Visual Basic

CHAPTER 11

偏微分方程式

(Partial Differential Equations)

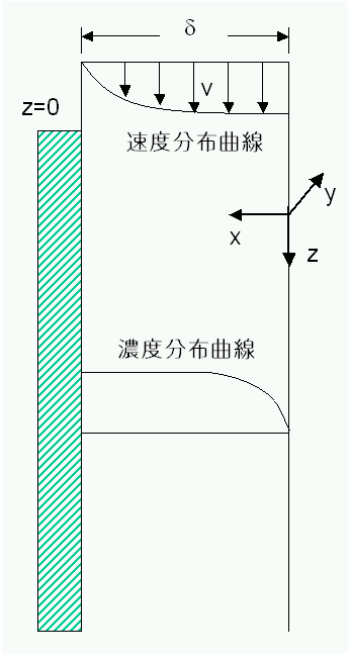
張 榮 興 博士

豐映科技股份有限公司

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

利用簡單的數學模式描述複雜現象是最高明的科學方法

垂直層流液膜的吸附現象



考慮一流體薄膜沿著一垂直平板往下流動，假設其流動場在 $z = 0$ 起，爲一充份展開的拋物線速度分布曲線。假設自由表面位置爲 $x = 0$，且在液體自由表面處，氣體與液體相介面沒有質量傳送阻力。在 x 方向的質量傳遞，主要爲滲透，且在固體平板處 $x = 1$，質傳滲透梯度爲零。博德等人 (Bird, Stewart and Lightfoot, Chap. 17) 建立此問題之數學模式爲

$$(1-x^2)\frac{\partial y}{\partial z}=\frac{\partial^2 y}{\partial x^2}$$

$$y=1 \qquad @\, z=0$$

$$y=0 \qquad @\, x=0 \qquad z>0$$

$$\frac{dy}{dx}=0 \qquad @\, x=1 \qquad z>0$$

在液體薄膜內的平均濃度及質傳薛武德 (Sherwood) 常數，分別可以用以下方程式表示

$$\langle y(z)\rangle=\frac{3}{2}\int_0^1(1-x^2)y(x,z)dx$$

$$Sh=-\frac{2}{3}\frac{(d\langle y\rangle/dz)}{\langle y\rangle}$$

試利用數值解法，求出液體薄膜內的濃度分布曲線，並求出平均濃度及薛武德常數。

工程科學是一門經常需要面對複雜現象，需要大量仰賴經驗及資料的科學，工程師或研究人員時常需面對複雜的自然現象，然後利用簡化的數學模式進行分析探討，以便進行設計或計算。

許多自然界的現象都可以利用偏微分方程式加以描述；典型的二階線性偏微分方程式，可以利用以下的一般式表示之：

$$\sum_{i=1}^{N} A_i \frac{\partial^2 y}{\partial x_i^2} + \sum_{i=1}^{N} B_i \frac{\partial y}{\partial x_i} + cy + D = 0 \qquad \textbf{(11-0.1)}$$

其中 x 爲自變數，y 爲因變數。

第一節　偏微分方程式的分類

Visual Basic

線性二階偏微分方程式依係數A_i及B_i的特性，可分類爲拋物線型偏微分方程式、橢圓曲線型偏微分方程式及雙曲線型偏微分方程式三大類。詳見工程數學相關書籍。

1. 拋物線型偏微分方程式

偏微分方程式 (11-0.1) 中的二次導函數有一個係數A_k爲零；其他A_i都不等於零，且正負符號都相同；且 $B_k \neq 0$。例如；

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = \frac{\partial U}{\partial t} \qquad \textbf{(11-1.1)}$$

2. 橢圓型偏微分方程式

偏微分方程式 (11-0.1) 中所有的二次導函數係數A_i都不等於零，且正負符號都相同。例如；

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2} = 0 \qquad \textbf{(11-1.2)}$$

3. 雙曲線型偏微分方程式

偏微分方程式 (11-0.1) 中所有的二次導函數係數A_i都不等於零，且其中一A_i正負符號與其他A_j不相同。例如；

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = \frac{\partial^2 U}{\partial t^2} \qquad \textbf{(11-1.3)}$$

第二節　有限差分近似法

Visual Basic

本書第八章已經針對如何利用有限差分法處理常微分方程式的導函數作了詳細說明，各種導函數的差分近似方程式，在第八章中也作過詳細討論。對於偏微分方程式中的導函數之有限差分，可以利用完全相同的方式處理之。

假設偏微分方程式中的應變數 u 爲自變數 x 及 y 的函數，或以函數關係表示爲 $u = u(x, y)$，則利用泰勒級數將 $u(x, y)$ 展開，可以得到

$$u(x+h, y+k) = u(x, y) + (h\frac{\partial}{\partial x} + k\frac{\partial}{\partial y})u(x, y) + \frac{1}{2!}(h\frac{\partial}{\partial x} + k\frac{\partial}{\partial y})^2 u(x, y) + \cdots + \frac{1}{(n-1)!}(h\frac{\partial}{\partial x} + k\frac{\partial}{\partial y})^{n-1} u(x, y) + R_n \qquad \textbf{(11-2.1)}$$

$$R_n = \frac{1}{n!}(h\frac{\partial}{\partial x} + k\frac{\partial}{\partial y})^n u(x+\xi h, y+\zeta k)$$
$$= O\left[(|h|+|k|)^n\right] \leq M(|h|+|k|)^n \qquad \textbf{(11-2.2)}$$

當自變數x及y的差分增量h及k趨近於零時，n項泰勒級數展開的捨去誤差R_n值也會趨近於零，使方程式 (11-2.1) 的誤差變成小得可以接受。

考慮如圖 11.1 所示之有限差分格子點，其中格子點 $(i\Delta x, j\Delta y)$ 可以簡稱爲 (i,j)，其周圍的格子點標示符號如圖 11.1 所示。

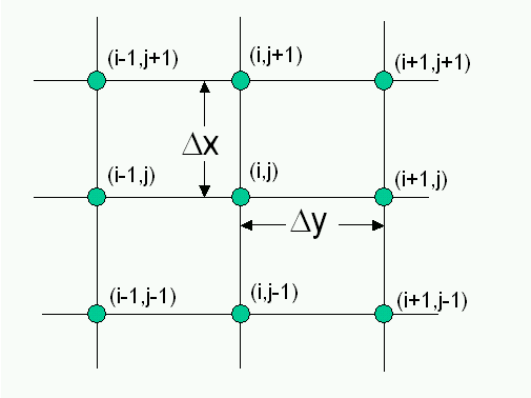


圖 11.1　有限差分隔子點之安排

若針對格子點 (i, j) 鄰近的格子點 $u_{i-1,\,j+1}$, $u_{i,\,j+1}$, $u_{i+1,\,j+1}$, $u_{i-1,\,j}$, $u_{i+1,\,j}$, $u_{i-1,\,j-1}$, $u_{i,\,j-1}$, $u_{i+1,\,j-1}$ 對中間格子點 $u_{i,\,j}$ 作泰勒級數開，則由方程式 (11-2.1) 的一般式，可以分別得到

$$u_{i-1,j} \cong u_{i,j} - \Delta x\, u_x + \frac{(\Delta x)^2}{2!}u_{xx} - \frac{(\Delta x)^3}{3!}u_{xxx} + \frac{(\Delta x)^4}{4!}u_{xxxx} \qquad \textbf{(11-2.3)}$$

$$u_{i+1,j} \cong u_{i,j} + \Delta x\, u_x + \frac{(\Delta x)^2}{2!}u_{xx} + \frac{(\Delta x)^3}{3!}u_{xxx} + \frac{(\Delta x)^4}{4!}u_{xxxx} \qquad \textbf{(11-2.4)}$$

$$u_{i,j-1} \cong u_{i,j} - \Delta y\, u_y + \frac{(\Delta y)^2}{2!}u_{yy} - \frac{(\Delta y)^3}{3!}u_{yyy} + \frac{(\Delta y)^4}{4!}u_{yyyy} \qquad \textbf{(11-2.5)}$$

$$u_{i,j+1} \cong u_{i,j} + \Delta y\, u_y + \frac{(\Delta y)^2}{2!}u_{yy} + \frac{(\Delta y)^3}{3!}u_{yyy} + \frac{(\Delta y)^4}{4!}u_{yyyy} \qquad \textbf{(11-2.6)}$$

$$u_{i+1,j+1} \cong u_{i,j} + \Delta x\, u_x + \Delta y\, u_y + \frac{(\Delta x)^2}{2!}u_{xx} + \frac{(\Delta y)^2}{2!}u_{yy}$$
$$+\Delta x\Delta y u_{xy} + \frac{1}{3!}(\Delta x\frac{\partial}{\partial x} + \Delta y\frac{\partial}{\partial y})u(x,y) + \mathrm{O}\left[(\Delta x + \Delta y)^4\right] \qquad \textbf{(11-2.7)}$$

$$u_{i-1,j-1} \cong u_{i,j} - \Delta x\, u_x - \Delta y\, u_y + \frac{(\Delta x)^2}{2!}u_{xx} + \frac{(\Delta y)^2}{2!}u_{yy}$$
$$+\Delta x\Delta y u_{xy} - \frac{1}{3!}(\Delta x\frac{\partial}{\partial x} + \Delta y\frac{\partial}{\partial y})u(x,y) + \mathrm{O}\left[(\Delta x + \Delta y)^4\right] \qquad \textbf{(11-2.8)}$$

$$u_{i+1,j-1} \cong u_{i,j} + \Delta x\, u_x - \Delta y\, u_y + \frac{(\Delta x)^2}{2!}u_{xx} + \frac{(\Delta y)^2}{2!}u_{yy}$$
$$-\Delta x\Delta y u_{xy} + \frac{1}{3!}(\Delta x\frac{\partial}{\partial x} - \Delta y\frac{\partial}{\partial y})u(x,y) + \mathrm{O}\left[(\Delta x + \Delta y)^4\right] \qquad \textbf{(11-2.9)}$$

$$u_{i-1,j+1} \cong u_{i,j} - \Delta x\, u_x + \Delta y\, u_y + \frac{(\Delta x)^2}{2!}u_{xx} + \frac{(\Delta y)^2}{2!}u_{yy}$$
$$-\Delta x\Delta y u_{xy} + \frac{1}{3!}(-\Delta x\frac{\partial}{\partial x} + \Delta y\frac{\partial}{\partial y})^3 u(x,y) + \mathrm{O}\left[(\Delta x + \Delta y)^4\right] \qquad \textbf{(11-2.10)}$$

在以上的方程式中，$u_x \equiv \partial u/\partial x$ ，$u_{xx} \equiv \partial^2 u/\partial x^2$ ，餘此類推。且所有的導函數值需在格子點 (i, j) 位置計算。

1. 向後差分

由方程式 (11-2.3) 及方程式 (11-2.5)，可以得到一次導函數的向後差分 (Backward Difference) 表示式爲

$$\frac{\partial u}{\partial x}=\frac{u_{i,j}-u_{i-1,j}}{\Delta x}+\mathrm{O}(\Delta x) \qquad \textbf{(11-2.11)}$$

$$\frac{\partial u}{\partial y}=\frac{u_{i,j}-u_{i,j-1}}{\Delta y}+\mathrm{O}(\Delta y) \qquad \textbf{(11-2.12)}$$

2. 前進差分

由方程式 (11-2.4) 及方程式 (11-2.6)，可以得到一次導函數的前進差分 (Forward Difference) 表示式爲

$$\frac{\partial u}{\partial x}=\frac{u_{i+1,j}-u_{i,j}}{\Delta x}+\mathrm{O}(\Delta x) \qquad \textbf{(11-2.13)}$$

$$\frac{\partial u}{\partial y}=\frac{u_{i,j+1}-u_{i,j}}{\Delta y}+\mathrm{O}(\Delta y) \qquad \textbf{(11-2.14)}$$

3. 中央差分

由方程式 (11-2.3) 及方程式 (11-2.4) 相減，可以得到一次導函數的中央差分 (Central Difference) 表示式爲

$$\frac{\partial u}{\partial x}=\frac{u_{i+1,j}-u_{i-1,j}}{2\Delta x}+\mathrm{O}(\Delta x^2) \qquad \textbf{(11-2.15)}$$

同理由方程式 (11-2.5) 及方程式 (11-2.6) 相減，可以得到

$$\frac{\partial u}{\partial y}=\frac{u_{i,j+1}-u_{i,j-1}}{\Delta y}+\mathrm{O}(\Delta y^2) \qquad \textbf{(11-2.16)}$$

又由方程式 (11-2.3) 及方程式 (11-2.4) 相加，可以得到二次導函數的中央差分 (Central Difference) 表示式爲

$$\frac{\partial^2 u}{\partial x^2}=\frac{u_{i-1,j}-2u_{i,j}+u_{i+1,j}}{2\Delta x}+\mathrm{O}(\Delta x^2) \qquad \textbf{(11-2.17)}$$

$$\frac{\partial^2 u}{\partial y^2}=\frac{u_{i,j-1}-2u_{i,j}+u_{i,j+1}}{2\Delta y}+\mathrm{O}(\Delta y^2) \qquad \textbf{(11-2.18)}$$

將方程式 (11-2.7) 加上方程式 (11-2.8)，然後減去方程式 (11-2.9) 及方程式 (11-2.10)，得到u_{xy}的差分表示式爲

$$u_{xy} = \frac{u_{i+1,j+1} - u_{i+1,j-1} - u_{i-1,j+1} + u_{i-1,j-1}}{4\Delta x \Delta y} + \mathrm{O}\left[(\Delta x + \Delta y)^2\right] \qquad \textbf{(11-2.19)}$$

將方程式 (11-2.7)、(11-2.8)、(11-2.9) 及 (11-2.10) 加以整理，且假設 $\Delta x = \Delta y \equiv \Delta h$，則可以得到

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \frac{u_{i+1,j+1} + u_{i+1,j-1} - 4u_{i,j} + u_{i-1,j+1} + u_{i-1,j-1}}{2(\Delta h)^2} + \mathrm{O}\left[(\Delta h)^2\right]$$

$$= \frac{\begin{bmatrix} +u_{i-1,j+1} & 0 & +u_{i+1,j+1} \\ 0 & -4u_{i,j} & 0 \\ +u_{i-1,j-1} & 0 & +u_{i+1,j-1} \end{bmatrix}}{2(\Delta h)^2} + \mathrm{O}\left[(\Delta h)^2\right] \qquad \textbf{(11-2.20)}$$

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \frac{\begin{bmatrix} +u_{i-1,j+1} & +4u_{i,j+1} & +u_{i+1,j+1} \\ +4u_{i-1,j} & -20u_{i,j} & +4u_{i+1,j} \\ +u_{i-1,j-1} & +4u_{i,j-1} & +u_{i+1,j-1} \end{bmatrix}}{6(\Delta h)^2} + \mathrm{O}\left[(\Delta h)^4\right] \qquad \textbf{(11-2.21)}$$

利用這種方法將原來的偏微分方程式改寫成的差分方程式，由於只用到前一時間所得到的資料，因此，稱爲顯式差分法 (Explicit Form)。顯式差分法在使用上應特別注意穩定性的問題，下一節將以實際範例作說明。

第三節　顯式有限差分近似法之穩定性

Visual Basic

考慮最簡單的拋物線偏微分方程式或暫態熱傳導方程式

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} \qquad 0 \le x \le 1 \qquad 0 \le t < T \qquad \textbf{(11-3.1)}$$

$$IC \qquad u(x,0) = f(x) \qquad 0 \le x \le 1$$

$$BC1 \qquad u(0,t) = g_0(t) \qquad 0 < t < T$$

$$BC2 \qquad u(1,t) = g_1(t) \qquad 0 < t < T$$

首先將所考慮的偏微分方程式控制空間轉換成差分格子的形式，如圖 11.2 所示，其中$u_{i,n}$表示在時間$t = t_n$時在$x = x_i$位置的函數值。

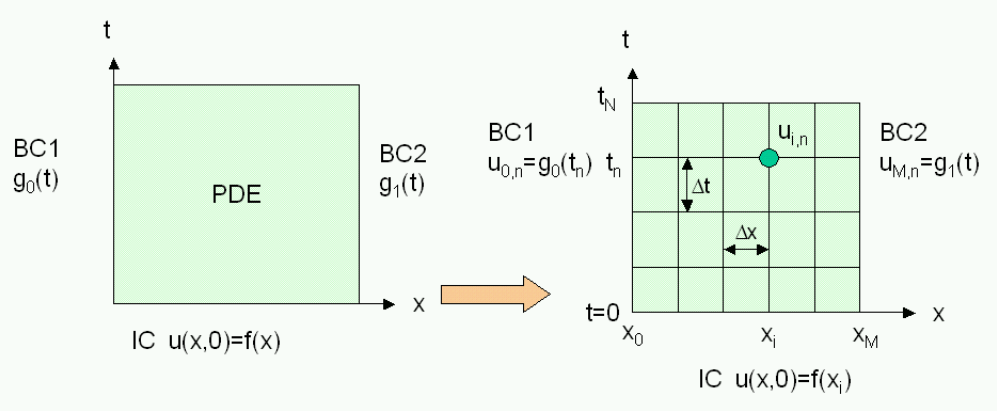


圖 11.2　控制空間轉換為差分格子

其次，將方程式 (11-3.1) 改寫成顯式差分方程式，可以得到

$$\frac{u_{i,n+1}-u_{i,n}}{\Delta t}=\frac{u_{i-1,n}-2u_{i,n}+u_{i+1,n}}{\Delta x^2} \qquad \textbf{(11-3.2)}$$

定義 $\lambda=\Delta t/(\Delta x)^2$，則方程式 (11-3.2) 及對應的邊界條件可以簡化成

$$u_{i,n+1}=\lambda u_{i-1,n}+(1-2\lambda)u_{i,n}+\lambda u_{i+1,n} \qquad \textbf{(11-3.3)}$$

IC.　$u_{i,0}=f(x_i)$

BC1.　$u_{0,n+1}=g_0(t_{n+1})$

BC2.　$u_{M,n+1}=g_1(t_{n+1})$

方程式 (11-3.3) 執行計算時，由前一時刻的三點資料，如圖 11.3 以圓點表示者，即可以求得下一時刻的函數值，如圖中星狀點所示。但利用程式 (11-3.3) 執行計算時，所選定的參數 $\lambda=\Delta t/(\Delta x)^2$ 會影響計算的準確度及穩定性，應特別注意。

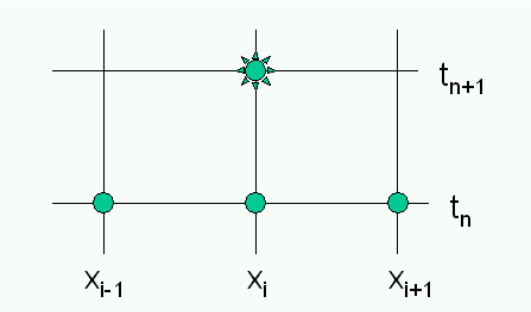


圖 11.3　由前一時刻的資料推算 u_{xx}

令方程式 (11-3.3) 的解可以表示爲 $u_{j,n} = \psi(t)\, e^{i\beta x}$，然後，代回方程式 (11-3.3)，可以得到

$$\frac{\psi(t+\Delta t)e^{i\beta x} - \psi(t)e^{i\beta x}}{\Delta t} = \frac{\psi(t)}{(\Delta x)^2}\left[e^{i\beta(x-\Delta x)} - 2e^{i\beta x} + e^{i\beta(x+\Delta x)}\right]$$

由於 $e^{i\beta x} = \cos(\beta x) + i\sin(\beta x)$，代入上式中，整理後可以得到

$$\psi(t+\Delta t) = \psi(t)\left[1 - 4\lambda \sin^2(\frac{\beta \Delta x}{2})\right] \qquad \textbf{(11-3.4)}$$

又因爲 $\psi(0) = 1$，因此，可以得到 $\psi(t)$ 的解爲

$$\psi(t) = \left[1 - 4\lambda \sin^2(\frac{\beta \Delta x}{2})\right]^{t/\Delta t} \qquad \textbf{(11-3.5)}$$

由於 $\psi(t)$ 在任何 Δx 及 Δt 情況下，都必須是有限值。因此，可以得到 $\psi(t)$ 的解存在且有意義的條件爲

$$\left|1 - 4\lambda \sin^2(\frac{\beta \Delta x}{2})\right| \le 1 \qquad \textbf{(11-3.6)}$$

由於 $\sin^2\theta$ 最大值爲 1，最小值爲 0；因此，可以推論滿足上式的條件爲 $0 \le \lambda \le 1/2$。另外，由泰勒級數展開式，可以得到方程式 (11-3.3) 的差分化誤差爲

$$\begin{aligned} w_{\max}(N) &\le T\left|\frac{\Delta t}{2}u_{tt} - \frac{(\Delta x)^2}{12}u_{xxxx} + \mathrm{O}\left[(\Delta t)^2\right] + \mathrm{O}\left[(\Delta x)^4\right]\right|_{\max} \\ &\equiv T\left|(\Delta x)^2\left[\frac{\lambda}{2} - \frac{1}{12}\right]u_{xxxx} + \mathrm{O}\left[(\Delta t)^2\right] + \mathrm{O}\left[(\Delta x)^4\right]\right|_{\max} \end{aligned}$$

因此，可知當 $0 \le \lambda \le 1/2$ 時，差分化誤差爲 $\mathrm{O}(\Delta x)^2$。但當 $\lambda = 1/6$ 的特定值時，差分化誤差會變成爲 $\mathrm{O}(\Delta x)^4$，積分準確度會更佳。

例題 11-1　暫態熱傳導

考慮一無限平板，在時間爲零時，其溫度分布爲f (x) =0；當時間大於零時，平板兩側溫度分別爲g_0 (t) =100 及g_1 (t) =100。此平板之溫度變化可以用以下的拋物線偏微分方程式描述之。

$$\frac{\partial u}{\partial t}=\frac{\partial^2 u}{\partial x^2} \qquad 0\le x\le 1 \qquad 0\le t<T$$

$$IC \quad u(x,0)=f(x) \qquad 0\le x\le 1$$
$$BC1 \quad u(0,t)=g_0(t) \qquad 0<t<T$$
$$BC2 \quad u(1,t)=g_1(t) \qquad 0<t<T$$

1. 試將此偏微分方程式改寫成差分方程式。

2. 利用 Excel 撰寫程式，描述積分結果。

解：

1. 所得到的差分方程式爲

$$u_{i,n+1}=\lambda u_{i-1,n}+(1-2\lambda)u_{i,n}+\lambda u_{i+1,n}$$

IC. $u_{i,0}=f(x_i)=0$

BC1. $u_{0,n+1}=g_0(t_{n+1})=100$

BC2. $u_{M,n+1}=g_1(t_{n+1})=100$

2. 將差分方程式寫成 Excel 程式，如下圖所示。詳本書所附光碟。

程式編寫方法

1. 先規劃好δx、δt 及 λ 之位置。

2. 於 C2 位置編寫程式碼；C2=I2/F2/F2。

3. 規劃程式所需的表格格式。

4. 於 D7 位置編寫程式碼；D7=$C$2*C6+ (1-2*$C$2) *D6+$C$2*E6。

5. 將 D7 複製到 (D7..L26)。

6. 使用時更改 I2 位置的δt 值，程式即自動計算。

	A	C	D	E	F	G	H	I	J	K	L	M
1												
2	Lamda=	0.500		dx=	0.100		dt=	0.005				
3												
4	Time	格子點編號										
5	t	0	1	2	3	4	5	6	7	8	9	10
6	0.000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7	0.005	100.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0
8	0.010	100.0	50.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	50.0	100.0
9	0.015	100.0	50.0	25.0	0.0	0.0	0.0	0.0	0.0	25.0	50.0	100.0
10	0.020	100.0	62.5	25.0	12.5	0.0	0.0	0.0	12.5	25.0	62.5	100.0
11	0.025	100.0	62.5	37.5	12.5	6.3	0.0	6.3	12.5	37.5	62.5	100.0
12	0.030	100.0	68.8	37.5	21.9	6.3	6.3	6.3	21.9	37.5	68.8	100.0
13	0.035	100.0	68.8	45.3	21.9	14.1	6.3	14.1	21.9	45.3	68.8	100.0
14	0.040	100.0	72.7	45.3	29.7	14.1	14.1	14.1	29.7	45.3	72.7	100.0
15	0.045	100.0	72.7	51.2	29.7	21.9	14.1	21.9	29.7	51.2	72.7	100.0
16	0.050	100.0	75.6	51.2	36.5	21.9	21.9	21.9	36.5	51.2	75.6	100.0
17	0.055	100.0	75.6	56.1	36.5	29.2	21.9	29.2	36.5	56.1	75.6	100.0
18	0.060	100.0	78.0	56.1	42.6	29.2	29.2	29.2	42.6	56.1	78.0	100.0
19	0.065	100.0	78.0	60.3	42.6	35.9	29.2	35.9	42.6	60.3	78.0	100.0
20	0.070	100.0	80.2	60.3	48.1	35.9	35.9	35.9	48.1	60.3	80.2	100.0
21	0.075	100.0	80.2	64.1	48.1	42.0	35.9	42.0	48.1	64.1	80.2	100.0
22	0.080	100.0	82.1	64.1	53.1	42.0	42.0	42.0	53.1	64.1	82.1	100.0
23	0.085	100.0	82.1	67.6	53.1	47.5	42.0	47.5	53.1	67.6	82.1	100.0
24	0.090	100.0	83.8	67.6	57.6	47.5	47.5	47.5	57.6	67.6	83.8	100.0
25	0.095	100.0	83.8	70.7	57.6	52.6	47.5	52.6	57.6	70.7	83.8	100.0
26	0.100	100.0	85.3	70.7	61.6	52.6	52.6	52.6	61.6	70.7	85.3	100.0

使用這個 Excel 程式時，應該要特別注意觀察當 λ 值由小變大，對計算結果的影響。此外，也應該要注意當 $\lambda > 2$ 時，會產生何種計算結果。

	A	C	D	E	F	G	H	I	J	K	L	M
1												
2	Lamda=	0.250		dx=	0.100		dt=	0.003				
3												
4	Time	格子點編號										
5	t	0	1	2	3	4	5	6	7	8	9	10
6	0.000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7	0.003	100.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0
8	0.005	100.0	25.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	25.0	100.0
9	0.008	100.0	37.5	6.3	0.0	0.0	0.0	0.0	0.0	6.3	37.5	100.0
10	0.010	100.0	45.3	12.5	1.6	0.0	0.0	0.0	1.6	12.5	45.3	100.0
11	0.013	100.0	50.8	18.0	3.9	0.4	0.0	0.4	3.9	18.0	50.8	100.0
12	0.015	100.0	54.9	22.7	6.5	1.2	0.2	1.2	6.5	22.7	54.9	100.0
13	0.018	100.0	58.1	26.7	9.2	2.3	0.7	2.3	9.2	26.7	58.1	100.0
14	0.020	100.0	60.7	30.2	11.9	3.6	1.5	3.6	11.9	30.2	60.7	100.0
15	0.023	100.0	62.9	33.2	14.4	5.1	2.5	5.1	14.4	33.2	62.9	100.0
16	0.025	100.0	64.8	35.9	16.8	6.8	3.8	6.8	16.8	35.9	64.8	100.0
17	0.028	100.0	66.4	38.4	19.1	8.6	5.3	8.6	19.1	38.4	66.4	100.0
18	0.030	100.0	67.8	40.5	21.3	10.4	6.9	10.4	21.3	40.5	67.8	100.0
19	0.033	100.0	69.0	42.5	23.4	12.2	8.7	12.2	23.4	42.5	69.0	100.0
20	0.035	100.0	70.1	44.4	25.4	14.1	10.4	14.1	25.4	44.4	70.1	100.0
21	0.038	100.0	71.2	46.1	27.3	16.0	12.3	16.0	27.3	46.1	71.2	100.0
22	0.040	100.0	72.1	47.6	29.2	17.9	14.2	17.9	29.2	47.6	72.1	100.0
23	0.043	100.0	73.0	49.1	31.0	19.8	16.0	19.8	31.0	49.1	73.0	100.0
24	0.045	100.0	73.8	50.6	32.7	21.6	17.9	21.6	32.7	50.6	73.8	100.0
25	0.048	100.0	74.5	51.9	34.4	23.5	19.8	23.5	34.4	51.9	74.5	100.0
26	0.050	100.0	75.2	53.2	36.0	25.3	21.6	25.3	36.0	53.2	75.2	100.0

以上計算顯示當 $0 \le \lambda \le 1/2$ 時，計算進行基本上都是穩定的。取 $\lambda > 1/2$ 時，計算結果將呈現震盪情況，如下圖所示。

	A	C	D	E	F	G	H	I	J	K	L	M
1												
2	Lamda=	1.000		dx=	0.100		dt=	0.010				
3												
4	Time	格子點編號										
5	t	0	1	2	3	4	5	6	7	8	9	10
6	0.000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7	0.010	100.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0
8	0.020	100.0	100.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0	100.0
9	0.030	100.0	0.0	100.0	0.0	0.0	0.0	0.0	0.0	100.0	0.0	100.0
10	0.040	100.0	200.0	(100.0)	100.0	0.0	0.0	0.0	100.0	(100.0)	200.0	100.0
11	0.050	100.0	(200.0)	400.0	(200.0)	100.0	0.0	100.0	(200.0)	400.0	(200.0)	100.0
12	0.060	100.0	700.0	(800.0)	700.0	(300.0)	200.0	(300.0)	700.0	(800.0)	700.0	100.0
13	0.070	100.0	(1400.0)	2200.0	(1800.0)	1200.0	(800.0)	1200.0	(1800.0)	2200.0	(1400.0)	100.0
14	0.080	100.0	3700.0	(5400.0)	5200.0	(3800.0)	3200.0	(3800.0)	5200.0	(5400.0)	3700.0	100.0
15	0.090	100.0	(9000.0)	14300.0	(14400.0)	12200.0	(10800.0)	12200.0	(14400.0)	14300.0	(9000.0)	100.0
16	0.100	100.0	23400.0	(37700.0)	40900.0	(37400.0)	35200.0	(37400.0)	40900.0	(37700.0)	23400.0	100.0

由於平板的溫度一定呈穩定變化，計算產生震盪現象，即證明所選擇參數有誤。當所選擇的積分步伐使得 $\lambda > 1/2$ 時，會產生震盪現象的原因，是由於建立差分方程式 $u_{i,n+1} = \lambda u_{i-1,n} + (1-2\lambda)u_{i,n} + \lambda u_{i+1,n}$ 時，所引進的誤差所致。由於 $\lambda = \Delta t/(\Delta x)^2$，因此，可以推論，在使用差分法求解微分方程式時，並不是一味將時間及空間分割成越多小段越好，而是應該利用 $\lambda = \Delta t/(\Delta x)^2$ 作爲決定積分步伐的判斷依據。

