微分方程式
基本數值方法

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

Visual Basic

CHAPTER 8

常微分方程式—初值問題

(Ordinary Differential Equation – Initial Value Problem)

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

描述實際問題所得到的微分方程式
極少能利用解析方法求解，通常需利用數值方法求解

反應系統之模擬

在一個批次反應器中，進行以下的液相反應

$$A + D \xrightarrow[k_1]{} B$$
$$B + D \xrightarrow[k_2]{} C$$
$$D + D \xrightarrow[k_3]{} E$$

反應器中最初含有 0.2 mole/liter 的 A 及 0.4 mole/liter 的 D。

反應速率常數分別爲 $k_1 = 0.4$, $k_2 = 0.2$, $k_3 = 0.05$。

反應物及產物濃度與時間關係可利用下列質量平衡微分方程式描述之：

$$\frac{dy_1}{dt} = -0.4 y_1 y_4$$
$$\frac{dy_2}{dt} = -\frac{dy_1}{dt} - 0.2 y_2 y_4$$
$$\frac{dy_3}{dt} = 0.2 y_2 y_4$$
$$\frac{dy_4}{dt} = \frac{dy_1}{dt} - \frac{dy_3}{dt} - 0.05 y_4^2$$
$$\text{I.C.} \quad y_1(0) = 0.2,\ y_2(0) = 0,\ y_3(0) = 0,\ y_4(0) = 0.4$$

試模擬此反應系統，並求 0~200 秒間，反應器中各成分濃度隨時間之變化。

工程及科學領域中，許多問題都可以利用微分方程式來描述。微分方程式又可分成偏微分方程式及常微分方程式兩大類。方程式中若只含一個獨立變數的導函數，即稱爲常微分方程式；否則若含有一個以上獨立變數的導函數，即稱爲偏微分方程式。

常微分方程式依其邊界條件分類，又可分成初值問題 (IVP, Initial Value Problem) 及邊界值問題 (BVP, Boundary Value Problem)。例如

$$\text{IVP：}\ y'' = x^2 + y^2 \quad y(0)=1,\ y'(0)=1 \tag{8-0.1}$$

$$\text{BVP：}\ y'' = x^2 + y^2 \quad y(0)=1,\ y(1)=2 \tag{8-0.2}$$

其中 y' 表示 (dy/dx)。IVP 與 BVP 的差別在於 IVP 的邊界條件均定在同一 x 位置上，但 BVP 邊界值則定在兩個不同位置上。

由於描述實際問題所得到的微分方程式，極少能利用解析方法得到理論解，因此，通常需利用數值方法求解。常微分方程式中 IVP 及 BVP 的數值解法有相當大的差異，需要完全不同的處理方法，因此，本章首先討論初值問題 (IVP)，邊界值問題 (BVP) 則留待第九章及第十章詳加討論。

考慮 m 次常微分方程式

$$y^{(m)} = f(x, y, y', y'', \cdots, y^{(m-1)}) \tag{8-0.3}$$

其起始條件爲

$$\begin{aligned} y(x_0) &= y_0 \\ y'(x_0) &= y_0' \\ &\vdots \\ y^{(m-1)}(x_0) &= y_0^{(m-1)} \end{aligned}$$

其中 f 爲已知的函數，且 $y_0, y_0', y_0'', \cdots, y_0^{(m-1)}$ 均爲常數。方程式 (8-0.3) 通常可改寫成一組一次微分方程式。首先定義一組新的獨立變數 $y_1(x), y_2(x), \cdots, y_m(x)$ 爲

$$\begin{aligned} y_1 &= y \\ y_2 &= y' \\ &\vdots \\ y_m &= y^{(m-1)} \end{aligned} \tag{8-0.4}$$

則方程式 (8-0.3) 即可被轉換成

$$\begin{cases} y_1' = y_2 & \equiv f_1(x, y_1, y_2, \cdots, y_m) \\ y_2' = y_3 & \equiv f_2(x, y_1, y_2, \cdots, y_m) \\ \quad \vdots & \\ y_m' = f(x, y_1, y_2, \cdots, y_m) & \equiv f_m(x, y_1, y_2, \cdots, y_m) \end{cases} \quad \textbf{(8-0.5)}$$

起始條件爲

$$\begin{aligned} y_1(x_0) &= y_0 \\ y_2(x_0) &= y_0' \\ y_3(x_0) &= y_0'' \\ &\vdots \\ y_m(x_0) &= y_0^{(m-1)} \end{aligned}$$

將方程式 (8-0.5) 寫成向量符號，可以得到

$$\underline{y}'(x) = \underline{f}(x, \underline{y}) \qquad \underline{y}(0) = \underline{y}_0 \quad \textbf{(8-0.6)}$$

其中

$$\underline{y}(x) = \begin{bmatrix} y_1(x) \\ y_2(x) \\ \vdots \\ y_m(x) \end{bmatrix} \qquad \underline{f}(x, \underline{y}) = \begin{bmatrix} f_1(x, \underline{y}) \\ f_2(x, \underline{y}) \\ \vdots \\ f_m(x, \underline{y}) \end{bmatrix} \qquad \underline{y}_0 = \begin{bmatrix} y_0 \\ y_0' \\ \vdots \\ y_0^{(m-1)} \end{bmatrix}$$

方程式 (8-0.6) 可用於表示一個 m 次常微分方程式，一組次數和爲 m 的聯立微分方程式，或 m 個一次聯立微分方程式。一般而言，解常微分方程式的方法就是要解方程式 (8-0.6)。但爲了簡化分析起見，我們首先考慮最簡單的初值問題。

$$y' = f(x, y) \quad ; \quad x_0 \le x \le x_N \qquad \text{I.C.} \quad y(x_0) = y_0 \quad \textbf{(8-0.7)}$$

然後再推展至方程式 (8-0.6) 的一般情況。

第一節　歐以勒法

Visual Basic

歐以勒法 (Euler's Method) 是常微分方程式的數值解法中最簡單的一種，效率不高、準確度較差，但卻是說明數值解法運作原理的最佳工具。我們首先考慮方程式 (8-0.7) 所表示的一次常微分方程式，並將 $[x_0, x_N]$ 分割成 N 個等間距的小區間，使得

$$h = \frac{x_N - x_0}{N}$$
$$x_i = x_0 + i\,h \quad , \quad i = 0, 1, 2, \cdots, N \tag{8-1.1}$$

h 稱爲積分間距 (step size)。假設 $y(x)$ 爲方程式 (8-0.7) 的正確解，則利用泰勒定理將 $y(x)$ 對 x_i 展開，得到

$$y(x_{i+1}) = y(x_i) + (x_{i+1} - x_i)y'(x_i) + \frac{(x_{i+1} - x_i)^2}{2!}y''(\xi_i)$$
$$x_i \le \xi_i \le x_{i+1} \tag{8-1.2}$$

將方程式 (8-0.7) 代入上式，得到

$$y(x_{i+1}) = y(x_i) + hf(x_i, y(x_i)) + \frac{h^2}{2!}f'(\xi_i, y(\xi_i)) \tag{8-1.3}$$

若將上式中 h^2 項截去，即可獲得最簡單的數值方法。令 $u_i = y(x_i)$, $u_{i+1} = y(x_{i+1})$ ，則上式可改寫成

$$u_{i+1} = u_i + hf(x_i, u_i) \qquad i = 0, 1, \cdots, N-1$$
$$u_0 = y_0 \tag{8-1.4}$$

此方法即稱爲**歐以勒法**。

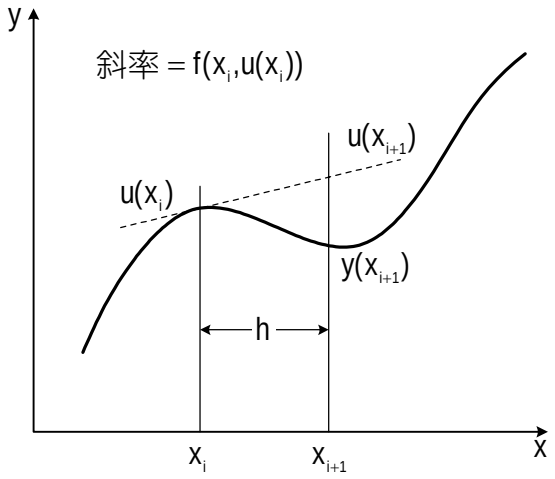


圖 8.1　歐以勒法

由方程式 (8-1.3) 可以得知，歐以勒法的截尾誤差爲

$$e_T = \frac{h^2}{2} y''(\xi_i) \equiv O(h^2)\,;\, x_i \le \xi_i \le x_{i+1} \qquad \textbf{(8-1.5)}$$

因此，我們可以預測 h 值愈小，則計算準確度愈高 (在機器準確度以內而言)。但是 h 值愈小，所需計算時間也愈多。因此，積分間距的適當選擇，在微分方程式的數值解法中，即成爲非常重要的步驟。

考慮最簡單的一次常微分方程式

$$\frac{dy}{dx} = \lambda y \qquad \text{I.C.} \quad y(0) = y_0 \qquad \textbf{(8-1.6)}$$

其中 λ 爲一常數。利用歐以勒法，則由方程式 (8-1.6) 可以得到

$$\begin{aligned} u_{i+1} &= u_i + \lambda h u_i \\ &= (1+\lambda h) u_i \\ &= (1+\lambda h)^2 u_{i-1} \\ &\vdots \\ &= (1+\lambda h)^{i+1} u_0 \end{aligned} \qquad \textbf{(8-1.7)}$$

而方程式 (8-1.6) 的理論解爲

$$y(x_{i+1}) = y_0 e^{\lambda x_{i+1}} = y_0 e^{(i+1)h\lambda} \qquad \textbf{(8-1.8)}$$

比較方程式 (8-1.7) 及 (8-1.8)，顯示歐以勒法是以 $(1+\lambda h)$ 取代 $\exp(\lambda h)$ 。由於任何數字都無法正確地表示成一機器數字，因此，利用 u_0 表示 y_0 亦可能存在捨入誤差，即 $e_0 = (y_0 - u_0)$ 可能不爲零。將方程式 (8-1.7) 中的 u_0 用 $y_0 - e_0$ 表示，得

$$u_{i+1} = (1+\lambda h)^{i+1} (y_0 - e_0) \qquad \textbf{(8-1.9)}$$

與方程式 (8-1.8) 比較，得到計算的總誤差爲

$$\begin{aligned} \mathrm{E}_{i+1} &= y(x_{i+1}) \\ &= \left[e^{(i+1)\lambda h} - (1+\lambda h)^{i+1} \right] y_0 + (1+\lambda h)^{i+1} e_0 \end{aligned} \qquad \textbf{(8-1.10)}$$

計算的總誤差是由兩部分所組成，前者是由於利用歐以勒法以 $(1+\lambda h)$ 取代 $\exp(\lambda h)$ 所產生的誤差，後者是由於最初的誤差 e_0 所造成的。很明顯地，如果以上的方程式中 $|1+\lambda h| > 1$，則 e_0 所造成的誤差會快速成長，變成 E_{i+1} 的主要部分。因此，要使歐以勒法維持絕對穩定的條件爲

$$-1 \le 1+\lambda h \le 1 \text{ 或 } -2 \le \lambda h \le 0 \qquad \textbf{(8-1.11)}$$

但此系統要維持不產生震盪的情況，則需 $0 \le 1+\lambda h \le 1$ 。否則 e_0 所造成的誤差會成爲一正一負的震盪情況。

例題 8-1　同分異構化反應

Cocks and Egger [1] 研究正亞丙基環丙胺 (m-propylidenecyclo-propylamine) 變成 5-乙基-1-梅笠草烯 (5-ethyl-1-pyroline) 的同分異構化反應，發現此反應爲一次反應，其反應速率常數爲

$$k_C = 3.50 \times 10^{13} \exp(-\frac{46429}{RT})$$

其中 $R = 1.9872$ cal/g mole °K ，T 爲反應溫度。

若此一反應在一恆溫管狀反應器中進行，且流動狀況趨近於塞式流動 (plug flow)，試計算正亞丙基環丙胺在反應器內的濃度分布。設反應溫度爲 700°K 。反應器滯留時間爲 200 秒及 50 秒兩種。

解：

一次反應的速率表示式爲 [2]

$$-r_A = k_C C_A$$

反應器的穩定態質量平衡，可以利用正亞丙基環丙胺在反應器軸向流動的變化量等於反應掉的量表示之，

$$-\frac{dn_A}{dV} = k_C C_A \qquad \text{I.C.} \qquad C_A(v=0) = C_{A0}$$

其中 n_A 爲反應物的質量流率，與濃度 C_A 及反應溶液的體積流量 u_T 關係爲

$$n_A = C_A u_T$$

定義反應器滯留時間爲 $\theta = V_T / u_T$ ，V_T 爲反應器總體積，Z 爲反應器軸向座標，L 爲反應器總長度， $y = C_A / C_{A0}$, $x = Z / L$ ，則質量平衡式可改寫成

$$\frac{dy}{dx} = -k_C \theta y \qquad ; \qquad y(0) = 1$$

$$= -3.50 \times 10^{13} \exp(-\frac{46429}{\mathrm{RT}}) \theta y$$

或 $$\frac{dy}{dx} = \begin{bmatrix} -22.365\, y & ; & \theta = 200 \sec. \\ -5.591\, y & ; & \theta = 50 \sec. \end{bmatrix}$$

利用方程式 (8-1.6) 的方式表示，λ 分別等於 −22.365 及 −5.591。根據定性分析，數值方法穩定條件爲 $-2 \le h\lambda \le 0$ ，誤差不會震盪的條件爲 $-1 \le h\lambda \le 0$ 。

穩定條件	$\theta = 200$ sec.	$\theta = 50$ sec.
不穩定	$h > 0.0894$	$h > 0.3577$
穩定，但誤差會震盪	$0.0447 < h < 0.0894$	$0.1789 < h < 0.3577$
穩定，且誤差不震盪	$0 < h < 0.0447$	$0 < h < 0.1789$

由於歐以勒法是所有數值方法中最簡單的一種，因此，本例題的程式設計留供讀者自行完成，並實際驗證上表的結果 (參見本章末問題 1)。

第二節　阮奇庫塔法

Visual Basic

使用最簡單的歐以勒法解微分方程式的時候，計算所使用的方程式 (8-1.4)，$u_{i+1} = u_i + hf(x_i, u_i)$ ，所代表的意義是要計算 u_{i+1} 時，可利用前一點的函數值 u_i 值，加上在前一點處的斜率 $f(x_i, u_i)$ 乘上 h 值來估算。這種方法很明顯的將使積分誤差逐漸累積而蔓延，使得結果的正確性較差，這也是可以預期的。

要改善這種方法，一個基本想法就是在利用前述方法計算 u_{i+1} 時，先在求函數 $f(x_i, u_i)$ 的近似值時加以適當修正。考慮在 x_i 至 x_{i+1} 中選取幾個特定點，分別計算它的函數值 K_j ，並加上適當的配重 ω_j ，用以計算及表示 $f(x_i, u_i)$ ，這種方法就稱爲**阮奇庫塔法** (Runge-Kutta Method)，其一般表示式可以寫成

$$u_{i+1} = u_i + \sum_{j=1}^{n} \omega_j \, h \, K_j \qquad \textbf{(8-2.1)}$$

其中 ω_j 爲配重因子，n 代表所選取 x_i 至 x_{i+1} 中的點數， K_j 爲 x_i 至 x_{i+1} 中的某一點的函數 f 值。 K_j 可以表示成

$$
\begin{aligned}
K(x,u) &= f(x,u) \qquad x_i < x < x_{i+1} \\
K_j &= f(x_i + c_j \mathrm{h}, u_i + \sum_{s=1}^{j-1} a_{js}\, h\, K_s) \\
c_1 &= 0 \\
u_{i+1}^{*} &= u_i + \sum_{j=1}^{n} \omega_j'\, h\, K_j \\
E_t &= u_{i+1} - u_{i+1}^{*}
\end{aligned}
\qquad \textbf{(8-2.2)}
$$

爲了便於表達起見，阮奇庫塔法的係數可以用下列的方塊表示法表示：

c_1	a_{11}	a_{12}	$\cdots$	a_{1n}
c_2	a_{21}	a_{22}	$\cdots$	a_{2n}
$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$
c_n	a_{n1}	a_{n2}	$\cdots$	a_{nn}
	ω_1	ω_2	$\cdots$	ω_n
	ω_1'	ω_2'	$\cdots$	ω_n'

當 $n=1$ 時，$\omega_1=1$，且 $K_1=f(x_i,u_i)$，方程式 (8-2.1) 可以被簡化成 $u_{i+1}=u_i+hf(x_i,u_i)$，即爲上一節所討論的歐以勒法。因此，歐以勒法也可以稱爲最低階的阮奇庫塔法。利用方塊表示法表示，即成爲

歐以勒法	
0	0
	1

阮奇庫塔法具有以下三點特色：

1. 阮奇庫塔法爲單間距積分法：要求得 u_{m+1}，我們只需前一點的資料 (x_m,u_m) 即可。
2. 使用阮奇庫塔法求解常微分方程式，不需計算 $f(x,y)$ 的導函數，只需使用 $f(x,y)$ 函數值即可進行積分運算。
3. p 階的阮奇庫塔法與泰勒級數比較，至 h^p 均相同；但 p 值隨不同方法而異。

二階阮奇庫塔法

較高階的阮奇庫塔法方程式中，參數 ω_j，c_j 及 a_{js} 基本上是利用泰勒級數展開與方

程式 (8-2.1) 比較來求得。例如，當 $n=2$ 時，首先將原微分方程式 $dy/dx=f(x,y)$ 的正確解利用泰勒級數展開，得到

$$y(x_{i+1})=y(x_i)+hf(x_i,y(x_i))+\frac{h^2}{2}f'(x_i,y(x_i))+O(h^3) \qquad \textbf{(8-2.3)}$$

由於 f 爲 x 及 y 的函數，而 y 又爲 x 的函數，因此，可以再將 $f'(x_i,y(x_i))$ 改寫成

$$f'(x_i,y(x_i))=\frac{df_i}{dx}=\frac{\partial f_i}{\partial x}+\frac{\partial f_i}{\partial y}\frac{dy}{dx}\bigg|_{x=x_i}=(f_x+f_y f)_i \qquad \textbf{(8-2.4)}$$

其中下註標 i 表示第 i 個點， f_x 表示 f 對 x 的導函數， f_y 表示 f 對 y 的導函數。將方程式 (8-2.4) 代入方程式 (8-2.3) 中，並將 h^3 項截掉，得到

$$u_{i+1}=u_i+hf_i+\frac{h^2}{2}(f_x+f_y f)_i \qquad \textbf{(8-2.5)}$$

在以上各方程式中，y 代表正確解，而 u 則代表數值方法的近似解。

其次，再將 x_i 至 x_{i+1} 中的某一點的函數值 K_j 對 x_i 位置作泰勒級數展開。由於 K 爲 x 及 u 的函數，因此，作泰勒級數展開時可以得到

$$\begin{aligned}K(x,u)&=f(x,u) \qquad x_i<x<x_{i+1}\\&=f(x_i,u_i)+(x-x_i)f_x(x_i,u_i)+(u-u_i)f_y(x_i,u_i)\end{aligned} \qquad \textbf{(8-2.6)}$$

由方程式 (8-2.2) K_j 的定義及方程式 (8-2.6)，可以將 K_j 寫成

$$K_1=f(x_i,u_i) \qquad \textbf{(8-2.7)}$$

$$\begin{aligned}K_2&=f(x_i+c_2h,u_i+a_{21}hK_1)\\&=f(x_i+c_2h,u_i+a_{21}hf)\\&=f_i+h(c_2f_x+a_{21}f_y f)_i\end{aligned} \qquad \textbf{(8-2.8)}$$

方程式 (8-2.8) 中， K_2 由第二行改寫成第三行時，是利用對 x_i 位置作泰勒級數展開。將以上二方程式代回方程式 (8-2.1)，整理後可以得到

$$u_{i+1}=u_i+(\omega_1+\omega_2)hf_i+\omega_2c_2h^2(f_x)_i+a_{21}\omega_2h^2(f_y f)_i \qquad \textbf{(8-2.9)}$$

上式與 u_{i+1} 的泰勒級數展開式 (8-2.5) 比較，得到以下聯立方程式：

$$\begin{cases}\omega_1+\omega_2=1\\ \omega_2c_2=\frac{1}{2}\\ a_{21}\omega_2=\frac{1}{2}\end{cases} \qquad \textbf{(8-2.10)}$$

由於二階阮奇庫塔法總共有四個待定係數 ω_1, ω_2, c_2 及 a_{21}，但只有上列三個方程式，自由度爲 1。因此，仍留有一個可自由設定的參數。就 $n=2$ 而言，常見的參數設定值如下：

c_2	ω_1	ω_2	a_{21}
$\frac{1}{2}$	0	1	$\frac{1}{2}$
1	$\frac{1}{2}$	$\frac{1}{2}$	1

亦即二階阮奇庫塔法可以寫成以下兩種型式：

$c_2 = \frac{1}{2}$ ；稱爲中點法或稱爲修正歐以勒法 (Modified Euler Method)

$$u_{i+1} = u_i + hf(x_i + \frac{h}{2}, u_i + \frac{1}{2}hf_i) \qquad \textbf{(8-2.11)}$$
$$u_0 = y_0$$

或用方塊法表示爲

修正歐以勒法		
0	0	
1/2	1/2	
	0	1

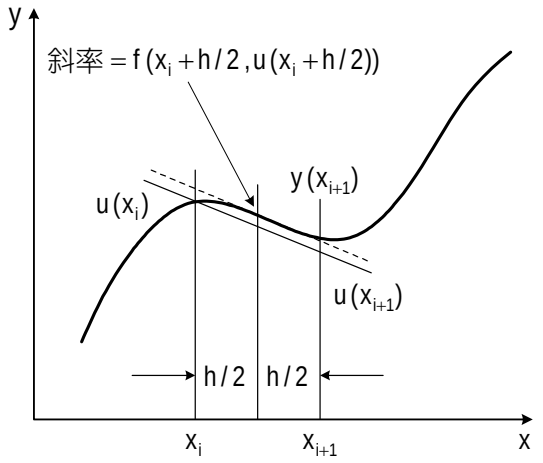


圖 **8.2**　修正歐以勒法

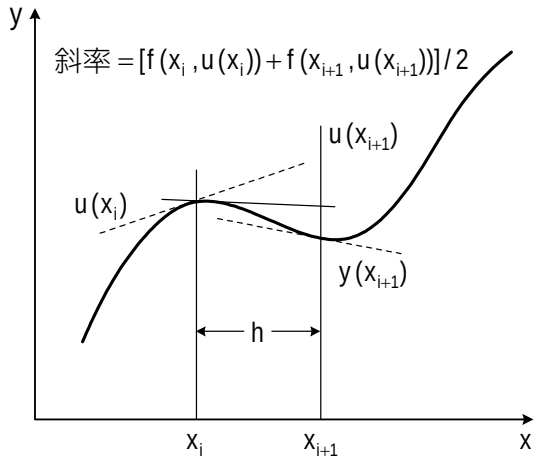


圖 **8.3**　改良式歐以勒法

$c_1 = 1$；稱爲端點平均值法或稱爲改良式歐以勒法 (Improved Euler Method)

$$u_{i+1} = u_i + \frac{h}{2}\left[f_i + f(x_i + h, u_i + hf_i) \right]$$
$$u_0 = y_0 \qquad \textbf{(8-2.12)}$$

或用方塊法表示爲

改良式歐以勒法		
0	0	
1	1	
	1/2	1/2

注意原微分方程式爲 $dy/dx = f(x, y)$，因此，f 所代表的物理意義是在 (x, y) 點處的曲線斜率。故方程式 (8-2.11) 代表利用 $[x_i, x_{i+1}]$ 的中點斜率進行計算，而方程式 (8-2.12) 則是利用 x_i 及 x_{i+1} 兩端點位置的斜率平均值進行計算。

方程式 (8-2.12) 通常也被稱爲預測修正法 (Predictor-Corrector Method)，並可以改寫成以下的型式：

$$\text{預測式：} u_{i+1}^{*} = u_i + hf(x_i, u_i)$$
$$\text{修正式：} u_{i+1} = u_i + \frac{h}{2}[f_i + f(x_{i+1}, u_{i+1}^{*})] \qquad \textbf{(8-2.13)}$$

阮奇庫塔法基本上是一種演導方法，仿以上的做法，可以推出各種不同的阮奇庫塔法。較高階的阮奇庫塔法由於計算費時，除了特殊需要，通常較少被使用。目前最常被使用的阮奇庫塔法有：

1. 三階庫塔法 (3^{rd} Order Kutta Method)；
2. 四階阮奇庫塔法 (4^{th} Order Runge-Kutta Method)；
3. 阮奇庫塔基爾法 (Runge-Kutta-Gill Method)，及
4. 阮奇庫塔摩森法 (Runge- Kutta- Merson Method)。

三階庫塔法 (3^{rd} Order Kutta Method)

$$u_{i+1} = u_i + \frac{h}{6}\left[K_1 + 4K_2 + K_3 \right]; i = 0, 1, 2, \cdots, N-1$$
$$K_1 = f(x_i, u_i) \qquad K_2 = f\left(x_i + \frac{h}{2}, u_i + \frac{h}{2}K_1\right) \qquad \textbf{(8-2.14)}$$

$$K_3 = f(x_{i+1}, u_i - hK_1 + 2hK_2)$$
$$u_0 = y_0$$

或用方塊法表示爲

三階庫塔法			
0	0		
1/2	1/2		
1	−1	2	
	1/6	2/3	1/6

四階阮奇庫塔法 (4th Order Runge-Kutta Method)

$$u_{i+1} = u_i + \frac{h}{6}[K_1 + 2K_2 + 2K_3 + K_4]; \quad i = 0, 1, 2, \cdots, N-1$$
$$K_1 = f(x_i, u_i) \qquad K_2 = f(x_i + \frac{h}{2}, u_i + \frac{h}{2}K_1) \qquad \textbf{(8-2.15)}$$
$$K_3 = f(x_i + \frac{h}{2}, u_i + \frac{h}{2}K_2) \qquad K_4 = f(x_{i+1}, u_i + hK_3)$$
$$u_0 = y_0$$

或用方塊法表示爲

四階阮奇庫塔法				
0	0			
1/2	1/2			
1/2	0	1/2		
1	0	0	1	
	1/6	1/3	1/3	1/6

四階阮奇庫塔基爾法 (Runge-Kutta-Gill Method)

用方塊法表示爲

四階阮奇庫塔基爾法				
0	0			
1/2	1/2			
1/2	α	β		
1	0	γ	δ	
	1/6	$\beta/3$	$\delta/3$	1/6

或寫成

$$u_{i+1} = u_i + \frac{h}{6}〔K_1 + 2\beta K_2 + 2\delta K_3 + K_4〕; \quad i = 0, 1, 2, \cdots, N-1$$

$$K_1 = f(x_i, u_i)$$

$$K_2 = f(x_i + \frac{h}{2}, u_i + \frac{h}{2}K_1)$$

$$K_3 = f(x_i + \frac{h}{2}, u_i + \alpha hK_1 + \beta hK_2) \qquad \textbf{(8-2.16)}$$

$$K_4 = f(x_1 + h, u_i + \gamma hK_2 + \delta hK_3)$$

$$\alpha = \frac{\sqrt{2}-1}{2}; \beta = \sqrt{2}\alpha; \gamma = -\frac{1}{\sqrt{2}}; \delta = 1-\gamma$$

$$u_0 = y_0$$

五階阮奇庫塔摩森法 (Runge-Kutta-Merson Method)

用方塊法表示爲

五階阮奇庫塔摩森法					
0	0				
1/3	1/3				
1/3	1/6	1/6			
1/2	1/8	0	3/8		
1	1/2	0	–3/2	2	
	1/6	0	0	2/3	1/6
	1/10	0	3/10	4/10	2/10

或寫成

$$u_{i+1} = u_i + \frac{h}{6}〔K_1 + 4K_4 + K_5〕; \quad i = 0, 1, 2, \cdots, N-1 \qquad \textbf{(8-2.17)}$$

$$u_{i+1}^{*} = u_i + \frac{1}{10}〔K_1 + 3K_3 + 4K_4 + K_5〕$$
$$K_1 = f(x_i, u_i)$$
$$K_2 = f(x_i + \frac{h}{3}, u_i + \frac{h}{3}K_1)$$
$$K_3 = f(x_i + \frac{h}{3}, u_i + \frac{h}{6}(K_1 + K_2))$$
$$K_4 = f(x_i + \frac{h}{2}, u_i + \frac{h}{8}(K_1 + 3K_3))$$
$$K_5 = f(x_i + h, u_i + \frac{h}{2}(K_1 - 3K_3 + 4K_4))$$
$$E_T = \frac{h}{15}(K_1 - \frac{9}{2}K_3 + 4K_4 - \frac{1}{2}K_5) \equiv u_{i+1} - u_{i+1}^{*}$$

六階阮奇庫塔布裘法 (Runge-Kutta-Butcher Method)

