else
                ConvergentFlag = 0
        End If
    Loop While ConvergentFlag = 1
    Print
    Print "** Solution :"
    For I = 1 To 3
        Print "U("; I; ") = ";
        Print Format(U(I), "    ##.00 m/s")
    Next I
    End Sub

    Private Sub Start_Click()
    XPos = 7500
    Ypos = 6000
    Call Substitution(XPos, Ypos)
    End Sub
```

測試結果：



除了上述的連續取代法以外，在化學工程問題中，時常會見到方程組中有一方程式爲各變數和等於一定值者，例如莫耳分率的和爲 1。此時，可將原來的連續取代法略作修正，以提高其穩定性及收斂速率。例如考慮一方程組：

$$f_1(x_1, x_2, x_3) = 0 \quad \textbf{(5-6.18)}$$

$$f_2(x_1, x_3) = 0 \quad \textbf{(5-6.19)}$$

$$x_1 + x_2 + x_3 = T \quad \textbf{(5-6.20)}$$

首先提供 x_1 之最初估計值 x_1^0，則由以上各式得

$$x_3 = g_2(x_1^0) \quad \textbf{(5-6.21)}$$

$$x_2 = g_1(x_1^0, x_3) \quad \textbf{(5-6.22)}$$

$$x_1^0 + x_2 + x_3 = S \quad \textbf{(5-6.23)}$$

然後重新估計 x_1 爲

$$x_1^1 = x_1^0(T / S) \quad \textbf{(5-6.24)}$$

再利用 (5-6.21) 至 (5-6.23)，求得更接近的估計值。重複此步驟直到 S 值變得相當接近 T 值爲止。注意原來的連續取代法所得 x_1 新估計值爲 $x_1 = T - x_2 - x_3$。此處所使用的方法，會使收斂速率加快。

利用這種改良的連續取代法對於化學平衡、蒸餾等計算特別有效。

第八節　牛頓拉福森法

Visual Basic

牛頓拉福森法和解單一方程式所使用的牛頓法相當類似，例如考慮兩個聯立方程式：

$$f_1(x_1, x_2) = 0 \quad \textbf{(5-7.1)}$$

$$f_2(x_1, x_2) = 0 \quad \textbf{(5-7.2)}$$

首先估計一組近似解 $x_{1,0}$，$x_{2,0}$，若這組近似解與正確解之距離爲 h_1 及 h_2，則將函數 f_1 及 f_2 作泰勒級數展開，並捨去 h^2 以上諸項，得到

$$f_1(x_{1,0}+h_1\,,\,x_{2,0}+h_2)\cong f_1(x_{1,0}\,,\,x_{2,0})+h_1\frac{\partial f_1(x_{1,0}\,,\,x_{2,0})}{\partial x_1}+h_2\frac{\partial f_1(x_{1,0}\,,\,x_{2,0})}{\partial x_2}$$

(5-7.3)

$$f_2(x_{1,0}+h_1\,,\,x_{2,0}+h_2)\cong f_2(x_{1,0}\,,\,x_{2,0})+h_1\frac{\partial f_2(x_{1,0}\,,\,x_{2,0})}{\partial x_1}+h_2\frac{\partial f_2(x_{1,0}\,,\,x_{2,0})}{\partial x_2}$$

(5-7.4)

我們的目的是希望讓 $x_{1,0}+h_1$ 及 $x_{2,0}+h_2$ 變成方程組的解，因此，要求 $f_1(x_{1,0}+h_1\,,\,x_{2,0}+h_2)=0$，且 $f_2(x_{1,0}+h_1\,,\,x_{2,0}+h_2)=0$。故方程式 (5-7.3) 及 (5-7.4) 右側均等於零。而方程式中 $f_1(x_{1,0}\,,\,x_{2,0})$ 及 $f_2(x_{1,0}\,,\,x_{2,0})$ 分別代表方程式 (5-7.1) 及 (5-7.2) 的殘餘值，只要將近似解$(x_{1,0}\,,\,x_{2,0})$代入，即可計算求出。四個導函數分別爲各方程式殘值的 $x_{1,0}$ 及 $x_{2,0}$ 導函數，只要能求得其導函數表示式，再將近似解 $x_{1,0}$ 及 $x_{2,0}$ 代入式中，即可求得各導函數值。故方程式 (5-7.3) 及 (5-7.4) 可改寫成：

$$\left[\frac{\partial f_1(x_{1,0},\,x_{2,0})}{\partial x_1}\right]h_1+\left[\frac{\partial f_1(x_{1,0},\,x_{2,0})}{\partial x_2}\right]h_2=-f_1(x_{1,0}+x_{2,0})\qquad\textbf{(5-7.5)}$$

$$\left[\frac{\partial f_2(x_{1,0},\,x_{2,0})}{\partial x_1}\right]h_1+\left[\frac{\partial f_2(x_{1,0},\,x_{2,0})}{\partial x_2}\right]h_2=-f_2(x_{1,0}+x_{2,0})\qquad\textbf{(5-7.6)}$$

此方程組含有兩個線性方程式及兩個未知數 h_1 及 h_2，故可解出 h_1 及 h_2，求得較佳的近似解 $x_{1,0}+h_1$ 及 $x_{2,0}+h_2$。這種方法可認爲是將一組非線性聯立方程式簡化成一組線性聯立方程式，只要利用第四章所介紹的方法即可求得其解。但由於我們已將原非線性方程組作線性化近似處理，因此，所得的解並非正確解，必須重複迭代，以求得準確的解。

仿上述方法，若考慮含 n 個方程式的聯立方程組

$$\begin{aligned}&f_1(x_1\,,\,x_2\,,\cdots\cdots,\,x_n)=0\\&f_2(x_1\,,\,x_2\,,\cdots\cdots,\,x_n)=0\\&\qquad\vdots\\&f_n(x_1\,,\,x_2\,,\cdots\cdots,\,x_n)=0\end{aligned}\qquad\textbf{(5-7.7)}$$

則我們首先定義導函數矩陣 $\Phi(\underline{x})$ 爲

$$\Phi(\underline{x})=\left[f_{ij}(\underline{x})\right]\;;\quad 1\le i\le n\,,1\le j\le n\qquad\textbf{(5-7.8)}$$

其中

$$f_{ij}(\underline{x}) = \frac{\partial f_i(\underline{x})}{\partial x_j} \quad \textbf{(5-7.9)}$$

導函數矩陣 $\Phi(\underline{x})$ 稱爲 Jacobian，其中 $(\underline{x}) = [x_1, x_2, \cdots\cdots, x_n]^T$ 。再定義向量 $\underline{f}(\underline{x})$ 爲：

$$\underline{f}(\underline{x}) = [f_1(\underline{x}),\ f_2(\underline{x}), \cdots\cdots f_n(\underline{x})]^T \quad \textbf{(5-7.10)}$$

最初估計的近似解向量爲 $\underline{x_0} = [x_{10}, x_{20}, \cdots\cdots, x_{n0}]^T$ 。

$\underline{x}$ 之迭代式爲：

$$\underline{x}_{k+1} = \underline{x}_k + \underline{h}_k \quad \textbf{(5-7.11)}$$

其中 $\underline{h}_k$ 則仿方程式 (5-7.5) 及 (5-7.6) 爲下列線性聯立方程組之解：

$$\Phi(\underline{x})\ \underline{h}_k = -\underline{f}(\underline{x}_k) \quad \textbf{(5-7.12)}$$

利用牛頓拉福森法最重要的要求是所有導函數的數值都必須能夠求得。牛頓拉福森法每一次迭代都需要重新計算這些導函數值。這種方法收斂速度雖然相當快，但需不斷的計算大量的導函數是其最大的缺點。此外，程式設計雖簡單，但要在紙上作業先導出 n^2 個導函數卻是一件即令人無法忍受的工作，想想看，如果要解 20 個非線性聯立方程式，就得先導出 400 個導函數呢！

例題 5-6 反應平衡計算

考慮利用甲烷部分氧化製造合成氣 (Synthesis Gas) 的主要反應：

$$CH_{4(g)} + \tfrac{1}{2}O_{2(g)} = CO_{(g)} + 2H_{2(g)} \quad \textbf{(5-7.13)}$$

$$CH_{4(g)} + H_2O_{(g)} = CO_{(g)} + 3H_{2(g)} \quad \textbf{(5-7.14)}$$

$$H_{2(g)} + CO_{2(g)} = CO_{(g)} + H_2O_{(g)} \quad \textbf{(5-7.15)}$$

假設所有氣體均符合理想氣體定律，則在 2200°F 時這些化學反應的平衡常數可利用反應物及生成物的分壓來表示，分別爲：

$$K_1 = \frac{P_{co}P^2{}_{H_2}}{P_{CH_4}P_{O_2}^{1/2}} = 1.3\times10^{11} \quad \textbf{(5-7.16)}$$

$$K_2 = \frac{P_{co}P^3{}_{H_2}}{P_{CH_4}P_{H_2O}} = 1.7837\times10^{5} \quad \textbf{(5-7.17)}$$

$$K_3 = \frac{P_{co} P_{H_2O}}{P_{CO_2} P_{H_2}} = 2.6058 \quad \textbf{(5-7.18)}$$

其中 P_{CO} ， P_{H_2O} ， P_{H_2} ， P_{CH_4} 及 P_{O_2} 分別爲一氧化碳、水蒸氣、氫氣、甲烷及氧氣的分壓。

甲烷在高溫條件下反應時，也會產生碳的沉積，其反應式爲：

$$C_{(s)} + CO_{2(g)} = 2\,CO_{(g)} \quad \textbf{(5-7.19)}$$

其平衡常數爲

