

第四章 類別程式庫

精銳矩陣計算求解器類別(Class)約有二百多個，列在下面供參考，有補助性質類別、矩陣的分解類別、矩陣合併類別、矩陣的對角化類別、求取對稱與非對稱矩陣的特徵值和特徵向量(複數矩陣)、特徵值和特徵向量相乘的運算，奇異值矩陣的分解、多個自由度和多階微分方程式的求解，包含初始值和邊界值的求解等等。至於類別的方法、屬性等元素，可以使用智慧型的感知器(IntelliSense)查詢，即輸入類別名稱再加“.”，可顯示類別元素內容。

精銳矩陣計算類別程式庫：

Add1	Add2	Add3	Affine	Arnoldi	AUG
AUG2	Arange				
CG	ChkNullMat	ChkSqMat	CHOL	class1	
ClearZeroCol	ComMat	ComMat2	CoorMat	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	ComMat3	CxPR	CxPR2	CxPROJ
CxRotator	CxScalar	CxSetMatrix2		CxSwapRow	CxToDexp
CxToX	CxToHexp	CxTP			
Deflator	Descent	DET	DET2	DET3	DET4
DETZero	DirectSum				
EIG	EIG2	EIG3	EIG4	EntryMax	ErrValue
EV	EVCm				
Gauss	GaussJordan		GaussJordan2		
GaussSeidel	GetMatrixQ	GetMatrixQ2		GetMatrixQ3	
GetMatrixQ4		GetRow	GetVector	GetMatrix	GS
GS2	GS3				
HM	HM2	Hollow			
Iden	Insert	INV	IP	IsColZero	IsDetOne
IsDetZero	IsDiag	IsDiagZero	IsElementBad		IsEqual
IsInvertible	IsOrthogonal		IsPivotZero	IsRowColZero	
IsSchur	IsSelfDiagonal		IsSelfOrthonormal		IsSymm
IsUpperTriangular					
Jacobi	Jacobi2				

KroneckerProd		Krlov			
Lanczos	LDU	LeastSQ	LeastSQ2	LRGS	LRGS2
LRReflector	LRReflector2	LRRotator	LRUnsymm	LS	LU
LinSpace					
M1	M2	M3	MatPro	MatrixClass	MC2
MKCMatrix	MCKMatrix2	MKCMatrix3	MKCMatrix4	MQ2	
Neg	OP				
PartMat2	PartMat3	PartMat4	PermCount	PivotPro	Polar
Poster	PowerMax	PowerMin	PR	PR_Model_Parameter	
PR_State_Space		PR2	PR3	PROJ	PROJ2
Purify	QR				
RandMatrix	RandMatrix2	Ranker	ReadTextFile	RedMatPro	Reflector
Reflector2	ReMainder	ReMatPro	ReMatrix	ReRotator	Roots
Rotator	RowSlice				
SchurToDiag	SetMatrix	SetMatrix2	SetPos	SOR	Sum1
Sum2	Sum3	Sum4	Sum5	Sum6	Sum7
Sum8	Sum9	Sum10	Sum11	Sum12	SVD
SVD2	SVD3	SwapCol	SwapDiag	SwapRow	Sylvester
Sylvester2	Special	Sub			
TimeCal	ToCompanion				
ToEigenValue	TP	ToDexp	ToMat	ToMatrixA	ToRow
ToSquare	ToVec	ToHexp	UV		
VectorOne	VERT	VERT2	XAB	XAB2	XLB
XUB	Zero				

以下是將比較常用到的類別，分數個群組並以實例說明，並將輸出結果列於後。

4.1 第一群組類別：

C#程式語言的陣列、實數矩陣(ReMatrix)、複數矩陣(CxMatrix)、複數純量(CxScalar)、矩陣列印(PR)。

實例 1

```
using System;
using Matrix_0;

namespace ConsoleApp33
{
```

```

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

        double[, ] A = { {5, 8}, {4.5, 7.9}, {1.53, 8} };
        Console.WriteLine("\n 陣列A[2, 0]的元素 = {0,10:F3}\n\n", A[2, 0]);
        Console.WriteLine("\n 陣列A[1, 1]的元素 = {0}\n\n",
            A[1, 1].ToString("F5"));
    }
}

```

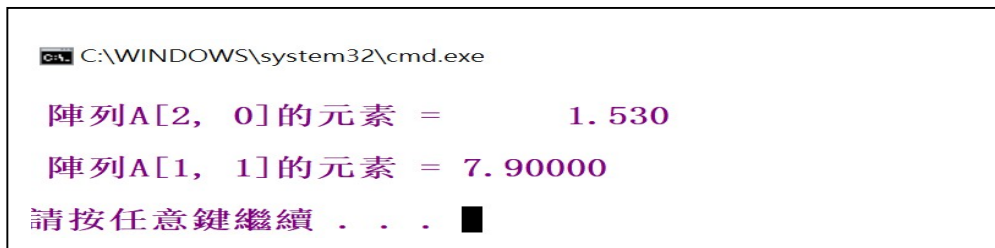
/*輸出結果如下：

陣列A[2, 0]的元素 = 1.530

陣列A[1, 1]的元素 = 7.90000

*/

螢幕快照如下：



```

C:\WINDOWS\system32\cmd.exe

陣列A[2, 0]的元素 = 1.530
陣列A[1, 1]的元素 = 7.90000
請按任意鍵繼續 . . . 

```

實例 2

```
using System;
```

```
using Matrix_0;
```

```
namespace ConsoleApp33
```

```
{
```

```
    internal class Program
```

```
    {
```

```
        static void Main(string[] args)
```

```
        {
```

```
double[, ] A_array = { {3.5, 4.9, 5}, {-4.8, 4.1, 7} };
```

```
ReMatrix A = new ReMatrix(A_array);
```

```
// 如何顯示Class A的元素？可以輸入"A"，再加入".",
```