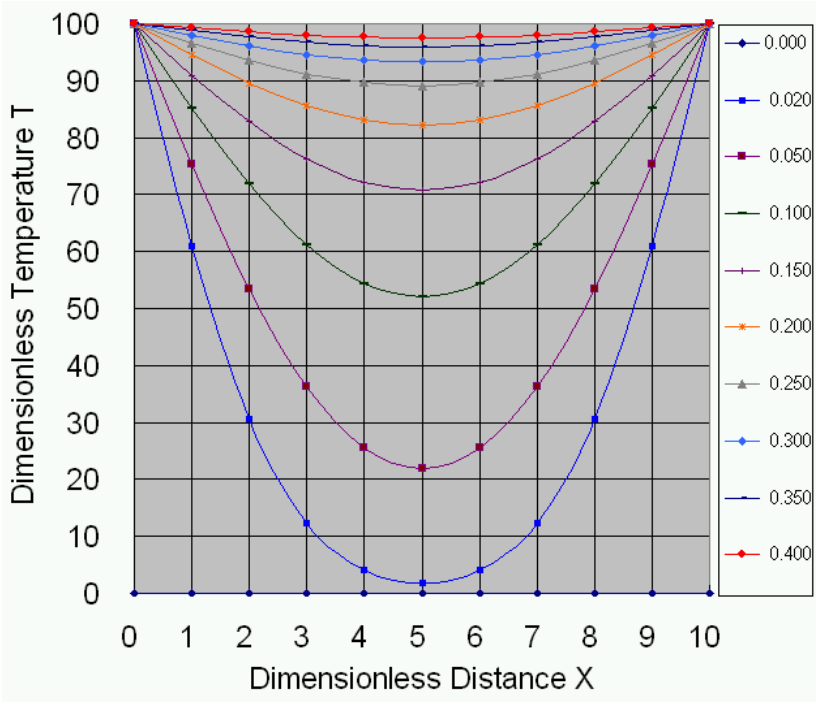


圖 11.4　暫態熱傳導的溫度變化

第四節　隱式有限差分近似法

Visual Basic

上二節中所介紹的顯式有限差分法，由於使用前一時刻的函數值來計算微分值，並作爲此刻的推估值，必然會產生微量誤差，當計算重複執行而未加以修正，則所產生的誤差將會逐漸累增，因此，必須考慮計算的穩定性問題。若將方程式 (11-3.2) 的差分表示式，更改成使用所考慮時刻的資料計算當時的微分值，如圖 11.5 所示；則得到

$$\frac{u_{i,n+1}-u_{i,n}}{\Delta t}=\frac{u_{i-1,n+1}-2u_{i,n+1}+u_{i+1,n+1}}{\Delta x^2} \qquad \textbf{(11-4.1)}$$

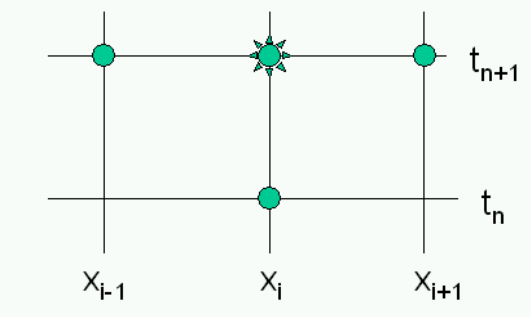


圖 11.5　由此刻的資料推算u_{xx}

定義 $\lambda=\Delta t/(\Delta x)^2$，則上一方程式及對應的邊界條件可以簡化成

$$-\lambda u_{i-1,n+1}+(1+2\lambda)u_{i,n+1}-\lambda u_{i+1,n+1}=u_{i,n} \qquad \textbf{(11-4.2)}$$

IC.　$u_{i,0}=f(x_i)$

BC1.　$u_{0,n+1}=g_0(t_{n+1})$

BC2.　$u_{M,n+1}=g_1(t_{n+1})$

在時間$t = t_n$時，將邊界條件代入，可以將方程式 (11-4.2) 寫成以下的三對角線方程式

$$\begin{bmatrix} 1+2\lambda & -\lambda & 0 & \cdots & 0 & 0 & 0 \\ -\lambda & 1+2\lambda & -\lambda & \cdots & 0 & 0 & 0 \\ 0 & -\lambda & 1+2\lambda & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1+2\lambda & -\lambda & 0 \\ 0 & 0 & 0 & \cdots & -\lambda & 1+2\lambda & -\lambda \\ 0 & 0 & 0 & \cdots & 0 & -\lambda & 1+2\lambda \end{bmatrix} \begin{bmatrix} u_{1,n+1} \\ u_{2,n+1} \\ u_{3,n+1} \\ \vdots \\ \vdots \\ \vdots \\ u_{M-1,n+1} \end{bmatrix} = \begin{bmatrix} u_{1,n} + 2\lambda g_0 \\ u_{2,n} \\ u_{3,n} \\ \vdots \\ \vdots \\ u_{M-2,n} \\ u_{M-1,n} + 2\lambda g_1 \end{bmatrix}$$

(11-4.3)

這種三對角線矩陣方程式，可以利用本書第四章所介紹的上下分解方法求解。這種方法具有絕對穩定性，且其計算過程不會因爲選擇的 λ 值而造成震盪現象。

科瑞克 - 尼可森隱式差分法 (Crank-Nicolson Method)

以上所介紹的方法中，顯式差分法是將u_{xx}利用前一時刻所得到的資料作差分法表示；隱式差分法是將u_{xx}利用同一時刻所得到或所要求解的資料作差分法表示。由於時間方向的差分，基本上應該代表在二時刻間某一時間點的平均微分值，因此，不論顯式積分法或隱式積分法都有過與不及的缺憾。科瑞克-尼可森隱式差分法則是將u_{xx}利用前述二種差分法的某一種平均值來表示。如圖 11.6。

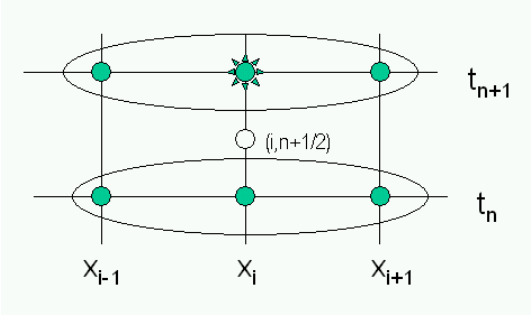


圖 11.6　由前一時刻的差分及此刻的差分，取適當平均值推算u_{xx}

考慮如方程式 (11-3.1) 所表示之拋物線偏微分方程式

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} \qquad 0 \le x \le 1 \qquad 0 \le t < T$$

採用以上所說明的邏輯，並利用以下的差分表示方法；

$$\frac{\partial u}{\partial t} = \frac{u_{i,n+1} - u_{i,n}}{\Delta t} + \mathrm{O}\left[(\Delta t)^2\right]$$

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{@\,t=t_n} = \frac{u_{i-1,n} - 2u_{i,n} + u_{i+1,n}}{\Delta x^2} + \mathrm{O}\left[(\Delta x)^2\right] = \delta_x^2 u_{i,n}$$

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{@\,t=t_{n+1}} = \frac{u_{i-1,n+1} - 2u_{i,n+1} + u_{i+1,n+1}}{\Delta x^2} + \mathrm{O}\left[(\Delta x)^2\right] = \delta_x^2 u_{i,n+1}$$

若將u_{xx}利用前後二時間點的兩種差分法，取某一平均值表示。則可以將u_{xx}寫成

$$\frac{\partial^2 u}{\partial x^2} \cong \vartheta\, \delta_x^2 u_{i,n+1} + (1-\vartheta)\, \delta_x^2 u_{i,n} \qquad \textbf{(11-4.4)}$$

代回微分方程式 (11-3.1)，可以將微分方程式改寫成以下絕對穩定的差分方程式

$$\frac{u_{i,n+1} - u_{i,n}}{\Delta t} = \vartheta\, \delta_x^2 u_{i,n+1} + (1-\vartheta)\, \delta_x^2 u_{i,n} + \mathrm{O}\left[(\Delta t)^2 + (\Delta x)^2\right] \qquad \textbf{(11-4.5)}$$

其中$0 < \vartheta \le 1$。應注意當$\vartheta = 0$時即爲顯式差分法；當$0 < \vartheta \le 1$時稱爲**科瑞克 - 尼可森隱式差分法一般表示式**。

方程式 (11-4.5) 經整理以後，可以得到

$$-\lambda u_{i-1,n+1} + (\frac{1}{\vartheta} + 2\lambda)u_{i,n+1} - \lambda u_{i+1,n+1} = \lambda u_{i-1,n} + (\frac{1}{\vartheta} - 2\lambda)u_{i,n} + \lambda u_{i+1,n} \qquad \textbf{(11-4.6)}$$

方程式 (11-4.6) 中，當$\vartheta = 1/2$時，即稱爲**科瑞克 - 尼可森隱式差分法**。此時，方程式 (11-4.6) 可以進一步簡化及整理成

$$-\lambda u_{i-1,n+1} + 2(1+\lambda)u_{i,n+1} - \lambda u_{i+1,n+1} = \lambda u_{i-1,n} + 2(1-\lambda)u_{i,n} + \lambda u_{i+1,n} \qquad \textbf{(11-4.7)}$$

在時間$t = t_n$時，將邊界條件代入，修正第一個及最後一個方程式，即可以將方程式 (11-4.6) 寫成以下的三對角線方程式

$$\underline{\underline{A}}\,\underline{U}^{(n+1)} = \underline{\underline{B}}\,\underline{U}^{(n)} + \underline{C} \qquad \textbf{(11-4.8)}$$

其中

$$\underline{\underline{A}}=\begin{bmatrix} \frac{1}{\vartheta}+2\lambda & -\lambda & 0 & \cdots & 0 & 0 \\ -\lambda & \frac{1}{\vartheta}+2\lambda & -\lambda & \cdots & 0 & 0 \\ 0 & -\lambda & \frac{1}{\vartheta}+2\lambda & \cdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & -\lambda & 0 \\ 0 & 0 & 0 & \cdots & \frac{1}{\vartheta}+2\lambda & -\lambda \\ 0 & 0 & 0 & \cdots & -\lambda & \frac{1}{\vartheta}+2\lambda \end{bmatrix}$$

$$\underline{\underline{B}}=\begin{bmatrix} \frac{1}{\vartheta}-2\lambda & \lambda & 0 & \cdots & 0 & 0 \\ \lambda & \frac{1}{\vartheta}-2\lambda & \lambda & \cdots & 0 & 0 \\ 0 & \lambda & \frac{1}{\vartheta}-2\lambda & \cdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \lambda & 0 \\ 0 & 0 & 0 & \cdots & \frac{1}{\vartheta}-2\lambda & \lambda \\ 0 & 0 & 0 & \cdots & \lambda & \frac{1}{\vartheta}-2\lambda \end{bmatrix}$$

$$\underline{C}=\begin{bmatrix} \lambda(g_{0,n}+g_{0,n+1}) \\ 0 \\ 0 \\ \vdots \\ 0 \\ \lambda(g_{1,n}+g_{2,n+1}) \end{bmatrix}$$

$$\underline{U}^{(n)}=\begin{bmatrix} u_{1,n} & u_{2,n} & u_{3,n} & \cdots & u_{M-2,n} & u_{M-1,n} \end{bmatrix}^T$$

$$\underline{U}^{(n+1)}=\begin{bmatrix} u_{1,n+1} & u_{2,n+1} & u_{3,n+1} & \cdots & u_{M-2,n+1} & u_{M-1,n+1} \end{bmatrix}^T$$

這種三對角線矩陣方程式，可以利用本書第四章所介紹的上下分解方法求解。這種差分方法因爲具有絕對穩定性，其計算過程不會因爲選擇的 λ 值而造成震盪現象。

佛希施及瓦梭 (Forsythe & Wasow) 證明當 $\vartheta=(6\lambda-1)/12\lambda$ 時，差分化捨去誤差爲 $O\left[(\Delta x)^4\right]$。當 $\vartheta=(6\lambda-1)/12\lambda$ 且 $\lambda=1/\sqrt{20}$ 時，差分化捨去誤差得到最小值，爲 $O\left[(\Delta x)^6\right]$。佛希施及瓦梭所提供的參數值，理論上雖然可以提供較高計算準確度，但其缺點是積分位置並不一定在所想要列表的位置，通常需再作內差處理，以便輸出計算結果。所以一般較少被使用。

例題 11-2　利用科瑞克 - 尼可森隱式差分法解暫態熱傳導問題

考慮如例 9-1 的無限平板，在時間爲零時，其溫度分布爲$f\ (x)\ =0$；當時間大於零時，平板兩側溫度分別爲$g_0\ (t)\ =100\,t^{\alpha}$及$g_1\ (t)\ =\ 100\,t^{\alpha}$。此平板之溫度變化可以用以下的拋物線偏微分方程式描述之。

$$\frac{\partial u}{\partial t}=\frac{\partial^2 u}{\partial x^2} \qquad 0\le x\le 1 \qquad 0\le t<T$$

$$IC \quad u(x,0)=f(x) \qquad 0\le x\le 1$$

$$BC1 \quad u(0,t)=g_0(t) \qquad 0<t<T$$

$$BC2 \quad u(1,t)=g_1(t) \qquad 0<t<T$$

1. 試將此偏微分方程式改寫成差分方程式。

2. 假設$\alpha=1$，試利用視訊培基語言撰寫程式，描述積分結果。

解：所得到的差分方程式爲

$$-\lambda u_{i-1,n+1}+(\frac{1}{\vartheta}+2\lambda)u_{i,n+1}-\lambda u_{i+1,n+1}=\lambda u_{i-1,n}+(\frac{1}{\vartheta}-2\lambda)u_{i,n}+\lambda u_{i+1,n}$$

IC..　$u_{i,0}=f(x_i)=0$

BC1..　$u_{0,n+1}=g_0(t_{n+1})=100t_{n+1}^{\alpha}$

$BC2.$　$u_{M,n+1}=g_1(t_{n+1})=100t_{n+1}^{\alpha}$

有關本問題解法之流程圖，請直接見程式說明及本書第四章三對角線方程式解法說明。

程式列印：

```
'
'****************************************
' SOLUTION OF Parabolic P.D.E.
' BY    Crank-Nicolson Method
'****************************************
'
' Program Developed by Dr. Ron Hsin Chang
' Copyright 1988, 2001
'
' Example 11-2
'
Sub CrankNicolson (Xpos, Ypos)
```

```
Dim U (50) As Double
'
'     Enter Basic Data & DEFINE THE PROBLEM
'
      Cls
      Print
      Print " Solution of Parabolic PDE by Crank-Nicolson Method"
      Print

      ID = 2
      Print "Define Finite Difference Parameters"
      Print "ID = 1    Parameter to be defined by user"
      Print "ID = 2    Crank-Nicolson's Theta = 1/2 will be used."
      Print "ID = 3    The Forsythe & Wasow's optimal parameters will be used."
      ID = Val (InputBox ("ID = ", "Crank-Nicolson Method ID", ID) )
      Print "ID = "; ID

      N = 20
      Print "Number of X-Grid"
      N = Val (InputBox ("N = ", "No. of X-grid", N) )
      Print "N = "; N

      Tmax = 0.25
      Tmax = Val (InputBox ("Maximun Time = ", "Tmax", Tmax) )
      Print "Maximum T =", Tmax
      Print

      For I = 1 To N - 2
            U (I) = 0
      Next I
      BC10 = 0
      BC20 = 0

      Call CrankNicolsonSolver (ID, N, U, BC10, BC20, Tmax)

End Sub


Sub DefineProblem (Entry, Nm2, A, B, C, D1, D2, D3, U, Z, BC1, BC1old, BC2, BC2old, Lamda, Theta)
'     -------------------
'      DEFINE The Problem
'     -------------------
If Entry = 1 Then
'
'     Entry #1
'     Set up Parameters
'
      For I = 1 To Nm2
            C (I) = -Lamda
            A (I) = 1 / Theta + 2 * Lamda
            B (I) = -Lamda
            D3 (I) = Lamda
```

```
            D1 (I) = 1 / Theta - 2 * Lamda
            D2 (I) = Lamda
        Next I
        B (1) = 0
        C (Nm2) = 0
        D2 (1) = 0
        D3 (Nm2) = 0
'
'          ECHO
'
        Call Echo (Nm2, A, B, C, D1, D2, D3)
Else
'
'     Entry #2
'     Get Parameters back
'
        BC1old = BC1
        BC2old = BC2
  '
  '   Calculate Z (I)
  '
        Z (1) = Lamda * (BC1 + BC1old) + D1 (1) * U (1) + D3 (1) * U (2)
        For I = 2 To Nm2 - 1
            Z (I) = D2 (I) * U (I - 1) + D1 (I) * U (I) + D3 (I) * U (I + 1)
        Next I
        Z (Nm2) = D2 (Nm2) * U (Nm2 - 1) + D1 (Nm2) * U (Nm2) + Lamda * (BC2 + BC2old)

End If
End Sub


Sub BoundaryConditions (BC1, BC2, Time)
Alpha = 0
BC1 = 100 * Time ^ Alpha
BC2 = 100 * Time ^ Alpha
End Sub


Sub CrankNicolsonSolver (ID, N, U, BC1old, BC2old, Tmax)
'
'     Finite Difference Parameters
'     ID              = 1    Parameter to be defined by user
'                     = 2    Crank-Nicolson's Theta = 1/2 will be used.
'                     = 3    The Forsythe & Wasow's optimal parameters will be used.
'     N               = NUMBER OF X-GRID
'     U               = INITIAL CONDITION VECTOR
'                       /RETURN : SOLUTION
'     Tmax            = MAXIMUM TIME
'
'     A,B,C           = TRIDIAGONAL MATRIX COEFFICIENT VECTOR
'     D1,D2,D3        = TRIDIAGONAL MATRIX COEFFICIENT VECTOR
'
```

```
Dim A (50) , B (50) , C (50) , D1 (50) , D2 (50) , D3 (50) , Z (50) , V (50) As Double
'
'     Define Crank-Nicolson Parameters
'
      Nm1 = N - 1
      Nm2 = N - 2
      Print "Case ID = "; ID
      Select Case ID
      Case 1, 2
            Dtau = 0.5 / Nm1 / Nm1
            Print "Time Step Size"
            Dtau = Val (InputBox ("dt = ", "Time Step", Dtau) )
            Print "Dtau = "; Dtau

            Lamda = Dtau * Nm1 * Nm1
            Print "Lamda = "; Lamda
            Theta = 0.5
            If ID = 1 Then
                  Theta = Val (InputBox ("Theta = ", "Crand Nicolson Theta", Theta) )
            End If
      Case 3
            Lamda = 1 / Sqr (20)
            Theta = (6 * Lamda - 1) / 12 / Lamda
      End Select

      Print "Lamda = ", Lamda
      Print "Theta = ", Theta
      MsgBox ("Ready to Proceed")

      Cls
      Dtau = Lamda / Nm1 / Nm1
      M = Int (Tmax / Dtau + 0.99)

      Call DefineProblem (1, Nm2, A, B, C, D1, D2, D3, U, Z, BC1, BC1old, BC2, BC2old, Lamda, Theta)
'
'     ------------------------
'       SOLVE & PRINT RESULT
'     ------------------------
'
      Debug.Print "*** SOLUTION:"
      Itime = 0
      Debug.Print Format (Itime * Dtau, "0.00E+00      ") ;
      Debug.Print Format (BC1old, "0.000E+00    ") ;
      For I = 1 To Nm2
            Debug.Print Format (U (I) , "0.000E+00    ") ;
            If (Int (I / 10) = I / 10) Then Debug.Print
      Next I
      Debug.Print Format (BC2old, "0.000E+00    ") ;
      Debug.Print

      For Itime = 1 To M
            Call BoundaryConditions (BC1, BC2, Itime * Dtau)
```

```
            Call DefineProblem (2, Nm2, A, B, C, D1, D2, D3, U, Z, BC1, BC1old, BC2, BC2old,
        Lamda, Theta)
            Call Tridiagonal (1, Nm2, A, B, C, Z, V)
            Debug.Print Format (Itime * Dtau, "0.00E+00      ") ;
            Debug.Print Format (BC1, "0.000E+00    ") ;
            For I = 1 To Nm2
                Debug.Print Format (V (I) , "0.000E+00    ") ;
                If (Int (I / 10) = I / 10) Then Debug.Print
                U (I) = V (I)
            Next I
            Debug.Print Format (BC2, "0.000E+00    ") ;
            Debug.Print
        Next Itime
End Sub


Sub Tridiagonal (Nstart, N, A, B, C, Z, V)
'
'     ---------------------
'        SUBROUTINE JACOBI
'     ---------------------
'
' N               = No. of Equations
' A, B, C      = Tridiagonal Coefficients of Equations
' Alpha         = Intermediate Coefficients
' Gamma         = Intermediate Coefficients
' Z               = Constant Vector
' V               = Solution Vector
'
Dim Alpha (50) , Gamma (50) As Double
Alpha (Nstart) = A (Nstart)
Gamma (Nstart) = Z (Nstart) / Alpha (Nstart)
N1 = Nstart + 1
'
'    -- LU DECOMPOSITION
'
For I = N1 To N
        Alpha (I) = A (I) - B (I) * C (I - 1) / Alpha (I - 1)
        Gamma (I) = (Z (I) - B (I) * Gamma (I - 1) ) / Alpha (I)
Next I
'
'    -- BACK SUBSTITUTION
'
V (N) = Gamma (N)
Last = N - Nstart
For K = 1 To Last
        I = N - K
        V (I) = Gamma (I) - C (I) * V (I + 1) / Alpha (I)
Next K
'
'-- CHEER by Ron Hsin Chang, Copyright 2001
'
End Sub
```

副程式使用說明：

1. 副程式 **Sub CrankNicolsonSolver (ID, N, U, BC1old, BC2old, Tmax)**
   ID 爲決定 Crank-Nicolson 參數策略用之指標。
   N 爲 *x* 座標的格子點數目。
   U 爲起始條件。
   BC1old 及 BC2old 爲時間 *t*=0 時的邊界條件。
   Tmax 爲時間 t 之最大值。
   給定以上參數，呼叫 CrankNicolsonSolver (ID, N, U, BC1old, BC2old, Tmax) 即可進行計算。

2. 副程式 **Sub BoundaryConditions (BC1, BC2, Time)**
   此副程式應由使用者視實際問題更改之；用於計算在時間 *t*=Time 時的邊界條件。

3. 副程式 **Sub DefineProblem (Entry, Nm2, A, B, C, D1, D2, D3, U, Z, Aold, Bold, Cold, BC1, BC1old, BC2, BC2old, Lamda, Theta)**
   此副程式應由使用者視實際問題更改之。
   Entry=1 時，用於計算基本運作陣列 A，B，C，…。
   Entry=2 時，用於計算在時間 *t*=Time 時的陣列 Z。

4. 副程式 **Sub Tridiagonal (N, A, B, C, Z)**
   此副程式爲求解三對角線方程式的副程式。詳本書第四章。
   使用本書第四章所介紹的三對角線方程式的副程式時，要注意經過 **Sub Tridiagonal** (N, A, B, C, Z) 計算，程式中的陣列 A, B, C, Z 會被計算結果替換，因此，若需使用原陣列值，必須重行宣告。
   此處所附之副程式則修正中間計算過程，並引進 Alpha (I) 及 Gamma (I) 作爲中間暫存值，解決前述問題。

數據與計算結果：

N=11，ID=2 所得結果如下；

陣列 C, A, B 所建構的三對角線矩陣

```
3.00E+00    -5.00E-01
-5.00E-01   3.00E+00    -5.00E-01
            -5.00E-01   3.00E+00    -5.00E-01
                        -5.00E-01   3.00E+00    -5.00E-01
                                    -5.00E-01   3.00E+00    -5.00E-0
                                                -5.00E-01   3.00E+00    -5.00E-01
                                                            -5.00E-01   3.00E+00    -5.00E-01
                                                                        -5.00E-01   3.00E+00
```

陣列 D3, D1, D2 所建構的三對角線矩陣

```
1.00E+00    5.00E-01
5.00E-01    1.00E+00    5.00E-01
            5.00E-01    1.00E+00    5.00E-01
                        5.00E-01    1.00E+00    5.00E-01
                                    5.00E-01    1.00E+00    5.00E-01
                                                5.00E-01    1.00E+00    5.00E-01
                                                            5.00E-01    1.00E+00    5.00E-01
                                                                        5.00E-01    1.00E+00
```

*** SOLUTION：

Time	x=0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.00E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
5.00E-03	1.000E+02	3.431E+01	5.888E+00	1.011E+00	1.784E-01	5.947E-02	1.784E-01	1.011E+00	5.888E+00	3.431E+01	1.000E+02
1.00E-02	1.000E+02	4.853E+01	1.665E+01	4.294E+00	1.024E+00	4.205E-01	1.024E+00	4.294E+00	1.665E+01	4.853E+01	1.000E+02
1.50E-02	1.000E+02	5.650E+01	2.528E+01	9.071E+00	2.879E+00	1.441E+00	2.879E+00	9.071E+00	2.528E+01	5.650E+01	1.000E+02
2.00E-02	1.000E+02	6.171E+01	3.197E+01	1.398E+01	5.592E+00	3.304E+00	5.592E+00	1.398E+01	3.197E+01	6.171E+01	1.000E+02
2.50E-02	1.000E+02	6.544E+01	3.728E+01	1.860E+01	8.830E+00	5.909E+00	8.830E+00	1.860E+01	3.728E+01	6.544E+01	1.000E+02
3.00E-02	1.000E+02	6.830E+01	4.163E+01	2.288E+01	1.235E+01	9.029E+00	1.235E+01	2.288E+01	4.163E+01	6.830E+01	1.000E+02
3.50E-02	1.000E+02	7.059E+01	4.531E+01	2.684E+01	1.598E+01	1.245E+01	1.598E+01	2.684E+01	4.531E+01	7.059E+01	1.000E+02
4.00E-02	1.000E+02	7.250E+01	4.851E+01	3.052E+01	1.963E+01	1.602E+01	1.963E+01	3.052E+01	4.851E+01	7.250E+01	1.000E+02
4.50E-02	1.000E+02	7.415E+01	5.136E+01	3.396E+01	2.323E+01	1.963E+01	2.323E+01	3.396E+01	5.136E+01	7.415E+01	1.000E+02
5.00E-02	1.000E+02	7.560E+01	5.394E+01	3.720E+01	2.674E+01	2.320E+01	2.674E+01	3.720E+01	5.394E+01	7.560E+01	1.000E+02
5.50E-02	1.000E+02	7.691E+01	5.631E+01	4.026E+01	3.014E+01	2.670E+01	3.014E+01	4.026E+01	5.631E+01	7.691E+01	1.000E+02
6.00E-02	1.000E+02	7.810E+01	5.850E+01	4.315E+01	3.341E+01	3.008E+01	3.341E+01	4.315E+01	5.850E+01	7.810E+01	1.000E+02
6.50E-02	1.000E+02	7.921E+01	6.056E+01	4.589E+01	3.655E+01	3.335E+01	3.655E+01	4.589E+01	6.056E+01	7.921E+01	1.000E+02
7.00E-02	1.000E+02	8.025E+01	6.249E+01	4.849E+01	3.955E+01	3.648E+01	3.955E+01	4.849E+01	6.249E+01	8.025E+01	1.000E+02
7.50E-02	1.000E+02	8.122E+01	6.431E+01	5.096E+01	4.242E+01	3.948E+01	4.242E+01	5.096E+01	6.431E+01	8.122E+01	1.000E+02
8.00E-02	1.000E+02	8.213E+01	6.604E+01	5.331E+01	4.516E+01	4.235E+01	4.516E+01	5.331E+01	6.604E+01	8.213E+01	1.000E+02
8.50E-02	1.000E+02	8.300E+01	6.768E+01	5.554E+01	4.777E+01	4.509E+01	4.777E+01	5.554E+01	6.768E+01	8.300E+01	1.000E+02
9.00E-02	1.000E+02	8.382E+01	6.923E+01	5.767E+01	5.026E+01	4.771E+01	5.026E+01	5.767E+01	6.923E+01	8.382E+01	1.000E+02
9.50E-02	1.000E+02	8.459E+01	7.071E+01	5.969E+01	5.263E+01	5.020E+01	5.263E+01	5.969E+01	7.071E+01	8.459E+01	1.000E+02
1.00E-01	1.000E+02	8.533E+01	7.211E+01	6.162E+01	5.489E+01	5.257E+01	5.489E+01	6.162E+01	7.211E+01	8.533E+01	1.000E+02
1.05E-01	1.000E+02	8.604E+01	7.344E+01	6.345E+01	5.704E+01	5.484E+01	5.704E+01	6.345E+01	7.344E+01	8.604E+01	1.000E+02
1.10E-01	1.000E+02	8.671E+01	7.471E+01	6.520E+01	5.910E+01	5.699E+01	5.910E+01	6.520E+01	7.471E+01	8.671E+01	1.000E+02
1.15E-01	1.000E+02	8.734E+01	7.592E+01	6.686E+01	6.105E+01	5.905E+01	6.105E+01	6.686E+01	7.592E+01	8.734E+01	1.000E+02
1.20E-01	1.000E+02	8.795E+01	7.707E+01	6.845E+01	6.291E+01	6.100E+01	6.291E+01	6.845E+01	7.707E+01	8.795E+01	1.000E+02
1.25E-01	1.000E+02	8.852E+01	7.817E+01	6.995E+01	6.468E+01	6.286E+01	6.468E+01	6.995E+01	7.817E+01	8.852E+01	1.000E+02
1.30E-01	1.000E+02	8.907E+01	7.921E+01	7.139E+01	6.637E+01	6.464E+01	6.637E+01	7.139E+01	7.921E+01	8.907E+01	1.000E+02
1.35E-01	1.000E+02	8.959E+01	8.021E+01	7.276E+01	6.797E+01	6.633E+01	6.797E+01	7.276E+01	8.021E+01	8.959E+01	1.000E+02
1.40E-01	1.000E+02	9.009E+01	8.115E+01	7.406E+01	6.950E+01	6.794E+01	6.950E+01	7.406E+01	8.115E+01	9.009E+01	1.000E+02
1.45E-01	1.000E+02	9.056E+01	8.205E+01	7.530E+01	7.096E+01	6.947E+01	7.096E+01	7.530E+01	8.205E+01	9.056E+01	1.000E+02

