第三章 矩陣計算運算子

一般我們所熟知的純量運算子，譬如加、減、乘、除、求餘等等，運算元是純量數值，矩陣計算的運算子與純量計算類似，運算元則是矩陣，兩者可相互比擬。然而矩陣與純量相比，是更複雜的資料結構，C#程式語言提供陣列(Array)資料結構，精銳矩陣計算器，是將陣列包在類別(Class)中，也就是 ReMatrix 類別和 CxMatrix 類別，分別處理實數和複數的矩陣，並提供各種矩陣的運算子，使得矩陣的計算，就像純量的數值計算一樣方便。

3.1 先將陣列轉爲矩陣

先由 C#程式語言的陣列 double[,]，再轉換為矩陣物件類別，實數矩陣類別為 ReMatrix、複數矩陣類別為 CxMatrix，就可以使用矩陣計算運算子。

實列 1

```
using System;
using Matrix_0;

namespace ConsoleApp33
{
    internal class Program
    {
        static void Main(string[] args)
        {
            double[,] A0 = { {-3, 6}, {9, -12} };
            Console.Write("\n A0 : 陣列\n{0}", new PR(A0));
            ReMatrix A = new ReMatrix(A0);
            Console.Write(" A : 矩陣\n{0}\n", new PR(A));
        }
    }
}
/* 輸出結果如下:
```

```
 A0 : 陣列
       -3.00000          6.00000
        9.00000        -12.00000
 A : 矩陣
       -3.00000          6.00000
        9.00000        -12.00000
*/
```

實例 2

```
using System;
using Matrix_0;

namespace ConsoleApp33
{
    internal class Program
    {
        static void Main(string[] args)
        {
            double[,] Are = { {-5, 3}, {-3, 2} };
            double[,] Aim = { { 5, -9 }, { -3, 8 } };
            CxMatrix A = new CxMatrix(Are, Aim);
            Console.Write(" A : 矩陣\n{0}\n", new PR(A));
        }
    }
}
/* 輸出結果如下:
 A : 矩陣
  -5.00000  +   5.00000i,    3.00000  -   9.00000i
  -3.00000  -   3.00000i,    2.00000  +   8.00000i
*/
```

3.2 加(+)、減(-)算術運算子

矩陣相加(+)，或是相減(-)。

實列 1

```
using System;
using Matrix_0;
```

```
namespace ConsoleApp33
{
    internal class Program
    {
        static void Main(string[] args)
        {
            double[,] Are = { {-5, 3}, {-3, 2}, {-5, 7} };
            double[,] Aim = { { 5, -9}, { -3, 8}, {5, 0} };
            CxMatrix A = new CxMatrix(Are, Aim);
            Console.Write(" A : 矩陣\n{0}\n", new PR(A));
            double[,] B0 = { { -4, 8}, { 9, -3}, {4, 9 } };
            ReMatrix B = new ReMatrix(B0);
            Console.Write(" B : 矩陣\n{0}\n", new PR(B));

            CxMatrix C = B + A + 2 * B - B;
            //印出矩陣C :
            Console.Write("\n C : 矩陣(即 2 * B + A)\n{0}\n", new
PR(C));
        }
    }
}
/* 輸出結果如下:
 A : 矩陣
  -5.00000  +   5.00000i,   3.00000  -    9.00000i
  -3.00000  -   3.00000i,   2.00000  +    8.00000i
  -5.00000  +   5.00000i,   7.00000  +    0.00000i
 B : 矩陣
  -4.00000      8.00000
   9.00000     -3.00000
   4.00000      9.00000
 C : 矩陣(即 2 * B + A)
 -13.00000  +    5.00000i,    19.00000  -    9.00000i
  15.00000  -    3.00000i,    -4.00000  +    8.00000i
   3.00000  +    5.00000i,    25.00000  +    0.00000i
*/
```

3.3　乘(*)、除(/)算算術運算子

矩陣相乘(*)和相除(/)，但需注意矩陣的列和行的個數，否則產生錯誤(Exeception)。

實列 1

```
using System;
using Matrix_0;

namespace ConsoleApp33
{
    internal class Program
    {
        static void Main(string[] args)
        {
            double[,] Are = { {-5, 3}, {-3, 2} };
            double[,] Aim = { { 5, -9}, { -3, 8} };
            CxMatrix A = new CxMatrix(Are, Aim);
            Console.Write(" A : 矩陣\n{0}\n", new PR(A));
            double[,] B0 = { { -4, 8}, { 9, -3} };
            ReMatrix B = new ReMatrix(B0);
            Console.Write(" B : 矩陣\n{0}\n", new PR(B));

            CxMatrix C = B + B / A;
            //印出矩陣C :
            Console.Write("\n C : 矩陣(即B + B * ~A)\n{0}\n", new
PR(C));
        }
    }
}
/* 輸出結果如下:
 A : 矩陣
  -5.00000  +   5.00000i,   3.00000  -   9.00000i
  -3.00000  -   3.00000i,   2.00000  +   8.00000i
 B : 矩陣
       -4.00000          8.00000
        9.00000         -3.00000
 C : 矩陣(即B + B * ~A)
 -3.93600  +    0.35200i,   8.08000  -    0.56000i
  7.74000  -    0.18000i,  -4.20000  -    0.60000i
```