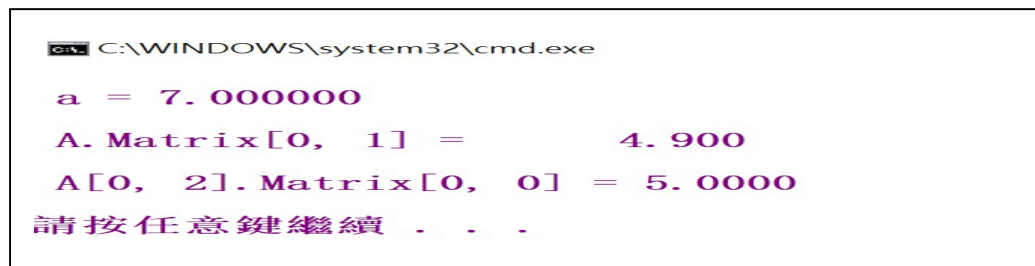
```
// 由智慧型感知器(IntelliSense)可以顯示出來。
double a = A[1, 2].GetValue;
Console.WriteLine("\n a = {0}\n", a.ToString("F6"));
double b = A.Matrix[0, 1];
Console.WriteLine("\n A.Matrix[0, 1] = {0,10:F3}\n", b);
double c = A[0, 2].Matrix[0, 0];
Console.WriteLine("\n A[0, 2].Matrix[0, 0] = {0}\n\n", c.ToString("F4"));
    }
}
}
```

/*輸出結果如下：

```
a = 7.000000
A.Matrix[0, 1] =      4.900
A[0, 2].Matrix[0, 0] = 5.0000
```

*/

螢幕快照如下：



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

a = 7.000000
A.Matrix[0, 1] =      4.900
A[0, 2].Matrix[0, 0] = 5.0000
請按任意鍵繼續 . . .
```

實例 3

```
using System;
```

```
using Matrix_0;
```

```
namespace ConsoleApp33
```

```
{
```

```
    internal class Program
```

```
    {
```

```
        static void Main(string[] args)
```

```
        {
```

```
double[,] Are = { { -3, 4, 6 }, { 6, 4, 3 }, { 2, -3, 6 } };
```

```
double[,] Aim = { { 4, -3, -5 }, { -2, 4, 9 }, { 0, -4, 8 } };
```

```
CxMatrix A = new CxMatrix(Are, Aim);
```

```
// 列印複數矩陣A
```

```

Console.Write("\n 複數矩陣A :\n{0}", new PR(A));
// A[2, 2] = ?
CxScalar a = A[2, 2].GetCxScalar;
Console.Write("\n a = A[2, 2] :\n{0}", new PR(a));
// CxMatrix類別包含實數矩陣Re和虛數矩陣Im，其元素為[2, 2]
double b = A.Re[2, 2];
double c = A.Im[2, 2];
Console.Write($" \n A[2, 2] 實數 = {b} 複數 = {c} \n\n");
    }
}
}

```

/*輸出結果如下：

複數矩陣A：

```

-3.00000 + 4.00000i, 4.00000 - 3.00000i, 6.00000 - 5.00000i
6.00000 - 2.00000i, 4.00000 + 4.00000i, 3.00000 + 9.00000i
2.00000 + 0.00000i, -3.00000 - 4.00000i, 6.00000 + 8.00000i

```

a = A[2, 2]：

6.00000 + 8.00000i,

A[2, 2] 實數 = 6 複數 = 8

*/

螢幕快照如下：

```

C:\WINDOWS\system32\cmd.exe
複數矩陣A：
-3.00000 + 4.00000i, 4.00000 - 3.00000i, 6.00000 - 5.00000i
6.00000 - 2.00000i, 4.00000 + 4.00000i, 3.00000 + 9.00000i
2.00000 + 0.00000i, -3.00000 - 4.00000i, 6.00000 + 8.00000i

a = A[2, 2]：
6.00000 + 8.00000i,

A[2, 2] 實數 = 6 複數 = 8
請按任意鍵繼續 . . . █

```

實例 4

```
using System;
```

```
using Matrix_0;
```

```
namespace ConsoleApp33
```

```
{
```

```
    internal class Program
```

```

    {
        static void Main(string[] args)
        {
            // 使用PR Class作純量複數、實數矩陣、複數矩陣的列印。

            CxScalar a = new CxScalar(5, -9);
            Console.WriteLine("\n*** 列印純量複數 5 - 9i ***\n{0}", new PR(a));

            double[,] Re = { {4, 5, -4}, {3, -5, -9}, {-2, 4, 6} };
            double[,] Im = { {-2, 5, -2}, {4, -2, -2}, {-7, -3, 1} };
            CxMatrix A = new CxMatrix(Re, Im);
            Console.WriteLine("\n*** 列印複數矩陣 ***\n{0}", new PR(A));
        }
    }
}

```

/*輸出結果如下：

```

*** 列印純量複數 5 - 9i ***
      5.00000 -      9.00000i,
*** 列印複數矩陣 ***
      4.00000 - 2.00000i,      5.00000 + 5.00000i,      -4.00000 - 2.00000i
      3.00000 + 4.00000i,      -5.00000 - 2.00000i,      -9.00000 - 2.00000i
      -2.00000 - 7.00000i,      4.00000 - 3.00000i,      6.00000 + 1.00000i
*/

```

螢幕快照如下：

```

C:\WINDOWS\system32\cmd.exe
*** 列印純量複數 5 - 9i ***
      5.00000 -      9.00000i,

*** 列印複數矩陣 ***
      4.00000 - 2.00000i,      5.00000 + 5.00000i,      -4.00000 - 2.00000i
      3.00000 + 4.00000i,      -5.00000 - 2.00000i,      -9.00000 - 2.00000i
      -2.00000 - 7.00000i,      4.00000 - 3.00000i,      6.00000 + 1.00000i

請按任意鍵繼續 . . . █

```

4.2 第二群組類別：

行列式(DET, CxDET)、向量內積 (IP, CxIP)、逆矩陣(INV, CxINV)、轉置(TP, CxTP, CxHerm)。

實例 1

```
using System;
```

```
using Matrix_0;
```

```
namespace ConsoleApp33
```

```
{
```

```
    internal class Program
```

```
    {
```

```
        static void Main(string[] args)
```

```
        {
```

```
            double[, ] A0 = { {20, 0, -1}, {0, 15, -2}, {-1, -2, 13} };
```

```
            Console.WriteLine("\n 矩陣A0 :\n{0}", new PR(A0));
```

```
            DET dt = new DET(A0);
```

```
            double val = dt.Value;
```

```
            Console.WriteLine("\n 行列式 = {0}\n\n", val.ToString("F3"));
```

```
            double[, ] Are = { {-2.3, 7.9, 6.5}, {3.5, 6.9, 4}, {3.7,  
6, -8} };
```

```
            double[, ] Aim = { {5.8, 4.9, 8.3}, {3.6, 7.9, 4}, {2.5,  
5.8, 6} };
```

