

S_{harp} M_{atrix} S_{olver}

精銳矩陣計算求解器： SMS-2029-C#類別庫

主控台應用程式 實例應用與解說

精銳矩陣求解器工作室

第一章 SMS-2029 類別庫的環境設定

1 緒論

微軟在 2002 年時，正式推出 C# 程式語言，至今 (2021 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 (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) C:\WINDOWS\System 32\cmd.exe 的設定：

先執行空白的主控制台應用程式，或按 Control+F5 的熱鍵，則自動開啟 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，也可以依個人自己的喜好設定。

(3) 名稱空間 Matrix 的設定：

將 Matrix.dll 檔案，設置任何位置，譬如 C:\Matrix.dll。開啟空白的 Console 應用程式，除了名稱空間 using System; 之外，再加入 using Matrix; 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，表示可撰寫 C# 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 $A * Q = P * D$ ，(一般線性代數的書使用 $A = U \Sigma V^T$ 表示)，都是所探討的主題，而精銳矩陣計算器(SMS-2029)均能快速求得解答，目前作者個人尚未發現，以 C#物件導向程式語言撰寫，並在微軟 Visual Studio 整合性的環境中執行，具有此特色的矩陣類別庫(Matrix Class Library)求解器。

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，這或許與一般數學的書籍，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、和資料科學方面，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。譬如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，在中間計算過程中，須要計算複數矩陣的逆矩陣，也須要利用複數矩陣相乘，這是程式撰寫時，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，而且最後運算後的結果，轉回到實數的數據。

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，惟作者個人尚未發現有人，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nX1的元素結構都是矩陣，但數學上稱nX1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。而使用者無須了解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。

故使用者只須了解數值和矩陣的輸入和輸出，其他矩陣計算的部分，並無須去了解，也能很容易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：

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

```
using System;
using Matrix;

namespace ConsoleApp2
{
    class Program
    {
        static void Main(string[] args)
        {

double[,] Are = { {3, -7}, {-4, 8} };
double[,] Aim = { {5, 3 }, {-2, 6 } };
CxMatrix A = new CxMatrix(Are, Aim);
Console.Write("\nA (已知2X2矩陣)\n{0}", new PR(A));
```

```

CxMatrix Ai = ~A;
Console.WriteLine("\nAi (即A之逆矩陣) :\n{0}", new PR(Ai));

ReMatrix B = (ReMatrix)(A * Ai);
Console.WriteLine("\nB (即A * Ai = I) :\n{0}", new PR(B));

    }
}
}

```

/* 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)：

```

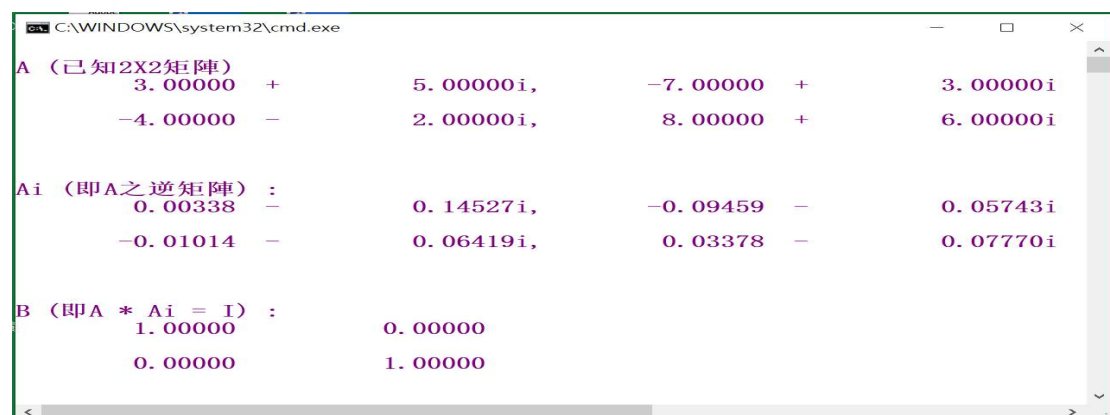
1.00000 0.00000
0.00000 1.00000

```

請按任意鍵繼續 . . .

*/

螢幕快照



6 小結

一般的語言編譯器相對較單純，僅將 Source Code 編譯為機械語言，而微軟的 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 (Intermedia Language)，再建構 (Build) 為 Common 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，也請熟悉相關的設定，譬如字型的大小、顏色、列印等等，請參見附圖。

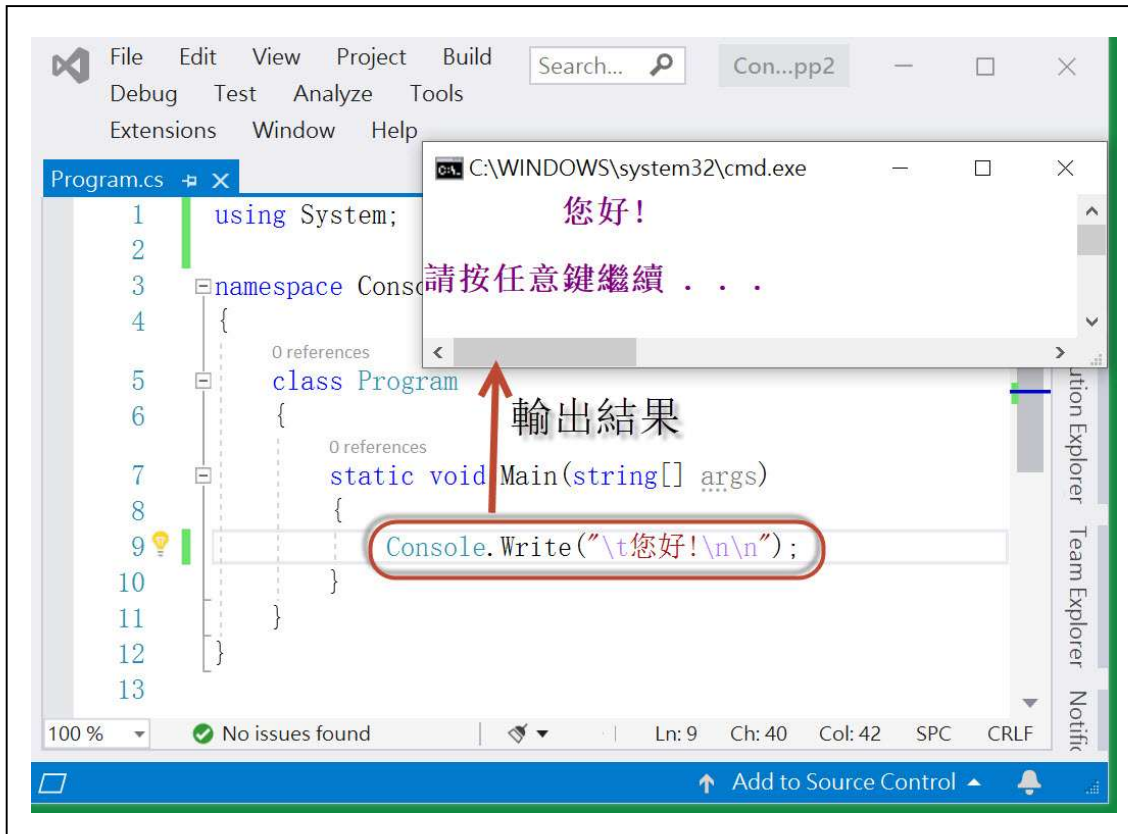
精銳矩陣計算器是主控台 (Console) 應用程式，經由 C:\WINDOWS\system32\cmd.exe 輸出，因為矩陣大小輸出寬度不一，故對於 C:\WINDOWS\system32\cmd.exe 作必要的設定，請參考附圖。