用方塊法表示爲

六階阮奇庫塔布裘法						
0	0					
1/4	1/4					
1/4	1/8	1/8				
1/2	0	–1/2	1			
3/4	3/16	0	0	9/16		
1	–3/7	2/7	12/7	–12/7	8/7	
	7/90	0	32/90	12/90	32/90	7/90

或寫成

$$u_{i+1} = u_i + \frac{h}{90}〔7K_1 + 32K_3 + 12K_4 + 32K_5 + 7K_6〕$$
$$K_1 = f(x_i, u_i)$$
$$K_2 = f(x_i + \frac{h}{4}, u_i + \frac{h}{4}K_1) \qquad \textbf{(8-2.18)}$$
$$K_3 = f(x_i + \frac{h}{4}, u_i + \frac{h}{8}(K_1 + K_2))$$
$$K_4 = f(x_i + \frac{h}{2}, u_i + \frac{h}{2}(-K_2 + 2K_3))$$

$$K_5 = f(x_i + \frac{3}{4}h, u_i + \frac{h}{16}(3K_1 + 9K_4))$$

$$K_6 = f(x_i + h, u_i + \frac{h}{7}(-3K_1 + 2K_2 + 12K_3 - 12K_4 + 8K_5)$$

阮奇庫塔費勃格法 (Runge-Kutta-Fehlberg Method)

RKF-1(2)			
0	0		
1/2	1/2		
1	1/256	255/256	
	1/256	255/256	0
	1/512	255/256	1/512

RKF-2(3)			
0	0		
1/4	1/4		
27/40	–189/800	729/800	
1	214/891	1/33	650/891
	214/891	1/33	650/891
	533/2106	0	800/1053

RKF-3(4)					
0	0				
2/7	2/7				
7/15	77/900	343/900			
35/36	805/1444	–77175/54872	97125/54872		
1	79/490	0	2175/3626	2166/9065	
	79/490	0	2175/3626	2166/9065	
	229/1470	0	1125/1813	13718/81585	1/18

RKF-4(5)						
0	0					
1/4	1/4					
3/8	3/32	9/32				
12/13	1932/2197	–7200/2197	7296/2197			
1	439/216	–8	3680/513	–845/4104		
1/2	–8/27	2	–3544/2565	1859/4104	–11/40	
	25/216	0	1408/2565	2197/4104	–1/5	
	16/135	0	6656/12825	28561/56430	–9/50	2/55

RKF-4(5) 可以寫成

$$u_{i+1} = u_i + (\frac{25}{216}K_1 + \frac{1408}{2565}K_3 + \frac{2197}{4104}K_4 - \frac{1}{5}K_5) \qquad \textbf{(8-2.19)}$$

其中 $K_1 = hf(x_i, u_i)$

$$K_2 = hf(x_i + \frac{h}{4}, u_i + \frac{1}{4}K_1)$$

$$K_3 = hf(x_i + \frac{3}{8}h, u_i + \frac{3}{32}K_1 + \frac{9}{32}K_2)$$

$$K_4 = hf(x_i + \frac{12}{13}h, u_i + \frac{1932}{2197}K_1 - \frac{7200}{2197}K_2 + \frac{7296}{2197}K_3)$$

$$K_5 = hf(x_i + h, u_i + \frac{439}{216}K_1 - 8K_2 + \frac{3680}{513}K_3 - \frac{845}{4104}K_4)$$

$$K_6 = hf(x_i + \frac{h}{2}, u_i - \frac{8}{27}K_1 + 2K_2 - \frac{3544}{2565}K_3 + \frac{1859}{4104}K_4 - \frac{11}{40}K_5)$$

$$E_T = h\frac{1}{360}K_1 - \frac{128}{4275}K_3 - \frac{2197}{75240}K_4 + \frac{1}{50}K_5 + \frac{2}{55}K_6$$

商業化程式名：RKF45, GERK，本書附有 Visual Basic 版 RKF 程式及利用 Excel 程式的說明，以便讓讀者更了解這種積分方法的應用。

一般而言，利用阮奇庫塔法編寫程式時，可以採用該方法本身估算誤差的功能，編寫成可以自動調節積分間距 (Step Size) 的程式，以加速積分速度。其策略可以利用以下的 Top-Down 設計圖表示之。

<table>
<tr><td colspan="3">首先設定一個起始積分間距 h</td></tr>
<tr><td rowspan="6"></td><td rowspan="5"></td><td>計算 u_{i+1} 及 u^*_{i+1}</td></tr>
<tr><td>估算誤差 $E_t = u_{i+1} - u^*_{i+1}$</td></tr>
<tr><td>推估新的積分間距爲 $h^* = ph(\varepsilon / E_t)^{1/(g+1)}$
其中 $0 < p < 1$，通常取 $p = 0.8$
$g = E_t$ 的階次
ε = 積分誤差容忍度</td></tr>
<tr><td>If $\varepsilon > E_t$
Then　　Else</td></tr>
<tr><td>淘汰 u_{i+1}　　接受 u_{i+1}</td></tr>
<tr><td>重複計算，直到 u_{i+1} 被接受</td></tr>
<tr><td colspan="3">重複計算到完成積分作業</td></tr>
</table>

編寫程式時，同時宜設定 h 之上限值及下限值，以免 ε 估計不當產生不良結果。積分誤差容忍度最好同時設定絕對誤差及相對誤差。

例題 8-2 烤火雞

聰明的李表哥看上了以上一大堆的方程式，覺得累極了，決定回家烤隻火雞吃。他按照食譜的說明將各種作料填滿了火雞胸，然後將重達 8 英鎊的火雞放入烤箱中。食譜說烤箱溫度需設定 350°F，烤 3 小時又 20 分鐘。

李表哥一想反正還有 3 小時多才享受這餐美食，不如算一下火雞的溫度變化及拿出烤箱時溫度爲何，因此，他整理出以下的數據，請你也幫他設計一程式，計算火雞體溫隨時間之變化情況。

火雞原始溫度　$T_0 = 60°\text{F}$

對流及輻射熱傳係數　$h = 2\text{Btu} / \text{hr} \cdot \text{ft}^2 \cdot °\text{F}$

火雞平均密度　$\rho = 70\,\text{lb}_\text{m} / \text{ft}^3$

熱傳導率　$k = 0.40\,\text{Btu} / \text{hr} \cdot \text{ft} \cdot °\text{F}$

熱容量　$C_p = 1.2\,\text{Btu} / \text{lb}_\text{m} \cdot °\text{F}$

體積　$V = 0.11\,\text{ft}^3$

表面積　$A = 1.6\,\text{ft}^2$

解：

由於此問題的 Biot 數 $= \dfrac{hL}{k} \cong 1$，因此，火雞體溫可認爲相當均勻。由熱量平衡可建立火雞體溫與時間關係式爲：

$$\rho C_p V \frac{dT}{dt} = hA(T_\infty - T)$$

定義無因次溫度爲 $\theta = \dfrac{T - T_\infty}{T_0 - T_\infty}$，則上式可改寫成

$$\frac{d\theta}{dt} = -\frac{hA}{\rho C_p V}\theta \equiv -\lambda\theta = -0.34632\,\theta \quad ; \quad \theta(t = \theta) = 1$$

此方程式的理論解爲

$$\theta = e^{-0.34632\,t} \; ; 0 \le t$$

利用四階阮奇庫塔法解此微分方程式，則計算公式 (8-2.14) 可寫成：

$$u_0 = 1$$

$$u_{i+1} = u_i + \frac{\Delta t}{6}〔K_1 + 2K_2 + 2K_3 + K_4〕;\quad i = 0, 1, 2, \cdots, N-1$$

$$K_1 = -\lambda u_i$$

$$K_2 = -\lambda(u_i + \frac{\Delta t}{2} K_1)$$

$$K_3 = -\lambda(u_i + \frac{\Delta t}{2} K_2)$$

$$K_4 = -\lambda(u_i + \Delta t K_3)$$

其中 $\lambda = 0.34632$ 。利用此方程式即可設計出簡單的程式。

TOP-DOWN 設計

主程式及阮奇庫塔副程式的 TOP-DOWN 設計較簡單，故從略，讀者請直接閱讀程式。

副程式 Sub RungeKutta(Ent%, X0, Y0, X, D, W, TolErr, IPrt$, H, H1)

@ 變數宣告及啓動程式做全間距積分	
T2 = H	
T = T2	
E5 = TolErr / 100	
Ent% = 0	
Call RungeKuttaModual(Ent%, S, X0, Y0, T, X, D, W, Y)	
Q1 = W	
	@第一半間距積分
	T = T2 / 2
	Ent% = 0
	Call RungeKuttaModual(Ent%, S, X0, Y0, T, X, D, W, Y)
	Z = W
	S = Z
	@第二半間距積分
	X=X0+T
	Call RungeKuttaModual(Ent%, S, X0, Y0, T, X, D, W, Y)
	@清除誤差比較旗幟，並準備誤差比較變數
	I8 = 0
	Q3 = (W - Q1) / 15
	DetErr = Abs(Q3) / T2

	@比較最適誤差範圍 TolErr / 100 ≤ DetErr ≤ TolErr		
	If DetErr >= TolErr		
	Then	Else	
	@未滿足收斂條件	@滿足收斂條件	
	ConvergentFlag = 0	ConvergentFlag = 1	
		Y0 = W + Q3	
		H = T2	
	T2 = T2 / 2	X0 = X0 + H	
		H1 = H	
		@檢驗是否誤差過小	
	Q1 = Z	If DetErr >= E5	
		Then	Else
		I8 = 1	H = 2 * H
		Exit Sub	Exit Sub
Do Loop While ConvergentFlag = 0			

符號說明

Upper： 積分上限

Lower： 積分下限

StepCtrl$：積分間距自動控制指標，

StepCtrl$＝”Y”間距控制，StepCtrl$＝”N”不作間距控制。

TolErr： 誤差容忍度

E5： TolErr/100

F： 積分函數，$\frac{dy}{dx} = F(x)$

H： 積分間距

H0： 原始設定積分間距

NP： 列印結果的列數

PT$： 列印指標

Q1： 全間距結果

S： 半間距結果

T： 暫存目前積分間距

T0： 暫存原積分間距

W：　　工作區

X0：　　積分起點

XF：　　積分中點

XPT：　　X 的列印間距

Y0：　　起始條件

Z：　　半間距積分結果

程式列印

```
' ***********************************
' RUNGE-KUTTA METHOD
' ***********************************
'
Private Sub RungeKuttaMethod(Xpos, Ypos)
'
'        USER'S SPACE for Data Entry
'
Cls
Print "***              Runge-Kutta Method              ***"
Print

Print ">ENTER LIMITS OF INTERGRATION"
Print "Lower Limit of X = ";
Lower = Val(InputBox("    Lower Limit,    Lower = ", "Low Limit of X", Lower, Xpos, Ypos))
Print Lower

Print "Upper Limit of X = ";
Upper = Val(InputBox("    Upper Limit,    Upper = ", "Upper Limit XS", Upper, Xpos, Ypos))
Print Upper

Print "> Initial Step Size = ";
H0 = Val(InputBox("> Initial Step Size = ", "Step Size", (Upper - Lower) / 100, Xpos, Ypos))
Hsav = H0
Print H0

Print "> Error Tolerance on Y = ";
TolErr = Val(InputBox("> ERROR TOLERANCE ON Y = ", "Error Tolerance", H0 * 0.00001, Xpos,
Ypos))
Print TolErr

Print "> Need Step Size Control <Y/N> = ";
StepCtrl$ = InputBox(">STEPSIZE CONTRAL ? (Y/N) ", "STEP CONTROL", , Xpos, Ypos)
```

```
Print StepCtrl$

Print "> Need Detail Printout <Y/N> = ";
PNT$ = UCase(InputBox(">Need Detail Printout ? (Y/N) ", "PRINT CONTROL", "N", Xpos, Ypos))
Print PNT$

Print
NP = Val(InputBox("> NUMBER OF PRINTING INTERVALS = (<50) ", "NP", NP, Xpos, Ypos))
XPT = (Upper - Lower) / NP
Print

' --INITIAL CONDITIONS
Print "> Enter Initial Conditions:"
Print "    Y0 = ";
Y0 = Val(InputBox(" Initial Condition = ", "DATA ENTRY", , Xpos, Ypos))
Print Y0

MsgBox ("Ready to Start")

Call RungeKuttaCaller(X0, Y0, TolErr, PNT$, H0, Lower, Upper, NP, StepCtrl$)

End Sub


Private Sub DeriveFunction(F, X, Y)
'
' USER DEFINED FUNCTIONS
'
F = -0.34632 * Y
End Sub


Public Sub RungeKuttaCaller(X0, Y0, TolErr, PNT$, H0, Lower, Upper, NP, StepCtrl$)
'
' Runge-Kutta Integration Caller
'
' This subroutine controls the RK-4 Algorithm.
'
' X0              = Independent Variable for Initial Condition
' Y0              = Initial Condition at X0
' TolErr        =Local Error Tolerance.
' PNT$           = Print Control, Y for detail printing
' H0              =Initial Step Size.
' Lower          =Lower Bound
' Upper          =Upper Bound
' NP              = Total Number of Printing Intervals
```

```
' StepCtrl$ =Auto. Step Size Control ID.
'
' F                =Function
'
Cls
'
' Print Report Tittle
'
PrtEnt% = 1
Call ReportRK4(PrtEnt%, X0, Y0)

GoBackPrt = 1
PT$ = PNT$
XPT = (Upper - Lower) / NP
X0 = Lower
H = H0
XF = X0 + XPT

'
' Print Initial Condition
'
PrtEnt% = 2
Call ReportRK4(PrtEnt%, X0, Y0)

Do
        If PT$ = "Y" And GoBackPrt = 1 Then
                PrtEnt% = 2
                Call ReportRK4(PrtEnt%, X0, Y0)
                PT$ = PNT$
        End If

        If X0 >= Upper Then Exit Do
        Call RungeKutta(X0, Y0, TolErr, PT$, H, H1, StepCtrl$)

' ==CHECK FOR END INTERVAL==

        If X0 = XF Then
                XF = X0 + XPT
                PT$ = "Y"
        End If

        If X0 + H > XF Then
                H = XF - X0
        End If

        If (X0 - H1) <= XF And PT$ = "Y" Then
```

```
            GoBackPrt = 1
        Else
            GoBackPrt = 0
        End If

    Loop While X0 <= Upper

    Print "=== END EXECUTION ==="
    End Sub


    Public Sub ReportRK4(PrtEnt%, X0, Y0)
    '
    ' Report Generator
    '
    Select Case PrtEnt%

    Case 1
        Print
        Print "SOLUTION"
        Print "**************************************************"
        Print "              X                          ";
        Print "              Y                          ";
        Print "        Temperature (F) "

    Case 2
        T0 = 60
        Tinfinite = 350
        Temperature = Y0 * (T0 - Tinfinite) + Tinfinite
        Print Format(X0, "      0.000000E+00");
        Print Format(Y0, "      0.000000E+00");
        Print Format(Temperature, "      0.0000E+00")

    End Select
    End Sub


    Public Sub RungeKutta(X0, Y0, TolErr, IPrt$, H, H1, StepCtrl$)
    '
    ' Runge-Kutta Controller
    ' Internal Subroutine Called by RungeKuttaCaller
    '
    T2 = H
    T = T2
    E5 = TolErr / 100
    Ent% = 0
```

```
Call RungeKuttaModual(Ent%, S, X0, Y0, T, X, D, W)
Q1 = W

Do
    T = T2 / 2
    Ent% = 0
    Call RungeKuttaModual(Ent%, S, X0, Y0, T, X, D, W)

    Z = W
    S = Z
    X = X0 + T
    Call RungeKuttaModual(Ent%, S, X0, Y0, T, X, D, W)

    I8 = 0
    Q3 = (W - Q1) / 15
    DetErr = Abs(Q3) / T2

    If DetErr >= TolErr Then
'
'       ==CONVERGENCE NOT ATTAINED==
'
        ConvergentFlag = 0

        If StepCtrl$ = "N" Then
            Print
            Print "    **ERROR** "
            Print "       CONVERGENCE NOT ATTAINED "
            Print "       WITHIN REQUIRED TOLERANCE "
            Print "       FOR GIVEN STEP-SIZE"
            Print "       H="; H; "    AT X="; X0
        Else
            T2 = T2 / 2
            Q1 = Z
        End If
    Else
'
'       ==CONVERGENCE ATTAINED==
'
        ConvergentFlag = 1
        Y0 = W + Q3
        H = T2
        X0 = X0 + H
        H1 = H
'
' ==CHECK FOR STEP-SIZE CONTROL==
'
```

```
            If StepCtrl$ = "N" Then Exit Sub

            If DetErr >= E5 Then
                I8 = 1
                Exit Sub
            Else
                H = 2 * H
                Exit Sub
            End If

        End If
    Loop While ConvergentFlag = 0
    End Sub


    Public Sub RungeKuttaModual(Ent%, S, X0, Y0, T, X, D, W)
    '
    ' RUNGE-KUTTA MODUAL
    '
    ' ==INITIALIZE VARIABLES ==
    '
    If Ent% = 0 Then
        X = X0
        S = Y0
        Ent% = 1
    End If
    '
    ' ==LOOP ENTRY
    '
    Y = S
    W = S
    '
    ' ==COMPUTE W VALUE==
    '
    ' --1st PASS
    '
    Call DeriveFunction(F, X, Y)
    D = T * F
    W = W + D / 6
    Y = S + D / 2
    X = X + T / 2
    '
    ' --2nd PASS
    '
    Call DeriveFunction(F, X, Y)
    D = T * F
```

```
    W = W + D / 3
    Y = S + D / 2
    '
    ' --3rd PASS
    '
    Call DeriveFunction(F, X, Y)
    D = T * F
    Y = S + D
    W = W + D / 3
    X = X + T / 2
    '
    ' --4th PASS
    '
    Call DeriveFunction(F, X, Y)
    D = T * F
    W = W + D / 6
    '
    '    Program Developed by Dr. Ron Hsin Chang
    '
    End Sub
```

副程式使用方法

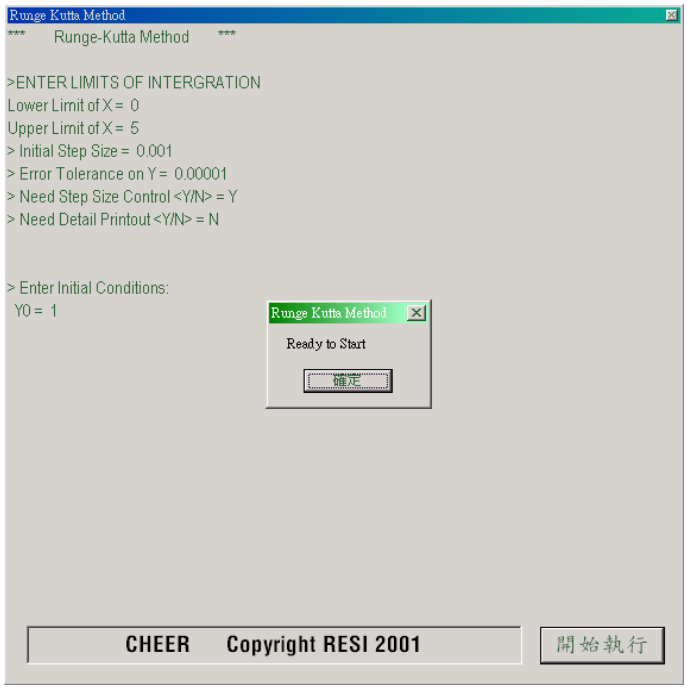
1. 副程式 **DeriveFunction(F, X, Y)**
   將微分方程式寫成以下型式；其中 F(i)=dY(i)/dx
   SUB DeriveFunction(F, X, Y)
   F = -0.34632 * Y
   End Sub

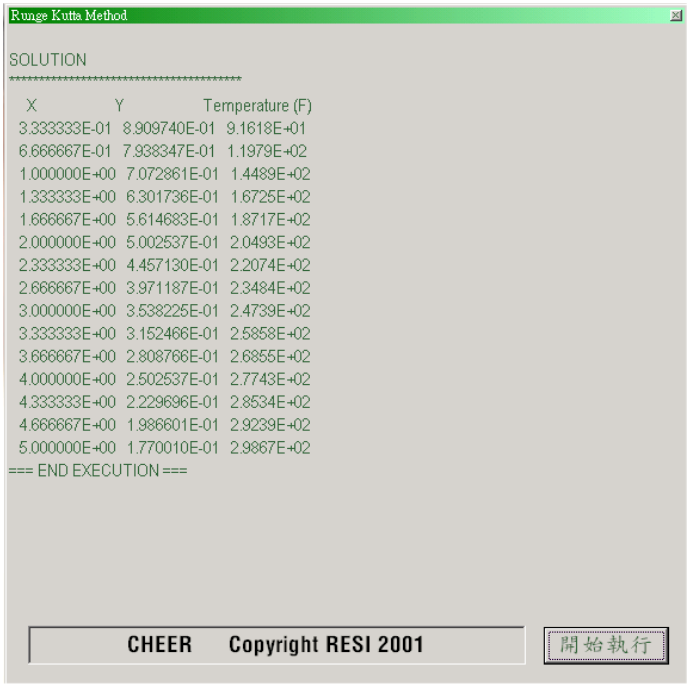
2. 副程式 **RungeKuttaCaller(X0,Y0, TolErr, PNT$, H0, Upper, Lower, NP, StepCtrl$)**
   爲阮奇庫塔法的呼叫控制副程式，使用者只要輸入 X0, Y0, TolErr, PNT$, H0, Upper, Lower, NP, StepCtrl$，然後執行 RungeKuttaCaller 副程式，其餘工作即由程式自行處理。

3. 副程式 **RungeKutta(X0, Y0, TolErr, H, H1, StepCtrl$)**
   爲阮奇庫塔法的工作控制副程式，用於控制計算流程及收斂檢驗。

4. 副程式 **RungeKuttaModual(Neqn%, S, X0, Y0, T, X, Y, D, W)**
   爲阮奇庫塔法的工作模組副程式，用於計算 K_1 , K_2 , K_3 , K_4 。

執行結果


Runge Kutta Method
*** Runge-Kutta Method ***
>ENTER LIMITS OF INTERGRATION
Lower Limit of X = 0
Upper Limit of X = 5
> Initial Step Size = 0.001
> Error Tolerance on Y = 0.00001
> Need Step Size Control <Y/N> = Y
> Need Detail Printout <Y/N> = N
> Enter Initial Conditions:
Y0 = 1
Runge Kutta Method
Ready to Start
CHEER Copyright RESI 2001
開始執行


Runge Kutta Method
SOLUTION
X Y Temperature (F)
3.333333E-01 8.909740E-01 9.1618E+01
6.666667E-01 7.938347E-01 1.1979E+02
1.000000E+00 7.072861E-01 1.4489E+02
1.333333E+00 6.301736E-01 1.6725E+02
1.666667E+00 5.614683E-01 1.8717E+02
2.000000E+00 5.002537E-01 2.0493E+02
2.333333E+00 4.457130E-01 2.2074E+02
2.666667E+00 3.971187E-01 2.3484E+02
3.000000E+00 3.538225E-01 2.4739E+02
3.333333E+00 3.152466E-01 2.5858E+02
3.666667E+00 2.808766E-01 2.6855E+02
4.000000E+00 2.502537E-01 2.7743E+02
4.333333E+00 2.229696E-01 2.8534E+02
4.666667E+00 1.986601E-01 2.9239E+02
5.000000E+00 1.770010E-01 2.9867E+02
=== END EXECUTION ===
CHEER Copyright RESI 2001
開始執行

烤 3 小時 20 分 ($t = 3.333$ hr) 後，火雞體溫爲

$$\frac{T - T_\infty}{T_0 - T_\infty} = 0.31525 \qquad ; \qquad T = 258.6^\circ \text{F}(125.9^\circ \text{C})$$

第三節　聯立微分方程式

Visual Basic

高階微分方程式或聯立微分方程式所使用的數值解法與一次微分方程式所使用的方法完全相同。例如要解微分方程式 (8-0.1) 時，可先將原方程式 $y'' = x^2 + y^2$　I.C. $y(0) = 1, y'(0) = 1$轉換成聯立方程式 (8-0.6) 的型式

$$\begin{cases} y_1' = y_2 \\ y_2' = x^2 + y_1^2 \end{cases} \quad \text{I.C.} \quad y_1(0) = 1, y_2(0) = 1 \qquad \textbf{(8-0.6)}$$

四階阮奇庫塔法

將方程式 (8-2.14) 的**四階阮奇庫塔法**，延伸成可以解如方程式 (8-0.6) 的聯立微分方程式，則得到

$$\underline{u}_{i+1} = \underline{u}_i + \frac{h}{6}\left[\underline{K}_1 + 2\underline{K}_2 + 2\underline{K}_3 + \underline{K}_4\right] \qquad \textbf{(8-3.1)}$$

其中　$\underline{u}_i = \left[u_i^{(1)}, u_i^{(2)}, \cdots, u_i^{(m)}\right]^T$;　$i = 0, 1, 2, \cdots, N-1$

$\underline{K}_i = \left[K_i^{(1)}, K_i^{(2)}, \cdots, K_i^{(m)}\right]^T$

$K_1^{(j)} = f_j(x_i, \underline{u}_i)$;　$j = 1, 2, \cdots, \text{m}$

$K_2^{(j)} = f_j(x_i + \frac{h}{2}, \underline{u}_i + \frac{h}{2}\underline{K}_1)$

$K_3^{(j)} = f_j(x_i + \frac{h}{2}, \underline{u}_i + \frac{h}{2}\underline{K}_2)$

$K_4^{(j)} = f_j(x_{i+1}, \underline{u}_i + h\underline{K}_3)$

$\underline{u}_0 = \underline{y}_0$

阮奇庫塔基爾法

用於解聯立微分方程式的阮奇庫塔基爾法爲

$$\underline{u}_{i+1}=\underline{u}_i+\frac{h}{6}[\underline{K}_1+2\beta\underline{K}_2+2\delta\underline{K}_3+\underline{K}_4];\quad i=0,1,2,\cdots,N-1 \qquad \textbf{(8-3.2)}$$

其中 $\underline{u}_i=[u_i^{(1)},u_i^{(2)},\cdots,u_i^{(m)}]^T$

$\underline{K}_i=[K_i^{(1)},K_i^{(2)},\cdots,K_i^{(m)}]^T$

$K_1^{(j)}=f_j(x_i,\underline{u}_i);\quad j=1,2,\cdots,\mathrm{m}$

$K_2^{(j)}=f_j(x_i+\frac{h}{2},\underline{u}_i+\frac{h}{2}\underline{K}_i)$

$K_3^{(j)}=f_j(x_i+\frac{h}{2},\underline{u}_i+\alpha h\underline{K}_1+\beta h\underline{K}_2)$

$K_4^{(j)}=f_j(x_i+h,\underline{u}_i+\gamma h\underline{K}_2+\delta h\underline{K}_3)$

$\alpha=\frac{\sqrt{2}-1}{2}$ $\beta=\sqrt{2}\alpha$

$\gamma=-\frac{1}{\sqrt{2}}$ $\delta=1-\gamma$

$\underline{u}_0=\underline{y}_0$

阮奇庫塔摩森法

用於解聯立微分方程式的阮奇庫塔摩森法爲

$$\underline{u}_{i+1}=\underline{u}_i+\frac{h}{6}[\underline{K}_1+4\underline{K}_4+\underline{K}_5];\quad i=0,1,2,\cdots,N-1 \qquad \textbf{(8-3.3)}$$

其中 $\underline{u}_i=[u_i^{(1)},u_i^{(2)},\cdots,u_i^{(m)}]^T$

$\underline{K}_i=[K_i^{(1)},K_i^{(2)},\cdots,K_i^{(m)}]^T$

$K_1^{(j)}=f_j(x_i,\underline{u}_i);\quad j=1,2,\cdots,\mathrm{m}$

$$K_2^{(j)} = f_i(x_i + \frac{h}{3}, \underline{u}_i + \frac{h}{3}\underline{K}_1)$$

$$K_3^{(j)} = f_j(x_i + \frac{h}{3}, \underline{u}_i + \frac{h}{6}(\underline{K}_1 + \underline{K}_2))$$

$$K_4^{(j)} = f_j(x_i + \frac{h}{2}, \underline{u}_i + \frac{h}{8}(\underline{K}_1 + 3\underline{K}_3))$$

$$K_5^{(j)} = f_j(x_i + h, \underline{u}_i + \frac{h}{2}(\underline{K}_1 - 3\underline{K}_3 + 4\underline{K}_4))$$

$$\underline{E}_T = \frac{h}{15}〔\underline{K}_1 - \frac{9}{2}\underline{K}_3 + 4\underline{K}_4 - \frac{1}{2}\underline{K}_5〕$$