1.50E-01	1.000E+02	9.102E+01	8.291E+01	7.648E+01	7.235E+01	7.093E+01	7.235E+01	7.648E+01	8.291E+01	9.102E+01	1.000E+02
1.55E-01	1.000E+02	9.144E+01	8.373E+01	7.760E+01	7.367E+01	7.231E+01	7.367E+01	7.760E+01	8.373E+01	9.144E+01	1.000E+02
1.60E-01	1.000E+02	9.185E+01	8.450E+01	7.867E+01	7.493E+01	7.364E+01	7.493E+01	7.867E+01	8.450E+01	9.185E+01	1.000E+02
1.65E-01	1.000E+02	9.224E+01	8.524E+01	7.969E+01	7.613E+01	7.490E+01	7.613E+01	7.969E+01	8.524E+01	9.224E+01	1.000E+02
1.70E-01	1.000E+02	9.261E+01	8.595E+01	8.066E+01	7.727E+01	7.610E+01	7.727E+01	8.066E+01	8.595E+01	9.261E+01	1.000E+02
1.75E-01	1.000E+02	9.297E+01	8.662E+01	8.159E+01	7.835E+01	7.724E+01	7.835E+01	8.159E+01	8.662E+01	9.297E+01	1.000E+02
1.80E-01	1.000E+02	9.330E+01	8.726E+01	8.246E+01	7.939E+01	7.833E+01	7.939E+01	8.246E+01	8.726E+01	9.330E+01	1.000E+02
1.85E-01	1.000E+02	9.362E+01	8.787E+01	8.330E+01	8.037E+01	7.936E+01	8.037E+01	8.330E+01	8.787E+01	9.362E+01	1.000E+02
1.90E-01	1.000E+02	9.393E+01	8.845E+01	8.410E+01	8.131E+01	8.035E+01	8.131E+01	8.410E+01	8.845E+01	9.393E+01	1.000E+02
1.95E-01	1.000E+02	9.422E+01	8.900E+01	8.486E+01	8.220E+01	8.129E+01	8.220E+01	8.486E+01	8.900E+01	9.422E+01	1.000E+02
2.00E-01	1.000E+02	9.449E+01	8.953E+01	8.558E+01	8.305E+01	8.218E+01	8.305E+01	8.558E+01	8.953E+01	9.449E+01	1.000E+02
2.05E-01	1.000E+02	9.476E+01	9.003E+01	8.627E+01	8.386E+01	8.303E+01	8.386E+01	8.627E+01	9.003E+01	9.476E+01	1.000E+02
2.10E-01	1.000E+02	9.501E+01	9.050E+01	8.693E+01	8.463E+01	8.384E+01	8.463E+01	8.693E+01	9.050E+01	9.501E+01	1.000E+02
2.15E-01	1.000E+02	9.525E+01	9.096E+01	8.755E+01	8.537E+01	8.461E+01	8.537E+01	8.755E+01	9.096E+01	9.525E+01	1.000E+02
2.20E-01	1.000E+02	9.547E+01	9.139E+01	8.815E+01	8.607E+01	8.535E+01	8.607E+01	8.815E+01	9.139E+01	9.547E+01	1.000E+02
2.25E-01	1.000E+02	9.569E+01	9.180E+01	8.871E+01	8.673E+01	8.605E+01	8.673E+01	8.871E+01	9.180E+01	9.569E+01	1.000E+02
2.30E-01	1.000E+02	9.589E+01	9.219E+01	8.925E+01	8.737E+01	8.672E+01	8.737E+01	8.925E+01	9.219E+01	9.589E+01	1.000E+02
2.35E-01	1.000E+02	9.609E+01	9.256E+01	8.977E+01	8.797E+01	8.735E+01	8.797E+01	8.977E+01	9.256E+01	9.609E+01	1.000E+02
2.40E-01	1.000E+02	9.628E+01	9.292E+01	9.025E+01	8.854E+01	8.795E+01	8.854E+01	9.025E+01	9.292E+01	9.628E+01	1.000E+02
2.45E-01	1.000E+02	9.646E+01	9.326E+01	9.072E+01	8.909E+01	8.853E+01	8.909E+01	9.072E+01	9.326E+01	9.646E+01	1.000E+02
2.50E-01	1.000E+02	9.662E+01	9.358E+01	9.116E+01	8.961E+01	8.908E+01	8.961E+01	9.116E+01	9.358E+01	9.662E+01	1.000E+02

第五節　絕對穩定的顯式有限差分程序

Visual Basic

隱式差分法具有絕對穩定性，但卻需要較複雜的計算。第二節所介紹的單一步驟顯式差分法計算簡單，但卻有 $\lambda \le 1/2$ 穩定性限制的考量。

除了第二節所介紹的顯式差分法以外，以下介紹幾種沒有這種穩定性限制的顯式差分法。

1. 督福特 - 法蘭克法

督福特 - 法蘭克法 (Dufort-Frankel Method) 利用三個積分步驟的資料建構差分方程式，如圖 11.7 所示，其差分表示式爲

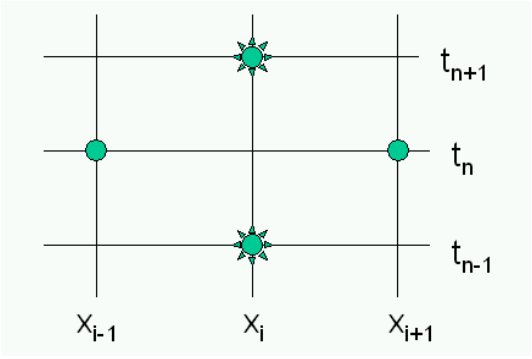


圖 11.7　督福特 - 法蘭克法推算 u_{xx}

$$\frac{u_{i,n+1}-u_{i,n-1}}{\Delta t}=\frac{u_{i-1,n}-u_{i,n-1}-u_{i,n+1}+u_{i+1,n}}{\Delta x^2} \qquad \textbf{(11-5.1)}$$

這種方法其實就是將時間差分及空間差分都利用前一步驟的時間點建立。其中空間差分是利用$t = t_n$所得到的資料建立；時間差分則利用$t = t_{n-1}$及$t = t_{n+1}$的資料建立在$t = t_n$時的微分平均值。利用這種方法由於需使用前二時間點上的資料，在編寫程式時，第一步驟可以先利用顯式差分法計算$t = t_1$的值；然後，由$t = t_2$開始，即利用督福特-法蘭克法的差分方程式 (11-5.1) 作計算。

2. 沙爾業夫法

督福特 - 法蘭克法雖然使用前二時間點的資料，建構差分方程式，已經明顯改善顯式差分法的缺點。但這種方法並未善用全部已經更新的資料。

沙爾業夫法 (Saul'yev Method) 就是進一步改善督福特-法蘭克法的策略，將已經獲得的資料妥善利用。其做法基本上是利用交錯方向執行積分，如圖 11.8 所示，積分時第一步驟由 $x = 0$ 開始，採用 x 正方向積分；第二步驟由 x 的最大值開始，採用 x 反向積分。執行積分時，以成對方式進行，一次正向積分，一次反向積分。其差分表示式爲

$$\frac{u_{i,n+1}-u_{i,n}}{\Delta t}=\frac{u_{i-1,n+1}-u_{i,n+1}-u_{i,n}+u_{i+1,n}}{\Delta x^2} \qquad \textbf{(11-5.2)}$$

$$\frac{u_{i,n+2}-u_{i,n+1}}{\Delta t}=\frac{u_{i-1,n+1}-u_{i,n+1}-u_{i,n+2}+u_{i+1,n+2}}{\Delta x^2} \qquad \textbf{(11-5.3)}$$

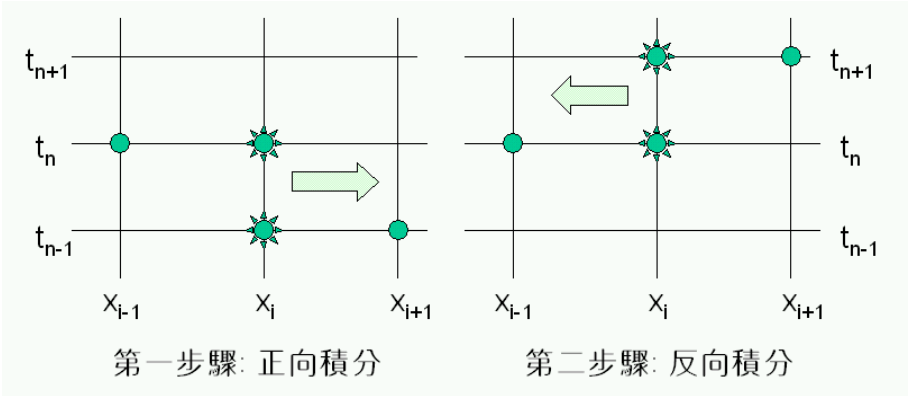


圖 11.8 沙爾業夫法利用交錯方向執行積分

3. 拉金法

拉金法 (Larkin Method) 與沙爾業夫法相似，利用交錯方向執行積分，如圖 11.9 所示，積分時由 $x = 0$ 開始，採用 x 正方向積分一次；在同一時間點，再由 x 的最大值開始，採用 x 反向再積分一次。然後，取其平均值作爲函數近似值。其差分表示式爲

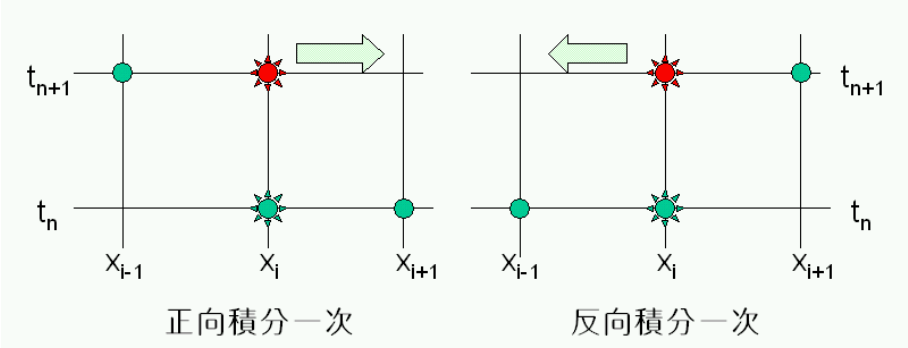


圖 11.9　拉金法利用正反二方向積分取平均值

$$\frac{p_{i,n+1}-u_{i,n}}{\Delta t}=\frac{p_{i-1,n+1}-p_{i,n+1}-u_{i,n}+u_{i+1,n}}{\Delta x^2} \qquad \textbf{(11-5.4)}$$

$$\frac{q_{i,n+1}-u_{i,n}}{\Delta t}=\frac{u_{i-1,n}-u_{i,n}-q_{i,n+1}+q_{i+1,n+1}}{\Delta x^2} \qquad \textbf{(11-5.5)}$$

$$u_{i,n+1}=\frac{1}{2}(p_{i,n+1}+q_{i,n+1}) \qquad \textbf{(11-5.6)}$$

第六節　二維及三維偏微分方程式

Visual Basic

二維拋物線偏微分方程式

考慮二維拋物線偏微分方程式的典型實例—二維熱傳導問題

$$\frac{\partial u}{\partial t}=\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2} \qquad \textbf{(11-6.1)}$$

仿照前面各節的說明，可以將方程式 (11-6.1) 改寫成差分方程式。

1. 顯式差分法

方程式 (11-6.1) 改寫成顯式差分方程式時，只要將 t 方向往前邁進一步，x 及 y 方向的導函數則利用已有資料計算之。

$$\frac{u_{i,j,n+1}-u_{i,j,n}}{\Delta t}=\frac{u_{i-1,j,n}-2u_{i,j,n}+u_{i+1,j,n}}{\Delta x^2}+\frac{u_{i,j-1,n}-2u_{i,j,n}+u_{i,j+1,n}}{\Delta y^2}$$

$$\equiv \delta_x^2 u_{i,j,n}+\delta_y^2 u_{i,j,n} \qquad \textbf{(11-6.2)}$$

使用顯式差分法，由於計算簡單，此處不再詳加說明其計算方法。但應注意，使用顯式差分法時，要注意其穩定條件爲

$$\Delta t\left[(\Delta x)^{-2}+(\Delta y)^{-2}\right]\le\frac{1}{2} \qquad \textbf{(11-6.3)}$$

2. 隱式差分法

方程式 (11-6.1) 改寫成隱式差分方程式時，x及y方向的導函數要利用t_{n+1}的資料計算之。

$$\frac{u_{i,j,n+1}-u_{i,j,n}}{\Delta t}=\frac{u_{i-1,j,n+1}-2u_{i,j,n+1}+u_{i+1,j,n+1}}{\Delta x^2}+\frac{u_{i,j-1,n+1}-2u_{i,j,n+1}+u_{i,j+1,n+1}}{\Delta y^2}$$

$$\equiv \delta_x^2 u_{i,j,n+1}+\delta_y^2 u_{i,j,n+1} \qquad \textbf{(11-6.4)}$$

爲了簡化分析起見，若假設 $\Delta x=\Delta y$ 且 $\lambda=\Delta t/(\Delta x)^2$；則方程式 (11-6.4) 可以整理成爲

$$-\lambda u_{i-1,j,n+1}-\lambda u_{i,j-1,n+1}+(1+4\lambda)u_{i,j,n+1}-\lambda u_{i,j+1,n+1}-\lambda u_{i+1,j,n+1}=u_{i,j,n}$$

(11-6.5)

所得到的差分方程式每一方程式有五項未知數，無法利用三對角線方程式的解法求解，但可以利用本書第四章所介紹的高斯消去法或高斯 - 賽德迭代法求解。一般而言，利用高斯消去法由於會使計算誤差累加，重複計算將使誤差逐漸增大；當我們將 Δx 及 Δt 取較小的值，希望得到較高計算準確度時，卻要面對解方程式的計算誤差累增的困擾。但若利用高斯 - 賽德迭代法計算，由於誤差不會累加，通常能得到較滿意的結果。這種方法爲絕對穩定的方法，但計算較麻煩，是其主要缺點。

3. 道格拉斯隱式交替差分法

方程式 (11-6.1) 改寫成隱式差分方程式時，如上一小節所討論，由於所得到的差分方程式必須利用完整的矩陣方程式求解，而無法使用效率較好的三對角線矩陣解法。爲了改善這種缺點，道格拉斯 (Douglas) 乃提出隱式交替差分法，引進一個在$t = t_{n+1/2}$時的中間步驟變數v。首先在$t = t_{n+1/2}$時，計算 x方向的差分，y方向的導函數則利用$t = t_n$時的資料計算之。其次，再在$t = t_{n+1}$時計算 y方向的差分，x方向的導函數則利用前一中間步驟$t = t_{n+1/2}$的資料計算之。

道格拉斯隱式交替差分法寫成差分方程式如下：

$$\frac{v_{i,j,n+1/2}-u_{i,j,n}}{\Delta t/2}=\frac{v_{i-1,j,n+1/2}-2v_{i,j,n+1/2}+v_{i+1,j,n+1/2}}{\Delta x^2}+\frac{u_{i,j-1,n+1}-2u_{i,j,n+1}+u_{i,j+1,n+1}}{\Delta y^2}$$
$$\equiv \delta_x^2 v_{i,j,n+1/2}+\delta_y^2 u_{i,j,n+1} \qquad \textbf{(11-6.6)}$$

$$\frac{u_{i,j,n+1}-v_{i,j,n+1/2}}{\Delta t/2}=\frac{v_{i-1,j,n+1/2}-2v_{i,j,n+1/2}+v_{i+1,j,n+1/2}}{\Delta x^2}+\frac{u_{i,j-1,n+1}-2u_{i,j,n+1}+u_{i,j+1,n+1}}{\Delta y^2}$$
$$\equiv \delta_x^2 v_{i,j,n+1/2}+\delta_y^2 u_{i,j,n+1} \qquad \textbf{(11-6.7)}$$

整理成三對角線差分方程式形式，得到

$$-\lambda v_{i-1,j,n+1/2}+2(1+\lambda)v_{i,j,n+1/2}-\lambda v_{i+1,j,n+1/2}=\lambda u_{i,j-1,n}+2(1-\lambda)u_{i,j,n}+\lambda u_{i,j+1,n} \qquad \textbf{(11-6.8)}$$

$$-\lambda u_{i-1,j,n+1}+2(1+\lambda)u_{i,j,n+1}-\lambda u_{i+1,j,n+1}=\lambda v_{i,j-1,n+1/2}+2(1-\lambda)v_{i,j,n+1/2}+\lambda v_{i,j+1,n+1/2} \qquad \textbf{(11-6.9)}$$

計算時，首先利用方程式 (11-6.8) 計算在$t = t_{n+1/2}$時的中間步驟變數v。然後，利用方程式 (11-6.9) 計算下一時刻 (亦即在$t = t_{n+1}$時) 的結果u。計算方程式 (11-6.8) 及方程式 (11-6.9) 時，都可以利用本書第四章所介紹的三對角線方程式解法求解。與隱式差分法比較，這種方法效率高且處理容易。

三維拋物線偏微分方程式

其次，考慮三維拋物線偏微分方程式的典型實例—三維熱傳導問題

$$\frac{\partial u}{\partial t}=\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2} \qquad \textbf{(11-6.10)}$$

仿照前面各節的說明，可以將方程式 (11-6.10) 改寫成差分方程式。

1. 科瑞克 - 尼可森隱式差分法 (Modified Crank-Nicolson Method)

首先，對 *x* 方向作科瑞克-尼可森隱式差分法處理，計算求出中間結果 *v*：

$$\frac{v_{i,j,k,n+1}-u_{i,j,k,n}}{\Delta t}=\frac{1}{2}\delta_x^2(v_{i,j,k,n+1}+u_{i,j,k,n})+\delta_y^2 u_{i,j,k,n}+\delta_z^2 u_{i,j,k,n} \qquad \textbf{(11-6.11)}$$

其次，再對 *y* 方向作科瑞克-尼可森隱式差分法處理，計算求出中間結果 *p*：

$$\frac{p_{i,j,k,n+1}-u_{i,j,k,n}}{\Delta t}=\frac{1}{2}\delta_x^2(v_{i,j,k,n+1}+u_{i,j,k,n})+\frac{1}{2}\delta_y^2(p_{i,j,k,n+1}+u_{i,j,k,n})+\delta_z^2 u_{i,j,k,n} \qquad \textbf{(11-6.12)}$$

最後，再對*z*方向作科瑞克-尼可森隱式差分法處理，計算求出結果u_{n+1}：

$$\frac{u_{i,j,k,n+1}-u_{i,j,k,n}}{\Delta t}=\frac{1}{2}\delta_x^2(v_{i,j,k,n+1}+u_{i,j,k,n})+\frac{1}{2}\delta_y^2(p_{i,j,k,n+1}+u_{i,j,k,n})+\frac{1}{2}\delta_z^2(u_{i,j,k,n+1}+u_{i,j,k,n}) \qquad \textbf{(11-6.13)}$$

這些差分方程式都可以利用三對角線方程式的解法求解。

2. 布萊恩隱式差分法 (Brain Method)

首先，在$t = t_{n+1/2}$時，先對*x*方向作隱式差分法處理，計算求出中間結果*v*：

$$\frac{v_{i,j,k,n+1/2}-u_{i,j,k,n}}{\Delta t/2}=\delta_x^2(v_{i,j,k,n+1/2})+\delta_y^2 u_{i,j,k,n}+\delta_z^2 u_{i,j,k,n} \qquad \textbf{(11-6.14)}$$

其次，再在$t = t_{n+1/2}$時，對*y*方向作隱式差分法處理，計算求出中間結果*p*：

$$\frac{p_{i,j,k,n+1/2}-u_{i,j,k,n}}{\Delta t/2}=\delta_x^2(v_{i,j,k,n+1/2})+\delta_y^2(p_{i,j,k,n+1/2})+\delta_z^2 u_{i,j,k,n} \qquad \textbf{(11-6.15)}$$

最後，再對z方向作隱式差分法處理，計算求出結果u_{n+1}：

$$\frac{u_{i,j,k,n+1}-p_{i,j,k,n+1/2}}{\Delta t/2}=\delta_x^2(v_{i,j,k,n+1/2})+\delta_y^2(p_{i,j,k,n+1/2})+\delta_z^2(u_{i,j,k,n+1}) \quad \textbf{(11-6.16)}$$

這些差分方程式也都可以利用三對角線方程式的解法求解。

同時含有一次微分及二次微分的偏微分方程式

在圓柱座標及球形座標中，一次微分及二次微分常常同時存在。其處理方式，可以仿照以下的實例處理。

$$\frac{\partial^2 u}{\partial r^2}+\frac{s}{r}\frac{\partial u}{\partial r}\cong\frac{u_{i-1}-2u_i+u_{i+1}}{(\Delta r)^2}+\frac{s}{i\Delta r}\frac{u_{i+1}-u_{i-1}}{2\Delta r} \quad \textbf{(11-6.17)}$$

例題 11-3　流體在垂直加熱平板邊的輸送現象

一體積無限大靜止的冷流體，與加熱的垂直平板接觸。如圖 11.10 所示。假設此流體是不可壓縮流體，但受熱溫度改變時會有浮力改變現象。

令τ，X 及 Y 分別爲無因次時間及距離座標。Pr 爲該流體的普郎特數 (Prandtl Number)。則此系統的簡化境界層方程式爲：

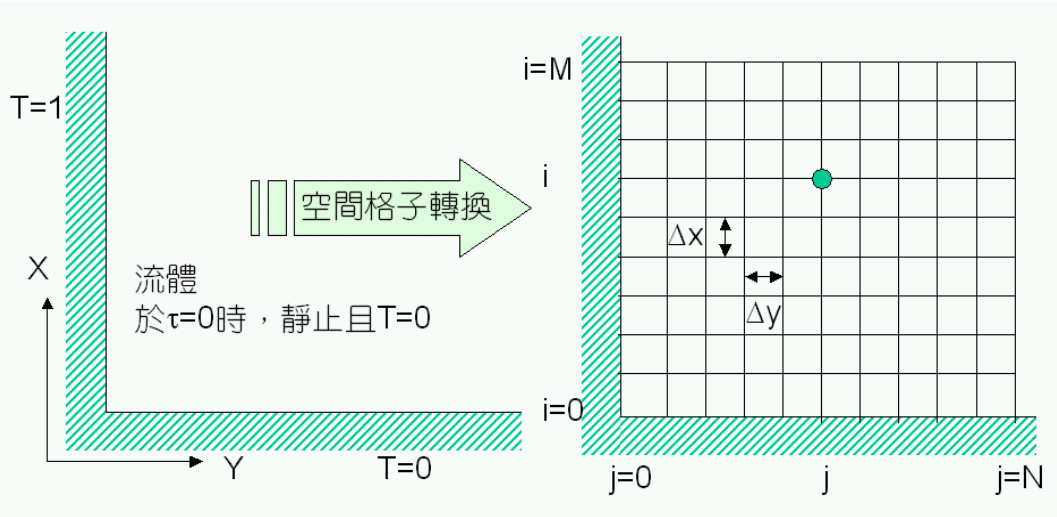


圖 11.10　垂直加熱平板邊與邊界層

動量平衡方程式

$$\frac{\partial U}{\partial \tau}+U\frac{\partial U}{\partial X}+V\frac{\partial U}{\partial Y}=T+\frac{\partial^2 U}{\partial Y^2} \qquad \textbf{(11-6.18)}$$

能量平衡方程式

$$\frac{\partial T}{\partial \tau}+U\frac{\partial T}{\partial X}+V\frac{\partial T}{\partial Y}=\frac{1}{\mathrm{Pr}}\frac{\partial^2 T}{\partial Y^2} \qquad \textbf{(11-6.19)}$$

質量平衡方程式

$$\frac{\partial U}{\partial X}+\frac{\partial V}{\partial Y}=0 \qquad \textbf{(11-6.20)}$$

起始條件爲

$$\begin{aligned} X=0 \quad &; \quad U=V=T=0 \\ Y=0 \quad &; \quad U=V=0,\ T=1 \\ Y=\infty \quad &; \quad U=V=T=0 \\ \tau=0 \quad &; \quad U=V=T=0 \end{aligned}$$

試解本問題所列聯立偏微分方程式，建立 U, V 及 T 與自變數 τ, X 及 Y 的函數關係。並找出穩定態解答 (Steady State Solution)。

解：

雖然所考慮的問題是體積無限大靜止的冷流體，與加熱的垂直平板接觸。但是，由於流體受邊界的影響到一定距離後，其影響就快速減弱；因此，可以將此問題考慮成有限空間的流體受熱問題。例如，可以考慮假設平板長度X_{max} = 100，Y_{max} = 25 即當作是無窮遠。差分格子化做法，如圖 11.10 所示。爲了簡化說明起見，本例題採用顯式差分法加以說明。

首先將微分方程式 (11-6.18)、(11-6.19) 及 (11-6.20) 改寫成差分方程式

$$\begin{aligned} &\frac{U^*_{i,j}-U_{i,j}}{\Delta\tau}+U_{i,j}\frac{U_{i,j}-U_{i-1,j}}{\Delta x}+V_{i,j}\frac{U_{i,j+1}-U_{i,j}}{\Delta y} \\ &\qquad =T^*_{i,j}+\frac{U_{i,j+1}-2U_{i,j}+U_{i,j-1}}{(\Delta y)^2} \end{aligned} \qquad \textbf{(11-6.21)}$$

$$\frac{T^*_{i,j}-T_{i,j}}{\Delta\tau}+U_{i,j}\frac{T_{i,j}-T_{i-1,j}}{\Delta x}+V_{i,j}\frac{T_{i,j+1}-T_{i,j}}{\Delta y}=\frac{1}{\mathrm{Pr}}\frac{T_{i,j+1}-2T_{i,j}+T_{i,j-1}}{(\Delta y)^2} \qquad \textbf{(11-6.22)}$$

$$\frac{U^*_{i,j} - U^*_{i-1,j}}{\Delta x} + \frac{V^*_{i,j} - V^*_{i,j-1}}{\Delta y} = 0 \qquad \textbf{(11-6.23)}$$

計算策略：

1. 在每一計算步驟中，先假設$U_{i,j}$、$V_{i,j}$、及$T_{i,j}$均爲定數，利用方程式 (11-6.22) 先計算$T^*_{i,j}$。
2. 利用方程式 (11-6..21) 計算$U^*_{i,j}$。
3. 利用方程式 (11-6..23) 計算$V^*_{i,j}$。
4. 完成全部空間點的計算後，將計算值放置進$U_{i,j}$, $V_{i,j}$ 及$T_{i,j}$位置。
5. 作時間增量$t = t + \Delta\tau$，重複以上步驟。

程式列印：

```
'
'*****************************************
' SOLUTION OF Parabolic P.D.E.
' BY    Crank-Nicolson Method
'*****************************************
'
' Program Developed by Dr. Ron Hsin Chang
' Copyright 1988, 2001
'
' Example 11-3
'
Sub ImplicitFiniteDifference (Xpos, Ypos)
Dim U (100, 100) , UNew (100, 100) , V (100, 100) , T (100, 100) , TNew (100, 100) As Double
Call GetData (M, N, Iprt, Jmax, Xmax, Ymax, PR, Dtau, TauMax)
Nm1 = N - 1
Mm1 = M - 1
DX = Xmax / M
DY = Ymax / N
DYsq = DY * DY
DYsqPR = DYsq * PR
YoverX = DY / DX
Icount = 0
For I = 0 To M
      For J = 0 To N
            U (I, J) = 0
            UNew (I, J) = 0
            V (I, J) = 0
            T (I, J) = 0
            TNew (I, J) = 0
      Next J
Next I
For I = 0 To M
      T (I, 0) = 1
```

```
Next I
'
'     Perform Calculation Over Successive TimeStep
'
      Tau = 0
      Do
            Tau = Tau + Dtau
            Icount = Icount + 1
'
'           Calculate New Temperature
'
            For I = 1 To M
                  For J = 1 To Nm1
                        A = (T (I, J + 1) - 2 * T (I, J) + T (I, J - 1) ) / DYsqPR
                        A = A - U (I, J) * (T (I, J) - T (I - 1, J) ) / DX
                        A = A - V (I, J) * (T (I, J + 1) - T (I, J) ) / DY
                        TNew (I, J) = T (I, J) + Dtau * A
                  Next J
            Next I
'
'           Calculate New Velocity U
'
            For I = 1 To M
                  For J = 1 To Nm1
                        A = (U (I, J + 1) - 2 * U (I, J) + U (I, J - 1) ) / DYsq + TNew (I, J)
                        A = A - U (I, J) * (U (I, J) - U (I - 1, J) ) / DX
                        A = A - V (I, J) * (U (I, J + 1) - U (I, J) ) / DY
                        UNew (I, J) = U (I, J) + Dtau * A
                  Next J
            Next I
'
'           Calculate New Velocity V
'
            For I = 1 To M
                  For J = 1 To N
                        V (I, J) = V (I, J - 1) + YoverX * (UNew (I - 1, J) - UNew (I, J) )
                  Next J
            Next I
'
'           Substitute Tnew and Unew into T and U
'
            For I = 1 To M
                  For J = 1 To N - 1
                        U (I, J) = UNew (I, J)
                        T (I, J) = TNew (I, J)
                  Next J
            Next I
'
'           Print U, V, and T
'
            If (Icount = Iprt) Then
                  Icount = 0
                  Debug.Print "At Time = "; Tau
                  Debug.Print "Field of Velocity U"
```

```
                    For K = 0 To M
                        I = M - K
                        For J = 0 To Jmax
                            Debug.Print Format (U (I, J) , "0.00000    ") ;
                        Next J
                        Debug.Print
                    Next K
                    Debug.Print "Field of Velocity V"
                    For K = 0 To M
                        I = M - K
                        For J = 0 To Jmax
                            Debug.Print Format (V (I, J) , "0.00000    ") ;
                        Next J
                        Debug.Print
                    Next K
                    Debug.Print "Field of Temperature T"
                    For K = 0 To M
                        I = M - K
                        For J = 0 To Jmax
                            Debug.Print Format (T (I, J) , "0.00000    ") ;
                        Next J
                        Debug.Print
                    Next K
                End If
            Loop While Tau <= TauMax
        End Sub


        Sub GetData (M, N, Iprt, Jmax, Xmax, Ymax, PR, Dtau, TauMax)
        M = 10
        M = Val (InputBox ("Number of Grid Spacing in X", "M", M) )
        Debug.Print "Number of X-Grid Spacing M = "; M

        N = 10
        N = Val (InputBox ("Number of Grid Spacing in Y", "N", N) )
        Debug.Print "Number of Y-Grid Spacing N = "; N

        Iprt = 20
        Iprt = Val (InputBox ("Number of Time steps between successive printing IPrt =", "IPrt", Iprt) )
        Debug.Print "Number of Time Steps Between Printing IPrt = "; Iprt

        Jmax = 10
        Jmax = Val (InputBox ("Largest Column Script JMax =", "JMax", Jmax) )
        Debug.Print "Largest Column Script JMax = "; Jmax

        Xmax = 100
        Xmax = Val (InputBox ("Height of the Plate XMax =", "XMax", Xmax) )
        Debug.Print "Height of the Plate XMax = "; Xmax

        Ymax = 25
        Ymax = Val (InputBox ("Maximum Distance from the Plate YMax =", "YMax", Ymax) )
        Debug.Print "Maximum Distance from the Plate YMax = "; Ymax

        PR = 0.733
```

```
    PR = Val (InputBox ("Prandtl Number PR", "PR", PR) )
    Debug.Print "Prandtl Numbr PR = "; PR

    Dtau = 0.5
    Dtau = Val (InputBox ("Time Step Dtau =", "Dtau", Dtau) )
    Debug.Print "Time Step Dtau = "; Dtau

    TauMax = 80
    TauMax = Val (InputBox ("Maximum Time TauMax =", "TauMax", TauMax) )
    Debug.Print "Maximum Time TauMax = "; TauMax

    End Sub
```

