第六章 數學式推導和矩陣計算演算法

領域知識(Domain knowledge)是工程師、科學家必備的基本知識，而電腦程式語言是一種處理數值計算的基本工具，如何利用工具也是必須下的基本功夫。早期的數學家們僅能憑空想像，無法以電腦程式語言實作，複數的計算是一個難題，譬如程式語言以 i 或 J 表示虛數合理嗎? C#程式語言中，虛數無法輸入 C#程式語言中。在動態系統中，複數的計算是不可避免的，也是令人感到困惑，譬如希爾伯特空間(Hilbert Space)是複數向量的內積(Inner Product)，量子力學的 Hermitian 矩陣等的處理。而 SMS(*S*harp *M*atrix *S*olver)精銳矩陣計算器求解器，能有效處理複數矩陣和微分方程式的計算問題。

在真實的世界裡，輸入系統的數值或系統的響應值，都應該是實數(Real Number)，數值的模擬計算中，可能暫時須要使用複數作計算，譬如計算不對稱矩陣的特徵值與特徵向量、或是求取多項式的根等都是複數，這些都是中間過程，但最後系統的響應值都應該是實數。

6.1 由微分方程式求得系統矩陣 A

矩陣常微分方程式(*O*rdinary *D*ifferential *E*quations)ODE(m, r, t)，以 m、r 和 t 三個參數表示，m 表示空間維度(Space Dimension)共有 m 個節點或稱自由度，r 表示狀態維度(States Dimension)共有 r 個自由度或稱階度(Order)，兩者都是時間 t 的函數，時間永遠是存在，並且一直不斷的在進行中，此三個軸是獨立且互相垂直，空間維度和狀態維度都是離散(Discrete)的型式，時間是連續(Continue)的型式，若要畫成圖形，若以空間維度是水平軸(Space DOF = 1, 2，… ，m)，狀態維度是垂直軸(State DOF = 0, 1, … ，r)，共計有 m *(r + 1)個離散數值，時間 t 是垂直於此平面的連續數值。

作者將 ODE(t, m, n)微分方程式分成兩類表示方式 :

Type I ODE(t, m, n) : 依照微分階的高低順序排列微分方程式

$$M * \ddot{y} + C * \dot{y} + K * y = \{f\}$$

Type II ODE(t, m, n) : 依照最高階微分排在等號的左邊，且其係數為 1，表示 Identity 矩陣。

$$\ddot{y} = B * \dot{y} + C * y + \{g\}$$

Type II ODE(t, m, n)的表示方式，是為了要從 ODE(t, m, n)，容易建構系統矩陣 A，ODE(t, m, n)若包含大掛號{ } ，表示可有可無的外力，依實際狀況而定的特別解。

SMS-2029 精銳矩陣計算求解器中，只有矩陣運算元(Matrix Operand)和矩陣運算子(Matrix Operator)，並無所謂於線性代數或是矩陣計算中的轉換矩陣(Transorm Matrix)，故 Ay 僅表示是一個矩陣變數名稱(Matrix Variable Name)，而不是 A 乘 y，A 乘 y 應寫成 A * y。

(0) ODE(t, m, n=0)

是 0 階方程式不是微分方程式，常常在線性代數中使用 Ay = b 的方式求解向量變數 y，稱 A 為 y 向量的轉換矩陣。但在 SMS-2029 精銳矩陣求解器中，以 A * y = b 的方式表示，即 A 是系統矩陣，y 是向量，b 是已知向量，無論是矩陣或是向量都是二維的陣列。故可求得 y = Ai * b。Ai 是矩陣變數的名稱，Ai 變數表示是 A 矩陣的逆矩陣，實際的程式碼是，y = ~A * b;

若系統矩陣非正方形時，則 (At * A) * y = At * b， 則

$$y = \left(A_t * A\right)_i * A_t * b$$

A_t 是矩陣變數的名稱，表示 A 矩陣的轉置，在 SMS-2029 精銳矩陣求解器中，提供矩陣計算的不同的運算子，和多個矩陣的類別(Class)。使用最小二乘法(Lease-Square Method)，SMS-2029 程式碼如下 :

```
ReMatrix  y = ~( !A * A ) * !A * b;
```

