Lecture 7

第6章 解線性方程組：迭代法

大綱

- Jacobi法
- Gauss-Seidel法
- 逐次過鬆弛法 (Successive Over-relaxation)
- 進階問題

簡介

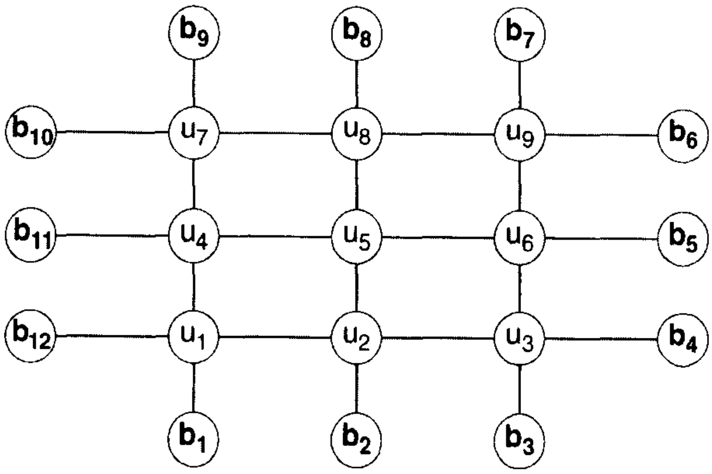
- 要求數值解的線性方程組通常都非常大，這使得一般直接法，如高斯消去法的計算成本高到無法負擔。對於係數矩陣具有適當結構的方程組-特別是大型、稀疏(Sparse)矩陣 (亦即方程組中有許多項的係數是零)，使用迭代法會比較合適。
- 介紹三種最常用於線性方程組的傳統迭代法:Jacobi法、Gauss-Seidel法和逐次過鬆弛法(SOR)法。

問題6-A 偏微分方程的有限差分解

- 在求偏微分方程的數值解時需求解一個線性方程組，而它的係數矩陣有許多元素為零，且不為零的元素成帶狀結構。

問題6-A 偏微分方程的有限差分解

- 在以數值方法求解二維勢能方程式(Potential equation)的時候，可以用網格點定義出要求解的區域，再以差分式近似微分項。(有限差分法)



問題6-A 偏微分方程的有限差分解

- 勢能方程式(Potential equation)

$$u_{xx} + u_{yy} = 0, \quad 0 \le x \le 1, \quad 0 \le y \le 1$$

時，已知邊界條件

$$u(0, y) = y_2, \quad u(1, y) = 1, \quad 0 < y < 1$$
$$u(x, 0) = x_2, \quad u(x, 1) = 1, \quad 0 < x < 1$$

及網格點 $\Delta x = \Delta y = 0.2$，我們可得以下線性方程組：

$$\begin{array}{cccccccccccc}
4u_1 & -u_2 & & -u_4 & & & & & & = & b_1 + b_{12} \\
-u_1 & +4u_2 & -u_3 & & -u_5 & & & & & = & b_2 \\
& -u_2 & +4u_3 & & & -u_6 & & & & = & b_3 + b_4 \\
-u_1 & & & +4u_4 & -u_5 & & -u_7 & & & = & b_{11} \\
& -u_2 & & -u_4 & +4u_5 & -u_6 & & -u_8 & & = & 0 \\
& & -u_3 & & -u_5 & +4u_6 & & & -u_9 & = & b_5 \\
& & & -u_4 & & & +4u_7 & -u_8 & & = & b_9 + b_{10} \\
& & & & -u_5 & & -u_7 & +4u_8 & -u_9 & = & b_8 \\
& & & & & -u_6 & & -u_8 & -u_9 & = & b_6 + b_7
\end{array}$$

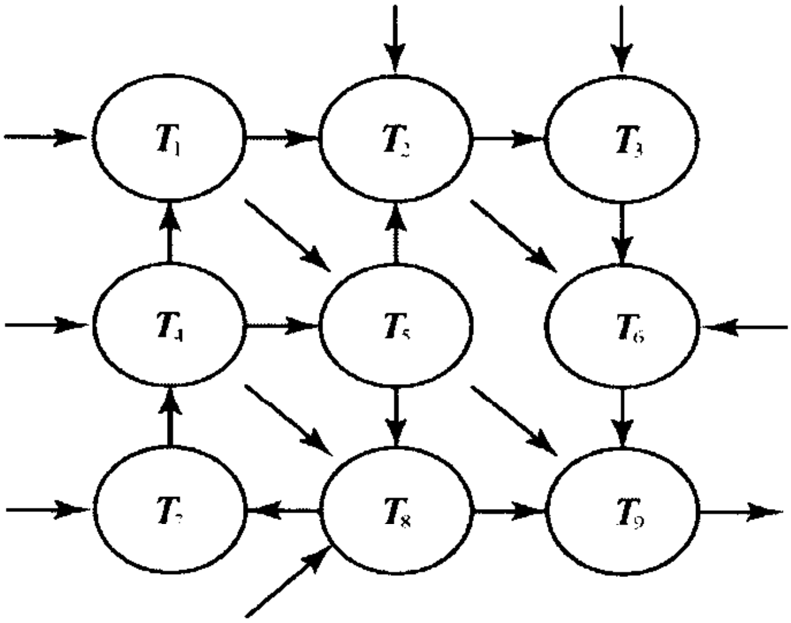
問題6-A 偏微分方程的有限差分解

- 此方程組非常適合用迭代法求解。在實用上，線性方程組通常非常大，我們並不會真的去構建出整個係數矩陣。

問題6-B 穩態濃度

- 對一個化工系統，每一個槽的特性取決於流入與流出率。假設在每一個槽內，化學物都是混合均勻，所以槽內化學濃度是均一的。
- 由質量平衡方程式可得，每一個槽化學物的流入率應等於流出率。化學物的傳輸速率為，流束中化學物濃度(質量/體積)和流速(體積/時間)的乘積。

問題6-B 穩態濃度



問題6-B 穩態濃度

對圖示的系統，質量平衡方程式爲

$$r_{01}c_{01} + r_{41}c_4 = (r_{12} + r_{15})c_1$$

$$r_{02}c_{02} + r_{12}c_1 + r_{52}c_5 = (r_{23} + r_{26})c_2$$

$$r_{03}c_{03} + r_{23}c_2 = r_{36}c_3$$

$$r_{04}c_{04} + r_{74}c_7 = (r_{41} + r_{48} + r_{45})c_4$$

$$r_{15}c_1 + r_{45}c_4 = (r_{52} + r_{58} + r_{59})c_5$$

問題6-B 穩態濃度

$$r_{06}c_{06} + r_{26}c_2 + r_{36}c_3 = r_{69}c_6$$

$$r_{07}c_{07} + r_{87}c_8 = r_{74}c_7$$

$$r_{08}c_{08} + r_{58}c_5 + r_{48}c_4 = (r_{87} + r_{89})c_8$$

$$r_{59}c_5 + r_{69}c_6 + r_{89}c_8 = rc_9$$

- 若已知流率及流入濃度，可解出每個槽的濃度。

迭代法概述

- 解線性方程組的傳統迭代方式是將方程組 **Ax=b** 轉換成同義方程組 **x=Cx+d**(迭代方程組)，並產生一系列的近似解 $\mathbf{x}^{(1)},\mathbf{x}^{(2)},\ldots$，其中

$$\mathbf{x}^{(k)} = \mathbf{C}\mathbf{x}^{(k-1)} + \mathbf{d}$$

- 此種做法類似於第1章用於單變數非線性函數的固定點迭代。
- 在本章中考慮三種常用來解線性方程組的迭代法：**Jacobi**、**Gauss-Seidel**和**SOR**法。

迭代法概述

- 基本觀念是，解第i個方程式的第i個變數。線性方程組如下：

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 = b_1$$

$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + a_{24}x_4 = b_2$$

$$a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + a_{34}x_4 = b_3$$

$$a_{41}x_1 + a_{42}x_2 + a_{43}x_3 + a_{44}x_4 = b_4$$

迭代法概述

- 將方程組轉換成：$\mathbf{x}^{(k)}=\mathbf{C}\mathbf{x}^{(k-1)}+\mathbf{d}$，即

$$x_1 = -\frac{a_{12}}{a_{11}}x_2 - \frac{a_{13}}{a_{11}}x_3 - \frac{a_{14}}{a_{11}}x_4 + \frac{b_1}{a_{11}}$$

$$x_2 = -\frac{a_{21}}{a_{22}}x_1 - \frac{a_{23}}{a_{22}}x_3 - \frac{a_{24}}{a_{22}}x_4 + \frac{b_2}{a_{22}}$$

$$x_3 = -\frac{a_{31}}{a_{33}}x_1 - \frac{a_{32}}{a_{33}}x_3 - \frac{a_{34}}{a_{33}}x_4 + \frac{b_3}{a_{33}}$$

$$x_4 = -\frac{a_{41}}{a_{44}}x_1 - \frac{a_{42}}{a_{44}}x_2 - \frac{a_{43}}{a_{44}}x_3 + \frac{b_4}{a_{44}}$$