執行結果：

```
Number of X-Grid Spacing M =  10
Number of Y-Grid Spacing N =  10
Number of Time Steps Between Printing IPrt =  200
Largest Column Script JMax =  10
Height of the Plate XMax =  100
Maximum Distance from the Plate YMax =  25
Prandtl Numbr PR =  0.733
Time Step Dtau =  0.5
Maximum Time TauMax =  100
At Time =  100

Field of Velocity U
 0.00000   4.46951   5.02705   4.06646   2.86153   1.85420   1.12232   0.62708   0.30802   0.11225   0.00000
 0.00000   4.26863   4.71273   3.74780   2.59747   1.65995   0.99192   0.54758   0.26595   0.09590   0.00000
 0.00000   4.04958   4.37672   3.41431   2.32624   1.46367   0.86206   0.46947   0.22513   0.08023   0.00000
 0.00000   3.80818   4.01525   3.06427   2.04757   1.26571   0.73325   0.39316   0.18581   0.06533   0.00000
 0.00000   3.53851   3.62342   2.69567   1.76127   1.06662   0.60616   0.31918   0.14831   0.05133   0.00000
 0.00000   3.23195   3.19470   2.30616   1.46741   0.86733   0.48177   0.24827   0.11306   0.03841   0.00000
 0.00000   2.87515   2.72015   1.89311   1.16656   0.66938   0.36153   0.18142   0.08059   0.02677   0.00000
 0.00000   2.44596   2.18709   1.45378   0.86035   0.47540   0.24768   0.12011   0.05168   0.01667   0.00000
 0.00000   1.90398   1.57683   0.98630   0.55300   0.29029   0.14394   0.06658   0.02742   0.00850   0.00000
 0.00000   1.16413   0.86104   0.49335   0.25534   0.12396   0.05692   0.02446   0.00941   0.00275   0.00000
 0.00000   0.00000   0.00000   0.00000   0.00000   0.00000   0.00000   0.00000   0.00000   0.00000   0.00000

Field of Velocity V
 0.00000  -0.05022  -0.12880  -0.20846  -0.27448  -0.32304  -0.35564  -0.37552  -0.38604  -0.39012  -0.39012
 0.00000  -0.05476  -0.13876  -0.22213  -0.28994  -0.33901  -0.37148  -0.39100  -0.40121  -0.40513  -0.40513
 0.00000  -0.06035  -0.15072  -0.23823  -0.30790  -0.35739  -0.38959  -0.40867  -0.41850  -0.42222  -0.42222
 0.00000  -0.06742  -0.16537  -0.25753  -0.32910  -0.37887  -0.41064  -0.42914  -0.43851  -0.44201  -0.44201
 0.00000  -0.07664  -0.18382  -0.28120  -0.35466  -0.40448  -0.43558  -0.45331  -0.46212  -0.46535  -0.46535
 0.00000  -0.08920  -0.20784  -0.31110  -0.38632  -0.43581  -0.46587  -0.48258  -0.49069  -0.49360  -0.49360
 0.00000  -0.10730  -0.24056  -0.35039  -0.42694  -0.47544  -0.50390  -0.51923  -0.52645  -0.52898  -0.52898
 0.00000  -0.13550  -0.28806  -0.40493  -0.48177  -0.52805  -0.55398  -0.56737  -0.57343  -0.57548  -0.57548
 0.00000  -0.18496  -0.36391  -0.48714  -0.56156  -0.60314  -0.62490  -0.63543  -0.63993  -0.64137  -0.64137
```

0.00000	-0.29103	-0.50629	-0.62963	-0.69347	-0.72446	-0.73869	-0.74480	-0.74715	-0.74784	-0.74784
0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

Field of Temperature T

1.00000	0.70450	0.45040	0.26914	0.15454	0.08651	0.04722	0.02471	0.01180	0.00434	0.00000
1.00000	0.69570	0.43735	0.25686	0.14513	0.08005	0.04311	0.02229	0.01052	0.00383	0.00000
1.00000	0.68545	0.42255	0.24330	0.13498	0.07322	0.03884	0.01980	0.00923	0.00332	0.00000
1.00000	0.67328	0.40553	0.22820	0.12397	0.06599	0.03440	0.01726	0.00793	0.00282	0.00000
1.00000	0.65845	0.38559	0.21116	0.11194	0.05829	0.02978	0.01467	0.00663	0.00232	0.00000
1.00000	0.63972	0.36168	0.19167	0.09869	0.05006	0.02497	0.01203	0.00533	0.00183	0.00000
1.00000	0.61487	0.33211	0.16892	0.08393	0.04124	0.01998	0.00937	0.00405	0.00136	0.00000
1.00000	0.57937	0.29387	0.14170	0.06727	0.03174	0.01481	0.00671	0.00281	0.00092	0.00000
1.00000	0.52205	0.24097	0.10789	0.04817	0.02152	0.00955	0.00412	0.00165	0.00052	0.00000
1.00000	0.40556	0.15894	0.06353	0.02590	0.01067	0.00438	0.00176	0.00066	0.00020	0.00000
1.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

Number of X-Grid Spacing M = 40
Number of Y-Grid Spacing N = 40
Number of Time Steps Between Printing IPrt = 1000
Largest Column Script JMax = 10
Height of the Plate XMax = 100
Maximum Distance from the Plate YMax = 25
Prandtl Numbr PR = 0.733
Time Step Dtau = 0.1
Maximum Time TauMax = 100
At Time = 99.9999999999986

Field of Velocity U

0.00000	1.67508	2.98841	3.97310	4.66566	5.10482	5.33009	5.38029	5.29216	5.09922	4.83107
0.00000	1.66294	2.96434	3.93756	4.61941	5.04890	5.26582	5.30920	5.21589	5.01947	4.74948
0.00000	1.65057	2.93980	3.90133	4.57226	4.99192	5.20038	5.23687	5.13838	4.93851	4.66675
0.00000	1.63794	2.91476	3.86437	4.52418	4.93386	5.13374	5.16328	5.05958	4.85629	4.58283
0.00000	1.62504	2.88919	3.82664	4.47513	4.87465	5.06584	5.08835	4.97944	4.77277	4.49768
0.00000	1.61187	2.86308	3.78811	4.42506	4.81425	4.99661	5.01204	4.89790	4.68789	4.41126
0.00000	1.59840	2.83638	3.74874	4.37391	4.75259	4.92600	4.93428	4.81491	4.60159	4.32352
0.00000	1.58462	2.80907	3.70848	4.32164	4.68961	4.85394	4.85500	4.73039	4.51383	4.23440
0.00000	1.57052	2.78112	3.66728	4.26818	4.62524	4.78036	4.77413	4.64429	4.42454	4.14386
0.00000	1.55607	2.75249	3.62510	4.21347	4.55942	4.70518	4.69160	4.55652	4.33365	4.05184
0.00000	1.54125	2.72315	3.58187	4.15743	4.49205	4.62832	4.60731	4.46701	4.24109	3.95828
0.00000	1.52605	2.69304	3.53754	4.10000	4.42306	4.54968	4.52118	4.37567	4.14678	3.86311
0.00000	1.51044	2.66212	3.49204	4.04108	4.35234	4.46916	4.43312	4.28241	4.05065	3.76627
0.00000	1.49438	2.63034	3.44528	3.98058	4.27980	4.38666	4.34300	4.18713	3.95261	3.66768
0.00000	1.47786	2.59764	3.39720	3.91840	4.20532	4.30205	4.25072	4.08972	3.85256	3.56726
0.00000	1.46084	2.56396	3.34769	3.85444	4.12876	4.21521	4.15615	3.99007	3.75039	3.46494
0.00000	1.44328	2.52923	3.29666	3.78855	4.05000	4.12598	4.05914	3.88804	3.64600	3.36061
0.00000	1.42515	2.49336	3.24399	3.72061	3.96888	4.03421	3.95954	3.78349	3.53927	3.25418
0.00000	1.40638	2.45626	3.18955	3.65045	3.88521	3.93973	3.85718	3.67627	3.43007	3.14554
0.00000	1.38695	2.41784	3.13321	3.57791	3.79882	3.84232	3.75187	3.56622	3.31824	3.03459
0.00000	1.36677	2.37798	3.07479	3.50277	3.70947	3.74177	3.64340	3.45313	3.20364	2.92119
0.00000	1.34580	2.33655	3.01411	3.42482	3.61692	3.63783	3.53154	3.33681	3.08608	2.80522
0.00000	1.32394	2.29339	2.95096	3.34380	3.52090	3.53021	3.41600	3.21702	2.96539	2.68653

0.00000	1.30111	2.24833	2.88509	3.25940	3.42106	3.41858	3.29651	3.09349	2.84135	2.56497
0.00000	1.27720	2.20117	2.81620	3.17128	3.31704	3.30259	3.17271	2.96595	2.71373	2.44035
0.00000	1.25208	2.15166	2.74397	3.07903	3.20840	3.18179	3.04422	2.83405	2.58227	2.31251
0.00000	1.22561	2.09951	2.66798	2.98218	3.09463	3.05570	2.91058	2.69743	2.44668	2.18123
0.00000	1.19761	2.04437	2.58775	2.88013	2.97513	2.92372	2.77128	2.55566	2.30664	2.04629
0.00000	1.16785	1.98581	2.50268	2.77221	2.84915	2.78517	2.62572	2.40826	2.16179	1.90748
0.00000	1.13605	1.92331	2.41204	2.65755	2.71583	2.63921	2.47318	2.25465	2.01174	1.76452
0.00000	1.10186	1.85618	2.31491	2.53509	2.57406	2.48484	2.31281	2.09420	1.85604	1.61718
0.00000	1.06482	1.78355	2.21009	2.40347	2.42249	2.32082	2.14361	1.92616	1.69419	1.46517
0.00000	1.02432	1.70426	2.09602	2.26092	2.25938	2.14562	1.96434	1.74964	1.52567	1.30827
0.00000	0.97951	1.61671	1.97057	2.10509	2.08245	1.95730	1.77352	1.56364	1.34990	1.14626
0.00000	0.92918	1.51860	1.83072	1.93273	1.88868	1.75334	1.56933	1.36703	1.16634	0.97908
0.00000	0.87146	1.40650	1.67205	1.73920	1.67388	1.53049	1.34954	1.15856	0.97458	0.80693
0.00000	0.80329	1.27481	1.48759	1.51752	1.43217	1.28445	1.11155	0.93711	0.77460	0.63055
0.00000	0.71908	1.11356	1.26551	1.25664	1.15494	1.00968	0.85263	0.70216	0.56744	0.45195
0.00000	0.60660	0.90217	0.98358	0.93806	0.82983	0.69987	0.57132	0.45557	0.35695	0.27591
0.00000	0.42941	0.58826	0.59499	0.53062	0.44187	0.35259	0.27330	0.20744	0.15497	0.11432
0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

Field of Velocity V

0.00000	-0.00303	-0.00905	-0.01793	-0.02950	-0.04348	-0.05955	-0.07732	-0.09639	-0.11633	-0.13672
0.00000	-0.00309	-0.00923	-0.01829	-0.03007	-0.04432	-0.06068	-0.07876	-0.09813	-0.11837	-0.13906
0.00000	-0.00316	-0.00942	-0.01866	-0.03068	-0.04519	-0.06185	-0.08025	-0.09995	-0.12051	-0.14149
0.00000	-0.00322	-0.00962	-0.01905	-0.03131	-0.04611	-0.06309	-0.08182	-0.10185	-0.12274	-0.14402
0.00000	-0.00329	-0.00982	-0.01946	-0.03197	-0.04708	-0.06438	-0.08346	-0.10384	-0.12506	-0.14667
0.00000	-0.00337	-0.01004	-0.01988	-0.03267	-0.04809	-0.06574	-0.08518	-0.10593	-0.12750	-0.14944
0.00000	-0.00344	-0.01027	-0.02034	-0.03340	-0.04915	-0.06716	-0.08698	-0.10811	-0.13005	-0.15233
0.00000	-0.00353	-0.01051	-0.02081	-0.03418	-0.05027	-0.06866	-0.08888	-0.11041	-0.13273	-0.15537
0.00000	-0.00361	-0.01077	-0.02132	-0.03499	-0.05145	-0.07024	-0.09088	-0.11282	-0.13554	-0.15855
0.00000	-0.00370	-0.01104	-0.02185	-0.03586	-0.05270	-0.07191	-0.09298	-0.11536	-0.13850	-0.16189
0.00000	-0.00380	-0.01133	-0.02241	-0.03677	-0.05402	-0.07368	-0.09521	-0.11804	-0.14162	-0.16541
0.00000	-0.00390	-0.01163	-0.02301	-0.03774	-0.05542	-0.07555	-0.09756	-0.12088	-0.14491	-0.16912
0.00000	-0.00401	-0.01196	-0.02365	-0.03877	-0.05691	-0.07753	-0.10006	-0.12388	-0.14839	-0.17304
0.00000	-0.00413	-0.01230	-0.02433	-0.03987	-0.05849	-0.07964	-0.10271	-0.12707	-0.15208	-0.17718
0.00000	-0.00426	-0.01268	-0.02505	-0.04104	-0.06018	-0.08189	-0.10554	-0.13045	-0.15599	-0.18157
0.00000	-0.00439	-0.01307	-0.02583	-0.04230	-0.06199	-0.08430	-0.10855	-0.13406	-0.16016	-0.18624
0.00000	-0.00453	-0.01350	-0.02667	-0.04365	-0.06394	-0.08688	-0.11178	-0.13791	-0.16460	-0.19120
0.00000	-0.00469	-0.01396	-0.02757	-0.04511	-0.06603	-0.08965	-0.11524	-0.14204	-0.16934	-0.19650
0.00000	-0.00486	-0.01446	-0.02855	-0.04669	-0.06829	-0.09264	-0.11897	-0.14648	-0.17444	-0.20217
0.00000	-0.00504	-0.01501	-0.02961	-0.04840	-0.07073	-0.09587	-0.12299	-0.15126	-0.17991	-0.20826
0.00000	-0.00524	-0.01560	-0.03077	-0.05026	-0.07340	-0.09938	-0.12735	-0.15643	-0.18582	-0.21481
0.00000	-0.00546	-0.01625	-0.03204	-0.05230	-0.07631	-0.10321	-0.13209	-0.16204	-0.19221	-0.22189
0.00000	-0.00571	-0.01697	-0.03344	-0.05454	-0.07950	-0.10740	-0.13728	-0.16816	-0.19917	-0.22956
0.00000	-0.00598	-0.01777	-0.03499	-0.05702	-0.08302	-0.11202	-0.14297	-0.17486	-0.20676	-0.23792
0.00000	-0.00628	-0.01866	-0.03672	-0.05978	-0.08694	-0.11713	-0.14926	-0.18223	-0.21510	-0.24706
0.00000	-0.00662	-0.01965	-0.03865	-0.06287	-0.09131	-0.12283	-0.15624	-0.19040	-0.22429	-0.25711
0.00000	-0.00700	-0.02079	-0.04084	-0.06635	-0.09623	-0.12922	-0.16405	-0.19949	-0.23450	-0.26823
0.00000	-0.00744	-0.02208	-0.04335	-0.07033	-0.10182	-0.13646	-0.17285	-0.20970	-0.24591	-0.28062
0.00000	-0.00795	-0.02358	-0.04624	-0.07490	-0.10823	-0.14472	-0.18286	-0.22126	-0.25877	-0.29451
0.00000	-0.00855	-0.02533	-0.04961	-0.08023	-0.11567	-0.15426	-0.19435	-0.23447	-0.27339	-0.31023

0.00000	-0.00926	-0.02742	-0.05362	-0.08653	-0.12442	-0.16542	-0.20772	-0.24974	-0.29020	-0.32820
0.00000	-0.01012	-0.02995	-0.05847	-0.09410	-0.13488	-0.17868	-0.22350	-0.26763	-0.30976	-0.34899
0.00000	-0.01120	-0.03309	-0.06445	-0.10341	-0.14764	-0.19472	-0.24243	-0.28893	-0.33287	-0.37337
0.00000	-0.01258	-0.03711	-0.07207	-0.11516	-0.16360	-0.21459	-0.26564	-0.31479	-0.36068	-0.40248
0.00000	-0.01443	-0.04246	-0.08212	-0.13051	-0.18420	-0.23992	-0.29486	-0.34698	-0.39492	-0.43796
0.00000	-0.01704	-0.04996	-0.09608	-0.15150	-0.21193	-0.27344	-0.33294	-0.38830	-0.43830	-0.48239
0.00000	-0.02105	-0.06137	-0.11689	-0.18211	-0.25141	-0.32011	-0.38483	-0.44357	-0.49536	-0.54001
0.00000	-0.02812	-0.08097	-0.15145	-0.23109	-0.31237	-0.38982	-0.46015	-0.52180	-0.57442	-0.61843
0.00000	-0.04430	-0.12277	-0.21992	-0.32178	-0.41877	-0.50559	-0.58010	-0.64213	-0.69263	-0.73302
0.00000	-0.10735	-0.25442	-0.40316	-0.53582	-0.64629	-0.73443	-0.80276	-0.85462	-0.89336	-0.92194
0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

Field of Temperature T

1.00000	0.92782	0.85583	0.78446	0.71431	0.64607	0.58044	0.51806	0.45949	0.40512	0.35522
1.00000	0.92734	0.85487	0.78305	0.71247	0.64385	0.57789	0.51526	0.45650	0.40203	0.35210
1.00000	0.92684	0.85389	0.78159	0.71057	0.64155	0.57526	0.51237	0.45343	0.39886	0.34889
1.00000	0.92633	0.85286	0.78007	0.70860	0.63918	0.57255	0.50939	0.45027	0.39559	0.34559
1.00000	0.92580	0.85181	0.77851	0.70656	0.63672	0.56974	0.50632	0.44701	0.39223	0.34220
1.00000	0.92524	0.85071	0.77689	0.70446	0.63418	0.56685	0.50314	0.44365	0.38876	0.33872
1.00000	0.92467	0.84957	0.77521	0.70227	0.63155	0.56384	0.49986	0.44017	0.38519	0.33513
1.00000	0.92408	0.84839	0.77346	0.70000	0.62882	0.56074	0.49646	0.43659	0.38150	0.33144
1.00000	0.92346	0.84716	0.77164	0.69764	0.62599	0.55751	0.49294	0.43287	0.37770	0.32763
1.00000	0.92282	0.84588	0.76975	0.69519	0.62304	0.55416	0.48929	0.42903	0.37376	0.32370
1.00000	0.92214	0.84455	0.76778	0.69263	0.61998	0.55068	0.48551	0.42505	0.36969	0.31964
1.00000	0.92144	0.84315	0.76572	0.68997	0.61678	0.54706	0.48157	0.42091	0.36548	0.31544
1.00000	0.92071	0.84169	0.76357	0.68718	0.61345	0.54328	0.47747	0.41662	0.36111	0.31110
1.00000	0.91994	0.84016	0.76132	0.68426	0.60996	0.53934	0.47320	0.41215	0.35657	0.30660
1.00000	0.91913	0.83856	0.75896	0.68121	0.60631	0.53522	0.46874	0.40750	0.35185	0.30194
1.00000	0.91828	0.83687	0.75647	0.67800	0.60248	0.53090	0.46409	0.40265	0.34695	0.29710
1.00000	0.91738	0.83510	0.75385	0.67462	0.59846	0.52637	0.45921	0.39758	0.34183	0.29207
1.00000	0.91644	0.83322	0.75109	0.67106	0.59422	0.52161	0.45410	0.39228	0.33650	0.28683
1.00000	0.91543	0.83123	0.74817	0.66730	0.58975	0.51660	0.44872	0.38672	0.33092	0.28138
1.00000	0.91437	0.82912	0.74507	0.66331	0.58502	0.51131	0.44306	0.38088	0.32508	0.27568
1.00000	0.91324	0.82687	0.74176	0.65907	0.58000	0.50571	0.43708	0.37473	0.31895	0.26972
1.00000	0.91202	0.82447	0.73824	0.65455	0.57466	0.49976	0.43076	0.36825	0.31250	0.26348
1.00000	0.91072	0.82189	0.73446	0.64971	0.56896	0.49343	0.42404	0.36140	0.30571	0.25692
1.00000	0.90932	0.81911	0.73040	0.64452	0.56286	0.48667	0.41690	0.35412	0.29854	0.25003
1.00000	0.90780	0.81611	0.72601	0.63892	0.55629	0.47942	0.40927	0.34639	0.29094	0.24275
1.00000	0.90615	0.81285	0.72125	0.63285	0.54919	0.47161	0.40108	0.33813	0.28287	0.23507
1.00000	0.90435	0.80927	0.71604	0.62623	0.54148	0.46317	0.39227	0.32928	0.27426	0.22691
1.00000	0.90235	0.80534	0.71031	0.61898	0.53306	0.45398	0.38273	0.31975	0.26505	0.21824
1.00000	0.90014	0.80097	0.70397	0.61096	0.52379	0.44393	0.37235	0.30945	0.25515	0.20898
1.00000	0.89765	0.79607	0.69687	0.60203	0.51351	0.43284	0.36098	0.29825	0.24447	0.19905
1.00000	0.89483	0.79052	0.68884	0.59198	0.50201	0.42052	0.34844	0.28599	0.23287	0.18835
1.00000	0.89159	0.78414	0.67965	0.58052	0.48899	0.40669	0.33448	0.27247	0.22019	0.17678
1.00000	0.88779	0.77668	0.66894	0.56727	0.47405	0.39097	0.31878	0.25743	0.20624	0.16417
1.00000	0.88323	0.76776	0.65622	0.55163	0.45662	0.37284	0.30091	0.24052	0.19076	0.15035
1.00000	0.87761	0.75679	0.64067	0.53274	0.43583	0.35154	0.28024	0.22127	0.17340	0.13509
1.00000	0.87038	0.74275	0.62098	0.50916	0.41034	0.32594	0.25588	0.19903	0.15372	0.11808
1.00000	0.86050	0.72370	0.59466	0.47835	0.37788	0.29420	0.22647	0.17285	0.13110	0.09897
1.00000	0.84558	0.69533	0.55659	0.43535	0.33434	0.25324	0.18990	0.14138	0.10473	0.07732
1.00000	0.81828	0.64556	0.49382	0.36913	0.27152	0.19758	0.14276	0.10269	0.07367	0.05276

1.00000 0.73912 0.52511 0.36514 0.25110 0.17179 0.11731 0.08009 0.05471 0.03740 0.02560
1.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000

例題 11-4　二維暫態熱傳導問題

考慮一無限長的金屬棒，每邊長度均爲 2L，其熱滲透係數 α 爲定值。此金屬棒原始溫度爲 $T_0 = 400°C$，在時間 $t = 0$ 時，突然將其表面溫度降爲 $T_1 = 100°C$。試求此金屬棒的內部溫度變化 $T(x, y, t)$。

解： 首先引進無因次變數

$$X = \frac{x}{L} \qquad Y = \frac{y}{L}$$

$$\tau = \frac{\alpha t}{L^2} \qquad \theta = \frac{T - T_0}{T_1 - T_0}$$

由於金屬棒成對稱形狀，解析其溫度分布時，只需考慮中心對稱的四分之一部分即可。

因此，可將此暫態熱傳導問題的微分方程式簡化成

$$\frac{\partial \theta}{\partial \tau} = \frac{\partial^2 \theta}{\partial X^2} + \frac{\partial^2 \theta}{\partial Y^2} \qquad \textbf{(11-6.24)}$$

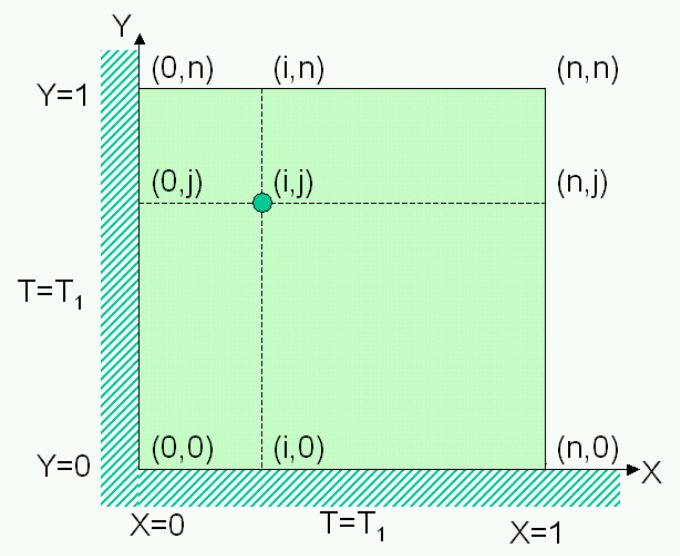


圖 11.11　二維暫態熱傳導

$\theta(X, Y, 0) = 0$ 在 $\tau = 0$ 時

$\theta(1, Y, \tau) = 1$ 在 $X = 1$ 處

$\theta(x, 1, \tau) = 1$ 在 $Y = 1$ 處

$\frac{\partial \theta}{\partial X} = 0$ 在 $X = 0$ 處

$\frac{\partial \theta}{\partial Y} = 0$ 在 $Y = 0$ 處

仿照方程式 (11-6.8) 的形式，將偏微分方程 (11-6.24) 改寫成道格拉斯交替差分方程式，可以得到

$$\begin{bmatrix} 2(\frac{1}{\lambda}+1) & -2 & 0 & \cdots & 0 & 0 \\ -1 & 2(\frac{1}{\lambda}+1) & -1 & \cdots & 0 & 0 \\ 0 & -1 & 2(\frac{1}{\lambda}+1) & \cdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \cdots & -1 & 0 \\ 0 & 0 & \cdots & -1 & 2(\frac{1}{\lambda}+1) & -1 \\ 0 & 0 & \cdots & 0 & -1 & 2(\frac{1}{\lambda}+1) \end{bmatrix} \begin{bmatrix} \theta^*_{0,j} \\ \theta^*_{1,j} \\ \vdots \\ \vdots \\ \theta^*_{n-2,j} \\ \theta^*_{n-1,j} \end{bmatrix} = \begin{bmatrix} A_{0,j} \\ A_{1,j} \\ \vdots \\ \vdots \\ A_{n-2,j} \\ A_{n-1,j} \end{bmatrix}$$

(11-6.25)

$$\begin{bmatrix} 2(\frac{1}{\lambda}+1) & -2 & 0 & \cdots & 0 & 0 \\ -1 & 2(\frac{1}{\lambda}+1) & -1 & \cdots & 0 & 0 \\ 0 & -1 & 2(\frac{1}{\lambda}+1) & \cdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \cdots & -1 & 0 \\ 0 & 0 & \cdots & -1 & 2(\frac{1}{\lambda}+1) & -1 \\ 0 & 0 & \cdots & 0 & -1 & 2(\frac{1}{\lambda}+1) \end{bmatrix} \begin{bmatrix} \theta_{i,0} \\ \theta_{i,1} \\ \vdots \\ \vdots \\ \theta_{i,n-2} \\ \theta_{i,n-1} \end{bmatrix} = \begin{bmatrix} B_{i,0} \\ B_{i,1} \\ \vdots \\ \vdots \\ B_{i,n-2} \\ B_{i,n-1} \end{bmatrix}$$

(11-6.26)

其中

$$\left.\begin{aligned} A_{i,0} &= 2\theta_{i,1} + 2(\frac{1}{\lambda}-1)\theta_{i,0} && i = 0,1,....,n-2 \\ A_{n-1,0} &= 2\theta_{n-1,1} + 2(\frac{1}{\lambda}-1)\theta_{n-1,0} + \theta_{n,0} \end{aligned}\right\} j = 0$$

$$\left.\begin{aligned} A_{i,\,j} &= \theta_{i,\,j-1} + 2(\frac{1}{\lambda}-1)\theta_{i,\,j} + \theta_{i,\,j+1} && i = 0\,,1\,,....\,,n-2 \\ A_{n-1,\,j} &= \theta_{n-1,\,j-1} + 2(\frac{1}{\lambda}-1)\theta_{n-1,\,j} + \theta_{n-1,\,j+1} + \theta_{n,\,j} \end{aligned}\right\} j \neq 0$$

$$\left.\begin{aligned} B_{o,\,j} &= 2\theta^{*}_{1,\,j} + 2(\frac{1}{\lambda}-1)\theta^{*}_{0,\,j} && j = 0\,,1\,,....\,,n-2 \\ B_{0,\,n-1} &= 2\theta^{*}_{1,\,n-1} + 2(\frac{1}{\lambda}-1)\theta^{*}_{0,\,n-1} + \theta^{*}_{0,\,n} \end{aligned}\right\} i = 0$$

$$\left.\begin{aligned} B_{i,\,j} &= \theta^{*}_{i-1,\,j} + 2(\frac{1}{\lambda}-1)\theta^{*}_{i,\,j} + \theta^{*}_{i+1,\,j} && j = 0\,,1\,,....\,,n-2 \\ B_{i,\,n-1} &= \theta^{*}_{i-1,\,n-1} + 2(\frac{1}{\lambda}-1)\theta^{*}_{i,\,n-1} + \theta^{*}_{i+1,\,n-1} + \theta^{*}_{i,\,n} \end{aligned}\right\} i \neq 0$$

求解時，將前一步驟所得結果 $\theta_{i,\,j}$ 利用以上的道格拉斯交替差分方程式 (11-6.25)，先計算中間步驟$t = t_{n+1/2}$時的中間函數值 $\theta^{*}_{i,\,j}$；再利用差分方程式 (11-6.26)，求出下一時間$t = t_{n+1}$時的函數值 $\theta_{i,\,j}$。

程式列印：

```
'
'*********************************************************************
' SOLUTION OF Parabolic 2D-P.D.E.
' BY    Douglas Implicit Alternating Direction Method
'*********************************************************************
'
' Program Developed by Dr. Ron Hsin Chang
' Copyright 1988, 2001
'
' Example 11-4
'
Sub Douglas (Xpos, Ypos)
Dim T (50, 50) , Tstar (50, 50) As Double
'
'     Enter Basic Data & DEFINE THE PROBLEM
'
      Cls
      Print
      Print " Solution of Parabolic PDE by Douglas IAD Method"
      Print
```

```
        N = 20
        Print "Number of X-Intervals"
        N = Val (InputBox ("N = ", "No. of X-Intervals", N) )
        Print "N = "; N
        NP1 = N + 1

        DX = 1 / N
        Print "DX = "; DX

        Dtau = 0.05
        Dtau = Val (InputBox ("Time Step = ", "Dtau", Dtau) )
        Print "Time Step =", Dtau
        Print

        Tmax = 0.95
        Tmax = Val (InputBox ("Maximun Center Temperature = ", "Tmax", Tmax) )
        Print "Maximum Temp at center =", Tmax
        Print

        NPT = 1
        NPT = Val (InputBox ("Number of Time Step before printing = ", "NPT", NPT) )
        Print "No. Time Step Before Printing =", NPT
        Print
'
'   Set Initial and Boundary Conditions
'
        For I = 1 To N
            T (I, NP1) = 1#
            T (NP1, I) = 1#
            Tstar (I, NP1) = 1#
            Tstar (NP1, I) = 1#
            For J = 1 To N
                T (I, J) = 0#
                Tstar (I, J) = 0#
            Next J
        Next I
        T (NP1, NP1) = 1#
        Tstar (NP1, NP1) = 1#

        Call IADSolver ( N, T, Tstar, Dtau, DX, Tmax, NPT)

End Sub


Sub IADSolver ( N, U, Ustar, Dtau, DX, Umax, NPT)
'
'   Finite Difference Parameters
'   N               = NUMBER OF X-Intervals
'   U               = INITIAL CONDITION VECTOR
'                     /RETURN : SOLUTION
'   Ustar           = Half Step Solution
'                     /RETURN : SOLUTION
'   Umax            = MAXIMUM Value of U (1,1)
'   NPT             = No. of Time Steps before Printing
'
```

```
'     A,B,C             = TRIDIAGONAL MATRIX COEFFICIENT VECTOR
'
Dim A (50) , B (50) , C (50) , Z (50) , Uprime (50) As Double
'
'     Set up Parameters
'
      NP1 = N + 1
      Lamda = Dtau / DX / DX
      F = 2 * (1 / Lamda - 1)
      A (1) = 2 * (1 / Lamda + 1)
      C (1) = -2
      For I = 2 To N
            C (I) = -1
            A (I) = A (1)
            B (I) = -1
      Next I
'
'     ------------------------
'       SOLVE & PRINT RESULT
'     ------------------------
'
      Icount = 0
      Tau = 0

'
'     Perform Calculations Over Successive Time Step
'
      Do
            Tau = Tau + Dtau
            Icount = Icount + 1
'
'           Calculate U at End of Half Time Increment
'
            For J = 1 To N
                  For I = 1 To N
                        If (J = 1) Then
                              Z (I) = 2 * U (I, 2) + F * U (I, 1)
                        Else
                              Z (I) = U (I, J - 1) + F * U (I, J) + U (I, J + 1)
                        End If
                        Z (N) = Z (N) + U (NP1, J)
                  Next I
                  Call Tridiagonal (1, N, A, B, C, Z, Uprime)
                  For I = 1 To N
                        Ustar (I, J) = Uprime (I)
                  Next I
            Next J
'
'           Calculate U at End of Full Time Increment
'
            For I = 1 To N
                  For J = 1 To N
                        If (I = 1) Then
                              Z (J) = 2 * Ustar (2, J) + F * Ustar (1, J)
                        Else
```

```
                        Z (J) = Ustar (I - 1, J) + F * Ustar (I, J) + Ustar (I + 1, J)
                    End If
                    Z (N) = Z (N) + Ustar (I, NP1)
                Next J
                Call Tridiagonal (1, N, A, B, C, Z, Uprime)
                For J = 1 To N
                    U (I, J) = Uprime (J)
                Next J
            Next I
'
'           Print U Throughout the Domain
'
            If (Icount = NPT) Then
                Icount = 0
                Debug.Print "At Time Tau=";
                Debug.Print Format (Tau, "    0.0000E+00")
                Debug.Print "Temperature are:"
                For I = 1 To NP1
                    For J = 1 To NP1
                        Debug.Print Format (U (I, J) , " 0.0000    ") ;
                    Next J
                    Debug.Print
                Next I
            End If
        Loop While U (1, 1) < Umax
End Sub


Sub Tridiagonal (Nstart, N, A, B, C, Z, V)
'
'     ----------------------
'        SUBROUTINE JACOBI
'     ----------------------
'
' N               = No. of Equations
' A, B, C      = Tridiagonal Coefficients of Equations
' Alpha          = Intermediate Coefficients
' Gamma          = Intermediate Coefficients
' Z               = Constant Vector
' V               = Solution Vector
'
Dim Alpha (50) , Gamma (50) As Double

Alpha (Nstart) = A (Nstart)
Gamma (Nstart) = Z (Nstart) / Alpha (Nstart)
N1 = Nstart + 1
'
'   -- LU DECOMPOSITION
'
For I = N1 To N
    Alpha (I) = A (I) - B (I) * C (I - 1) / Alpha (I - 1)
    Gamma (I) = (Z (I) - B (I) * Gamma (I - 1) ) / Alpha (I)
Next I
'
'   -- BACK SUBSTITUTION
```

```
'
V (N) = Gamma (N)
Last = N - Nstart
For K = 1 To Last
        I = N - K
        V (I) = Gamma (I) - C (I) * V (I + 1) / Alpha (I)
Next K
'
'-- CHEER by Ron Hsin Chang, Copyright 2001
'
End Sub
```