*/

3.4 是否相等(==)、是否不相等(!=)邏輯運算子

矩陣是否相等(==)，是否不相等(!=)，傳回 true 或是 false，但是系統認定，絕對值小於 1.0 *10^{-6} 為 0。

實例 1

```
using System;
using Matrix_0;

namespace ConsoleApp33
{
    internal class Program
    {
        static void Main(string[] args)
        {
            double[,] A0 = { {-5}, {0} };
            ReMatrix A = new ReMatrix(A0);
            Console.Write(" A : 矩陣\n{0}\n", new PR(A));
            double[,] B0 = { {-5}, {0} };
            ReMatrix B = new ReMatrix(B0);
            Console.Write(" B : 矩陣\n{0}\n", new PR(B));

            if( A != B )
            { Console.Write(" *** A矩陣不等於B矩陣！ ***\n\n");}
            else
            { Console.Write(" *** A矩陣等於B矩陣！ ***\n\n");}

            double[,] C0 = { {-5}, {3.0 * 1E-6} };
            ReMatrix C = (ReMatrix)C0;
            Console.Write("\n C : 矩陣\n{0}\n", new PR(C));

            if (A == C)
            { Console.Write(" *** A矩陣等於C矩陣！ ***\n\n"); }
            else
            { Console.Write(" *** A矩陣不等於C矩陣！ ***\n\n"); }
        }
```

```
    }
}
/* 輸出結果如下:
 A : 矩陣
       -5.00000
        0.00000
 B : 矩陣
       -5.00000
        0.00000
 *** A矩陣等於B矩陣！ ***
 C : 矩陣
       -5.00000
        0.00000
 *** A矩陣不等於C矩陣！ ***
*/
```

3.5 水平合併(&)、垂直合併(|)運算子

兩個矩陣，若列相同時，可以水平合併為一個矩陣，若行相同時，可以垂直合併為一個矩陣。

實列 1

```
using System;
using Matrix_0;

namespace ConsoleApp33
{
    internal class Program
    {
        static void Main(string[] args)
        {
            double[,] Are = { {-3, 6}, {5, 9} };
            double[,] Aim = { {-5, 0}, {4, -2 } };
            CxMatrix A = new CxMatrix(Are, Aim);
            Console.Write("\n A矩陣 :\n{0}\n", new PR(A));

            double[,] B0 = { {4, -4}, {1, -9 } };
            ReMatrix B = new ReMatrix(B0);
```

```
            Console.Write("\n B矩陣 : \n{0}", new PR(B));

            CxMatrix C = B & A;
            Console.Write("\n C矩陣(水平合併) :\n{0}", new PR(C));

            CxMatrix D = A | B;
            Console.Write("\n D矩陣(垂直合併) :\n{0}", new PR(D));
        }
    }
}
/*輸出結果如下:
 A矩陣 :
  -3.00000  -   5.00000i,    6.00000  +   0.00000i
   5.00000  +   4.00000i,    9.00000  -   2.00000i
 B矩陣 :
        4.00000          -4.00000
        1.00000          -9.00000
 C矩陣(水平合併) :
   4.00000  +   0.00000i,  -4.00000  +   0.00000i,
  -3.00000  -   5.00000i,    6.00000  +   0.00000i

   1.00000  +   0.00000i,  -9.00000  +   0.00000i,
   5.00000  +   4.00000i,    9.00000  -   2.00000i
 D矩陣(垂直合併) :
  -3.00000  -   5.00000i,    6.00000  +   0.00000i
   5.00000  +   4.00000i,    9.00000  -   2.00000i
   4.00000  +   0.00000i,  -4.00000  +   0.00000i
   1.00000  +   0.00000i,  -9.00000  +   0.00000i
*/
```