```
            CxMatrix A = new CxMatrix(Are, Aim);
```

```
            Console.WriteLine("\n 複數矩陣A :\n{0}", new PR(A));
```

```
            CxDET cxDet = new CxDET(A);
```

```
            CxScalar cxVal = cxDet.GetCxScalar;
```

```
            Console.WriteLine("\n 行列式 :\n{0}\n", new PR(cxVal));
```

```
            double val2 = cxDet.GetModuleVal;
```

```
            Console.WriteLine("\n 複數矩陣模數 = {0,10:F5}\n\n", val2);
```

```
        }
```

```
    }
```

```
}
```

/*輸出結果如下:

矩陣A0 :

20.00000	0.00000	-1.00000
0.00000	15.00000	-2.00000

$$\begin{vmatrix} 20.00000 & 0.00000 & -1.00000 \\ 0.00000 & 15.00000 & -2.00000 \\ -1.00000 & -2.00000 & 13.00000 \end{vmatrix}$$
 行列式 = 3805.000
 複數矩陣A :

$$\begin{bmatrix} -2.30000 + 5.80000i & 7.90000 + 4.90000i & 6.50000 + 8.30000i \\ 3.50000 + 3.60000i & 6.90000 + 7.90000i & 4.00000 + 4.00000i \\ 3.70000 + 2.50000i & 6.00000 + 5.80000i & -8.00000 + 6.00000i \end{bmatrix}$$
 行列式 :

$$909.26400 + 6.33200i,$$
 複數矩陣模數 = 909.28605

*/

螢幕快照：

```

C:\WINDOWS\system32\cmd.exe

矩陣A0 :
20.00000      0.00000     -1.00000
0.00000      15.00000     -2.00000
-1.00000     -2.00000      13.00000

行列式 = 3805.000

複數矩陣A :
-2.30000 + 5.80000i, 7.90000 + 4.90000i, 6.50000 + 8.30000i
3.50000 + 3.60000i, 6.90000 + 7.90000i, 4.00000 + 4.00000i
3.70000 + 2.50000i, 6.00000 + 5.80000i, -8.00000 + 6.00000i

行列式 :
909.26400 + 6.33200i,

複數矩陣模數 = 909.28605

請按任意鍵繼續 . . .
  
```

實例 2

using System;

using Matrix_0;

namespace ConsoleApp33

{

internal class Program

{

static void Main(string[] args)

{

double[,] A0 = { {5.7}, {4.9}, {-3.1} };

ReMatrix A = new ReMatrix(A0);

Console.WriteLine(" A矩陣(向量) :\n{0}", new PR(A));


```

double[, ] B0 = { {-4}, {5.2}, {3} };
ReMatrix B = new ReMatrix(B0);
Console.WriteLine(" B向量 : \n{0}", new PR(B));

IP ip = new IP(A, B);
double val = ip.Value;
Console.WriteLine("\n 內積 = {0,10:F4}\n", val);

ReMatrix C = A ^ B;
Console.WriteLine("\n A^B 是1x1的矩陣(兩者相同): \n{0}", new
PR(C));
    }
}
}

```

/*輸出結果如下:

A矩陣(向量) :

```

5.70000
4.90000
-3.10000

```

B向量 :

```

-4.00000
5.20000
3.00000

```

內積 = -6.6200

A^B 是1x1的矩陣(兩者相同):

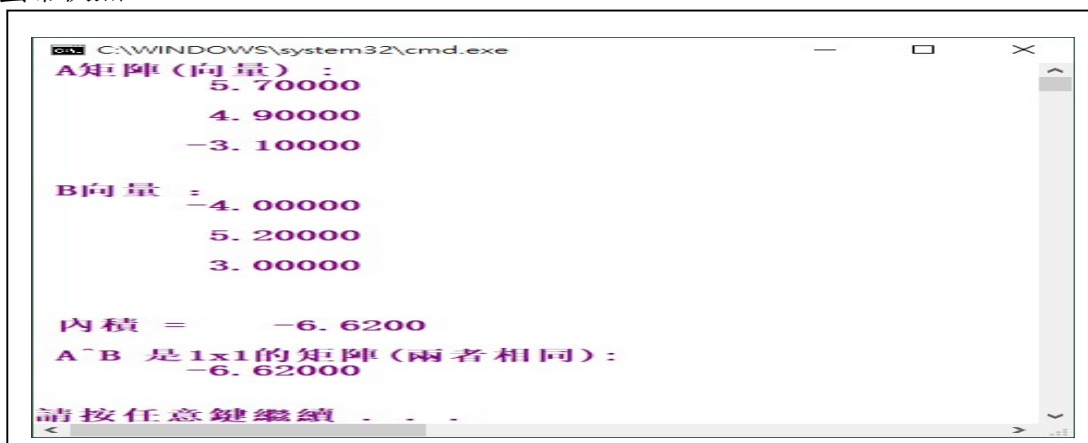
```

-6.62000

```

*/

螢幕快照



實例 3

```
using System;
using Matrix_0;

namespace ConsoleApp33
{
    internal class Program
    {
        static void Main(string[] args)
        {
            double[, ] Are = { {5.7}, {4.9}, {-3.1} };
            double[, ] Aim = { {-3.6}, {4.9}, {1.5} };
            CxMatrix A = new CxMatrix(Are, Aim);
            Console.WriteLine(" A向量 :\n{0}", new PR(A));

            double[, ] Bre = { {-4}, {5.2}, {3} };
            double[, ] Bim = { {4.2}, {5}, {4} };
            CxMatrix B = new CxMatrix(Bre, Bim);
            Console.WriteLine(" B向量 :\n{0}", new PR(B));

            CxIP ip = new CxIP(A, B);
            CxScalar val = ip.GetCxScalar;
            Console.WriteLine(" 內積 :\n{0}", new PR(val));