例題 8-3　設計問題 D-VIII

試利用四階阮奇庫塔法解設計問題 D-VIII 的聯立微分方程式

$$\frac{dy_1}{dt} = -0.4y_1y_4$$

$$\frac{dy_2}{dt} = -\frac{dy_1}{dt} - 0.2y_2y_4$$

$$\frac{dy_3}{dt} = 0.2y_2y_4$$

$$\frac{dy_4}{dt} = \frac{dy_1}{dt} - \frac{dy_3}{dt} - 0.05y_4^2$$

I.C.　$y_1(0) = 0.2$　　$y_2(0) = 0.0$　　$y_3(0) = 0.0$　　$y_4(0) = 0.4$

解：

由於本例題所使用程式結構與例 8-2 完全相同，只有在積分作業及積分間距調整上略作調整，故流程圖及符號說明從略。

程式列印

```
' ************************************************
'   RUNGE-KUTTA METHOD FOR SYSTEM OF ODE
' ************************************************
'
Private Sub RungeKuttaMethod(Xpos, Ypos)
'
'       USER'S SPACE
```

```
'
Dim Y0(50) As Double

Cls

Print "Runge-Kutta Method "
Print "for a System of First Order Eqns"
Print

Print "Number of Equations N = ";
Neqn% = Val(InputBox("    Number of Equations = ", "NO. EQN", Neqn%, Xpos, Ypos))
Print Neqn%

Print ">ENTER LIMITS OF INTERGRATION"
Print "Lower Limit of X = ";
Lower = Val(InputBox("    Lower Limit,    Lower = ", "Low Limit of X", Lower, Xpos, Ypos))
Print Lower

Print "Upper Limit of X = ";
Upper = Val(InputBox("    Upper Limit,    Upper = ", "Upper Limit XS", Upper, Xpos, Ypos))
Print Upper

Print "> Initial Step Size = ";
H0 = Val(InputBox("> Initial Step Size = ", "Step Size", (Upper - Lower) / 1000, Xpos, Ypos))
Hsav = H0
Print H0

Print "> Error Tolerance on Y = ";
TolErr = Val(InputBox("> ERROR TOLERANCE ON Y = ", "Error Tolerance", H0 * 0.01, Xpos, Ypos))
Print TolErr

Print "> Need Step Size Control <Y/N> = ";
StepCtrl$ = InputBox(">STEPSIZE CONTRAL ? (Y/N) ", "STEP CONTROL", , Xpos, Ypos)
Print StepCtrl$

Print "> Need Detail Printout <Y/N> = ";
PNT$ = UCase(InputBox(">Need Detail Printout ? (Y/N) ", "PRINT CONTROL", "N", Xpos, Ypos))
Print PNT$

Print
NP = Val(InputBox("> NUMBER OF PRINTING INTERVALS = (<50) ", "NP", NP, Xpos, Ypos))

' --INITIAL CONDITIONS
Print "> Enter Initial Conditions:"
For K = 1 To Neqn%
    Print "    Y0("; K; ") = ";
    Y0(K) = Val(InputBox(" Initial Condition = ", "DATA ENTRY", , Xpos, Ypos))
    Print Y0(K)
Next K
```

```
MsgBox ("Ready to Start")

Call RungeKuttaCaller(Neqn%, Y0, TolErr, PNT$, H0, Upper, Lower, NP, StepCtrl$)

Print "=== END EXECUTION ==="
End Sub


Private Sub DeriveFunction(F, X, Y)
'
' USER DEFINED FUNCTIONS
'
F(1) = -0.4 * Y(1) * Y(4)
F(2) = -F(1) - 0.2 * Y(2) * Y(4)
F(3) = 0.2 * Y(2) * Y(4)
F(4) = F(1) - F(3) - 0.05 * Y(4) ^ 2
End Sub


Public Sub RungeKuttaCaller(Neqn%, Y0, TolErr, PNT$, H0, Upper, Lower, NP, StepCtrl$)
'
' Runge-Kutta Caller
' for System of ODE
'
' Neqn%          =No. of Eqns.
' Y0             =Initial Condition
' TolErr        =Local Error Tolerance.
' PNT$           = Detail Printing Control, Y for detailed print
' H0             =Initial Step Size.
' Upper          =Upper Bound
' Lower          =Lower Bound
' NP             = Total Number of Printing Intervals
' StepCtrl$ =Auto. Step Size Control ID.
'
' F=Function; Yi = F(I)
'
Cls
PrtEnt% = 1
Call ReportRK4(PrtEnt%, Neqn%, X0, Y0, TolErr)

GoBackPrt% = 1
PT$ = PNT$
XPT = (Upper - Lower) / NP
X0 = Lower
H = H0
XF = X0 + XPT

Do
      If PT$ = "Y" Or GoBackPrt% = 1 Then
            PrtEnt% = 2
```

```
            Call ReportRK4(PrtEnt%, Neqn%, X0, Y0, TolErr)
            PT$ = PNT$
        End If

        If X0 >= Upper Then Exit Do
        Call RungeKutta(Neqn%, X0, Y0, TolErr, H, H1, StepCtrl$)

' ==CHECK FOR END INTERVAL==

        If X0 = XF Then
            XF = X0 + XPT
            PT$ = "Y"
        End If
        If X0 + H > XF Then
            H = XF - X0
        End If
        If (X0 - H1) <= XF And PT$ = "Y" Then
            GoBackPrt% = 1
        Else
            GoBackPrt% = 0
        End If

Loop While X0 <= Upper
End Sub


Public Sub ReportRK4(PrtEnt%, Neqn%, X0, Y0, TolErr)
Select Case PrtEnt%
Case 1
        Print
        Print "SOLUTION            with Error Tolerance =";
        Print Format(TolErr, "    0.00E+00")
        Print "****************************************"
        Print "          X                              ";
        Print "          Y                              "
Case 2
        Print Format(X0, "      0.000000E+00");
        For IK = 1 To Neqn%
            Print Format(Y0(IK), "      0.000000E+00");
        Next IK
        Print
End Select
End Sub


Public Sub RungeKutta(Neqn%, X0, Y0, TolErr, H, H1, StepCtrl$)
Dim Y(50), W(50), Z(50), S(50), Q1(50), Q3(50) As Double
'
' Runge-Kutta Controller
'
```

```
T2 = H
T = T2
E5 = TolErr / 100

Ent% = 0
Call RungeKuttaModual(Ent%, Neqn%, S, X0, Y0, T, X, Y, D, W)

For K = 1 To Neqn%
    Q1(K) = W(K)
Next K

Do
    T = T2 / 2
    Ent% = 0
    Call RungeKuttaModual(Ent%, Neqn%, S, X0, Y0, T, X, Y, D, W)

    For K = 1 To Neqn%
        Z(K) = W(K)
        S(K) = Z(K)
    Next K

    X = X0 + T
    Ent% = 1
    Call RungeKuttaModual(Ent%, Neqn%, S, X0, Y0, T, X, Y, D, W)
    I8 = 0

    For K = 1 To Neqn%
        Q3(K) = (W(K) - Q1(K)) / 15
        DetErr = Abs(Q3(K)) / T2
        If DetErr >= TolErr Then
            ConvergentFlag% = 0: ' Not Converged
        Else
            If DetErr >= E5 Then I8 = 1
            ConvergentFlag% = 1
        End If
    Next K
'
' ==CONVERGENCE ATTAINED==
'
    If ConvergentFlag% = 1 Then
        For K = 1 To Neqn%
            Y0(K) = W(K) + Q3(K)
        Next K
        H = T2
        X0 = X0 + H
        H1 = H
'
'       ==CHECK FOR STEP-SIZE CONTROL==
'
        If StepCtrl$ = "N" Then Exit Sub
```

```
            If I8 = 1 Then Exit Sub
            H = 2 * H
            Exit Sub
        End If
'
' ==CONVERGENCE NOT ATTAINED==
'
        If StepCtrl$ = "N" Then
            Print
            Print "    **ERROR** "
            Print "        CONVERGENCE NOT ATTAINED"
            Print "        WITHIN REQUIRED TOLERANCE "
            Print "        FOR GIVEN STEP-SIZE"
            Print "        H="; H; "    AT X="; X0
        Else
            T2 = T2 / 2
            For K = 1 To Neqn%
                Q1(K) = Z(K)
            Next K
        End If
Loop While ConvergentFlag% = 0
End Sub


Public Sub RungeKuttaModual(Ent%, Neqn%, S, X0, Y0, T, X, Y, D, W)
'
' RUNGE-KUTTA MODUAL
'
' ==INITIALIZE VARIABLES ==
'
Dim F(50) As Double
If Ent% = 0 Then
        X = X0
        For K = 1 To Neqn%
            S(K) = Y0(K)
        Next K
End If

For K = 1 To Neqn%
        Y(K) = S(K)
        W(K) = S(K)
Next K
'
' ==COMPUTE W VALUE==
'
' --1st PASS
'
For K = 1 To Neqn%
        Call DeriveFunction(F, X, Y)
        D = T * F(K)
```

```
        W(K) = W(K) + D / 6
        Y(K) = S(K) + D / 2
    Next K
    X = X + T / 2
    '
    ' --2nd PASS
    '
    For K = 1 To Neqn%
        Call DeriveFunction(F, X, Y)
        D = T * F(K)
        W(K) = W(K) + D / 3
        Y(K) = S(K) + D / 2
    Next K
    '
    ' --3rd PASS
    '
    For K = 1 To Neqn%
        Call DeriveFunction(F, X, Y)
        D = T * F(K)
        Y(K) = S(K) + D
        W(K) = W(K) + D / 3
    Next K
    X = X + T / 2
    '
    ' --4th PASS
    '
    For K = 1 To Neqn%
        Call DeriveFunction(F, X, Y)
        D = T * F(K)
        W(K) = W(K) + D / 6
    Next K
    End Sub
```

副程式使用方法

1. 副程式 **Private Sub DeriveFunction(F, X, Y)**

   將微方方程式寫成以下型式；其中 F(i)=dY(i)/dx

   SUB DeriveFunction(F, X, Y)

   F(1) = -0.4 * Y(1) * Y(4)

   F(2) = -F(1) - 0.2 * Y(2) * Y(4)

   F(3) = 0.2 * Y(2) * Y(4)

   F(4) = F(1) - F(3) - 0.05 * Y(4) ^ 2

   End Sub

2. 副程式 **Public Sub RungeKuttaCaller(Neqn%, Y0, TolErr, PNT$, H0, Upper, Lower, NP, StepCtrl$)**

   爲阮奇庫塔法的呼叫控制副程式，使用者只要輸入 Neqn%, Y0, TolErr, PNT$, H0, Upper, Lower, NP, StepCtrl$，然後執行 RungeKuttaCaller 副程式，其餘工作即由程式自行處理。以上變數之定義，詳見程式說明。

3. 副程式 **Public Sub RungeKutta(Neqn%, X0, Y0, TolErr, H, H1, StepCtrl$)**

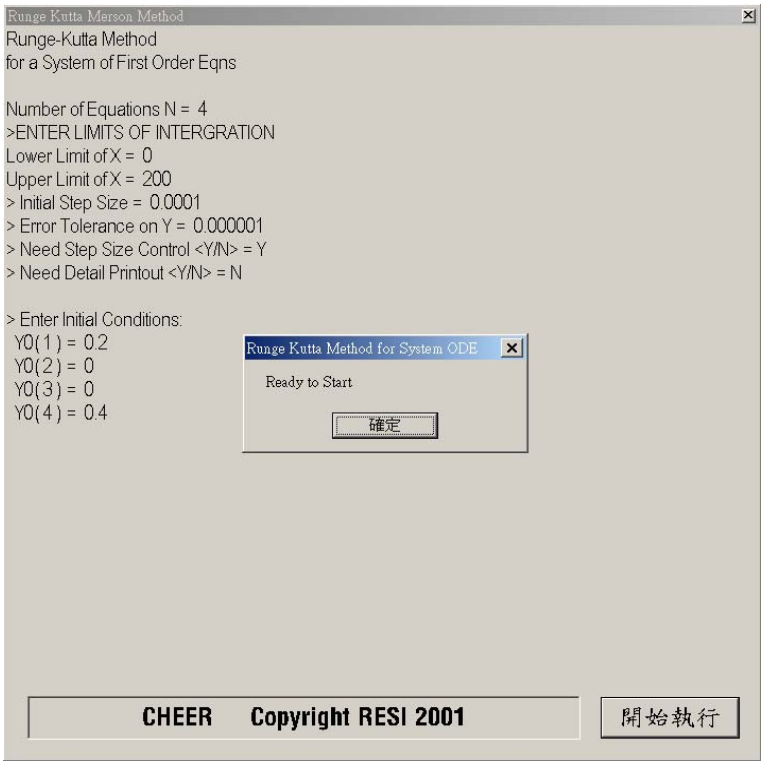
   爲阮奇庫塔法的工作控制副程式，用於控制計算流程及收斂檢驗。

4. 副程式 **Public Sub RungeKuttaModual(Ent%, Neqn%, S, X0, Y0, T, X, Y, D, W)**

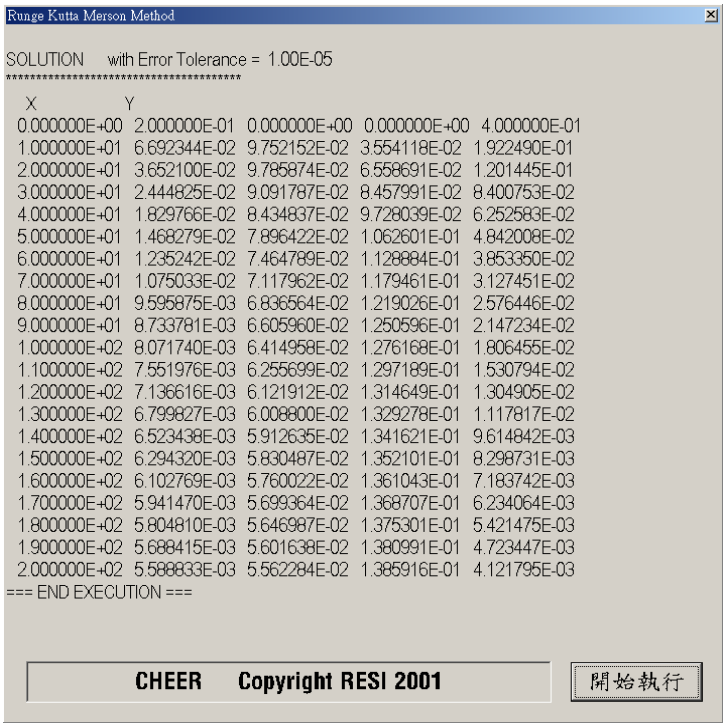
   爲阮奇庫塔法的工作模組副程式，用於計算 K_1 , K_2 , K_3 , K_4 。

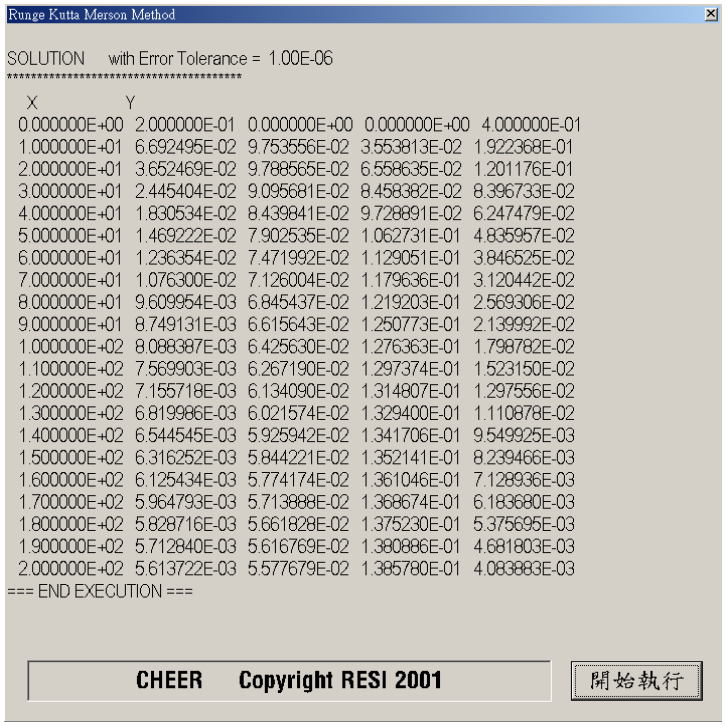
程式執行結果

輸入之基本數據



計算所得結果如下：


Runge Kutta Merson Method
SOLUTION with Error Tolerance = 1.00E-05
****
X Y
0.000000E+00 2.000000E-01 0.000000E+00 0.000000E+00 4.000000E-01
1.000000E+01 6.692344E-02 9.752152E-02 3.554118E-02 1.922490E-01
2.000000E+01 3.652100E-02 9.785874E-02 6.558691E-02 1.201445E-01
3.000000E+01 2.444825E-02 9.091787E-02 8.457991E-02 8.400753E-02
4.000000E+01 1.829766E-02 8.434837E-02 9.728039E-02 6.252583E-02
5.000000E+01 1.468279E-02 7.896422E-02 1.062601E-01 4.842008E-02
6.000000E+01 1.235242E-02 7.464789E-02 1.128884E-01 3.853350E-02
7.000000E+01 1.075033E-02 7.117962E-02 1.179461E-01 3.127451E-02
8.000000E+01 9.595875E-03 6.836564E-02 1.219026E-01 2.576446E-02
9.000000E+01 8.733781E-03 6.605960E-02 1.250596E-01 2.147234E-02
1.000000E+02 8.071740E-03 6.414958E-02 1.276168E-01 1.806455E-02
1.100000E+02 7.551976E-03 6.255699E-02 1.297189E-01 1.530794E-02
1.200000E+02 7.136616E-03 6.121912E-02 1.314649E-01 1.304905E-02
1.300000E+02 6.799827E-03 6.008800E-02 1.329278E-01 1.117817E-02
1.400000E+02 6.523438E-03 5.912635E-02 1.341621E-01 9.614842E-03
1.500000E+02 6.294320E-03 5.830487E-02 1.352101E-01 8.298731E-03
1.600000E+02 6.102769E-03 5.760022E-02 1.361043E-01 7.183742E-03
1.700000E+02 5.941470E-03 5.699364E-02 1.368707E-01 6.234064E-03
1.800000E+02 5.804810E-03 5.646987E-02 1.375301E-01 5.421475E-03
1.900000E+02 5.688415E-03 5.601638E-02 1.380991E-01 4.723447E-03
2.000000E+02 5.588833E-03 5.562284E-02 1.385916E-01 4.121795E-03
=== END EXECUTION ===
CHEER Copyright RESI 2001
開始執行


Runge Kutta Merson Method
SOLUTION with Error Tolerance = 1.00E-06
****
X Y
0.000000E+00 2.000000E-01 0.000000E+00 0.000000E+00 4.000000E-01
1.000000E+01 6.692495E-02 9.753556E-02 3.553813E-02 1.922368E-01
2.000000E+01 3.652469E-02 9.788565E-02 6.558635E-02 1.201176E-01
3.000000E+01 2.445404E-02 9.095681E-02 8.458382E-02 8.396733E-02
4.000000E+01 1.830534E-02 8.439841E-02 9.728891E-02 6.247479E-02
5.000000E+01 1.469222E-02 7.902535E-02 1.062731E-01 4.835957E-02
6.000000E+01 1.236354E-02 7.471992E-02 1.129051E-01 3.846525E-02
7.000000E+01 1.076300E-02 7.126004E-02 1.179636E-01 3.120442E-02
8.000000E+01 9.609954E-03 6.845437E-02 1.219203E-01 2.569306E-02
9.000000E+01 8.749131E-03 6.615643E-02 1.250773E-01 2.139992E-02
1.000000E+02 8.088387E-03 6.425630E-02 1.276363E-01 1.798782E-02
1.100000E+02 7.569903E-03 6.267190E-02 1.297374E-01 1.523150E-02
1.200000E+02 7.155718E-03 6.134090E-02 1.314807E-01 1.297556E-02
1.300000E+02 6.819986E-03 6.021574E-02 1.329400E-01 1.110878E-02
1.400000E+02 6.544545E-03 5.925942E-02 1.341706E-01 9.549925E-03
1.500000E+02 6.316252E-03 5.844221E-02 1.352141E-01 8.239466E-03
1.600000E+02 6.125434E-03 5.774174E-02 1.361046E-01 7.128936E-03
1.700000E+02 5.964793E-03 5.713888E-02 1.368674E-01 6.183680E-03
1.800000E+02 5.828716E-03 5.661828E-02 1.375230E-01 5.375695E-03
1.900000E+02 5.712840E-03 5.616769E-02 1.380886E-01 4.681803E-03
2.000000E+02 5.613722E-03 5.577679E-02 1.385780E-01 4.083883E-03
=== END EXECUTION ===
CHEER Copyright RESI 2001
開始執行

例題 8-5 再解設計問題 D-VIII

試利用阮奇庫塔摩森法重解設計問題 D-VIII，並與上例結果作比較。

$$\frac{dy_1}{dt} = -0.4y_1y_4$$
$$\frac{dy_2}{dt} = -\frac{dy_1}{dt} - 0.2y_2y_4$$
$$\frac{dy_3}{dt} = 0.2y_2y_4$$
$$\frac{dy_4}{dt} = \frac{dy_1}{dt} - \frac{dy_3}{dt} - 0.05y_4^2$$

I.C. $y_1(0) = 0.2$ $y_2(0) = 0.0$ $y_3(0) = 0.0$ $y_4(0) = 0.4$

解：

用於解聯立微分方程式的阮奇庫塔摩森法，基本上與阮奇庫塔法相似，程式結構也類似。其積分間距則可以利用截尾誤差估算方程式做最適化調整。

程式列印

```
' ************************************************
'   RUNGE-KUTTA MERSON METHOD
'   FOR SYSTEM OF ODES
' ************************************************
'
Private Sub RungeKuttaMersonMethod(Xpos, Ypos)
'
'        USER'S SPACE
'
Dim Y0(51) As Double
Cls

Print "Runge-Kutta Merson Method "
Print " for a System of First Order Eqns"
Print

Print "Number of Equations N = ";
Neqn% = Val(InputBox("    Number of Equations = ", "NO. EQN", Neqn%, Xpos, Ypos))
Print Neqn%
```

```
Print ">ENTER LIMITS OF INTERGRATION"
Print "Lower Limit of X = ";
Lower = Val(InputBox("    Lower Limit,    Lower = ", "Low Limit of X", Lower, Xpos, Ypos))
Print Lower
Y0(1) = Lower

Print "Upper Limit of X = ";
Upper = Val(InputBox("    Upper Limit,    Upper = ", "Upper Limit XS", Upper, Xpos, Ypos))
Print Upper

Print "> Initial Step Size = ";
H0 = Val(InputBox("> Initial Step Size = ", "Step Size", (Upper - Lower) / 1000, Xpos, Ypos))
Hsav = H0
Print H0

Print "> Error Tolerance on Y = ";
TolErr = Val(InputBox("> ERROR TOLERANCE ON Y = ", "Error Tolerance", H0 * 0.01, Xpos, Ypos))
Print TolErr

Print "> Need Detail Printout <Y/N> = ";
PNT$ = UCase(InputBox(">Need Detail Printout ? (A/Y/N) ", "PRINT CONTROL", "N", Xpos, Ypos))
Print PNT$

Print
NP = Val(InputBox("> NUMBER OF PRINTING INTERVALS = (<50) ", "NP", NP, Xpos, Ypos))

' --INITIAL CONDITIONS

Print "> Enter Initial Conditions:"
For K = 1 To Neqn%
      Print "    Y0("; K; ") = ";
      Y0(K + 1) = Val(InputBox(" Initial Condition = ", "DATA ENTRY", , Xpos, Ypos))
      Print Y0(K + 1)
Next K
MsgBox ("Ready to Start")

Call RungeKuttaMersonCaller(Neqn%, Y0, TolErr, PNT$, H0, Upper, Lower, NP)
Print "=== END EXECUTION ==="
End Sub


Private Sub DeriveFunction(DY, X, Y)
'
' ========== DEFINE FUNCTIONS
'
DY(1) = -0.4 * Y(1) * Y(4)
DY(2) = -DY(1) - 0.2 * Y(2) * Y(4)
DY(3) = 0.2 * Y(2) * Y(4)
```

```
DY(4) = DY(1) - DY(3) - 0.05 * Y(4) ^ 2
End Sub


Public Sub RungeKuttaMersonCaller(Neqn%, Y0, TolErr, PNT$, H0, Upper, Lower, NP)
'
' RUNGE KUTTA CALLER
'
' Neqn%     = No. of Eqns.
' Y0        = Initial Condition
' TolErr    = Local Error Tolerance.
' PNT$      = Detail Print Control, Y for Print Everything
' H0        = Initial Step Size.
' Upper     = Upper Bound
' Lower     = Lower Bound
' NP        = Total Number of Printing Intervals
'
' F=Function; Yi = F(I)
'
Cls
PrtEnt% = 1
Call ReportRK4(PrtEnt%, Neqn%, Y0, TolErr)
XPT = (Upper - Lower) / NP
X0 = Lower
XF = X0 + XPT
PrtEnt% = 2
Call ReportRK4(PrtEnt%, Neqn%, Y0, TolErr)
Do
    StepSize = H0
    Call RungeKuttaMerson(Neqn%, XF, Y0, TolErr, StepSize, PNT$)
    X0 = Y0(1)
    XF = X0 + XPT
    If PNT$ <> "Y" Or PNT$ <> "A" Then
        PrtEnt% = 2
        Call ReportRK4(PrtEnt%, Neqn%, Y0, TolErr)
    End If
    If XF > Upper Then Exit Do
Loop While X0 <= Upper
End Sub


Public Sub ReportRK4(PrtEnt%, Neqn%, Y0, TolErr)
Select Case PrtEnt%
Case 1
    Print
    Print "SOLUTION          with Error Tolerance =";
    Print Format(TolErr, "    0.00E+00")
    Print "***************************************"
```

```
        Print "          X                          ";
        Print "          Y                          "
Case 2
        For IK = 1 To Neqn% + 1
                Print Format(Y0(IK), "      0.000000E+00");
        Next IK
        Print
End Select
End Sub


Public Sub RungeKuttaMerson(Neqn%, XS, Y0, TolErr, StepSize, IP$)
'====================================
'     KUTTA MERSON SUBROUTINE
'====================================
Dim Work(204) As Double
Hsave = StepSize
NN = Neqn% + 1
ND = NN * 2
NX = NN * 3
NY = NN * 4
Do
        NewStepSize = StepSize
        LpFinal$ = "N"
        Call RungeKuttaMersonModule(Neqn%, Y0, Work, StepSize)
'
'       TEST ERROR ON EACH Y
'
        For J = 1 To Neqn%
                ErrValue = (Abs(Work(NX + J + 1) - Work(J + 1))) / 5!
                If ErrValue > TolErr Then
'
'                       ERROR TOO LARGE, REDUCE StepSize
'
                        StepSize = StepSize * (TolErr / ErrValue) ^ 2
                        If IP$ = "A" Then
                                Print "Step size reduced to ";
                                Print Format(StepSize, "    0.00E+00")
                        End If
                        ChangeStep$ = "R"
                        Exit For
                ElseIf ErrValue < 0.95 * TolErr Then
'
'                       ERROR TOO SMALL, INCREASE StepSize
'
                        If (ErrValue - 100000000000#) <= 0 Then
                                NewStepSize = 5! * StepSize
                        Else
```

```
                        NewStepSize = StepSize * NewStepSize ^ 2
                        If (NewStepSize - 5 * StepSize) > 0 Then
                            NewStepSize = 5! * StepSize
                        End If
                    End If
                    If IP$ = "A" Then
                        Print "Step size provisional increase to ";
                        Print Format(NewStepSize, " 0.00E+00")
                    End If
                    ChangeStep$ = "I"
                Else
                    ChangeStep$ = "OK"
                End If
            Next J

            If ChangeStep$ <> "R" Then
                StepSize = NewStepSize
'
'           ERROR ON THAT INTERVAL WAS NICELY WITHIN BOUNDS,
'           TEST WHETHER UPPER BOUND WITHIN NEXT INTERVAL OR NOT
'
                If (Work(1) + StepSize - XS) > 0 Then
                    StepSize = XS - Work(1)
                    If StepSize <= 0 Then LpFinal$ = "Y"
                End If
'
'           PRINT VARIABLES CALCULATED
'
                If LoopFinal$ <> "Y" Then
                    For I = 1 To NN
                        Y0(I) = Work(I)
                    Next I
                    If IP$ = "Y" Or IP$ = "Y" Then
                        PrtEnt% = 2
                        Call ReportRK4(PrtEnt%, Neqn%, Y0, TolErr)
                    End If
                    If IP$ = "A" Then
                        Print
                        Print "Step size = ";
                        Print Format(StepSize, " 0.00E+00")
                        Print
                    End If
                End If
            End If

            If StepSize <= 0 Then LpFinal$ = "Y"
        Loop While LpFinal$ <> "Y"
```

```
For I = 1 To NN
        Y0(I) = Work(I)
Next I

End Sub


Public Sub RungeKuttaMersonModule(Neqn%, Y0, Work, StepSize)
'
'      RUNGE-KUTTA MODUAL
'
Dim Y(51), YS(51), DY(50) As Double
‘
' ************* GET START
‘
NN = Neqn% + 1
ND = NN * 2
NX = NN * 3
NY = NN * 4
'
' --STORE YO, CREATE Y1, DY1
'
For I = 1 To NN
        YS(I) = Y0(I)
Next I
For I = 1 To Neqn%
        Y(I) = YS(I + 1)
Next I
Call DeriveFunction(DY, Y0(1), Y)
'
' --CREATE Y2, DY2
'
For I = 1 To Neqn%
        Work(NN + I) = DY(I)
        Work(I + 1) = YS(I + 1) + StepSize * DY(I) / 3!
Next I
Work(1) = YS(1) + StepSize / 3
For I = 1 To Neqn%
        Y(I) = Work(I + 1)
Next I
Call DeriveFunction(DY, Y0(1), Y):    '
'
' -- STORE DY2, CREATE Y3, DY3
'
For I = 1 To Neqn%
        Work(I + 1) = YS(I + 1) + StepSize * Work(NN + I) / 6! + StepSize * DY(I) / 6!
Next I
For I = 1 To Neqn%
```

```
        Y(I) = Work(I + 1)
    Next I
    Call DeriveFunction(DY, Y0(1), Y):    '
    '
    ' --CREATE Y4, STORE Y4, DY3
    '
    For I = 1 To Neqn%
        Work(ND + I) = DY(I)
        Work(I + 1) = YS(I + 1) + StepSize * Work(NN + I) / 8! + StepSize * DY(I) * 3! / 8!
    Next I
    Work(1) = YS(1) + StepSize * 0.5
    For I = 1 To Neqn%
        Y(I) = Work(I + 1)
    Next I
    Call DeriveFunction(DY, Y0(1), Y):    '
    '
    ' --CREATE Y5
    '
    For I = 1 To Neqn%
        Work(I + 1) = YS(I + 1) + StepSize * Work(NN + I) * 0.5 - Work(ND + I) * StepSize * 1.5
        Work(I + 1) = Work(I + 1) + StepSize * DY(I) * 2!
        Work(NX + I + 1) = Work(I + 1)
        Work(ND + I) = DY(I)
    Next I
    Work(1) = YS(1) + StepSize
    For I = 1 To Neqn%
        Y(I) = Work(I + 1)
    Next I
    Call DeriveFunction(DY, Y0(1), Y):    '
    '
    ' -- Generate U(I+1)
    '
    For I = 1 To Neqn%
        Work(I + 1) = YS(I + 1) + StepSize * Work(NN + I) / 6! + StepSize * Work(ND + I) * 2 / 3
        Work(I + 1) = Work(I + 1) + StepSize * DY(I) / 6
    Next I

    ' CopyRight Dr. Ron Hsin Chang

    End Sub
```