迭代法概述

- **Jacobi法和Gauss-Seidel法的差異在於，它們更新等號右側變數值的方式。**
- **SOR法的更新則是以Gauss-Seidel的更新值與前次之解向量的凸組合 (convex combination)。**

在實數空間中，當有限點$x_1, x_2, \cdots, x_n$，則其線性組合

$\alpha_1 x_1 + \alpha_2 x_2 + \cdots + \alpha_n x_n$必存在於這些已知點的凸殼(Convex Hull)內。

其中

凸組合係數$\alpha_i \geq 0$且$\alpha_1 + \alpha_2 + \cdots + \alpha_n = 1$

迭代停止條件

- **有兩種方式可用來停止迭代，**
  - ➢ ***兩次迭代間，解向量x之變量的範數足夠小。***
  - ➢ ***殘值向量的範數‖Ax-b‖小於指定的容許誤差。***

Jacobi迭代法

- Jacobi法基本上是將線性方程組Ax=b轉換成方程組x=Cx+d，其中C的對角線為零。在計算右側項的值時，其中的向量X是用前一次估計之x的所有分量一次更新。

$$x^{(k)} = x^{(k-1)} + d$$

Jacobi迭代演算法

- Step 1 Set k=1
- Step 2 While k≦max_it do Steps 3-6
- Step 3 For i=1:n

$$set\ \ x_i = \frac{-\sum_{\substack{j=1 \\ j\neq i}}^{n}(a_{ij}x_{0j}) + b_i}{a_{ii}}$$

- Step 4 If ∥ $x\text{-}x_0$ ∥ < Tol then OUTPUT (x1,x2,…,xn)
- STOP
- Step 5 Set k=k+1
- Step 6 For i=1:n set $x_{0i}=x_i$
- Step 7 OUTPUT (‘results after maximum number of iterations’)
- STOP

Jacobi迭代演算法

- 另一個程式終止條件：
- Step 4 If $\| x^{(k)} - x^{(k)} \| / \| x^{(k)} \| < Tol$
- then OUTPUT (x1,x2,…,xn)
- STOP

例題6.1 Jacobi迭代法說明

- 解方程組：

$$\begin{cases} 2x + y = 6 \\ x + 2y = 6 \end{cases} \Rightarrow \begin{cases} x = -0.5y + 3 \\ y = -0.5x + 3 \end{cases}$$

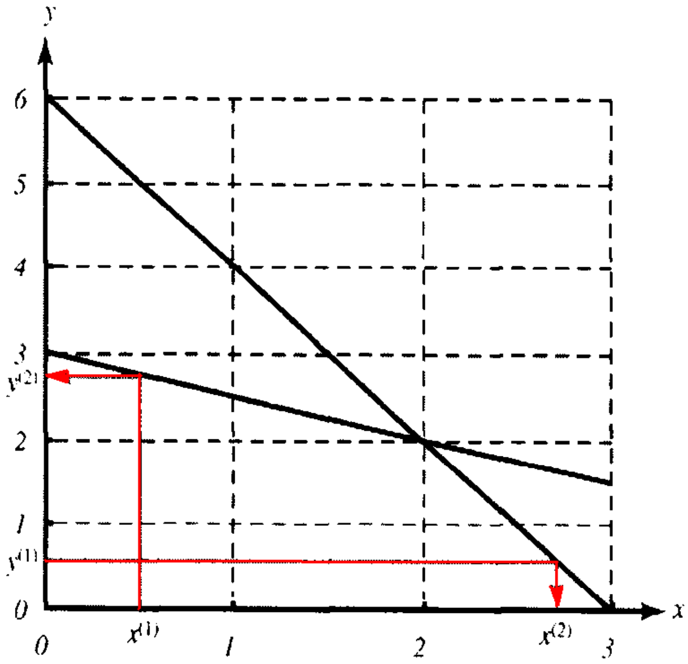
- 使用Jacobi

$$\begin{cases} x^{(2)} = -0.5y^{(1)} + 3 \\ y^{(2)} = -0.5x^{(1)} + 3 \end{cases}$$

例題6.1 Jacobi迭代法說明

- 初始值： $x^{(1)}=1/2, y^{(1)}=1/2$

$$\begin{cases} x^{(2)}=-0.5y^{(1)}+3=-0.25+3=2.75 \\ y^{(2)}=-0.5x^{(1)}+3=-0.25+3=2.75 \end{cases}$$



例題 6.2 Jacobi迭代用於小型方程組

- 解方程組：

$$\begin{cases} 2x_1-x_2+x_3=-1 \\ x_1+2x_2-x_3=6 \\ x_1-x_2+2x_3=-3 \end{cases}$$

轉成迭代方程組

$$\begin{cases} x_1=0.5x_2-0.5x_3-0.5 \\ x_2=-0.5x_1+0.5x_3+3 \\ x_3=-0.5x_1+0.5x_2-1.5 \end{cases}$$

例題 6.2 Jacobi迭代用於小型方程組

- 用矩陣表示

$$\begin{bmatrix} 2 & -1 & 1 \\ 1 & 2 & -1 \\ 1 & -1 & 2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -1 \\ 6 \\ -3 \end{bmatrix}$$

轉成迭代方程組

$$\begin{bmatrix} x_1^{(k)} \\ x_2^{(k)} \\ x_3^{(k)} \end{bmatrix} = \begin{bmatrix} 0 & 0.5 & -0.5 \\ -0.5 & 0 & 0.5 \\ -0.5 & 0.5 & 0 \end{bmatrix}\begin{bmatrix} x_1^{(k-1)} \\ x_2^{(k-1)} \\ x_3^{(k-1)} \end{bmatrix} + \begin{bmatrix} -0.5 \\ 3 \\ -1.5 \end{bmatrix}$$

例題 6.2 Jacobi迭代用於小型方程組

由 $x^{(0)} = (0,0,0)^T$ 開始，我們得到

$$\begin{bmatrix} x_1^{(1)} \\ x_2^{(1)} \\ x_3^{(1)} \end{bmatrix} = \begin{bmatrix} 0.0 & 0.5 & -0.5 \\ -0.5 & 0.0 & 0.5 \\ -0.5 & 0.5 & 0.0 \end{bmatrix}\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -0.5 \\ 3.0 \\ -1.5 \end{bmatrix} = \begin{bmatrix} -0.5 \\ 3.0 \\ -1.5 \end{bmatrix}$$

第二次迭代，我們得

$$\begin{bmatrix} x_1^{(2)} \\ x_2^{(2)} \\ x_3^{(2)} \end{bmatrix} = \begin{bmatrix} 0.0 & 0.5 & -0.5 \\ -0.5 & 0.0 & 0.5 \\ -0.5 & 0.5 & 0.0 \end{bmatrix}\begin{bmatrix} -0.5 \\ 3.0 \\ -1.5 \end{bmatrix} + \begin{bmatrix} -0.5 \\ 3.0 \\ -1.5 \end{bmatrix} = \begin{bmatrix} 1.75 \\ 2.50 \\ 0.25 \end{bmatrix}$$

Jacobi 法在 13 次迭代後收斂到向量

$$\mathbf{x} = \begin{bmatrix} 1.0002 & 2.0001 & -0.9997 \end{bmatrix}^T$$

例題 6.2 Jacobi迭代用於小型方程組

- 所用的停止條件為，兩次迭代之解向量之差值的歐幾里得範數小於0.001。
- 實際的真解應為

$$X = \begin{bmatrix} 1 \\ 2 \\ -1 \end{bmatrix}$$

例題 6.2 Matlab程式

- 主程式： Main_Jacobi.m
- Jacobi程式：Jacobi.m

```
clear all;
close all;

A=[2 -1 1;1 2 -1;1 -1 2];
b=[-1;6;-3];
x0=[0 0 0]';
tol=0.001;
max_it=30;

tic;
x=Jacobi(A, b, x0, tol, max_it);
Jacobi_Time=toc;
fprintf('Solution: \n');
fprintf('x=[');
fprintf('\n  %6.4f',x);
fprintf(' ];');
fprintf('\nRun Time for Jacobi iteration =%6.4f (sec)\n\n',Jacobi_Time);
```