3.6 向量內積(^)運算子

向量內積是計算向量的長度，向量 a 和 b 的內積以 a ^ b 表示，一般線性代數以<a, b>表示，但這種表示方式並不適合程式語言的表示方式，向量可以是實數也可以是複數，複數可以包含實數。a ^ b = bh * a。bh 變數的最後字母 h 代表 b 向量的 Hermitian，亦即 b 向量的轉置再取共軛複數。

a ^ b 的物理意義，是 a 向量投影在 b 向量後，再乘 b 向量，或是 b 向量投

影在 a 向量後，再乘 a 向量。

實例 1

```
using System;
using Matrix_0;

namespace ConsoleApp33
{
    internal class Program
    {
        static void Main(string[] args)
        {
            double[,] a0 = { {-5.4133}, {7}, {-2} };
            ReMatrix a = new ReMatrix(a0);
            Console.Write("\n a 向量 ：\n{0}", new PR(a));
            double[,] b0 = { { -3 }, { 9 }, {1.93} };
            ReMatrix b = new ReMatrix(b0);
            Console.Write("\n b 向量 :\n{0}", new PR(b));

            ReMatrix c = a ^ b;
            Console.Write("\n c 向量(R1x1的矩陣): \n{0}", new PR(c));

            double d = c.GetValue;
            Console.Write("\n d(純量數值) = {0,-15:F7}\n\n", d);
        }
    }
}
/*輸出結果如下:
 a 向量 ：
       -5.41330
        7.00000
       -2.00000
 b 向量 :
       -3.00000
        9.00000
        1.93000
 c 向量(R1x1的矩陣):
       75.37990
```

```
 d(純量數值) = 75.3799000
*/
```

實例 2

```
using System;
using Matrix_0;

namespace ConsoleApp33
{
    internal class Program
    {
        static void Main(string[] args)
        {
            double[,] aReal = { {1.3}, {17}, {-2} };
            double[,] aImage = { {-4.46}, {0}, {3.9} };
            CxMatrix a = new CxMatrix(aReal, aImage);
            Console.Write("\n a 向量 ：\n{0}", new PR(a));

            CxMatrix b = a ^ a;
            Console.Write("\n a^a (C1X1) : \n{0}", new PR(b));

            ReMatrix c = (ReMatrix)b;
            double d = c.GetValue;
            d = Math.Sqrt(d);
            Console.Write($"\n a 複數向量的長度 = {d} \n\n");
        }
    }
}
/*輸出結果如下:
 a 向量 ：
        1.30000  -         4.46000i
       17.00000  +         0.00000i
       -2.00000  +         3.90000i
 a^a (C1X1) :
      329.79160  +         0.00000i
 a 複數向量的長度 = 18.1601651974865
*/
```

3.7 逆算(~)運算子

逆矩陣存在的條件是，矩陣應爲正方形且行列式不爲 0。

實列 1

```
using System;
using Matrix_0;

namespace ConsoleApp33
{
    internal class Program
    {
        static void Main(string[] args)
        {
            double[,] Are = { { 0, 5.9, -0.45 },
                {4.7, 7.9, 6.9 }, {-2.4, 5.1, 0.7} };
            double[,] Aim = { {0, -9.3, 0.3},
                {8.3, 7.3, 7.1},{3.9, 7.3, 5.0} };
            CxMatrix A = new CxMatrix(Are, Aim);
            Console.Write("\n 複數矩陣A ：\n{0}", new PR(A));

            CxMatrix B = ~A;
            Console.Write("\n 複數矩陣B(即矩陣A的逆矩陣) :\n{0}",
            new PR(B));

            // 檢核矩陣B是否為矩陣A的逆矩陣？
            CxMatrix C = A * B;
            Console.Write("\n 複數矩陣C :\n{0}", new PR(C));

            // 雖然複數矩陣C為Identity矩陣，但矩陣C內的虛元素大於
            // 系統允許的1.0E-6，故無法轉換為0，產生Exception。
            ReMatrix D = (ReMatrix)A;
            Console.Write("\n 實數矩陣D :\n{0}", new PR(D));
        }
    }
}
/*輸出結果如下:
```

```
 複數矩陣A ：
  0.00000 +  0.00000i,   5.90000 -  9.30000i,  -0.45000 +  0.30000i
  4.70000 +  8.30000i,   7.90000 +  7.30000i,   6.90000 +  7.10000i
 -2.40000 +  3.90000i,   5.10000 +  7.30000i,   0.70000 +  5.00000i
 複數矩陣B(即矩陣A的逆矩陣）：
  0.10747 +  0.13719i,   0.23651 +  0.02051i,  -0.38972 +  0.22480i
  0.04756 +  0.06432i,  -0.00354 -  0.00907i,   0.02009 +  0.00267i
 -0.12207 -  0.22107i,  -0.14068 -  0.13959i,   0.39610 -  0.11622i
 複數矩陣C ：
  1.00000 +  0.00000i,   0.00000 +  0.00000i,   0.00000 +  0.00000i
  0.00000 +  0.00000i,   1.00000 +  0.00000i,   0.00000 +  0.00000i
  0.00000 +  0.00000i,   0.00000 +  0.00000i,   1.00000 +  0.00000i

***  Cannot convert from CxMatrix to ReMatrix  ***

***  Stop Execution!  ***
*/
```