            CxMatrix C = ip.GetCxMatrix;
            Console.WriteLine(" C1x1複數矩陣 :\n{0}", new PR(C));
        }
    }
}
```

/*輸出結果如下：

A矩陣(向量)：

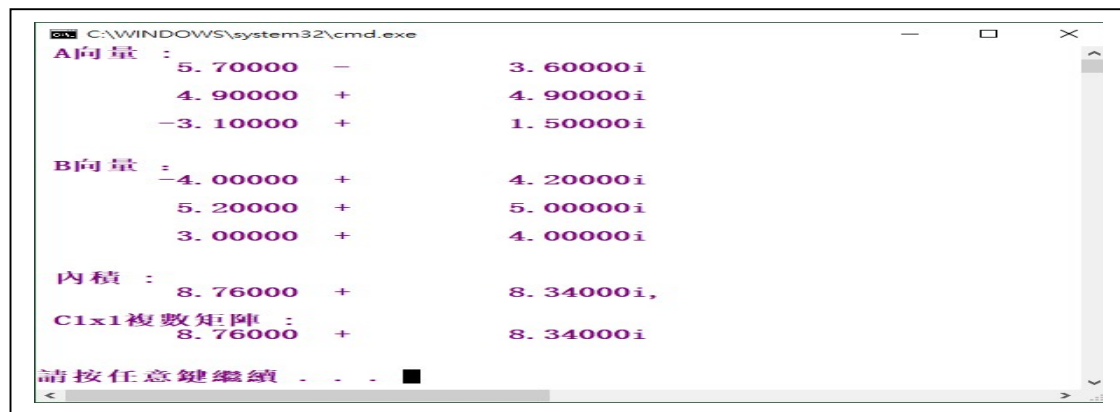
5.70000	-	3.60000i
4.90000	+	4.90000i
-3.10000	+	1.50000i

B向量：

-4.00000	+	4.20000i
----------	---	----------

$$\begin{matrix} 5.20000 & + & 5.00000i \\ 3.00000 & + & 4.00000i \end{matrix}$$
 內積：

$$\begin{matrix} 8.76000 & + & 8.34000i, \\ \text{C1x1複數矩陣：} \\ 8.76000 & + & 8.34000i \end{matrix}$$
 */
 螢幕快照：



實例 4

```

using System;
using Matrix_0;

namespace ConsoleApp33
{
    internal class Program
    {
        static void Main(string[] args)
        {
            double[,] A0 = { {5, -2, 3}, {-3, 6, 9 }, {-3, 9, 9} };
            ReMatrix A = new ReMatrix(A0);
            Console.WriteLine("\n 矩陣A :\n{0}", new PR(A));

            INV inv = new INV(A);
            ReMatrix Ai = inv.Matrix;
            Console.WriteLine("\n 矩陣Ai :\n{0}", new PR(Ai));
        }
    }
}
  
```

```

double[,] A1 = { {-5, 5, 4}, {4.5, 8, 9}, {1, -4, 3.9} };
CxMatrix B = new CxMatrix(A0, A1);
Console.WriteLine("\n 複數矩陣B :\n{0}", new PR(B));

CxINV cxInv = new CxINV(B);
CxMatrix Bi = cxInv.GetCxMatrix;
Console.WriteLine("\n 矩陣Bi :\n{0}", new PR(Bi));

CxMatrix C = B * Bi;
Console.WriteLine("\n 複數矩陣C :\n{0}", new PR(C));

ReMatrix D = (ReMatrix)C;
Console.WriteLine("\n 將複數矩陣C，轉為實數矩陣D :\n{0}", new
PR(D));
    }
}
}

```

/*輸出結果如下：

矩陣A：

5.00000	-2.00000	3.00000
-3.00000	6.00000	9.00000
-3.00000	9.00000	9.00000

矩陣Ai：

0.16667	-0.27778	0.22222
0.00000	-0.33333	0.33333
0.05556	0.24074	-0.14815

複數矩陣B：

5.00000 - 5.00000i,	-2.00000 + 5.00000i,	3.00000 + 4.00000i
-3.00000 + 4.50000i,	6.00000 + 8.00000i,	9.00000 + 9.00000i
-3.00000 + 1.00000i,	9.00000 - 4.00000i,	9.00000 + 3.90000i

矩陣Bi：

0.05873 + 0.07070i,	-0.02501 - 0.06557i,	-0.00925 + 0.04492i
-0.04092 + 0.00087i,	-0.02727 - 0.06247i,	0.02145 + 0.09835i
0.05648 - 0.02649i,	0.04459 + 0.01195i,	0.01373 - 0.07876i

複數矩陣C：

1.00000 + 0.00000i,	0.00000 + 0.00000i,	0.00000 + 0.00000i
---------------------	---------------------	--------------------

$$\begin{matrix} 0.00000 + 0.00000i, & 1.00000 + 0.00000i, & 0.00000 + 0.00000i \\ 0.00000 + 0.00000i, & 0.00000 + 0.00000i, & 1.00000 + 0.00000i \end{matrix}$$

將複數矩陣C，轉為實數矩陣D：

1.00000	0.00000	0.00000
0.00000	1.00000	0.00000
0.00000	0.00000	1.00000

*/

螢幕快照：

```

C:\WINDOWS\system32\cmd.exe
矩陣A：
5.00000 -2.00000 3.00000
-3.00000 6.00000 9.00000
-3.00000 9.00000 9.00000

矩陣Ai：
0.16667 -0.27778 0.22222
0.00000 -0.33333 0.33333
0.05556 0.24074 -0.14815

複數矩陣B：
5.00000 - 5.00000i, -2.00000 + 5.00000i, 3.00000 + 4.00000i
-3.00000 + 4.50000i, 6.00000 + 8.00000i, 9.00000 + 9.00000i
-3.00000 + 1.00000i, 9.00000 - 4.00000i, 9.00000 + 3.90000i

矩陣Bi：
0.05873 + 0.07070i, -0.02501 - 0.06557i, -0.00925 + 0.04492i
-0.04092 + 0.00087i, -0.02727 - 0.06247i, 0.02145 + 0.09835i
0.05648 - 0.02649i, 0.04459 + 0.01195i, 0.01373 - 0.07876i

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

將複數矩陣C，轉為實數矩陣D：
1.00000 0.00000 0.00000
0.00000 1.00000 0.00000
0.00000 0.00000 1.00000

請按任意鍵繼續 . . . █
  
```

實例 5：

using System;

using Matrix_0;

namespace ConsoleApp33

{

```

internal class Program
{
    static void Main(string[] args)
    {
        double[, ] A0 = { { -5, -4}, {0, 4} };
        ReMatrix A = (ReMatrix)A0;
        Console.WriteLine("\n 矩陣A :\n{0}", new PR(A));

        TP tp = new TP(A);
        ReMatrix At = tp.Matrix;
        Console.WriteLine("\n A的轉置矩陣 :\n{0}", new PR(At));

        double[, ] Bre = { { -4, 6}, {-5, 8} };
        double[, ] Bim = { {0, 4}, {7, 0} };
        CxMatrix B = new CxMatrix(Bre, Bim);
        Console.WriteLine("\n 複數矩陣B :\n{0}", new PR(B));