例題 6.2 Matlab程式

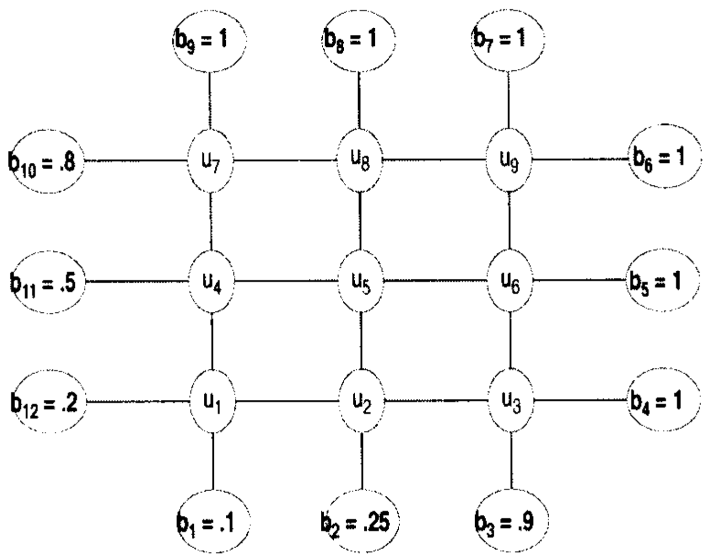
- 執行結果：

```
      i          x1          x2          x3
   11.0000     0.9985     2.0010     -1.0005
   12.0000     1.0007     2.0005     -0.9988
   13.0000     0.9996     2.0002     -1.0001

Jacobi method converged
Solution:
x=[
  1.0002
  2.0001
  -0.9997 ];
Run Time for Jacobi iteration =0.0630 (sec)
```


例題6.3 解PDE的線性方程組

- 使用有限差分法解偏微分方程(PDE)時會出現線性方程組。考慮應用問題6-A中的線性方程組。



例題6.3 解PDE的線性方程組

- 任一點的值乘四，減去相鄰四點的值 (該點的上、下、左及右)等於零，如下所示。

$$
\begin{aligned}
4u_1 - u_2 - u_4 - b_{12} - b_1 &= 0 \\
4u_2 - u_3 - u_5 - u_1 - b_2 &= 0 \\
4u_3 - b_4 - u_6 - u_2 - b_3 &= 0 \\
4u_4 - u_5 - u_7 - b_{11} - u_1 &= 0 \\
4u_5 - u_6 - u_8 - u_4 - u_2 &= 0 \\
4u_6 - b_5 - u_9 - u_5 - u_3 &= 0 \\
4u_7 - u_8 - b_9 - b_{10} - u_4 &= 0 \\
4u_8 - u_9 - b_8 - u_7 - u_5 &= 0 \\
4u_9 - b_6 - b_7 - u_8 - u_6 &= 0
\end{aligned}
$$

例題6.3 解PDE的線性方程組

- 將已知數移到右側，並依未知數排列

$$
\begin{array}{ccccccccccc}
4u_1 & -u_2 & & -u_4 & & & & & & = & 0.3 \\
-u_1 & +4u_2 & -u_3 & & -u_5 & & & & & = & 0.25 \\
 & -u_2 & +4u_3 & & & -u_6 & & & & = & 1.9 \\
-u_1 & & & +4u_4 & -u_5 & & -u_7 & & & = & 0.5 \\
 & -u_2 & & -u_4 & +4u_5 & -u_6 & & -u_8 & & = & 0 \\
 & & -u_3 & & -u_5 & +4u_6 & & & -u_9 & = & 1.0 \\
 & & & -u_4 & & & +4u_7 & -u_8 & & = & 1.8 \\
 & & & & -u_5 & & -u_7 & +4u_8 & -u_9 & = & 1.0 \\
 & & & & & -u_6 & & -u_8 & +4u_9 & = & 2.0
\end{array}
$$

例題6.3 解PDE的線性方程組

- 使用矩陣型式Ax=b，其中

$$A=\begin{bmatrix} 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 4 & 0 & 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 4 & -1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 & 4 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 4 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 \end{bmatrix}, b=\begin{bmatrix} 0.3 \\ 0.25 \\ 1.9 \\ 0.5 \\ 0 \\ 1.0 \\ 1.8 \\ 1.0 \\ 2.0 \end{bmatrix}$$

例題6.3 解PDE的線性方程組

- 使用一個 Jacobi法的電腦程式 (並設初始值全為零)，在十次迭代後我們得到的解向量 (顯示到小數四位)為

$$x^T=[0.3481 \quad 0.5218 \quad 0.8160 \quad 0.5816 \quad 0.6963 \quad 0.8530 \quad 0.8053 \quad 0.8503 \quad 0.9231]$$

- 誤差度量是使用殘值向量$r=x^{(k)}-x^{(k-1)}$的歐幾里得範數；十次迭代後$\|r\|=2.7930e-002$。
- 將這些值用解PDE的二維網格來表示

	1.0	1.0	1.0	
0.8	0.8053	0.8503	0.9231	1.0
0.5	0.5816	0.6963	0.8530	1.0
0.2	0.3481	0.5218	0.8160	1.0
	0.1	0.25	0.9	

例題6.3 解PDE的線性方程組

- 20次迭代後，其解為

$$x^T = [0.3590 \quad 0.5383 \quad 0.8269 \quad 0.5981 \quad 0.7180 \quad 0.8696 \quad 0.8162 \quad 0.8669 \quad 0.9340]$$

- 以網格型式表示

	1.0	1.0	1.0	
0.8	0.8162	0.8669	0.9340	1.0
0.5	0.5981	0.7180	0.8696	1.0
0.2	0.3590	0.5383	0.8269	1.0
	0.1	0.25	0.9	

- 其殘值範數 ‖r‖ =8.7282e-004。

例題6.3 解PDE的線性方程組

- 在30次迭代之後，近似解與確切解吻合到小數三位(多數分量到小數四位)。
- 在40次迭代後，近似解與確切解一致(到小數四位)。

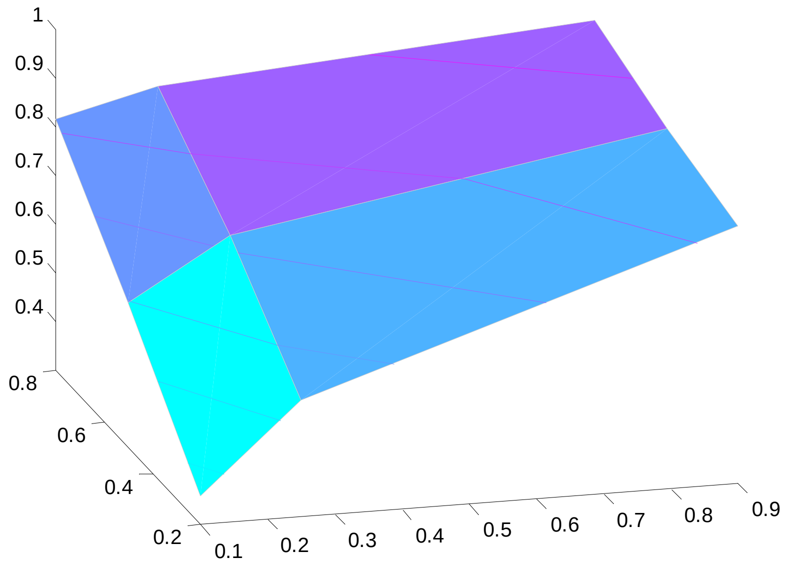
$$y^T = [0.3594 \quad 0.5388 \quad 0.8272 \quad 0.5987 \quad 0.7188 \quad 0.8701 \quad 0.8165 \quad 0.8674 \quad 0.9344]$$

- 殘值r=My-b的範數 ‖r‖ =8.5237e-007。
- 以網格型式表示

	1.0	1.0	1.0	
0.8	0.8165	0.8674	0.9344	1.0
0.5	0.5987	0.7188	0.8701	1.0
0.2	0.3594	0.5388	0.8272	1.0
	0.1	0.25	0.9	