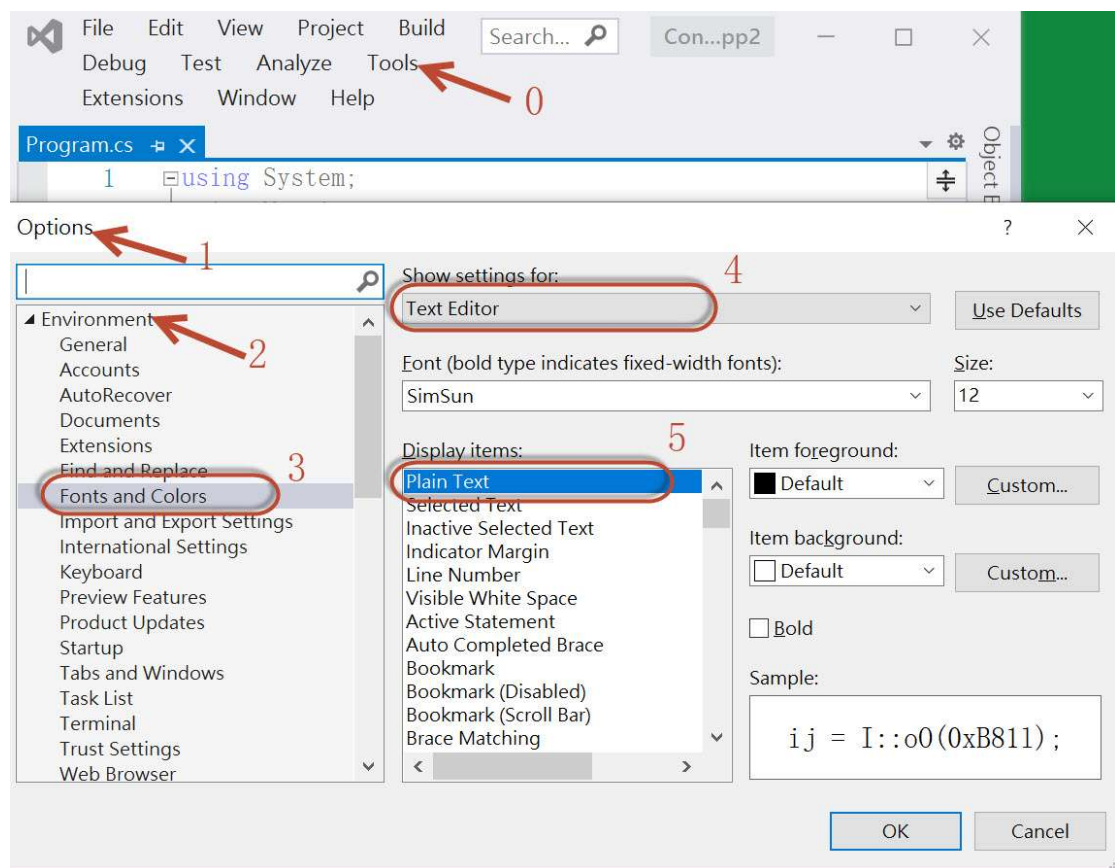
在 VS 的環境中，使用者所輸入的程式碼，須要參考 (Reference) 名稱空間 Matrix，故使用者須要再對 VS 作參考設定，請參考附圖。