$$K_4 = \frac{P_{co}{}^2}{a_c P_{CO_2}} = 1329.5 \quad \textbf{(5-7.20)}$$

其中 a_c 爲固態碳之活性 (Activity)，假設 $a_c = 1$。

假設進料氣體的預熱溫度爲 T_i =1,000°F ，試求要在 20 atm 下進行反應，使絕熱反應平衡溫度爲 T_c = 2,200°F ，則反應產物中氧與甲烷之比值 O_2/CH_4 爲何？

數　據：

各成分之焓 (Btu / 1b-mole)

成分	1000°F	2200°F
CH_4	−13,492	8,427
H_2O	−90,546	−78,213
CO_2	−154,958	−139,009
CO	−38,528	−28,837
H_2	10,100	18,927
O_2	10,690	20,831

解：

由於 K_1 相當大，因此，可考慮在2200°F時，第一個反應 (5-7.13) 爲完全反應，表示達平衡時產物氣體中不含未反應的氧氣。此外，在平衡計算時不考慮 (5-7.19) 所示之反應。則高溫氣體反應平衡關係以莫耳分率表示爲：

1. 反應平衡

$$K_2 = \frac{P^2 x_1 x_4{}^3}{x_3 x_5} \quad \textbf{(5-7.21)}$$

$$K_3 = \frac{x_1 x_3}{x_2 x_4} \qquad \textbf{(5-7.22)}$$

2. 各成分莫耳分率之和爲 1：

$$x_1 + x_2 + x_3 + x_4 + x_5 = 1 \qquad \textbf{(5-7.23)}$$

3. 原子守恆平衡：

氧： $x_6 = (\frac{1}{2} x_1 + x_2 + \frac{1}{3} x_3) x_7$ **(5-7.24)**

氫： $4 = (2x_3 + 2x_4 + 4x_5) x_7$ **(5-7.25)**

碳： $1 = (x_1 + x_2 + x_5) x_7$ **(5-7.26)**

以上縯導中

$P =$ 總壓力

$x_1 =$ 平衡時混合氣體中 CO 之莫耳分率

$x_2 =$ 平衡時混合氣體中 CO_2 之莫耳分率

$x_3 =$ 平衡時混合氣體中 H_2O 之莫耳分率

$x_4 =$ 平衡時混合氣體中 H_2 之莫耳分率

$x_5 =$ 平衡時混合氣體中 CH_4 之莫耳分率

$$x_6 = \frac{\text{進料氣體中 } O_2 \text{ 之莫耳數}}{\text{進料氣體中 } CH_4 \text{ 之莫耳數}}$$

$$x_7 = \frac{\text{平衡時混合氣體中生成物之總莫耳數}}{\text{進料氣體中 } CH_4 \text{ 之莫耳數}}$$

4. 除了以上的物質平衡關係式，因爲反應是在絕熱下進行，沒有外來能量的進出，故可由生成物及反應物的焓平衡建立能量平衡關係式。

$$[H_{CH_4} + x_6 H_{O_2}]\Big|_{T=T_i} = x_7 [x_1 H_{CO} + x_2 H_{CO_2} + x_3 H_{H_2O} + x_4 H_{H_2} + x_5 H_{CH_4}]\Big|_{T=T_e} \qquad \textbf{(5-7.27)}$$

其中 $T_i =$ 反應物進料之預熱溫度

$T_e =$ 反應平衡溫度

在已知的反應系統溫度及壓力下，由以上方程式可求出 x_i , $i = 1, 2, \ldots\ldots, 7$，但這些結果必須同時滿足以下條件：

$$1 \geq x_i \geq 0 \qquad i = 1, 2, \cdots\cdots, 5$$

$$x_i \geq 0 \qquad i = 6, 7 \qquad \textbf{(5-7.28)}$$

TOP-DOWN 設計：

本問題事實上就是要解以下的聯立方程式 $\underline{f}(\underline{x})=0$ ：

$$f_1(\underline{x})=\frac{1}{2}x_1+x_2+\frac{1}{2}x_3-\frac{x_6}{x_7}=0 \quad \textbf{(5-7.29)}$$

$$f_2(\underline{x})=x_3+x_4+2x_5-\frac{2}{x_7}=0 \quad \textbf{(5-7.30)}$$

$$f_3(\underline{x})=x_1+x_2+x_5-\frac{2}{x_7}=0 \quad \textbf{(5-7.31)}$$

$$f_4(\underline{x})=-28837x_1-139009x_2-78213\frac{1}{2}x_3+18927x_4+8427x_5$$
$$+\frac{13492}{x_7}-10690\frac{x_6}{x_7}=0 \quad \textbf{(5-7.32)}$$

$$f_5(\underline{x})=x_1+x_2+x_3+x_4+x_5-1=0 \quad \textbf{(5-7.33)}$$

$$f_6(\underline{x})=\mathrm{P}^2x_1{x_4}^3-1.7897\times10^5x_3x_5=0 \quad \textbf{(5-7.34)}$$

$$f_7(\underline{x})=x_1x_3-2.6058x_2x_4=0 \quad \textbf{(5-7.35)}$$

重點摘要：

牛頓拉福森法的執行步驟可歸納如下：

1. 假設一組 $\underline{x}$ 的起始估計值 $\underline{x}_o=〔\ \mathrm{x}_{10},\ x_{20},\cdots\cdots x_{n0}〕^T$

2. 計算 Jacobian 矩陣 $\Phi(\underline{x})$

3. 解線性聯立方程式

$$\Phi(\underline{x}_k)\underline{h}_k=-\underline{f}(\underline{x}_k) \quad \textbf{(5-7.36)}$$

其中

$$\underline{f}(\underline{x}_k)=〔f_1(\underline{x}_k),\ f_2(\underline{x}_k),\cdots\cdots,\ f_n(\underline{x}_k)〕^T \quad \textbf{(5-7.37)}$$

4. 計算新的估計值

$$\underline{x}_{k+1}=\underline{x}_k+\underline{h}_k \quad \textbf{(5-7.38)}$$

5. 檢查是否已收斂

$$\left|\frac{h_{ik}}{x_{ik}}\right|<\varepsilon \quad ;\quad i=1,2,\cdots\cdots,n \quad \textbf{(5-7.39)}$$

若仍未收斂，則重複步驟 (2) 至 (5)。

6. 列印結果

主 程 式

輸入起始估計值
設定 ITMAX, ED, EZ
Call *NewtonRaphson(N, Z, DZ, P, Itmax, ED, EZ)*
If N <= 0
Then: 列印未收斂訊息 / Else: 列印計算結果
END

副程式 NewtonRaphson(N, Z, DZ, P, Itmax, ED, EZ)

II = 0			
	For I = 1 to N		
		Print Z(I)	
	@利用方程式(5-7.8)計算 $\Phi(\underline{x})=\{f_{ij}(\underline{x})\}$； $1\le i\le n$ ，$1\le j\le n$		
	Call Jacobian(N, AJ, P, Z)		
	@解線性聯立方程式 $\Phi(\underline{x}_k)\underline{h}_k=-\underline{f}(\underline{x}_k)$		
	Call Trixee(N, AJ, BB, DZ, ED, DET, ConvergentFlag)		
	If ConvergentFlag = 0		
	Then: Print Matrix Singular	Else: *Call DefineFunction(Z, DZ, P)*	
		For L1=1 to N	
			Z2(L1) = 0
			For KH = 1 to N
	N = -N		@求 $h_k=-\Phi^{-1}(\underline{x})\underline{f}(\underline{x}_k)$
			Z2(L1)=Z2(L1)+AJ(L1,KH)*DZ(KH)
	Exit Sub		@計算新估計值 $\underline{x}_{k+1}=\underline{x}_k+\underline{h}_k$
			Z2(L1)=Z(L1)-Z2(L1)
	For KS = 1 to N		
		@檢驗收斂性 $\lvert h_i/x_i\rvert<\varepsilon$ ； i = 1, 2, ……, n	
		If (Abs((Z2(KS) - Z(KS)) / Z(KS)) - EZ) <= 0	
		Then: ConvergentFlag = ConvergentFlag	Else: ConvergentFlag = ConvergentFlag +1

	If ConvergentFlag = 1	
	Then	Else
	Exit Do	II = II + 1
		For I = 1 To N
		Z(I) = Z2(I)
Loop While II < ITMAX		

副程式 **Jacobian(N, AJ, P, Z)**

@清除 Jacobian 矩陣
For I = 1 to N
For J = 1 to N
AJ(I,J) = 0
@利用方程式(5-7.8)建立 $\Phi(\underset{\sim}{x}) = \partial f_i / \partial x_j$
AJ(I,J) = $\partial f_i / \partial x_j$
RETURN

副程式 Trixee(N, AJ, BB, DZ, ED, DET, ConvergentFlag) 詳見本書第四章。

符號說明：

AJ(I , J)：Jacobian 矩陣；$AJ(I , J) = \dfrac{\partial f_i}{\partial x_j}$。

DZ(I)：　函數 $f_i(\underset{\sim}{x})$ 之殘值；$DZ(I) = f_i(\underset{\sim}{x}_k)$。

ED：　判定矩陣判別式爲零之下限，通常選用 1.E - 30。

EZ：　可接受之解的相對誤差，通常用 1.E – 4。

II：　迭代次數。

ITMAX：最大迭代次數，通常使用 10。

N：　方程式數目；無解或有錯誤時，副程式回應 N 爲負值。

Z：　最初估計值及解答。

Z2：　上依次迭代值。

程式列印：