其條件是矩陣 A 已知，且列(Row)大於或是等於行(Column)，上式中，”~” 和

"!" 是矩陣運算子，"逆算"和"轉置"，而且運算子的優先順序(Operator Precedence)為 "!"大於"*" ，故上式中 * ! A 相連在一起是可以的，即表示矩陣 A 先轉置後再相乘。此部分作者已測試過是 OK，可能較複雜，初學者可以忽視。SMS-2029 程式碼如下 ：

```
using System;
using Matrix_0;
namespace ConsoleApp5
{
    internal class Program
    {
        static void Main(string[] args)
        {
double[,] A0 = { { 1, 3 }, { 2, 4 } };
double[,] b = { { 7 }, { 10 } };
// 將C#陣列A0轉為SMS-2029實數資料類別ReMatrix。
ReMatrix A = new ReMatrix(A0);
// y = (At * A)i * At * b
ReMatrix y = ~(!A * A) * ! A * b;
// 輸出向量 y :
Console.Write(" ** 向量 y : **\n{0}", new PR(y));
```

```
C:\WINDOWS\system32\cmd.exe
** 向量 y : **
        1.00000

        2.00000

請按任意鍵繼續 . . .
```

SMS-2029 精銳矩陣計算器，亦提供奇異值 SVD(Singular Value Decomposition)的求解，奇異值都是正實數值，程式碼如下 ：

```
SVD svd = new SVD( A );
ReMatrix D = svd.MatrixD;
ReMatrix Q = svd.MatrixQ;
ReMatrix P = svd.MatrixP;
ReMatrxi Qi = ~Q;
```

讀者可以自行測試計算結果，是否 A = P * D * Qi 成立?

(1) ODE(t, m, n=1)

n = 1 表示一階常微分方程式，與上式相同，微分方程式共計有兩種表示方式。

Type I ODE(t, m, n=1)的微分方程式寫成 :

$$B * \dot{y} + C * y = \{f\}$$

式中 B 和 C 是實數 mxm 的矩陣，$\dot{y}, y,$ 和 f 是 mx1 的向量。

Type II ODE(t, m, n=1)的微分方程式寫成 :

$$\dot{y} = A * y + \{g\}$$

式中 A 是實數 mxm 的矩陣，$\dot{y}, y,$ 和 g 是 mx1 的向量。

一般 ODE(t, m, n=1)的常微分方程式，作者從 Wikipedia 或是相關的書籍，都稱為狀態空間方程式(State Space Equation)。但二階以上作者稱之為空間相依狀態方程式(Spatial Dependency States Equation)。

(2) ODE(t, m, n=2)

n = 2 表示一階常微分方程式，與上式相同，微分方程式共計有兩種表示方式。

Type I ODE(t, m, n=2)的微分方程式可寫成 :

$$M * \ddot{y} + C * \dot{y} + K * y = \{f\}$$

式中 M，C，和 K 是實數 mxm 的矩陣，$\ddot{y}, \dot{y}, y$ 和 f 是 mx1 的向量。

Type II ODE(t, m, n=2)的微分方程式寫成 :

$$\ddot{y} = -M_i * C * \dot{y} - M_i * K * y + \{M_i * f\}$$

另外再加入平衡方程式 :

$$\dot{y} = I * \dot{y} + 0 * y + \{0\}$$

使用友矩陣(Companion Matrix)的表示方式，可將兩式合併寫成 :

$$\begin{pmatrix} \ddot{y} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} -M_i * C & -M_i * K \\ I & 0 \end{pmatrix} * \begin{pmatrix} \dot{y} \\ y \end{pmatrix} + \begin{pmatrix} g \\ 0 \end{pmatrix}$$

設系統矩陣 A 為 :

$$A = \begin{pmatrix} -M_i * C & -M_i * K \\ I & 0 \end{pmatrix}$$

則

$$\begin{pmatrix} \ddot{y} \\ \dot{y} \end{pmatrix} = A * \begin{pmatrix} \dot{y} \\ y \end{pmatrix} + \begin{pmatrix} g \\ 0 \end{pmatrix}$$

M, C, K 和 Mi，I，0 分別是實數 mxm 的質量、阻尼、和勁度矩陣，Mi 是 M 的逆矩陣，I 是 Iden 矩陣，0 是 Zero 矩陣，而 $\ddot{y}, \dot{y}, y$ 是 mx1 的向量。
系統矩陣 A 是(mxn)x(mxn)的矩陣，$(\ddot{y} \mid \dot{y}), (\dot{y} \mid y), (g \mid 0)$ 是(mxn)x1 的向量。

