

第一章 SMS-2029 類別庫的環境設定

1 矩陣計算器

微軟在 2002 年時，正式推出 C# 程式語言，至今 (2022 年) 將近 20 年了，重要的改版包含泛型 (Generics)、語言查詢 (Language Integrated Query) 等等，雖然語言的改版不斷的進行，但其本身的架構是固定，對一般性程式的撰寫，並不一定需要隨著版本的更新而改變。

至於程式語言開發的撰寫，個人認為 Visual Studio 是最合適的整合開發環境，程式碼的編輯、編譯、偵錯、和連結等等的整合性作業，再配合熱鍵的使用，智慧型的感知功能 (IntelliSense)，能提供正面的效應。至於編譯器 (Compiler) 以及編譯後的執行元件 (Runtime Component)，這部分是屬於微軟內部的運作，程式的撰寫者無從得知，也較少人去研究。在此特別強調，整合開發環境的方便性，這裡所指整合開發環境，是微軟的 Visual Studio，而不是語言編輯器 Visual Studio Code。

精銳矩陣求解器 (Sharp Matrix Solver SMS-2029) 是使用 C# 程式語言撰寫而成的矩陣計算 (Matrix Computations) 類別庫 (Class Library)，其執行環境是微軟 Visual Studio 整合性 (IDE) 開發環境，求解矩陣的分解和運算結果。使用者並不一定需要懂得 C# 程式語言，仍然能夠運用自如，在 Visual Studio 的環境中，只要加入 using Matrix 的名稱空間 (namespace)，再輸入矩陣參數，按 VS 的熱鍵 Control+F5，就可以快速輸出矩陣計算後的解答。

2 主控台應用程式的基本設定

當第一次執行精銳矩陣計算器時，應先作三項必要的設定。

(1) Visual Studio IDE 環境的設定：

若您尚未安裝微軟的 Visual Studio，請上網查詢網址，並下載安裝，請特別

注意，是 Visual Studio 而不是 Visual Studio Code。測試的環境是螢幕主控台(Console)應用程式，Visual Studio(VS)環境的設定，請點選功能表(Menu)工具(Tools)，再點選選項(Options)作必要的設定。

(2) 主控台輸出環境的設定：

在 C:\WINDOWS\System 32\cmd.exe 的輸出環境設定，先執行空白的主控台應用程式，或按 Control+F5 的熱鍵，則自動開啓 C:\WINDOWS\System32\cmd.exe，再點選 C:\圖像(Icon)，接著點選內容，開啟內容對談視窗，共計五個選項，即選項、字型、版面配置、色彩、和終端機。其中版面配置和色彩是我個人的設定提供參考，亦可以依個人自己的喜好設。

(3) 名稱空間 Matrix 的設定：

將磁碟片內的 Matrix.dll 檔案，複製到硬碟內任何位置，譬如 C:\Matrix.dll。開啟空白的主控台應用程式，除了名稱空間 using System 之外，再加入 using Matrix 的名稱空間，點選功能表 Project，再點選 Add Reference...，開啟 Reference Manager 對談框(Dialog)，點選 Browse... 按鈕，尋找及點選 C:\Matrix.dll 檔案，回到主控台應用程式，此時名稱空間 using Matrix 之底下，就不再出現紅色折線的底線了，表示可使用 SMS-2029 的 C#類別庫程式碼了。

3 線性代數與矩陣計算

矩陣(Matrix)是一種資料結構(Data Structure), 由列(Row)和行(Column)所組成方形(正方形或是長方形)的陣列(Array)。重要的是 C#程式語言支援這種型態的資料結構，可以在電腦中執行，包含輸入矩陣、矩陣計算、和輸出矩陣。