```
'       ******************************************************
'       *  NEWTON      RAPHSON      METHOD   *
'       ******************************************************
```

```
Sub NewtonRaphsonMethod(Xpos, Ypos)

'     N         = NUMBER OF EQNS., RETURN WITH ERROR CODE
'     DZ        = RESIDUALS (DIMENSION N)
'      Z        = INITIAL ESTIMATE OF SOLUTION
'     ED        = VALUE OF DETERMINANT CONSIDERED TO BE ZERO FOR
'                   CHECKING    WHERE THE MATRIX IS SINGULAR,
'                   USUSALLY TAKEN AS 1E-30
'     AJ        = JACOBIAN MATRIX
'     EZ        = ACCEPTABLE RELATIVE ERROR IN Z
'     ITMAX = MAXIMUM LNUMBER OF ITERATIONS ALLOWED
'
'      ----------------
'         MAIN PROGRAM
'      ----------------
'
Dim Z(10), DZ(10)
'       == INITIAL GUESSES
P = 20: ' PRESSURE
N = 7:    ' NO. OF EQUATIONS
Cls
       Print "ENTER INITIAL GUESSES:"
       For I = 1 To N
              Print "Z("; I; ") = ";
              Z(I) = Val(InputBox("INPUT Z(I)", "Input Z(I)", Z(I), Xpos, Ypos))
              Print Z(I)
       Next I
       MsgBox ("Hit Any Key to Continue")
'
'      -----------------------
'         SET ERROR CRITERION
'      -----------------------
'
       Itmax = 10
       ED = 1E-30
       EZ = 0.0001
'
'      -----------------------
'         CALL NEWTON RAPHSON
'      -----------------------
'
Cls
Call NewtonRaphson(N, Z, DZ, P, Itmax, ED, EZ)
If N <= 0 Then
       Print String$(79, "*")
       N = -N
Else
       Print
```

```
        Print "SOLUTION OF THE SYSTEM: "
End If

For I = 1 To N
        Print Format(Z(I), "0.0000E+00");
        Print "      ";
        If (Int(I / 10) - I / 10) = 0 Then Print
Next I
Print
End Sub


Sub DefineFunction(Z, DZ, P)
'
'     -------------------
'         DEFINE FUNCTION
'     -------------------
'
DZ(1) = 0.5 * Z(1) + Z(2) + 0.5 * Z(3) - Z(6) / Z(7)
DZ(2) = Z(3) + Z(4) + 2 * Z(5) - 2 / Z(7)
DZ(3) = Z(1) + Z(2) + Z(5) - 1 / Z(7)
DZ(4) = -28837 * Z(1) - 139009! * Z(2) - 78213! * Z(3) + 18927 * Z(4) + 8427 * Z(5)
              + 13492 / Z(7) - 10690 * Z(6) / Z(7)
DZ(5) = Z(1) + Z(2) + Z(3) + Z(4) + Z(5) - 1
DZ(6) = P ^ 2 * Z(1) * Z(4) ^ 3 - 178370! * Z(3) * Z(5)
DZ(7) = Z(1) * Z(3) - 2.6058 * Z(2) * Z(4)
End Sub


Sub Jacobian(N, AJ, P, Z)
'
'     ------------------------
'         SUBROUTINE JACOBIAN
'     ------------------------
'
For I = 1 To N
        For J = 1 To N
                AJ(I, J) = 0
        Next J
Next I
AJ(1, 1) = 0.5
AJ(1, 2) = 1
AJ(1, 3) = 0.5
AJ(1, 6) = -1 / Z(7)
AJ(1, 7) = Z(6) / Z(7) ^ 2
'
AJ(2, 3) = 1
AJ(2, 4) = 1
```

```
AJ(2, 5) = 2
AJ(2, 7) = 2 / Z(7) ^ 2
'
AJ(3, 1) = 1
AJ(3, 2) = 1
AJ(3, 5) = 1
AJ(3, 7) = 1 / Z(7) ^ 2
'
AJ(4, 1) = -28837!
AJ(4, 2) = -139009!
AJ(4, 3) = -78213!
AJ(4, 4) = 18927
AJ(4, 5) = 8427
AJ(4, 6) = -10690 / Z(7)
AJ(4, 7) = (-13492 + 10690 * Z(6)) / Z(7) ^ 2
'
AJ(5, 1) = 1
AJ(5, 2) = 1
AJ(5, 3) = 1
AJ(5, 4) = 1
AJ(5, 5) = 1
'
AJ(6, 1) = P ^ 2 * Z(4) ^ 3
AJ(6, 3) = -178370! * Z(5)
AJ(6, 4) = 3! * P ^ 2 * Z(1) * Z(4) ^ 2
AJ(6, 5) = -178370! * Z(3)
'
AJ(7, 1) = Z(3)
AJ(7, 2) = -2.6058 * Z(4)
AJ(7, 3) = Z(1)
AJ(7, 4) = -2.6058 * Z(2)
End Sub