這些附圖僅是作者個人使用的偏好，第一次使用者可參考附圖設定，若有經驗時，可依個人喜好作設定。

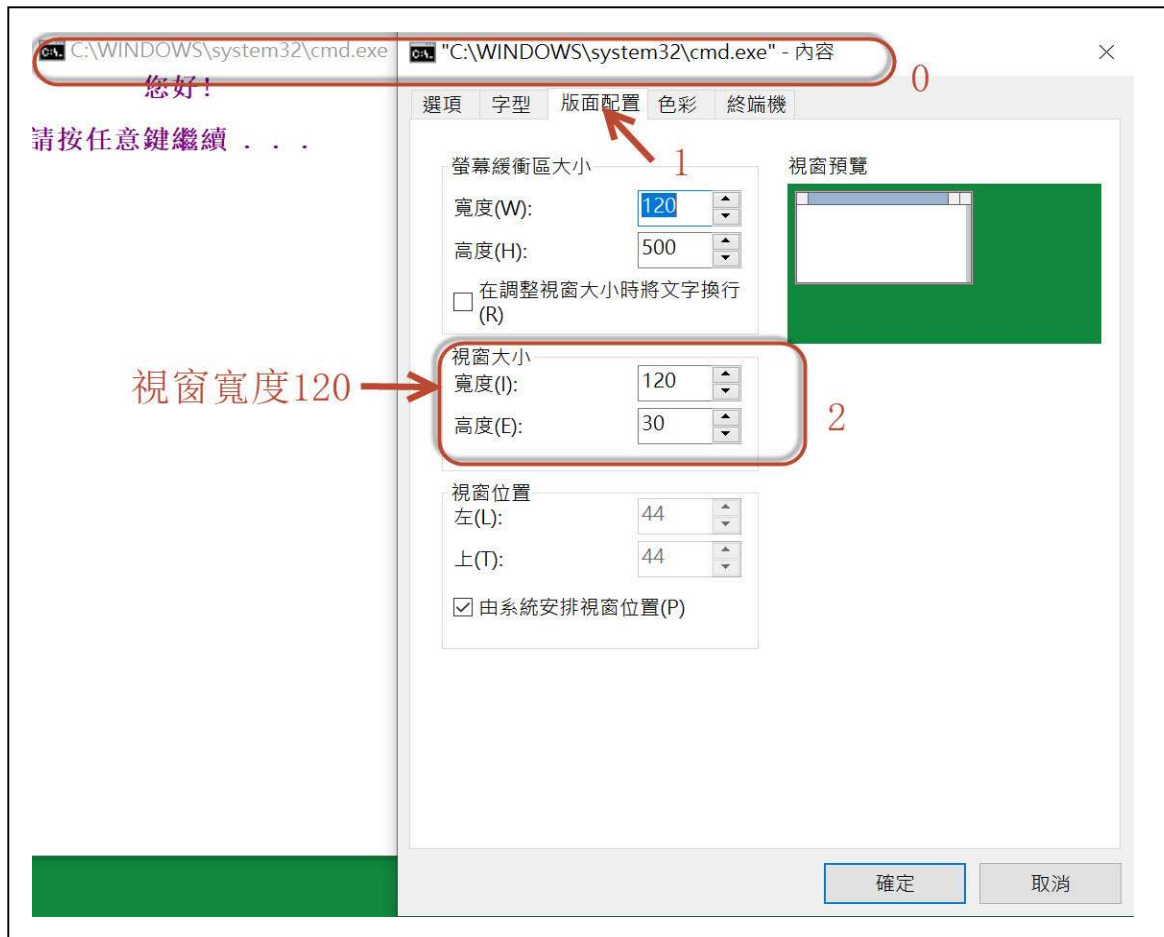
附圖

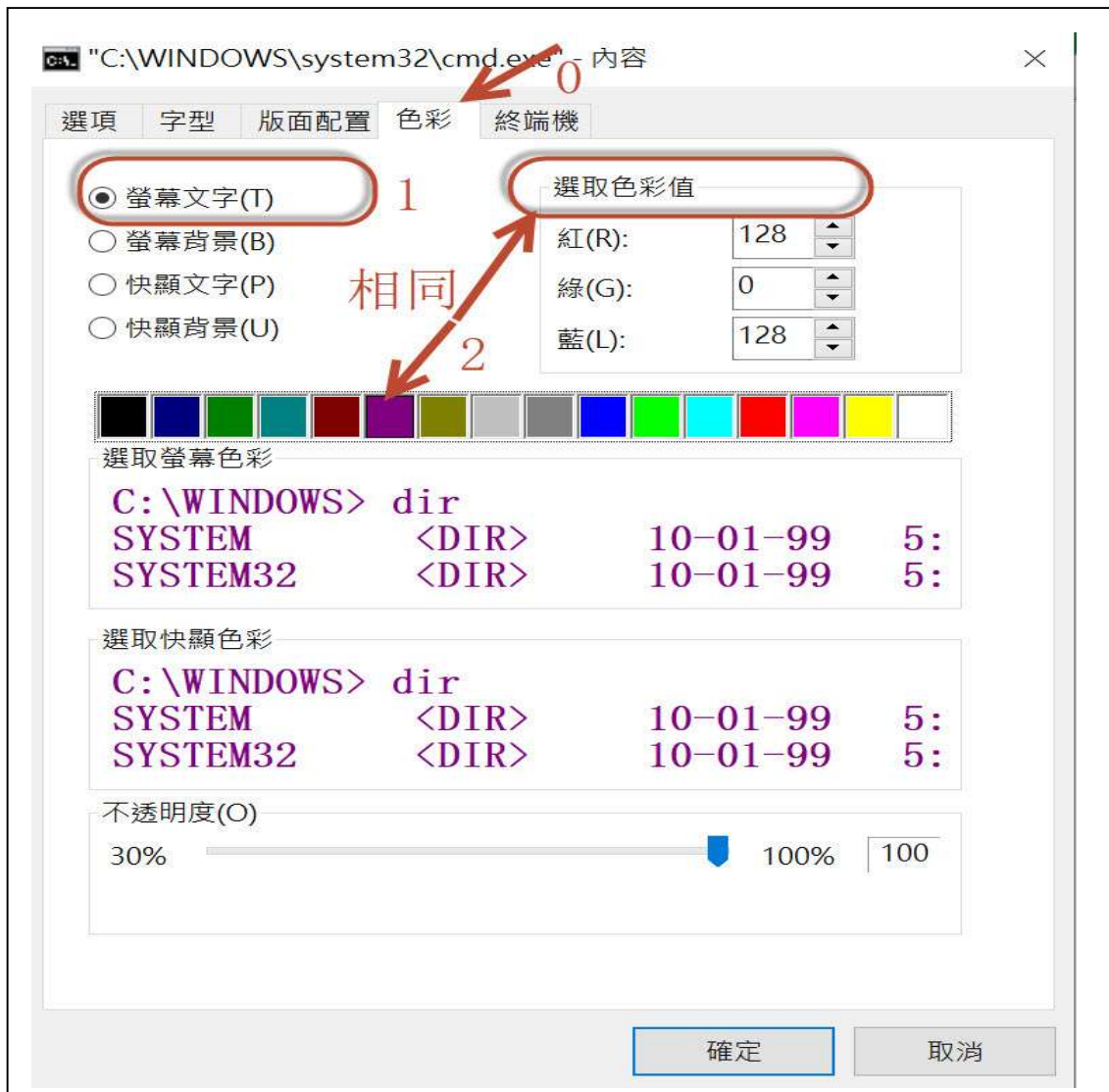
(1) Visual Studio 的基本設點

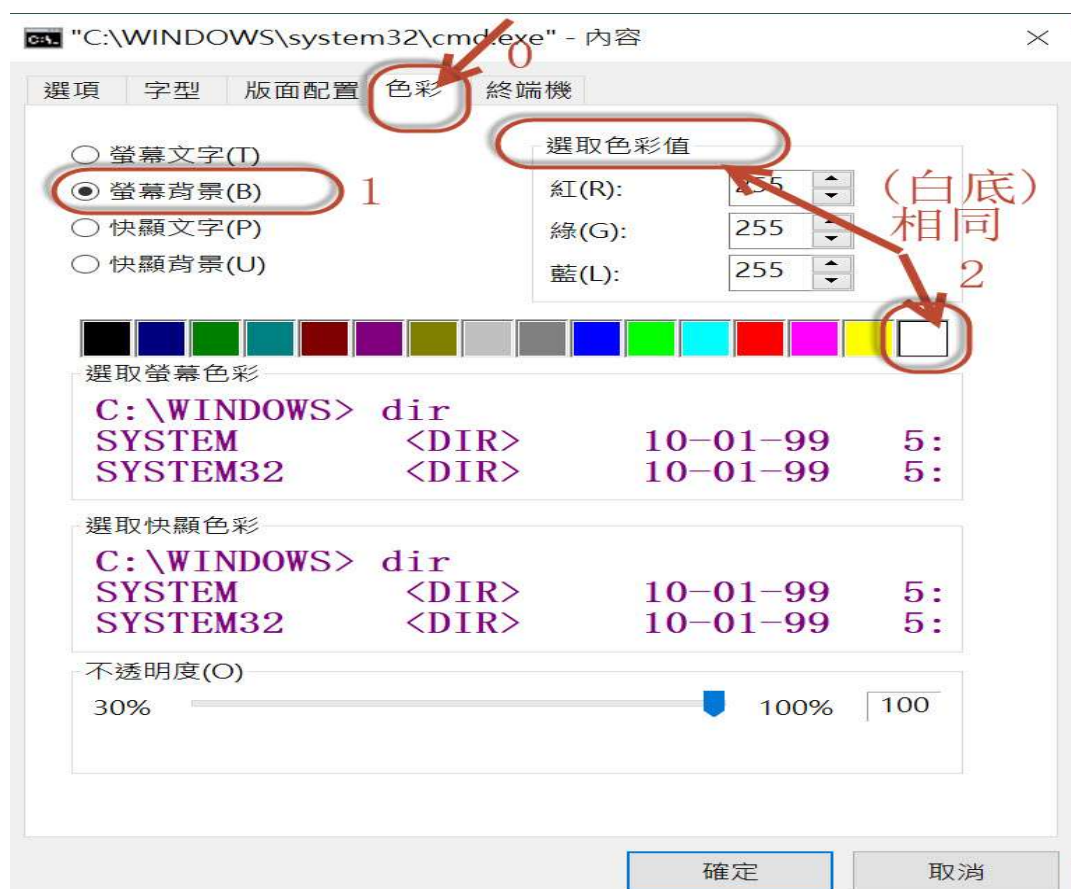




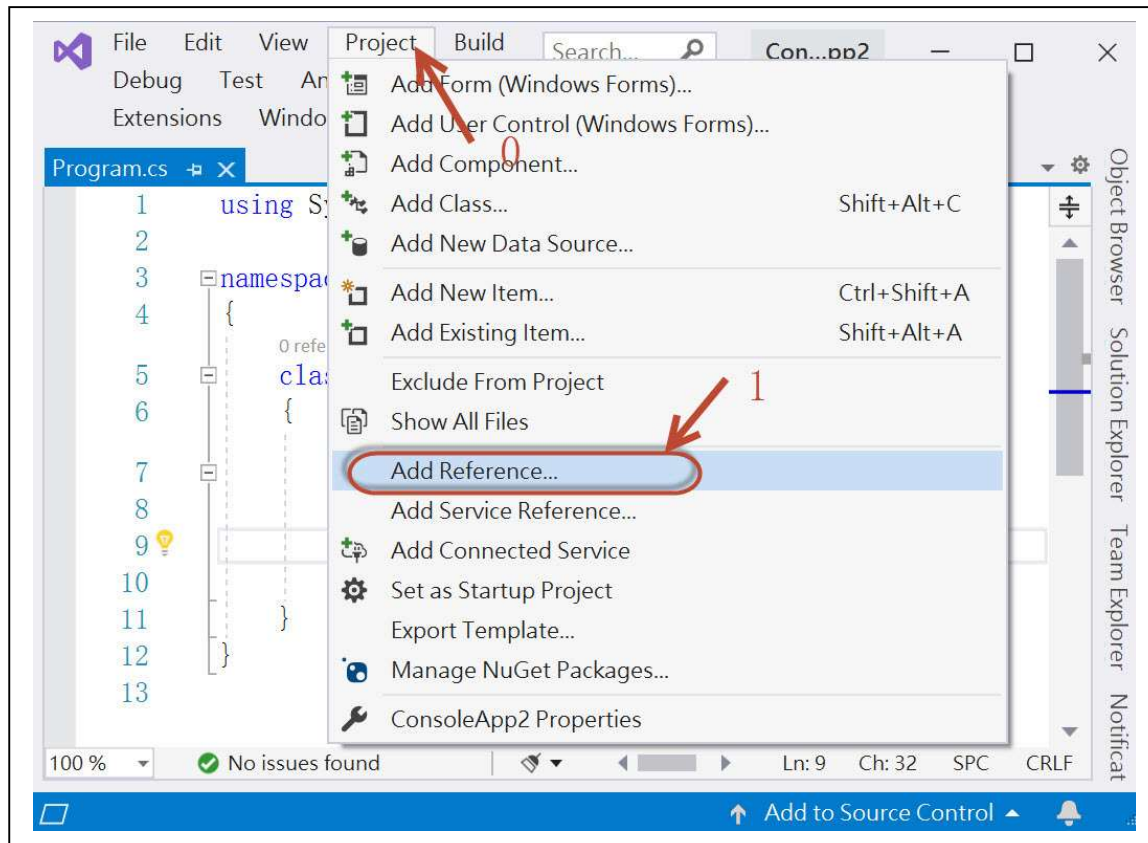
(2) C:/WINDOWS/System32/cmd.exe 輸出設定

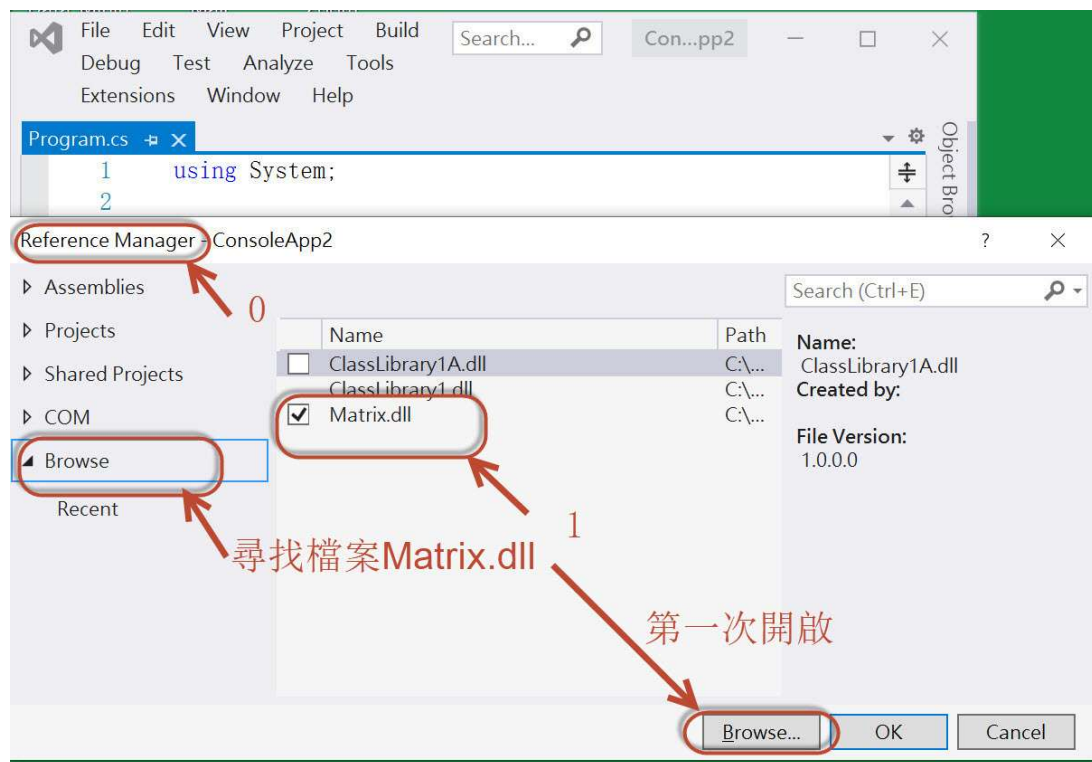




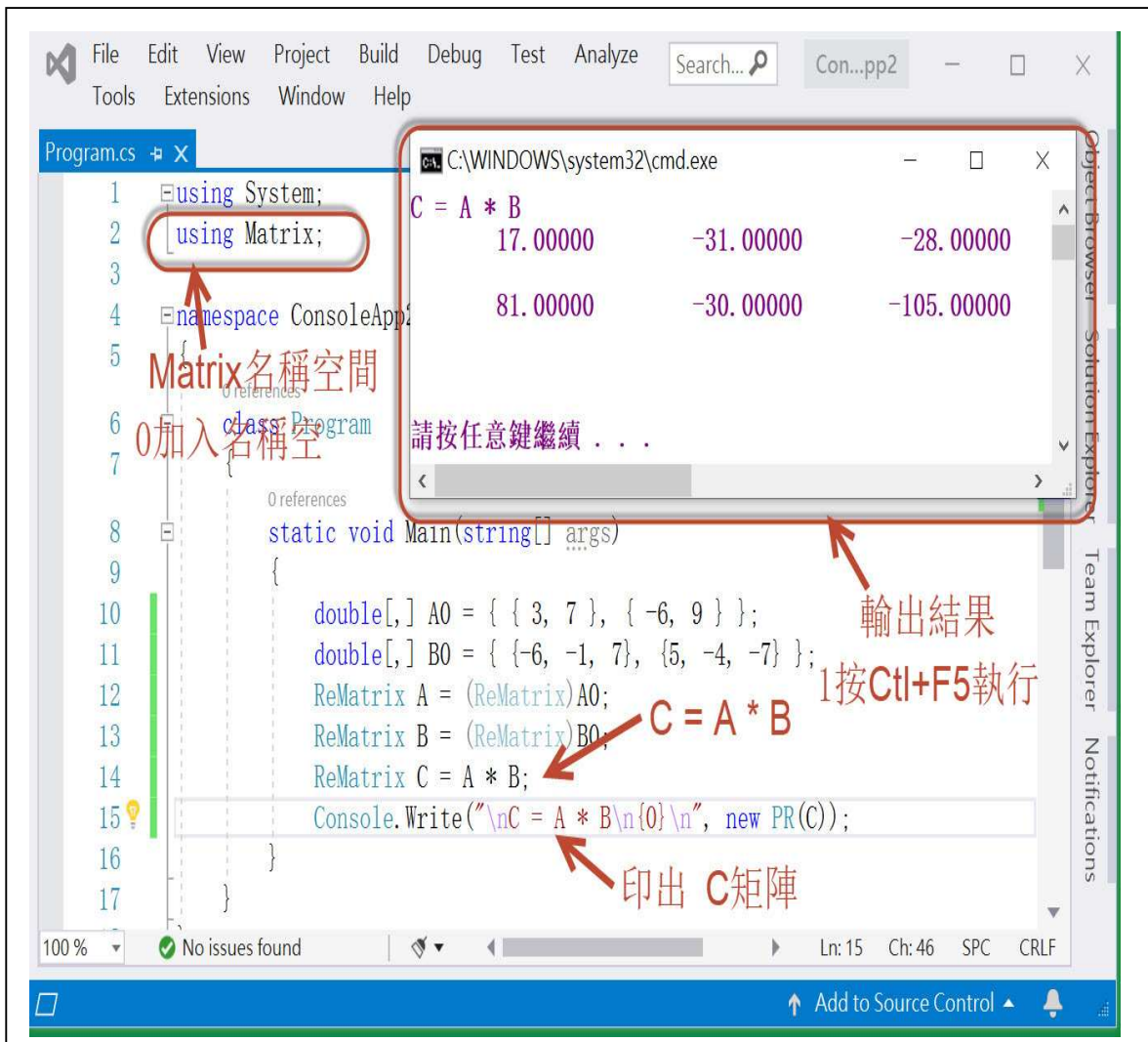


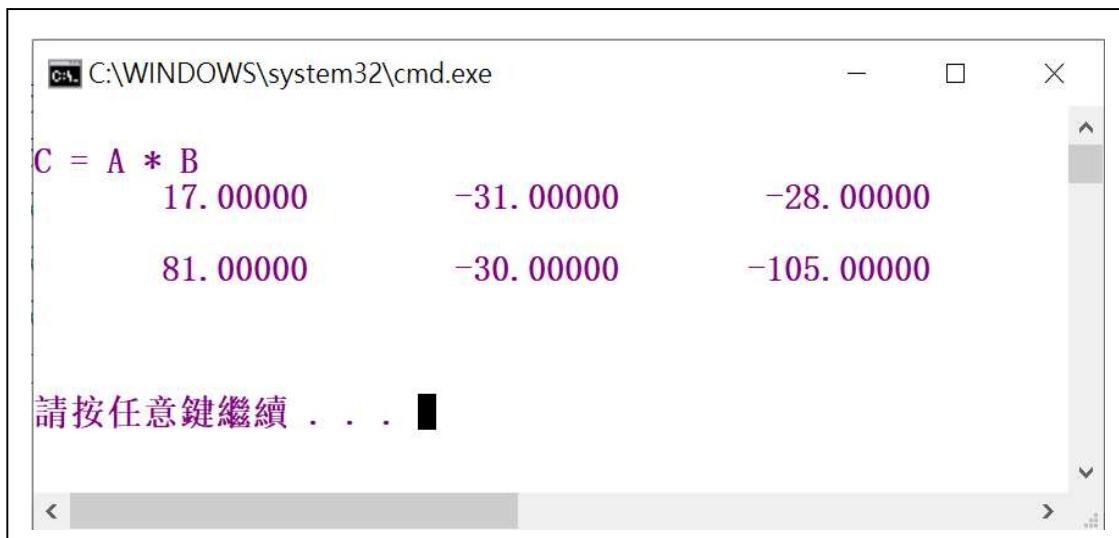
(3) Matrix.dll 檔案的設置與參考設定





(4) 精銳矩陣求解器執行結果