例題6.3 Matlab程式

- **主程式： Main_Jacobi_Ex63.m**
- **Jacobi程式：Jacobi_1.m**



Jacobi法之特性分析

- **Jacobi是將原方程組Ax=b改寫成分解的型式(L+D+U)x=b，其中D為對角線矩陣，L是下三角矩陣，而U為上三角矩陣。此一分解型式可依以下方式表示為x=Cx+d:**

$$Dx = (-L-U)x + b$$

$$x = D^{-1}(-L-U)x + D^{-1}b$$

$$x = Cx + d$$

迭代式：

$$x^{(k)} = D^{-1}(-L-U)x^{(k-1)} + D^{-1}b$$

- **此公式假設，A的對角線元素都不為零。如果矩陣A為非奇異，但對角線上有元素為零，則可經由調換列或行，以得到非奇異的D。而且是對角線元素大於非對角線元素(稱為對角線優勢)。**

Jacobi法之特性分析

$$A=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$$

$$=\begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & a_{nn} \end{bmatrix}+\begin{bmatrix} 0 & \cdots & \cdots & 0 \\ a_{21} & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ a_{n1} & \cdots & a_{n,n-1} & 0 \end{bmatrix}+\begin{bmatrix} 0 & a_{12} & \vdots & a_{1n} \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & a_{n-1,n} \\ 0 & \cdots & \cdots & 0 \end{bmatrix}$$

$$=D+L+U$$

Jacobi法收斂的理論

- Jacobi迭代可以收斂到方程組Ax=b之解的充分條件(Sufficient Conditions)為，原來的矩陣A是嚴格的對角線優勢 (Strictly diagonally dominant)。這也就是說，在每一列中，對角線元素的絕對值，大於同列其它元素之絕對值的和。
- Jacobi法收斂的充分且必要條件(Necessary and Sufficient Conditions)為，迭代矩陣C的最大特徵值小於1。特徵值和特徵向量已在第5章中討論過。用MATLAB的函數求例題6.4之迭代矩陣C的特徵值，得3.4641和-3.4641。很明顯的，此問題不適合使用Jacobi迭代。原來排列順序下的迭代矩陣的特徵值為0.28868和-0.28868。當然，通常迭代法不會用於這麼小的方程組，在此只是藉此指出，對這麼簡單的問題，此方法不會收斂。

例題6.4 Jacobi法對方程式的順序敏感

- 考慮本節一開始的例子，但是將方程式的順序對調，所以方程組不再是對角線優勢：

$$\begin{cases} x+2y=6 \\ 2x+y=6 \end{cases} \qquad \begin{cases} x=-2y+6 \\ y=-2x+6 \end{cases}$$

$$\begin{cases} x^{(2)}=-2y^{(1)}+6=5 \\ y^{(2)}=-2x^{(1)}+6=5 \end{cases}$$

例題6.4 Jacobi法對方程式的順序敏感

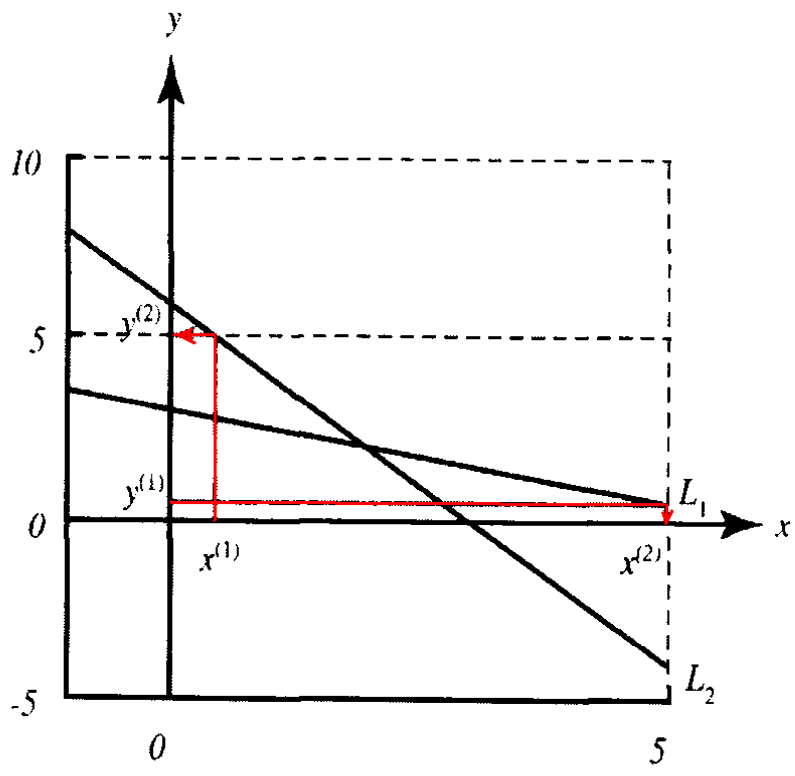


圖 6.2 發散的 Jacobi 迭代

Jacobi法計算量和其它考慮事項

- Jacobi法的每次迭代需要一個矩陣-向量乘法，或$(n-1)^2$個純量乘法。因此，若此方法在合理的迭代次數內收斂，它的計算量將遠低於高斯消去法所需的$O(n^3)$次乘法。
- Jacobi法特別適合平行計算，因為解答的每一個分量都可單獨的更新。

6.2 Gauss-Seidel法

- 解線性方程組的Gauss-Seidel法是一種簡單迭代法，它將線性方程組Ax=b轉換成方程組x=Cx+d，其中矩陣C的對角線元素為零。但是和Jacobi法不一樣的是，轉換後方程式等號右側向量x的各個分量，隨著迭代的進行，每產生一個值就立即更新一個分量。此一程序叫做循序更新。

Gauss-Seidel迭代演算法

- Step 1 Set k=1
- Step 2 While k≦max_it do Steps 3-6
- Step 3 For i=1:n

$$set\ \ x_i = \frac{-\sum_{j=1}^{i-1}(a_{ij}x_j) - \sum_{j=i+1}^{n}(a_{ij}x_{0j}) + b_i}{a_{ii}}$$

- Step 4 If $\| x-x_0 \| <$ Tol then OUTPUT (x1,x2,…,xn)
- STOP
- Step 5 Set k=k+1
- Step 6 For i=1:n set $x_{0i}=x_i$
- Step 7 OUTPUT ('results after maximum number of iterations')
- STOP

Gauss-Seidel迭代演算法

- 另一個程式終止條件：
- Step 4 If $\| x^{(k)}-x^{(k)} \| / \| x^{(k)} \| <$ Tol
- then OUTPUT (x1,x2,…,xn)
- STOP

例題6.5 Gauss-Seidel迭代法說明

- 解方程組：

$$\begin{cases} 2x+y=6 \\ x+2y=6 \end{cases} \Rightarrow \begin{cases} x=-0.5y+3 \\ y=-0.5x+3 \end{cases}$$

- 使用Gauss-Seidel

$$\begin{cases} x^{(2)}=-0.5y^{(1)}+3 \\ y^{(2)}=-0.5x^{(2)}+3 \end{cases}$$

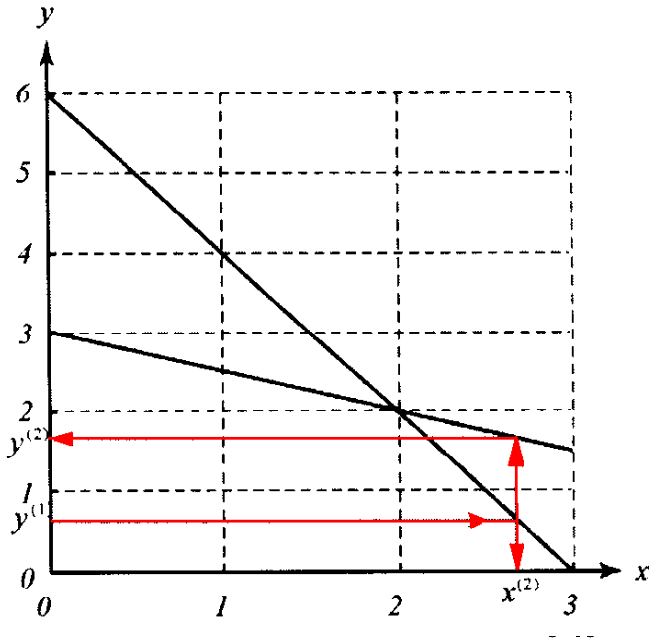
例題6.5 Gauss-Seidel迭代法說明

- 初始值： $x^{(1)}=1/2, y^{(1)}=1/2$

$$\begin{cases} x^{(2)}=-0.5y^{(1)}+3=-0.25+3=2.75 \\ y^{(2)}=-0.5x^{(2)}+3=-1.375+3=1.625 \end{cases}$$