Sub NewtonRaphson(N, Z, DZ, P, Itmax, ED, EZ)
'
'     ---------------------
'        SUBROUTINE NEWTON
'     ---------------------
'
Dim AJ(10, 10), BB(10), Z2(10)
II = 0
Do
      Print "ITER="; II
      For ZM = 1 To N
            Print Format(Z(ZM), "0.00E+00");
            If ZM / 8 - Int(ZM / 8) = 0 Then Print
      Next ZM
```

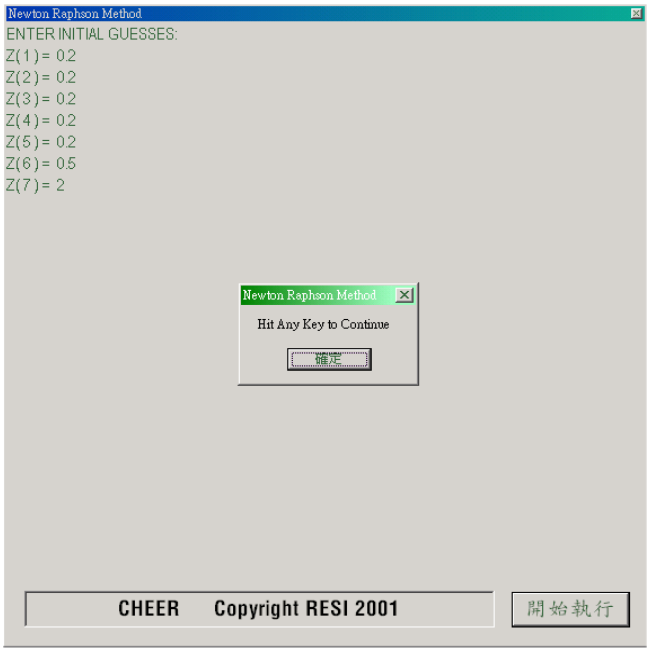
```
        Print

        Call Jacobian(N, AJ, P, Z)
        Call Trixee(N, AJ, BB, DZ, ED, DET, ConvergentFlag)

        If ConvergentFlag = 0 Then
            Print "** Matrix singular in nonlinear equations solver, DELTA = "; DET
            N = -N
            Exit Do
        End If
        Call DefineFunction(Z, DZ, P)
        For L1 = 1 To N
            Z2(L1) = 0
            For KH = 1 To N
                Z2(L1) = Z2(L1) + AJ(L1, KH) * DZ(KH)
            Next KH
            Z2(L1) = Z(L1) - Z2(L1)
        Next L1
        For KS = 1 To N
            If (Abs((Z2(KS) - Z(KS)) / Z(KS)) - EZ) <= 0 Then
                ConvergentFlag = ConvergentFlag
            Else
                ConvergentFlag = ConvergentFlag + 1
            End If
        Next KS
        If ConvergentFlag = 1 Then
            Exit Do
        Else
            II = II + 1
            For I = 1 To N
                Z(I) = Z2(I)
            Next I
        End If
    Loop While II < Itmax

    If ConvergentFlag > 1 Then
        Print "** Poor convergence in nonlinear system equation solver, ITER = "; II
        N = -N
    End If
    End Sub


    Sub Trixee(N, AJ, BB, DZ, ED, DET, KSG)
    '
    '     ----------------------------------------
    '        SUBROUTINE TRIXEE
    '     ----------------------------------------
    '     SIMULTANEOUS EQNS. SOLVER
```

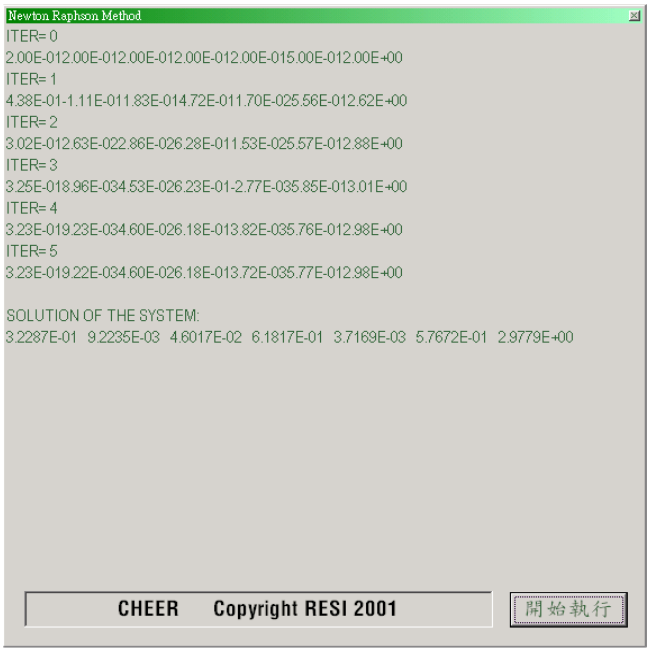
```
'
Dim MP(10), MQ(10)
DET = 1

'   -- DETERMINE PIVOTAL ELEMENT

For K = 1 To N
      PIV = 0
      For I = K To N
            For J = K To N
                  If (Abs(AJ(I, J)) > Abs(PIV)) Then
                        PIV = AJ(I, J)
                        MP(K) = I
                        MQ(K) = J
                  End If
            Next J
      Next I
      DET = DET * PIV
      If (Abs(DET) <= ED) Then
            KSG = 0
            Exit For
      End If
      KSG = 1

'   -- TRANSPOSITION OF THE PIVOTAL ROW WITH THE KTH ROW

      If (MP(K) <> K) Then
            For J = 1 To N
                  K1 = MP(K)
                  Call SWAP(AJ(K1, J), AJ(K, J))
            Next J
      End If

'   -- TRANSPOSITION OF THE PIVOTAL COLUMN WITH THE KTH COLUMN

      If (MQ(K) <> K) Then
            For I = 1 To N
                  K2 = MQ(K)
                  Call SWAP(AJ(I, K2), AJ(I, K))
            Next I
      End If

'   -- JORDAN TRANSFORMATION

      For J = 1 To N
            If J = K Then
                  BB(J) = 1 / PIV
                  DZ(J) = 1
```

```
            Else
                BB(J) = -AJ(K, J) / PIV
                DZ(J) = AJ(J, K)
            End If
            AJ(K, J) = 0
            AJ(J, K) = 0
        Next J
        For I = 1 To N
            For J = 1 To N
                AJ(I, J) = AJ(I, J) + DZ(I) * BB(J)
            Next J
        Next I
Next K
'
'   REARRANGEMENT OF THE MATRIX
'
If KSG = 1 Then
        For K = 1 To N
            K3 = N - K + 1
            If (MP(K3) <> K3) Then
                DET = -DET
                For I = 1 To N
                    K4 = MP(K3)
                    ZS = AJ(I, K4)
                    AJ(I, K4) = AJ(I, K3)
                    AJ(I, K3) = ZS
                Next I
            End If
            If MQ(K3) <> K3 Then
                DET = -DET
                For J = 1 To N
                    K5 = MQ(K3)
                    ZS = AJ(K5, J)
                    AJ(K5, J) = AJ(K3, J)
                    AJ(K3, J) = ZS
                Next J
            End If
        Next K
End If
End Sub