A screenshot of a Windows command prompt window. The title bar shows the path "C:\WINDOWS\system32\cmd.exe". The window contains a multiplication table for values 17, 81, -31, -30, -28, and -105. The prompt is in Chinese, asking the user to press any key to continue. The window has standard Windows window controls (minimize, maximize, close) and a scrollbar on the right.

```
C:\WINDOWS\system32\cmd.exe
```

C = A * B		
17.00000	-31.00000	-28.00000
81.00000	-30.00000	-105.00000

請按任意鍵繼續 . . . █

(5)由 C#再到精銳矩陣求解器的矩陣輸入方式：

已知 A 和 B 兩個矩陣。

$$A = \begin{pmatrix} 3 & 7 \\ -6 & 9 \end{pmatrix} \quad B = \begin{pmatrix} -6 & -1 & 7 \\ 5 & -4 & -7 \end{pmatrix}$$

則 C#程式語言的矩陣表示方式：

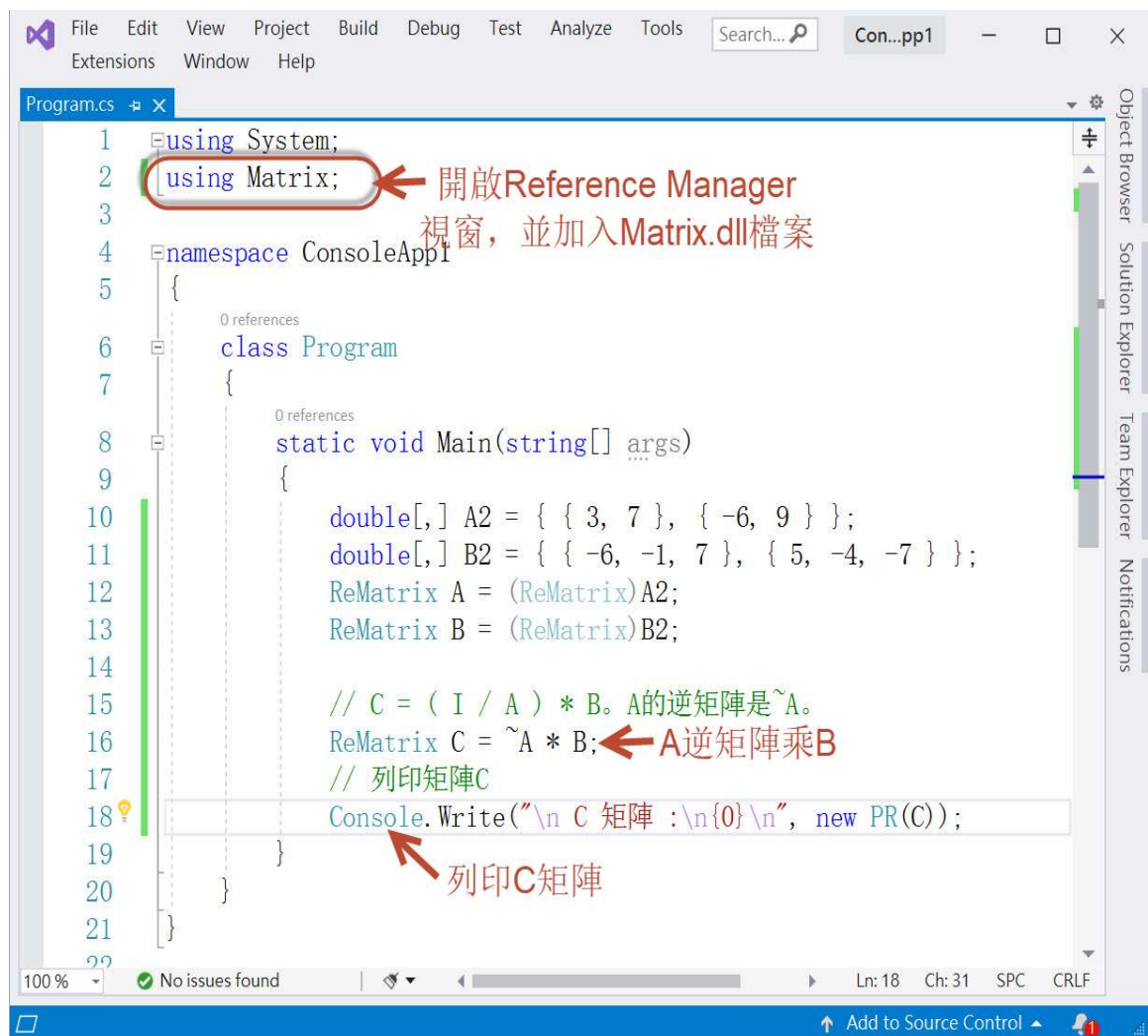
```
double[,] A0 = { { 3, 7 }, { -6, 9 } };  
double[,] B0 = { { -6, -1, 7 }, { 5, -4, -7 } };
```

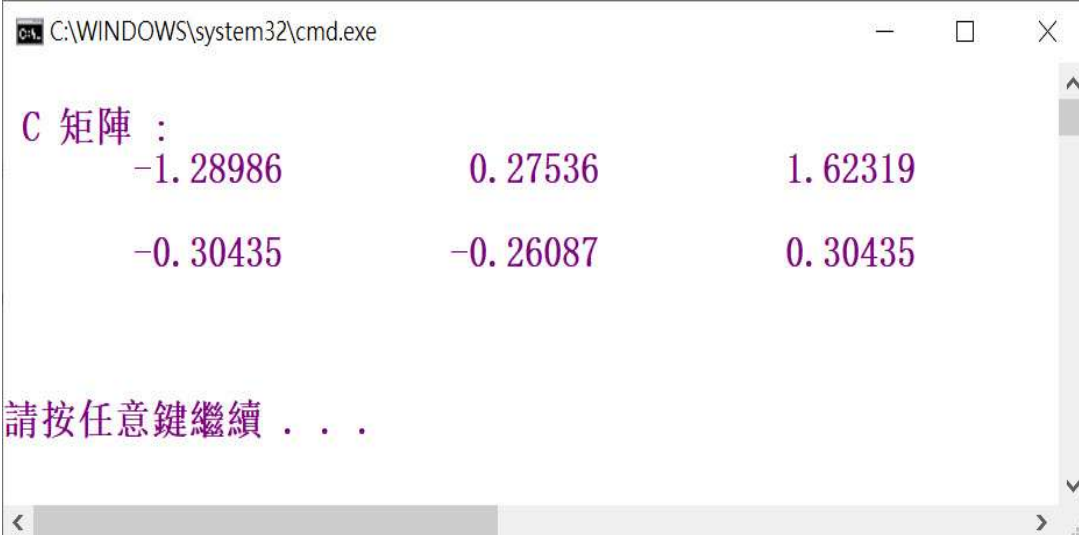
則精銳矩陣求解器的矩陣表示方式：

```
ReMatrix A = (ReMatrix)A0;  
ReMatrix B = (ReMatrix)B0;
```