副程式使用說明：

副程式 **Sub IADSolver (N, U, Ustar, Dtau, DX, Umax, NPT)**

N　　= 格子點區間數

U　　= 呼叫時輸入起始條件，回覆計算結果

Ustar = 呼叫時輸入起始條件，回覆中間計算結果

Dtau　= 時間增量

DX　　= 空間增量

Umax = 在中央點的最大 U 值

NPT　= 計算結果輸出所間隔的數目

輸出結果：

At Time Tau=　5.0000E-02
Temperature are:

0.0158	0.0174	0.0227	0.0340	0.0557	0.0964	0.1726	0.3147	0.5794	1.0728	1.0000
0.0174	0.0189	0.0243	0.0355	0.0572	0.0979	0.1739	0.3158	0.5801	1.0727	1.0000
0.0227	0.0243	0.0296	0.0408	0.0623	0.1028	0.1784	0.3195	0.5824	1.0723	1.0000
0.0340	0.0355	0.0408	0.0518	0.0731	0.1131	0.1879	0.3273	0.5872	1.0714	1.0000
0.0557	0.0572	0.0623	0.0731	0.0939	0.1330	0.2061	0.3424	0.5965	1.0698	1.0000
0.0964	0.0979	0.1028	0.1131	0.1330	0.1705	0.2404	0.3708	0.6139	1.0668	1.0000
0.1726	0.1739	0.1784	0.1879	0.2061	0.2404	0.3045	0.4239	0.6464	1.0612	1.0000
0.3147	0.3158	0.3195	0.3273	0.3424	0.3708	0.4239	0.5228	0.7071	1.0507	1.0000
0.5794	0.5801	0.5824	0.5872	0.5965	0.6139	0.6464	0.7071	0.8203	1.0311	1.0000
1.0728	1.0727	1.0723	1.0714	1.0698	1.0668	1.0612	1.0507	1.0311	0.9946	1.0000
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

At Time Tau=　1.0000E-01
Temperature are:

0.0933	0.0994	0.1189	0.1555	0.2151	0.3050	0.4297	0.5808	0.7096	0.6642	1.0000
0.0994	0.1055	0.1248	0.1611	0.2203	0.3096	0.4336	0.5836	0.7115	0.6665	1.0000
0.1189	0.1248	0.1437	0.1793	0.2372	0.3246	0.4458	0.5926	0.7178	0.6737	1.0000

0.1555	0.1611	0.1793	0.2133	0.2689	0.3526	0.4688	0.6095	0.7295	0.6872	1.0000
0.2151	0.2203	0.2372	0.2689	0.3205	0.3983	0.5063	0.6371	0.7486	0.7093	1.0000
0.3050	0.3096	0.3246	0.3526	0.3983	0.4672	0.5628	0.6787	0.7774	0.7426	1.0000
0.4297	0.4336	0.4458	0.4688	0.5063	0.5628	0.6413	0.7363	0.8173	0.7888	1.0000
0.5808	0.5836	0.5926	0.6095	0.6371	0.6787	0.7363	0.8062	0.8657	0.8448	1.0000
0.7096	0.7115	0.7178	0.7295	0.7486	0.7774	0.8173	0.8657	0.9070	0.8924	1.0000
0.6642	0.6665	0.6737	0.6872	0.7093	0.7426	0.7888	0.8448	0.8924	0.8757	1.0000
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

At Time Tau= 3.0000E-01
Temperature are:

0.6322	0.6368	0.6505	0.6728	0.7026	0.7387	0.7812	0.8343	0.9017	0.9187	1.0000
0.6368	0.6414	0.6549	0.6769	0.7063	0.7420	0.7839	0.8364	0.9030	0.9197	1.0000
0.6505	0.6549	0.6679	0.6890	0.7174	0.7517	0.7920	0.8425	0.9066	0.9227	1.0000
0.6728	0.6769	0.6890	0.7088	0.7353	0.7675	0.8053	0.8525	0.9126	0.9276	1.0000
0.7026	0.7063	0.7174	0.7353	0.7595	0.7887	0.8230	0.8660	0.9205	0.9342	1.0000
0.7387	0.7420	0.7517	0.7675	0.7887	0.8143	0.8445	0.8823	0.9302	0.9422	1.0000
0.7812	0.7839	0.7920	0.8053	0.8230	0.8445	0.8698	0.9014	0.9415	0.9516	1.0000
0.8343	0.8364	0.8425	0.8525	0.8660	0.8823	0.9014	0.9253	0.9557	0.9634	1.0000
0.9017	0.9030	0.9066	0.9126	0.9205	0.9302	0.9415	0.9557	0.9737	0.9783	1.0000
0.9187	0.9197	0.9227	0.9276	0.9342	0.9422	0.9516	0.9634	0.9783	0.9820	1.0000
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

At Time Tau= 5.0000E-01
Temperature are:

0.8628	0.8645	0.8695	0.8778	0.8891	0.9031	0.9189	0.9369	0.9614	0.9744	1.0000
0.8645	0.8662	0.8711	0.8793	0.8905	0.9043	0.9199	0.9376	0.9618	0.9747	1.0000
0.8695	0.8711	0.8759	0.8837	0.8945	0.9078	0.9229	0.9399	0.9632	0.9756	1.0000
0.8778	0.8793	0.8837	0.8911	0.9012	0.9137	0.9278	0.9437	0.9656	0.9772	1.0000
0.8891	0.8905	0.8945	0.9012	0.9104	0.9217	0.9345	0.9490	0.9688	0.9793	1.0000
0.9031	0.9043	0.9078	0.9137	0.9217	0.9316	0.9427	0.9554	0.9727	0.9819	1.0000
0.9189	0.9199	0.9229	0.9278	0.9345	0.9427	0.9521	0.9627	0.9772	0.9849	1.0000
0.9369	0.9376	0.9399	0.9437	0.9490	0.9554	0.9627	0.9709	0.9822	0.9882	1.0000
0.9614	0.9618	0.9632	0.9656	0.9688	0.9727	0.9772	0.9822	0.9891	0.9928	1.0000
0.9744	0.9747	0.9756	0.9772	0.9793	0.9819	0.9849	0.9882	0.9928	0.9952	1.0000
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

At Time Tau= 7.5000E-01
Temperature are:

0.9600	0.9605	0.9620	0.9644	0.9676	0.9717	0.9765	0.9821	0.9870	0.9943	1.0000
0.9605	0.9610	0.9624	0.9648	0.9680	0.9720	0.9768	0.9823	0.9872	0.9944	1.0000
0.9620	0.9624	0.9638	0.9661	0.9692	0.9731	0.9776	0.9830	0.9877	0.9946	1.0000
0.9644	0.9648	0.9661	0.9683	0.9712	0.9748	0.9791	0.9841	0.9885	0.9949	1.0000
0.9676	0.9680	0.9692	0.9712	0.9738	0.9771	0.9810	0.9855	0.9895	0.9954	1.0000
0.9717	0.9720	0.9731	0.9748	0.9771	0.9800	0.9834	0.9873	0.9908	0.9960	1.0000
0.9765	0.9768	0.9776	0.9791	0.9810	0.9834	0.9862	0.9895	0.9924	0.9966	1.0000
0.9821	0.9823	0.9830	0.9841	0.9855	0.9873	0.9895	0.9920	0.9942	0.9974	1.0000
0.9870	0.9872	0.9877	0.9885	0.9895	0.9908	0.9924	0.9942	0.9958	0.9982	1.0000
0.9943	0.9944	0.9946	0.9949	0.9954	0.9960	0.9966	0.9974	0.9982	0.9992	1.0000
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

第七節　邊界條件與差分方程式

Visual Basic

考慮拋物線偏微分方程式，可能的邊界條件可區分爲以下三大類。

1. 第一類邊界條件：

應變數u在邊界上爲定值。例如，在物體表面溫度等於100°C ，或在流體表面濃度等於C_0。其一般表示式爲$u = g$。

2. 第二類邊界條件：

應變數u在邊界上的導函數爲定值，其中u_n表示垂直邊界方向的導函數，u_s表示切線方向的導函數。例如，在傳熱面上有一定的熱通量將熱量傳遞給傳熱面；或管壁作斷熱或絕熱處理，使其熱通量爲零。其一般表示式爲 $\alpha u_n + \beta u_s = g$ 。

3. 第三類邊界條件：

應變數u及其在邊界上的導函數的和爲定值，其中u_n表示垂直邊界方向的導函數，u_s表示切線方向的導函數。例如，在流體介面的質傳滲透率等於表面對流所帶走的量；或傳熱面的熱傳導通量等於對流熱傳所帶走的熱量。其一般表示式爲 $\alpha u_n + \beta u_s + \gamma u = g$ 。

考慮二維拋物線偏微分方程式 $\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$ ，若使用差分法解偏微分方程式時，邊界條件爲第一類邊界條件，直接可以宣告函數值，是最簡單的情況。若邊界條件是第二類或第三類邊界條件時，由於邊界條件含有導函數，必須作適當處理。假設偏微分方程式的邊界條件爲

$$-\frac{du}{dx} + au = g \quad ， \quad 在\ x = 0\ 處 \tag{11-7.1}$$

針對鄰近於邊界的點$u_{1,j}$，可以利用泰勒級數展開，得到

$$u_{1,j} = u_{0,j} + u_x \Delta x + \frac{(\Delta x)^2}{2!} u_{xx} + \frac{(\Delta x)^3}{3!} + \mathrm{O}\left[(\Delta x)^3\right] \tag{11-7.2}$$

$$u_{1,j} = u_{2,j} - u_x \Delta x + \frac{(\Delta x)^2}{2!} u_{xx} - \frac{(\Delta x)^3}{3!} + \mathrm{O}\left[(\Delta x)^3\right] \tag{11-7.3}$$

由方程式 (11-7.2) 可以得到u_{xx}的表示式爲

$$u_{xx}=\frac{2}{(\Delta x)^2}\left[u_{1,j}-u_{0,j}-u_x\Delta x\right]+\mathrm{O}\left[(\Delta x)\right] \tag{11-7.4}$$

又將方程式 (11-7.2) 及方程式 (11-7.3) 相加及相減，分別可以得到u_{xx}及u_x的表示式爲

$$u_{xx}=\frac{2}{(\Delta x)^2}\left[-u_{0,j}+2u_{i,j}-u_{2,j}\right]+\mathrm{O}\left[(\Delta x)^2\right] \tag{11-7.5}$$

$$u_x=\frac{1}{2\Delta x}\left[u_{2,j}-u_{0,j}\right]+\mathrm{O}\left[(\Delta x)^2\right] \tag{11-7.6}$$

由於邊界條件爲$-\frac{du}{dx}+au=g$，利用方程式 (11-7.6) 寫成差分式，可以得到

$$\frac{1}{2\Delta x}\left[u_{2,j}-u_{0,j}\right]-au_{0,j}=g \tag{11-7.7}$$

或整理得到

$$u_{2,j}=u_{0,j}+2a\Delta x\,u_{0,j}+2\Delta x\,g \tag{11-7.8}$$

將方程式 (11-7.8) 代入方程式 (11-7.5) 中，將$u_{2,j}$消去，可以得到u_{xx}的表示式爲

$$u_{xx}=\frac{1}{(\Delta x)^2}\left[2(a\Delta x-1)u_{0,j}+2u_{i,j}+2\Delta xg\right] \tag{11-7.9}$$

因此，二維拋物線偏微分方程式$\frac{\partial u}{\partial t}=\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}$改寫成差分方程式時，在邊界上的條件可以寫成

$$\begin{aligned}\frac{u_{0,j,n+1}-u_{0,j,n}}{\Delta t}&=\delta_x^2u_{0,j,n+1}+\delta_y^2u_{0,j,n+1}\\&=\frac{\left[2(a\Delta x-1)u_{0,j}+2u_{i,j}+2\Delta xg\right]}{(\Delta x)^2}+\delta_y^2u_{0,j,n+1}+\mathrm{O}\left[(\Delta x)^2+(\Delta y)^2\right]\end{aligned} \tag{11-7.10}$$

非規則邊界的處理

以上所介紹的邊界條件處理原則，基本上都是考慮邊界正好落在格子點上的情況。但如遇到邊界並非正好在格子點上，或邊界呈非規則形狀時，如圖 11.12，可以針

對鄰近格子點作泰勒級數展開，加以處理。

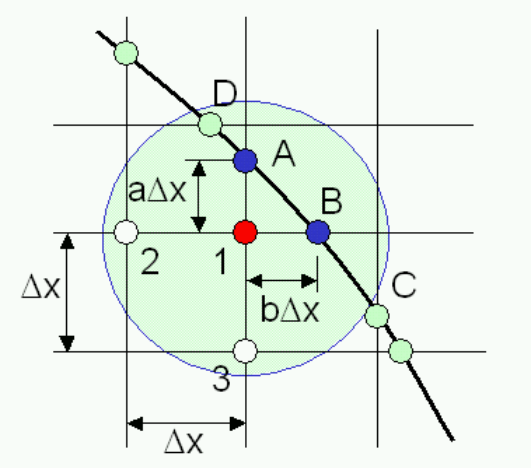


圖 11.12　非規則邊界之處理

以圖 11.12 爲例，最接近點 1 的邊界位置爲 A 及 B，鄰近的格子點爲 2 及 3。分別作泰勒級數展開，可以得到

$$u_A = u_1 + a\Delta x u_y + \frac{(a\Delta x)^2}{2!}u_{yy} + \mathrm{O}\left[(\Delta x)^3\right] \qquad \textbf{(11-7.11)}$$

$$u_B = u_1 + b\Delta x u_x + \frac{(b\Delta x)^2}{2!}u_{xx} + \mathrm{O}\left[(\Delta x)^3\right] \qquad \textbf{(11-7.12)}$$

$$u_2 = u_1 - \Delta x u_x + \frac{(\Delta x)^2}{2!}u_{xx} + \mathrm{O}\left[(\Delta x)^3\right] \qquad \textbf{(11-7.13)}$$

$$u_3 = u_1 - \Delta x u_y + \frac{(\Delta x)^2}{2!}u_{yy} + \mathrm{O}\left[(\Delta x)^3\right] \qquad \textbf{(11-7.14)}$$

將方程式 (11-7.11) 及方程式 (11-7.12) 相加，再加上方程式 (11-7.13) 乘以 a 及方程式 (11-7.14) 乘以 b，可以得到在編號 1 的點上之二階導函數爲

$$u_{xx} + u_{yy} = \frac{2}{(\Delta x)^2}\left[\frac{u_A}{a(a+1)} + \frac{u_B}{b(b+1)} + \frac{u_2}{b+1} + \frac{u_3}{a+1} - \frac{(a+b)u_1}{ab}\right] + \mathrm{O}\left[\Delta x\right] \qquad \textbf{(11-7.15)}$$

第八節　特徵值問題

Visual Basic

偏微分方程式常利用分離變數法 (Separation of Variables) 來求解，是工程應用上最常使用方法之一。利用分離變數法解偏微分方程式時，會將原微分方程式解離成二組或更多組含有任意參數λ的常微分方程式。再利用邊界條件決定這些未定參數λ，以建立有意義的解答。

史特姆 - 陸亦威爾 (Sturm-Liouville) 方程式即是利用分離變數法所得到常見的常微分方程式，其一般表示式爲

$$\frac{d}{dx}\left[p(x)\frac{dy}{dx}\right]+\left[q(x)+\lambda r(x)\right]y=0 \qquad a<x<b \qquad \textbf{(11-8.1)}$$

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

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

其中函數 $p(x)$、$q(x)$ 及 $r(x)$ 爲已知的函數，且在所考慮的區間 (a , b) 間爲連續函數，且 $r(x) > 0$。此方程式只有在特定的 λ 值才有解；此特定 λ 值即稱爲方程式的特徵值；對應的解答稱爲特徵函數。

爲了簡化說明起見，首先假設$p(x) =0$，且$c_1 = c_2 =0$，$a = 0$，$b = 1$。則簡化後的方程式變成

$$\frac{d^2y}{dx^2}+\left[q(x)+\lambda r(x)\right]y=0 \qquad 0<x<1 \qquad \textbf{(11-8.2)}$$

$$BC1 \quad y=0 \quad @x=0$$

$$BC2 \quad y=0 \quad @x=1$$

利用有限差分法解方程式 (11-8.2) 時，首先將 x 區間分割成 n 個小區間，並令以下變數

$$x_k=hk \quad , \quad h=1/(n+1)$$

$$y(x_k)=u_k \quad , \quad u_0=u_{n+1}=0$$

然後將方程式 (11-8.2) 改寫成差分方程式

$$\frac{u_{k-1}-2u_k+u_{k+1}}{h^2}+\left[q_k+\lambda r_k\right]u_k=0 \qquad \textbf{(11-8.3)}$$

或寫成矩陣形式爲

$$\underline{\underline{Q}}\,\underline{u}=\lambda h^2\,\underline{\underline{R}}\,\underline{u} \qquad \textbf{(11-8.4)}$$

$$\begin{bmatrix}(2-q_1h^2) & -1 & 0 & \cdots & 0 & 0\\ -1 & (2-q_2h^2) & -1 & \cdots & 0 & 0\\ 0 & -1 & (2-q_3h^2) & \cdots & \vdots & \vdots\\ \vdots & \vdots & \vdots & \vdots & -1 & 0\\ 0 & 0 & 0 & \cdots & (2-q_{n-1}h^2) & -1\\ 0 & 0 & 0 & \cdots & -1 & (2-q_nh^2)\end{bmatrix}\begin{bmatrix}u_1\\u_2\\u_3\\\vdots\\\vdots\\u_n\end{bmatrix}$$

$$=\begin{bmatrix}\lambda h^2r_1 & 0 & 0 & \cdots & 0 & 0\\ 0 & \lambda h^2r_2 & 0 & \cdots & 0 & 0\\ 0 & 0 & \lambda h^2r_3 & \cdots & \vdots & \vdots\\ \vdots & \vdots & \vdots & \vdots & 0 & 0\\ 0 & 0 & 0 & \cdots & \lambda h^2r_{n-1} & 0\\ 0 & 0 & 0 & \cdots & 0 & \lambda h^2r_n\end{bmatrix}\begin{bmatrix}u_1\\u_2\\u_3\\\vdots\\\vdots\\u_n\end{bmatrix} \qquad \textbf{(11-8.5)}$$

令 $\underline{u}=\underline{\underline{R}}^{-1/2}\underline{V}$ ，且 $\underline{\underline{A}}=h^{-2}\underline{\underline{R}}^{-1/2}\underline{\underline{Q}}\underline{\underline{R}}^{-1/2}$ ，則可將方程式 (11-8.4) 所表示的矩陣方程式，改寫成

$$\underline{\underline{A}}\,\underline{V}=\lambda\underline{V} \qquad \textbf{(11-8.6)}$$

由本書第四章的說明可知，方程式 (11-8.6) 的解答爲矩陣 $\underline{A}$ 的代數特徵值問題，其數值解法詳見本書第四章。方程式 (11-8.6) 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。其實際應用，以下例加以說明之。

例題 11-5 內管絕熱之金屬管暫態熱傳問題

考慮如圖 11.13 的金屬管，其內管利用絕熱材料絕熱。在時間t = 0 時，其原始溫度$\theta=0$，當t > 0 以後，金屬管表面溫度維持在$\theta=1$。金屬管之內徑爲r_1，外徑爲r_2。試求其暫態溫度分布。

解：金屬管的徑向熱傳導數學模式爲

$$\frac{\partial \theta}{\partial t}=\frac{\partial^2 \theta}{\partial r^2}+\frac{1}{r}\frac{\partial \theta}{\partial r}$$

BC1　$r=r_1$ 時，　$\dfrac{d\theta}{dr}=0$

BC2　$r=r_2$ 時，　$\theta=1$

IC　$t=0$ ，　$\theta=0$

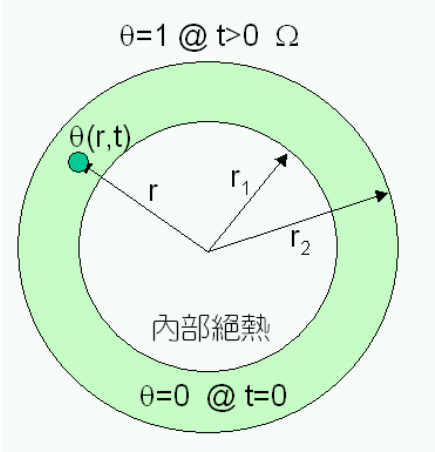


圖 11.13　內管絕熱之金屬管暫態熱傳問題

由於使用特徵值問題求解時，邊界條件必須是均勻條件，才能獲得均勻解；然後利用加成原理，配合非均勻的起始條件，求未定係數。由於以上的數學模式中，BC2並非均勻條件，因此，定義輔助變數 $\phi=1-\theta$，代回原爲分方程式，可將微分方程式改寫成滿足分離變數法的形式

$$\frac{\partial \phi}{\partial t}=\frac{\partial^2 \phi}{\partial r^2}+\frac{1}{r}\frac{\partial \phi}{\partial r} \qquad \textbf{(11-8.7)}$$

BC1　$r=r_1$ 時，　$\dfrac{d\phi}{dr}=0$

BC2　$r=r_2$ 時，　$\phi=0$

IC　$t=0$ ，　$\phi=1$

方程式 (11-8.7) 可以利用分離變數法 (Separation of Variables) 求解。首先假設函數 $\phi(r,t)$ 可以分離成徑向函數 $R(r)$ 及時間函數 $T(t)$ 的乘積，用方程式形式表示爲

$$\phi(r,t)=R(r)T(t) \qquad \textbf{(11-8.8)}$$

代入方程式 (11-8.7) 中，經整理後，可以得到

$$\frac{1}{R}\frac{d^2R}{dr^2}+\frac{1}{rR}\frac{dR}{dr}=\frac{1}{T}\frac{dT}{dt}=-\lambda \qquad \textbf{(11-8.9)}$$

方程式 (11-8.9) 中，時間函數的解爲 $T(t)=c_i\exp(-\lambda_i t)$。假設存在 N 個不同的λ數值，則方程式 (11-8.7) 的解答爲

$$\phi(r,t)=\sum_{i=1}^{N}c_iR_i(r)\exp(-\lambda_i t) \qquad \textbf{(11-8.10)}$$

其中未定係數c_i、特徵值λ_i及均勻解$R_i(r)$，均等待決定。

首先考慮方程式 (11-8.9) 中徑向函數 $R(r)$ 的部分

$$\frac{d^2R}{dr^2}+\frac{1}{r}\frac{dR}{dr}+\lambda R=0 \qquad \textbf{(11-8.11)}$$

利用有限差分法解方程式 (11-8.11) 時，假設將$r_1 < r < r_2$區間分割成N個等間距Δr的小區間，再$r=r_1$位置的格子點定義爲 1 號格子，$r=r_2$的格子點定義爲第N+1 個格子。則可將方程式 (11-8.11) 改寫成以下的差分方程式：

$$\frac{R_{i-1}-2R_i+R_{i+1}}{(\Delta r)^2}+\frac{1}{r_i}\frac{R_{i+1}-R_{i-1}}{2\Delta r}+\lambda R_i=0 \qquad i=2,3,...,N \qquad \textbf{(11-8.12)}$$

定義

$$D=\frac{2}{(\Delta r)^2} \qquad E_i=\frac{D}{2}(1-\frac{\Delta r}{r_i}) \qquad F_i=\frac{D}{2}(1+\frac{\Delta r}{2r_i})$$

則方程式 (11-8.12) 可以簡化成

$$-E_iR_{i-1}+(D-\lambda)R_i-F_iR_{i+1}=0 \qquad i=2,3,....,N \qquad \textbf{(11-8.13)}$$

邊界條件的處理：

1. 當 $i=1$ 時，$dR/dr=0$。且利用差分法可以得到

$$\frac{d^2R}{dr^2}=\frac{2}{(\Delta r)^2}(R_2-R_1) \qquad \frac{1}{r_1}\frac{dR}{dr}=\frac{1}{r_1}\frac{(R_2-R_1)}{\Delta r}$$

代至方程式 (11-8.11) 中，經整理後得到

$$(D-\lambda)R_1 - DR_2 = 0 \qquad \textbf{(11-8.14)}$$

2. 當i = N+1 時，R_{N+1}=0。

矩陣方程式：

將以上邊界條件及方程式 (11-8.13) 加以整理，可以得到以下的矩陣方程式

$$\begin{bmatrix} D & -D & 0 & 0 & \cdots & 0 & 0 \\ -E_2 & D & -F_2 & 0 & \cdots & 0 & 0 \\ 0 & -E_3 & D & -F_3 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & D & -F_{N-2} & 0 \\ 0 & 0 & 0 & \cdots & -E_{N-1} & D & -F_{N-1} \\ 0 & 0 & 0 & \cdots & 0 & -E_N & D \end{bmatrix} \begin{bmatrix} R_1 \\ R_2 \\ R_3 \\ \vdots \\ \vdots \\ R_{N-1} \\ R_N \end{bmatrix} = \lambda \begin{bmatrix} R_1 \\ R_2 \\ R_3 \\ \vdots \\ \vdots \\ R_{N-1} \\ R_N \end{bmatrix}$$

(11-8.15)

或寫成與方程式 (11-8.6) 相同形式的矩陣方程式

$$\underline{\underline{A}}\,\underline{r} = \lambda \underline{r} \qquad \textbf{(11-8.16)}$$

一般而言，方程式 (11-8.15) 會有N個不同的特徵值 λ_j，同時可以得到N組特徵陣列 (Eigenvectors) $\underline{r_j}$構成特徵矩陣$\underline{\underline{R}}$。

將特徵值及特徵陣列代回方程式 (11-8.10)，可以整理得到微分方程式的解爲

$$\phi_i(t) = \sum_{j=1}^{N} c_j r_{ij} \exp(-\lambda_j t) \qquad \textbf{(11-8.17)}$$

其中r_{ij}爲第j個特徵陣列的第i個元素。由於 $\underline{\underline{A}} = \underline{\underline{R}}\underline{\underline{\Lambda}}\underline{\underline{R}}^{-1}$，且令 $\underline{\underline{S}} = \underline{\underline{R}}^{-1}$；則可以利用微分方程式的起始值，求得未定係數$c_j$。

首先將方程式 (11-8.17) 兩側各乘以 $\underline{\underline{S}} = \underline{\underline{R}}^{-1}$，則得到

$$\sum_{i=1}^{N} s_{ik}\phi_i(0) = \sum_{i=1}^{N} \sum_{j=1}^{N} c_j s_{ik} r_{ij} \exp(-\lambda_j t) \qquad \textbf{(11-8.18)}$$

由於 $\underline{\underline{S}} = \underline{\underline{R}}^{-1}$，

$$\sum_{i=1}^{N} s_{ik} r_{ij} = 0 \qquad \text{若 } j \neq k$$

$$= \sum_{i=1}^{N} s_{ij} r_{ij} \qquad \text{若 } j = k$$

因此， 可以得到未定係數的表示式爲

$$c_j = \left[\sum_{i=1}^{N} s_{ij} \right] \Big/ \left[\sum_{i=1}^{N} s_{ij} r_{ij} \right] \qquad \textbf{(11-8.19)}$$

計算策略：

1. 先將偏微分方程式利用分離變數法，解離成常微分方程式 (11-8.9) 的形式。
2. 將均勻邊界條件部分的方程式，如方程式 (11-8.11)，改寫成差分方程式。
3. 將邊界條件改寫成差分方程式，並代入第 2 步驟所得到的差分方程式中。
4. 將差分方程式整理成 $\underline{\underline{A}}\underline{r} = \lambda \underline{r}$ 形式。
5. 利用第四章所介紹的特徵值求法，求出矩陣 $\underline{\underline{A}}$ 的特徵值及特徵矩陣 $\underline{\underline{R}}$ 及 $\underline{\underline{S}}$。
6. 利用方程式 (11-8.19) 求未定係數。
7. 利用方程式 (11-8.17) 求出函數值。