Sub SWAP(PAR1, PAR2)
Temp = PAR1
PAR1 = PAR2
PAR2 = Temp
End Sub
```

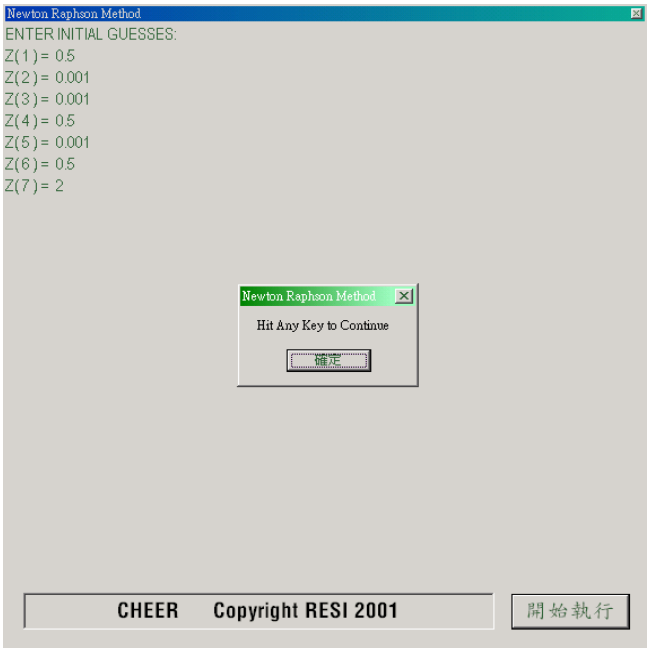
測試結果：

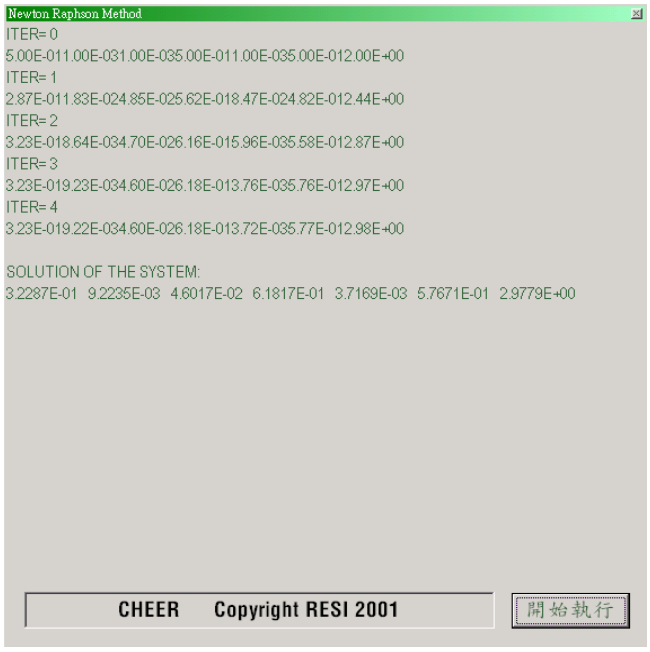
第一組


Newton Raphson Method
ENTER INITIAL GUESSES:
Z(1) = 0.2
Z(2) = 0.2
Z(3) = 0.2
Z(4) = 0.2
Z(5) = 0.2
Z(6) = 0.5
Z(7) = 2
Newton Raphson Method
Hit Any Key to Continue
確定
CHEER Copyright RESI 2001
開始執行


Newton Raphson Method
ITER= 0
2.00E-012.00E-012.00E-012.00E-012.00E-015.00E-012.00E+00
ITER= 1
4.38E-01-1.11E-011.83E-014.72E-011.70E-025.56E-012.62E+00
ITER= 2
3.02E-012.63E-022.86E-026.28E-011.53E-025.57E-012.88E+00
ITER= 3
3.25E-018.96E-034.53E-026.23E-01-2.77E-035.85E-013.01E+00
ITER= 4
3.23E-019.23E-034.60E-026.18E-013.82E-035.76E-012.98E+00
ITER= 5
3.23E-019.22E-034.60E-026.18E-013.72E-035.77E-012.98E+00
SOLUTION OF THE SYSTEM:
3.2287E-01 9.2235E-03 4.6017E-02 6.1817E-01 3.7169E-03 5.7672E-01 2.9779E+00
CHEER Copyright RESI 2001
開始執行

第二組


Newton Raphson Method
ENTER INITIAL GUESSES:
Z(1) = 0.5
Z(2) = 0.001
Z(3) = 0.001
Z(4) = 0.5
Z(5) = 0.001
Z(6) = 0.5
Z(7) = 2
Newton Raphson Method
Hit Any Key to Continue
確定
CHEER Copyright RESI 2001
開始執行


Newton Raphson Method
ITER= 0
5.00E-011.00E-031.00E-035.00E-011.00E-035.00E-012.00E+00
ITER= 1
2.87E-011.83E-024.85E-025.62E-018.47E-024.82E-012.44E+00
ITER= 2
3.23E-018.64E-034.70E-026.16E-015.96E-035.58E-012.87E+00
ITER= 3
3.23E-019.23E-034.60E-026.18E-013.76E-035.76E-012.97E+00
ITER= 4
3.23E-019.22E-034.60E-026.18E-013.72E-035.77E-012.98E+00
SOLUTION OF THE SYSTEM:
3.2287E-01 9.2235E-03 4.6017E-02 6.1817E-01 3.7169E-03 5.7671E-01 2.9779E+00
CHEER Copyright RESI 2001
開始執行

第九節　牛頓拉福森割線法

Visual Basic

牛頓拉福森法收斂速度相當快，但最大的缺點是使用者必須先縝導利用此方法所必備的 n^2 個偏微分表示式，以建立 Jacobian 矩陣。當方程式數目 n 很大時，這項工作將變得相當沉重。本書第六章將討論數值微分及積分，利用數值微分的技巧，就可以將微分工作交由計算機處理。牛頓拉福森割線法正是利用這種方法克服此困難，割線法的基本原理與牛頓拉福森法完全相同，唯一的差別是導函數不必先行縝導，而是直接利用原方程式作數值微分求得。

由於在工程科學上的應用，通常方程式的數目都相當多，而且函數可能相當複雜，不易求得偏微分表示式，因此，利用割線法將遠較牛頓拉福森法便捷。而就程式執行速度及收斂性質而論，二者則完全相同，因此，本書推薦讀者宜盡量使用割線法，而不採用牛頓拉福森法。

例題 5-7　反應平衡計算

利用牛頓拉福森割線法重作例 5-6。

解：

TOP-DOWN 設計：

根據微積分中微分的定義，函數 $f(x)$ 之導函數可寫成：

$$\frac{\partial f(x)}{\partial x} = \frac{f(x+\Delta x) - f(x-\Delta x)}{2\Delta x} \qquad \textbf{(5-8.12)}$$

故 Jacobian 矩陣中諸元素即可就 $\underline{x}_k$ 作微擾，仿上式求得 $f_{ij}(\underline{x}_k)$。

割線法的程式設計基本架構與牛頓拉福森法完全相同，唯一差別是 Jacobian 矩陣的建立方法不同，讀者可參照例 5-6 的 TOP-DOWN 設計。

副程式 **Jacobian(N, AJ, P, Z)**

For I = 1 to N
@求 $\underline{f}(\underline{x}_i - \Delta\underline{x}_i)$
XX = Z(I)
Z(I) = XX * (1 – EX)
X1 = Z(I)
Call **DefineFunction(Z, DZ, P)**
For J = 1 to N
BB(J) = DZ(J)
@求 $\underline{f}(\underline{x}_i + \Delta\underline{x}_i)$
Z(I) = XX * (1 + EX)
X2 = Z(I)
Call **DefineFunction(Z, DZ, P)**
For J = 1 to N
AJ(J,I) = (DZ(J) – BB(J)) / (X2 – X1)
RETURN

符號說明：(參考例 5-6)

EX：$\underline{x}_k$ 的微擾比率，$\Delta\underline{x}_k = (EX) * \underline{x}_k$

BB：存放 $\underline{f}(\underline{x}_i - \Delta\underline{x}_i)$

程式列印：

```
'       ********************************************************
'       *   NEWTON RAPHSON SECANT METHOD
'       ********************************************************

Sub NewtonRaphsonSecantMethod(Xpos, Ypos)

'    N        = NUMBER OF EQNS., RETURN WITH ERROR CODE
'    DZ       = RESIDUALS (DIMENSION N)
'     Z       = INITIAL ESTIMATE OF SOLUTION
'    ED       = VALUE OF DETERMINANT CONSIDERED TO BE ZERO FOR
'                  CHECKING WHERE THE MATRIX IS SINGULAR,
'                  USUSALLY TAKEN AS 1E-30
'    AJ       = JACOBIAN MATRIX
'    EX       = STEP LENGTH FOR PARTIAL DERIVATIVE ESTIMATIONS,
'                   FUNCTION EVALUATED AT (1+-EX)*Z
'    EZ       = ACCEPTABLE RELATIVE ERROR IN Z
'    ITMAX = MAXIMUM LNUMBER OF ITERATIONS ALLOWED
```