使用精銳矩陣求解器的表示方式，其優點是矩陣(包含向量)可以有不同的運算子，如：

+(加)、-(減)、*(乘)、/(除)、&(水平合併)、|(垂直合併)、==(是否相等)、!= (是否不相等)、^(向量內積)、+(單位向量)、~(逆矩陣)、!(轉置)。共計有 12 種運算子。





A screenshot of a Windows command prompt window. The title bar shows the path 'C:\WINDOWS\system32\cmd.exe'. The window contains the following text in purple: 'C 矩陣 :', followed by a 2x3 matrix of values: -1.28986, 0.27536, 1.62319 in the first row, and -0.30435, -0.26087, 0.30435 in the second row. Below the matrix, it says '請按任意鍵繼續 . . .'. The window has standard Windows window controls (minimize, maximize, close) and a scrollbar on the right.

```
C:\WINDOWS\system32\cmd.exe

C 矩陣 :
-1.28986      0.27536      1.62319
-0.30435     -0.26087      0.30435

請按任意鍵繼續 . . .
```

第二章 矩陣的輸入與輸出

第三章 矩陣計算運算子

1 矩陣的基本計算

矩陣計算的運算子共計 12 個：

+(加), -(減), *(乘), /(除), &(水平合併), |(垂直合併), ==(是否相等), !=(是否不等), ^(內積), ~(逆), !(轉置), +(單位向量)。

(1)

$$A = \begin{pmatrix} 5 & 8 & -9 \\ -7 & 12 & 3 \end{pmatrix} \quad B = \begin{pmatrix} -3 & 9 \\ 7 & -3 \\ 4 & 5 \end{pmatrix} \quad C = \begin{pmatrix} 9 & 12 \\ -7 & -5 \end{pmatrix}$$

矩陣 A、矩陣 B、和矩陣 C、以 C#程式語言的表示，並列印矩陣 A，列印矩陣變數 $ABC = A * B / C$ 。

圖 1 是 C#程式碼，圖 2 是輸出結果。

(2)

$$D = \begin{pmatrix} 5 + 3i & 4 - 4i \\ -9 - 5i & 5 + 2i \end{pmatrix}$$

矩陣 D 以 C#程式語言的表示方式，列印矩陣 D，求矩陣 D 的逆矩陣 D_i ，並列印矩陣 D_i ，求矩陣變數 $C = B / D + !A / 3$ ，並列印矩陣 C。

圖 3 是 C#程式碼，圖 4 是輸出結果。

(3)

$$a = \begin{pmatrix} -7 \\ 9 \\ 5 \end{pmatrix} \quad b = \begin{pmatrix} 0 \\ -4 \\ 1 \end{pmatrix} \quad c = \begin{bmatrix} -6 & 3 & 8 \end{bmatrix} \quad d = \begin{bmatrix} 4 & 8 & 3 \end{bmatrix}$$

向量 a 和向量 b 的內積為 $e = a \wedge b$ ，由 C#程式碼求向量 e。另兩個列向量 c 和 d，應轉置為行向量，再求內積，即向量 $f = !c \wedge !d$ ，由 C#程式碼求 f 向量。

圖 5 是 C# 程式碼，圖 6 是輸出結果。

```
1 using System;
2 using Matrix_0;
3
4 namespace ConsoleApp1
5 {
6     class Program
7     {
8         static void Main(string[] args)
9         {
10             double[,] Aa = { { 5, 8, -9 }, { -7, 12, 3 } };
11             double[,] Bb = { { -3, 9 }, { 7, -3 }, { 4, 5 } };
12             double[,] Cc = { { 9, 12 }, { -7, -5 } };
13
14             ReMatrix A = (ReMatrix)Aa;
15             ReMatrix B = (ReMatrix)Bb;
16             ReMatrix C = (ReMatrix)Cc;
17
18             ReMatrix ABC = A * B / C;
19
20             Console.WriteLine("\n矩陣A :\n{0}\n", new PR(A));
21             Console.WriteLine("\n矩陣ABC :\n{0}\n", new PR(ABC));
22         }
23     }
24 }
25 /* 輸出結果 :
26 矩陣A :
27 5.00000    8.00000   -9.00000
28 -7.00000   12.00000    3.00000
29 矩陣ABC :
30 -4.94872   -7.07692
31 -30.07692 -55.38462
32 */
```


1 圖 - 1

```
C:\WINDOWS\system32\cmd.exe

矩陣A :
      5.00000      8.00000     -9.0000
     -7.00000     12.00000      3.0000

矩陣ABC :
     -4.94872     -7.07692
    -30.07692    -55.38462

請按任意鍵繼續 . . . █
```

1 圖 - 2

```

1  using System;
2  using Matrix_0;
3
4  namespace ConsoleApp2
5  {
6      class Program
7      {
8          static void Main(string[] args)
9          {
10 double[,] Dreal = { { 5, 4 }, { -9, 5 } };
11 double[,] Dimag = { { 3, -4 }, { -5, 2 } };
12 CxMatrix D = new CxMatrix(Dreal, Dimag);
13 Console.Write("複數矩陣D :\n{0}", new PR(D));
14
15 CxMatrix Di = ~D;
16 Console.Write("矩陣Di (矩陣D之逆矩陣) :\n{0}", new PR(Di));
17
18 ReMatrix D1 = (ReMatrix) (D * Di);
19 Console.Write("矩陣D1 (矩陣D乘逆矩陣D) :\n{0}", new PR(D1));
20
21 double[,] A0 = { { 5, 8, -9 }, { -7, 12, 3 } };
22 ReMatrix A = (ReMatrix)A0;
23 double[,] B0 = { { -3, 9 }, { 7, -3 }, { 4, 5 } };
24 ReMatrix B = (ReMatrix)B0;
25
26 CxMatrix C = B / D + !A / 3; // !A為矩陣A之轉置
27 Console.Write("複數矩陣C :\n{0}", new PR(C));
28     }
29 }
39 }

```

1 圖 - 3

```
C:\WINDOWS\system32\cmd.exe
複數矩陣D :
    5.00000 +      3.00000i,      4.00000 -      4.00000i
   -9.00000 -      5.00000i,      5.00000 +      2.00000i

矩陣Di (矩陣D之逆矩陣) :
    0.06887 +      0.01840i,    -0.04627 +      0.05889i
    0.12618 +      0.05152i,     0.07045 +      0.03155i

矩陣D1 (矩陣D乘逆矩陣D) :
    1.00000      0.00000
    0.00000      1.00000

複數矩陣C :
    2.59569 +      0.40852i,    -1.56046 +      0.10726i
    2.77024 -      0.02576i,     3.46477 +      0.31756i
   -2.09359 +      0.33123i,     1.16719 +      0.39327i

請按任意鍵繼續 . . .
```