        CxHerm Her = new CxHerm(B);
        CxMatrix Bh = Her.GetCxMatrix;
        Console.WriteLine("\n Hermitian矩陣Bh :\n{0}", new PR(Bh));
    }
}

```

/*輸出結果如下：

矩陣A :

-5.00000	-4.00000
0.00000	4.00000

A的轉置矩陣 :

-5.00000	0.00000
-4.00000	4.00000

複數矩陣B :

-4.00000 +	0.00000i,	6.00000 +
4.00000i		
-5.00000 +	7.00000i,	8.00000 +
0.00000i		

Hermitian矩陣Bh :

-4.00000 +	0.00000i,	-5.00000 -
------------	-----------	------------

```

7.00000i
      6.00000 -      4.00000i,      8.00000 +
0.00000i
*/

```

4.3 第三群組類別：

線分割段(Arange, LinSpace)，係參考 Python 的 `arange()`和 `linspace()`兩函式而建構，兩者最大差別是 Arange 和 LinSpace 是屬於一維陣列物件，而不是函式，且可隱性轉為二維陣列。

實例 1

```
using System;
```

```
using Matrix_0;
```

```
namespace ConsoleApp33
```

```
{
```

```
    internal class Program
```

```
    {
```

```
        static void Main(string[] args)
```

```
        {
```

```
            Arange range = new Arange(0, 1, 0.1);
```

```
            ReMatrix A = range.GetArray;
```

```
            Console.WriteLine("\n 陣列A : \n{0}", new PR(A));
```

```
        }
```

```
    }
```

```
}
```

```
/*輸出結果如下：
```

```
陣列A：
```

```

0.00000  0.10000  0.20000  0.30000  0.40000  0.50000
0.60000  0.70000  0.80000  0.90000  1.00000

```

```
*/
```

實例 2

```
using System;
```

```
using Matrix_0;
```

```

namespace ConsoleApp33
{
    internal class Program
    {
        static void Main(string[] args)
        {
            LinSpace space = new LinSpace(0, 1, 50);
            ReMatrix A = space.GetArray;
            Console.WriteLine("\n 陣列A : \n{0}", new PR(A));
        }
    }
}

```

/*輸出結果如下：

陣列A：

0.00000	0.02000	0.04000	0.06000	0.08000	0.10000
0.12000	0.14000	0.16000	0.18000	0.20000	0.22000
0.24000	0.26000	0.28000	0.30000	0.32000	0.34000
0.36000	0.38000	0.40000	0.42000	0.44000	0.46000
0.48000	0.50000	0.52000	0.54000	0.56000	0.58000
0.60000	0.62000	0.64000	0.66000	0.68000	0.70000
0.72000	0.74000	0.76000	0.78000	0.80000	0.82000
0.84000	0.86000	0.88000	0.90000	0.92000	0.94000
0.96000	0.98000	1.00000			

*/

4.4 第四群組類別：

由微分方程式建構系統矩陣(MCKMatrix, ToCompanion, Roots 類別)

在這裏所稱的系統矩陣指的是將多個小型的矩陣，或是微分方程式，合併成一個大型的矩陣，亦即整個系統可用此矩陣表示。

$M * y''(t) + C * y'(t) + K * y(t) = 0$ ， M 、 C 、 K 、是實數 $m \times m$ 的矩陣，則其系統矩陣 A 為：

$$A = \begin{bmatrix} -M^{-1} * C & -M^{-1} * K \\ I & 0 \end{bmatrix}$$

M^{-1} 表示 M 矩陣之逆矩陣， I 是 $m \times m$ 單位矩陣(Iden 類別)， 0 是 $m \times m$ 零矩陣

(Zero 類別)，故矩陣 A 是 $(m \times n) \times (m \times n)$ 的大型實數矩陣。 n 是階數(Order)，故上

式 $n=2$ ，即系統矩陣 A 是 $(2m) \times (2m)$ 的矩陣。矩陣 A 的建構可使用 MCKMatrix 類

別，亦可使用 **ReMatrix** 類別的運算子，即乘 "*"、逆 "~"、水平合併 "&"、和垂直合併 "|"。

實例 1

```
using System;
using Matrix_0;

namespace ConsoleApp33
{
    internal class Program
    {
        static void Main(string[] args)
        {
            // 建構系統矩陣A。
            double[,] M0 = { {2, 0}, {0, 1} };
            ReMatrix M = new ReMatrix(M0);
            double[,] K0 = { {3, -1}, {-1, 1} };
            ReMatrix K = new ReMatrix(K0);
            double[,] C0 = { {0.4, -0.05}, {-0.05, 0.2} };
            ReMatrix C = new ReMatrix(C0);
            Console.WriteLine("\n 質量矩陣M :\n{0}", new PR(M));
            Console.WriteLine("\n 勁度矩陣K :\n{0}", new PR(K));
            Console.WriteLine("\n 阻尼矩陣C :\n{0}", new PR(C));

            MKCMatrix MKC = new MKCMatrix(M, K, C);
            ReMatrix A = MKC.Matrix;
            Console.WriteLine("\n 系統矩陣A :\n{0}\n", new PR(A));

            // 建構系統矩陣B：（使用矩陣運算子，兩者相同）
            Iden Id = new Iden(2);
            ReMatrix I = Id.GetMatrix;
            Zero Ze = new Zero(2);
            ReMatrix O = Ze.GetMatrix;

            ReMatrix B = (-1.0 * ~M * C & -1.0 * ~M * K) | (I & O);
            Console.WriteLine("\n 系統矩陣B :\n{0}\n", new PR(B));
        }
    }
}
```

```

}
/*輸出結果如下：
質量矩陣M：
    2.00000    0.00000
    0.00000    1.00000
勁度矩陣K：
    3.00000   -1.00000
   -1.00000    1.00000
阻尼矩陣C：
    0.40000   -0.05000
   -0.05000    0.20000
系統矩陣A：
   -0.20001    0.02500   -1.50000    0.50000
    0.05000   -0.20000    1.00000   -1.00000
    1.00000    0.00000    0.00000    0.00000
    0.00000    1.00000    0.00000    0.00000
系統矩陣B：
   -0.20000    0.02500   -1.50000    0.50000
    0.05000   -0.20000    1.00000   -1.00000
    1.00000    0.00000    0.00000    0.00000
    0.00000    1.00000    0.00000    0.00000
*/

```

與上式運動微分方程式相同，求多項式的根，可使用相同的方法，先建構系統矩陣 **A**，也就是建構友矩陣(Companion Matrix)。其中 **Roots** 類別與 **Matlab** 的 **Roots** 函式相似。

實例 2