線性代數和矩陣計算，都是闡述矩陣計算的理論、矩陣數學的基本運算，譬如，矩陣的加、減、乘、和除的算術運算，矩陣的水平合併或是垂直合併，組成一

個較大型的系統矩陣，檢查兩個矩陣是否相等？兩個矩陣是否不相等？求向量內積(Vector Inner Product)、求逆矩陣、矩陣的轉置(Transpose)、計算單位向量、行列式(Determinant)、矩陣的分解、如 Cholesky、Gauss、Gauss-Jordan、LU、QR，求特徵值矩陣(D)和特徵向量矩陣(Q)，即 $A * Q = Q * D$ 、求奇異值矩陣和奇異向量矩陣，即 $A * Q = P * D$ ，但一般線性代數的書使用 $A = U \Sigma V^T$ 表示方式，都是所探討的主題，使用精銳矩陣計算器 SMS-2029 類別庫(Matrix Class Library)求解器，於微軟 Visual Studio 整合性的環境中執行。

矩陣內的元素(Elements、Items)為純量(Scalar)，純量是數值，廣義的數值是複數(Complex Value)，當複數的虛數(Imaginary Value)為零時，即為實數(Real Value)，實數包含有理數、無理數、和整數。這是為了配合 C#主控台應用程式作這樣的分類並予以簡化，不同的型態(Type)可以自動轉換或是強迫轉換，即複數的範疇大於等於實數的範疇，而實數的範疇大於等於整數的範疇，與一般數學的書籍，在定義數值上有所不同，但不失矩陣計算的正確性和方便性。

在工程、物理等的計算科學方面，矩陣計算須使用到複數的數值，這是無法避免的，尤其在動態系統和量子力學的計算方面。實數不對稱的系統矩陣，一般而言，其特徵值是複數，相對的特徵向量也是複數，在計算過程中須要計算複數矩陣的逆矩陣，也須要利用複數矩陣相乘，一般程式撰寫所遇到的困難點，但精銳矩陣計算求解器，具有複數矩陣求解的功能，且最後運算的結果，回到實數的數據，也就是真實的實數世界，故複數僅是中間計算的過程而已。

實體的世界是三度空間，若再考慮時間的因素，就成為 4D 的問題。譬如一個動態系統，求解狀態空間響應(State Space Response)，每一個節點都有三個狀態，即變位、速度、和加速度，雖然系統矩陣是實數的矩陣，但求解過程中，特徵值和特徵向量都是複數，複數矩陣運算的結果，即變位、速度、和加速度，

應該是實數值，但個人尚未發現，有人實際使用複數矩陣計算求解，最後算出實數的結果。

4 矩陣、座標、和C#程式語言之整合

矩陣是方形的資料結構，由列(Row)和行(Column)所組成的，即使是1X1單一元素或是mX1的元素結構都是矩陣，但數學上稱mX1的矩陣為向量。矩陣內的每一個元素值是純量(Scalar)，純量可以是複數、實數、或整數。在C#程式語言中，矩陣的型態(Type)是陣列(Array)，也是一種參考型態(Reference Type)。在C#程式語言中，矩陣的變數(Variable)是由一個以上的英文字母所組成，一般第一個變數的字母大寫表示矩陣，小寫表示行向量(Column Vector)。譬如一般線性代數 $A+B$ ，表示矩陣A + 矩陣B， AB 表示矩陣A乘矩陣B。但C#程式語言中， AB 表示是一個變數AB，矩陣A乘矩陣B，則以 $A*B$ 表示， A/B 表示矩陣A除以矩陣B。

Fortran程式語言的次序(Order)是由1開始，即第1、第2、第3、等等，但C#程式語言是由0開始，即第0、第1、第2、等等次序位置。數學的座標軸是以數值表示，C#是以次序位置為坐標軸，在該位置的數值為該元素值。譬如第0坐標軸上有兩個數值，第0個數據是10000，第1個數據是0.0001，如果以數值為座標表示有難度，如果以次序座標表示則很容易。而C#程式語言中，以這種次序座標表示方式。即`double[,] A = {{10000, 0.0001}}`，甚至數值也可以是複數。如果是`double[,] A = {{1000, 0.0001}, {2, 3}}`，則表示第0個座標軸上的數據就是以上所表示，而第1個座標軸上的第0位置的數據是實數2，第一個位置的數據是實數3，依此類推。