程式列印：

```
'
'****************************************"
' SOLUTION OF P.D.E.
' BY SEPARATION OF VARIABLES
'****************************************"
'
' Program Developed by Dr. Ron Hsin Chang
' Copyright 1988, 2001
'
' Example 11-5
'
Sub EigenPDE (Xpos, Ypos)
Dim A (150, 50) , P (50, 50) , Theta (50) As Double
Dim C (50) , NC (50) , AbsErr As Double

'
'     Enter Basic Data & DEFINE THE PROBLEM
'
      Cls
      Print
      Print " Solution of PDE by Separation of Variables"
      Print
```

```
        Call BasicData (Xpos, Ypos, R1, R2, DR, Tmax, DT, ThetaMax, N)
        NM1 = N - 1
        NP1 = N + 1
        D = 2 / DR / DR
        DRm2 = D / 2

        For I = 1 To N
            Theta (I) = 0
            For J = 1 To N
                A (I, J) = 0
            Next J
        Next I
        For I = 1 To NM1
            A (I, I) = D
            A (I, I + 1) = -DRm2 * (1 + 1 / 2 / (R1 / DR + I - 1) )
            A (I + 1, I) = -DRm2 * (1 - 1 / 2 / (R1 / DR + I) )
        Next I
        A (N, N) = D
        A (1, 2) = -D

        Debug.Print "** Starting Matrix is"
        For I = 1 To N
            For J = 1 To N
                Debug.Print Format (A (I, J) , " 0.0000    ") ;
            Next J
            Debug.Print
        Next I
'
'   Find Eigenvalues, Eigenvectors, and Eigenrows
'
        EigenIndex = 1
        AbsErr = 0.00000001
        Call EigenSystem (N, EigenIndex, AbsErr, NC, A)
'
'   Vectors Eigenrows and Eigencolumns
'   and check orthogonality
'
        Cls
        For I = 1 To N
            For J = 1 To N
                P (I, J) = 0
                For K = 1 To N
                    P (I, J) = P (I, J) + A (K + N, I) * A (J + 2 * N, K)
                Next K
            Next J
        Next I
'
'   Output Matrices
'
        Debug.Print "** Tranformed    Matrix is"
        For I = 1 To N
            For J = 1 To N
```

```
                Debug.Print Format (A (I, J) , " 0.000000     ") ;
            Next J
            Debug.Print
        Next I

        Debug.Print "** Eigenrow Matrix is"
        For I = 1 To N
            For J = 1 To N
                Debug.Print Format (A (I + N, J) , " 0.000000     ") ;
            Next J
            Debug.Print
        Next I


        Debug.Print "** Eigencolumn Matrix is"
        For I = 1 To N
            For J = 1 To N
                Debug.Print Format (A (I + 2 * N, J) , " 0.000000     ") ;
            Next J
            Debug.Print
        Next I


        Debug.Print "** Orthogonality requires off-digonal elements to be zero"
        For I = 1 To N
            For J = 1 To N
                Debug.Print Format (P (I, J) , " 0.00000 ") ;
            Next J
            Debug.Print
        Next I
'
'   Determine coefficients C (J) from initial conditions
'
        For J = 1 To N
            C (J) = 0
            For I = 1 To N
                C (J) = C (J) + A (J + 2 * N, I) / P (J, J)
            Next I
        Next J

        Debug.Print "** The Coefficient C (J) are"
        For I = 1 To N
            Debug.Print Format (C (I) , " 0.00000 ") ;
        Next I
        Debug.Print
'
'   Compute Value of Temperature
'
        Debug.Print
        Debug.Print "       Time                    Temp at Grid Points."
        Debug.Print "-----------------------------------------------"

        Theta (NP1) = 1
```

```
        Min = 1
        T = 0
        Do
            Do While (Abs (A (Min, Min) * T) > 40)
                Min = Min + 1
            Loop
            For I = 1 To N
                Theta (I) = 1
                For J = Min To N
                    Theta (I) = Theta (I) - C (J) * A (I + N, J) * Exp (-A (J, J) * T)
                Next J
            Next I

            Debug.Print Format (T, "0.00E+00    ") ;
            For I = 1 To NP1
                Debug.Print Format (Theta (I) , "    0.00000") ;
            Next I
            Debug.Print
            T = T + DT
        Loop While T <= Tmax And Theta (1) <= ThetaMax
End Sub


Sub BasicData (Xpos, Ypos, R1, R2, DR, Tmax, DT, ThetaMax, N)
'     Basic Data for Example 11-5
'
'     DR          = Radial grid spacing
'     DT          = Time Increment
'     N           = No. of grid points
'     R1          = Inner Radius
'     R2          = Outer Radius
'     T           = Time
'     Tmax        = Maximum value of time to be considered
'     ThetaMax= Maximum Temp at Grid #1
'
R1 = 10
R1 = Val (InputBox ("    Inner Radius R1 ", "R1", R1, Xpos, Ypos) )
Print "R1 = "; R1

R2 = 19
R2 = Val (InputBox ("    Outer Radius R2 = ", "R2", R2, Xpos, Ypos) )
Print "R2 = "; R2

Tmax = 100
Tmax = Val (InputBox ("    Max. Time to be considered ", "Tmax", Tmax, Xpos, Ypos) )
Print "Tmax = "; Tmax

DT = 5
DT = Val (InputBox ("    Time increment step ", "DT", DT, Xpos, Ypos) )
Print "DT = "; DT

N = 9
N = Val (InputBox ("    No. of grid intervals ", "N", N, Xpos, Ypos) )
```

```
Print "N = "; N

ThetaMax = 0.95
ThetaMax = Val (InputBox ("    Max. Temp at grid #1 ", "Theta Max", ThetaMax, Xpos, Ypos) )
Print "Theta Max = "; ThetaMax

DR = (R2 - R1) / N

End Sub


Sub EigenSystem (N, EigenIndex, AbsErr, NComplex, A)
'
'     EigenSystem Finds the Eigenvalues, Eigenvectors and Eigenrows
'     of an NxN Square Matrix A
'
'     N is the Size of the Actual Matrix
'
'     EigenIndex is an Indicator
'     EigenIndex    = -1    Gives Eignevalues only.
'                       The results are stored in the first column of A.
'                       Real and imaginary parts of eventual complex eigenvalues
'                       being in adjacent positions.
'     EigenIndex    = 0     Gives Eigenvalues and Eigenvectors.
'                       The Eigenvalues are on the main diagonal of A.
'                       Complex parts as a 2*2 block with the real parts (identical)
'                       on the diagonal, and the imaginary parts (identical, but in the
'                       opposite sign) as adjacent off-diagonal elements.
'                       The matrix of eigenvectors, Q, is stored in row N+1 - 2N
'     EigenIndex    = 1     Gives Eigenvalues, Eigenvectors, and Eigenrows.
'                       Eigenvalues and eigenvectors being stored as for EigenIndex =0.
'                       The eigenrows, Qinv, are stored in row 2N+1 - 3N
'
'     On exit, EigenIndex gives the number of iterations in the QR-step index.
'
'     AbsErr is the tolerance, and a suggested value is 1.E-8
'
'     NComplex is an indicator array for the Eigenvalues.
'     NComplex (I) =0         indicates that the I'th eigenvalue is real.
'     NComplex (I) =1         and NComplex (I+1) =2     indicate a complex pair of eigenvalues, with real
'                       part in A (I+1) and a complex part in A (I+2) for EigenIndex = -1.
'
'     The diagonalized matrix, D, the matrix of eigenvectors, Q, and
'     the matrix of eigenrows, Qinv, have the property, that
'           A = Q * D * Qinv
'
'     Check Error Criteria and EigenIndex
'
If (AbsErr <= 0) Then AbsErr = 0.000000000001

If (EigenIndex < -1) Then
      EigenIndex = -1
ElseIf EigenIndex > 1 Then
```

```
        EigenIndex = 1
End If
'
'     Define Matrix Size and Reset
'
If (EigenIndex <= 0) Then
        NS = N
Else
        NS = 2 * N
End If

NTotal = 2 * N
Index = EigenIndex

For I = 1 To N
        If (EigenIndex >= 0) Then
                For J = 1 To N
                        A (J + N, I) = 0
                Next J
                A (I + N, I) = 1
        End If
        NComplex (I) = 0
Next I

Call ArrangeMatrix (N, A, EigenIndex)
Call QRutish (N, AbsErr, A, EigenIndex)

If (Index > 0) Then
        For I = 1 To N
                For J = 1 To N
                        A (I + 2 * N, J) = A (J + N, I)
                Next J
        Next I
End If

Call LRAlgorithm (N, Index, AbsErr, NComplex, A)

If (Index >= 0) Then
        Call EigenVector (N, NTotal, NS, NComplex, A)
End If

End Sub


Sub ArrangeMatrix (N, A, EigenIndex)
Dim SumA2, SX, SK As Double
If (N <= 2) Then Exit Sub
NMinus2 = N - 2
NTotal = N + N
If (EigenIndex < 0) Then NTotal = N
For I = 1 To NMinus2
        IPlus1 = I + 1
        IPlus2 = I + 2
```

```
        SumA2 = 0

        For J = IPlus2 To N
            SumA2 = SumA2 + A (J, I) * A (J, I)
        Next J

        If (SumA2 <> 0) Then
            SumA2 = Sqr (SumA2 + A (IPlus1, I) * A (IPlus1, I) )
            If (A (IPlus1, I) < 0) Then SumA2 = -SumA2
            A (IPlus1, I) = A (IPlus1, I) + SumA2
            SK = SumA2 * A (IPlus1, I)
            For J = IPlus1 To N
                SX = 0
                For K = IPlus1 To N
                    SX = SX + A (K, I) * A (K, J)
                Next K
                If (SX <> 0) Then
                    SX = SX / SK
                    For K = IPlus1 To N
                        A (K, J) = A (K, J) - SX * A (K, I)
                    Next K
                End If
            Next J

            For J = 1 To NTotal
                SX = 0
                For K = IPlus1 To N
                    SX = SX + A (K, I) * A (J, K)
                Next K
                If (SX <> 0) Then
                    SX = SX / SK
                    For K = IPlus1 To N
                        A (J, K) = A (J, K) - SX * A (K, I)
                    Next K
                End If
            Next J

            A (IPlus1, I) = -SumA2
            For J = IPlus2 To N
                A (J, I) = 0
            Next J
        End If
    Next I

    End Sub


    Sub QRutish (N, AbsErr, A, Index)
    Dim X, Y, Z, SX, SY As Double
    MT = 0
    If (Index < 0) Then MT = 1
    Index = 0
    NTotal = 2 * N
```

```
SX = 0
SY = 0
I = N + 1
Do
    I = I - 1

    If (I <= 2) Then Exit Sub
    Do
        IFLAG = 0
        M = 1
        Index = Index + 1
        II = I
        For JA = 2 To II
            J = I + 2 - JA
            K = J - 1
            X = Abs (A (J, K) ) / (Abs (A (J, J) ) + Abs (A (K, K) ) + AbsErr)
            If (X <= AbsErr) Then
                M = J
                A (J, K) = 0
                If (M <= (I - 2) ) Then Exit For
                If (M = (I - 1) ) Then I = I - 1
                IFLAG = 1
                Exit For
            End If
        Next JA

        If IFLAG = 0 Then
            If (Index <> 1) Then
                SX = A (I, I) + A (I - 1, I - 1)
                SY = A (I, I) * A (I - 1, I - 1) - A (I, I - 1) * A (I - 1, I)
                If (SX = 0 And SY = 0) Then SX = A (I, I - 1)
            End If
            X = A (M, M) / A (M + 1, M) * (A (M, M) - SX) + A (M, M + 1) + SY / A (M + 1, M)
            Y = A (M, M) + A (M + 1, M + 1) - SX
            Z = A (M + 2, M + 1)
            IL = I - 1
            NL1 = N + MT * (I - N)
            NL2 = 1 + (M - 1) * MT
            NL3 = NTotal + (I - NTotal) * MT
            For J = M To IL
                SumSquare = Sqr (X * X + Y * Y + Z * Z)
                If (X < 0) Then SumSquare = -SumSquare
                X = X + SumSquare
                Y = Y / X
                Z = Z / X
                X = X / SumSquare
                For K = J To NL1
                    SX = A (J, K) + Y * A (J + 1, K)
                    If (J < IL) Then SX = SX + Z * A (J + 2, K)
                    SX = SX * X
                    A (J, K) = A (J, K) - SX
                    If (J < IL) Then A (J + 2, K) = A (J + 2, K) - Z * SX
                    A (J + 1, K) = A (J + 1, K) - Y * SX
```

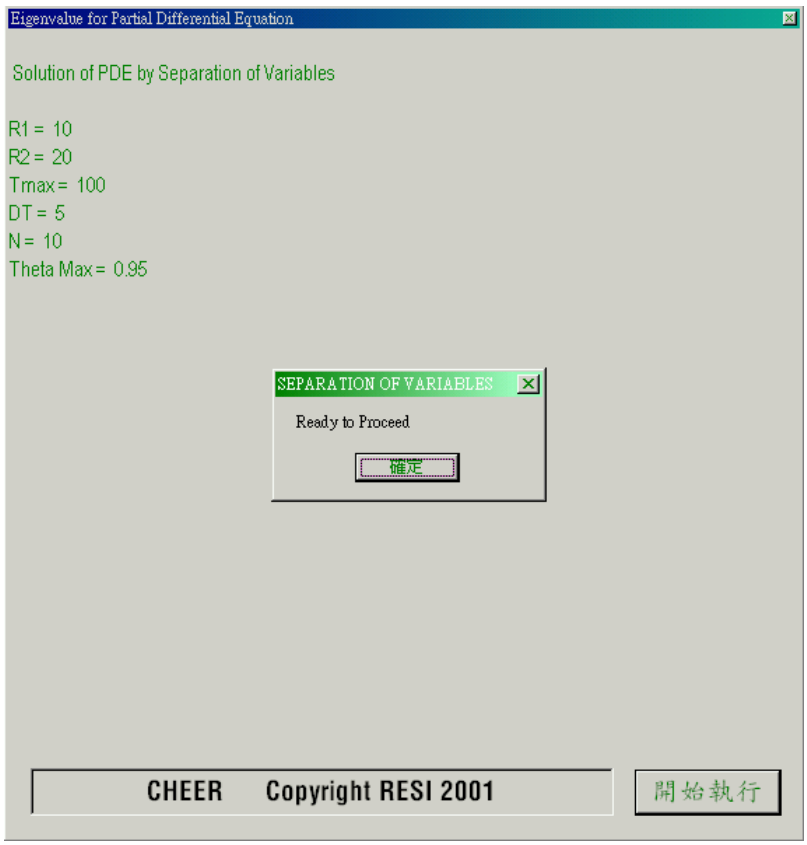
```
                    Next K
                    For K = NL2 To NL3
                        JFLAG = 0
                        If ( (K > N) Or ( (K < J + 4) And (K <= I) ) ) Then
                            SX = A (K, J) + Y * A (K, J + 1)
                            If (J < IL) Then SX = SX + Z * A (K, J + 2)
                            SX = SX * X
                            A (K, J) = A (K, J) - SX
                            A (K, J + 1) = A (K, J + 1) - Y * SX
                            If (J < IL) Then A (K, J + 2) = A (K, J + 2) - Z * SX
                        End If
                    Next K
                    If (J > M) Then
                        A (J, J - 1) = -SumSquare
                        A (J + 1, J - 1) = 0
                        If (J < I - 1) Then A (J + 2, J - 1) = 0
                    End If
                    X = A (J + 1, J)
                    Y = 0
                    Z = 0
                    If (J < IL) Then Y = A (J + 2, J)
                    If (J < I - 2) Then Z = A (J + 3, J)
                Next J
            Else
                Exit Do
            End If
            If (Index >= 10 * N) Then Exit Sub
        Loop While (Index < 10 * N)
    Loop While IFLAG = 1
    End Sub


    Sub LRAlgorithm (N, Index, AbsErr, NComplex, A)
    Dim Q, X, Y, Z, R As Double

    NS = 2 * N
    NTotal = NS
    If (Index <= 0) Then NS = N
    I = 0
    Do
        ILoop = 0
        I = I + 1

        If (I - N > 0) Then Exit Sub
        If (I < N) Then
            X = Abs (A (I + 1, I) ) / (Abs (A (I, I) ) + Abs (A (I + 1, I + 1) ) + AbsErr)

            If (X > AbsErr) Then
                Y = A (I, I) + A (I + 1, I + 1)
                Z = A (I, I) * A (I + 1, I + 1) - A (I, I + 1) * A (I + 1, I)
                X = Y * Y - 4 * Z

                If (X >= 0) Then
```

```
            R = (Abs (Y) + Sqr (X) ) / 2
            If (Y < 0) Then R = -R
            If (Index < 0) Then
                A (I, 1) = R
                A (I + 1, 1) = Z / R
                ILoop = 1
            Else
                Q = A (I, I) - R
                S = A (I, I) - Z / R
                If (Abs (S) > Abs (Q) ) Then Q = S
                Q = A (I + 1, I) / Q
                C = 1 / Sqr (Q * Q + 1)
                S = Q * C
            End If
        Else
            R = (A (I + 1, I + 1) - A (I, I) ) / (A (I + 1, I) + A (I, I + 1) )
            If (Index < 0) Then
                A (I, 1) = Y / 2
                A (I + 1, 1) = Sqr (-X) / 2
                NComplex (I) = 1
                NComplex (I + 1) = 2
                ILoop = 1
            Else
                Q = 1 / Sqr (1 + R * R)
                C = Sqr ( (Q + 1) / 2)
                S = R * Q / C / 2
            End If
        End If

        If ILoop = 0 Then
            For J = 1 To NS
                K1 = I
                K2 = J
                If (J > N) Then
                    K1 = I + 2 * N
                    K2 = J - N
                End If
                Q = A (K1, K2)
                A (K1, K2) = Q * C + S * A (K1 + 1, K2)
                A (K1 + 1, K2) = A (K1 + 1, K2) * C - S * Q
            Next J

            For J = 1 To NTotal
                If ( (J < I + 2) Or (J > N) ) Then
                    Q = A (J, I)
                    A (J, I) = A (J, I) * C + A (J, I + 1) * S
                    A (J, I + 1) = C * A (J, I + 1) - Q * S
                End If
            Next J

            If (X < 0) Then
                NComplex (I) = 1
                NComplex (I + 1) = 2
```

```
                                R = Sqr (Abs (A (I + 1, I) / A (I, I + 1) ) )
                                For J = 1 To NTotal
                                    A (J, I + 1) = A (J, I + 1) * R
                                    If (J <= N) Then
                                        A (I + 1, J) = A (I + 1, J) / R
                                    ElseIf (J <= NS) Then
                                        A (I + 2 * N + 1, J - N) = A (I + 2 * N + 1, J - N) / R
                                    End If
                                Next J
                                ILoop = 1
                            Else
                                A (I + 1, I) = 0
                            End If
                        End If
                    Else
                        A (I + 1, I) = 0
                        ILoop = 0
                    End If
                End If

                If (ILoop = 1) Then
                    I = I + 1
                ElseIf (Index < 0) Then
                    A (I, 1) = A (I, I)
                End If

            Loop
            End Sub


            Sub EigenVector (N, NVEC, NROW, NComplex, A)
            Dim B, D, DB, Det, R1, R2, RI1, RI2, X (2, 2) As Double
            IX = N
            NE = N + NROW
            Do While (IX > 0)
                C = A (IX, IX)
                D = 0
                I1 = 2
                If (NComplex (IX) > 0) Then
                    I1 = 1
                    D = A (IX - 1, IX)
                End If
                JX = IX + I1 - 3
                Do While (JX > 0)
                    AC = A (JX, JX) - C
                    J1 = 2
                    B = 0

                    If (NComplex (JX) > 0) Then
                        J1 = 1
                        B = A (JX - 1, JX)
                    End If
```

```
            For J = 1 To 2
                For I = 1 To 2
                    X (I, J) = 0
                Next I
            Next J

            For I = I1 To 2
                For J = J1 To 2
                    X (J, I) = A (JX + J - 2, IX + I - 2)
                    A (JX + J - 2, IX + I - 2) = 0
                Next J
            Next I

            U = (X (1, 1) + X (2, 2) ) / 2
            S = X (1, 1) - U
            V = (X (1, 2) - X (2, 1) ) / 2
            T = X (1, 2) - V

            DB = B - D
            Det = AC * AC + DB * DB
            R1 = (U * AC + V * DB) / Det
            RI1 = (V * AC - U * DB) / Det

            DB = D + B
            Det = AC * AC + DB * DB
            R2 = (S * AC - T * DB) / Det
            RI2 = (T * AC + S * DB) / Det

            X (1, 1) = R1 + R2
            X (2, 2) = R1 - R2
            X (1, 2) = RI2 - RI1
            X (2, 1) = RI2 + RI1

            For K = 1 To NE
                If (K > N Or K < JX + J1 - 2) Then
                    For I = I1 To 2
                        For J = J1 To 2
                            If (K <= 2 * N) Then
                                A (K, IX + I - 2) = A (K, IX + I - 2) - X (I, J) * A (K, JX + J - 2)
                            Else
                                A (JX + J + 2 * N - 2, K - 2 * N) = A (JX + J + 2 * N - 2, K - 2 * N)
                                    + X (I, J) * A (IX + I + 2 * N - 2, K - 2 * N)
                            End If
                        Next J
                    Next I
                End If
            Next K

            JX = JX + J1 - 3
        Loop
        IX = IX + I1 - 3
    Loop
    End Sub
```

測試數據與結果：



** Starting Matrix is

2.0000	-2.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
-0.9545	2.0000	-1.0455	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	-0.9583	2.0000	-1.0417	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	-0.9615	2.0000	-1.0385	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	-0.9643	2.0000	-1.0357	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	-0.9667	2.0000	-1.0333	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	-0.9688	2.0000	-1.0313	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	-0.9706	2.0000	-1.0294	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	-0.9722	2.0000	-1.0278
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	-0.9737	2.0000

** Tranformed Matrix is

3.967979	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
0.000000	3.774435	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
0.000000	0.000000	3.408106	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
0.000000	0.000000	0.000000	2.904043	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
0.000000	0.000000	0.000000	0.000000	2.311510	0.000000	0.000000	0.000000	0.000000	0.000000
0.000000	0.000000	0.000000	0.000000	0.000000	1.688490	0.000000	0.000000	0.000000	0.000000

0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.095957	0.000000	0.000000	0.000000
0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.591894	0.000000	0.000000
0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.225565	0.000000
0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.032021

** Eigenrow Matrix is

-0.437758	-0.504747	-0.515492	-0.519606	-0.521809	0.523034	0.523338	-0.521916	-0.514331	0.457603
0.430749	0.447821	0.362934	0.234873	0.081274	0.081465	0.236560	-0.367457	-0.456323	0.450277
-0.411157	-0.299224	-0.018163	0.271320	0.452217	-0.453279	-0.273268	-0.018390	-0.304905	0.429796
0.380493	0.097720	-0.309346	-0.451556	-0.210008	-0.210501	-0.454799	0.313202	-0.099575	0.397742
-0.340369	0.110083	0.436277	0.141885	-0.355723	0.356558	-0.142904	0.441714	0.112174	0.355799
0.292488	-0.279581	-0.305129	0.296567	0.302515	0.303225	0.298697	0.308932	0.284889	0.305748
-0.238633	0.377114	0.007664	-0.392192	0.241577	-0.242144	0.395009	0.007760	0.384274	0.249451
0.180633	-0.386250	0.276172	0.065221	-0.357154	-0.357992	0.065689	-0.279613	0.393583	0.188821
-0.120328	0.310228	-0.384994	0.312503	-0.119694	0.119975	-0.314748	-0.389792	0.316119	0.125783
0.059534	-0.170231	0.266218	-0.336577	0.374126	0.375005	-0.338994	-0.269535	0.173464	0.062233

** Eigencolumn Matrix is

-0.197160	0.406483	-0.423267	0.424342	-0.408793	0.376379	-0.327549	0.263434	-0.185808	0.097038
-0.201075	0.373784	-0.272459	0.096394	0.116943	-0.318216	0.457843	-0.498243	0.423719	-0.245424
-0.200984	0.296484	-0.016187	-0.298655	0.453599	-0.339904	0.009107	0.348666	-0.514645	0.375640
-0.200538	0.189928	0.239345	-0.431536	0.146025	0.327022	-0.461298	0.081508	0.413513	-0.470110
-0.200061	0.065289	0.396295	-0.199374	-0.363689	0.331382	0.282270	-0.443399	-0.157339	0.519114
0.199592	0.065136	-0.395367	-0.198907	0.362837	0.330605	-0.281609	-0.442360	0.156970	0.517897
0.199108	0.188573	-0.237638	-0.428459	-0.144984	0.324690	0.458008	0.080926	-0.410565	-0.466758
-0.198510	-0.292835	-0.015987	0.294979	0.448015	0.335720	0.008994	-0.344374	-0.508311	-0.371016
-0.197328	-0.366819	-0.267382	-0.094598	0.114764	0.312287	0.449312	0.488959	0.415824	0.240851
0.188610	0.388855	0.404911	0.405939	0.391065	0.360056	0.313344	0.252009	0.177750	0.092830

** Orthogonality requires off-digonal elements to be zero

1.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
0.00000	1.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
0.00000	0.00000	1.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
0.00000	0.00000	0.00000	1.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
0.00000	0.00000	0.00000	0.00000	1.00000	0.00000	0.00000	0.00000	0.00000	0.00000
0.00000	0.00000	0.00000	0.00000	0.00000	1.00000	0.00000	0.00000	0.00000	0.00000
0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	1.00000	0.00000	0.00000	0.00000
0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	1.00000	0.00000	0.00000
0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	1.00000	0.00000
0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	1.00000

** The Coefficient C (J) are

0.02510	-0.06673	0.11312	-0.16614	0.23049	0.31479	-0.43710	-0.64332	1.09587	2.97537

Time Temp at Grid Points.

0.00E+00	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	1.00000
5.00E+00	0.00588	0.00853	0.01748	0.03661	0.07264	0.13463	0.23226	0.37249	0.55538	0.77105	1.00000
1.00E+01	0.06970	0.07913	0.10657	0.15280	0.21931	0.30736	0.41704	0.54644	0.69131	0.84518	1.00000
1.50E+01	0.17684	0.18818	0.22028	0.27167	0.34101	0.42669	0.52659	0.63781	0.75670	0.87898	1.00000
2.00E+01	0.28855	0.29935	0.32964	0.37744	0.44067	0.51708	0.60417	0.69914	0.79893	0.90031	1.00000
2.50E+01	0.39054	0.40010	0.42688	0.46891	0.52413	0.59034	0.66518	0.74616	0.83069	0.91617	1.00000
3.00E+01	0.47965	0.48791	0.51104	0.54728	0.59476	0.65153	0.71550	0.78452	0.85639	0.92893	1.00000

3.50E+01	0.55629	0.56337	0.58318	0.61419	0.65479	0.70328	0.75785	0.81666	0.87785	0.93956	1.00000
4.00E+01	0.62182	0.62787	0.64478	0.67124	0.70589	0.74724	0.79376	0.84387	0.89599	0.94854	1.00000
4.50E+01	0.67773	0.68289	0.69731	0.71988	0.74941	0.78465	0.82430	0.86700	0.91140	0.95616	1.00000
5.00E+01	0.72540	0.72979	0.74208	0.76132	0.78648	0.81652	0.85030	0.88668	0.92451	0.96265	1.00000
5.50E+01	0.76602	0.76977	0.78024	0.79663	0.81807	0.84366	0.87245	0.90345	0.93568	0.96818	1.00000
6.00E+01	0.80064	0.80383	0.81275	0.82671	0.84499	0.86679	0.89132	0.91773	0.94520	0.97289	1.00000
6.50E+01	0.83013	0.83285	0.84045	0.85235	0.86792	0.88650	0.90740	0.92991	0.95331	0.97690	1.00000
7.00E+01	0.85526	0.85758	0.86406	0.87419	0.88746	0.90329	0.92110	0.94028	0.96021	0.98032	1.00000
7.50E+01	0.87667	0.87865	0.88417	0.89281	0.90411	0.91760	0.93277	0.94911	0.96610	0.98323	1.00000
8.00E+01	0.89492	0.89660	0.90130	0.90867	0.91830	0.92979	0.94272	0.95664	0.97112	0.98571	1.00000
8.50E+01	0.91047	0.91190	0.91591	0.92218	0.93038	0.94018	0.95119	0.96306	0.97539	0.98782	1.00000
9.00E+01	0.92371	0.92493	0.92835	0.93369	0.94068	0.94903	0.95841	0.96852	0.97903	0.98962	1.00000
9.50E+01	0.93500	0.93604	0.93895	0.94350	0.94946	0.95657	0.96457	0.97318	0.98213	0.99116	1.00000
1.00E+02	0.94461	0.94550	0.94798	0.95186	0.95694	0.96299	0.96981	0.97715	0.98478	0.99247	1.00000

第九節　配重殘值法

Visual Basic

考慮設計問題-XI 的流體薄膜吸附問題。此問題之數學模式爲

$$
\begin{aligned}
&(1-x^2)\frac{\partial y}{\partial z}=\frac{\partial^2 y}{\partial x^2}\\
&\qquad IC \qquad y=1 \qquad\quad @z=0\\
&\qquad BC1 \qquad y=0 \qquad\quad @x=0 \qquad\quad z>0\\
&\qquad BC2 \qquad \frac{dy}{dx}=0 \quad @x=1 \;\; z>0
\end{aligned}
\qquad \textbf{(11-9.1)}
$$

仿照第十章常微分方程式所介紹的配重殘值法 (Method of Weighted Residuals, MWR)，令此偏微分方程式的N階近似解爲一多項式基礎函數的配重和。由於基礎函數 $u_i(x)$ 必須滿足偏微分方程式的均勻邊界條件 (即常數項爲零的邊界條件)，由於在本例子中

$$y(0)=0 \qquad y'(1)=0 \qquad \textbf{(11-9.2)}$$

因此，可以將多項式基礎函數寫成

$$u_i(x)=(i+1)x-x^{i+1} \qquad \textbf{(11-9.3)}$$

$$y_N(x,z)=u_0(x,z)+\sum_{i=1}^{N} a_i(z)\,u_i(x) \qquad \textbf{(11-9.4)}$$

其中$u_0(x,z)$ 爲滿足邊界條件及偏微分方程式的函數，基礎函數$u_i(x)$ 只爲x的函數，係數$a_i(z)$ 爲z的函數。將此基礎函數代入偏微分方程式 (11-9.1) 中，利用配重殘值法求

解$a_i(z)$。

如第十章所述，配重殘值法的基本觀念，是利用一組多項式表示近似於問題的正確解，將正確解與近似解的誤差殘值，乘以適當的配重函數後，使其總和或積分值最小化，以達到最接近正確解的目標。微分方程式 (11-9.1) 的殘值爲

$$\begin{aligned} R_N(\underline{a},x,z) &= (1-x^2)\frac{\partial y_N}{\partial z} - \frac{\partial^2 y_N}{\partial x^2} \\ &= (1-x^2)\sum_{i=1}^{N}\frac{da_i(z)}{dz}u_i(x) - \sum_{i=1}^{N}a_i(z)\frac{d^2u_i(x)}{dx^2} \\ &= (1-x^2)\sum_{i=1}^{N}\frac{da_i}{dz}\left[(i+1)x - x^{i+1}\right] - \sum_{i=1}^{N}a_i\left[-(i+1)ix^{i-1}\right] \end{aligned} \quad \textbf{(11-9.5)}$$

葛勒金法 (Galerkin's Method)

葛勒金法的配重函數爲$W = \partial y_N / \partial a_i = u_i$ ，亦即，R_N與u_i正交；

$$\int_0^1 R_N(\underline{a}\,,x\,,z)u_i(x)dx = 0 \qquad j = 1,2,....,N \quad \textbf{(11-9.6)}$$