```

using System;
using Matrix_0;

namespace ConsoleApp42
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("\n多項式: 5*y(4)-3*y(3)+7*y(2)-2*y(1)-7=0\n\n");

```

```

double[, ] Ar = { {5, -3, 7, -2, -7} };
Roots rt = new Roots(Ar);
CxMatrix cxMatrix = rt.GetCxMatrix;
Console.WriteLine("\n** 多項式之根: **\n\n{0}\n\n", new PR(cxMatrix));
    }
}
}

/* 輸出結果!
多項式: 5*y(4)-3*y(3)+7*y(2)-2*y(1)-7=0
** 多項式之根: **
      0.14871 +      1.40901i
      0.14871 -      1.40901i
      1.00000 +      0.00000i
     -0.69741 +      0.00000i
請按任意鍵繼續 . . .
*/

```

4.5 第五群組類別：

特徵值和特徵向量(EIG, CxEIG)，奇異值和奇異向量(SVD)。

兩者最大的差別是，系統矩陣為實數時，其奇異值和奇異向量永遠是實數。實數對稱系統矩陣，系統特徵值和系統特徵向量是實數，但非對稱實數系統矩陣，則系統特徵值和系統特徵向量是複數(但少數情況是實數)。雖然複數令人困惑，但精銳矩陣類別庫默認複數包含實數，兩者可以顯性和隱性轉換，故除非確認系統矩陣是對稱矩陣外，精銳矩陣類別庫默認系統特徵值和系統特徵向量為均為複數矩陣。

實例 1

```

using System;
using Matrix_0;

namespace ConsoleApp34
{
    internal class Program
    {
        static void Main(string[] args)
        {

```

```

double[,] A = { {-0.20, 0.025, -1.5, 0.5},
    {0.05, -0.20, 1, -1}, {1, 0, 0, 0}, {0, 1, 0, 0} };
Console.WriteLine("\n 系統矩陣 A :\n{0}", new PR(A));

EIG eig = new EIG(A);
CxMatrix D = eig.CxMatrixD;
CxMatrix V = eig.CxVector;
CxMatrix Q = eig.CxMatrixQ;
Console.WriteLine("\n 系統特徵值 :\n{0}", new PR(V));
Console.WriteLine("\n 系統特徵向量 :\n{0}", new PR(Q));

// 檢核系統特徵值V, 和系統特徵向量Q。 A = Q * D * Qi
CxMatrix B0 = Q * D * ~Q;
ReMatrix B = (ReMatrix)B0;
Console.WriteLine("\n 系統矩陣B :\n{0}", new PR(B));
Console.WriteLine("\n A和B兩個矩陣相同:表示計算結果正確!\n\n");

    }
}
}
/*
系統矩陣 A :
-0.20000    0.02500   -1.50000    0.50000
 0.05000   -0.20000    1.00000   -1.00000
 1.00000    0.00000    0.00000    0.00000
 0.00000    1.00000    0.00000    0.00000
系統特徵值 :
-0.11666 + 1.40933i
-0.11666 - 1.40933i
-0.08334 + 0.70221i
-0.08334 - 0.70221i
系統特徵向量 :
0.57745 + 0.00000i, 0.57745 + 0.00000i, 0.25800 + 0.00000i,
0.25800 + 0.00000i
-0.57707 - 0.01365i, -0.57707 + 0.01365i, 0.51648 - 0.00609i,
0.51648 + 0.00609i
-0.03369 - 0.40695i, -0.03369 + 0.40695i, -0.04300 - 0.36231i,

```

-0.04300 + 0.36231i
0.02405 + 0.40748i, 0.02405 - 0.40748i, -0.09463 - 0.72428i,
-0.09463 + 0.72428i

系統矩陣B：

-0.20000	0.02500	-1.50000	0.50000
0.05000	-0.20000	1.00000	-1.00000
1.00000	0.00000	0.00000	0.00000
0.00000	1.00000	0.00000	0.00000

A和B兩個矩陣相同:表示計算結果正確!

*/

實例 2

```
using System;
```

```
using Matrix_0;
```

```
namespace ConsoleApp35
```

```
{
```

```
    internal class Program
```

```
    {
```

```
        static void Main(string[] args)
```

```
        {
```

```
            double[,] Are = { {9, 3}, {3, -8} };
```

```
            double[,] Aim = { {0, 5}, {-5, 0} };
```

```
            CxMatrix A = new CxMatrix(Are, Aim);
```

```
            Console.WriteLine("\n ** 量子力學之厄米特矩陣 A : **\n{0}", new  
PR(A));
```

```
            CxEIG eig = new CxEIG(A);
```

```
            CxMatrix D = eig.CxMatrixD;
```

```
            CxMatrix V = eig.CxVector;
```

```
            Console.WriteLine("\n 系統特徵值 :\n{0}", new PR(V));
```

```
            CxMatrix Q = eig.CxMatrixQ;
```

```
            Console.WriteLine("\n 系統特徵向量 :\n{0}", new PR(Q));
```

```
            // A = Q * D * Qi
```

```
            CxMatrix B = Q * D * ~Q;
```

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



```

        Console.WriteLine("\n 已知系統矩陣A :\n{0}", new PR(A));

        SVD svd = new SVD(A);
        ReMatrix D = svd.MatrixD;
        ReMatrix Q = svd.MatrixQ;
        ReMatrix P = svd.MatrixP;
        Console.WriteLine("\n 奇異值D :\n{0}", new PR(D));
        Console.WriteLine("\n 奇異向量Q :\n{0}", new PR(Q));
        Console.WriteLine("\n 奇異向量P :\n{0}", new PR(P));

        // 檢核是否A = P * D * Qi;
        ReMatrix B = P * D * ~Q;
        Console.WriteLine("\n 系統矩陣B :\n{0}", new PR(B));