1 圖 - 4

```

using System;
using Matrix_0;

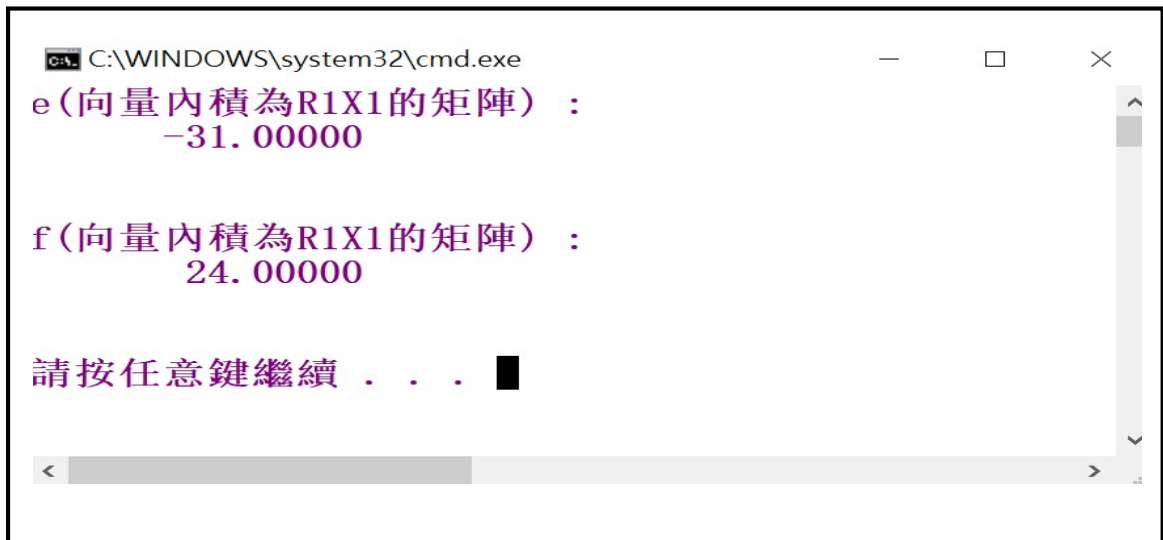
namespace ConsoleApp3
{
    class Program
    {
        static void Main(string[] args)
        {
            // C#程式語言接受矩陣的型式。
            double[, ] a0 = { {-7}, {9}, {5} };
            double[, ] b0 = { {0}, {-4}, {1} };
            double[, ] c0 = { {-6, 3, 8} };
            double[, ] d0 = { {4, 8, 3} };

            // 精銳矩陣計算器接受矩陣的型式。
            ReMatrix a = (ReMatrix)a0;
            ReMatrix b = (ReMatrix)b0;
            ReMatrix c = (ReMatrix)c0;
            ReMatrix d = (ReMatrix)d0;

            ReMatrix e = a ^ b;
            Console.WriteLine("e(向量內積為R1X1的矩陣) :\n{0}", new PR(e));
            // ! 是轉置的運算子。^ 是內積的運算子。
            ReMatrix f = !c ^ !d;
            Console.WriteLine("f(向量內積為R1X1的矩陣) :\n{0}", new PR(f));
        }
    }
}

```

1 圖 - 5



1 圖 - 6

2 行列式

行列式是方形矩陣的數值內容量。由第 0 列或是第 0 行開始，元素與其對應的 cofactor 的乘積，內容量是矩陣特徵值的乘積，矩陣若使用 Gauss 消去法，也是對角元素的乘積。

$$A = \begin{pmatrix} -8 & 7 & 4 \\ -5 & 9 & 2 \\ 1 & 7 & -4 \end{pmatrix}$$

已知矩陣 A，求矩陣 A 的行列式，其 C# 程式碼若圖 1 和輸出如圖 2。

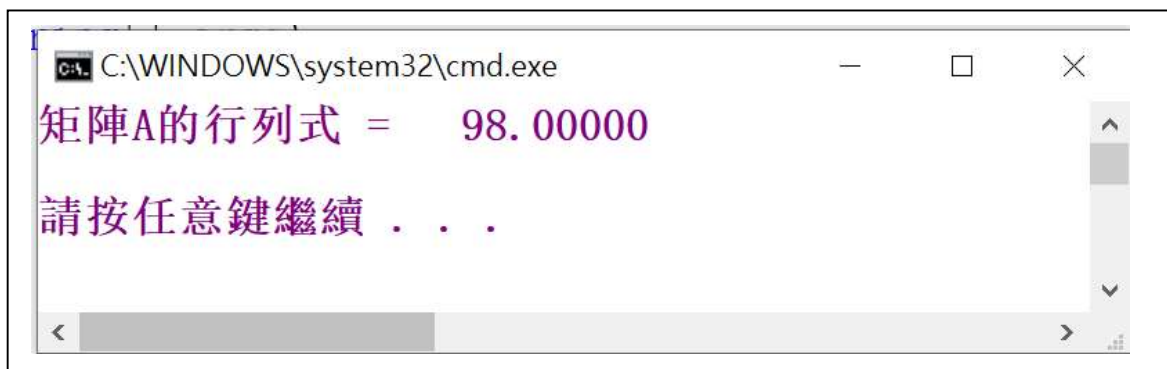
C# 程式碼的特徵值矩陣如圖 3 和輸出如圖 4，C# 程式碼的 Gauss 消去法如圖 5 和其輸出如圖 6。

```

1  using System;
2  using Matrix_0;
3
4  namespace ConsoleApp4
5  {
6      class Program
7      {
8          static void Main(string[] args)
9          {
10 double[,] A =
11 {
12     {-8, 7, 4}, {-5, 9, 2}, {1, 7, -4}
13 };
14 DET dt = new DET(A);
15 double val = dt.Value;
16 Console.WriteLine("矩陣A的行列式 = {0,10:F5} \n\n",
17     val);
18     }
19 }
20 }

```

2 圖 - 1



2 圖 - 2

```

1  using System;
2  using Matrix_0;
3
4  namespace ConsoleApp4
5  {
6      class Program
7      {
8          static void Main(string[] args)
9          {
10             double[,] A =
11             {
12                 {-8, 7, 4}, {-5, 9, 2}, {1, 7, -4}
13             };
14             EIG eig = new EIG(A);
15             ReMatrix D = eig.MatrixD;
16             Console.WriteLine("\n特徵值矩陣D : \n{0}", new PR(D));
17         }
18     }
19 }

```

2 圖 - 3

```

C:\WINDOWS\system32\cmd.exe
特徵值矩陣D :
-8.55670      0.00000      0.00000
0.00000      7.15696      0.00000
0.00000      0.00000     -1.60026

請按任意鍵繼續 . . .

```

2 圖 - 4


```

1  using System;
2  using Matrix_0;
3
4  namespace ConsoleApp4
5  {
6      class Program
7      {
8          static void Main(string[] args)
9          {
10             double[,] A =
11             {
12                 {-8, 7, 4}, {-5, 9, 2}, {1, 7, -4}
13             };
14             Gauss gs = new Gauss(A);
15             ReMatrix G = gs.Matrix;
16             Console.WriteLine("\n矩陣G :\n{0}", new PR(G));
17         }
18     }
19 }

```

2 圖 - 5

```

C:\WINDOWS\system32\cmd.exe
矩陣G :
-8.00000      7.00000      4.00000
 0.00000      4.62500     -0.50000
 0.00000      0.00000     -2.64865

請按任意鍵繼續 . . . 