ODE(t, m, n)微分方程式，當 m>=2 是多空間點，且 n>=2 多階常微分方程式，作者稱之為空間相依狀態(SDS Spatial Dependency States)系統。

建構 I 和 0 矩陣，在空間點 m 已知的條件下，程式碼如下 :

```
Iden  iden = new Iden(m);
ReMatrix  I = iden.Matrix;
```

```
Zero  zero = new Zero(m);
ReMatrix  O = zero.Matrix;
```

(3) ODE(t, m, n=3)

n = 3 表示一階常微分方程式，與上式相同，微分方程式也是共計有兩種表示方式。

Type II ODE(t, m, n=3)的微分方程式寫成 :

$$\begin{pmatrix} \dddot{y} \\ \ddot{y} \\ \dot{y} \end{pmatrix} = A * \begin{pmatrix} \ddot{y} \\ \dot{y} \\ y \end{pmatrix} + \begin{pmatrix} g \\ 0 \\ 0 \end{pmatrix}$$

其中 A 是系統矩陣，是(mx3)x(mx3)的矩陣，其餘是(mx3)x1 的向量。

故我們知道 ODE(t, m, n)，包含時間點、空間節點、和最高階數，任何微分方程式，都是相同的 ODE(t, m, n)的微分方式來表示，可從 Type II 的微分方程式，求得系統矩陣 A。

6.2 由系統矩陣 A 求得特徵值 D 和特徵向量 Q

ODF(t, m, n=0)是零階的方程式，表示的方式如下 :

$$A * y = b$$

求 y 向量，可由系統矩陣 A 的分解，或是矩陣 A 的逆矩陣。

$$A = L * U, \quad A = Q * R$$

所謂 QR 是指 Givens Rotator，Householder Reflector，和 Gram-Schmidt Process 等分解方法。精銳矩陣求解器亦提供各種不同的求解類別(class)，包含直接求逆矩陣。

ODE(t, m, n)，階數參數 n 可以是 1、2、...，系統矩陣 A 是(mxn)x(mxn)的正方形實數矩陣，將系統矩陣 A 對角線化，亦即求系統矩陣 A 的特徵值矩陣 D ，和特徵向量矩陣 Q ，其關係如下 :

$$A * Q = Q * D$$

系統特徵向量應符合條件如下 :

$$DET(Q) \neq 0$$

則系統矩陣符合以下條件如下 :

$$A = Q * D * Q_i$$

系統矩陣 A 是(mxn)x(mxn)的正方形實數矩陣，Qi 是 Q 的逆矩陣，* 是矩陣相乘的運算子(Matrix Operator)，矩陣本身是運算元(Matrix Operand)。在精銳矩陣求解器類別(EIG class)，輸入系統矩陣變數 A ，可求得系統特徵值 D 和系統特徵向量 Q ，和逆系統特徵向量 Qi，SMS-2029 的程式碼如下 :

```
EIG  eig  =  new  EIG( A );
CxMatrix  D  =  eig.CxMatrixD;
CxMatrix  Q  =  eig.CxMatrixQ;
CxMatrix  Qi = eig.CxMatrixQi;
```

系統特徵值 D、系統特徵向量 Q、和逆系統特徵向量 Qi，其預設值(Default Value)都是複數矩陣(CxMatrix)的資料型態，另外尚有實數矩陣(ReMatrix)的資料型態的資料，兩者矩陣型態的資料相互顯性或是隱性(Explicit Or Implicit)的轉換，可以處理矩陣計算的問題，尤其是各種動態模型系統(Dynamic Model Systems)。

6.3 由特徵值 D 和特徵向量 Q 求取空間狀態響應。

作者從 Wikipedia 查看"State Space Representation"，似乎無法解惑，甚至有些不同的看法，其中"state space"指的是多個空間節點且是一階微分方程式 ODE(t, m, n=1)，作者認為多個空間節點且二階以上之常微分方程式 ODE(t, m, n>=2)，應稱為空間相依狀態(Spatial Dependency states)系統。

在數學的書籍裡，常常看到特徵與特徵向量，但如何利用呢？似乎並未談及。作者認為兩者是共軛的關係，不可單獨存在，也就是俗稱"Self-Adjoint"。而且兩者的預設值是複數，但是我們明明知道現實的世界是實數可量測，這些可能是使我們感到困惑的地方？作者利用 C#程式語言，對於特徵值與特徵向量，利用複數矩陣的計算，最後可以求得實數的響應資料(Data)，這是一種新穎的演算法。