下一次迭代

$$\begin{cases} x^{(3)}=-0.5y^{(2)}+3=2.1875 \\ y^{(3)}=-0.5x^{(3)}+3=1.9062 \end{cases}$$



例題 6.6 Gauss-Seidel迭代法用於小型方程組

• 解方程組：

$$\begin{cases} 2x_1 - x_2 + x_3 = -1 \\ x_1 + 2x_2 - x_3 = 6 \\ x_1 - x_2 + 2x_3 = -3 \end{cases}$$

轉成迭代方程組

$$\begin{cases} x_1 = 0.5x_2 - 0.5x_3 - 0.5 \\ x_2 = -0.5x_1 + 0.5x_3 + 3 \\ x_3 = -0.5x_1 + 0.5x_2 - 1.5 \end{cases}$$

例題 6.6 Gauss-Seidel迭代法用於小型方程組

• 將其轉換成迭代方程組

$$\begin{cases} x_1^{(new)} = 0.5x_2^{(old)} - 0.5x_3^{(old)} - 0.5 \\ x_2^{(new)} = -0.5x_1^{(new)} + 0.5x_3^{(old)} + 3 \\ x_3^{(new)} = -0.5x_1^{(new)} + 0.5x_2^{(new)} - 1.5 \end{cases}$$

例題 6.6 Gauss-Seidel迭代法用於小型方程組

以 $x^{(0)} = (0,0,0)$爲起始，我們得

$$x_1^{(1)} = \qquad +0.5(0) - 0.5(0) - 0.5 = -0.5$$
$$x_2^{(1)} = -0.5(-0.5) \qquad +0.5(0) + 3.0 = +3.25$$
$$x_3^{(1)} = -0.5(-0.5) + 0.5(3.25) \qquad -1.5 = +0.375$$

第二次迭代，

$$x_1^{(2)} = \qquad +0.5(3.25) - 0.5(0.375) - 0.5 = +0.9375$$
$$x_2^{(2)} = -0.5(0.9375) \qquad +0.5(0.375) + 3.0 = +2.7188$$
$$x_3^{(2)} = -0.5(0.9375) + 0.5(2.7188) \qquad -1.5 = -0.6094$$

例題 6.6 Gauss-Seidel迭代法用於小型方程組

利用以下所列的 MATLAB 函數，前幾次迭代的結果爲

i	x1	x2	x3
1	-0.5000	3.2500	0.3750
2	0.9375	2.7188	-0.6094
3	1.1641	2.1133	-1.0254
4	1.0693	1.9526	-1.0583
5	1.0055	1.9681	-1.0187
6	0.9934	1.9939	-0.9997
7	0.9968	2.0017	-0.9976
8	0.9996	2.0014	-0.9991
9	1.0003	2.0003	-1.0000
10	1.0001	1.9999	-1.0001

Gauss-Seidel 十次迭代收斂到向量

$$\mathbf{x} = \begin{bmatrix} 1.0001 & 1.9999 & -1.0001 \end{bmatrix}^T$$

例題 6.6 Matlab程式

- 主程式： Main_GaussSeidel.m
- Gauss-Seidel程式：GaussSeidel.m

```
clear all;
close all;

A=[2 -1 1;1 2 -1;1 -1 2];
b=[-1;6;-3];
x0=[0 0 0]';
tol=0.001;
max_it=30;

tic;
x=GaussSeidel(A, b, x0, tol, max_it);
GaussSeidel_Time=toc;
fprintf('Solution: \n');
fprintf('x=[');
fprintf('\n  %6.4f',x);
fprintf(' ];');
fprintf('\nRun Time for Jacobi iteration =%6.4f (sec)\n\n',GaussSeidel_Time);
```


例題 6.6 Matlab程式

- 結果：

```
      i          x1         x2          x3
   1.0000    -0.5000     3.2500     0.3750
   2.0000     0.9375     2.7188    -0.6094
   3.0000     1.1641     2.1133    -1.0254
   4.0000     1.0693     1.9526    -1.0583
   5.0000     1.0055     1.9681    -1.0187
   6.0000     0.9934     1.9939    -0.9997
   7.0000     0.9968     2.0017    -0.9976
   8.0000     0.9996     2.0014    -0.9991
   9.0000     1.0003     2.0003    -1.0000
  10.0000     1.0001     1.9999    -1.0001
Gauss-Seidel method converged
Solution:
x=[
  1.0001
  1.9999
  -1.0001 ];
Run Time for Gauss-Seidel iteration =0.1570 (sec)
```

Gauss-Seidel法之特性分析

- Gauss-Seidel是將原方程組Ax=b改寫成分解的型式(L+D+U)x=b，其中D為對角線矩陣，L是下三角矩陣，而U為上三角矩陣。此一分解型式可依以下方式表示為x=Tx+d：

$$(D+L)x=-Ux+b$$

$$x=-(D+L)^{-1}Ux+(D+L)^{-1}b$$

$$x=Tx+d$$

- Gauss-Seidel法和Jacobi法一樣，GauSs-Seidel法也對係數矩陣A(或是迭代矩陣T)的形式敏感；將例題6.5中方程式的順序對調，會和Jacobi法一樣得到發散的結果。

Gauss-Seidel法之收斂性

- 對一個實數且對稱的矩陣A，若且唯若A的所有特徵值均為實數且為正值，則Gauss-Seidel法會收斂(對任何$x^{(0)}$)。在以數值方法解偏微分方程時出現的線性方程組，它們的係數矩陣就具有此種特性。對於不滿足對稱條件的情形，理論分析更為困難。
- 如果A為對稱且正定，對任何初始向量Gauss-Seidel迭代都收斂。

Jacobi法和Gauss-Seidel法的關係

- 如果迭代矩陣C是非負的(解偏微分方程的數值方法經常如此)，則Jacobi法和Gauss-Seidel法要不是同樣收斂就是同樣發散；當它們都收斂時，Gauss-Seidel迭代法收斂得較快(除了兩個迭代矩陣的最大特徵值都是零的情況)。雖然Gauss-Seidel法有收斂較快的優點，但Jacobi較適用於平行計算。

6.3 逐次過鬆弛法 (Successive Over-Relaxation)

- 經由引入一個額外的參數ω(Omega)，可修改Gauss-Seidel法以加速其收斂。它的基本觀念是將前一次的x值和新的值(由Gauss-Seidel法求得的)做結合。參數ω用來控制在更新的值中，其中有多少比例來自前一次的解，有多少比例來自這一次的計算。
- 當 $0<\omega<1$時，此方法叫做逐次欠鬆弛 (Successive under-relaxation);
- 當$1<\omega<2$時，此方法叫做逐次過鬆弛(Successive over-relaxation, SOR)。

SOR迭代演算法

- Step 1 Set k=1
- Step 2 While k≦max_it do Steps 3-6
- Step 3 For i=1:n

$$set\ \ x_i = (1-\omega)x_{0j} + \frac{\omega(-\sum_{j=1}^{i-1}(a_{ij}x_j) - \sum_{j=i+1}^{n}(a_{ij}x_{0j}) + b_i)}{a_{ii}}$$

- Step 4 If ‖x-x_0‖ < Tol then OUTPUT (x1,x2,…,xn)
- STOP
- Step 5 Set k=k+1
- Step 6 For i=1:n set $x_{0i}=x_i$
- Step 7 OUTPUT (‘results after maximum number of iterations’)
- STOP

6.3 逐次過鬆弛法(SOR法)

- 考慮以下3x3方程組

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1$$
$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2$$
$$a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3$$

- 此方程組轉換為

$$x_1^{(new)} = (1-\omega)x_1^{(old)} + \frac{\omega}{a_{11}}(b_1 - a_{12}x_2^{(old)} - a_{13}x_3^{(old)})$$

$$x_2^{(new)} = (1-\omega)x_2^{(old)} + \frac{\omega}{a_{22}}(b_2 - a_{21}x_1^{(new)} - a_{23}x_3^{(old)})$$

$$x_3^{(new)} = (1-\omega)x_3^{(old)} + \frac{\omega}{a_{33}}(b_3 - a_{31}x_1^{(new)} - a_{32}x_2^{(new)})$$

6.3 逐次過鬆弛法(SOR法)