```

2 圖 - 6

3 純量、矩陣、向量、直角坐標、歐幾里得空間、和希伯特空間

個人電腦再加上物件導向程式語言的崛起，約在二十多年前誕生，故早期的數學家對於數值的計算，因沒有電腦和程式語言的協助，也只能憑空想像而已。若能將線性代數、矩陣計算與 C# 程式語言結合在一起，是一件頗為方便的事情，當然以下的闡述，純粹是個人的想法。

純量是單一的數值量，除了實數外尚包含複數，實數僅是複數的一個特例而已，換言之，當複數的虛數值為 0 時，就是實數，我們是存在實數可量測的世界中，複數僅是中間計算過程中，不可避免的程序，最後計算的結果，仍然須要回到實數的結果。本人曾於狀態空間計算(state space Computation)的過程中，得到此驗證。

矩陣包含水平方向和垂直方向的陣列 (Array)，水平和垂直方向可以有多個維度。在 C# 程式語言中，陣列尚包含鋸齒型，而一般線性代數指的是長方形(包含正方形)的陣列。精銳矩陣計算器(Sharp Matrix Solver)中，使用 CxMatrix 類別和 ReMatrix 類別代表複數矩陣和實數矩陣，還包含 12 種矩陣的運算子(binary and unary operators)，幾乎算術、邏輯、條件等等運算子。

所謂向量指的是行向量，所以向量也是矩陣的一個特例，即線性代數指的 $Cm \times 1$ 或 $Rm \times 1$ 的矩陣，另 $C1 \times 1$ 和 $R1 \times 1$ 也是矩陣，但該元素的值是純量。

直角座標系統指的是卡氏座標，包含圓柱座標等，換言之圓柱座標也是直角坐標的一種。

在座標系統內的空間，即點與點的連線都是直線，是所謂歐幾里得空間，若點坐標值是複數時，稱作希伯特空間(Hilbert Space)。就如同強調的，複數包含實

數，實數包含整數一樣，故希伯特空間包含歐幾里得空間一樣，在精銳矩陣計算器中，實數可以隱性(implicit)轉換為複數，而複數須顯性 (explicit)轉換為實數一樣，使用 `cast`，即(`ReMatrix`)將複數矩陣轉為實數矩陣，同理整數可隱性轉換為實數一樣。

線性代數中，常常使用實數坐標軸和垂直的虛數坐標軸，表示複數平面，但僅能表示一個點而已，而且僅能表示向量加減，功能有限。但精銳矩陣計算器，複數純量使用 `CxScalar` 類別，複數和實數矩陣可分別使用 `CxMatrix` 和 `ReMatrix` 類別，共計有 12 運算子，此外尚有其他種類的輔助類別。

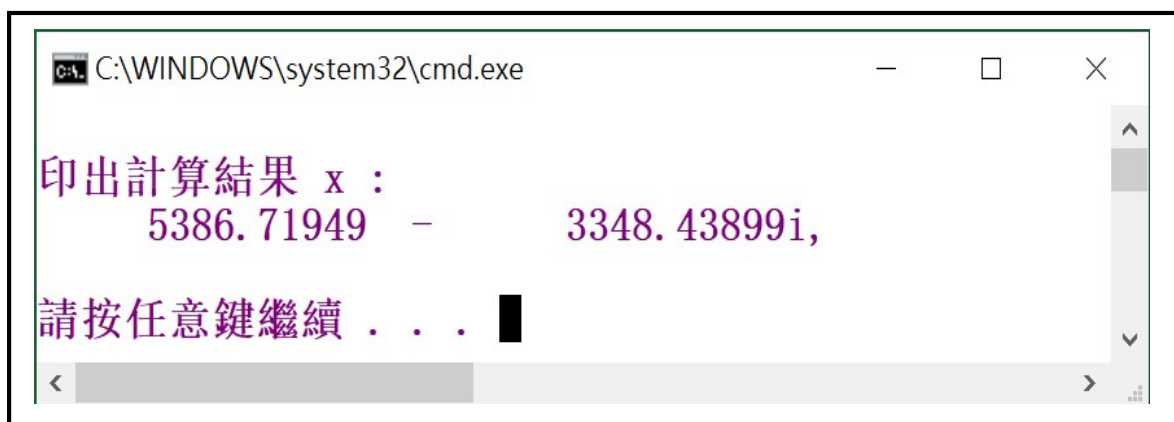
舉一個簡單的複數純量的計算，譬如：

$$(5 - 3.5i) / (0.3 + 0.009i) * (93 + 10.7i) / (0.3 + 0.009i) = ?$$

精銳矩陣求解器的程式碼如下：

```
CxScalar a = new CxScalar(5, -3,5);  
CxScalar b = new CxScalar(0.3, 0.009);  
CxScalar c = new CxScalar(93, 10.7);  
CxScalar x = a / b * c / b ;  
Console.WriteLine("\n 印出計算結果  x : \n{0}\n", new PR(x));
```

輸出結果如下：



4 狀態空間的求解

- 5 向量的類別
- 6 特徵值矩陣與特徵向量矩陣
- 7 奇異值矩陣與奇異向量矩陣
- 8 系統矩陣與矩陣的分解
- 9 精銳矩陣求解器的重要類別
- 10 舒爾 (Schur) 矩陣
- 11 精銳矩陣求解器類別庫的類別

第四章 類別程式庫

A				
Add1	Add2	Add3	Affine	Arnoldi
AUG	AUG2	Arange		
C				
CG	ChkNullMat	ChkSqMat	CHOL	class1
ClearZeroCol	ComMat	ComMat2	CoordMat	COS
CxAdd1	CxAdd2	CxAdd3	CxClearZeroCol	CxComMat2
CxComMat3	CxCOS	CxDET	CxEIG	CxFromX
CxGetRow	CxGetVector	CxHerm	CxINV	CxIP
CxIP2	CxIP3	CxIP4	CxIP5	CxIsDiag
CxIsImZero	CxIsUpperTri	CxLRRotator	CxM1	CxM2
CxM3	CxMatPro	CxMatPro2	CxMatrix	CxPivotPro

CxPivotPro	CxPR	CxPR2	CxPROJ	CxRotator
CxScalar	CxSetMatrix2	CxSwapRow	CxToDexp	CxToX
CxIP				
D				
Deflator	Descent	DET	DET2	DET3
DET4	DETZero	DirectSum		
E				
EIG	EIG2	EIG3	EIG4	EntryMax
ErrValue	EV	EVCN		
G				
Gauss	GaussJordan	GaussJordan2	GaussSeidel	GetMatrixQ
GetMatrixQ2	GetMatrixQ3	GetMatrixQ4	GetRow	GetVector
GMatrix	GS	GS2	GS3	
H				
HM	HM2	Hollow		
I				
Iden	Insert	INV	IP	IsColZero
IsDetOne	IsDetZero	IsDiag	IsDiagZero	IsElementBad
IsEqual	IsInvertible	IsOrthogonal	IsPivotZero	IsRowColZero
IsSchur	IsSelfDiagonal	IsSelfOrthonormal	IsSymm	IsUpperTriangular
J				
Jacobi	Jacobi2			
K				
KroneckerProd	Krlov			
L				
Lanczos	LDU	LeastSQ	LeastSQ2	LRGS

LRGS2	LRReflector	LRReflector2	LRRotator	LRUnsymm
LS	LU	LinSpace		
M				
M1	M2	M3	MatPro	MatrixClass
MC2	MKCMatrix	MKCMatrix2	MKCMatrix3	MKCMatrix4
MQ2				
N				
Neg				
O				
OP				
P				
PartMat2	PartMat3	PartMat4	PermCount	PivotPro
Polar	Poster	PowerMax	PowerMin	PR
PR_Modal_Parameter		PR_State_Space		PR2
PR3	PROJ	PROJ2	Purify	
Q				
QR				
R				
RandMatrix	RandMatrix2	Ranker	ReadTextFile	RedMatPro
Reflector	Reflector2	Remainder	ReMatPro	ReMatrix
ReRotator	Roots	Rotator		
S				
SchurToDiag	SetMatrix	SetMatrix2	SetPos	SOR
Special	Sub	Sum1	Sum10	Sum11
Sum12	Sum2	Sum3	Sum4	Sum5
Sum6	Sum7	Sum8	Sum9	SVD

SVD2	SVD3	SwapCol	SwapDiag	SwapRow
Sylvester	Sylvester2			
T				
TimeCal	TimeExpired	TimeExpired2	ToCompanion	ToEigenValue
ToMat	ToMatrixA	ToRow	ToSquare	ToVec
TP				
U				
UV				
V				
VectorOne	VERT	VERT2	XAB	XAB2
XLB				
Z				
Zero				

第五章 以 Python Matplotlib 套件輸出計算結果

第六章 實列應用