以下是數學的推導和精銳矩陣程式的實作。

ODE(t, m, n)常微分程式，即參數 t 是時間、m 是空間節點數，n 是最高階，即 n=1、n=2、n=3、... ，先以 n=2 開始推導，至於 n=1，n=3、或是 n>=4 都是相同的推導方法。一般僅至二階，很少見到三階及三階以上的微分方程式。

(0) 當 ODE(t, m, n=2)常微分程式，SDS(Spatial Dependency States)響應公式 :

$$\begin{pmatrix} \ddot{y} \\ \dot{y} \end{pmatrix} = Q * D * Q_i * \begin{pmatrix} \dot{y} \\ y \end{pmatrix} + \begin{pmatrix} g \\ 0 \end{pmatrix}$$

可能稍繁複，但簡化可得下式 :

$$\begin{pmatrix} \dot{y}(t) \\ y(t) \end{pmatrix} = H_{exp}(D, Q, t) * d + \begin{pmatrix} \dot{y}_p(t) \\ y_p(t) \end{pmatrix}$$

或是

$$\left(\dot{y}(t) \mid y(t)\right) = H_{exp}(D, Q, t) * d + \left(\dot{y}_p(t) \mid y_p(t)\right)$$

"|"是矩陣或是向量的垂直合併運算子，上式數學式是空間相依狀態SDS(Spatial Dependency States)響應公式，式中下標p指的是特別解(Particular Solution)，可由Type I或是Type II的常微分方程ODE(t, m, n)求得特別解。SDS輸出矩陣是(mx2)x(mx2)的矩陣 ：

$$H_{exp}(D, Q, t) * d$$

d是向量係數，是(mx2)x1的向量 ：

$$\left(\dot{y} \mid y\right), \left(\dot{y}_p \mid y_p\right), d$$

（１）當ODE(t, m, n=1)常微分程式，SDS(Spatial Dependency States)響應公式 ：

$$\left(y(t)\right) = H_{exp}(D, Q, t) * d + \left(y_p(t)\right)$$

（２）當ODE(t, m, n=3)常微分程式，SDS(Spatial Dependency States)響應公式 ：

$$\left(\ddot{y}(t) \mid \dot{y}(t) \mid y(t)\right) = H_{exp}(D, Q, t) * d + \left(\ddot{y}_p(t) \mid \dot{y}_p(t) \mid y_p(t)\right)$$

無論n是1、2、3或是4階以上，輸出矩陣H_{exp}(D, Q, t)乘係數向量d都是固定的。其中D是系統特徵值矩陣，Q是系統特徵向量矩陣，兩者的預設值都是複數矩陣(Complex Matrix)，其資料型態是CxMatrix的類別，其中H_{exp}(D, Q, t)也是(mxn)x(mxn)的複數矩陣。至於n>=4階有相同的表示方式，故不再贅述了。

6.4　空間相依狀態參數為D、Q、和d

空間相依狀態 SDS(Spatial Dependency States)響應(Response)，等於 SDS 輸出矩陣 H_{exp}(D, Q, t)乘向量係數 d，故稱系統特徵值 D，系統特徵向量 Q，和向量係數 d 為空間狀態參數(Parameter)，只要知道 D，Q, 和 d，即可求得到響應。

如果初始條件已知，就可求得向量係數 d，此為較普遍性的求解法。若以 ODE(t, m, n=2)為例加以說明。
設

$$B(ta) = \left(\dot{y}(ta) \mid y(ta)\right) - \left(\dot{y}_p(ta) \mid y_p(ta)\right)$$

則向量係數 d :

$$d = [\ H_{exp}(D, Q, ta)\ ]_i * B(ta)$$

或是

$$d = \tilde{}\ [\ H_{exp}(D, Q, ta)\ ] * B(ta)$$

如果邊界條件已知，求向量係數 d 的方法稍複雜，即將輸出矩陣 H_{exp}(D, Q, t)分割與再合併。SMS-2029 求解器提供 RowSplice 類別，將輸出矩陣，依階數 n 分割，每個分割共計有 m 列數據，最後再合併成新的輸出矩陣，由已知的邊界條件求得向量係數 d，此為新穎的求解法，作者認為尚屬首見。如果是相同的系統，無論已知條件是初始值，或是邊界值，最後得到相同向量係數 d，亦即相同的空間狀態參數，則有相同的空間狀態響應，雖然使用文字說明較難，故請參考第七章的程式碼和輸出計算資料。