- 每一個方程式的右側是X所有分量前一次的估計值，和Gauss-Seidel法求得之新值的線性組合。當ω=1，SOR還原為Gauss-Seidel法。

例題6.7 使用SOR法解線性方程組

- 考慮3x3方程組Ax=b，其中

$$A=\begin{bmatrix} 4 & -2 & 0 \\ -2 & 6 & -5 \\ 0 & -5 & 11 \end{bmatrix}, b=\begin{bmatrix} 8 \\ -29 \\ 43 \end{bmatrix}$$

- 此方程組轉換為

$$x_1^{(new)} = (1-\omega)x_1^{(old)} + \frac{\omega}{4}(8 - 2x_2^{(old)} + 0x_3^{(old)})$$

$$x_2^{(new)} = (1-\omega)x_2^{(old)} + \frac{\omega}{6}(-29 - 2x_1^{(new)} - 5x_3^{(old)})$$

$$x_3^{(new)} = (1-\omega)x_3^{(old)} + \frac{\omega}{11}(43 + 0x_1^{(new)} - 5x_2^{(new)})$$

例題6.7 使用SOR法解線性方程組

- 使用 ω=1.2，以x(0)=(0,0,0)做開始求得

$$x_1^{(new)} = (-0.8)0 + 0.3(8 - 2(0)) = 2.4$$

$$x_2^{(new)} = (-0.8)0 + 0.2(-29 - 2(2.4) - 5(0)) = -4.84$$

$$x_3^{(new)} = (-0.8)0 + \frac{1.2}{11}(43 - 5(-4.84)) = 2.05$$

例題6.7 Matlab程式

- 主程式：Main_SOR.m
- SOR程式：SOR.m

```
clear all;
close all;

A=[4 -2 0;-2 6 -5;0 -5 11];
b=[8 -29 43]';
x0=[0 0 0]';
tol=0.0001;
w=1.2;
max_it=30;

tic;
x=SOR(A, b, x0, w,tol, max_it);
SOR_Time=toc;
fprintf('Solution: \n');
fprintf('x=[');
fprintf('\n  %6.4f',x);
fprintf(' ];');
fprintf('\nRun Time for SOR Method =%6.4f (sec)\n\n',SOR_Time);
```

例題6.7 Matlab程式

- 結果：

```
     i          x1         x2          x3
  1.0000     2.4000    -4.8400     2.0509
  2.0000    -0.9840    -3.1747     2.5491
  3.0000     0.6920    -2.3392     2.9052
  4.0000     0.8581    -2.0838     2.9733
  5.0000     0.9781    -2.0187     2.9951
  6.0000     0.9931    -2.0039     2.9989
  7.0000     0.9991    -2.0007     2.9998
  8.0000     0.9997    -2.0001     3.0000
  9.0000     1.0000    -2.0000     3.0000
SOR method converged
Solution:
x=[
  1.0000
  -2.0000
  3.0000 ];
Run Time for SOR Method =0.0940 (sec)
```


SOR法之特性分析

- 在推導SOR法時，可將Gauss-Seidel法中分解的方程組乘上鬆弛參數ω，亦即

$$\omega(D+L)x=-\omega Ux+\omega b$$

兩側同乘$(1-\omega)Dx$，得

$$(D+\omega L)x=((1-\omega)D-\omega U)x+\omega b$$

$$x=(D+\omega L)^{-1}((1-\omega)D-\omega U)x+\omega(D+\omega L)^{-1}b$$

因此，SOR迭代矩陣$C=(D+\omega L)^{-1}((1-\omega)D-\omega U)$

可證明$\det C=(1-\omega)^n$

SOR法之特性分析

- 一個矩陣的行列式值等於它所有特徵值的乘積，所以，如果$\omega \leqq 0$或$\omega \geqq 2$則C的特徵值中至少有一個絕對值大於等於1。因此，迭代參數ω的選取必須限制在$0<\omega<2$。
- 令$x^{(k)}$為線性方程組$Ax=b$的一組近似解，殘值向量r定義為$r=b-Ax^{(k)}$ 。在設計上，SOR可以讓殘值向量遞減得比Gauss-Seidel法更快。

SOR法之收斂性

- 迭代過程是否會收斂到解答，一直是一個重要的問題。對於正定矩陣，Ostrowski定理提供SOR法收斂性的資訊。如果對所有的$x=0$都有$x^TAx>0$，則矩陣A為正定。

SOR法之收斂性

- **Ostrowski定理**
  - *若A是正定矩陣，且0<ω<2，則對任何初始向量x，SOR法都會收斂。*
- **最佳ω定理**
  - *如果A為正定且三對角線，則最佳之ω為*

$$\omega = \frac{2}{1+\sqrt{1-\rho_J^2}}$$

  - *其中ρ_J為Jacobi法之迭代矩陣的頻譜半徑(Spectral radius)(特徵值中絕對值最大的)。*

例題6.8　求解PDE過程的線性方程組

- **用數個ω值來探討例題6.3之邊界值問題的收斂性，用$x^{(0)}$ =[0 0 ・・・ 0] 及容許誤差=0.00001。當然，當ω=1時，SOR法和Gauss-Seidel迭代是一樣。**

$$\begin{array}{ccccccccccc}
4u_1 & -u_2 & & -u_4 & & & & & & = & 0.3 \\
-u_1 & +4u_2 & -u_3 & & -u_5 & & & & & = & 0.25 \\
 & -u_2 & +4u_3 & & & -u_6 & & & & = & 1.9 \\
-u_1 & & & +4u_4 & -u_5 & & -u_7 & & & = & 0.5 \\
 & -u_2 & & -u_4 & +4u_5 & -u_6 & & -u_8 & & = & 0 \\
 & & -u_3 & & -u_5 & +4u_6 & & & -u_9 & = & 1.0 \\
 & & & -u_4 & & & +4u_7 & -u_8 & & = & 1.8 \\
 & & & & -u_5 & & -u_7 & +4u_8 & -u_9 & = & 1.0 \\
 & & & & & -u_6 & & -u_8 & +4u_9 & = & 2.0
\end{array}$$

例題6.8　求解PDE過程的線性方程組

- 使用矩陣型式Ax=b，其中

$$A=\begin{bmatrix} 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 4 & 0 & 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 4 & -1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 & 4 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 4 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 \end{bmatrix}, b=\begin{bmatrix} 0.3 \\ 0.25 \\ 1.9 \\ 0.5 \\ 0 \\ 1.0 \\ 1.8 \\ 1.0 \\ 2.0 \end{bmatrix}$$

例題6.8　求解PDE過程的線性方程組

$$y^T=\begin{bmatrix} 0.3594 & 0.5388 & 0.8272 & 0.5987 & 0.7188 & 0.8701 & 0.8165 & 0.8674 & 0.9344 \end{bmatrix}$$

ω	0.8	0.9	1	1.2	1.25	1.3	1.4
迭代次數	28	23	18	9	10	12	14

6.4 進階問題

- Jacobi、Gauss-Seidel和SOR法都屬於靜態迭代法，也就是寫成$x^{(k+1)}=Gx^{(k)}+c$的形式時，其中G和c為常數。

MATLAB內建函數所使用之方法

- 共軛梯度法(Conjugate gradient method)用於解方程組Ax=b，其中A為對稱正定矩陣，它求解方式是經由找$(1/2)x^TAx-x^Tb$的最小值。共軛梯度法所需儲存空間相對較小並保證會收斂，但如果A是病態矩陣(Ill-conditioned)則可能收斂緩慢。經過改良，前置條件(Preconditioned)共軛梯度法可增進收斂速率；但仍僅限於對稱且正定矩陣。
- 雙共軛梯度(Biconjugate gradient)法 (和它的前置條件形式)允許一般類型的係數矩陣，不再必須是對稱或正定(雖然此方法和最小化方法沒有直接關係)。

MATLAB內建函數所使用之方法

- GMRES [一般化最小殘值法(Generalized minimal residual method)]，此方法適用於係數矩陣很大、稀疏、準正定、且非(通常)對稱的線性方程組。

MATLAB內建函數

- 前置條件共軛梯度法(Preconditioned conjugate gradient method)：MATLAB函數pcg
- 最小殘值法(Minimal residual method)：MATLAB函數minres
- 雙共軛梯度法(Biconjugate gradient method)：MATLAB函數bicg
- 一般化最小殘值法(Generalized minimal residual method, GMRES)：MATLAB函數gmres
- 準最小殘值法(Quasi-minimal residual method, QMR)：MATLAB函數qmr