將方程式 (11-9.5) 代入方程式 (11-9.6) 中，經整理後可以得到矩陣方程式

$$\underline{\underline{C}}\frac{d\underline{a}}{dz} = \underline{\underline{B}}\underline{a} \quad \textbf{(11-9.7)}$$

$$C_{ji} = \int_0^1 \left[(i+1)x - x^{i+1}\right]\left[(j+1)x - x^{j+1}\right](1-x^2)dx \quad \textbf{(11-9.8)}$$

$$B_{ji} = -i(i+1)\int_0^1 x^{i-1}\left[(j+1)x - x^{j+1}\right]dx \quad \textbf{(11-9.9)}$$

未定係數陣列 a 可以由方程式 (11-9.7) 求得。首先將方程式 (11-9.7) 改寫成方程式 (8-9.8) 的形式

$$\frac{d\underline{a}}{dz} = \underline{\underline{C}}^{-1}\underline{\underline{B}}\underline{a} \equiv \underline{\underline{A}}\underline{a} \quad \textbf{(11-9.10)}$$

$$\begin{aligned} &\frac{d\underline{a}}{dz} = \underline{\underline{U}}\,\underline{\underline{\Lambda}}\,\underline{\underline{U}}^{-1}\underline{a} \\ &\frac{d(\underline{\underline{U}}^{-1}\underline{a})}{dz} = \underline{\underline{\Lambda}}(\underline{\underline{U}}^{-1}\underline{a}) \end{aligned} \quad \textbf{(11-9.11)}$$

仿方程式 (8-9.8) 的處理方式，解方程式 (11-9.11)，可以得到

$$\underline{a} = \underline{\underline{U}} \exp(\underline{\underline{\Lambda}} t) \underline{\underline{U}}^{-1} \underline{a}_0 \qquad \textbf{(11-9.12)}$$

計算策略：

利用葛勒金法及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 (11-9.4) 的標準形式。

2. 代回微分方程式中，建立殘值R_N的表示式，如方程式 (11-9.5)。

3. 由起始條件求出$\underline{a}_0 = \underline{a}$ (z=0)，使用葛勒金法

$$\int_0^1 \left\{ u_0(x,0) - \sum_{i=1}^{N} a_{i0} \left[(i+1)x - x^{i+1} \right] \right\} (1-x^2) \left[(j+1)x - x^{j+1} \right] dx = 0 \qquad \textbf{(11-9.13)}$$

其中u_0 (x , 0) =1，代入上式，並利用C_{ji}之定義方程式 (11-9.8)，可將上式改寫成

$$\underline{\underline{C}} \underline{a}_0 = \underline{s}$$

$$s_j = \frac{1}{4}(j+1) - \frac{2}{(j+2)(J+4)} \qquad \textbf{(11-9.14)}$$

4. 依方程式 (11-9.7) 的定義，建立矩陣 $\underline{\underline{A}}$。

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

6. 利用方程式 (11-9.12)，計算$\underline{a}$。

(1) 首先將$\underline{\underline{U}}^{-1}$與$\underline{a}_0$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 $\underline{a} = \underline{\underline{U}} \underline{w}_2$ 。

7. 利用方程式 (11-9.4) 計算$\underline{y}$。

正交配置法 (Orthogonal Collocation Method)

利用正交配置法解偏微分方程式 (11-9.1) 時，第N個基礎函數可以利用配置點x_i上的函數值$y(x_i , z)$ 及邊界條件$y(0,z) = y_0 = 0$ 及$y(1,z) = y_{N+1}$來表示。起始條件y (x_i , 0) 可以直接寫成配置點上的起始條件；

$$y(x_i , 0) = 1 \qquad i = 1, 2, \cdots, N$$

或

$$\underline{y}_0^T = \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix} \qquad \textbf{(11-9.15)}$$

由於使用正交配置法時，近似函數在配置點上的殘值均爲零。利用第十章所介紹的微分操作矩陣，可以將偏微分方程式改寫成

$$(1-x_j^2)\frac{dy_j}{dz}=\sum_{i=0}^{N+1}B_{ji}y_i \qquad \textbf{(11-9.16)}$$

其中B_{ji}爲二次導函數操作矩陣，詳第十章說明。

邊界條件可以寫成

$$\begin{aligned} &y(0,z)=y_0=0\\ &\sum_{i=0}^{N+1}A_{N+1,i}y_i=0 \end{aligned} \qquad \textbf{(11-9.17)}$$

將方程式 (11-9.17) 的邊界條件，仿第十章說明，代入方程式 (11-9.16) 中，經整理後得到

$$\frac{d\underline{y}}{dz}=\underline{\underline{C}}\,\underline{y} \qquad \textbf{(11-9.18)}$$

或

$$\frac{dy_j}{dz}=\sum_{i=0}^{N+1}C_{ji}y_i$$

$$C_{ji}=\frac{1}{1-x_j^2}\left[B_{ji}-\frac{B_{j,N+1}A_{N+1,j}}{A_{N+1,N+1}}\right]$$

方程式 (11-9.18) 的解爲

$$\underline{y}=\underline{\underline{U}}\exp(\underline{\underline{\Lambda}}t)\underline{\underline{U}}^{-1}\underline{y}_0 \qquad \textbf{(11-9.19)}$$

計算策略：

利用正交配置法及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 (11-9.4) 的標準形式。
2. 代回微分方程式中，建立殘值R_N的表示式，如方程式 (11-9.5)。
3. 由起始條件仿方程式 (11-9.15)，建立$\underline{y}_0$。
4. 利用正交配置法建立微分操作矩陣 $\underline{\underline{A}}$及 $\underline{\underline{B}}$。
5. 依方程式 (11-9.18) 的定義，建立矩陣 $\underline{\underline{C}}$。
6. 求解矩陣 $\underline{\underline{C}}$的特徵值，並建立其特徵矩陣 $\underline{\underline{U}}$。

7. 利用方程式 (11-9.19)，計算$\underline{y}$。

(1) 首先將$\underline{\underline{U}}^{-1}$與$\underline{y}_0$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 $\underline{y}=\underline{\underline{U}}\underline{w}_2$。

例題 11-6　利用正交配置法解設計問題 -XI

設計問題-XI 的流體薄膜吸附問題數學模式爲

$$(1-x^2)\frac{\partial y}{\partial z}=\frac{\partial^2 y}{\partial x^2}$$

$$IC \quad y=1 \quad @z=0$$

$$BC1 \quad y=0 \quad @x=0 \quad z>0$$

$$BC2 \quad \frac{dy}{dx}=0 \quad @x=1 \quad z>0$$

在液體薄膜內的平均濃度及質傳薛武德 (Sherwood) 常數，分別可以用以下方程式表示

$$\langle y(z)\rangle=\frac{3}{2}\int_0^1(1-x^2)y(x,z)dx$$

$$Sh=-\frac{2}{3}\frac{(d\langle y\rangle/dz)}{\langle y\rangle}$$

利用正交配置法，求濃度分布及在各位置之質傳薛武德常數。

解：

1. 詳細計算方法及執行策略，已在前文中說明。

2. 此問題中所需使用的正交配置法，其中配置點的計算，微分操作矩陣的建立及積分配重函數的建立，詳本書第十章說明。

3. 矩陣特徵值及特徵矩陣之計算，詳本書第四章說明。

4. 因此，程式撰寫相當容易且直接；只需將以前說明過的程式作適當整合即可解此問題。

5. 詳程式內的說明文件。

程式列印：

```
'
' ORTHOGONAL COLLOCATION (3)
'
'*****************************************"
' SOLUTION OF P.D.E.
' BY    ORTHOGONAL    COLLOCATION
'*****************************************"
'
' Program Developed by Dr. Ron Hsin Chang
' Copyright 1988, 2001
'
' Example 11-6
'
Sub OrthogonalCollocationPDE (Xpos, Ypos)
Dim Z (50) , Coef (50) , XP (50) As Double
Dim Z1 (50) , Z2 (50) , Z3 (50) As Double
Dim Diff1 (50) , Diff2 (50) , Diff3 (50) , Root (50) As Double
Dim A (50, 50) , B (150, 50) , W (50) As Double
Dim Vect (50) , NC (50) , AbsErr As Double
Dim Xtab (20) As Double
Do
'
'     Enter Basic Data & DEFINE THE PROBLEM
'
      Cls
      Print
      Print " Solution of PDE by Collocation Method"
      Print
      Call Xpositions (Xtab, Ntab)
      Call CollocationBasicData (Xpos, Ypos, GeoFact, Alpha, Beta, N, N1, N2, NoColloPoints, Diff1,
                                 Diff2, Diff3, Root, Vect, A, B, W)
      ND = N + N1 + N2
'
'     Define the Operation Matrix
'
      For I = 1 To N
          For J = 1 To N
              B (I, J) = (B (I + 1, J + 1) - B (I + 1, ND) / A (ND, ND) * A (ND, J + 1) ) /
                         (1 - Root (I + 1) * Root (I + 1) )
          Next J
      Next I
'
'     Modified Radau Qudrature Weight = W * 1.5 * (1-x^2)
'     and eliminate boundary conditions at y=0 and y=1
'
      Print
      Print
      Print "** MODIFIED QUDRATURE VECTOR W:"
      Print
      For I = 1 To N
          W (I) = W (I + 1) * 1.5 * (1 - Root (I + 1) * Root (I + 1) )
          Print Format (W (I) , "      0.0000E+00") ;
```

```
        Next I
        Print
        MsgBox ("Ready to Proceed")
'
'   Find Eigenvalues, Eigenvectors, and Eigenrows
'
        EigenIndex = 1
        AbsErr = 0.000000000001
        Call EigenSystem (N, EigenIndex, AbsErr, NC, B)
'
'   Find Vectors U (Z2) and V (Z1)
'   and the expansion coefficient (Z3)
'
        Cls
        For I = 1 To N
            Z1 (I) = 0
            Z2 (I) = 0
            For J = 1 To N
                Z1 (I) = Z1 (I) + B (I + 2 * N, J)
                Z2 (I) = Z2 (I) + W (J) * B (J + N, I)
            Next J
            Z3 (I) = Z1 (I) * Z2 (I)
        Next I
        Print
        Print "** SOLUTION SERIES"
        For I = 1 To N
            Print Format (Z3 (I) , "    0.00000000") ;
            Print " * Exp (";
            Print Format (B (I, I) , "0.000000E+00 ") ;
            Print " * Z) "
        Next I
'
'   Calculate the Sherwood Number at specicied positions
'
        Print
        Print "Distance                    Sherwood No."
        Print "-----------------------------------------------------"
        For I = 1 To Ntab
            X = 0
            S = 0
            For J = 1 To N
                T = B (J, J) * Xtab (I)
                If (T >= -60) Then
                    T = Z3 (J) * Exp (T)
                    X = X + T
                    S = S - T * B (J, J)
                End If
            Next J
            S = 2 * S / X / 3
            Print Format (Xtab (I) , " 0.000E+00            ") ;
            Print Format (S, "0.00000000E+00")
        Next I
        Print "-----------------------------------------------------"
        NxtJob = InputBox ("Run Next Job <Y/N> ? ", "NEXT JOB", "Y", Xpos, Ypos)
    Loop While NxtJob = "Y"
```

```
End Sub


Sub Xpositions (Xtab, Ntab)
'
'     Positions for Printing Out Results
'
Ntab = 16
Xtab (1) = 0.00001
Xtab (2) = 0.00002
Xtab (3) = 0.00005
Xtab (4) = 0.0001
Xtab (5) = 0.0002
Xtab (6) = 0.0005
Xtab (7) = 0.001
Xtab (8) = 0.002
Xtab (9) = 0.005
Xtab (10) = 0.01
Xtab (11) = 0.02
Xtab (12) = 0.05
Xtab (13) = 0.1
Xtab (14) = 0.2
Xtab (15) = 0.5
Xtab (16) = 1
End Sub


Sub CollocationBasicData (Xpos, Ypos, GeoFact, Alpha, Beta, N, N1, N2, NoColloPoints, Diff1,
Diff2, Diff3, Root, Vect, A, B, W)
'
'     Basic Data for Collocation Method
'
'     GeoFact S            = 0-planar; 1-cylinder; 2-sphere"
'     Alpha                 = 1
'     Beta                  = (GeoFact - 1) /2
'     N                       = Number of Interior Collocation Points
'     N1                      = 1 if x=0 is a collocation point, otherwise N1=0
'     N2                      = 1 if x=1 is a collocation point, otherwise N2=0
'     NoColloPoints     = Total number of collocation points, = N + N1 + N2
'     Diff1                 = First order derivative vector
'     Diff2                 = Second order derivative vector
'     Diff3                 = Third order derivative vector
'     Root                    = Root of Jacobi Polynomial, Collocation Points
'     Vect                    = Operation vector
'     A                       = First order derivative matrix
'     B                       = Second order derivative matrix
'     W                       = Radau Lobatto quadrature weights
'
Call GeoFactor (Xpos, Ypos, GeoFact, Alpha, Beta, N, N1, N2, NoColloPoints)
'
' ==CALL JACOBI to find collocation points
'
Call Jacobi (Alpha, Beta, NoColloPoints, N, N1, N2, Diff1, Diff2, Diff3, Root)
MsgBox ("Ready to Continue")
'
```

```
' ==CALL DEFMAT to define operation matrix
'
Cls
Call DefMatrix (NoColloPoints, N1, N2, Diff1, Diff2, Diff3, Root, Vect, A, B)
MsgBox ("Ready to Continue")
'
' ==RADAU AND LOBATTO QUADRATURE
'
ID = 2
Alpha = Alpha
Beta = Beta - 1

Call RadauQuadratureWeights (2, NoColloPoints, Root, N1, N2, Diff1, Vect, Alpha, Beta)

Print "** Radau Lobatto Qudratue Weight W:"
Print
For I = 1 To NoColloPoints
        W (I) = Vect (I)
        Print Format (W (I) , "      0.0000E+00") ;
Next I
End Sub


Sub GeoFactor (Xpos, Ypos, GeoFact, Alpha, Beta, N, N1, N2, ND)
'
'** DEFINE THE JACOBI POLYNOMIAL "
'      Generally, ALPHA=1, BETA= (S-1) /2"
'      S=0-planar; 1-cylinder; 2-sphere"
'
Alpha = 0
Alpha = Val (InputBox ("    Alpha = ", "Alpha", Alpha, Xpos, Ypos) )
Print "Alpha = "; Alpha

Beta = 1
Beta = Val (InputBox ("    Beta = ", "BETA", Beta, Xpos, Ypos) )
Print "Beta = "; Beta

N = 10
N = Val (InputBox (">> NUMBER OF COLLOCATION POINTS:", "Collocation Points", N, Xpos,
Ypos) )
Print "Number of Collocation Points N = "; N

Print
Print "      BOUNDARY COLLOCATION POINT (S) :"
Print "      0=not a collocation point"
Print "      1= is a collocation point"

N1 = 1
N1 = Val (InputBox ("Collocation at X=0        (0/1) ? ", "Collocation @ X=0", N1, Xpos, Ypos) )
Print "N1 = "; N1

N2 = 1
N2 = Val (InputBox ("Collocation at X=1        (0/1) ? ", "Collocation @ X=1", N2, Xpos, Ypos) )
Print "N2 = "; N2
```

```
ND = N + N1 + N2
End Sub


Sub DefMatrix (ND, N1, N2, Diff1, Diff2, Diff3, Root, Vect, A, B)
' ============================
' DERIVATIVE OPERATION MATRICES
' ============================
N = ND - N1 - N2
For I = 1 To ND
    Call OpMatrix (I, 1, ND, N1, N2, Diff1, Diff2, Diff3, Root, Vect)
    For J = 1 To ND
        A (I, J) = Vect (J)
    Next J
    Call OpMatrix (I, 2, ND, N1, N2, Diff1, Diff2, Diff3, Root, Vect)
    For J = 1 To ND
        B (I, J) = Vect (J)
    Next J
Next I

Print
Print "** MATRIX A:"
Print
For I = 1 To ND
    For J = 1 To ND
        Print Format (A (I, J) , " 0.00000E+00    ") ;
    Next J
    Print
Next I
Print

Print "** LAPLACIAN MATRIX B:"
Print
For I = 1 To ND
    For J = 1 To ND
        Print Format (B (I, J) , " 0.00000E+00    ") ;
    Next J
    Print
Next I
End Sub


Sub Jacobi (Alpha, Beta, ND, N, N1, N2, Diff1, Diff2, Diff3, Root)
' ========================
' ROOTS OF JACOBI POLYNOMIAL
' ========================
AlphaPlusBeta = Alpha + Beta
AlphaMinusBeta = Beta - Alpha
AlphaBeta = Beta * Alpha
Diff1 (1) = (AlphaMinusBeta / (AlphaPlusBeta + 2) + 1) / 2
Diff2 (1) = 0

If N >= 2 Then
    For I = 2 To N
        IM1 = I - 1
```

```
                Z = AlphaPlusBeta + 2 * IM1
                Diff1 (I) = (AlphaPlusBeta * AlphaMinusBeta / Z / (Z + 2) + 1) / 2
                If I = 2 Then
                        Diff2 (I) = (AlphaPlusBeta + AlphaBeta + IM1) / Z / Z / (Z + 1)
                Else
                        Z = Z * Z
                        Y = IM1 * (AlphaPlusBeta + IM1)
                        Y = Y * (AlphaBeta + Y)
                        Diff2 (I) = Y / Z / (Z - 1)
                End If
        Next I
End If
'
X = 0
For I = 1 To N
        Do
                XD = 0
                XN = 1
                XE = 0
                XM = 0
                For J = 1 To N
                        XP = (Diff1 (J) - X) * XN - Diff2 (J) * XD
                        XQ = (Diff1 (J) - X) * XM - Diff2 (J) * XE - XN
                        XD = XN
                        XE = XM
                        XN = XP
                        XM = XQ
                Next J
                ZC = 1
                Z = XN / XM
                If I > 1 Then
                        For J = 2 To I
                                ZC = ZC - Z / (X - Root (J - 1) )
                        Next J
                End If
                Z = Z / ZC
                X = X - Z
        Loop While (Abs (Z) > 0.000000001)
        Root (I) = X
        X = X + 0.0001
Next I

If (N1 = 1) Then
        For I = 1 To N
                J = N + 1 - I
                Root (J + 1) = Root (J)
        Next I
        Root (1) = 0
End If
If (N2 = 1) Then Root (ND) = 1

Print
Print "** COLLOCATION POINTS:"
Print
For I = 1 To ND
```

```
        Print Format (Root (I) , " 0.000000E+00     ") ;
        If (Int (I / 5) * 5 = I) Then Print
    Next I

    For I = 1 To ND
        X = Root (I)
        Diff1 (I) = 1
        Diff2 (I) = 0
        Diff3 (I) = 0
        For J = 1 To ND
            If J <> I Then
                Y = X - Root (J)
                Diff3 (I) = Y * Diff3 (I) + 3 * Diff2 (I)
                Diff2 (I) = Y * Diff2 (I) + 2 * Diff1 (I)
                Diff1 (I) = Y * Diff1 (I)
            End If
        Next J
    Next I
    End Sub


    Sub OpMatrix (Index, ID, ND, N1, N2, Diff1, Diff2, Diff3, Root, Vect)
    ' ================
    ' OPERATION MATRIX
    ' ================
    '
    ' ID = 1 for first order derivatives
    ' ID = 2 for second order derivatives
    ' ID = 3
    '
    ' --ENTRY POINT
    If ID = 3 Then
        Call GaussQuadratureWeights (ND, Root, N1, N2, Diff1, Vect)
    Else
        For J = 1 To ND
            If J = Index Then
                If ID = 1 Then
                    Vect (J) = Diff2 (Index) / Diff1 (Index) / 2
                Else
                    Vect (J) = Diff3 (Index) / Diff1 (Index) / 3
                End If
            Else
                Y = Root (Index) - Root (J)
                Vect (J) = Diff1 (Index) / Diff1 (J) / Y
                If ID = 2 Then Vect (J) = Vect (J) * (Diff2 (Index) / Diff1 (Index) - 2 / Y)
            End If
        Next J
    End If
    End Sub


    Sub GaussQuadratureWeights (ND, Root, N1, N2, Diff1, Vect)
    ' ============================
    ' GAUSSIAN QUADRATURE WEIGHTS
    ' ============================
```

```
Y = 0
For J = 1 To ND
        X = Root (J)
        AX = X * (1 - X)
        If N1 = 0 Then AX = AX / X / X
        If N2 = 0 Then AX = AX / (1 - X) / (1 - X)
        Vect (J) = AX / Diff1 (J) / Diff1 (J)
        Y = Y + Vect (J)
Next J
For J = 1 To ND
        Vect (J) = Vect (J) / Y
Next J
End Sub


Sub RadauQuadratureWeights (ID, ND, Root, N1, N2, Diff1, Vect, Alpha, Beta)
' ========================
' RADAU QUADRATURE WEIGHTS
' ========================
'
'       ID=1 Radau quad. weights including x=1
'       ID=2 Radau quad. weights including x=0
'       ID=3 Lobatto quad wieghts including both
'       WEIGHT FUNCTION W (X) = (1-X) ^APLHA*X^BETA
'
Sum = 0
For I = 1 To ND
        X = Root (I)
        Select Case ID
                Case 1
                        AX = X
                        If N1 = 0 Then AX = 1 / AX
                Case 2
                        AX = 1 - X
                        If N2 = 0 Then AX = 1 / AX
                Case 3
                        AX = 1
        End Select
        Vect (I) = AX / Diff1 (I) / Diff1 (I)
Next I

If ID <> 2 Then Vect (ND) = Vect (ND) / (1 + Alpha)
If ID > 1 Then Vect (1) = Vect (1) / (1 + Beta)

For I = 1 To ND
        Sum = Sum + Vect (I)
Next I

For I = 1 To ND
        Vect (I) = Vect (I) / Sum
Next I
End Sub


Sub EigenSystem (N, EigenIndex, AbsErr, NComplex, A)
```

```
'
'     EigenSystem Finds the Eigenvalues, Eigenvectors and Eigenrows
'     of an NxN Square Matrix A
'
'     N is the Size of the Actual Matrix
'
'     EigenIndex is an Indicator
'     EigenIndex    = -1    Gives Eignevalues only.
'                           The results are stored in the first column of A.
'                           Real and imaginary parts of eventual complex eigenvalues
'                           being in adjacent positions.
'     EigenIndex    = 0     Gives Eigenvalues and Eigenvectors.
'                           The Eigenvalues are on the main diagonal of A.
'                           Complex parts as a 2*2 block with the real parts (identical)
'                           on the diagonal, and the imaginary parts (identical, but in the
'                           opposite sign) as adjacent off-diagonal elements.
'                           The matrix of eigenvectors, Q, is stored in row N+1 - 2N
'     EigenIndex    = 1     Gives Eigenvalues, Eigenvectors, and Eigenrows.
'                           Eigenvalues and eigenvectors being stored as for EigenIndex =0.
'                           The eigenrows, Qinv, are stored in row 2N+1 - 3N
'
'     On exit, EigenIndex gives the number of iterations in the QR-step index.
'
'     AbsErr is the tolerance, and a suggested value is 1.E-8
'
'     NComplex is an indicator array for the Eigenvalues.
'     NComplex (I) =0         indicates that the I'th eigenvalue is real.
'     NComplex (I) =1         and NComplex (I+1) =2     indicate a complex pair of eigenvalues, with real
'                           part in A (I+1) and a complex part in A (I+2) for EigenIndex = -1.
'
'     The diagonalized matrix, D, the matrix of eigenvectors, Q, and
'     the matrix of eigenrows, Qinv, have the property, that
'           A = Q * D * Qinv
'
'     Check Error Criteria and EigenIndex
'
If (AbsErr <= 0) Then AbsErr = 0.000000000001

If (EigenIndex < -1) Then
      EigenIndex = -1
ElseIf EigenIndex > 1 Then
      EigenIndex = 1
End If
'
'     Define Matrix Size and Reset
'
If (EigenIndex <= 0) Then
      NS = N
Else
      NS = 2 * N
End If

NTotal = 2 * N
Index = EigenIndex
```

```
    For I = 1 To N
        If (EigenIndex >= 0) Then
            For J = 1 To N
                A (J + N, I) = 0
            Next J
            A (I + N, I) = 1
        End If
        NComplex (I) = 0
    Next I
    '
    Call ArrangeMatrix (N, A, EigenIndex)
    Call QRutish (N, AbsErr, A, EigenIndex)

    If (Index > 0) Then
        For I = 1 To N
            For J = 1 To N
                A (I + 2 * N, J) = A (J + N, I)
            Next J
        Next I
    End If

    Call LRAlgorithm (N, Index, AbsErr, NComplex, A)

    If (Index >= 0) Then
        Call EigenVector (N, NTotal, NS, NComplex, A)
    End If

    End Sub


    Sub ArrangeMatrix (N, A, EigenIndex)
    Dim SumA2, SX, SK As Double
    If (N <= 2) Then Exit Sub
    NMinus2 = N - 2
    NTotal = N + N
    If (EigenIndex < 0) Then NTotal = N
    For I = 1 To NMinus2
        IPlus1 = I + 1
        IPlus2 = I + 2
        SumA2 = 0

        For J = IPlus2 To N
            SumA2 = SumA2 + A (J, I) * A (J, I)
        Next J

        If (SumA2 <> 0) Then
            SumA2 = Sqr (SumA2 + A (IPlus1, I) * A (IPlus1, I) )
            If (A (IPlus1, I) < 0) Then SumA2 = -SumA2
            A (IPlus1, I) = A (IPlus1, I) + SumA2
            SK = SumA2 * A (IPlus1, I)
            For J = IPlus1 To N
                SX = 0
                For K = IPlus1 To N
                    SX = SX + A (K, I) * A (K, J)
                Next K
```

```
                If (SX <> 0) Then
                    SX = SX / SK
                    For K = IPlus1 To N
                        A (K, J) = A (K, J) - SX * A (K, I)
                    Next K
                End If
            Next J

            For J = 1 To NTotal
                SX = 0
                For K = IPlus1 To N
                    SX = SX + A (K, I) * A (J, K)
                Next K
                If (SX <> 0) Then
                    SX = SX / SK
                    For K = IPlus1 To N
                        A (J, K) = A (J, K) - SX * A (K, I)
                    Next K
                End If
            Next J

            A (IPlus1, I) = -SumA2
            For J = IPlus2 To N
                A (J, I) = 0
            Next J
        End If
    Next I

    End Sub


    Sub QRutish (N, AbsErr, A, Index)
    Dim X, Y, Z, SX, SY As Double
    MT = 0
    If (Index < 0) Then MT = 1
    Index = 0
    NTotal = 2 * N
    SX = 0
    SY = 0
    I = N + 1
    Do
        I = I - 1

        If (I <= 2) Then Exit Sub
        Do
            IFLAG = 0
            M = 1
            Index = Index + 1
            II = I
            For JA = 2 To II
                J = I + 2 - JA
                K = J - 1
                X = Abs (A (J, K) ) / (Abs (A (J, J) ) + Abs (A (K, K) ) + AbsErr)
                If (X <= AbsErr) Then
                    M = J
```

```
                A (J, K) = 0
                If (M <= (I - 2) ) Then Exit For
                If (M = (I - 1) ) Then I = I - 1
                IFLAG = 1
                Exit For
        End If
Next JA

If IFLAG = 0 Then
        If (Index <> 1) Then
                SX = A (I, I) + A (I - 1, I - 1)
                SY = A (I, I) * A (I - 1, I - 1) - A (I, I - 1) * A (I - 1, I)
                If (SX = 0 And SY = 0) Then SX = A (I, I - 1)
        End If
        X = A (M, M) / A (M + 1, M) * (A (M, M) - SX) + A (M, M + 1) + SY / A (M + 1, M)
        Y = A (M, M) + A (M + 1, M + 1) - SX
        Z = A (M + 2, M + 1)
        IL = I - 1
        NL1 = N + MT * (I - N)
        NL2 = 1 + (M - 1) * MT
        NL3 = NTotal + (I - NTotal) * MT
        For J = M To IL
                SumSquare = Sqr (X * X + Y * Y + Z * Z)
                If (X < 0) Then SumSquare = -SumSquare
                X = X + SumSquare
                Y = Y / X
                Z = Z / X
                X = X / SumSquare
                For K = J To NL1
                        SX = A (J, K) + Y * A (J + 1, K)
                        If (J < IL) Then SX = SX + Z * A (J + 2, K)
                        SX = SX * X
                        A (J, K) = A (J, K) - SX
                        If (J < IL) Then A (J + 2, K) = A (J + 2, K) - Z * SX
                        A (J + 1, K) = A (J + 1, K) - Y * SX
                Next K
                For K = NL2 To NL3
                        JFLAG = 0
                        If ( (K > N) Or ( (K < J + 4) And (K <= I) ) ) Then
                                SX = A (K, J) + Y * A (K, J + 1)
                                If (J < IL) Then SX = SX + Z * A (K, J + 2)
                                SX = SX * X
                                A (K, J) = A (K, J) - SX
                                A (K, J + 1) = A (K, J + 1) - Y * SX
                                If (J < IL) Then A (K, J + 2) = A (K, J + 2) - Z * SX
                        End If
                Next K
                If (J > M) Then
                        A (J, J - 1) = -SumSquare
                        A (J + 1, J - 1) = 0
                        If (J < I - 1) Then A (J + 2, J - 1) = 0
                End If
                X = A (J + 1, J)
                Y = 0
                Z = 0
```

```
                    If (J < IL) Then Y = A (J + 2, J)
                    If (J < I - 2) Then Z = A (J + 3, J)
                Next J
            Else
                Exit Do
            End If
            If (Index >= 10 * N) Then Exit Sub
        Loop While (Index < 10 * N)
    Loop While IFLAG = 1
    End Sub


    Sub LRAlgorithm (N, Index, AbsErr, NComplex, A)
    Dim Q, X, Y, Z, R As Double

    NS = 2 * N
    NTotal = NS
    If (Index <= 0) Then NS = N
    I = 0
    Do
        ILoop = 0
        I = I + 1

        If (I - N > 0) Then Exit Sub
        If (I < N) Then
            X = Abs (A (I + 1, I) ) / (Abs (A (I, I) ) + Abs (A (I + 1, I + 1) ) + AbsErr)

            If (X > AbsErr) Then
                Y = A (I, I) + A (I + 1, I + 1)
                Z = A (I, I) * A (I + 1, I + 1) - A (I, I + 1) * A (I + 1, I)
                X = Y * Y - 4 * Z

                If (X >= 0) Then
                    R = (Abs (Y) + Sqr (X) ) / 2
                    If (Y < 0) Then R = -R
                    If (Index < 0) Then
                        A (I, 1) = R
                        A (I + 1, 1) = Z / R
                        ILoop = 1
                    Else
                        Q = A (I, I) - R
                        S = A (I, I) - Z / R
                        If (Abs (S) > Abs (Q) ) Then Q = S
                        Q = A (I + 1, I) / Q
                        C = 1 / Sqr (Q * Q + 1)
                        S = Q * C
                    End If
                Else
                    R = (A (I + 1, I + 1) - A (I, I) ) / (A (I + 1, I) + A (I, I + 1) )
                    If (Index < 0) Then
                        A (I, 1) = Y / 2
                        A (I + 1, 1) = Sqr (-X) / 2
                        NComplex (I) = 1
                        NComplex (I + 1) = 2
                        ILoop = 1
```

```
                    Else
                        Q = 1 / Sqr (1 + R * R)
                        C = Sqr ( (Q + 1) / 2)
                        S = R * Q / C / 2
                    End If
                End If

                If ILoop = 0 Then
                    For J = 1 To NS
                        K1 = I
                        K2 = J
                        If (J > N) Then
                            K1 = I + 2 * N
                            K2 = J - N
                        End If
                        Q = A (K1, K2)
                        A (K1, K2) = Q * C + S * A (K1 + 1, K2)
                        A (K1 + 1, K2) = A (K1 + 1, K2) * C - S * Q
                    Next J

                    For J = 1 To NTotal
                        If ( (J < I + 2) Or (J > N) ) Then
                            Q = A (J, I)
                            A (J, I) = A (J, I) * C + A (J, I + 1) * S
                            A (J, I + 1) = C * A (J, I + 1) - Q * S
                        End If
                    Next J

                    If (X < 0) Then
                        NComplex (I) = 1
                        NComplex (I + 1) = 2
                        R = Sqr (Abs (A (I + 1, I) / A (I, I + 1) ) )
                        For J = 1 To NTotal
                            A (J, I + 1) = A (J, I + 1) * R
                            If (J <= N) Then
                                A (I + 1, J) = A (I + 1, J) / R
                            ElseIf (J <= NS) Then
                                A (I + 2 * N + 1, J - N) = A (I + 2 * N + 1, J - N) / R
                            End If
                        Next J
                        ILoop = 1
                    Else
                        A (I + 1, I) = 0
                    End If
                End If
            Else
                A (I + 1, I) = 0
                ILoop = 0
            End If
        End If

        If (ILoop = 1) Then
            I = I + 1
        ElseIf (Index < 0) Then
            A (I, 1) = A (I, I)
```

```
        End If

    Loop
    End Sub


    Sub EigenVector (N, NVEC, NROW, NComplex, A)
    Dim B, D, DB, Det, R1, R2, RI1, RI2, X (2, 2) As Double
    IX = N
    NE = N + NROW
    Do While (IX > 0)
        C = A (IX, IX)
        D = 0
        I1 = 2
        If (NComplex (IX) > 0) Then
            I1 = 1
            D = A (IX - 1, IX)
        End If
        JX = IX + I1 - 3
        Do While (JX > 0)
            AC = A (JX, JX) - C
            J1 = 2
            B = 0

            If (NComplex (JX) > 0) Then
                J1 = 1
                B = A (JX - 1, JX)
            End If

            For J = 1 To 2
                For I = 1 To 2
                    X (I, J) = 0
                Next I
            Next J

            For I = I1 To 2
                For J = J1 To 2
                    X (J, I) = A (JX + J - 2, IX + I - 2)
                    A (JX + J - 2, IX + I - 2) = 0
                Next J
            Next I

            U = (X (1, 1) + X (2, 2) ) / 2
            S = X (1, 1) - U
            V = (X (1, 2) - X (2, 1) ) / 2
            T = X (1, 2) - V

            DB = B - D
            Det = AC * AC + DB * DB
            R1 = (U * AC + V * DB) / Det
            RI1 = (V * AC - U * DB) / Det

            DB = D + B
            Det = AC * AC + DB * DB
            R2 = (S * AC - T * DB) / Det
```

```
            RI2 = (T * AC + S * DB) / Det

            X (1, 1) = R1 + R2
            X (2, 2) = R1 - R2
            X (1, 2) = RI2 - RI1
            X (2, 1) = RI2 + RI1

            For K = 1 To NE
                If (K > N Or K < JX + J1 - 2) Then
                    For I = I1 To 2
                        For J = J1 To 2
                            If (K <= 2 * N) Then
                                A (K, IX + I - 2) = A (K, IX + I - 2) - X (I, J) * A (K, JX + J - 2)
                            Else
                                A (JX + J + 2 * N - 2, K - 2 * N) = A (JX + J + 2 * N - 2, K - 2 * N)
        + X (I, J) * A (IX + I + 2 * N - 2, K - 2 * N)
                            End If
                        Next J
                    Next I
                End If
            Next K

            JX = JX + J1 - 3
        Loop
        IX = IX + I1 - 3
    Loop
    End Sub
```