```
'
'       ----------------
'           MAIN PROGRAM
'       ----------------
'
Dim Z(10), DZ(10) As Double
'       == INITIAL GUESSES
P = 20: ' PRESSURE
N = 7:    ' NO. OF EQUATIONS
Cls
        Print "ENTER INITIAL GUESSES:"
        For I = 1 To N
                Print "Z("; I; ") = ";
                Z(I) = Val(InputBox("INPUT Z(I)", "Input Z(I)", Z(I), Xpos, Ypos))
                Print Z(I)
        Next I
        MsgBox ("Hit Any Key to Continue")
'
'       -----------------------
'           SET ERROR CRITERION
'       -----------------------
'
        Itmax = 10
        ED = 1E-30
        EX = 0.001
        EZ = 0.0001
'
'       ----------------------------
'           CALL NEWTON RAPHSON SECANT
'       ----------------------------
'
Cls
Call NewtonRaphsonSecant(N, Z, DZ, Itmax, P, ED, EZ, EX)
If N <= 0 Then
        Print String$(79, "*")
        N = -N
Else
        Print
        Print "SOLUTION OF THE SYSTEM: "
End If
For I = 1 To N
        Print Format(Z(I), "      0.00000E+00");
        If (Int(I / 8) - I / 8) = 0 Then Print
Next I
Print
End Sub
```

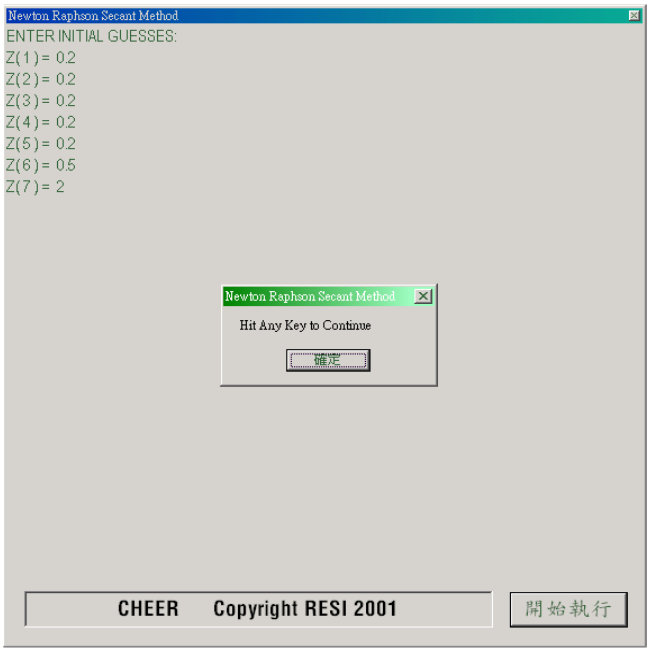
```
Sub DefineFunction(Z, DZ, P)
'
'     ------------------
'        DEFINE FUNCTION
'     ------------------
'
DZ(1) = 0.5 * Z(1) + Z(2) + 0.5 * Z(3) - Z(6) / Z(7)
DZ(2) = Z(3) + Z(4) + 2 * Z(5) - 2 / Z(7)
DZ(3) = Z(1) + Z(2) + Z(5) - 1 / Z(7)
DZ(4) = -28837 * Z(1) - 139009! * Z(2) - 78213! * Z(3) + 18927 * Z(4)
DZ(4) = DZ(4) + 8427 * Z(5) + 13492 / Z(7) - 10690 * Z(6) / Z(7)
DZ(5) = Z(1) + Z(2) + Z(3) + Z(4) + Z(5) - 1
DZ(6) = P ^ 2 * Z(1) * Z(4) ^ 3 - 178370! * Z(3) * Z(5)
DZ(7) = Z(1) * Z(3) - 2.6058 * Z(2) * Z(4)
End Sub


Sub NewtonRaphsonSecant(N, Z, DZ, Itmax, P, ED, EZ, EX)
'
'     -----------------------------------
'        SUBROUTINE NEWTON RAPHSON SECANT
'     -----------------------------------
'
Dim AJ(10, 10), BB(10), Z2(10)
II = 0
Do
      ConvergentFlag = 0
      Print "ITER="; II
      For ZM = 1 To N
            Print Format(Z(ZM), "    0.000E+00");
            If ZM / 8 - Int(ZM / 8) = 0 Then Print
      Next ZM
      Print
      Call Jacobian(N, AJ, BB, P, Z, DZ, EX)
      Call Trixee(N, AJ, BB, DZ, ED, DET, ConvergentFlag)
      If ConvergentFlag = 0 Then
            Print "** Matrix singular in nonlinear equations solver, DELTA = "; DET
            N = -N
            Exit Do
      End If
      Call DefineFunction(Z, DZ, P)
      For L1 = 1 To N
            Z2(L1) = 0
            For KH = 1 To N
```

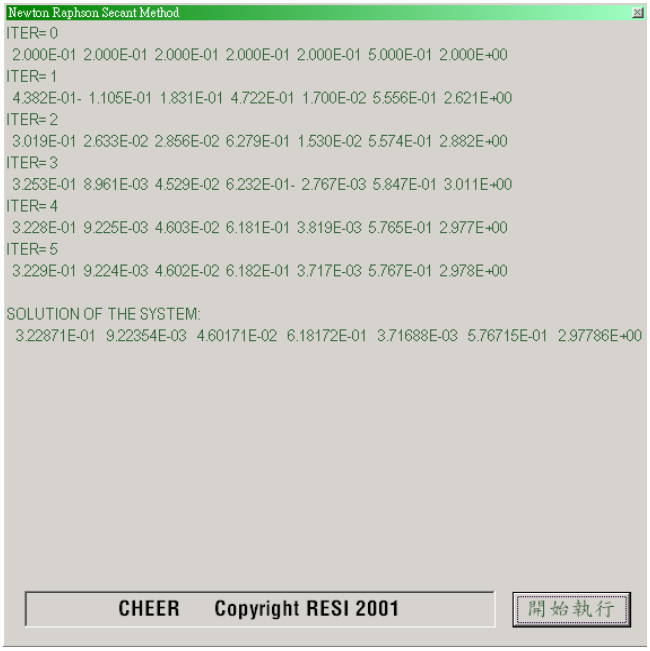
```
                Z2(L1) = Z2(L1) + AJ(L1, KH) * DZ(KH)
            Next KH
            Z2(L1) = Z(L1) - Z2(L1)
        Next L1
        For KS = 1 To N
            If (Abs((Z2(KS) - Z(KS)) / Z(KS)) - EZ) <= 0 Then
                ConvergentFlag = ConvergentFlag
            Else
                ConvergentFlag = ConvergentFlag + 1
            End If
        Next KS
        If ConvergentFlag = 1 Then
            Exit Do
        Else
            II = II + 1
            For I = 1 To N
                Z(I) = Z2(I)
            Next I
        End If
    Loop While II < Itmax
    If ConvergentFlag > 1 Then
        Print "** Poor convergence in nonlinear system equation solver, ITER = "; II
        N = -N
    End If
    End Sub


    Sub Jacobian(N, AJ, BB, P, Z, DZ, EX)
    '
    '     ----------------------
    '        SUBROUTINE JACOBIAN
    '     ----------------------
    '
    For I = 1 To N
        XX = Z(I)
        Z(I) = XX * (1 - EX)
        X1 = Z(I)
        Call DefineFunction(Z, DZ, P)
        For J = 1 To N
            BB(J) = DZ(J)
        Next J
        Z(I) = XX * (1 + EX)
        X2 = Z(I)
        Call DefineFunction(Z, DZ, P)
        For J = 1 To N
            AJ(J, I) = (DZ(J) - BB(J)) / (X2 - X1)
```

```
        Next J
        Z(I) = XX
    Next I
    End Sub


    Sub Trixee(N, AJ, BB, DZ, ED, DET, KSG)
    '
    '     ----------------------
    '        SUBROUTINE TRIXEE
    '     ----------------------
    '     SIMULTANEOUS EQNS. SOLVER
    '
    Dim MP(50), MQ(50) As Double
    DET = 1
    '    -- DETERMINE PIVOTAL ELEMENT
    For K = 1 To N
        PIV = 0
        For I = K To N
            For J = K To N
                If (Abs(AJ(I, J)) > Abs(PIV)) Then
                    PIV = AJ(I, J)
                    MP(K) = I
                    MQ(K) = J
                End If
            Next J
        Next I
        DET = DET * PIV
        If (Abs(DET) <= ED) Then
            KSG = 0
            Exit For
        End If
        KSG = 1
    '    -- TRANSPOSITION OF THE PIVOTAL ROW WITH THE KTH ROW
        If (MP(K) <> K) Then
            For J = 1 To N
                K1 = MP(K)
                Call SWAP(AJ(K1, J), AJ(K, J))
            Next J
        End If
    '    -- TRANSPOSITION OF THE PIVOTAL COLUMN WITH THE KTH COLUMN
        If (MQ(K) <> K) Then
            For I = 1 To N
                K2 = MQ(K)
                Call SWAP(AJ(I, K2), AJ(I, K))
            Next I
```