精銳矩陣計算求解器，使用C#程式語言撰寫而成，而C#程式語言支援矩陣，每一個類別代表不同的矩陣計算，由各種不同功能的類別，組成類別庫，類別庫檔案即Matrix.dll。使用者無須了解C#程式語言，都能運用自如，只要輸入矩陣，而矩陣的計算和矩陣的輸出，都由求解器自行處理，非常容易。

假設 $a=193.4585$ ，則輸出 a 寫成，`Console.Write("\na = {0,20:F4}\n", a)`，其中`\n`表示新的一列，`{0,20:F4}`表示輸出的格式，0是第0個參數，20表示20個空格，數值靠右，F4表示小數點以下列印4位數。若寫成-20則表示數值靠左。

假設已知矩陣 A ，則輸出 A 矩陣寫成，`Console.Write("\nA 矩陣 : \n{0}\n", new PR(A))`，其中`{0}`表示第0個參數，亦即`new PR(A)`，由螢幕印出矩陣 A 。

使用者只須了解數值和矩陣的輸入和輸出，就能求得矩陣計算的結果。

5 解2X2複數矩陣之逆矩陣並驗證結果

已知矩陣 A ：

$$\begin{pmatrix} 3 + 5i & -7 + 3i \\ -4 - 2i & 8 + 6i \end{pmatrix}$$

```

1  using System;
2  using Matrix;
3
4  namespace ConsoleApp2
5  {
6      class Program
7      {
8          static void Main(string[] args)
9          {
10
11             double[,] Are = { {3, -7}, {-4, 8} };
12             double[,] Aim = { {5, 3 }, {-2, 6 } };
13             CxMatrix A = new CxMatrix(Are, Aim);
14             Console.Write("\nA (已知2X2矩陣)\n{0}", new PR(A));
15
16             CxMatrix Ai = ~A;
17             Console.Write("\nAi (即A之逆矩陣) : \n{0}", new PR(Ai));
18
19             ReMatrix B = (ReMatrix)(A * Ai);
20             Console.Write("\nB (即A * Ai = I) : \n{0}", new PR(B));
21

```

```

22      }
23  }
24 }

/* 輸出結果：
A (已知2X2矩陣)
    3.00000 + 5.00000i,    -7.00000 + 3.00000i
   -4.00000 - 2.00000i,    8.00000 + 6.00000i

Ai (即A之逆矩陣)：
    0.00338 - 0.14527i,    -0.09459 - 0.05743i
   -0.01014 - 0.06419i,    0.03378 - 0.07770i

B (即A * Ai = I)：
    1.00000    0.00000
    0.00000    1.00000

請按任意鍵繼續 . . .
*/
另參見 1_Fig01 螢幕快照。

```

6 執行環境的設定檢查

程式語言編譯器將 Source Code 編譯為機械語言，微軟的 Visual Studio VS(不是 Visual Studio Code)是整合性的開發環境，除了編譯器外，尚包含 linker、Debugger 等等，相對較複雜，而 VS 是將 Source Code 編譯為中間語言 (Intermidia Language)，再建構 (Build) 為 Comman Language Runtime (CLR) 等等複雜的 Runtime Component，所以第一次開始使用前，必須作必要的設定，熟悉相關的設定要求，譬如字型的大小、顏色、列印等、矩陣寬度的輸出設定，輸入的程式碼的參考 (Reference)，使用名稱空間 Matrix，請參考附圖 1_Fig02, 1_Fig03、1_Fig04、1_Fig05、1_Fig06、1_Fig07、1_Fig08、1_Fig09, 1_Fig10、1_Fig11、1_Fig12、和 1_Fig13 的設定。以上附圖僅是個人使用偏好設定，若有經驗亦可依個人的喜好作設定。