程式執行結果：

內部配置點數 N=2

```
** COLLOCATION POINTS:
  0.000000E+00        3.550510E-01        8.449490E-01        1.000000E+00

** MATRIX A:
 -5.00000E+00         7.53197E+00        -5.53197E+00         3.00000E+00
 -1.05320E+00        -7.75255E-01         3.56784E+00        -1.73939E+00
  2.53197E-01        -1.16784E+00        -3.22474E+00         4.13939E+00
 -3.33333E-01         1.38214E+00        -1.00488E+01         9.00000E+00

** LAPLACIAN MATRIX B:
  1.46667E+01        -3.28922E+01         4.22255E+01        -2.40000E+01
  7.56565E+00        -1.39024E+01         9.03367E+00        -2.69694E+00
 -2.23231E+00         1.22997E+01        -3.67643E+01         2.66969E+01
 -5.33333E+00         2.05925E+01        -5.12592E+01         3.60000E+01

** Radau Lobatto Qudratue Weight W:
  1.1111E-01          5.1249E-01          3.7640E-01          0.0000E+00

** MODIFIED QUDRATURE VECTOR W:
  6.7182E-01          1.6151E-01
```

** SOLUTION SERIES
0.80764799 * Exp (-5.135816E+00 * Z)
0.02568535 * Exp (-3.461530E+01 * Z)

Distance	Sherwood No.
1.000E-05	4.02945658E+00
2.000E-05	4.02928359E+00
5.000E-05	4.02876488E+00
1.000E-04	4.02790133E+00
2.000E-04	4.02617782E+00
5.000E-04	4.02103578E+00
1.000E-03	4.01256009E+00
2.000E-03	3.99595711E+00
5.000E-03	3.94882933E+00
1.000E-02	3.87854958E+00
2.000E-02	3.76447643E+00
5.000E-02	3.56597956E+00
1.000E-01	3.45660296E+00
2.000E-01	3.42559623E+00
5.000E-01	3.42387738E+00
1.000E+00	3.42387714E+00

內部配置點數 N=3

** COLLOCATION POINTS:
0.000000E+00 2.123405E-01 5.905331E-01 9.114120E-01 1.000000E+00

** MATRIX A:

-8.50000E+00	1.21717E+01	-6.59524E+00	6.92349E+00	-4.00000E+00
-1.82214E+00	-6.34792E-01	3.98452E+00	-3.49247E+00	1.96488E+00
4.34791E-01	-1.75468E+00	-1.22110E+00	5.04921E+00	-2.50823E+00
-1.73878E-01	5.85901E-01	-1.92351E+00	-5.64411E+00	7.15559E+00
2.50000E-01	-8.20328E-01	2.37791E+00	-1.78076E+01	1.60000E+01

** LAPLACIAN MATRIX B:

4.50000E+01	-9.22761E+01	8.97825E+01	-1.02506E+02	6.00000E+01
1.94758E+01	-3.24253E+01	1.60127E+01	-5.55775E+00	2.49458E+00
-2.53439E+00	1.35646E+01	-2.40446E+01	1.91400E+01	-6.12559E+00
2.34433E+00	-8.29001E+00	3.37020E+01	-1.08530E+02	8.07739E+01
7.50000E+00	-2.41675E+01	6.44785E+01	-1.67811E+02	1.20000E+02

** Radau Lobatto Qudratue Weight W:
6.2500E-02 3.2884E-01 3.8819E-01 2.2046E-01 0.0000E+00

** MODIFIED QUDRATURE VECTOR W:
4.7103E-01 3.7923E-01 5.5996E-02

** SOLUTION SERIES
0.00108049 * Exp (-1.369366E+02 * Z)
0.11620398 * Exp (-3.731356E+01 * Z)
0.78896553 * Exp (-5.122464E+00 * Z)

Distance	Sherwood No.
1.000E-05	6.27065036E+00
2.000E-05	6.26974934E+00
5.000E-05	6.26704835E+00
1.000E-04	6.26255359E+00
2.000E-04	6.25358985E+00
5.000E-04	6.22690269E+00
1.000E-03	6.18309145E+00
2.000E-03	6.09787921E+00
5.000E-03	5.85988577E+00
1.000E-02	5.51330842E+00
2.000E-02	4.96395285E+00
5.000E-02	4.02915150E+00
1.000E-01	3.54064146E+00
2.000E-01	3.42002977E+00
5.000E-01	3.41497624E+00
1.000E+00	3.41497592E+00

內部配置點數 N=5

** COLLOCATION POINTS:

0.000000E+00	9.853509E-02	3.045357E-01	5.620252E-01	8.019866E-01	9.601901E-01	1.000000E+00

** MATRIX A:

-1.85000E+01	2.56391E+01	-1.16387E+01	8.23274E+00	-7.67606E+00	9.94298E+00	-6.00000E+00
-4.01713E+00	-5.54653E-01	6.81054E+00	-3.95154E+00	3.46401E+00	-4.38579E+00	2.63457E+00
9.26443E-01	-3.46004E+00	-7.18944E-01	5.06988E+00	-3.49150E+00	4.10824E+00	-2.43407E+00
-3.84542E-01	1.17802E+00	-2.97498E+00	-1.14162E+00	5.54452E+00	-5.18216E+00	2.96075E+00
2.02548E-01	-5.83382E-01	1.15741E+00	-3.13223E+00	-2.52508E+00	9.80284E+00	-4.92211E+00
-1.09086E-01	3.07104E-01	-5.66232E-01	1.21720E+00	-4.07583E+00	-1.25597E+01	1.57865E+01
1.66667E-01	-4.67082E-01	8.49409E-01	-1.76076E+00	5.18155E+00	-3.99698E+01	3.60000E+01

** LAPLACIAN MATRIX B:

2.21667E+02	-4.28240E+02	3.54196E+02	-2.75315E+02	2.64872E+02	-3.47180E+02	2.10000E+02
8.59933E+01	-1.35506E+02	5.85666E+01	-1.26678E+01	6.00596E+00	-5.31473E+00	2.92254E+00
-7.41642E+00	3.85677E+01	-5.73481E+01	3.20894E+01	-9.01719E+00	6.62453E+00	-3.49992E+00
2.24642E+00	-7.77295E+00	2.99001E+01	-5.04879E+01	3.35523E+01	-1.41981E+01	6.76010E+00
-1.52801E+00	4.60480E+00	-1.04985E+01	4.19243E+01	-8.40662E+01	7.44210E+01	-2.48574E+01
2.96739E+00	-8.42710E+00	1.59506E+01	-3.66895E+01	1.53909E+02	-5.24259E+02	3.96549E+02
1.16667E+01	-3.25936E+01	5.87147E+01	-1.18734E+02	3.20736E+02	-8.69790E+02	6.30000E+02

** Radau Lobatto Qudratue Weight W:

2.7778E-02	1.5982E-01	2.4269E-01	2.6046E-01	2.0845E-01	1.0079E-01	0.0000E+00

** MODIFIED QUDRATURE VECTOR W:

2.3740E-01	3.3028E-01	2.6729E-01	1.1157E-01	1.1798E-02

** SOLUTION SERIES

0.00000460 * Exp (-1.242065E+03 * Z)
0.04028875 * Exp (-1.748120E+02 * Z)
0.03172882 * Exp (-1.134879E+02 * Z)
0.09660611 * Exp (-3.952764E+01 * Z)
0.78970505 * Exp (-5.121676E+00 * Z)

Distance	Sherwood No.
1.000E-05	1.28682650E+01
2.000E-05	1.28581210E+01
5.000E-05	1.28277658E+01
1.000E-04	1.27774287E+01
2.000E-04	1.26777017E+01
5.000E-04	1.23859284E+01
1.000E-03	1.19232914E+01
2.000E-03	1.10798876E+01
5.000E-03	9.09420153E+00
1.000E-02	7.04514453E+00
2.000E-02	5.22770164E+00
5.000E-02	3.91966000E+00
1.000E-01	3.50407539E+00
2.000E-01	3.41733190E+00
5.000E-01	3.41445086E+00
1.000E+00	3.41445076E+00

內部配置點數 N=10

** COLLOCATION POINTS:

0.000000E+00	3.002903E-02	9.828901E-02	1.990211E-01	3.240555E-01	4.632612E-01
6.053602E-01	7.388403E-01	8.528886E-01	9.382679E-01	9.880824E-01	1.000000E+00

** SOLUTION SERIES

0.00000000 * Exp (-4.216796E+04 * Z)
0.00000255 * Exp (-2.659171E+03 * Z)
0.02250040 * Exp (-1.574559E+03 * Z)
0.00036557 * Exp (-7.043263E+02 * Z)
0.01278584 * Exp (-4.957977E+02 * Z)
0.01014988 * Exp (-3.344785E+02 * Z)
0.01874881 * Exp (-2.048482E+02 * Z)
0.03609253 * Exp (-1.062492E+02 * Z)
0.09725511 * Exp (-3.966084E+01 * Z)
0.78970262 * Exp (-5.121669E+00 * Z)

Distance	Sherwood No.
1.000E-05	4.07918800E+01
2.000E-05	4.04092018E+01
5.000E-05	3.92921441E+01
1.000E-04	3.75293790E+01
2.000E-04	3.43455598E+01
5.000E-04	2.70226119E+01
1.000E-03	1.97616720E+01
2.000E-03	1.36273893E+01
5.000E-03	9.03272316E+00
1.000E-02	6.75628416E+00
2.000E-02	5.18498892E+00
5.000E-02	3.92710591E+00
1.000E-01	3.50389377E+00
2.000E-01	3.41728140E+00
5.000E-01	3.41444629E+00
1.000E+00	3.41444620E+00

內部配置點數 N=20

** COLLOCATION POINTS:

0.000000E+00	8.300044E-03	2.764305E-02	5.753446E-02	9.730413E-02
1.460632E-01	2.027225E-01	2.660161E-01	3.345303E-01	4.067345E-01
4.810157E-01	5.557147E-01	6.291628E-01	6.997193E-01	7.658081E-01
8.259529E-01	8.788101E-01	9.231992E-01	9.581286E-01	9.828188E-01
9.967239E-01	1.000000E+00			

** SOLUTION SERIES

0.00000000 * Exp (-1.975400E+06 * Z)
0.00000000 * Exp (-1.092267E+05 * Z)
0.00664539 * Exp (-2.000831E+04 * Z)
0.00000040 * Exp (-2.212363E+04 * Z)
0.00000023 * Exp (-7.676205E+03 * Z)
0.00611615 * Exp (-5.257843E+03 * Z)
0.00000472 * Exp (-3.653163E+03 * Z)
0.00479693 * Exp (-2.591069E+03 * Z)
0.00031312 * Exp (-2.199423E+03 * Z)
0.00247958 * Exp (-1.760718E+03 * Z)
0.00250609 * Exp (-1.472205E+03 * Z)
0.00324556 * Exp (-1.177541E+03 * Z)
0.00415710 * Exp (-9.193759E+02 * Z)
0.00551793 * Exp (-6.927254E+02 * Z)
0.00767597 * Exp (-4.980971E+02 * Z)
0.01140176 * Exp (-3.354732E+02 * Z)
0.01868637 * Exp (-2.048561E+02 * Z)
0.03609362 * Exp (-1.062492E+02 * Z)
0.09725511 * Exp (-3.966084E+01 * Z)
0.78970262 * Exp (-5.121669E+00 * Z)

Distance	Sherwood No.
1.000E-05	1.30464843E+02
2.000E-05	1.15993866E+02
5.000E-05	8.53555959E+01
1.000E-04	5.91404679E+01
2.000E-04	4.04373545E+01
5.000E-04	2.62594356E+01
1.000E-03	1.88365523E+01
2.000E-03	1.36350230E+01
5.000E-03	9.03973281E+00
1.000E-02	6.75455499E+00
2.000E-02	5.18508079E+00
5.000E-02	3.92710594E+00
1.000E-01	3.50389377E+00
2.000E-01	3.41728140E+00
5.000E-01	3.41444629E+00
1.000E+00	3.41444620E+00

參考文獻

Visual Basic

1. Ames, W. F., "Numerical Methods for Partial Differential Equations", 2nd Ed., Academic Press, New York, (1977) .
2. Bird, R. B., W. E. Stewart, and E. N. Lightfoot, "Transport Phenomena", Wiley, (1960)
3. Brian, P. L. T., "A Finite-Difference Method of Higher Order Accuracy for the Solution of Three-Dimensional Transient Heat Conduction Problems", AIChE J., Vol. 7, pp.367-370, (1961) .
4. Carnahan, B., H. A. Luther, and J. O. Wilkes, "Applied Numerical Methods", John Wiley, New York, (1998) .
5. Crank, J., "The Mathematics of Diffusion", Clarendon Press, (1957) .
6. Crank, J., and P. Nicolson, "A Practical Method for Numerical Evaluation of Solution of Partial Differential Equations of the Heat Conduction Type", Proc. Camb. Phil. Soc., Vol. 43, pp.50-67, (1947) .
7. Douglas, J., "Alternating Direction Method for Three Space Variables", Numerische Mathematik, Vol. 4, pp. 41-63, (1962) .
8. Dufort, E. C., and S. P. Frankel, "Stability Conditions in the Numerical Treatment of Parabolic Differential Equations", Math. Tables Aids Comput., Vol.7, pp. 135-152 (1953) .
9. Finlayson, B. A., "The Method of Weighted Residuals and Variational Principles", Academic Press, (1972) .
10. Saul'yev, V. K., "Integration of Equations of Parabolic Type by the Method of Nets", Macmillan, New York, (1964) .
11. Larkin, B. K., "Some Stable Explicit Difference Approximations to the Differential Equations", Math. Of Comp., Vol. 18, pp. 196-202, (1964) .
12. Villadsen, J., and M. L. Michelson,"Solution of Differential Equation Models by Polynomial Approximation", Prentice Hall, (1978) .

習題

1. 例 11-3 流體在垂直加熱平板邊的輸送現象，所使用分析是使用顯式差分法，因此，當所取的格子點數目增多或Δx 及Δy 變小時，$\Delta\tau$必須跟著調小，否則積分穩定性會受影響。

$$\frac{U^*_{i,j}-U_{i,j}}{\Delta\tau}+U_{i,j}\frac{U_{i,j}-U_{i-1,j}}{\Delta x}+V_{i,j}\frac{U_{i,j+1}-U_{i,j}}{\Delta y}$$
$$=T^*_{i,j}+\frac{U_{i,j+1}-2U_{i,j}+U_{i,j-1}}{(\Delta y)^2} \quad \textbf{(11-6.20)}$$

$$\frac{T^*_{i,j}-T_{i,j}}{\Delta\tau}+U_{i,j}\frac{T_{i,j}-T_{i-1,j}}{\Delta x}+V_{i,j}\frac{T_{i,j+1}-T_{i,j}}{\Delta y}=\frac{1}{\Pr}\frac{T_{i,j+1}-2T_{i,j}+T_{i,j-1}}{(\Delta y)^2} \quad \textbf{(11-6.21)}$$

$$\frac{U^*_{i,j}-U^*_{i-1,j}}{\Delta x}+\frac{V^*_{i,j}-V^*_{i,j-1}}{\Delta y}=0 \quad \textbf{(11-6.22)}$$

(a) 試執行 $\Delta x=0.5$, $\Delta y=2$ 且 $\Delta\tau=0.5$ 的情況，程式是否能穩定執行？

(b) 由於使用顯式差分法，計算穩定性必須加以考慮，觀察差分方程式，其中方程式 (11-6.22) 由於未含 $\Delta\tau$ 項目，不需考慮計算穩定性。假設 U 及 V 分別可以用 $\psi(\tau)\exp(i\alpha x)\exp(i\beta y)$ 及 $\zeta(\tau)\exp(i\alpha x)\exp(i\beta y)$ 表示，代回方程式 (11-6.20) 及方程式 (11-6.21)，試證明經簡化後可以得到

$$\begin{bmatrix}\psi'\\ \zeta'\end{bmatrix}=\begin{bmatrix}A & B\Delta\tau\\ 0 & B\end{bmatrix}\begin{bmatrix}\psi\\ \zeta\end{bmatrix}$$

$$A=1-\frac{U\Delta\tau}{\Delta X}(1-e^{i\alpha\Delta X})-\frac{V\Delta\tau}{\Delta Y}(e^{i\beta\Delta Y}-1)+\frac{2\Delta\tau}{(\Delta Y)^2}(Cos\beta\Delta Y-1)$$

$$B=1-\frac{U\Delta\tau}{\Delta X}(1-e^{i\alpha\Delta X})-\frac{V\Delta\tau}{\Delta Y}(e^{i\beta\Delta Y}-1)+\frac{2\Delta\tau}{\Pr(\Delta Y)^2}(Cos\beta\Delta Y-1)$$

(c) 由於所得到的矩陣微分方程式之特徵值爲 A 及 B，故其絕對值均應小於 1。試證明穩定之必要條件爲

$$\frac{U\Delta\tau}{\Delta X}+\frac{|V|\Delta\tau}{\Delta Y}+\frac{2\Delta\tau}{(\Delta Y)^2}\le 1$$

或

$$\frac{U\Delta\tau}{\Delta X}+\frac{|V|\Delta\tau}{\Delta Y}+\frac{2\Delta\tau}{\Pr\,(\Delta Y)^2}\le 1$$

2. 試利用隱式差分法重解例 11-3。

(a) 將差分方程式 (11-6.20) 至 (11-6.22)，改寫成下列形式

$$\frac{T^*_{i,j}-T_{i,j}}{\Delta\tau}+U_{i,j}\frac{T^*_{i,j}-T^*_{i-1,j}}{\Delta x}+V_{i,j}\frac{T^*_{i,j+1}-T^*_{i,j}}{\Delta y}=\frac{1}{\Pr}\frac{T^*_{i,j+1}-2T^*_{i,j}+T^*_{i,j-1}}{(\Delta y)^2}$$

$$\frac{U^*_{i,j}-U_{i,j}}{\Delta\tau}+U_{i,j}\frac{U^*_{i,j}-U^*_{i-1,j}}{\Delta x}+V_{i,j}\frac{U^*_{i,j+1}-U^*_{i,j}}{\Delta y}=T^*_{i,j}+\frac{U^*_{i,j+1}-2U^*_{i,j}+U^*_{i,j-1}}{(\Delta y)^2}$$

$$\frac{U^*_{i,j}-U^*_{i-1,j}}{\Delta x}+\frac{V^*_{i,j}-V^*_{i,j-1}}{\Delta y}=0$$

(b) 試將以上的差分方程式整理成三對角線方程式形式，並建立求解的策略。

(c) 利用上下分解法，解此問題。

3. 一半徑爲R的長金屬棒，原始溫度均勻且爲T_0，在時間t = 0 時放入一溫度爲T_F的定溫流體中。自時間t = 0 起，此金屬棒並通以電流A安培 / 平方公分，使金屬棒每單位體積產生$Q=A^2/Ke$的熱量，其中Ke與溫度無關，且其單位是$\text{Ohm}^{-1}\text{cm}^{-1}$。金屬表面與流體的熱傳係數爲$h$。

(a) 此金屬棒的溫度分布可以用以下的微分方程式表示

$$\frac{\partial T}{\partial\tau}=\frac{\alpha}{r}\frac{\partial}{\partial r}\left(r\frac{\partial T}{\partial r}\right)+\frac{A^2}{\rho C_p Ke}$$

$$T=T_0 \qquad t=0 \qquad 0\le r\le R$$

$$\frac{\partial T}{\partial r}=0 \qquad t>0 \qquad r=0$$

$$-k\frac{\partial T}{\partial r}=h(T-T_F) \qquad t>0 \qquad r=R$$

(b) 試證明引入無因次變數，可以將偏微分方程式簡化成以下形式。

$$\frac{\partial\theta}{\partial\tau}=\frac{1}{y}\frac{\partial}{\partial y}\left(y\frac{\partial\theta}{\partial y}\right)+1$$

$$\theta = 0 \qquad \tau = 0 \qquad 0 \le y \le 1$$

$$\frac{\partial \theta}{\partial y} = 0 \qquad \tau > 0 \qquad y = 0$$

$$\frac{\partial \theta}{\partial y} = -\beta(\theta - \theta_F) \qquad \tau > 0 \qquad y = 1$$

其中

$$y = \frac{r}{R} \qquad \tau = \frac{t}{\rho C_p R^2 / k} \qquad \theta = \frac{T - T_0}{A^2 R^2 / k\, Ke}$$

$$\beta = \frac{hR}{k} \qquad \theta_F = \frac{T_F - T_0}{A^2 R^2 / k\, Ke}$$

(c) 試利用隱式差分法建立所需的差分方程式，並求解之。應特別注意，當 y 趨近於零時，$\partial\theta / \partial y \to 0$，利用 L'Hospital 原理，微分方程式將變成 $\frac{\partial \theta}{\partial \tau} = 2\frac{\partial^2 \theta}{\partial y^2} + 1$。

(d) 建議測試數據爲

$$\Delta\tau = 0.005 \quad 0.01 \quad 0.02 \quad 0.05 \quad 0.1$$
$$\beta = 0 \quad 0.25 \quad 0.5 \quad 1$$
$$\theta_F = 0.5 \quad 1 \quad 2 \quad 5$$
$$\tau_{\max} = 2.0$$

4. 若要利用分離變數法解問題 3，必須將邊界條件轉化成均勻邊界條件。令 $\xi = \theta - \theta_F$，將問題 3 所得到的微分方程式簡化。然後利用分離變數法及特徵值，求問題 3 的解答。

5. 一黏滯流體以平均速度u_m流經半徑爲a的加熱管子，假設流體的熱滲透係數爲α，流體進入管子前的溫度爲θ_0，管壁溫度維持θ_1。r及z分別爲徑向及軸向座標，定義無因次變數及座標爲 $T = (\theta - \theta_0)/(\theta_1 - \theta_0)$, $R = r/a$, $Z = \alpha z/(2u_m a^2)$。又假設軸向熱傳導可以忽略，流體流動現象爲穩定的層流 (Laminar Flow)，且黏度隨溫度之變化可忽略。此系統可以利用以下的偏微分方程式描述之。

$$(1 - R^2)\frac{\partial T}{\partial Z} = \frac{\partial^2 T}{\partial R^2} + \frac{1}{R}\frac{\partial T}{\partial R}$$

IC	$T=0$	在 $Z=0$,	$0 \le R \le 1$
BC	$T=1$	在 $R=1$,	$Z>0$
	$\frac{\partial T}{\partial R}=0$	在 $R=0$,	$Z>0$

(a) 試利用隱式差分法寫一程式解此問題。

(b) 試利用數學配置法解此問題。

6. 在問題 5 中，若管壁並非維持在定溫，而是利用電熱方式，提供一定的熱通量 (Heat Flux)，q，試重複問題 5 解此問題。解此問題時，可以引進新的無因次溫度，定義爲 $T=(\theta-\theta_0)/(aqk)$ 。k 爲流體之熱傳導係數。

7. 一片長 15 公分，寬 5 公分的印刷電路板，上面有三個點及一小塊區域產生熱源，如圖 11.14 所示。

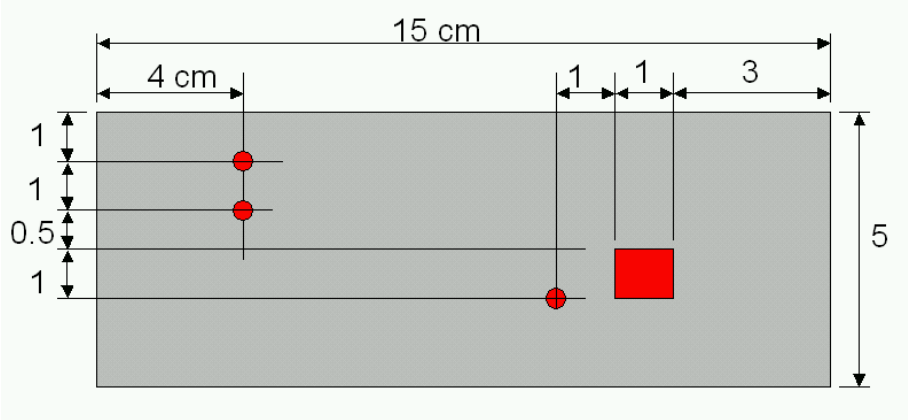


圖 11.14　積體電路板之熱傳分析

電路板的材質爲強化玻璃纖維 (FRP)，電熱源爲 AISI-304 鏍栓。方塊點熱源爲勘入式積體電路，其性質與 AISI-304 類似。

	密度 ρ gm/cm^3	熱容量 C_p J/g°C	熱傳導係數 k W/cm°C
FRP	1.93	1.432	0.00326
AISI-304	7.81	0.46	0.190

(a) 假設點熱源每一點產生 0.5 W 的熱量，塊狀熱源產生 10 W 熱量。週界溫度爲 30°C 。

(b) 試建立差分方程式，並計算溫度分布。

(c) 若 FRP 最高耐溫105°C ，是否有過熱現象，而會影響產品品質？

(d) 若加上強制對流散熱，熱傳係數爲 $3\ \text{W/cm}^2\text{°C}$ ，試重新計算溫度分布。

8. 在半徑爲 R 長度爲 L 的水平長管內，原爲靜止狀態的流體，在時間 $t = 0$ 時，承受一壓力梯度 $(P_0 - P_L)/L$ 。

(a) 描述流體運動的微分方程式爲

$$\rho\frac{\partial V}{\partial t} = \frac{P_0 - P_L}{L} + \frac{1}{r}\frac{\partial}{\partial r}(r\mu\frac{\partial V}{\partial r})$$

$$\begin{array}{lll} V = 0 & @\ t = 0 & 0 \le r \le R \\ \dfrac{\partial V}{\partial r} = 0 & @\ r = 0 & t \ge 0 \\ V = 0 & @\ r = R & t \ge 0 \end{array}$$

(b) 定義以下無因次參數

$$\tau = \frac{\mu t}{\rho R^2} \qquad \xi = \frac{r}{R} \qquad \eta = \frac{4\mu L V}{(P_0 - P_L)R^2}$$

試證明原微分方程式可以被簡化成

$$\frac{\partial \eta}{\partial \tau} = 4 + \frac{1}{\xi}\frac{\partial}{\partial \xi}(\xi\frac{\partial \eta}{\partial \xi})$$

$$\begin{array}{lll} \eta = 0 & \text{在 } \tau = 0 & 0 \le \xi \le 1 \\ \dfrac{\partial \eta}{\partial \xi} = 0 & \text{在 } \xi = 0 & \tau \ge 0 \\ \eta = 0 & \text{在 } \xi = 1 & \tau \ge 0 \end{array}$$

(c) 當 τ 趨近於無窮大時，系統會趨近於穩定狀態， η_∞ 。令

$$\eta(\tau, \xi) = \eta_\infty - \phi(\tau, \xi)$$

試證明 $\eta_\infty = 1 - \xi^2$ 。

(d) 試證明原偏微分方程式，可以被改寫成

$$\frac{\partial \phi}{\partial \tau} = \frac{1}{\xi}\frac{\partial}{\partial \xi}(\xi\frac{\partial \phi}{\partial \xi})$$

$$\phi = 1 - \xi^2 \qquad \text{在 } \tau = 0 \qquad 0 \le \xi \le 1$$

$$\frac{\partial \phi}{\partial \xi} = 0 \qquad \text{在 } \xi = 0 \qquad \tau \ge 0$$

$$\phi = 0 \qquad \text{在 } \xi = 1 \qquad \tau \ge 0$$

(e) 試將以上的偏微分方程式差分化，並計算當 $t = 0.1, 0.2, 0.5$ ，及無窮大的結果。

(f) 試利用正交配置法重解本問題。