        Console.WriteLine("\n ** 系統矩陣A與B相同，表示結果正確！
        **\n\n");
    }
}

/*執行結果如下：
已知系統矩陣A：
1.00000  0.00000  1.00000
0.00000  1.00000  0.00000
0.00000  1.00000  1.00000
0.00000  1.00000  0.00000
1.00000  1.00000  0.00000
奇異值D：
2.23607  0.00000  0.00000
0.00000  1.41421  0.00000
0.00000  0.00000  1.00000
0.00000  0.00000  0.00000
0.00000  0.00000  0.00000
奇異向量Q：
0.40825 -0.57735 -0.70711
0.81650  0.57735  0.00000
0.40825 -0.57735  0.70711
奇異向量P：
0.36515 -0.81650  0.00000  0.44221 -0.06673

```

0.36515	0.40825	0.00000	0.54772	0.63246
0.54772	0.00000	0.70711	-0.44221	0.06673
0.36515	0.40825	0.00000	0.33669	-0.76592
0.54772	0.00000	-0.70711	-0.44221	0.06673

系統矩陣B：

1.00000	0.00000	1.00000
0.00000	1.00000	0.00000
0.00000	1.00000	1.00000
0.00000	1.00000	0.00000
1.00000	1.00000	0.00000

** 系統矩陣A與B相同，表示結果正確！ **

*/

4.6 第六群組類別：

利用系統特徵值和特徵向量，求解多維度和多階微分方程式(ToDexp, ToHexp, CxToDexp, CxToHexp, RowSlice)。

多維度和多階微分方程式，先使用 MKCMatrix 類別，或是使用 ReMatrix、CxMatrix 的運算子，先建構系統矩陣 **A**，可參考以上之第四類群組類別，再求得特徵值和特徵向量，可參考以上第五類群組類別，兩者是共軛(Self-Adjoint)的關係。作者也發現教科書或是相關之論文，僅談及特徵值和特徵向量，致於如何使用特徵值和特徵向量，好像並未談及。可能的原因是牽涉到複數矩陣相乘和逆矩陣的求取，雖然理論簡單，但複數矩陣的程式實作或許較複雜。以下是相關的類別，由初始值或是邊界值等已知條件，求解複數向量係數，再求解多維度多階微分方程式，也就是狀態空間響應。

複數矩陣類別 CxToHexp 和 CxToDexp，實際已包含實數矩陣類別 ToHexp 和 ToDexp，我們可考慮特徵值和特徵向量是複數，因為實數可隱性(Implicity)轉為複數，複數可顯性(Explicit)轉為實數。

實例 1

以下這個實例是從 Jagmohan L. Humar, "Dynamics of Structures Third Edition", Page 502 至 Page 504, 共 $m = 2$ 個自由度(DOF), $n = 2$ 階 (Order), 其系統矩陣僅為 $R(m \times n) \times (m \times n)$ 即 $R4 \times 4$ 的矩陣，而且本書已有狀態響應(即變位和速度)解答。故作者擬以本例作為樣板，請讀者詳細研讀此實例 1，進一步的擴充增加維度(m)和階數(n)，甚至使用不同的初始值和邊界值作進一步之驗證。


```

using System;
using Matrix_0;

namespace ConsoleApp6K
{
    internal class Program
    {
        static void Main(string[] args)
        {

double[,] M = { { 2, 0 }, { 0, 1 } };
double[,] K = { { 3, -1 }, { -1, 1 } };
double[,] C = { { 0.4, -0.05 }, { -0.05, 0.2 } };
double[,] y0Start = { { 0 }, { 0 }, { 1 }, { 2 } };

ReMatrix y0 = new ReMatrix(y0Start);
MKCMatrix MKC = new MKCMatrix(M, K, C);
ReMatrix A = new ReMatrix(MKC.Matrix);

EIG eig = new EIG(A);
CxMatrix D = eig.CxMatrixD;
CxMatrix V = eig.CxVector;
CxMatrix Q = eig.CxMatrixQ;
CxToHexp Hexp0 = new CxToHexp(D, Q, 0);
CxMatrix HexpTime0 = Hexp0.GetCxMatrix;
CxMatrix d = ~HexpTime0 * y0;

// 列印系統狀態參數。
Console.WriteLine("\n***{0, 12} 狀{0, 7} 態{0, 7} 參{0, 7} 數{0, 12}***\n", "");
Console.WriteLine("\n***{0, 5} 系統特徵值V{0, 5}***\n{1}\n", "", new PR(V));
Console.WriteLine("\n***{0, 5} 系統特徵向量Q{0, 5}***\n{1}\n", "", new
PR(Q));
Console.WriteLine("\n***{0, 5} 係數向量d{0, 5}***\n{1}\n", "", new PR(d));

double step = 0.5;
int iRow = (int)(50 / step + 1);
int iCol = M.GetLength(1) + 1;
ReMatrix Disp = new ReMatrix(iRow, iCol);

```

```

ReMatrix Vel = new ReMatrix(iRow, iCol);
ReMatrix Acc = new ReMatrix(iRow, iCol);

for (int i = 0; i != iRow; i++)
{
    double t = step * i;

    CxToDexp Dexp = new CxToDexp(D, t);
    CxMatrix DexpTime = Dexp.GetCxMatrix;
    CxMatrix yh_Cx = Q * DexpTime * d;
    ReMatrix yh_Re = (ReMatrix)yh_Cx;

    Vel.Matrix[i, 0] = t;
    Vel.Matrix[i, 1] = yh_Re.Matrix[0, 0];
    Vel.Matrix[i, 2] = yh_Re.Matrix[1, 0];
    Disp.Matrix[i, 0] = t;
    Disp.Matrix[i, 1] = yh_Re.Matrix[2, 0];
    Disp.Matrix[i, 2] = yh_Re.Matrix[3, 0];

    CxMatrix yhDot_Cx = A * yh_Cx;
    ReMatrix yhDot_Re = (ReMatrix)yhDot_Cx;
    Acc.Matrix[i, 0] = t;
    Acc.Matrix[i, 1] = yhDot_Re.Matrix[0, 0];
    Acc.Matrix[i, 2] = yhDot_Re.Matrix[1, 0];
}

// 列印節點的變位，速度，和加速。
Console.WriteLine("\n{0,5}***位移反應量***{0,5}\n{0,8}時間(秒)" +
    "{0,8}第0點位移{0,8}第1點位移\n\n{1}", "", new PR(Disp));
Console.WriteLine("\n{0,5}***速度反應量***{0,5}\n{0,8}時間(秒)" +
    "{0,8}第0點速度{0,8}第1點速度\n\n{1}", "", new PR(Vel));
Console.WriteLine("\n***{0,5}加速度反應量{0,5}***\n{0,8}時間(秒)" +
    "{0,8}第0點加速度{0,7}第1點加速度\n\n{1}", "", new PR(Acc));