副程式使用方法

1. 副程式 **Private Sub DeriveFunction(DY, X, Y)**
   將微方方程式寫成以下型式；其中 DY(I)=dY(I)/dx

```
SUB DeriveFunction(F, X, Y)
DY(1) = -0.4 * Y(1) * Y(4)
DY(2) = -DY(1) - 0.2 * Y(2) * Y(4)
DY(3) = 0.2 * Y(2) * Y(4)
DY(4) = DY(1) - DY(3) - 0.05 * Y(4) ^ 2
End Sub
```

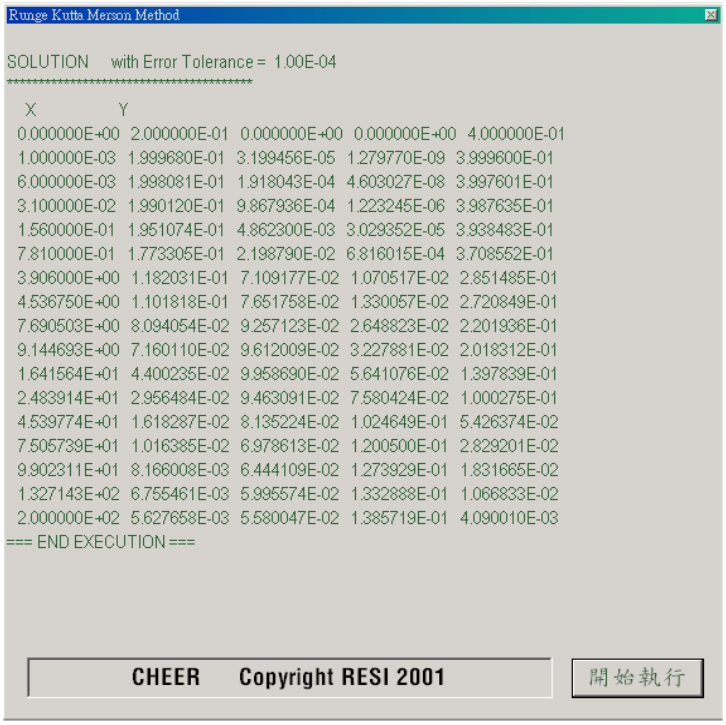
2. 副程式 **Public Sub RungeKuttaMersonCaller(Neqn%, Y0, TolErr, PNT$, H0, Upper, Lower, NP)**
   爲阮奇庫塔摩森法的呼叫控制副程式，使用者只要輸入 Neqn%, Y0, TolErr, PNT$, H0, Upper, Lower, NP，然後執行 RungeKuttaMersonCaller 副程式，其餘工作即由程式自行處理。

3. 副程式 **Public Sub RungeKuttaMerson(Neqn%, XS, Y0, TolErr, StepSize, IP$)**
   爲阮奇庫塔摩森法的工作控制副程式，用於控制計算流程及收斂檢驗工作。

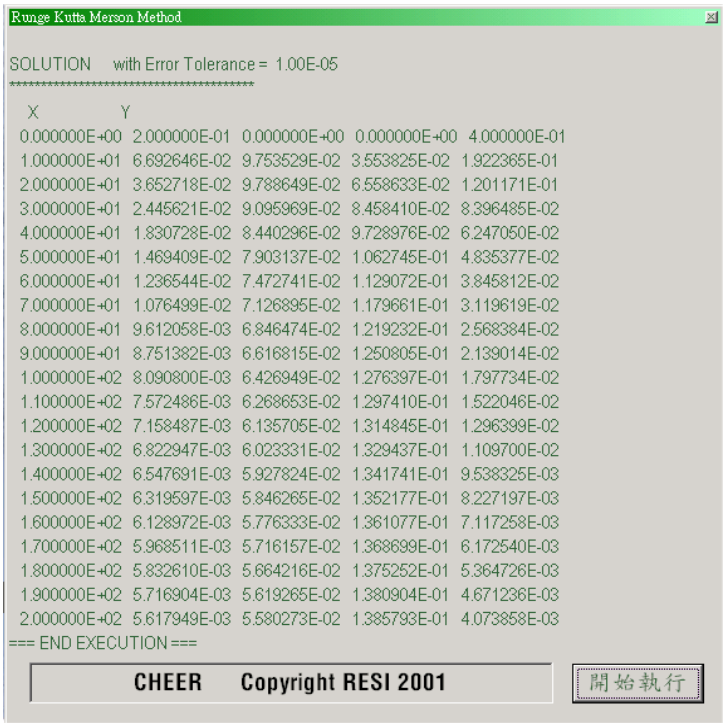
4. 副程式 **Public Sub RungeKuttaMersonModule(Neqn%, Y0, Work, StepSize)**
   爲阮奇庫塔摩森法的工作模組副程式，用於計算 K_1, K_2, K_3, K_4, K_5 。

程式執行結果

輸入資料仿照例 8-3，自動變區間積分所得結果如下：



分割成 20 區間輸出結果如下：



積分所使用方法及誤差容許度設定值對所得解答之影響如下表，一般而言，阮奇庫塔摩森法收斂速度較四階阮奇庫塔法快。另外，本程式除使用方法不同外，對於積分效率也作完全不同的處理。在阮奇庫塔法中，我們調節積分間距的策略是作加倍或減半處理；如果誤差太小，就將積分間距加倍；如果積分所得誤差太大，就將積分間距減半。但在撰寫阮奇庫塔摩森法程式中，積分間距調節策略則是採用相對誤差調整法，依實際誤差與所定範圍作比較，按比率調節；一般而言，最高一次可以調整達五倍。

方法	TolErr	Y1	Y2	Y3	Y4
RK4	1×10^{-4}	5.4657×10^{-3}	5.4894×10^{-2}	1.3873×10^{-1}	4.2692×10^{-3}
RK4	1×10^{-5}	5.5886×10^{-3}	5.5622×10^{-2}	1.3859×10^{-1}	4.1200×10^{-3}
RKM	1×10^{-4}	5.6277×10^{-3}	5.5800×10^{-2}	1.3857×10^{-1}	4.0900×10^{-3}
RKM	1×10^{-5}	5.6179×10^{-3}	5.5803×10^{-2}	1.3858×10^{-1}	4.0739×10^{-3}
RKM	1×10^{-6}	5.6177×10^{-3}	5.5803×10^{-2}	1.3858×10^{-1}	4.0738×10^{-3}

例題 8-5　再解設計問題 D-VIII

試利用阮奇庫塔費勃格法的解法撰寫一簡單的 Excel 程式，以了解這種方法的計算流程，然後設計一 Visual Basic 程式，重解設計問題 D-VIII，並與例 8-4 結果作比較。

$$\frac{dy_1}{dt} = -0.4y_1y_4$$
$$\frac{dy_2}{dt} = -\frac{dy_1}{dt} - 0.2y_2y_4$$
$$\frac{dy_3}{dt} = 0.2y_2y_4$$
$$\frac{dy_4}{dt} = \frac{dy_1}{dt} - \frac{dy_3}{dt} - 0.05y_4^2$$

I.C.　$y_1(0) = 0.2$　　$y_2(0) = 0.0$　　$y_3(0) = 0.0$　　$y_4(0) = 0.4$

利用 Excel 版本程式說明 RKF 程式運作原理：

1. 首先製作 A1..I14 的表格，填入積分間距值 (Step Size，例如 1.E-2)。

2. 設定相對誤差 (RelErr) 及絕對誤差 (AbsErr) 要求條件。

3. B3 填入 X 的起始值 0，然後在 C3..G3 依序照著填入 X 的表示式，例如 C3 填入

=B1 + I2 / 4。

4. B5..B8 填入 Y 的起始條件。
5. B11..B14 填入 K1=f(X1,y1) 的表示式，例如，B11 填入= - 0.4*B5*B8，B12 填入 = - B11 – 0.2 * B6 * B8，於此類推。
6. C5 即爲 $u_i+K_1/4$ 或寫成 C5=B5+B11/4，並複製到 C6..C8。
7. 同理，D5=B5+B11*3/32+C11*9/32，並複製到 D6..D8。
8. E5=B5+1932*B11/2197-7200*C11/2197+7296*D11/2197，並複製到 E6..E8。
9. F5=B5+439/216*B11-8*C11+3680/513*D11-845*E11/4104，並複製到 F6..F8。
10. G5=B5-8/27*B11+2*C11-3544/2565*D11+1859/4104*E11-11/40*F11 ， 並 複 製 到 G6..G8。
11. RKF 方法的結尾誤差爲 H5=B11/360-C11*128/4275-E11*2197/75240+F11/50+2/55*G11，並複製到 H6..H8。
12. 截尾誤差與相對及絕對誤差之比較計算爲 I5=H5*I$2/(I$10/2*(B5+B19)+I$11)，並複製到 I6..I8。
13. 選擇誤差比較計算值之最大誤差，最爲估算積分間距之依據，誤差最大值可以利用 Excel 的內定函數 Max 求之。即 I12=Max(I5..I8)。
14. 製作下一積分步驟的表格，複製第 1..14 列，成爲第 15..28 列。
15. 新的積分間距 (Step Size) 利用截尾誤差估算爲 I16=0.9/I12^0.2*I2。
16. 新的 X 值爲 B17=B3+I16。
17. 新的 Y 值爲 B19=B5+I$2*(25/216*B11+1408/2565*D11+2197/4104*E11-F11/5)，並複製到 B20..B22。即完成第二組表格。
18. 複製第 15..28 列，成爲第 29..42 列。此時完全不需修改內容，餘此類推，即可一步一步往前積分。

	A	B	C	D	E	F	G	H	I
1	Runge Kutta Fehlberg Method for IVP-ODE								
2		X0	X0+h/4	X0+3h/8	X0+12h/13	X0+h	X0+h/2	Step Size =	1.0000E-02
3	X	0.0000E+00	2.5000E-03	3.7500E-03	9.2308E-03	1.0000E-02	5.0000E-03		
4		1	2	3	4	5	6	Et	EeOet
5	Y1	2.0000E-01	1.9200E-01	1.8896E-01	1.7466E-01	1.7183E-01	1.8595E-01	3.7538E-05	3.1285E-02
6	Y2	0.0000E+00	8.0000E-03	1.0868E-02	2.4677E-02	2.7609E-02	1.3710E-02	-4.3112E-05	-4.3107E-02
7	Y3	0.0000E+00	0.0000E+00	1.6740E-04	6.6121E-04	5.6210E-04	3.4385E-04	5.5745E-06	5.5744E-03
8	Y4	4.0000E-01	3.7200E-01	3.6184E-01	3.1301E-01	3.0270E-01	3.5178E-01	1.4565E-04	1.0407E-01
9									

10		K1	K2	K3	K4	K5	K6	RelErr=	1.0000E-05
11	YP1	-3.2000E-02	-2.8570E-02	-2.7350E-02	-2.1868E-02	-2.0805E-02	-2.6165E-02	AbsErr =	1.0000E-05
12	YP2	3.2000E-02	2.7974E-02	2.6563E-02	2.0324E-02	1.9133E-02	2.5200E-02	EeOet=	1.0407E-01
13	YP3	0.0000E+00	5.9520E-04	7.8647E-04	1.5448E-03	1.6714E-03	9.6456E-04		
14	YP4	-1.1200E-01	-9.8357E-02	-9.3599E-02	-7.2401E-02	-6.8289E-02	-8.9004E-02		
15									
16		X0	X0+h/4	X0+3h/8	X0+12h/13	X0+h	X0+h/2	Step Size =	1.4151E-02
17	X	1.4151E-02	1.7688E-02	1.9457E-02	2.7213E-02	2.8301E-02	2.1226E-02		
18								Et	EeOet
19	Y1	1.9974E-01	1.9177E-01	1.8874E-01	1.7448E-01	1.7166E-01	1.8573E-01	3.7356E-05	4.8060E-02
20	Y2	2.5338E-04	8.2200E-03	1.1078E-02	2.4832E-02	2.7748E-02	1.3910E-02	-4.2885E-05	-5.0603E-02
21	Y3	9.2441E-06	1.4300E-05	1.8278E-04	6.8377E-04	5.8748E-04	3.6016E-04	5.5295E-06	7.8246E-03
22	Y4	3.9911E-01	3.7122E-01	3.6109E-01	3.1245E-01	3.0219E-01	3.5106E-01	1.4489E-04	1.5780E-01
23									
24		K1	K2	K3	K4	K5	K6	RelErr=	1.0000E-05
25	YP1	-3.1887E-02	-2.8475E-02	-2.7261E-02	-2.1807E-02	-2.0750E-02	-2.6081E-02	AbsErr =	1.0000E-05
26	YP2	3.1866E-02	2.7864E-02	2.6461E-02	2.0255E-02	1.9073E-02	2.5104E-02	EeOet=	1.5780E-01
27	YP3	2.0225E-05	6.1028E-04	8.0001E-04	1.5517E-03	1.6770E-03	9.7668E-04		
28	YP4	-1.1155E-01	-9.7987E-02	-9.3253E-02	-7.2170E-02	-6.8087E-02	-8.8680E-02		
29									
30		X0	X0+h/4	X0+3h/8	X0+12h/13	X0+h	X0+h/2	Step Size =	1.8424E-02
31	X	3.2575E-02	3.7181E-02	3.9484E-02	4.9582E-02	5.1000E-02	4.1787E-02		
32								Et	EeOet
33	Y1	1.9948E-01	1.9153E-01	1.8851E-01	1.7431E-01	1.7150E-01	1.8551E-01	3.7175E-05	6.2280E-02
34	Y2	5.0580E-04	8.4391E-03	1.1287E-02	2.4986E-02	2.7886E-02	1.4110E-02	-4.2660E-05	-6.5558E-02
35	Y3	1.8612E-05	2.8683E-05	1.9824E-04	7.0634E-04	6.1285E-04	3.7653E-04	5.4849E-06	1.0105E-02
36	Y4	3.9821E-01	3.7044E-01	3.6034E-01	3.1189E-01	3.0169E-01	3.5035E-01	1.4414E-04	2.0448E-01
37									
38		K1	K2	K3	K4	K5	K6	RelErr=	1.0000E-05
39	YP1	-3.1774E-02	-2.8380E-02	-2.7172E-02	-2.1746E-02	-2.0696E-02	-2.5998E-02	AbsErr =	1.0000E-05
40	YP2	3.1733E-02	2.7755E-02	2.6359E-02	2.0187E-02	1.9013E-02	2.5009E-02	EeOet=	2.0448E-01
41	YP3	4.0283E-05	6.2524E-04	8.1344E-04	1.5586E-03	1.6826E-03	9.8869E-04		
42	YP4	-1.1110E-01	-9.7618E-02	-9.2909E-02	-7.1941E-02	-6.7886E-02	-8.8359E-02		
43									
44		X0	X0+h/4	X0+3h/8	X0+12h/13	X0+h	X0+h/2	Step Size =	2.2777E-02
45	X	5.5353E-02	6.1047E-02	6.3894E-02	7.6378E-02	7.8130E-02	6.6741E-02		
46								Et	EeOet
47	Y1	1.9921E-01	1.9130E-01	1.8829E-01	1.7413E-01	1.7134E-01	1.8530E-01	3.6995E-05	7.6631E-02
48	Y2	7.5726E-04	8.6575E-03	1.1495E-02	2.5141E-02	2.8024E-02	1.4309E-02	-4.2436E-05	-8.0642E-02
49	Y3	2.8102E-05	4.3146E-05	2.1376E-04	7.2892E-04	6.3820E-04	3.9297E-04	5.4406E-06	1.2392E-02
50	Y4	3.9733E-01	3.6966E-01	3.5960E-01	3.1133E-01	3.0118E-01	3.4964E-01	1.4339E-04	2.5159E-01
51									
52		K1	K2	K3	K4	K5	K6	RelErr=	1.0000E-05
53	YP1	-3.1661E-02	-2.8286E-02	-2.7084E-02	-2.1685E-02	-2.0641E-02	-2.5915E-02	AbsErr =	1.0000E-05
54	YP2	3.1601E-02	2.7646E-02	2.6257E-02	2.0119E-02	1.8953E-02	2.4914E-02	EeOet=	2.5159E-01
55	YP3	6.0176E-05	6.4007E-04	8.2675E-04	1.5654E-03	1.6881E-03	1.0006E-03		
56	YP4	-1.1066E-01	-9.7252E-02	-9.2567E-02	-7.1712E-02	-6.7685E-02	-8.8038E-02		

RKF 積分法是一種非常普遍被使用的變異步伐積分法，由以上表格所顯示的結果可以看出，開始執行後，積分間距將逐步加大，以提昇積分效率。利用 Excel 表格製作常微分方程式的積分過程，可以充分建立邏輯思維，但如上表所示，若要由 X=0 積分至 X=200，將變成一組非常冗長的表格，通常較少被使用。

程式列印

```
' ********************************************
' RUNGE-KUTTA-FEHLBERG METHOD
' ********************************************
'
Private Sub RungeKuttaFehlbergMethod(Xpos, Ypos)
'
' Reserve Space for Initial Conditions
Dim Y0(10) As Double
'
' ================== USER'S SPACE
'
Neqn% = 4: ' Number of eqns.
Cls

Print "Number of Equations N = ";
Neqn% = Val(InputBox("    Number of Equations = ", "", Neqn%, Xpos, Ypos))
Print Neqn%

Print "> Enter Limits of Integration:"
Print "Lower Limit of X = ";
Lower = Val(InputBox("    Lower Limit    X0 = ", "Low Limit of X", Lower, Xpos, Ypos))
Print Lower

Print "Upper Limit of X = ";
Upper = Val(InputBox("    Upper Limit    XS = ", "Upper Limit XS", Upper, Xpos, Ypos))
Print Upper

Print "> Initial Step Size = ";
H0 = Val(InputBox("> Initial Step Size = ", "", (Upper - Lower) / 1000, Xpos, Ypos))
Hsav = H0
Print H0

' --ERROR TOLLERANCE
Print "> Error Tolerance on Y = ";
AbsErr = Val(InputBox("> ABSOLUTE ERROR TOLERANCE ON Y = ", "Error Tolerance", H0 * 0.001,
Xpos, Ypos))
Print AbsErr
```

```
Print "> Error Tolerance on Y = ";
RelErr = Val(InputBox("> RELATIVE ERROR TOLERANCE ON Y = ", "Error Tolerance", 0.0001,
Xpos, Ypos))
Print RelErr
'
' --PRINTING CONTROL
'
Print
NP = Val(InputBox("> NUMBER OF PRINTING INTERVALS = (<50) ", "NP", 20, Xpos, Ypos))
Print
'
' -- Print Details
'
Print
PrtAll$ = InputBox("> Print Detail Information <Y/N> ", "Print All ?", "N", Xpos, Ypos)
Print

' --INITIAL CONDITIONS
Print "> Enter Initial Conditions:"
For I = 1 To Neqn%
        Print
        Print "    Y0("; I; ") = ";
        Y0(I) = Val(InputBox(" Initial Condition = ", "DATA ENTRY", , Xpos, Ypos))
        Print Y0(I)
Next I
'
'    Call Runge Kutta Fehlberg Subroutine RKF2000
'
Call RKF2000(Neqn%, Lower, Upper, H0, Y0, AbsErr, RelErr, NP, PrtAll$)

End Sub


Private Sub DeriveFunction(T, Y, YP)
'
' ========== DEFINE FUNCTIONS
'
YP(1) = -0.4 * Y(1) * Y(4)
YP(2) = -YP(1) - 0.2 * Y(2) * Y(4)
YP(3) = 0.2 * Y(2) * Y(4)
YP(4) = YP(1) - YP(3) - 0.05 * Y(4) * Y(4)
End Sub


Public Sub RKF2000(Neqn%, Lower, Upper, H0, Y0, AbsErr, RelErr, NP, PrtAll$)
'
'    ************************************************************
```

```
'   Runge-Kutta-Fehlberg Method for Solving IVP-ODE
'   ************************************************************
'
'   Neqn%       = Number of Equations
'   Lower       = Lower Bound of Independent Variable
'   Upper       = Upper Bound of Independent Variable
'   H0          = Initial Step Size
'   Y0(I)       = Initial Conditions
'   AbsErr      = Absolute Error Tolerance
'   RelErr      = Relative Error Tolerance
'   NP          = Number of Printing Intervals
'   PrtAll$     = Y for Print Everything
'
'   Program Developed by Dr. Ron Hsin Chang @ Year 2000
'
Dim Y(10), YP(10), F1(10), F2(10), F3(10), F4(10), F5(10), SM(10) As Double
'
'   Define Print Control Interval
'
Iflag% = 1
XPT = (Upper - Lower) / NP
X = Lower
'
'   Put Y0 to Y for Initiation
'
For I = 1 To Neqn%
      Y(I) = Y0(I)
Next I
'
'   Print Report Tittle
'
Cls
PrtEnt% = 1
Call ReportRK4(PrtEnt%, Neqn%, X, Y, RelErr, AbsErr)
PrtEnt% = 2
Call ReportRK4(PrtEnt%, Neqn%, X, Y, RelErr, AbsErr)
'
' --CALL KUTTA-FEHLBERG
'
Do
      XF = X + XPT
      Do
            Call RKF(Neqn%, X, XF, Y, RelErr, AbsErr, YP, H0, F1, F2, F3, F4, F5, SM, SavRE,
SavAE, NoFunEst, Init, Iflag%, JFalg%, Kflag%, PrtAll$)

            Chk% = 0
'
'           Error Message Controls
```

```
'
            Select Case Iflag%
                Case 1
                    Print "Improper Call"
                Case 2
                    Exit Do
                Case 3
                    Print "Tolerance Reset"
                    Print Format(RelErr, "RelErr = 0.000000E+00")
                    Print Format(AbsErr, "AbsErr = 0.000000E+00")
                Case 4
                    Print "Many Steps"
                Case 5
                    AbsErr = AbsErr / 10
                Case 6
                    RelErr = RelErr / 10
                    Iflag% = 2
                Case 7
                    Print "Much Output"
                    Iflag% = 2
                Case 8
                    Print "Improper Call"
                    Exit Sub
            End Select
        Loop
'
'     Print Results
'
        If PrtAll$ <> "Y" Then
            PrtEnt% = 2
            Call ReportRK4(PrtEnt%, Neqn%, X, Y, RelErr, AbsErr)
        End If
Loop While X < Upper

End Sub


Sub ReportRK4(PrtEnt%, Neqn%, X, Y, RelErr, AbsErr)
Select Case PrtEnt%
Case 1
        Print "SOLUTION            with Abs Error =";
        Print Format(AbsErr, "    0.00E+00");
        Print "    RelErr = ";
        Print Format(RelErr, "    0.00E+00")
        Print "***********************************************"
        Print "          X                                  ";
        Print "               Y(i)                           "
```

```
    Case 2
        Print Format(X, "      0.000000E+00");
        For IK = 1 To Neqn%
            Print Format(Y(IK), "      0.000000E+00");
        Next IK
        Print
    End Select
    End Sub


    Public Sub RKF(Neqn%, T, Tout, Y, RelErr, AbsErr, YP, H, F1, F2, F3, F4, F5, SM, SavRE,
    SavAE, NoFunEst, Init, Iflag%, JFalg%, Kflag%, PrtAll$)
    '
    '    Runge-Kutta-Fehlberg Method for Solving IVP-ODE
    '
    '    ReMin = Minimum Relative Error Tolerance
    '    MaxFunEst = Maximun Number of Function Estimation
    '
    ReMin = 0.000000000001
    MaxFunEst = 3000
    '
    ' Check System Parameters
    '
    Call ParameterCheck(ErrFlag%, Neqn%, RelErr, AbsErr, Iflag%, Jflag%, Kflag%, Mflag%, T, Tout,
    SavRE, SavAE, NoFunEst)
    If ErrFlag% = 1 Then Exit Sub
    '
    ' Reset Flags
    '
    Jflag% = Iflag%
    Kflag% = 0
    SavRE = RelErr
    SavAE = AbsErr
    '
    ' Evaluate Machine Error
    '
    Call MachineErrorCheck(Eps, U20, ReMin, RelErr, Iflag%, Kflag%, ErrFlag%)
    If ErrFlag% = 1 Then Exit Sub
    '
    ' End Point Check
    '
    DT = Tout - T
    Chk% = 0

    If Mflag% = 1 Then
        Init = 0
        CountOperation = 0
```

```
        XA = T
        Call DeriveFunction(XA, Y, YP)
        NoFunEst = 1

        If T = Tout Then
            Iflag% = 2
            Exit Sub
        End If
    ElseIf Init <> 0 Then
        Chk% = 1
    End If

    If Chk% = 0 Then
        Init = 1
        H = Abs(DT)
        TolNew = 0

        For K = 1 To Neqn%
            Tol = RelErr * Abs(Y(K)) + AbsErr
            If Tol > 0 Then
                TolNew = Tol
                YPK = Abs(YP(K))
                If (YPK * H ^ 5 > Tol) Then H = (Tol / YPK) ^ 0.2
                If TolNew <= 0 Then H = 0
                H = AMax(H, U20 * AMax(Abs(T), Abs(DT)))
            End If
        Next K

        Jflag% = Sign(2, Iflag%)
    End If

    H = Sign(H, DT)

    If Abs(H) >= 2 * Abs(DT) Then CountOperation = CountOperation + 1

    If CountOperation = 100 Then
        CountOperation = 0
        Iflag% = 7
        Exit Sub
    End If

    If Abs(DT) <= U20 * Abs(T) Then
        For K = 1 To Neqn%
            Y(K) = Y(K) + DT * YP(K)
        Next K

        XA = Tout
        Call DeriveFunction(XA, Y, YP)
```

```
        NoFunEst = NoFunEst + 1
        T = Tout
        Iflag% = 2
        Exit Sub
End If

Out$ = "FALSE"
RelScale = 2 / RelErr
AE = RelScale * AbsErr
'
' Perform Integration
'
Do
        Call Integration(Neqn%, Y, T, Tout, U20, H, YP, F1, F2, F3, F4, F5, SM, AE, RelErr, AbsErr,
NoFunEst, MaxFunEst, RelScale, Out$, OutSub%, Iflag%, Jflag%, Kflag%, Mflag%)
        If PrtAll$ = "Y" Then
                PrtEnt% = 2
                Call ReportRK4(PrtEnt%, Neqn%, T, Y, RelErr, AbsErr)
        End If
        If OutSub% = 1 Then Exit Sub
Loop While Iflag% > 0

Iflag% = -2

End Sub


Public Sub Integration(Neqn%, Y, T, Tout, U20, H, YP, F1, F2, F3, F4, F5, SM, AE, RelErr,
AbsErr, NoFunEst, MaxFunEst, RelScale, Out$, OutSub%, Iflag%, Jflag%, Kflag%, Mflag%)
'
'    Subroutine for Determing Proper Step Size
'    and Perform RKF Integration
'
Dim ErrTrun(10) As Double

OutSub% = 0
Hfaild$ = "FALSE"

Hmin = U20 * Abs(T)
DT = Tout - T
'
'    Final Step Control
'
If Abs(DT) <= 2 * Abs(H) Then
        If Abs(DT) > Abs(H) Then
                H = 0.5 * DT
        Else
                H = DT
```

```
            Out$ = "TRUE"
        End If
    End If

    Do

        If NoFunEst > MaxFunEst Then
'
'           Too Many Steps
'
            Iflag% = 4
            Kflag% = 4
            OutSub% = 1
            Exit Sub

        Else
'
'           Call RKF Module
'
            Call FEHL(Neqn%, Y, T, H, YP, F1, F2, F3, F4, F5, SM, ErrTrun)
            EeOet = 0
'
'           Estimate Maximun Truncation Error
'
            For K = 1 To Neqn%
                F1(K) = SM(K)
                ET = Abs(Y(K)) + Abs(F1(K)) + AE

                If ET <= 0 Then
                    Iflag% = 5
                    OutSub% = 1
                    Exit Sub
                Else
                    EE = ErrTrun(K) / ET * Abs(H) * RelScale
                    EeOet = AMax(EeOet, EE)
                End If
            Next K

        End If

        If T >= Tout Then Out$ = "TRUE"
        Chk% = 0

        If (EeOet <= 1) Then
'
'           Error Within Bound
'
            Chk% = 1
```

```
                T = T + H
                For K = 1 To Neqn%
                    Y(K) = F1(K)
                Next K

                XA = T
                Call DeriveFunction(XA, Y, YP)
                NoFunEst = NoFunEst + 1
'
'           Optimize Step Size
'
                S = 5
                If (EeOet > 0.0001889568) Then S = 0.9 / EeOet ^ 0.2
                If Hfaild$ = "TRUE" Then S = AMin(S, 1#)
                H = Sign(AMax(S * Abs(H), Hmin), H)
'
'           Check for Integration Completion for this Call
'
                If Out$ = "TRUE" Then
                    T = Tout
                    Iflag% = 2
                    OutSub% = 1
                    Exit Sub
                End If
                Exit Do
        Else
'
'           Error Out of Bound
'           Reducing Step Size
'
                Hfaild$ = "TRUE"
                Out$ = "FALSE"
                S = 0.1
                If (EeOet < 59049) Then S = 0.9 / EeOet ^ 0.2
                H = H * S
        End If

    Loop While Abs(H) > Hmin

    If Chk% = 0 Then
        Iflag% = 6
        Kflag% = 6
        OutSub% = 1
    End If

    End Sub
```

```
Public Sub FEHL(Neqn%, Y, T, StepH, YP, F1, F2, F3, F4, F5, SM, ErrTrun)
'
'    Pass#0
'    K1 = YP
'
Call DeriveFunction(T, Y, YP)
'
'    Pass #1
'    K2 = F1
'
CurrentH = StepH / 4
For K = 1 To Neqn%
        F5(K) = Y(K) + CurrentH * YP(K)
Next K
Call DeriveFunction(T + CurrentH, F5, F1)
'
'    Pass #2
'    K3 = F2
'
CurrentH = 3 * StepH / 32
For K = 1 To Neqn%
        F5(K) = Y(K) + CurrentH * (YP(K) + 3 * F1(K))
Next K
Call DeriveFunction(T + 3 * StepH / 8, F5, F2)
'
'    Pass #3
'    K4 = F3
'
CurrentH = StepH / 2197
For K = 1 To Neqn%
        F5(K) = Y(K) + CurrentH * (1932 * YP(K) - 7200 * F1(K) + 7296 * F2(K))
Next K
Call DeriveFunction(T + 12 * StepH / 13, F5, F3)
'
'    Pass #4
'    K5 = F4
'
CurrentH = StepH / 4104
For K = 1 To Neqn%
        F5(K) = Y(K) + CurrentH * (8341 * YP(K) - 32832 * F1(K) + 29440 * F2(K) - 845 * F3(K))
Next K
Call DeriveFunction(T + StepH, F5, F4)
'
'    Pass #5
'    K6 = F5
'
CurrentH = StepH / 20520
For K = 1 To Neqn%
```

```
        SM(K) = Y(K) + CurrentH * (-6080 * YP(K) + 41040 * F1(K) - 28352 * F2(K) + 9295 * F3(K) -
5643 * F4(K))
Next K
Call DeriveFunction(T + StepH / 2, SM, F5)
'
'    Pass #6
'    Ui+1 = SM
'
CurrentH = StepH / 7618050
For K = 1 To Neqn%
        SM(K) = Y(K) + CurrentH * ((902880 * YP(K) - 1371249 * F4(K) + 3855735 * F3(K) + 3953664 *
F2(K) + 277020 * F5(K)))
Next K
'
'    Estimate Truncation Error ErrTrun
'
For K = 1 To Neqn%
ErrTrun(K) = Abs(YP(K) / 360 - 128 * F2(K) / 4275 - 2197 * F3(K) / 75240 + F4(K) / 50 + 2 * F5(K) /
55)
Next K
End Sub


Public Sub ParameterCheck(ErrFlag%, Neqn%, RelErr, AbsErr, Iflag%, Jflag%, Kflag%, Mflag%,
T, Tout, SavRE, SavAE, NoFunEst)
'
' Check Basic Parameters & Data
'
If Neqn% <= 0 Or RelErr < 0 Or AbsErr < 0 Then
        Iflag% = 8
        Exit Sub
Else
        Mflag% = Abs(Iflag%)
End If
'
' Check for Error Flag Out of Bounds
'
If Mflag% <= 0 Or Mflag% > 8 Then
        Iflag% = 8
        Exit Sub
ElseIf Mflag% <> 1 Then

        If T = Tout And Kflag% <> 3 Then
                Iflag% = 8
                Exit Sub
        End If

        If (Kflag% = 5 And AbsErr = 0) Or (Kflag% = 6 And RelErr <= SavRE And AbsErr <= SavAE)
```

```
Then
        ErrFlag% = 1
        Exit Sub
    End If

    If (Mflag% <> 2 And Ifalg% = 4) Or Kflag% = 4 Then
        NoFunEst = 0
    End If

    If (Kflag% = 3 Or Init = 0) Or (Mflag% <> 2 And (Iflag% = 3 Or Iflag% = 5 And AbsErr > 0 Or
NoFunEst = 0)) Then
        Iflag% = Jflag%
        If (Kflag% = 3) Then Mflag% = Abs(Ifalg%)
    End If

End If

End Sub


Public Sub MachineErrorCheck(Eps, U20, ReMin, RelErr, Iflag%, Kflag%, ErrFlag%)
'
'    Calculate Machine Round Off Error
'
Eps = 1#
Do
    Eps = Eps / 2
    EpsP1 = Eps + 1
Loop While EpsP1 > 1
U20 = 20 * Eps

Rer = 2 * Eps + ReMin
'
'   Check for proper error tolerance
'
If RelErr < Rer Then
    RelErr = Rer
    Iflag% = 3
    Kflag% = 3
    ErrFlag% = 1
End If
End Sub


Public Function Min(A, B)
If A > B Then
    Min = B
Else
```

```
        Min = A
End If
End Function


Public Function Max(A, B)
If A > B Then
        Max = A
Else
        Max = B
End If
End Function


Public Function AMax(A, B)
If Abs(A) > Abs(B) Then
        AMax = (A)
Else
        AMax = (B)
End If
End Function


Public Function AMin(A, B)
If Abs(A) > Abs(B) Then
        AMin = (B)
Else
        AMin = (A)
End If
End Function


Public Function Sign(A, B)
If B > 0 Then
        Sign = A
ElseIf B = 0 Then
        Sign = 0
Else
        Sign = -A
End If
End Function
'
' Program Developed by Ron Hsin Chang, Ph. D.
'Copyright Reserved, 2001
'
```