```
        End If
'   -- JORDAN TRANSFORMATION
        For J = 1 To N
            If J = K Then
                BB(J) = 1 / PIV
                DZ(J) = 1
            Else
                BB(J) = -AJ(K, J) / PIV
                DZ(J) = AJ(J, K)
            End If
            AJ(K, J) = 0
            AJ(J, K) = 0
        Next J
        For I = 1 To N
            For J = 1 To N
                AJ(I, J) = AJ(I, J) + DZ(I) * BB(J)
            Next J
        Next I
    Next K
'   REARRANGEMENT OF THE MATRIX
    If KSG = 1 Then
        For K = 1 To N
            K3 = N - K + 1
            If (MP(K3) <> K3) Then
                DET = -DET
                For I = 1 To N
                    K4 = MP(K3)
                    Call SWAP(AJ(I, K4), AJ(I, K3))
                Next I
            End If
            If MQ(K3) <> K3 Then
                DET = -DET
                For J = 1 To N
                    K5 = MQ(K3)
                    Call SWAP(AJ(K5, J), AJ(K3, J))
                Next J
            End If
        Next K
    End If
    End Sub


    Sub SWAP(PAR1, PAR2)
    Temp = PAR1
    PAR1 = PAR2
    PAR2 = Temp
    End Sub
```

執行結果：


Newton Raphson Secant Method
ENTER INITIAL GUESSES:
Z(1) = 0.2
Z(2) = 0.2
Z(3) = 0.2
Z(4) = 0.2
Z(5) = 0.2
Z(6) = 0.5
Z(7) = 2
Newton Raphson Secant Method
Hit Any Key to Continue
確定
CHEER Copyright RESI 2001
開始執行


Newton Raphson Secant Method
ITER= 0
2.000E-01 2.000E-01 2.000E-01 2.000E-01 2.000E-01 5.000E-01 2.000E+00
ITER= 1
4.382E-01- 1.105E-01 1.831E-01 4.722E-01 1.700E-02 5.556E-01 2.621E+00
ITER= 2
3.019E-01 2.633E-02 2.856E-02 6.279E-01 1.530E-02 5.574E-01 2.882E+00
ITER= 3
3.253E-01 8.961E-03 4.529E-02 6.232E-01- 2.767E-03 5.847E-01 3.011E+00
ITER= 4
3.228E-01 9.225E-03 4.603E-02 6.181E-01 3.819E-03 5.765E-01 2.977E+00
ITER= 5
3.229E-01 9.224E-03 4.602E-02 6.182E-01 3.717E-03 5.767E-01 2.978E+00
SOLUTION OF THE SYSTEM:
3.22871E-01 9.22354E-03 4.60171E-02 6.18172E-01 3.71688E-03 5.76715E-01 2.97786E+00
CHEER Copyright RESI 2001
開始執行

結果討論：

使用牛頓拉福森程式時，使用者需將所要解的方程組寫成 $DZ(J) = f_j(x_k)$ 的型式，置於 ***Sub DefineFunction(Z , DZ , P)***。並將偏微分表示式寫成 $AJ(I , J)= \partial f_i/\partial x_j$ 型式。使用牛頓拉福森割線法則只需將方程式表成 DZ(J)型式，其餘的工作由程式自行完成，因此，使用上較感便捷。

第十節　牛頓拉福森割線法與 Excel 應用

Visual Basic

牛頓拉福森割線法的基本原理與牛頓拉福森法完全相同，唯一的差別是導函數直接利用原方程式作數值微分求得。這種特性尤其適合利用 Excel 編寫程式。以下利用實例說明如何使用及編寫 Excel 版的牛頓拉福森割線法程式。

例題 5-8　牛頓拉福森割線法與 Excel

編寫 Excel 版的牛頓拉福森割線法程式，重解例 5-5。

$$u_1 + u_2 - 2.78u_3 = 0$$
$$58.9 + 40(u_1^{\,2} - u_2^{\,2}) = 0$$
$$-156.0 + 40(u_2^{\,2} + 0.3u_3^{\,2}) = 0$$

解：利用 Excel 編寫之程式如下所示：

	A	B	C	D	E	F	G	H	I	J
1	F1=	$u_1 + u_2 - 2.78u_3$				Jacobian				
2						$\frac{\partial f_i(x_j)}{\partial x_j} = \frac{f_i(x_j + \Delta x_j) - f(x_j - \Delta x_j)}{2\Delta x_j}$				
3	F2=	$58.9 + 40(u_1^{\,2} - u_2^{\,2})$								
4										
5	F3=	$-156.0 + 40(u_2^{\,2} + 0.3u_3^{\,2})$				X increment	Ex=	1.000E-03		
6										
7		Initial Trial	x + dx	x - dx		-F(x)				
8	U1	1.000E+00	1.001E+00	1.000E+00		7.800E-01				
9	U2	1.000E+00	1.001E+00	1.000E+00		-5.890E+01				
10	U3	1.000E+00	1.001E+00	1.000E+00		1.040E+02				
11										
12	x+dx	U1	U2	U3	x-dx	U1	U2	U3		

13	F1	-7.790E-01	-7.790E-01	-7.828E-01		-7.800E-01	-7.800E-01	-7.800E-01		
14	F2	5.898E+01	5.882E+01	5.890E+01		5.890E+01	5.890E+01	5.890E+01		
15	F3	-1.040E+02	-1.039E+02	-1.040E+02		-1.040E+02	-1.040E+02	-1.040E+02		
16										
17	Jacobian					Inverse Jacobian				h
18		1.000E+00	1.000E+00	-2.780E+00		8.876E-02	1.138E-02	1.028E-02		4.674E-01
19		8.004E+01	-8.004E+01	0.000E+00		8.876E-02	-1.109E-03	1.028E-02		1.203E+00
20		0.000E+00	8.004E+01	2.401E+01		-2.959E-01	3.696E-03	7.393E-03		3.204E-01
21										
22	#1	Trial Value	x + dx	x - dx		-F(x)				
23	U1	1.467E+00	1.469E+00	1.467E+00		3.153E-14				
24	U2	2.203E+00	2.205E+00	2.203E+00		4.915E+01				
25	U3	1.320E+00	1.322E+00	1.320E+00		-5.909E+01				
26										
27	x+dx	U1	U2	U3	x-dx	U1	U2	U3		
28	F1	1.467E-03	2.203E-03	-3.671E-03		-1.467E-06	-2.203E-06	3.671E-06		
29	F2	-4.897E+01	-4.953E+01	-4.915E+01		-4.915E+01	-4.914E+01	-4.915E+01		
30	F3	5.909E+01	5.948E+01	5.913E+01		5.909E+01	5.909E+01	5.909E+01		
31										
32	Jacobian					Inverse Jacobian				h
33		1.000E+00	1.000E+00	-2.780E+00		8.358E-02	7.803E-03	7.329E-03		-4.960E-02
34		1.174E+02	-1.763E+02	0.000E+00		5.567E-02	-4.740E-04	4.881E-03		-3.117E-01
35		0.000E+00	1.763E+02	3.170E+01		-3.096E-01	2.636E-03	4.392E-03		-1.300E-01
36										
37	#2	Trial Value	x + dx	x - dx		-F(x)				
38	U1	1.418E+00	1.419E+00	1.418E+00		3.508E-14				
39	U2	1.892E+00	1.893E+00	1.892E+00		3.813E+00				
40	U3	1.190E+00	1.192E+00	1.190E+00		-4.119E+00				