MATLAB內建函數

- **MATLAB函數pcg**
  - ➢ *使用前置條件共軛梯度法，矩陣A必須是對稱且正定。當迭代解之相對殘值norm(b-Ax)/norm(b)小於指定之容許誤差則視為收斂。當達到最大迭代次數時，程序也會終止。*

MATLAB內建函數

- **MATLAB函數pcg**
  - 基本形式：`x=pcg(A,b)`

    內定之容許誤差爲 10^{-6}。

    內定之最大迭代次數爲 n 與 20 中較小的一個。

    內定初始估計值爲零向量。
  - 使用者自定容許誤差：`x=pcg(A,b,tol)`
  - 輸入容許誤差和迭代次數：`x=pcg(A,b,tol,maxit)`
  - 輸入前置條件子：`x=pcg(A,b,tol,maxit,M)`
  - 輸入初始估計值：`x=pcg(A,b,tol,maxit,M,x0)`

    函數 `cgs` 則使用共軛梯度平方法。

MATLAB內建函數

- **MATLAB函數minres**
  - *最小殘值法是一種一般化的共軛梯度法，它適用於A為對稱但不一定是正定的情形。*
  - *基本形式：minres(A,b)*
  - *對函數pcg的各種不同型式各有其相對的變化形。*

MATLAB內建函數

- **MATLAB函數bicg**
  - *雙共軛梯度法可求Ax=b的解，其中A必須是方陣，且是大型稀疏矩陣，但不需要是對稱或正定的。*
  - *基本型式：x=bicg(A,b)*
  - *前面所列共軛梯度法的各種形式，雙共軛梯度法同樣都有。*
  - *雙共軛梯度法的所有選項都還有穩定化形式。此函數為bicgstab。*

MATLAB內建函數

- **MATLAB函數gmres**
  - *一般化最小殘值法，GMRES，被用於MATLAB的內建函數*
- **x=gmres(A,b)**
  - *使用者可以讓此函數在達到restart參數指定的迭代次數後再重新開始，呼叫方式為*
  - *x=gmres(A,b,restart)*
    - ☞restart的內定值為n，使用內定值則不會進行重開始。

MATLAB內建函數

- **MATLAB函數gmr**
  - *有幾種形式的準最小化殘值法(Quasi-minimal residual method, QMR)，分別對應到所列各方法的各種選項。*
  - *基本型式：x=qmr(A,b)*

共軛梯度法

- 若A是一個對稱且正定矩陣，則函數

$$\phi(x)=\frac{1}{2}x^T Ax - x^T b$$

  的最小值出現在它的梯度(即Ax-b)為零的時候。通常，最佳化方法使用一維搜尋。
- 在此方法中，可以用解析方式找出搜尋方向，因為新的殘值應該和目前的搜尋方向成正交。這些概念就是共軛梯度法的基礎。

共軛梯度法

- 因為求正交方向時只需用到r、s和x前一次的值，所以只需很小的儲存空間。
- 共軛梯度法保證在n步之內收斂(除了捨入誤差之外)，因為這些方向在空間中互為正交。在各次搜尋所用方向所展延的空間中，其誤差為最小。

MATLAB程式

- 主程式：Main_CG.m
- 結果：

```
      step        xl        x2         x3       norm(r)
         0     1.0000     1.0000     1.0000     16.1245
    1.0000    -0.1235     0.3580     1.0000      1.1919
    2.0000    -0.2764     0.5266     1.6272      0.0658
    3.0000    -0.2683     0.5122     1.6341      0.0000
x =
   -0.2683
    0.5122
    1.6341
```


前置條件共軛梯度法

- 如果A是病態矩陣，共軛梯度法的收斂會很慢。使用前置條件子，通常可以明顯的提昇收斂速度。前置條件子是一個矩陣(通常記為M)，它可使得$M^{-1}A$具有較佳的狀態，而求解係數矩陣為M的線性方程組，則相對較為容易。
- 基本觀念是將線性方程組$Ax=b$轉變成方程組$M^{-1}Ax=M^{-1}b$。(雖然為保留對稱性，必須要使用$M=LL^T$形式的前置條件子，但可改進演算法使得計算只和M^{-1}相關。)有多種不同形式的前置條件子。最簡單的是M=diag(A)。選擇此種形式時，其逆矩陣也是對角線的，所以牽涉到$M^{-1}r^{(k)}$的計算變得很簡單。
- 對於更複雜的前置條件子，這些計算可能包含，在每一階段解一個線性方程組，或實際求出逆矩陣。

前置條件共軛梯度法之演算法

```
Input
   A                                      %  係數矩陣
   M                                      %  前置條件子
   b                                      %  右側項向量
   x(1)                                   %  解的初始估計值
Initialize
   r(1) = b - A*x(1)                                     % 殘值
   s(1) = M^(-1)*r(1)                                    % 搜尋方向
For k = 1, ..., n
   c(k)   = r(k)^T*M^(-1)*r(k)/(s(k)^T*A*s(k))           % 搜尋參數
   x(k+1) = x(k) + c(k)*s(k)                             % 新的解
   r(k+1) = r(k)- c(k)*A*s(k)                            % 新的殘值
   d(k+1) = r(k+1)^T*M^(-1)*r(k+1)/(r(k)^T*M^(-1)*r(k))
   s(k+1) = M^(-1)*r(k+1) + d(k+1) s(k)                  % 搜尋方向
End
Return  x(k+1)
% 可加入收斂性的檢查，檢查是否r(k+1)<tol
```


MATLAB程式

- 主程式：Main_pcgMethod.m
- 內建程式x=pcg(a,b)

```
A = [ 10   4   1
   4 6 0
   1 0 2]
b = [ 1, 2, 3 ]';
tol = 0.00001;
max_it=20;

tic;
x=pcg(A, b,tol, max_it);
pcg_Time=toc;
fprintf('Solution: \n');
fprintf('x=[');
fprintf('\n  %6.4f',x);
fprintf(' ];');
fprintf('\nRun Time for Preconditioned Conjugated Gradient Method iteration =%6.4f (sec)\n\n',pcg_Time);
```

MATLAB程式

- 結果：

```
A =
   10     4     1
    4     6     0
    1     0     2

pcg converged at iteration 3 to a solution with relative residual 5.9e-017
Solution:
x=[
  -0.2683
  0.5122
  1.6341 ];
Run Time for Preconditioned Conjugated Gradient Method iteration =0.0470 (sec)
```


前置條件共軛梯度法之限制

- 使用共軛梯度法或前置條件共軛梯度法時，**係數矩陣A必須是對稱且正定的**，以確保最佳化問題存在有最小值以及搜尋參數的公式不會失敗。

雙共軛梯度法

- 雙共軛梯度法是共軛梯度法的一般化，它用以求解n x n線性方程組，不需要是對稱或正定。此方法和函數最小化並無直接的關係。
- 雙共輒梯度法使用四個序列的向量，分別記為r、s、p及q。序列r為殘值r=b-Ax。

雙共軛梯度法

- 此程序會在最多抑步時停止，因為在n步之後再也找不到正交方向了。由此程序所產生之向量的組合，滿足多個重要的關係式；此方法無法保證不會失敗或變得不穩定，但在實用上這種情況極少發生。

MATLAB程式

- 主程式：Main_BiConjGrad.m
- 雙共軛梯度法程式：BiConjGrad.m

```
A = [10 4 1
      4  6 0
      1  0 2]
b = [ 1, 2, 3 ]';  x0=[1 1 1]';
tol = 0.00001;
max_it=20;

tic;
x=BiConjGrad(A, b,tol, max_it,x0);
BiConjGrad_Time=toc;
```


MATLAB程式

- 結果：

```
     step       x1         x2         x3        norm(r)
        0     1.0000     1.0000     1.0000    16.1245
   1.0000    -0.1235     0.3580     1.0000     1.1919
   2.0000    -0.2764     0.5266     1.6272     0.0658
   3.0000    -0.2683     0.5122     1.6341     0.0000

BiConjugated Gradient method converged
Solution:
x=[
  -0.2683
  0.5122
  1.6341 ];
Run Time for Bi-Conjugated Gradient Method iteration
=0.0000 (sec)
```

MATLAB程式

- 主程式：Main_PBCG.m
- 前置條件雙共軛梯度法程式：PBCG.m

```
A = [10 4 1
     4  6 0
     1  0 2]
b = [ 1, 2, 3 ]';  x0=[1 1 1]';
tol = 0.00001;
max_it=20;

tic;
x=PBCG(A,b,x0,tol, max_it);
PBCG_Time=toc;
```