3.8 轉置(!)運算子

矩陣的轉置運算子為"!"，是 Unitary 運算子(驚嘆號)，而兩個矩陣的垂直合併的運算子為"|"，是 Binary 運算子，兩者是不同的運算子。

實例 1

```
using System;
using Matrix_0;

namespace ConsoleApp33
{
    internal class Program
    {
        static void Main(string[] args)
        {
            double[,] Are = { {0, 5.9}, {4.7, 7.9}, {-2.4, 5.1} };
            double[,] Aim = { {0, -9.3},{8.3, 7.3}, {3.9, 7.3} };
            CxMatrix A = new CxMatrix(Are, Aim);
            Console.Write("\n 複數矩陣A ：\n{0}", new PR(A));
```

```
            CxMatrix B = !A;
            Console.Write("\n 複數矩陣B(即矩陣A的轉置矩陣) :\n{0}",
                new PR(B));
        }
    }
}
/*輸出結果如下:
 複數矩陣A  :
  0.00000  +  0.00000i,  5.90000  -   9.30000i
  4.70000  +  8.30000i,  7.90000  +   7.30000i
 -2.40000  +  3.90000i,  5.10000  +   7.30000i
 複數矩陣B(即矩陣A的轉置矩陣) :
  0.00000 +  0.00000i,  4.70000 +  8.30000i,  -2.40000 +  3.90000i
  5.90000 -  9.30000i,  7.90000 +  7.30000i,   5.10000 +  7.30000i
*/
```

3.9 單位向量(+)運算子

單位向量運算子為"+"，即 Unitary 運算子。請不要與 Binary 運算子"+"混肴，若是兩個數值相加，或是兩個向量相加，或是兩個矩陣相加，則使用 Binary 運算子"+"。

實列 1

```
using System;
using Matrix_0;

namespace ConsoleApp33
{
    internal class Program
    {
        static void Main(string[] args)
        {
            double[,] Are = { {10}, {47}, {24} };
            double[,] Aim = { {5}, {8}, {9} };
            CxMatrix A = new CxMatrix(Are, Aim);
            Console.Write("\n 複數向量A  :\n{0}", new PR(A));
```

```
            CxMatrix B = +A;
            Console.Write("\n 複數向量B(即複數向量A的單位向
量) :\n{0}",
                new PR(B));

            // 以下檢核複數向量B是否為單位向量。
            CxMatrix C = !B * B;
            CxScalar d = C.GetCxScalar;
            Console.Write("\n 複數純量d :\n{0}\n", new PR(d));
            double e = d.ModuleVal;
            Console.Write("\n 複數模數= {0} \n\n", e);
        }
    }
}
/*輸出結果如下:
 複數向量A  :
       10.00000  +         5.00000i
       47.00000  +         8.00000i
       24.00000  +         9.00000i
 複數向量B(即複數向量A的單位向量) :
        0.18092  +         0.09046i
        0.85034  +         0.14474i
        0.43422  +         0.16283i
 複數純量d :
        0.88871  +         0.42029i,
 複數模數= 0.983080743744007
*/
```

實例 2

```
using System;
using Matrix_0;

namespace ConsoleApp33
{
    internal class Program
    {
        static void Main(string[] args)
```

```
        {
            double[,] A0 = { {10}, {47}, {24} };
            ReMatrix A = (ReMatrix)A0;
            Console.Write("\n 實數數向量A ：\n{0}", new PR(A));

            ReMatrix B = +A;
            Console.Write("\n 實數向量B(即向量A的單位向量) :\n{0}",
                new PR(B));

            // 以下檢核向量B是否為單位向量。
            // 請多多利用Visual Studio的智慧感知器IntelliSense，
            // 瞭解class的元素，如屬性、方法、索引子、等等。
            ReMatrix C = !B * B;
            // 印出R1x1的矩陣。
            Console.Write("\n C :\n{0}\n", new PR(C));
            // 印出R1x1的數據。
            double d = C[0, 0].GetValue;
            Console.Write("\n C[0, 0]的數值 = {0,10:F5}\n\n\n", d);
        }
    }
}
/*輸出結果如下:
 實數數向量A ：
       10.00000
       47.00000
       24.00000
 實數向量B(即向量A的單位向量) :
        0.18618
        0.87503
        0.44683
 C :
        1.00000
 C[0, 0]的數值 =    1.00000
*/
```