副程式使用說明

1. 副程式 **Private Sub DeriveFunction(T, Y, YP)** 為使用者定義之微分方程式，其基本寫法如下：

```
Sub DeriveFunction(T, Y, YP)
YP(1) = -0.4 * Y(1) * Y(4)
YP(2) = -YP(1) - 0.2 * Y(2) * Y(4)
YP(3) = 0.2 * Y(2) * Y(4)
YP(4) = YP(1) - YP(3) - 0.05 * Y(4) * Y(4)
End Sub
```

2. 副程式 **Public Sub RKF2000(Neqn%, Lower, Upper, H0, Y0, AbsErr, RelErr, NP)** 爲使用者主要呼叫之 RKF 副程式。使用者只要輸入所要求解的方程式數目 Neqn%，積分下限 Lower，積分上限 Upper，最初積分間距 (Step Size)H0，起始條件 Y0(I)，絕對誤差 AbsErr，相對誤差 RelErr，及預定列印區間數 NP，然後，呼叫 RKF2000 即可。

3. 副程式 **Public Sub ReportRK4(PrtEnt%, Neqn%, X, Y, RelErr, AbsErr)** 爲解答列印副程式，由 RKF2000 驅動。

4. 副程式 **Public Sub RKF(Neqn%, T, Tout, Y, RelErr, AbsErr, YP, H, F1, F2, F3, F4, F5, SM, SavRE, SavAE, NoFunEst, Init, Iflag%, JFalg%, Kflag%)** 爲 RKF 系統控制副程式，由 RKF2000 驅動。

5. 副程式 **Public Sub Integration(Neqn%, Y, T, Tout, U20, H, YP, F1, F2, F3, F4, F5, SM, AE, RelErr, AbsErr, NoFunEst, MaxFunEst, RelScale, Out$, OutSub%, Iflag%, Jflag%, Kflag%, Mflag%)** 爲 RKF 系統積分作業副程式，由 RKF2000 驅動。

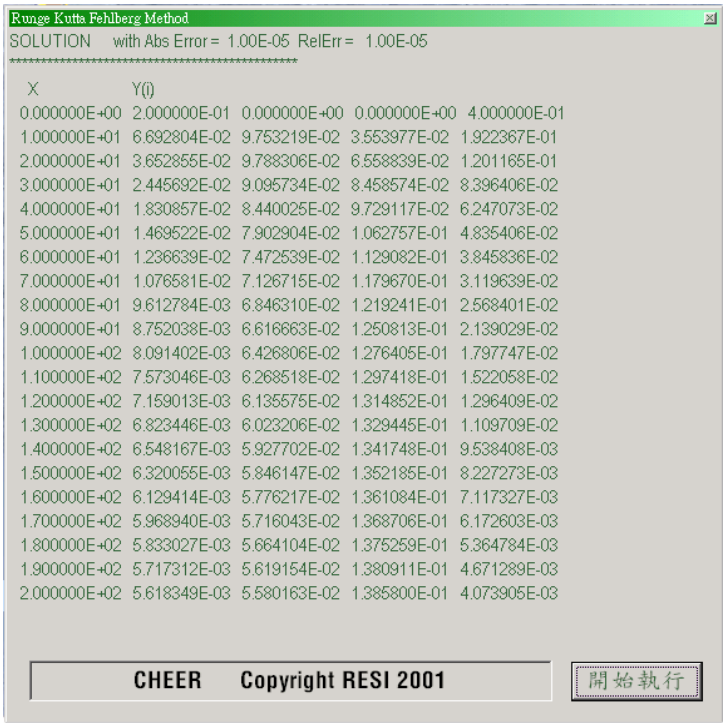
6. 副程式 **Public Sub FEHL(Neqn%, Y, T, StepH, YP, F1, F2, F3, F4, F5, SM, ErrTrun)** 爲 RKF 方法計算副程式，由 RKF2000 驅動。

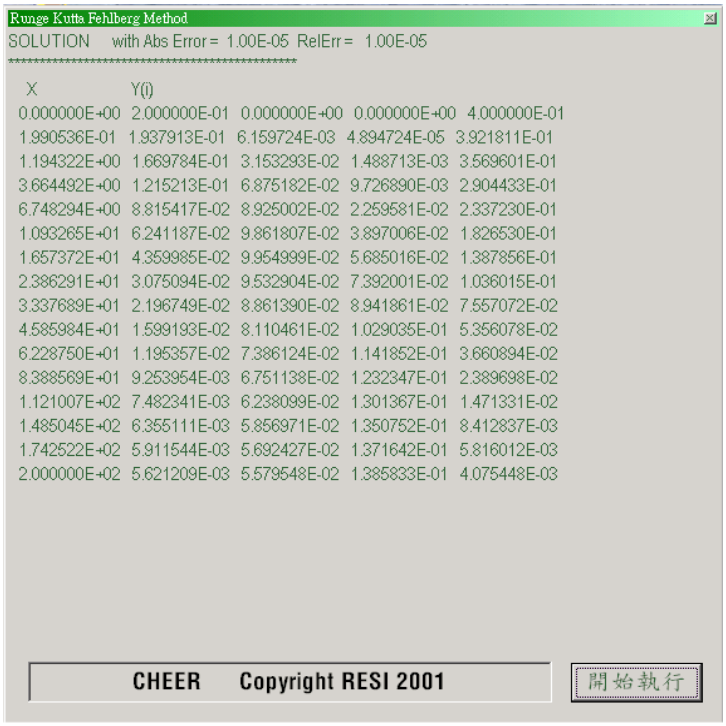
7. 副程式 **Public Sub ParameterCheck(ErrFlag%, Neqn%, RelErr, AbsErr, Iflag%, Jflag%, Kflag%, Mflag%, T, Tout, SavRE, SavAE, NoFunEst)**
   爲 RKF 系統參數檢驗副程式，由 RKF2000 驅動。

8. 副程式 **Public Sub MachineErrorCheck(Eps, U20, ReMin, RelErr, Iflag%, Kflag%, ErrFlag%)**
   爲 RKF 系統計算機計算誤差檢驗及控制副程式，由 RKF2000 驅動。

程式執行結果

典型的計算結果輸出情況如下圖。由圖中可看出，第一圖爲分割成 20 區間輸出的結果，第二圖爲自動變區間積分所得到的結果。由自動變區間積分所得到的結果，可以看到 RKF 程式積分時，積分間距由小轉大，快速調節的情況，可見其效率相當良好。





第四節　多間距積分法

Visual Basic

阮奇庫塔法（包括低階的歐以勒法）由於只需使用前一間距的計算值，因此，稱爲單間距積分法 (Single step methods)。這類方法只要有起始條件就可執行下一步驟的計算，而且積分間距可以任意改變，因此，是開始求解微分方程式的最佳工具。但是一旦利用這類方法求得部分資料以後，我們對所積分的函數（及其導函數）事實上已經有了額外的資料，此時如果能夠善用計算機的記憶，利用需要較多資料的多間距積分法，讓積分效率及穩定性都進一步提昇，當是明智之舉。

多間距積分法的基本原理是利用以前所獲得的 y 及 y' 資料建立導函數的近似多項式，並外插至下一積分間距。爲了數學處理方便起見，大部分的多間距積分方法都取用等積分間距，主要是讓多項式更易於建立，例如亞當斯法 (Adams method) 即爲典型的例子。使用多間距積分法的時候，所使用的已知數據點點數，會決定所使用多項式的次數，因此，也決定截尾誤差的階次。多間距積分法的階次，等於全部計算誤差中 h 的冪次，也等於近似多項式的階次加一。

亞當斯法基本上是利用數值積分的原理建立的。首先將微分方程式 $y' = f(x, y)$ 由 x_i 積分至 x_{i+1}，得到

$$\int_{x_i}^{x_{i+1}} dy = y_{n+1} - y_n = \int_{x_i}^{x_{i+1}} f(x, y)dx \qquad \textbf{(8-4.1)}$$

此方程式的右側積分式中，函數 $f(x, y)$ 可以利用已知的 k 個點 $x_i, x_{i-1}, \cdots, x_{i-k+1}$，建立一個多項式近似之。若利用牛頓內插法表示 $f(x, y(x))$，則可以得到 k 間距的亞當斯 - 貝希佛斯方程式 (Adams-Bashforth formula)，其型式爲

$$u_{i+1} = u_i + h\sum_{j=1}^{k} \beta_j u'_{i-j+1} \qquad \textbf{(8-4.2)}$$

其中 $u'_j = f(x_j, u_j)$。若只考慮三個間距的表示式，則 (8-4.2) 又可以進一步簡化成

$$u_{i+1} = u_i + h〔\beta_1 u'_i + \beta_2 u'_{i-1} + \beta_3 u'_{i-2}〕 \qquad \textbf{(8-4.3)}$$

利用泰勒級數將 u'_{i-1} 及 u'_{i-2} 對 x_i 展開，得到

$$u'_{i-1} = u'_i - hu''_i + \frac{h^2}{2!}u'''_i + \cdots$$

$$u'_{i-2} = u'_1 - 2hu''_i + \frac{(2h)^2}{2!}u'''_i + \cdots$$

代回方程式 (8-4.3)，整理後可以得到

$$u_{i+1} = u_i + hu'_i(\beta_1 + \beta_2 + \beta_3) - h^2 u''_i(\beta_2 + 2\beta_3) + \frac{h^3}{2!}u'''_i(\beta_2 + 4\beta_3) + \cdots \qquad \textbf{(8-4.4)}$$

方程式 (8-4.4) 與 u_{i+1} 的泰勒級數展開式比較

$$u_{i+1} = u_i + hu'_i + \frac{h^2}{2!}u''_i + \frac{h^3}{3!}u'''_i + \cdots \qquad \textbf{(8-4.5)}$$

可以得到 β 的聯立方程式

$$\begin{cases} \beta_1 + \beta_2 + \beta_3 = 1 \\ \beta_2 + 2\beta_3 = -\frac{1}{2} \\ \beta_2 + 4\beta_3 = \frac{1}{3} \end{cases}$$

這組線性方程式的解爲 $\beta_1 = \frac{23}{12}, \beta_2 = -\frac{16}{12}$ 及 $\beta_3 = \frac{5}{12}$。因此，三階亞當斯—貝希佛斯法可寫成

$$u_{i+1} = u_i + \frac{h}{12}\left[23u_i - 16u_{i-1} + 5u'_{i-2}\right] + 0(h^4) \qquad \textbf{(8-4.6)}$$

常見的亞當斯法如下表所示：

k	亞當斯貝希佛斯方程式	截尾誤差
1	$u_{i+1} = u_i + hu'_i$	$0(h^2)$
2	$u_{i+1} = u_i + \frac{h}{2}\left[3\,u'_i - u'_{i-1}\right]$	$0(h^3)$
3	$u_{i+1} = u_i + \frac{h}{12}\left[23\,u'_i - 16\,u'_{i-1} + 5\,u'_{i-2}\right]$	$0(h^4)$
4	$u_{i+1} = u_i + \frac{h}{24}\left[55\,u'_i - 59\,u'_{i-1} + 37\,u'_{i-2} - 9\,u'_{i-3}\right]$	$0(h^5)$
5	$u_{i+1} = u_i + \frac{h}{720}\left[1901\,u'_i - 2774\,u'_{i-1} + 2616\,u'_{i-2} - 1274\,u'_{i-3} + 251\,u'_{i-4}\right]$	$0(h^6)$
6	$u_{i+1} = u_i + \frac{h}{1440}\left[4227\,u'_i - 7673\,u'_{i-1} + 9482\,u'_{i-2} - 6798\,u'_{i-3} + 26271\,u'_{i-4} - 425\,u_{i-5}\right]$	$0(h^7)$

多間距積分法使用時，具有無法自動開始啓動運算的困難。習慣上利用兩種方式來克服：

1. 可以利用同準確度的阮奇庫塔法，建立最初所需資料；然後，再接續使用多間距積分法。

2. 利用 s 間距亞當斯法，由 $s = 1, 2, \cdots$ 至 $s = k$ ，依序執行。

其中尤其以第 **2.** 種策略的積分效率更佳。利用亞當斯法的積分間距改變策略及程式設計方法請參閱 Shampine 及 Gordon 的專書 [13]。

例題 8-6　亞當斯 - 貝希佛斯法

利用方程式 (8-4.6) 解微分方程式

$$y' = x + y; \quad \text{I.C.} \quad y(0) = 1$$

x	u	$u' = f(x, y) = x + u$
0	1	1.0
0.2	1.2428	1.4428
0.4	1.5836	1.9836

假設我們已知 $y(0.2)$ 及 $y(0.4)$ 的計算值 (可利用阮奇庫塔法求得)，利用積分間距 $h=0.2$ ，試求 $y(0.6)$ 的值。

解：

由方程式 (8-4.4)

$$\begin{aligned} u(0.6) &= u(0.4)+\frac{h}{12}\left[23\,u'(0.4)-16\,u'(0.2)+5\,u'(0)\right] \\ &= 1.5836+\frac{0.2}{12}\left[23\times 1.9836-16\times 1.4428+5\times 1.0\right] \\ &= 2.0426 \end{aligned}$$

【註】理論值 2.04424

第五節　亞當斯默頓法

Visual Basic

亞當斯貝希佛斯法基本上是利用已知的有限資料，推算下一未知資料的方法，是屬於顯式積分法 (Explicit Method)，因此，可以推論其穩定性可能仍然較差。若能使用未知資料與已知資料，一起建立方程式，再利用數學方法求解，以得到下一推估資料，即屬於隱式積分法 (Implicit Method)。隱式積分法通常可以克服穩定性問題；但一般而言，隱式積分法的程式規劃設計將較爲複雜。顯式多間距數值積分法由於穩定性差，對於複雜的科學運算，較少被使用。通常，商業化的程式大部分都是利用隱式積分法設計，並且作複雜的積分間距調控處理，以加速積分效率。

隱式多間距積分法的縯導方法與上一節所介紹的方法相當接近，首先將方程式 (8-4.2) 改寫成

$$\sum_{m=0}^{k_1}\alpha_m u_{n-m+1}+\sum_{m=0}^{k_2}\beta_m u'_{n-m+1}=0 \qquad \textbf{(8-5.1)}$$

若 $\beta_0=0$ ，稱爲顯式積分法，若 $\beta_0\neq 0$ 則稱爲隱式積分法。若 $\alpha_0=-\alpha_1$ 即稱爲**亞當斯法。顯式亞當斯法稱為亞當斯貝希佛斯法；隱式亞當斯法則稱為亞當斯默頓法。**

考慮 $k_1=1, k_2=3$ 的亞當斯默頓法，方程式 (8-5.1) 可以被改寫成

$$u_{i+1}=u_i+(\beta_0 u'_{i+1}+\beta_1 u'_i+\beta_2 u'_{i-1}+\beta_3 u'_{i-2}) \qquad \textbf{(8-5.2)}$$

仿上一節的縯導方式，將 $u'_{i+1}, u'_i, u'_{i-1}, u'_{i-2}$ 的泰勒級數展開式代入上式中，可以得到

$$u_{i+1} = u_i + u_i'(\beta_0 + \beta_1 + \beta_2 + \beta_3) + hu_i''(\beta_0 - \beta_2 - 2\beta_3) + \frac{h^2}{2}u_i'''(\beta_0 + \beta_2 + 4\beta_3) + \frac{h^3}{3!}u_i^{iv}(\beta_0 - \beta_2 - 8\beta_3) + \cdots \quad \textbf{(8-5.3)}$$

此方程式與u_{i+1}的泰勒級數展開方程式比較，

$$u_{i+1} = u_i + hu_i' + \frac{h^2}{2!}u_i'' + \frac{h^3}{3!}u_i''' + \frac{h^4}{4!}u^{iv} + \cdots \quad \textbf{(8-5.4)}$$

可以得到β_i的線性聯立方程式爲

$$\begin{cases} \beta_0 + \beta_1 + \beta_2 + \beta_3 = h & \beta_0 - \beta_2 - 2\beta_3 = \frac{1}{2}h \\ \beta_0 + \beta_2 + 4\beta_3 = \frac{1}{3}h & \beta_0 - \beta_2 - 8\beta_3 = \frac{1}{4}h \end{cases}$$

解此聯立方程式，得到

$$\beta_0 = \frac{9}{24}h \ ， \ \beta_1 = \frac{19}{24}h \ ， \ \beta_2 = \frac{-5}{24}h \ ， \ \beta_3 = \frac{1}{24}h$$

代回方程式 (8-5.2)，即得到亞當斯默頓修正式 (Adams-Moulton corrector) 爲

$$u_{i+1} = u_i + \frac{h}{24}〔9u_{i+1}' + 19u_i' - 5u_{i-1}' + u_{i-2}'〕 \quad \textbf{(8-5.5)}$$
$$\varepsilon_T = 0(h^5)$$

仿這種方法可以建立不同階次的亞當斯默頓修正式，歸納如下。

k	亞當斯默頓方程式	截尾誤差
1	$u_i = u_{i-1} + hu_i'$	$0(h^2)$
2	$u_i = u_{i-1} + \frac{h}{2}〔u_i' + u_{i-1}'〕$	$0(h^3)$
3	$u_i = u_{i-1} + \frac{h}{12}〔5u_i' + 8u_{i-1}' - u_{i-2}'〕$	$0(h^4)$
4	$u_i = u_{i-1} + \frac{h}{24}〔9u_i' + 19u_{i-1}' - 5u_{i-2}' + u_{i-3}'〕$	$0(h^5)$
5	$u_i = u_{i-1} + \frac{h}{720}〔251u_i' + 646u_{i-1}' - 264u_{i-2}' + 106u_{i-3}' - 19u_{i-4}'〕$	$0(h^6)$
6	$u_i = u_{i-1} + \frac{h}{1440}〔475u_i' + 1427u_{i-1}' - 798u_{i-2}' + 482u_{i-3}' - 173u_{i-4}' + 27u_{i-5}〕$	$0(h^7)$

亞當斯默頓法的使用方法，基本上可分成兩大類：

1. 預測修正法

(1) 先利用亞當斯貝希佛斯法做爲預測式，利用 $u_i, u_{i+1}, \cdots, u_{i-k+1}$ 等資料先求出 u_{i+1}。

(2) 再利用亞當斯默頓法做爲修正式，求出更正確的 u_{i+1} 值。

2. 迭代計算法

直接利用亞當斯默頓法作計算。由於 $u'_{i+1} = f(x_{i+1}, u_{i+1})$，因此，方程式 (8-5.4) 可視爲 u'_{i+1} 的非線性 (或線性) 方程式，可利用牛頓迭代法求得 u'_{i+1}。

此外，較著名的商業化程式，包括漢明法 (Hamming's Method) 及基爾法 (Gear Method)，分別摘要如下：

1. 漢明法 (Hamming's Method)

$$
\begin{aligned}
u^{*}_{i+1} &= u_{i-3} + \frac{4h}{3}(2u'_i - u'_{i-1} + 2u'_{i-2}) \\
u^{**}_{i+1} &= u^{*}_{i+1} - \frac{112}{121}(u^{*}_{i} - u^{***}_{i}) \\
u^{***}_{i+1} &= \frac{1}{8}[9u_i - u_{i-2} + 3h(u^{**'}_{i+1} + 2u'_i - u'_{i-1})] \\
u_{i+1} &= u^{***}_{i+1} + \frac{9}{121}(u^{*}_{i+1} - u^{**}_{i+1})
\end{aligned}
\qquad \textbf{(8-5.6)}
$$

商業化程式名：IBM 數學副程式 HPCG 及 HPCL。

2. 基爾法 (Gear Method)

$$u_{i+1} = u^{*}_{i+1} - \frac{95}{288}F \qquad \textbf{(8-5.7)}$$

其中 $u^{*}_{i+1} = -18u_i + 9u^{*}_{i-1,c} + 10u^{**}_{i-2,c} + 9hu'_i + 18hu'_{i-1} + 3hu'_{i-2}$

$hu^{*}_{i+1} = -57u_i + 24u^{*}_{i-1,c} + 33u^{**}_{i-2,c} + 24hu'_i + 57hu'_{i-1} + 10hu'_{i-2}$

$F = hu^{*'}_{i+1} - hu'(x_{i+1}, u^{*}_{i+1})$

$u^{*}_{i,c} = u_i + \frac{3}{160}F$

$u^{**}_{i-1,c} = u^{*}_{i-1,c} - \frac{11}{1440}F$

$$hu'_{i+1} = hu^{*'}_{i+1} - F$$

商業化程式名：GEAR、GEARB、EPISODE、EPISODEB

基爾法尤其適用於所謂的「棘手 (stiff)」問題。所謂的棘手度 (stiffness) 是指一組微分方程式的積分困難程度，通常利用方程式 Jacobian 矩陣的特徵值 (Eigenvalue) 做判斷。

第六節　特徵值與棘手度

Visual Basic

考慮一組聯立方程式

$$\begin{cases} \dfrac{dy_1}{dx} = a_{11}y_1 + a_{12}y_2 \\ \dfrac{dy_2}{dx} = a_{21}y_1 + a_{22}y_2 \end{cases} \quad \text{I.C.} \quad @\, x = 0 \quad y_1 = 1 \quad y_2 = 0 \qquad \textbf{(8-6.1)}$$