41										
42	x+dx	U1	U2	U3	x-dx	U1	U2	U3		
43	F1	1.418E-03	1.892E-03	-3.309E-03		-1.418E-06	-1.892E-06	3.309E-06		
44	F2	-3.652E+00	-4.099E+00	-3.813E+00		-3.813E+00	-3.813E+00	-3.813E+00		
45	F3	4.119E+00	4.405E+00	4.153E+00		4.119E+00	4.119E+00	4.119E+00		
46										
47	Jacobian					Inverse Jacobian				h
48		1.000E+00	1.000E+00	-2.780E+00		7.821E-02	8.123E-03	7.607E-03		-3.586E-04
49		1.135E+02	-1.514E+02	0.000E+00		5.862E-02	-5.166E-04	5.701E-03		-2.545E-02
50		0.000E+00	1.514E+02	2.858E+01		-3.105E-01	2.736E-03	4.787E-03		-9.285E-03
51										
52	#3	Trial Value	x + dx	x - dx		-F(x)				

53	U1	1.417E+00	1.419E+00	1.417E+00		2.665E-15				
54	U2	1.866E+00	1.868E+00	1.866E+00		2.781E-02				
55	U3	1.181E+00	1.182E+00	1.181E+00		-2.901E-02				
56										
57	x+dx	U1	U2	U3	x-dx	U1	U2	U3		
58	F1	1.417E-03	1.866E-03	-3.283E-03		-1.417E-06	-1.866E-06	3.283E-06		
59	F2	1.330E-01	-3.065E-01	-2.781E-02		-2.797E-02	-2.753E-02	-2.781E-02		
60	F3	2.901E-02	3.077E-01	6.250E-02		2.901E-02	2.873E-02	2.897E-02		
61										
62	Jacobian				Inverse Jacobian					h
63		1.000E+00	1.000E+00	-2.780E+00		7.764E-02	8.130E-03	7.610E-03		5.382E-06
64		1.134E+02	-1.494E+02	0.000E+00		5.897E-02	-5.198E-04	5.781E-03		-1.821E-04
65		0.000E+00	1.494E+02	2.836E+01		-3.106E-01	2.738E-03	4.817E-03		-6.358E-05

1. 首先定義 U1, U2, U3 的初步猜測值，並計算 $-F(x)$ 放置在 F8..F10 位置。

2. 依牛頓拉福森割線法需計算導函數矩陣 Jacobian，編寫程式時，將 $(x+\Delta x)$ 及 $(x-\Delta x)$ 計算放置在 C8..C10 及 D8..D10 位置，將 $f(x+\Delta x)$ 及 $f(x-\Delta x)$ 計算放置在 B13..D15 及 F13..H15 位置。

3. 計算導函數矩陣 Jacobian；$\dfrac{\partial f_i(x_j)}{\partial x_j}=\dfrac{f_i(x_j+\Delta x_j)-f(x_j-\Delta x_j)}{2\Delta x_j}$，放置在 B18..D20 位置。

4. 計算反矩陣，先選取 F18..H20 區域，輸入=Minverse(B18..D20)，再輸入 <Ctrl> <Shift> <Enter>，即可得到反矩陣值。Minverse(Array) 爲 Excel 內建的反矩陣運算函數。

5. 計算 h 陣列，先選取 J18..J20 區域，輸入=Mmult(F18..H20,F8..F10)，再輸入 <Ctrl> <Shift> <Enter>，即可得到 h 值。Mmult(Array 1, Array 2) 爲 Excel 內建的陣列乘法運算函數。

6. 計算所得 x 新的嘗試值爲 $x + h$。選取第 7 至 21 列區域，作複製，成第 22 至 36 列。修改 U1 的新嘗試值 B23 爲=B8+J18。複製 B23 到 B24 及 B25。即得到第一次計算結果。

7. 選取第 22 至 36 列區域，作複製，成第 37 至 51 列。即得到第二次計算結果。餘此類推。

由計算結果可以發現到第三次計算，計算誤差增量 h 即小到 10^{-4} 以下，顯示收斂效果相當良好。計算結果與例 5-6 一致。

Excel 程式編寫容易，但一般而言，若聯立方程式的數目較多時，由於導函數矩陣需一一輸入，較容易造成人爲疏忽，應特別注意。讀者可參考本例題做法，嘗試其他數值方法的程式編寫。

參考文獻

1. Coulson, J. M., J. F. Richardson, "Chemical Engineering", Vol.2, 3rd Ed., Pergamon Press, (1982).

2. Carnahan, B., H. A. Luther, and J. O. Wilkes, "Applied Numerical Methods" Wiley, (1970).

3. Traub, J. F., "Iterative Methods for the Solution of Equations", Prentice-Hall, (0964).

4. Henley, E. J., and E. M. Rosen, "Material and Energy Balance Computation", Wiley, (1969).

5. Box, M. J., Davies, D. and Swann, W. H., "Nonlinear Optimization Techniques", ICI Monograph No.5, Oliver & Boyd, London, (1969).

6. Murray, W. (Ed) "Numerical Methods for Unconstrained Optimization", Academic Press, London, (1972).

習　題

1. 試利用假位法及牛頓法求函數 $f(x) = \cos x - x$ 在 [0 , 1] 區間內之根，準確至小數以下第五位。

2. 試求超越函數 $x \tan x = 0.001$，$x > 0$ 之前四個根，準確至小數以下第六位。

3. 試求 $f(x) = x^5 + x^4 - 4x^3 - 3x^2 + 3x + 1 = 0$ 之根。

4. 試求 $f(x) = x^4 - 9x^3 - 2x^2 + 120x - 130 = 0$ 之根。

5. 試求 $f(x) = x^4 - 8x^3 + 39x^2 - 62x + 50 = 0$ 之根。

6. 解函數 $f(x) = 0$ 時，若假設其正確根爲 a，則利用牛頓法求解，基本上是一階準確度的數學方法，仿第四節的縯導法，也可以得到更高階的近似方法。

(a) 將 $f(x)$ 對 $x = x_n$ 作泰勒級數展開，得到

$$f(x_n+h)=f(x_n)+hf'(x_n)+\frac{h^2}{2}f''(x_n)+\cdots\cdots$$

(b) 令 x_n+h 接近於正確的根 a，則 $f(x_n+h)\cong 0$ ；捨去 h^3 以上諸項，得到

$$f(x_n)+h\left[f'(x_n)+\frac{h}{2}f''(x_n)\right]=0$$

(c) 將中括號內的 h 以一階牛頓法迭代式 $h=-f(x_n)/f'(x_n)$ 取代，試證可以得到牛頓法取至二階導函數之迭代式爲：

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

7. 利用問題 6 所得結果設計一程式，重作例 5-5。

8. 試利用連續取代法解例 5-7。

9. 試利用牛頓拉福森割線法重解設計問題 D-V。

10. 試解聯立方程組

$$\sin(x_1)-x_2+1.32=0$$
$$\cos(x_2)-x_1+0.85=0$$

11. 在一廠房內的高壓反應槽裝有一碟形安全閥，使氣體可由一截面積爲 0.07 m^2 之煙道排至大氣中。安全閥流動面積爲 4,000 mm^2，氣體可由此膨脹至整個煙道。若槽中氣體壓力爲 10 MN/m^2，溫度爲 500°K ，試求

(a) 在產生震波前之壓力及馬赫數，

(b) 在震波後之壓力及馬赫數。

假設氣體平均分子量爲 40 kg/k mole，比熱之比值 γ 爲 1.4，且氣體遵守理想氣體定律。[文獻 1，第 87 頁，例 4.4]。

由參考文獻 [1] 得聯立方程組

$$\frac{\omega^{-1.42}}{1-\omega^{0.29}}=9.25\times10^5A^2$$
$$u_s=141(1-\omega^{0.29})^{-0.5}$$
$$P_s=56.3\times10^6\omega(\omega^{-0.29}-1)$$
$$u_a{}^2-u_s{}^2=7P_s\hat{v}_s\left[1-(P_a/P_s)^{1/4}\right]$$

$$\frac{u_a}{\hat{v}_a} = 1224$$

$$\frac{u_a}{\hat{v}_s} = 85.7 / A$$

$$\hat{v}_a / \hat{v}_s = (P_a / P_s)^{-0.71}$$

$$P_a = 101.3 \times 10^3 N / m^2$$

$$M_a = 2.23(\omega^{-0.29} - 1)^{0.5}$$

上列方程組中計有八個未知數，分別爲

A：氣體膨脹流動之有效面積 (m^2)

u_a：煙道出口之流體流速 (m/s)

u_s：震波後之流體流速 (m/s)

P_s：震波後之流體壓力 (N/m^2)

$\hat{v}_a$：煙道出口處之流體體積 (m^3/kg)

$\hat{v}_s$：震波後之流體體積 (m^3/kg)

ω：震波前之壓力 / 反應器中之壓力

M_a：震波前之馬赫數

試利用牛頓拉福森割線法解此問題。