MATLAB程式

- 結果：

```
   step       x1         x2         x3         norm(r)
      0     1.0000     1.0000     1.0000     16.1245
 1.0000     0.0625     0.1072     1.0000      1.7930
 2.0000    -0.1617     0.3489     1.7491      0.8355
 3.0000    -0.2683     0.5122     1.6341      0.0000

Preconditioned Biconjugated gradient method converged
Solution:
x=[
  -0.2683
  0.5122
  1.6341 ];
Run Time for Preconditioned Bi-Conjugated Gradient Method
iteration =0.0000 (sec)
```

分析

- 共軛梯度法，是雙共軛梯度法在A為對稱時的特例，且參數s初始化為s=r。在此情況下，s和r維持相等，而p和q也相等，所以計算量少一半。理論上來說，如果A是對稱的，而且是正定的，則此方法適用。

一般化最小殘值法(GMRES)

- 一般化最小殘值法(GMRES)是用於求解線性方程組，該方程組的係數矩陣A很大且為稀疏、準正定，並且通常不為對稱。
- 一般化最小殘值法的解是來自Krylov子空間向量v,Av,A^2v,…的展延(span)。向量v=b-Ax是相對於解向量x的殘值。對於不對稱矩陣，可能會有複數特徵值。若A為nxn，則GMRES會在n步內求得確切解，但每一步都相當費時，所以此程序通常都只執行較少的步數(m)。在m步的近似解，可當做初始值以重新開始求解過程；不過以下的MATLAB函數並不包括自動重新開始的功能。

一般化最小殘值法(GMRES)

- 係數矩陣必須是正定的要求，代表所有特徵值的實部都一定為正。對於準正定係數矩陣，大多數特徵值的實部為正，而有少數實部為0。
- 當所有特徵值為正的時候，已知GMRES會收斂，但有人觀察到，當部份特徵值為負的時候它也會收斂。
- 它的基本觀念十分類似於Gram-Schmidt法(將一個線性獨立向量所成的集合，化為正交集合)。將此程序應用至Krylov子空間的方法稱做Amoldi程序。Amoldi程序(發表於1951)可用來將一個稠密矩陣化成Hessenberg形式。

一般化最小殘值法(GMRES)

- 此方法的基本程序是，建立一個矩陣V，它的各行是將向量v,Av,A^2v,...正交化的結果，以及一個上Hessenberg矩陣H，以使得$V^TAV=H$。矩陣V是nxm，其中A是nxn且m為步數。在建構好這些矩陣之後，求得y_k以使得

$$\left\| b - A(x_0 + Vy) \right\|$$

- 為最小，而這又是經由求

$$\left\| bbe_1 - H_k y_k \right\|$$

- 的最小值來達成，在此bb為初始殘值的範數。此最小值是經由下列方式獲得：

一般化最小殘值法(GMRES)

- 使用Givens旋轉將H轉換成上三角矩陣R，並將同樣的轉換用於向量bbe$_1$(以獲得向量G)。忽略H的最後一列，解Ry_k=G並得到解向量x如$x=x_0+Vy_k$。

MATLAB程式

- 主程式：Main_GMRES.m
- 程式：GMRES.m

```
A = [5 0 1 5 0 -1
     0 2 0 0 1 0
     1 0 1 -1 0 1
     5 0 -1 5 0 1
     0 1 0 0 2 0
     -1 0 1 1 0 1];
b = [ 10 3 2 10 3 2]';  x0=[0 0 0 0 0 0]';
tol = 0.00001;
max_it=20;
m=4;

tic;
[x,res]=GMRES(A,b,x0,m);
GMRES_Time=toc;
```

MATLAB程式

- 結果：

```
M=4
yk=[  1.9956 -1.3884  0.2999  0.0000 ];

Solution:
x=[   1.0000   1.0000   1.0000   1.0000   1.0000   1.0000 ];
res=[3.5527e-015 -8.8818e-016 8.8818e-016 3.5527e-015 -8.8818e-016 0.0000e+000 ];
Run Time for GMRES Method iteration =0.0160 (sec)
```


單體法(Simplex Method)

- 單體法為一種最佳化的數值方法。
- 可用在解線性或非線性問題。
- 可用在無限制條件(Non-constrant)或有限制條件(Constrant)的場合。

單體法(Simplex Method)

- 有一線性目標函數

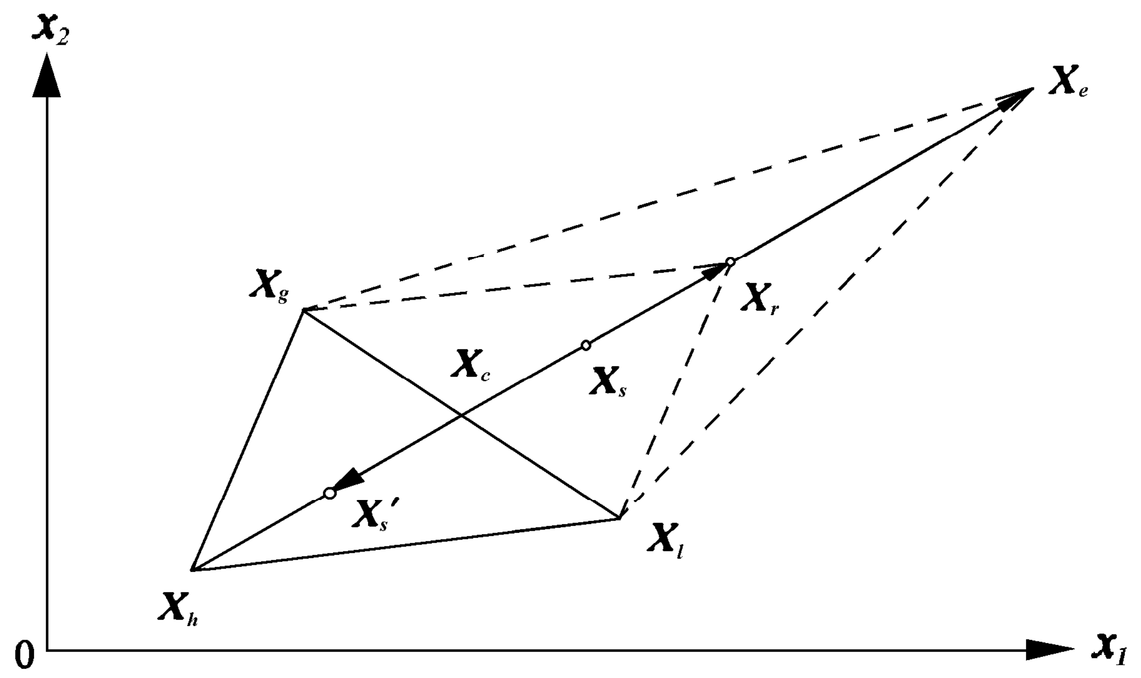
$$z = b_1 x_1 + b_2 x_2 + ... + b_n x_n$$

- 求其最大值，而同時滿足每個變數都不得為負值的限制條件，及另外m個等式或不等式的限制條件。滿足這些限制條件的向量，稱為可行解(Feasible solution)。

單體法(Simplex Method)

- 單體法的基礎為，所有的可行解是位於一個多維凸多面體或單體之內(因為邊界為超平面)，且最佳可行解一定是某一個頂點。在可行解區域的頂點上，某些約束不等式的等號會成立。總共會有n+m個約束條件;最佳可行解會完全滿足其中n個。單體法是一種有系統的步驟，它在每一步都使目標函數增加，而總步數 (幾乎一定是)不大於m和n中較大的一個。

Simplex 法的演算法



反射(Reflection)、 擴張(Expansion)、壓縮(Contraction)

MATLAB內建函數

- 解係數矩陣為對稱且正定之線性方程組的內建函數。
  - ➢*x=pcg(A,b)*
- 解係數矩陣為對稱(但不一定為正定)之線性方程組的內建函數。
  - ➢*x=minres(A,b)*
- 解係數矩陣為大型稀疏矩陣 (但不一定為對稱或正定)之線性方程組的內建函數。
  - ➢*x=bicg(a,b)*

MATLAB內建函數

- 用一般化最小殘值法解線性方程組的內建函數；可指定第三個輸入參數，以強制此方法在指定的迭代次數之後重新開始(內定值為係數矩陣的維度，此時不會重新開始)。
  - ➢ *x=gmres(A,b)*
- 用準最小殘值法解線性方程組的內建函數。
  - ➢ *x=qmr(A,b)*