其 Jacobian 矩陣爲

$$\underline{\underline{Q}} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \qquad \frac{d\,\underline{y}}{dx} = \underline{\underline{Q}}\,\underline{y} \qquad \textbf{(8-6.2)}$$

Jacobian 矩陣的特徵值 λ 爲以下方程式的解

$$\begin{bmatrix} a_{11} - \lambda & a_{12} \\ a_{21} & a_{22} - \lambda \end{bmatrix} = 0 \qquad \textbf{(8-6.3)}$$

或將此方程式展開成爲

$$\lambda^2 - (a_{11} + a_{22})\lambda + (a_{11}a_{22} - a_{12}a_{21}) = 0$$

定義棘手度 (Ψ, Stiffness Ratio) 爲

$$\Psi = \frac{\max\left|\,\lambda \text{ 之實數部份}\,\right|}{\min\left|\,\lambda \text{ 之實數部分}\,\right|} \qquad \lambda \text{ 爲 Jacobian 矩陣之特徵值} \qquad \textbf{(8-6.4)}$$

$\Psi < 10$　　非棘手的一般問題

$\Psi > 100$　　棘手的問題

$\Psi > 100000$　非常棘手的問題

例題 8-7 連續反應系統

考慮一連續化學反應 $\boldsymbol{A \leftrightarrow B \rightarrow C}$ ，其中反應物 A 反應生成中間產物 B 的反應速率常數爲 $k_1 = 1000$ ，中間產物 B 反應生成 A 的反應速率常數爲 $k_2 = 1$ ，中間產物 B 反應生成產物 C 的反應速率常數爲 $k_3 = 1$ 。試建立其微分方程式，並判斷此微分方程組的棘手度。

解：

反應器內的濃度變化可以用以下的聯立方程式表示：

$$\begin{cases} \dfrac{dC_A}{dt} = -k_1 C_A + k_2 C_B \\ \dfrac{dC_B}{dt} = k_1 C_A - (k_2 + k_3) C_B \end{cases} \qquad @\, t = 0 \quad C_A = C_{A0} \quad C_B = 0$$

將濃度 C_A 及 C_B 無因次化，可將原微分方程式改寫成：

$$\frac{d\underline{y}}{dt} = \underline{\underline{Q}}\,\underline{y} \qquad \underline{y}(0) = [1 \quad 0]^T$$

$$\underline{y} = \begin{bmatrix} \dfrac{C_A}{C_{A0}} & \dfrac{C_B}{C_{A0}} \end{bmatrix}^T \qquad \underline{\underline{Q}} = \begin{bmatrix} -k_1 & k_2 \\ k_1 & -(k_2 + k_3) \end{bmatrix}$$

1. 方程式的理論解爲

$$y_1 = \frac{1000}{1001} e^{-1001t} + \frac{1}{1001} e^{-t}$$

$$y_2 = -\frac{1000}{1001} e^{-1001t} + \frac{1000}{1001} e^{-t}$$

其中 y_1 將非常快速的消耗掉，而 y_2 則需要很長的時間才能完成反應。

2. Jacobian 矩陣的特徵值

$$\begin{bmatrix} -1000 - \lambda & 1 \\ 1000 & -2 - \lambda \end{bmatrix} = 0$$

$$\lambda^2 + 1002\lambda + 1000 = 0$$

$$\lambda_1 \cong 1 \qquad \lambda_2 \cong -1001$$

$$\Psi = 1001$$

3. $\Psi > 100$ 爲一棘手問題。

第七節 隱式阮奇庫塔積分法

Visual Basic

阮奇庫塔法爲典型的顯式積分法 (Explicit method)，因此，可以推論其穩定性可能較差。若使用隱式積分法 (Implicit method) 則可克服穩定性問題。

顯式阮奇庫塔法 (Runge-Kutta Method) 的一般式，如本章第二節所介紹，可寫成

$$u_{i+1} = u_i + \sum_{j=1}^{n} \omega_j \; h \, K_j \qquad \textbf{(8-7.1)}$$

其中 ω_j 爲配重因子，n 代表所選取 x_i 至 x_{i+1} 中的點數， K_j 爲 x_i 至 x_{i+1} 中的某一點的函數 f 值，

$$\begin{aligned} &K(x,u) = f(x,u) \qquad x_i < x < x_{i+1} \\ &K_j = f(x_i + c_j \mathrm{h}, u_i + \sum_{s=1}^{j-1} a_{js} \; h \, K_s) \\ &c_1 = 0 \end{aligned} \qquad \textbf{(8-7.2)}$$

隱式阮奇庫塔法則是將方程式 (8-7.2) 中的 K_j 表示式修改成隱式表示式：

$$K_j = f(x_i + c_j \mathrm{h}, u_i + \sum_{s=1}^{j} a_{js} \; h \, K_s) \qquad \textbf{(8-7.3)}$$

注意比較方程式 (8-7.2) 及方程式 (8-7.3)，可發現 $\sum_{s=1}^{j} a_{js} \; h \, K_s$ 的加總上限由 $j-1$ 改成 j。亦即計算 K_j 時，也需要 K_j 本身的資料。因此，才會說將 K_j 變成隱式表示法。

典型的隱式阮奇庫塔法舉例如下：

一階隱式阮奇庫塔法	
1/2	1/2
	1

$$\begin{aligned} &u_{i+1} = u_i + hK_1 \\ &K_1 = f(x_i + \frac{h}{2}, u_i + \frac{h}{2} K_1) \end{aligned} \qquad \textbf{(8-7.4)}$$

二階隱式阮奇庫塔法		
$\frac{1}{2}-\frac{\sqrt{3}}{6}$	$\frac{1}{4}$	$\frac{1}{4}-\frac{\sqrt{3}}{6}$
$\frac{1}{2}+\frac{\sqrt{3}}{6}$	$\frac{1}{4}+\frac{\sqrt{3}}{6}$	$\frac{1}{4}$
	$\frac{1}{2}$	$\frac{1}{2}$

$$u_{i+1}=u_i+\frac{h}{2}K_1+\frac{h}{2}K_2$$
$$K_1=f(x_i+(\frac{1}{2}-\frac{\sqrt{3}}{6})h,u_i+\frac{h}{4}K_1+(\frac{1}{4}-\frac{\sqrt{3}}{6})hK_2) \quad \textbf{(8-7.5)}$$
$$K_2=f(x_i+(\frac{1}{2}+\frac{\sqrt{3}}{6})h,u_i+(\frac{1}{4}+\frac{\sqrt{3}}{6})hK_1+\frac{h}{4}K_2)$$

第八節　半隱式阮奇庫塔法

Visual Basic

一階半隱式阮奇庫塔法

考慮一次常微分方程式 $dy/dx=f(x,y)$ ，其中 $f(x,y)$ 爲 x 及 y 的函數。這種函數關係在物理、化學及工程領域中相當常見。如果將此微分方程式作微分，得到

$$d\left(\frac{dy}{dx}\right)=\frac{\partial f}{\partial x}dx+\frac{\partial f}{\partial y}dy=f_x dx+f_y dy$$

或

$$\frac{d^2y}{dx^2}=f_x+\frac{\partial f}{\partial y}\frac{dy}{dx}=f_x+f_y f$$

同理

$$d\left(\frac{d^2y}{dx^2}\right)=\frac{\partial}{\partial x}(f_x+f_y f)dx+\frac{\partial}{\partial y}(f_x+f_y f)dy$$

或

$$\frac{d^3y}{dx^3}=f_{xx}+f_y\ f_x+(2\ f_{xy}+f_y^2+f\ f_{yy})f$$

將微分方程式對 (x_n, y_n) 作泰勒級數展開，得到

$$y_{n+1} = y_n + hf + \frac{h^2}{2}(f_x + f_y f) + \frac{h^3}{6}[f_{xx} + f_x f_y + (2f_{xy} + f_y^2 + ff_{yy})f] + \cdots O(h^4) \quad \textbf{(8-8.1)}$$

首先考慮最簡單的情況，$dy/dx = f(y)$，其中 $f(y)$ 與 x 無關。當積分間距小到某一情況時，x_n 到 x_{n+1} 間可以看作是一直線，二點間的斜率可以用其中間點的斜率 $f(y_{n+1/2})$ 代表。亦即

$$y_{n+1} = y_n + hf(y_{n+1/2}) \quad \textbf{(8-8.2)}$$

又由常微分方程式 $dy/dx = f(y)$，可以得到

$$\frac{y_{n+1/2} - y_n}{h/2} = f(y_{n+1/2}) \quad \textbf{(8-8.3)}$$

將方程式 (8-8.3) 代入方程式 (8-8.2) 中，可以得到

$$y_{n+1} = y_n + 2(y_{n+1/2} - y_n) \quad \textbf{(8-8.4)}$$

而由方程式 (8-8.1) 取到截取誤差爲 $O(h^3)$，則得到

$$\begin{aligned} y_{n+1} &= y_n + hf + \frac{h^2}{2} f_y f \\ &\equiv y_n + hf(y_n) + \frac{h^2}{2} f_y f(y_{n+1/2}) \end{aligned} \quad \textbf{(8-8.5)}$$

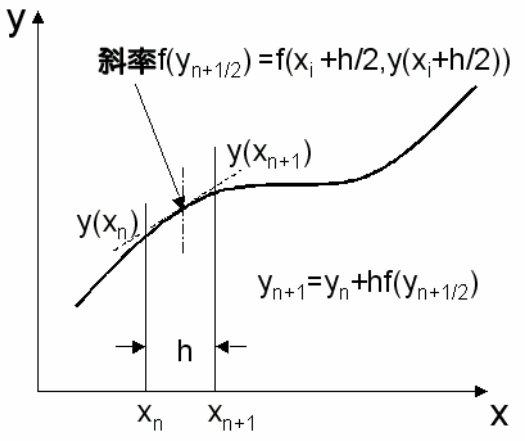


圖 8.4　以中點斜率作為近似值

將方程式 (8-8.3) 及方程式 (8-8.4) 代入方程式 (8-8.5) 中，得到

$$y_{n+1/2} - y_n = \frac{hf}{2 - hf_y} \tag{8-8.6}$$

代回方程式 (8-8.4)，得到

$$y_{n+1} = y_n + \frac{h\ f}{1 - \frac{h}{2} f_y} \tag{8-8.7}$$

將以上方程式利用泰勒級數展開爲 $y_{n+1} = y_n + hf + \frac{h^2}{2} f_y f + \frac{h^3}{4} f_y^2 f + \cdots$，與方程式 (8-8.1) 比較，其誤差爲 $O(h^3)$。

將方程式 (8-8.7) 寫成阮奇庫塔法的方式，可以得到一階半隱式阮奇庫塔法表示式爲

$$\begin{aligned} u_{i+1} &= u_i + K_1 \\ K_1 &= (1 - \frac{h}{2} f_y)^{-1} hf \\ E_t &= O(h^3) \end{aligned} \tag{8-8.8}$$

對於 M 個聯立方程式 $d\underline{y}/dx = \underline{f}(x, \underline{y})$，其中 $\underline{f}(x, \underline{y})$ 爲 x 及 $\underline{y}$ 的函數，可以仿照方程式 (8-8.8) 的方式，寫成

$$\begin{aligned} \underline{u}_{n+1} &= \underline{u}_n + \underline{K}_1 \\ \underline{K}_1 &= (\underline{\underline{I}} - \frac{h}{2} \underline{\underline{J}})^{-1} h \underline{f}(x, \underline{u}_n) \\ J_{ij} &= (\frac{\partial f_i}{\partial y_j})_{\underline{y}_n} \\ E_t &= O(h^3) \end{aligned} \tag{8-8.9}$$

二階半隱式阮奇庫塔法

仿以上的做法，假設考慮最簡單的情況，$dy/dx = f(y)$，且其中 $f(y)$ 與 x 無關；利用與上一小節相同的方法，可以擴展到較高階的表示式。將此微分方程式作微分，得到

$$d\left(\frac{dy}{dx}\right)=\frac{\partial f}{\partial x}dx+\frac{\partial f}{\partial y}dy=0+f_y dy=f_y dy$$

或

$$\frac{d^2y}{dx^2}=f_x+\frac{\partial f}{\partial y}\frac{dy}{dx}=0+f_y f=f_y f \qquad \textbf{(8-8.10)}$$

同理

$$d\left(\frac{d^2y}{dx^2}\right)=\frac{\partial}{\partial x}(f_x+f_y f)dx+\frac{\partial}{\partial y}(f_x+f_y f)dy$$

或

$$\frac{d^3y}{dx^3}=f_y^2 f+f^2\ f_{yy} \qquad \textbf{(8-8.11)}$$

將微分方程式對(x_n, y_n)作泰勒級數展開，得到

$$y_{n+1}=y_n+hf+\frac{h^2}{2}f_y f+\frac{h^3}{6}(f_y^2 f+f^2 f_{yy})+\cdots O(h^4) \qquad \textbf{(8-8.12)}$$

假設最簡單的二階半隱式阮奇庫塔法表示式爲

$$\begin{aligned} u_{i+1}&=u_i+\omega_1 K_1+\omega_2 K_2\\ K_1&=(1-ahf_y)^{-1}hf\\ K_2&=(1-ahf_y)^{-1}K_1 \end{aligned} \qquad \textbf{(8-8.13)}$$

將K_1及K_2的分式展開，得到

$$K_1=hf(1+ahf_y+a^2h^2f_y^2+\cdots) \qquad \textbf{(8-8.14)}$$

$$K_2=hf(1+2ahf_y+3a^2h^2f_y^2+\cdots) \qquad \textbf{(8-8.15)}$$

代入u_{i+1}的表示式，得到

$$\begin{aligned} u_{i+1}&=u_i+\omega_1 K_1+\omega_2 K_2\\ &=u_i+(\omega_1+\omega_2)hf+(\omega_1+2\omega_2)h^2af_y f+O(h^3) \end{aligned} \qquad \textbf{(8-8.16)}$$

與方程式 (8-8.12) 比較，得到$\omega_1+\omega_2=1$, $\omega_1+2\omega_2=\frac{1}{2a}$。解之，得到

$$\omega_1=\frac{4a-1}{2a},\ \omega_2=\frac{1-2a}{2a} \qquad \textbf{(8-8.17)}$$

爲了簡化分析，以取得適當參數，考慮方程式$dy/dx=f(y)=\lambda y$，方程式 (8-8.13)

可以寫成

$$K_1 = (1 - ahf_y)^{-1} hf = \frac{h\lambda u_i}{(1 - ah\lambda)}$$

$$K_2 = (1 - ahf_y)^{-1} K_1 = \frac{h\lambda u_i}{(1 - ah\lambda)^2}$$

代入方程式 (8-8.16)，並作整理，得到

$$\begin{aligned} u_{i+1} &= u_i + \omega_1 K_1 + \omega_2 K_2 \\ &= u_i [1 + \frac{4a-1}{2a} \frac{h\lambda}{(1 - ah\lambda)} + \frac{1-2a}{2a} \frac{h\lambda}{(1 - ah\lambda)^2}] \\ &= \frac{1 + (1-2a)h\lambda + (a^2 - 2a + \frac{1}{2})(h\lambda)^2}{(1 - ah\lambda)^2} u_i \end{aligned} \quad \textbf{(8-8.18)}$$

要提高積分效率，要求在 $h\lambda$ 較大時，仍能得到高準確度，可令以上方程式中的 $(h\lambda)^2$ 項係數永遠爲零；即 $a^2 - 2a + \frac{1}{2} = 0$ ，其解爲 $a = 1 \pm \sqrt{\frac{1}{2}}$ 。又當 $h\lambda$ 較小時，若也要有良好的積分效率，則所得結果以 $a = 1 - \sqrt{\frac{1}{2}}$ 較佳。因此，代回方程式 (8-8.17)，得到

$$a = 1 - \frac{\sqrt{2}}{2} \qquad \omega_1 = 1 - \frac{\sqrt{2}}{2} \qquad \omega_2 = \frac{\sqrt{2}}{2}$$

代回方程式 (8-8.13)，得到

$$\begin{aligned} u_{i+1} &= u_i + (1 - \frac{\sqrt{2}}{2}) K_1 + \frac{\sqrt{2}}{2} K_2 \\ K_1 &= [1 - (1 - \frac{\sqrt{2}}{2}) h\ f_y]^{-1}\ h\ f \\ K_2 &= [1 - (1 - \frac{\sqrt{2}}{2}) h\ f_y]^{-1} K_1 \end{aligned} \quad \textbf{(8-8.19)}$$

羅森布魯克 (Rosenbrock) 半隱式阮奇庫塔法

羅森布魯克將方程式 (8-8.13) 寫成一般式

$$\begin{aligned} u_{i+1} &= u_i + \omega_1 K_1 + \omega_2 K_2 \\ K_1 &= (1 - a_1 hf_y)^{-1} hf \\ K_2 &= [1 - a_2 hf_y (u_i + C_1 K_1)]^{-1} hf (u_i + b_1 K_1) \end{aligned} \quad \textbf{(8-8.20)}$$

仿上節展開整理後，可得到以下幾種不同結果。

參數	梯形積分法	卡拉漢	羅森布魯克	
	Trapezoidal	Calahan	Rosenbrock	
a_1	0.5	0.788 675 134	$1-\sqrt{2}/2$	1.408 248 29
a_2	0.5	0.788 675 134	$1-\sqrt{2}/2$	0.591 751 71
b_1	0	–1.154 700 540	$\sqrt{2}/2-1/2$	0.173 786 67
c_1	0	0	0	0.173 786 67
ω_1	1.0	0.75	0	–0.413 154 32
ω_2	0	0.25	1.0	1.413 154 32
E_t	$O(h^3)$	$O(h^4)$	$O(h^3)$	$O(h^4)$

凱勞得半隱式阮奇庫塔法 (Caillaud & Padmanabhan)

$$\begin{aligned} u_{i+1} &= u_i + \omega_1 K_1 + \omega_2 K_2 + \omega_3 K_3 \\ K_1 &= (1 - a_1 h f_y)^{-1} h f(u_i) \\ K_2 &= (1 - a_1 h f_y)^{-1} h f(u_i + b_2 K_1) \\ K_3 &= (1 - a_1 h f_y)^{-1} h f(b_{31} K_1 + b_{32} K_2) \end{aligned} \qquad \textbf{(8-8.21)}$$

其中 a_1 爲方程式 $a_1^3 - 3a_1^2 + 3/2a_1 - 1/6 = 0$ 的解。

$$a_1 = 0.43586652$$
$$b_2 = 3/4$$
$$b_{31} = -\frac{1}{18} - a_1 - b_{32} = -0.274684$$
$$b_{32} = \frac{4}{3}(\frac{1}{6} - a_1 + a_1^2) = -0.105627$$
$$\omega_1 = \frac{11}{27} \qquad \omega_2 = \frac{16}{27} \qquad \omega_3 = 1$$

密齊森半隱式阮奇庫塔法 (Michelsen)

密齊森修正方程式 (8-8.21) 中 K_3 的表示式，得到

$$\begin{aligned} u_{i+1} &= u_i + \omega_1 K_1 + \omega_2 K_2 + \omega_3 K_3 \\ K_1 &= (1 - a_1 h f_y)^{-1} h f(u_i) \\ K_2 &= (1 - a_1 h f_y)^{-1} h f(u_i + b_2 K_1) \\ K_3 &= (1 - a_1 h f_y)^{-1} (b_{31} K_1 + b_{32} K_2) \end{aligned} \qquad \textbf{(8-8.22)}$$

其中 a_1 爲方程式 $a_1^3 - 3a_1^2 + 3/2a_1 - 1/6 = 0$ 的解。

$$\begin{aligned} a_1 &= 0.435\,866\,521\,508\,4589 \\ b_2 &= 3/4 \\ b_{31} &= -\frac{1}{6a_1}(8a_1^2 - 2a_1 + 1) = -0.630\,202\,088\,724\,4523 \\ b_{32} &= \frac{2}{9a}(6a_1^2 - 6a_1 + 1) = -0.242\,337\,891\,260\,0452 \\ \omega_1 &= \frac{11}{27} - b_{31} = 1.037\,609\,496\,131\,859 \\ \omega_2 &= \frac{16}{27} - b_{32} = 0.834\,930\,483\,852\,6377 \\ \omega_3 &= 1 \end{aligned} \qquad \textbf{(8-8.23)}$$

對於 M 個聯立方程式 $d\underline{y}/dx = \underline{f}(x, \underline{y})$，其中 $\underline{f}(x, \underline{y})$ 爲 x 及 y 的函數；密齊森半隱式阮奇庫塔法爲一穩定的積分程序，使用上每一積分步驟需要計算一個 Jacobian，及計算兩次函數值。可以仿照方程式 (8-8.21) 的方式，寫成

$$\begin{aligned} \underline{u}_{i+1} &= \underline{u}_i + \omega_1 \underline{K}_1 + \omega_2 \underline{K}_2 + \omega_3 \underline{K}_3 \\ \underline{K}_1 &= (\underline{\underline{I}} - a_1 h \underline{\underline{J}})^{-1} h[\underline{f}(x_i, \underline{u}_i) + h a_1 \underline{f}_x] \\ \underline{K}_2 &= (\underline{\underline{I}} - a_1 h \underline{\underline{J}})^{-1} h[f(x_i + b_2 h, u_i + b_2 \underline{K}_1) + h a_1 \underline{f}_x] \\ \underline{K}_3 &= (\underline{\underline{I}} - a_1 h \underline{\underline{J}})^{-1} [b_{31}(\underline{K}_1 + h^2 a_1 \underline{f}_x) + b_{32}(\underline{K}_2 + h^2 a_1 \underline{f}_x)] \end{aligned} \qquad \textbf{(8-8.24)}$$

其中參數 $a_1, b_2, b_{31}, b_{32}, \omega_1, \omega_2$ 均採用方程式 (8-8.23) 的值。

例題 8-8　利用密齊森半隱式阮奇庫塔法再解設計問題 D-VIII

試利用密齊森半隱式阮奇庫塔法重解設計問題 D-VIII，並與例 8-4 結果作比較。

```
'
' Semi-Implicit Runge Kutta Method
' For Initial Value ODE
'
' Program Developed by Dr. Ron Hsin Chang
'
Private Sub SemiImplicitSolver(Xpos, Ypos)
Dim Y0(20), Weight(20), Y(20), DF(20, 20) As Double
Dim Neqn As Integer
Dim NP As Integer
'
' ================== USER'S SPACE
'
Neqn = 4:    ' Number of eqns.
Cls

Print "Number of Equations N = ";
Neqn = Val(InputBox("    Number of Equations = ", "", Neqn, Xpos, Ypos))
Print Neqn

Print "> Enter Limits of Integration:"
Print "Lower Limit of X = ";
Lower = Val(InputBox("    Lower Limit    X0 = ", "Low Limit of X", Lower, Xpos, Ypos))
Print Lower

Print "Upper Limit of X = ";
Upper = Val(InputBox("    Upper Limit    XS = ", "Upper Limit XS", Upper, Xpos, Ypos))
Print Upper

Print "> Initial Step Size = ";
H0 = Val(InputBox("> Initial Step Size = ", "", (Upper - Lower) / 1000, Xpos, Ypos))
Hsav = H0
Print H0

' --ERROR TOLLERANCE
Print "> Error Tolerance on Y = ";
RelErr = Val(InputBox("> RELATIVE ERROR TOLERANCE ON Y = ", "Error Tolerance", 0.0001,
Xpos, Ypos))
Print RelErr
'
' -- Print Details
```

```
Print
PrtAll$ = InputBox("> Print Detail Information <Y/N> ", "Print All ?", "N", Xpos, Ypos)
Print
'
' --PRINTING CONTROL
Print
NP = Val(InputBox("> PRINT INTERVAL = (<10 best) ", "NP", 5, Xpos, Ypos))
Print
MsgBox ("Ready To Proceed")

Cls
' --INITIAL CONDITIONS
Print "> Enter Initial Conditions:"
Print
For I = 1 To Neqn
    Print "    Y0("; I; ") = ";
    Y0(I) = Val(InputBox(" Initial Condition = ", "DATA ENTRY", , Xpos, Ypos))
    Print Y0(I)
Next I
Print

' --Vector of Error Multiplier for Individual Components
'
Print "> Enter Error Multiplier for Individual Components:"
Print
For I = 1 To Neqn
    Print "    Weight("; I; ") = ";
    Weight(I) = Val(InputBox(" Error Multiplier = ", "Error Multiplier", 1, Xpos, Ypos))
    Print Weight(I)
Next I
MsgBox ("Ready To Proceed")
'
'    Call Stiff Equation Solver - Stiff2000
'
Call Stiff2000(Neqn, Lower, Upper, H0, Y0, Y, DF, Weight, RelErr, NP, PrtAll$)
End Sub


Private Sub FunSys(X, Y, F)
'
'    User Defined Differential Equations
'    in the form of
'
'    F(I) = F(X, Y(J))
'
F(1) = -0.4 * Y(1) * Y(4)
F(2) = -F(1) - 0.2 * Y(2) * Y(4)
F(3) = 0.2 * Y(2) * Y(4)
```

```
F(4) = F(1) - F(3) - 0.05 * Y(4) * Y(4)
End Sub


Private Sub DFunSys(X, Y, DF)
'
'   User Defined Jacobian Derivatives Matrix
'   in the form of
'
'   DF(I,J) = dF(X, Y)/dY(J)
'
DF(1, 1) = -0.4 * Y(4)
DF(1, 2) = 0
DF(1, 3) = 0
DF(1, 4) = -0.4 * Y(1)

DF(2, 1) = -DF(1, 1)
DF(2, 2) = -DF(1, 2) - 0.2 * Y(4)
DF(2, 3) = -DF(1, 3)
DF(2, 4) = -DF(1, 4) - 0.2 * Y(2)

DF(3, 1) = 0
DF(3, 2) = 0.2 * Y(4)
DF(3, 3) = 0
DF(3, 4) = 0.2 * Y(2)

DF(4, 1) = DF(1, 1) - DF(3, 1)
DF(4, 2) = DF(1, 2) - DF(3, 2)
DF(4, 3) = DF(1, 3) - DF(3, 3)
DF(4, 4) = DF(1, 4) - DF(3, 4) - 0.1 * Y(4)
End Sub


Public Sub Stiff2000(Neqn, Lower, Upper, H0, Y0, Y, DF, Weight, RelErr, NP, PrtAll$)
'
'   ********************************************************
'
'   Implicit Runge Kutta Differential Equation Solver
'   Program Developed by Dr. Ron Hsin Chang @ Year 2000
'
'   ********************************************************
'
'   Neqn         Number of Equations
'   Lower        Lower Bound of Independent Variable
'   Upper        Upper Bound of Independent Variable
'   H0            Initial Step Size
'   Y0(I)        Initial Conditions
'   Y(I)         Function Value after Integration
```

```
'    DF(I,J)    Derivative Matrix after Integration
'    Weight      Vector of Error Multipliers for Y(I)
'    RelErr      Relative Error Tolerance
'    NP             Number of Printing Intervals
'    PrtAll$    Print Detail Control
'
Dim IP(20), Yold(20), Yold1(20), YA(20) As Double
Dim YK1(20), YK2(20), YK3(20), X As Double
Dim W(20), F(20), Fold(20), DFold(20, 20) As Double
'
'     Initialize
'
       Call MachineErrorCheck(U20, RelErr)
       Iflag = 1
       X = Lower
'
'     Put Y0 to Y for Initiation
'
       For I = 1 To Neqn
             Y(I) = Y0(I)
       Next I
'
'     Print Report Tittle & Initial Conditions
'
       PrtEnt = 1
             Call ReportStiff(PrtEnt, Neqn, X, Y, RelErr, AbsErr, HalfSizeCounter, AdjustRatio)
       PrtEnt = 2
             Call ReportStiff(PrtEnt, Neqn, X, Y, RelErr, AbsErr, HalfSizeCounter, AdjustRatio)
'
'     CALL Implicit Runge Kutta Solver
'
       Call IntegrationController(Neqn, NP, Lower, Upper, H0, RelErr, Y, Yold, Yold1, IP, YA, YK1,
YK2, YK3, DF, DFold, F, Fold, NPTT, Weight, HalfSizeCounter, AdjustRatio, PrtAll$, U20)
End Sub


Public Sub ReportStiff(PrtEnt, Neqn, X, Y, RelErr, AbsErr, HalfSizeCounter, AdjustRatio)
Select Case PrtEnt
Case 1
       Cls
       Print "SOLUTION            with Abs Error =";
       Print Format(AbsErr, "    0.00E+00");
       Print "    RelErr = ";
       Print Format(RelErr, "    0.00E+00")
       Print "***********************************************"
       Print "         X                                ";
       Print "               Y(i)                        "
Case 2
```

```
        Print Format(X, "      0.000000E+00");
        For IK = 1 To Neqn
            Print Format(Y(IK), "      0.000000E+00");
        Next IK
        Print Format(HalfSizeCounter, "    000");
        Print Format(AdjustRatio, "      0.00E+00")
        'MsgBox "Ready"
Case 3
        Print Format(X, "      0.000000E+00");
        For IK = 1 To Neqn
            Print Format(Y(IK), "      0.000000E+00");
        Next IK
        Print Format(HalfSizeCounter, "    000");
        Print Format(AdjustRatio, "      0.00E+00")
End Select
End Sub


Public Sub IntegrationController(Neqn, NP, X0, Xfinal, H0, RelErr, Y, Yold, Yold1, IP, YA, YK1,
YK2, YK3, DF, DFold, F, Fold, NPTT, Weight, HalfSizeCounter, AdjustRatio, PrtAll$, U20)
Static E, ES, ErrRatio, X As Double
Static IPivot(30), CheckFlag, Icount, NPrtOut As Integer
' Icount    = 0 except for last step which ends exactly at Xfinal
Icount = 0
NPrtOut = 0
X = X0
StepSize = H0
If (X0 + 2 * StepSize < Xfinal) Then
        CheckFlag = 2
Else
        ChexkFlag = 1
'      Step longer than interval
End If
Do
        If CheckFlag = 1 Then
            StepSize = (Xfinal - X) / 2
            Icount = 1
        End If
        Do
            If (Icount = 0 And X + 4 * StepSize > Xfinal) Then StepSize = (Xfinal - X) / 4
'
'           Evaluate Function and Jacobian
'
            HalfSizeCounter = -1
            Call FunSys(X, Y, F)
            Call DFunSys(X, Y, DF)
'
'           Keep Values which are used in Half-step Integration
```

```
'
            For I = 1 To Neqn
                Yold(I) = Y(I)
                Fold(I) = F(I)
                For J = 1 To Neqn
                    DFold(I, J) = DF(I, J)
                Next J
            Next I
'
'           Perform Full-Step Integration
'
            Call SemiImplicitRungeKutta3(Neqn, IP, F, X, Y, YK1, YK2, YK3, DF, 2 * StepSize, U20)
            For I = 1 To Neqn
                YA(I) = Y(I)
                Y(I) = Yold(I)
                F(I) = Fold(I)
                For J = 1 To Neqn
                    DF(I, J) = DFold(I, J)
                Next J
            Next I
'
'           Full-Step Finished
'           Compare Full-Step and Half-Step Results
'
            HalfSizeCounter = 0
            Do
'
'               HalfSizeCounter Counts Number of Steplength Bisections
'
                HalfSizeCounter = HalfSizeCounter + 1
                Call SemiImplicitRungeKutta3(Neqn, IP, F, X, Y, YK1, YK2, YK3, DF, StepSize,
U20)
                Call FunSys(X, Y, F)
                Call DFunSys(X, Y, DF)
                For I = 1 To Neqn
                    Yold1(I) = Y(I)
                Next I
                Call SemiImplicitRungeKutta3(Neqn, IP, F, X, Y, YK1, YK2, YK3, DF, StepSize, U20)
                E = 0
'
'               Half Step Integration Finished
'               Compute Derivatives and Compare with Error Tolerance
'
                For I = 1 To Neqn
                    ES = Weight(I) * Abs(YA(I) - Y(I)) / (1 + Abs(Y(I)))
                    If (ES > E) Then E = ES
                Next I
                ErrRatio = E / RelErr
```

```
                    AdjustRatio = (4 * ErrRatio) ^ 0.25
                    If (ErrRatio <= 1) Then
                        Exit Do
                    Else
'
'                       Deviation Too Large
'                       Return to Half Step with Smaller StepSize
'
                        For I = 1 To Neqn
                            YA(I) = Yold1(I)
                            F(I) = Fold(I)
                            Y(I) = Yold(I)
                            For J = 1 To Neqn
                                DF(I, J) = DFold(I, J)
                            Next J
                        Next I
                        StepSize = StepSize / 2
                        Icount = 0
                    End If
                Loop
'
'               Adjust Y-Vector
'
                For I = 1 To Neqn
                Y(I) = Y(I) + (Y(I) - YA(I)) / 7
                Next I
                X = X + 2 * StepSize
'
'               Compute Next Stesize and Print Detail
'
                AdjustRatio = 1 / (AdjustRatio + U20)
                If (AdjustRatio > 3) Then AdjustRatio = 3
                StepSize = AdjustRatio * StepSize
                NPrtOut = NPrtOut + 1
                If PrtAll$ = "Y" Then
                    PrtEnt = 3
                    Call ReportStiff(PrtEnt, Neqn, X, Y, RelErr, AbsErr, HalfSizeCounter, AdjustRatio)
                End If
                If (Int(NPrtOut / NP) * NP = NPrtOut Or Icount = 1) Then
                    PrtEnt = 2
                    Call ReportStiff(PrtEnt, Neqn, X, Y, RelErr, AbsErr, HalfSizeCounter, AdjustRatio)
                End If
                H0 = StepSize
                If (Icount = 1) Then Exit Sub
            Loop While (X + 2# * StepSize < Xfinal)
            CheckFlag = 1
        Loop While Icount <> 1
    End Sub
```

```
Public Sub BACK(Neqn, IPivot, A, V)
'
'    Solution of Linear Equation by
'    Back-substitution After Decomposition
'
Nm1 = Neqn - 1
For I = 1 To Nm1
        Ip1 = I + 1
        K = IPivot(I)
        If (K <> I) Then
                Call Swap(V(I), V(K))
        End If
        For J = Ip1 To Neqn
                V(J) = V(J) + A(J, I) * V(I)
        Next J
Next I
V(Neqn) = V(Neqn) / A(Neqn, Neqn)

For I2 = 2 To Neqn
        I = Neqn + 1 - I2
        Ip1 = I + 1
        For J = Ip1 To Neqn
                V(I) = V(I) - A(I, J) * V(J)
        Next J
        V(I) = V(I) / A(I, I)
Next I2
End Sub


Public Sub Swap(A, B)
X = A
A = B
B = X
End Sub


Public Sub LU(Neqn, IPivot, A)
'
' LU Decomposition for Tridiagonal Matrix Equation
'
IPivot(Neqn) = Neqn
Nm1 = Neqn - 1
For I = 1 To Nm1
        X = A(I, I)
        If (X < 0) Then X = -X
        IPivot(I) = I
```

```
        Ip1 = I + 1
        For J = Ip1 To Neqn
            Y = A(J, I)
            If (Y < 0) Then Y = -Y
            If (Y > X) Then
                X = Y
                IPivot(I) = J
            End If
        Next J
        If (IPivot(I) <> I) Then
            K = IPivot(I)
            For J = 1 To Neqn
                Call Swap(A(I, J), A(K, J))
            Next J
        End If
        For J = Ip1 To Neqn
            X = -A(J, I) / A(I, I)
            A(J, I) = X
            For K = Ip1 To Neqn
                A(J, K) = A(J, K) + X * A(I, K)
            Next K
        Next J
    Next I
    End Sub


    Public Sub SemiImplicitRungeKutta3(Neqn, IPivot, F, X, Y, YK1, YK2, YK3, DF, StepSize, B)
    Static Beta(4) As Double
    Static Alpha As Double
    Alpha = 0.435866521508459
    Beta(1) = 1.03760949613186
    Beta(2) = 0.834930483852638
    Beta(3) = -0.630202088724452
    Beta(4) = -0.242337891260045
    For I = 1 To Neqn
        For J = 1 To Neqn
            DF(I, J) = -StepSize * Alpha * DF(I, J)
            If (Abs(DF(I, J)) < B) Then DF(I, J) = 0
        Next J
        DF(I, I) = DF(I, I) + 1
    Next I
    '
    ' Perform Triangular Decomposition and Evaluate K1
    '
    Call LU(Neqn, IPivot, DF)
    Call BACK(Neqn, IPivot, DF, F)
    For I = 1 To Neqn
        YK1(I) = StepSize * F(I)
```

```
        YK2(I) = Y(I) + 3 / 4 * YK1(I)
Next I
Call FunSys(X, YK2, F)
Call BACK(Neqn, IPivot, DF, F)
'
'    Evaluate K2
'
For I = 1 To Neqn
        YK2(I) = StepSize * F(I)
        Y(I) = Y(I) + Beta(1) * YK1(I) + Beta(2) * YK2(I)
        YK2(I) = Beta(3) * YK1(I) + Beta(4) * YK2(I)
Next I
'
'    Evaluate K3
'
Call BACK(Neqn, IPivot, DF, YK2)
For I = 1 To Neqn
        Y(I) = Y(I) + YK2(I)
Next I
End Sub


Public Sub MachineErrorCheck(U20, RelErr)
'
'      Calculate Machine Round Off Error
'
ReMin = 0.0000000001
Eps = 1#
Do
        Eps = Eps / 2
        EpsP1 = Eps + 1
Loop While EpsP1 > 1
U20 = 20 * Eps

Rer = 2 * Eps + ReMin
'
'    Check for proper error tolerance
'
If RelErr < Rer Then
        RelErr = Rer
End If
End Sub
```

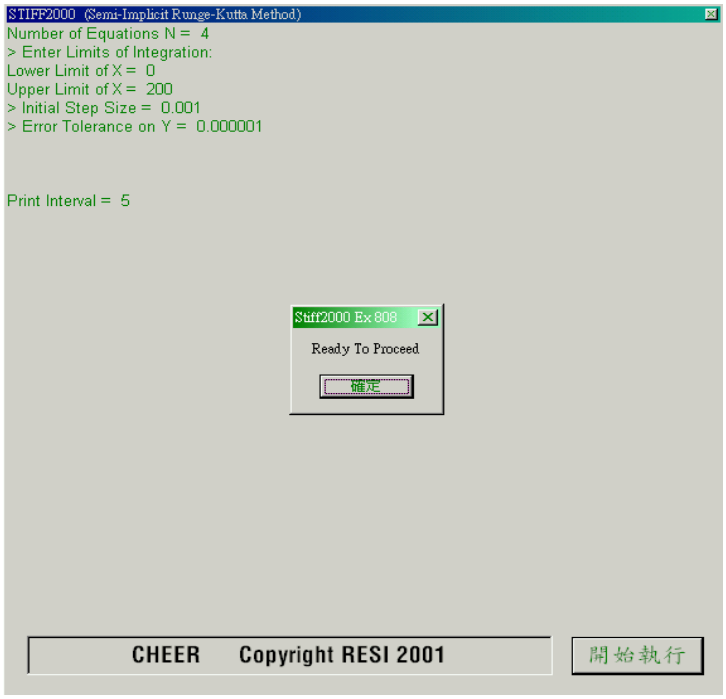
副程式使用說明

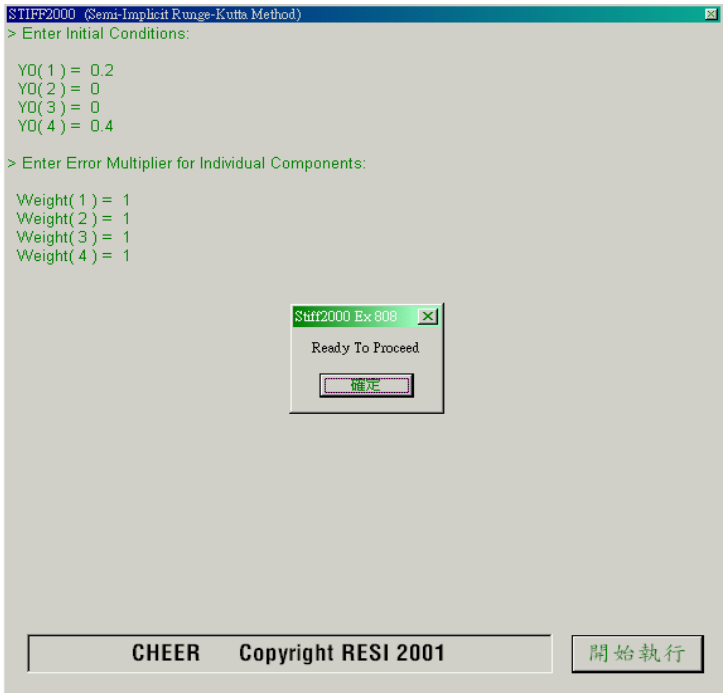
1. 副程式 **Private Sub FunSys(X, Y, F)**
   使用者定義之微分方程組，標準形式如下
   F(I) = F(X, Y(J))

2. 副程式 **Private Sub DFunSys(X, Y, DF)**
   使用者定義之微分方程組的導函數矩陣，標準形式如下
   DF(I,J) = dF(X, Y)/dY(J)

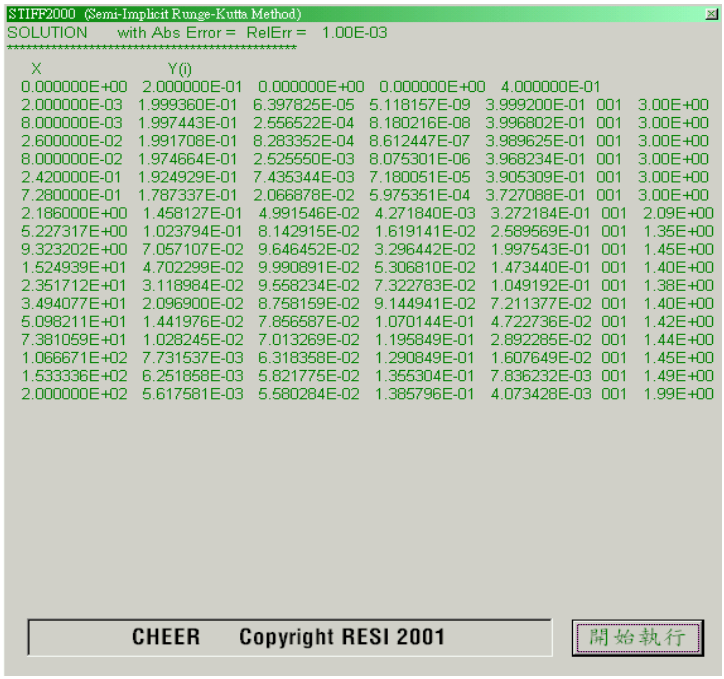
3. 副程式 **Public Sub Stiff2000(Neqn, Lower, Upper, H0, Y0, Weight, RelErr, NP, PrtAll$)**
   使用者在定義微分方程組及其導函數矩陣後，輸入下列參數由副程式 Stiff2000 自動執行積分運算及列印。
   Neqn = 方程式數目
   Lower = 積分下限，即 X0
   Upper = 積分上限，即 Xf
   H0 = 積分間距初設值
   Y0 = 起始條件
   Weight = 誤差計算配重
   RelErr = 相對誤差
   NP = 列印分割區間數
   PrtAll$ = 細節列印控制

測試數據與結果

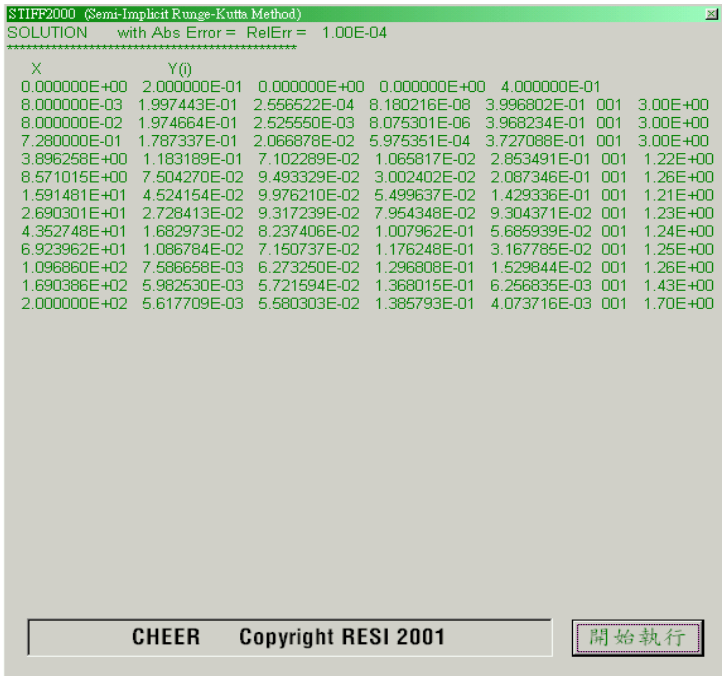

STIFF2000 (Semi-Implicit Runge-Kutta Method)
Number of Equations N = 4
> Enter Limits of Integration:
Lower Limit of X = 0
Upper Limit of X = 200
> Initial Step Size = 0.001
> Error Tolerance on Y = 0.000001
Print Interval = 5
Stiff2000 Ex 808
Ready To Proceed
確定
CHEER Copyright RESI 2001
開始執行


STIFF2000 (Semi-Implicit Runge-Kutta Method)
> Enter Initial Conditions:
Y0(1) = 0.2
Y0(2) = 0
Y0(3) = 0
Y0(4) = 0.4
> Enter Error Multiplier for Individual Components:
Weight(1) = 1
Weight(2) = 1
Weight(3) = 1
Weight(4) = 1
Stiff2000 Ex 808
Ready To Proceed
確定
CHEER Copyright RESI 2001
開始執行

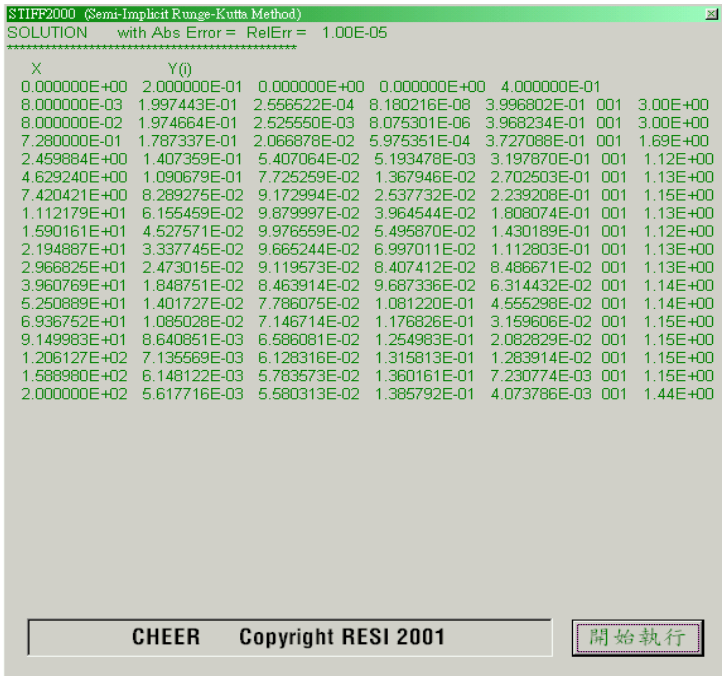
RelErr = 1.E-3　NP=1



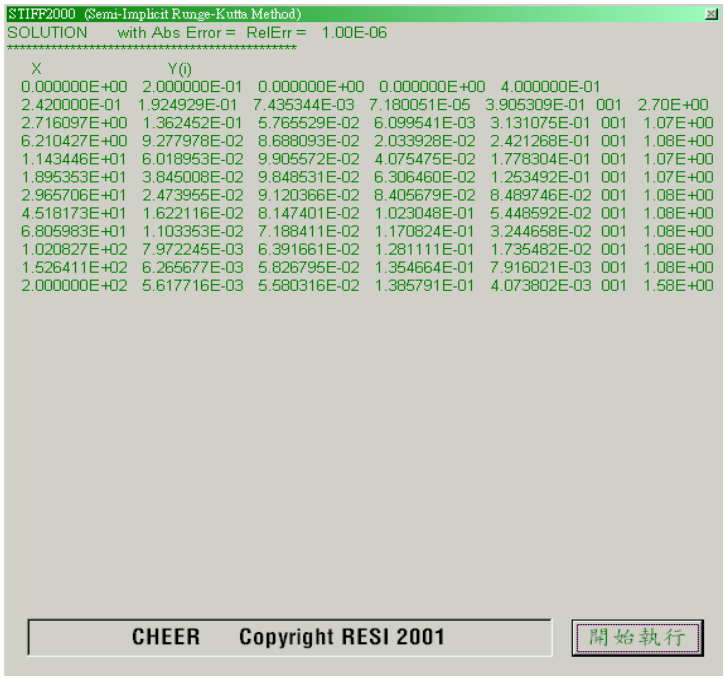
RelErr = 1.E-4　NP=2



RelErr = 1.E-5　NP=2



RelErr = 1.E-6　NP=5



結果討論

1. 由於半隱式阮奇庫塔法具有絕對穩定性，其收斂速度快，計算準確。因此，由以上計算結果可以發現，相對誤差由 1.E-3 改變至 1.E-6，對計算準確度影響並不大。
2. 改變相對誤差要求，對計算步驟之影響如圖 8.5 所示，二者的對數值約呈線性關係。

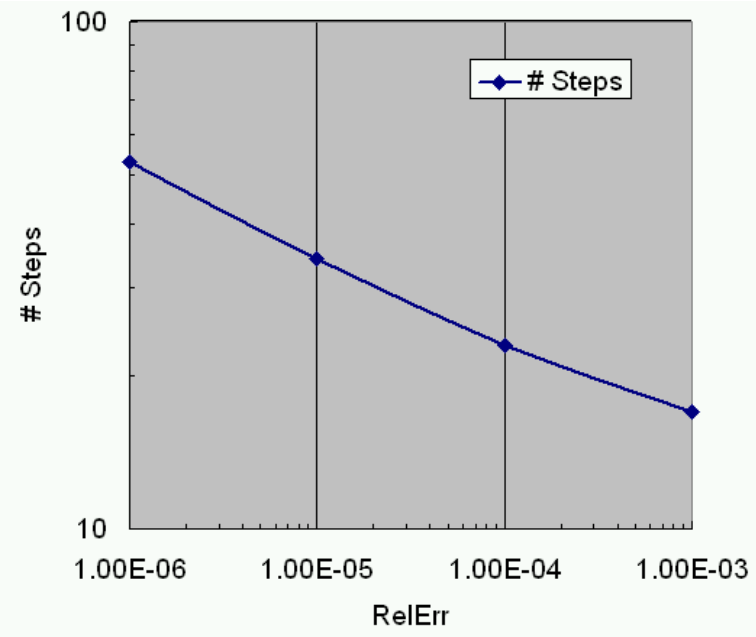


圖 8.5　相對誤差與計算次數

3. 若微分方程式的導函數不易推導出，也可以利用差分法，求得其導函數矩陣；程式碼如下。但應注意這種處理方法會引進誤差，使積分效率變差。

```
Sub DFunSys(X, Y, DF, U20)
'
'   User Defined Jacobian Derivatives Matrix
'   in the form of
'
'   DF(I,J) = dF(X, Y)/dY(J)
'
'   By Central Difference
```

```
'
Static F1(20), F2(20), Y1(20), Y2(20) As Double

For J = 1 To N

    Call YplusEps(Y1(J), 0.000001, U20)
    Call YplusEps(Y2(J), -0.000001, U20)

    For I = 1 To N
        If (I <> J) Then
            Y1(I) = Y(I)
            Y2(I) = Y(I)
        End If
    Next I

    Call FunSys(X, Y1, F1)
    Call FunSys(X, Y2, F2)

    For I = 1 To N
        DF(I, J) = (F1(I) - F2(I)) / (Y1(I) - Y2(I))
    Next I
Next J
End Sub


Sub YplusEps(Y, Eps, U20)

    If (Abs(Y) < U20) Then
        Y = Y + Eps
    Else
        Y = Y * (1 + Eps)
    End If

End Sub
```

例題 8-9 連續反應系統

考慮羅勃森方程式 (Robertson Rate Equation, [11])，並觀察實際計算之問題。

$$\frac{dy_1}{dt} = -0.04y_1 + 10^4 y_2 y_3$$

$$\frac{dy_2}{dt} = 0.04y_1 - 10^4 y_2 y_3 - 3\times 10^7 y_2^2$$

$$\frac{dy_3}{dt} = 3\times 10^7\, y_2^2$$

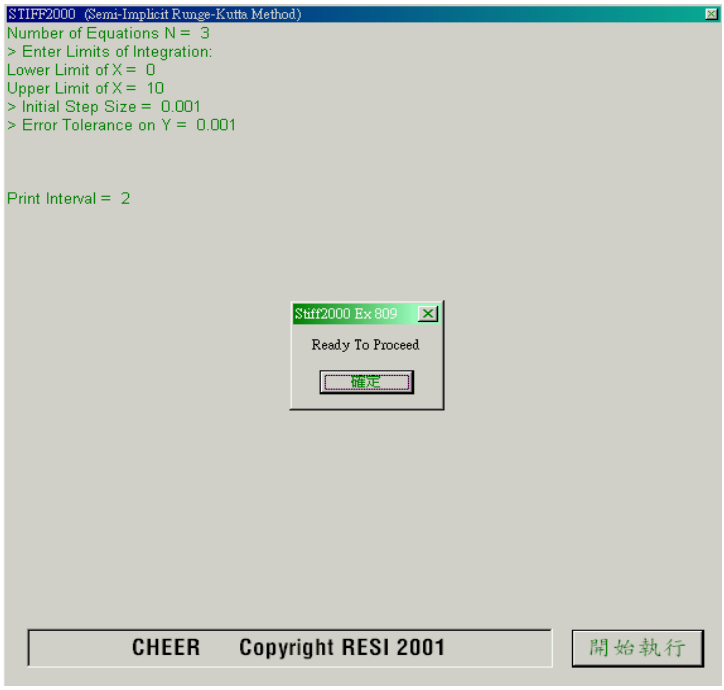
I.C.　$y_1 = 1,\ y_2 = y_3 = 0;\quad t = 0$

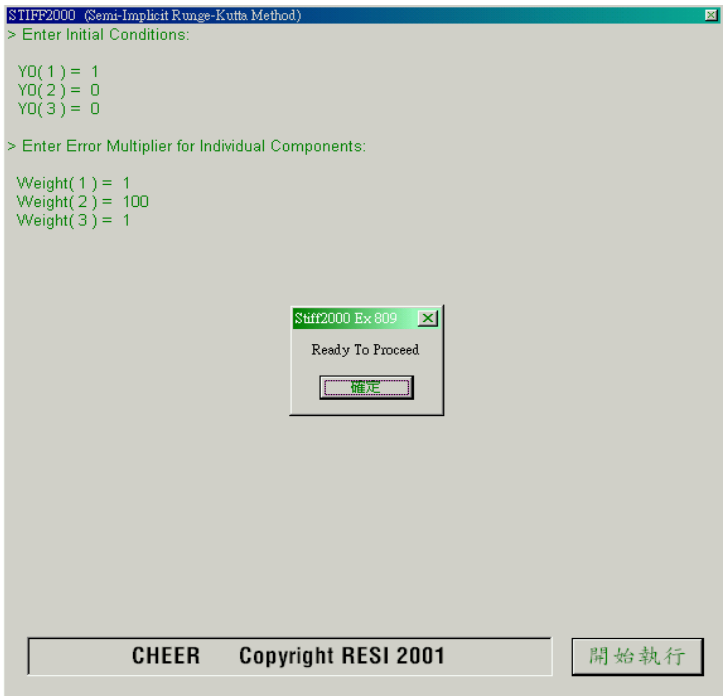
解：

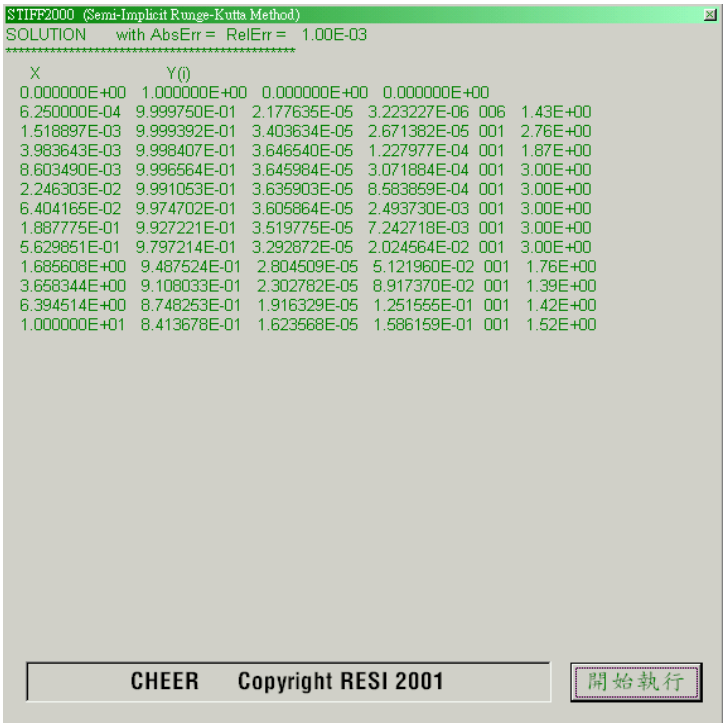
修正例 8-8 程式中的副程式 Sub FunSys(X, Y, F) 及副程式 **Sub DFunSys(X, Y, DF)**，如下圖所示。其他程式碼維持完全相同，即可建立本例題所需之程式。

```
Sub FunSys(X, Y, F)
'
'   User Defined Differential Equations
'   in the form of
'
'   F(I) = F(X, Y(J))
'
F(1) = -0.04 * Y(1) + 10000 * Y(2) * Y(3)
F(2) = -F(1) - 3 * 10 ^ 7 * Y(2) * Y(2)
F(3) = -F(1) - F(2)
End Sub


Sub DFunSys(X, Y, DF)
'
'   User Defined Jacobian Derivatives Matrix
'   in the form of
'
'   DF(I,J) = dF(X, Y)/dY(J)
'
DF(1, 1) = -0.04
DF(1, 2) = 10000 * Y(3)
DF(1, 3) = 10000 * Y(2)

DF(2, 1) = -DF(1, 1)
DF(2, 2) = -DF(1, 2) - 6 * 10 ^ 7 * Y(2)
DF(2, 3) = -DF(1, 3)

DF(3, 1) = -DF(1, 1) - DF(2, 1)
DF(3, 2) = -DF(1, 2) - DF(2, 2)
DF(3, 3) = -DF(1, 3) - DF(2, 3)
End Sub
```

執行結果


STIFF2000 (Semi-Implicit Runge-Kutta Method)
Number of Equations N = 3
> Enter Limits of Integration:
Lower Limit of X = 0
Upper Limit of X = 10
> Initial Step Size = 0.001
> Error Tolerance on Y = 0.001
Print Interval = 2
Stiff2000 Ex 809
Ready To Proceed
確定
CHEER Copyright RESI 2001
開始執行


STIFF2000 (Semi-Implicit Runge-Kutta Method)
> Enter Initial Conditions:
Y0(1) = 1
Y0(2) = 0
Y0(3) = 0
> Enter Error Multiplier for Individual Components:
Weight(1) = 1
Weight(2) = 100
Weight(3) = 1
Stiff2000 Ex 809
Ready To Proceed
確定
CHEER Copyright RESI 2001
開始執行



第九節　聯立微分方程式與特徵值

Visual Basic

聯立一階微分方程式是工程應用上最常使用的數學模式，由於方程式數目相對較多，不易求得理論解，因此，通常需要利用數值解析的方法求解。考慮 N 個係數爲定值的聯立一階微分方程式

$$\frac{d\underline{y}}{dt} = \underline{\underline{A}}\underline{y} + \underline{b} \qquad \underline{y}(t=0) = \underline{y}_0 \tag{8-9.1}$$

其中矩陣 $\underline{\underline{A}}$ 爲 (NXN) 的方陣，其元素 A_{ij} 爲第 i 個方程式中 $y_i(t)$ 項的係數。$\underline{b}$ 爲常數陣列。當時間 t 趨近於無窮大時，假設方程式 (8-9.1) 會得到穩定態的解答 $\underline{y}_\infty$。則考慮當時間 t 趨近於無窮大時，由方程式 (8-9.1) 可以得到

$$\underline{\underline{A}}\underline{y}_\infty = -\underline{b} \tag{8-9.2}$$

或得到穩定態解答爲 $\underline{y}_\infty = -\underline{\underline{A}}^{-1}\underline{b}$。由於聯立微分方程式 (8-9.1) 的解答，可以表示成均勻態方程式的解，再加上特定解。假設方程式的係數矩陣 $\underline{\underline{A}}$ 的所有特徵值都不是

零，則均勻態方程式 $d\underline{y}/dt = \underline{\underline{A}}\underline{y}$ 的解答，即爲 N 個均勻態解乘上待定係數 c_i 的總和。待定係數 c_i 則可以利用微分方程式的非均勻邊界條件來決定。

方程式 (8-9.1) 的均勻微分方程式爲 $d\underline{y}/dt = \underline{\underline{A}}\underline{y}$，此均勻微分方程式的嘗試解爲 $\underline{u}\exp(\lambda t)$，代入均勻微分方程式中，可以得到

$$\underline{u}\,\lambda\exp(\lambda t) = \underline{\underline{A}}\underline{u}\exp(\lambda t) \qquad \textbf{(8-9.3)}$$

或

$$\underline{\underline{A}}\underline{u} = \lambda\underline{u} \qquad \textbf{(8-9.4)}$$

由本書第四章的說明可知，方程式 (8-9.4) 爲矩陣 $\underline{\underline{A}}$ 的代數特徵值問題，其數值解法詳見本書第四章。方程式 (8-9.4) 總共可以得到 N 個不同的特徵值。利用這些特徵值，可以讓均勻微分方程式建立 N 個獨立的解。利用所求得的均勻解及特別解，可以利用加成原理 (Principle of Superposition) 得到微分方程式 (8-9.1) 的完整解答爲

$$\underline{y}(t) = \sum_{i=1}^{N} c_i\underline{u}_i\exp(\lambda_i t) + \underline{y}_\infty \qquad \textbf{(8-9.5)}$$

其中待定常數 c_i 可以由 $t=0$ 時的邊界條件 $\underline{y}(t=0) = \underline{y}_0$ 來決定。因此，由方程式 (8-9.5) 可以得到

$$\underline{y}_0 - \underline{y}_\infty = \sum_{i=1}^{N} c_i\underline{u}_i \equiv \underline{\underline{U}}\,\underline{c} \qquad \textbf{(8-9.6)}$$

其中 $\underline{\underline{U}}$ 爲矩陣 $\underline{\underline{A}}$ 的特徵陣列 $\underline{u}_i$ 所組成的特徵矩陣。由於矩陣 $\underline{\underline{A}}$ 與特徵矩陣的關係爲 $\underline{\underline{A}} = \underline{\underline{U}}\,\underline{\underline{\Lambda}}\,\underline{\underline{U}}^{-1}$，由方程式 (8-9.6) 可以得到待定常數爲

$$\underline{c} = \underline{\underline{U}}^{-1}(\underline{y}_0 - \underline{y}_\infty) \equiv \underline{Y}_0 - \underline{Y}_\infty \qquad \textbf{(8-9.7)}$$

將矩陣 $\underline{\underline{A}}$ 與特徵矩陣的關係式 $\underline{\underline{A}} = \underline{\underline{U}}\,\underline{\underline{\Lambda}}\,\underline{\underline{U}}^{-1}$，代回微分方程式 (8-9.1)，可以得到

$$\begin{aligned}\frac{d\underline{y}}{dt} &= \underline{\underline{U}}\,\underline{\underline{\Lambda}}\,\underline{\underline{U}}^{-1}\underline{y} + \underline{b}\\ \frac{d(\underline{\underline{U}}^{-1}\underline{y})}{dt} &= \underline{\underline{\Lambda}}(\underline{\underline{U}}^{-1}\underline{y}) + \underline{\underline{U}}^{-1}\underline{b}\end{aligned} \qquad \textbf{(8-9.8)}$$

令 $\underline{Y} = \underline{\underline{U}}^{-1}\underline{y}$，則方程式 (8-9.8) 爲陣列 $\underline{Y}$ 的一階常微分方程式，解方程式 (8-9.8)，可以得到

$$\underline{Y} = \underline{\underline{U}}^{-1}\underline{y} = \exp(\underline{\underline{\Lambda}}t)\underline{c} + \underline{Y}_\infty = \exp(\underline{\underline{\Lambda}}t)\underline{Y}_0 + (\underline{\underline{I}} - \exp(\underline{\underline{\Lambda}}t))\underline{Y}_\infty \qquad \textbf{(8-9.9)}$$

或寫成矩陣 $\underline{\underline{A}}$ 及陣列 $\underline{b}$ 的函數，爲

$$\begin{aligned}\underline{y} &= \underline{\underline{U}}\exp(\underline{\underline{\Lambda}}t)\underline{\underline{U}}^{-1}\underline{y}_0 + \underline{\underline{U}}\left[\underline{\underline{I}} - \exp(\underline{\underline{\Lambda}}t)\right]\underline{\underline{U}}^{-1}\underline{y}_\infty \\ &= \underline{\underline{U}}\exp(\underline{\underline{\Lambda}}t)\underline{\underline{U}}^{-1}\left(\underline{y}_0 + \underline{\underline{A}}^{-1}\underline{b}\right) - \underline{\underline{A}}^{-1}\underline{b}\end{aligned} \qquad \textbf{(8-9.10)}$$

計算策略

利用特徵矩陣求解微分方程式的方法，可以整理如下：

1. 先將所要求解的問題，表示成方程式 (8-9.1) 的標準形式。

2. 建立矩陣 $\underline{\underline{A}}$ 及陣列 $\underline{b}$。

3. 求解矩陣 $\underline{\underline{A}}$ 的特徵值，並建立其特徵矩陣 $\underline{\underline{U}}$。

4. 利用方程式 $\underline{y}_\infty = -\underline{\underline{A}}^{-1}\underline{b}$ 及 $\underline{\underline{A}} = \underline{\underline{U}}\,\underline{\underline{\Lambda}}\,\underline{\underline{U}}^{-1}$，計算 $\underline{y}_\infty = -\underline{\underline{U}}\,\underline{\underline{\Lambda}}^{-1}\underline{\underline{U}}^{-1}\underline{b}$。

(1) 首先將 $\underline{\underline{U}}^{-1}$ 與 $\underline{b}$ 乘積放置在陣列 $\underline{w}_1$ 中；

(2) 再將 $\underline{w}_1$ 的第 i 個元素除以特徵值 λ_i，放置在陣列 $\underline{w}_2$ 中；

(3) 然後，計算 $\underline{y}_\infty = -\underline{\underline{U}}\underline{w}_2$。

5. 利用方程式 (8-9.10) 計算 $\underline{y}$。

(1) 首先將 $\underline{\underline{U}}^{-1}$ 與 $\underline{y}_0 - \underline{y}_\infty$ 的乘積放置在陣列 $\underline{w}_3$ 中；

(2) 再將 $\underline{w}_3$ 的第 i 個元素與對角線矩陣 $\exp(\underline{\underline{\Lambda}}t)$ 相乘，得到第 i 個元素爲 $w_{3i}\exp(\lambda_i t)$，放置在陣列 $\underline{w}_4$ 中；

(3) 然後，計算 $\underline{y} = \underline{\underline{U}}\underline{w}_4 + \underline{y}_\infty$，即爲微分方程式的解答。

高階微分方程式的處理策略

係數爲常數的 M 階微分方程式，在引進 1 至 M-1 階的導函數作爲應變數後，可以轉化成 M 個一階聯立微分方程式。同理，N 個二階聯立方程式，也可以轉化成 2N 個一階聯立微分方程式。

例題 8-10　高階聯立微分方程式

試將以下方程式所示之 N 個二階聯立微分方程式，轉化成聯立一階微分方程式。

$$\frac{d^2\underline{\theta}}{dt^2} - \underline{\underline{A}}\frac{d\underline{\theta}}{dt} - \underline{\underline{B}}\underline{\theta} = \underline{0} \qquad \textbf{(8-9.11)}$$

解：

首先定義陣列

$$\underline{\Psi} = \begin{bmatrix} \theta_1 & \theta_2 & \cdots & \theta_N & \varphi_1 & \varphi_2 & \cdots & \varphi_N \end{bmatrix}^T$$

及

$$\underline{\varphi} = d\underline{\theta}/dt$$

則原方程式 (8-9.11) 可以寫成

$$\frac{d\underline{\varphi}}{dt} = \underline{\underline{B}}\underline{\theta} + \underline{\underline{A}}\underline{\varphi} \qquad \textbf{(8-9.12)}$$

$$\frac{d\underline{\theta}}{dt} = \underline{\underline{I}}\,\underline{\varphi} \qquad \textbf{(8-9.13)}$$

或整理成

$$\frac{d\underline{\Psi}}{dt} = \underline{\underline{M}}\,\underline{\Psi} \equiv \begin{bmatrix} \underline{\underline{O}} & \underline{\underline{I}} \\ \underline{\underline{B}} & \underline{\underline{A}} \end{bmatrix} \underline{\Psi} \qquad \textbf{(8-9.14)}$$

其中 M 爲 (2NX2N) 的方矩陣。所得到的聯立微分方程式 (8-9.14) 即可以利用本節所介紹的方法求解。

參考文獻

1. Caillaud, J. B., and L. Padmanabhan, “An Improved Semi-Implicit Runge Kutta Method for Stiff Systems”, Chem. Eng. J., Vol. 2, pp.227 (1971)
2. Cocks, A.T., and K.W. Egger, Int. J. Chem. Kinetics., 4, 169 (1972).
3. Davis, M.E. “Numerical Methods and Modeling for Chemical Engineers”, John Wiley & Sons, New York, (1984).
4. Forsythe, G. E., M. A. Malcom and C. B. Moler, “Computer Methods for Mathematical Computation”; Prentice Hall, (1977)
5. Gear, C.W. “Numerical Initial-Value Problems in Ordinary Differential Equations”, Pretice-Hall, Englewood Cliffs, N.J., (1971).
6. Holland, C.D., and R.G. Anthony, “Fundamentals of Chemical Reaction Engineering” (1987).
7. Johnston, R. L., “Numerical Method – A Software Approach”, Wiley (1982).

8. Krogh, F. T., “Algorithms for Changing Step Size”, SIAM J. Numerical Analysis, Vol. 10, pp. 949 (1973).

9. Michelsen, M. L., “An Efficient General Purpose Method for the Integration of Stiff Ordinary Differential Equations”, AIChE J., Vol. 22, pp.594 (1976).

10. Michelsen, M. L., “Application of the Semi-Implicit Runge-Kutta Methods for Integration of Ordinary and Partial Differential Equations”, Chem. Eng. J., Vol. 14, pp. 107 (1977).

11. Robertson, A. H. “Solution of a Set of Reaction Rate Equations” Numerical Analysis, J. Walsh (ed.), Thomson Brook Co., Washington, (1967).

12. Rosenbrock, H. H. “Some General Implicit Processes for the Numerical Solution of Differential Equations”, Computer J., Vol. 5, pp. 329 (1963).

13. Shampine, L.F., and M.K. Gordon, “Computer Solution of Ordinary Differential Equations: The Initial Value Problem”, Freeman, San Francisco, (1975).

14. Villadsen, J., and M.L. Michelsen, “Solution of Differential Equation Models by Polynomial Approximation”, Prentice-Hall, Englewood, Cliffs, N.J., (1978).

習　題

1. 試利用歐以勒法寫一程式，解例 8-1 的同分異構化反應問題。

$$\frac{dy}{dx} = \begin{bmatrix} -22.365\,y & ; \quad \theta = 200\,\text{sec.} \\ -5.591\,y & ; \quad \theta = 50\,\text{sec.} \end{bmatrix}$$

2. 利用亞當斯貝希佛斯法做爲預測式，並利用亞當斯默頓法做爲修正式，重解例 8-5。

$$y' = x + y\,; \quad \text{I.C.} \quad y(0) = 1$$

3. 試改變例 8-9 中的誤差配重函數值 Weight(I) 及相對誤差 RelErr，觀察計算結果的改變。

4. 試利用 RKF 副程式解以下聯立方程式。

$$\frac{dy_1}{dx} = y_2$$
$$\frac{dy_2}{dx} = 24y_2 - 60y_1$$
$$y_1(0) = 1 \quad ; \quad y_2(0) = 0$$

5. 試分別利用 RKM 副程式解以下聯立方程式。

$$\frac{dy_1}{dx} = 998y_1 + 1998y_2$$
$$\frac{dy_2}{dx} = -999y_1 - 1999y_2$$
$$y_1(0) = 1 \quad ; \quad y_2(0) = 1$$

6. 試分別利用 RKF 副程式解以下聯立方程式。

$$\frac{dy_1}{dx} = 998y_1 + 2998y_2$$
$$\frac{dy_2}{dx} = -999y_1 - 2999y_2$$
$$y_1(0) = 1 \quad ; \quad y_2(0) = 1$$

7. 李表哥戴著獵犬至野外打獵，他發現了一隻野兔，於是放開獵犬追逐野兔。獵犬追逐兔子時，會盯住兔子的方向追趕。若兔子奔跑的速度爲 15 m/sec，且假設只作直線奔跑 (不太實際！)，而獵犬速度爲 20 m/s，請問獵犬需用多少時間才能追到兔子？假設兔子原先與獵犬距離 150 m，且當牠一見獵犬追逐，就馬上朝垂直線方向奔跑。

提示：所得微分方程式爲

$$\begin{cases} \dfrac{dr}{dt} = V_R \cos\phi - V_D \\ \dfrac{d\phi}{dt} = -\dfrac{V_R \sin\phi}{r} \end{cases} \quad \text{I.C.} \quad t = 0 \quad r_0 = L \quad \phi_0 = \frac{\pi}{2}$$

V_R 爲兔子奔跑的速度，V_D 爲獵犬奔跑的速度。

註：理論解爲 17.14286 sec。

8. 一批式反應槽中，進行以下的化學反應

$$A + B \overset{k_1}{\leftrightarrow} C$$
$$C + B \overset{k_2}{\rightarrow} D$$

各成分的莫耳數平衡可得以下的微分方程式

$$\frac{dA}{d\theta} = -k_1 AB + k_1' C$$
$$\frac{dB}{d\theta} = -k_1 AB + k_1' C - k_2 BC$$
$$\frac{dC}{d\theta} = k_1 AB - k_1' C - k_2 BC$$
$$\frac{dD}{d\theta} = k_2 BC$$

利用以下數據求 A, B, C 及 D 的濃度隨時間變化的情況。

$$k_1 = 0.001$$
$$k_1' = 0.015$$
$$k_2 = 0.001$$

當 $\theta = 0$ 時， $A = 10, B = 20, C = D = 0$

9. 重複問題 8，且 $k_1 = 0.4, k_1' = 0.015, k_2 = 0.01$ 。

10. 重複問題 8，但令反應常數爲

$$k_1 = k_2 = 3 \times 10^{18} \exp(-\frac{15000}{T})$$
$$k_1' = 50 \times 10^{18} \exp(-\frac{15000}{T})$$

且溫度 T 利用熱量平衡得到

$$\frac{dT}{d\theta} = (k_1 AB - k_1' C + k_2 BC) * 10 + (300 - T)$$

11. 某一野生動物園的實驗區只飼養花鹿及花豹兩種動物。花鹿有豐富的天然食物可取用，但花豹則需完全仰賴獵食花鹿維生。假設在某一時刻 t，花鹿 A 及花豹 B 的數目分別爲 N_A 及 N_B 。

試證明利用簡單的假設，可縯導出描述兩種動物的滋生及死亡的情況的微分方程式爲

$$\frac{dN_A}{dt} = \alpha N_A - \beta N_A N_B$$
$$\frac{dN_B}{dt} = -\gamma N_B + \delta N_A N_B$$

其中 α, β, γ 及 δ 均爲正參數值。α 爲花鹿之自然生長率，β 爲花豹與花鹿共生情況下，花鹿之被毀滅率，γ 爲花豹在缺乏獵物情況下的之自然死亡率，δ 爲花豹與花鹿共生情況下，花豹之增長率。

(a) 試寫一程式解以上問題，並描述花豹與花鹿之消長情況。

(b) 試改變 N_A 及 N_B 起始值，並調整 α, β, γ 及 δ 值，觀察花豹與花鹿之消長結果變化情況，並以所給定參數值之特性加以說明。

12. U 型壓差計 (manometer) 常被用於測量氣體管道的壓力變化，如圖 8-6 所示。假設最初管道內壓力與大氣壓相同，且壓差計中的液體亦靜止保持在平衡位置。當時間大於 0 時，管道內壓力產生擾動變化

$$P = P_0 \quad ; \quad t = 0$$
$$P = A + B\sin(\omega t) \quad ; \quad t > 0$$

(a) 試證明壓差計中液位高度 h 與時間關係可用以下的常微分方程式表示。

$$\frac{d^2h}{dt^2} + \frac{8\mu}{\rho_\ell R^2}\frac{dh}{dt} + \frac{2g}{L}h = \frac{g_c}{\rho_\ell}\left[P(t) - P_0\right]$$

(b) 試利用阮奇庫塔摩森法解此問題。

(c) 試作圖說明差壓計液位之變化情況。

(d) 若差壓計流體之黏度增爲 0.32 g/cm-sec，所得結果有何變化。

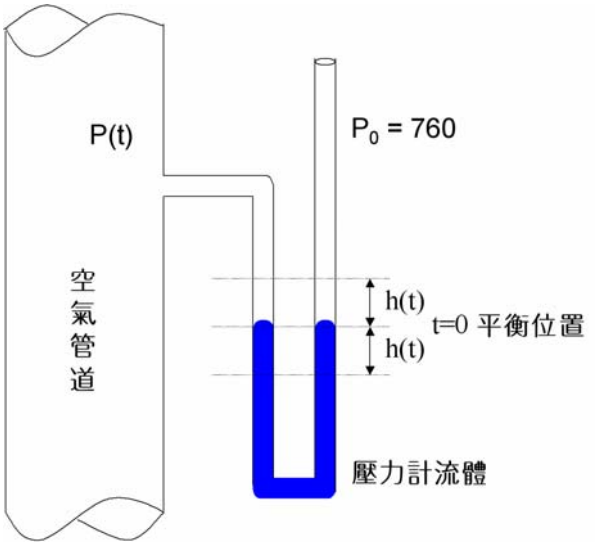


圖 8.6　U 型壓差計

計算參數：

ρ_ℓ = 壓差計液體密度 (13.6 g / cm^3)

μ = 壓差計液體黏度 (0.016 g/cm-sec)

R = 壓差計玻璃管內徑 (0.10 cm)

g = 重力加速度 (980 cm / sec^2)

L = 平衡時液柱總長 (200 cm)

A = 1000

B = 100

ω = 0.1　　1　　2

13. 考慮可逆化學反應

$$A_1 \underset{K_{21}}{\overset{K_{12}}{\Longleftrightarrow}} A_2 \underset{K_{32}}{\overset{K_{23}}{\Longleftrightarrow}} A_3$$

若在時間 $t=0$ 時， $A_1=1.0$, $A_2=A_3=0$ ，且已知反應參數

$$K_{12}=0.1, K_{21}=0.02, K_{23}=0.05, K_{32}=0.01$$

試寫出描述 A_1, A_2, A_3 變化之微分方程式，並求 A_1, A_2, A_3 隨時間之變化。

14. 以下方程式是輻射熱傳遞問題所獲得的常微分方程式

$$\frac{dx_1}{dt} = -2\,x_1 + x_2 + 2$$
$$\frac{dx_2}{dt} = \frac{1}{2}x_1 - x_2 + \frac{1}{2} + \frac{16 - x_2^4}{10}$$

I.C. $\quad x_1(0) = 1,\ x_2(0) = 1$

試求 x_1 及 x_2 隨時間之變化。

15. 仿本章所附程式，寫一阮奇庫塔吉爾法 (8-3.2) 副程式，並重複解設計問題 D-VIII。

16. 試利用亞當斯默頓法撰寫程式，解例 8-7 並與理論解作比較。

17. 毛細作用所形成的曲線，可以利用以下微分方程式描述

$$\frac{d^2y}{dx^2} = \frac{4y}{c^2}\left[1 + \left(\frac{dy}{dx}\right)^2\right]^{3/2}$$

IC1 $\quad x = 0 \quad , \quad y = \ell_0$

IC2 $\quad x = 0 \quad , \quad \frac{dy}{dx} = -\cot\alpha$

其中

$$c^2 = 4\sigma / \rho g$$

σ 爲表面張力

ρ 爲液體密度

g 爲重力加速度

α 爲液固接觸角

試解此問題，並探討 c^2 及 α 對曲線形狀的影響。

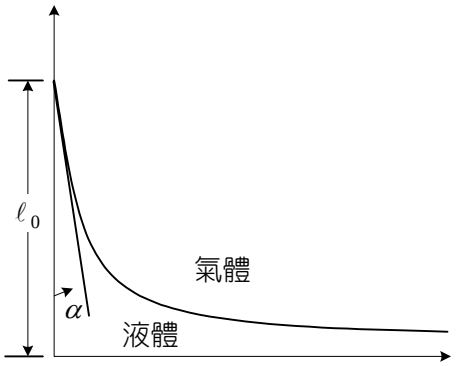


圖 8.7　毛細作用曲線

18. 試利用 RKF 副程式及 STIFF2000 副程式解例 8-9 所示之羅勃森方程式 (Robertson Rate Equation)，並觀察實際計算之問題。

$$\frac{dy_1}{dt} = -0.04y_1 + 10^4 y_2 y_3$$
$$\frac{dy_2}{dt} = 0.04y_1 - 10^4 y_2 y_3 - 3\times 10^7 y_2^2$$
$$\frac{dy_3}{dt} = 3\times 10^7 y_2^2$$
$$\text{I.C.} \qquad y_1 = 1,\ y_2 = y_3 = 0;\quad t = 0$$

(註：當 $t = 10$ 時， $y_1 = 0.84137,\ y_2 = 0.16234,\ y_3 = 0.15862$)

(a) 求出羅勃森方程式的雅可必矩陣 (Jacobian) 之特徵值，並判斷其棘手度。

(b) 分別利用 RKF 副程式及 STIFF2000 副程式求解羅勃森方程式。並比較二者差異。

(c) 畫出 y_1, y_2, y_3 與時間 $t = 0$ 到 10 之關係圖。

(d) 分析羅勃森方程式，解釋爲何在時間 t 較小時， y_2 會有快速的變化。

(e) 由本問題所得結果，說明利用 RKF 副程式解羅勃森方程式所遭遇的困難。

19. 一物體由靜止態落入一黏性流體中，若其阻力與速度平方成正比，則下降位移可用以下微分方程式描述之；

$$1 + (\frac{dy}{dx})^2 = k^2(a-x)^2(\frac{d^2y}{dx^2})^2$$
$$\text{IC1} \qquad x = 0, \qquad y = 0$$
$$\text{IC2} \qquad x = 0, \qquad \frac{dy}{dx} = 0$$

在下列條件下，試求 y 與 x 之關係。

	I	II	III	IV	V
k	0.1	1	10	10	10
a	5	5	0	5	10

20. 試利用 STIFF2000 副程式解以下的 Van der Pol 方程式；

$$\frac{dy_1}{dt} = y_2$$
$$\frac{dy_2}{dt} = K(1-y_1)^2 y_2 - y_1$$
I.C. $y_1(0) = y_2(0) = 1$

(a) 若 $K = 1000$，試改變 RelErr 值，觀察計算結果。是否發現 y 呈震盪情況？

(b) 試描繪 y_1 vs. t 圖形，並找出 y_1(min)、 y_1(max)及其震盪週期。 ($T = 1615.5$)

(c) 試改變 K 值由 0.1 至 100,000，重作以上計算及觀察。

(d) 觀察當 K 值趨近於無窮大， $\max|y_2| \approx \frac{4}{3}K$ ，且當 $y_1 = 2$ 或 -2 時， y_2 會趨近於 $y_2 \approx \frac{2}{3}K^{-1}$ 。

21. 一流體化床反應器的動態模式可以利用以下的聯立微分方程式表示；

$$\frac{d\underline{y}}{dt} = \begin{bmatrix} -1.30 & 10400\,k & 1.30 & 0 \\ 0 & -1880\,(1+k) & 0 & 1880 \\ 266.7 & 0 & -269.3 & 0 \\ 0 & 320 & 0 & -321 \end{bmatrix} \underline{y} + \begin{bmatrix} 0 \\ 0 \\ 1752 \\ 0.1 \end{bmatrix}$$
$$\underline{y}(t=0) = \begin{bmatrix} 759.167 & 0 & 600 & 0.1 \end{bmatrix}$$
$$k = 0.0006 \exp(20.7 - \frac{15000}{y_1})$$

其中 y_1 = 觸媒粒子的溫度

y_2 = 觸媒粒子內的反應物分壓

y_3 = 流體溫度

y_4 = 在流體內的反應物分壓

(a) 試利用 STIFF2000 計算此數學模式。時間 t 的範圍爲 0 至 4000。

(b) 試改變起始步驟的大小及相對誤差，觀察計算結果之差異。

22. 試求以下聯立微分方程式的一般解

$$\begin{bmatrix} \frac{dy_1}{dx} \\ \frac{dy_2}{dx} \\ \frac{dy_3}{dx} \end{bmatrix} = \begin{bmatrix} \alpha & -\beta & \beta \\ -\beta & \alpha & -\beta \\ \beta & -\beta & \alpha \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

23. 史特姆 - 陸亦威爾 (Sturm-Liouville) 方程式是實際物理問題中常見的數學模式

$$\frac{d}{dx}\left[p(x)\frac{dy}{dx}\right] + \left[q(x) + \lambda r(x)\right]y = 0 \qquad a < x < b$$

$$\text{BC1} \qquad y + c_1 \frac{dy}{dx} = 0 \qquad @\, x = a$$

$$\text{BC2} \qquad y + c_2 \frac{dy}{dx} = 0 \qquad @\, x = b$$

(a) 試將史特姆 - 陸亦威爾 (Sturm-Liouville) 方程式改寫成聯立微分方程式。

(b) 若假設 $p(x) = 0$ ，且 $c_1 = c_2 = 0$, $a = 0$, $b = 1$ ，試求下列情況之特徵值。並建立其數值解。

	1	2	3	4
$q(x)$	0	x^2	$1/x$	$1/x^2$
$r(x)$	$1/x$	$2/x$	x	$2x$