    }
}
}

/*輸出結果如下：

```

```

***          狀      態      參      數          ***
***    系統特徵值 V    ***
-0.11666  +      1.40933i
-0.11666  -      1.40933i
-0.08334  +      0.70221i
-0.08334  -      0.70221i
***    系統特徵向量 Q    ***
0.57745 + 0.00000i, 0.57745 + 0.00000i, 0.25800 + 0.00000i,
0.25800 + 0.00000i
-0.57707 - 0.01365i, -0.57707 + 0.01365i, 0.51648 - 0.00609i,
0.51648 + 0.00609i
-0.03369 - 0.40695i, -0.03369 + 0.40695i, -0.04300 - 0.36231i,
-0.04300 + 0.36231i
0.02405 + 0.40748i, 0.02405 - 0.40748i, -0.09463 - 0.72428i,
-0.09463 + 0.72428i
***    係數向量 d    ***
0.00484  +      0.00006i
0.00484  -      0.00006i
-0.01084  +      1.37914i
-0.01084  -      1.37914i
***位移反應量***
時間(秒)    第 0 點位移    第 1 點位移
0.00000      1.00000      2.00000
0.50000      0.93967      1.87981
1.00000      0.77181      1.54694
....
49.00000     -0.01639     -0.03270
49.50000     -0.01619     -0.03246
50.00000     -0.01408     -0.02838
***速度反應量***
時間(秒)    第 0 點速度    第 1 點速度
0.00000      0.00000      0.00000
0.50000     -0.23583     -0.46888
1.00000     -0.42608     -0.84336
....
49.00000     -0.00167     -0.00368
49.50000      0.00241      0.00452
50.00000      0.00587      0.01152

```

***	加速度反應量	***	
	時間(秒)	第 0 點加速度	第 1 點加速度
	0.00000	-0.50000	-1.00000
	0.50000	-0.43415	-0.85816
	1.00000	-0.32011	-0.62777
		
	49.00000	0.00847	0.01696
	49.50000	0.00768	0.01549
	50.00000	0.00605	0.01229

*/

實例 2

與實例 1 相同的系統，當初始值為 $t=15$ 時，狀態參數完全相同，也就是狀態空間響應也是相同，即使初始值不同，結果仍然相同。

```
using System;
```

```
using Matrix_0;
```

```
namespace ConsoleApp36
```

```
{
```

```
    internal class Program
```

```
    {
```

```
        static void Main(string[] args)
```

```
        {
```

```
double[,] M = { { 2, 0 }, { 0, 1 } };
```

```
double[,] K = { { 3, -1 }, { -1, 1 } };
```

```
double[,] C = { { 0.4, -0.05 }, { -0.05, 0.2 } };
```

```
// @t=15, [y' y] = [0.18252 0.36524 -0.15568 -0.31900]
```

```
double[,] y0Start = { {0.18252}, {0.36524}, {-0.15568}, {-0.31900} };
```

```
ReMatrix y0 = new ReMatrix(y0Start);
```

```
MKCMatrix MKC = new MKCMatrix(M, K, C);
```

```
ReMatrix A = new ReMatrix(MKC.Matrix);
```

```
EIG eig = new EIG(A);
```

```
CxMatrix D = eig.CxMatrixD;
```

```
CxMatrix V = eig.CxVector;
```

```
CxMatrix Q = eig.CxMatrixQ;
```

```
// @ t = 15
```

```
CxToHexp Hexp0 = new CxToHexp(D, Q, 15);
```

```

CxMatrix HexpTime0 = Hexp0.GetCxMatrix;
CxMatrix d = ~HexpTime0 * y0;

// 列印系統狀態參數。
Console.WriteLine("\n***{0,12} 狀{0,7} 態{0,7} 參{0,7} 數{0,12}***\n", "");
Console.WriteLine("\n***{0,5} 系統特徵值V{0,5}***\n{1}\n", "", new PR(V));
Console.WriteLine("\n***{0,5} 系統特徵向量Q{0,5}***\n{1}\n", "", new
PR(Q));
Console.WriteLine("\n***{0,5} 係數向量d{0,5}***\n{1}\n", "", new PR(d));
    }
}
}
/*
***          狀          態          參          數          ***
***    系統特徵值V    ***
    -0.11666 +      1.40933i
    -0.11666 -      1.40933i
    -0.08334 +      0.70221i
    -0.08334 -      0.70221i
***    系統特徵向量Q    ***
    0.57745 + 0.00000i, 0.57745 + 0.00000i, 0.25800 + 0.00000i,
    0.25800 + 0.00000i
    -0.57707 - 0.01365i, -0.57707 + 0.01365i, 0.51648 - 0.00609i,
    0.51648 + 0.00609i
    -0.03369 - 0.40695i, -0.03369 + 0.40695i, -0.04300 - 0.36231i,
    -0.04300 + 0.36231i
    0.02405 + 0.40748i, 0.02405 - 0.40748i, -0.09463 - 0.72428i,
    -0.09463 + 0.72428i
***    係數向量d    ***
    0.00485 +      0.00006i
    0.00485 -      0.00006i
    -0.01083 +      1.37915i
    -0.01083 -      1.37915i
*/

```

實例 3

相同的系統，由邊界值決定係數向量。即當 $t=10$ 時， $[y_0 \ y_1]=[0.35495 \ 0.71296]$ ，當 $t=35$ 時， $[y_0 \ y_1]=[0.04277 \ 0.08499]$ 。微分方程式的求解，使

用邊界值的條件，是一種的近似求解法，稱為打靶法(Shooting Method), 作者稱而這種方法是將矩陣分割再合併，是一種新穎的精確的解析法(Analysis Method)，並經由程式語言的實作和驗證結果。

```
using System;
using Matrix_0;

namespace ConsoleApp37
{
    internal class Program
    {
        static void Main(string[] args)
        {
            double[,] M = { { 2, 0 }, { 0, 1 } };
            double[,] K = { { 3, -1 }, { -1, 1 } };
            double[,] C = { { 0.4, -0.05 }, { -0.05, 0.2 } };
            // @t=10 y=[0.35495 0.71296], @t=35 y=[0.04277 0.08499]
            double[,] y0Start = { {0.35495}, {0.71296}, {0.04277}, {0.08499} };

            ReMatrix y0 = new ReMatrix(y0Start);
            MKCMatrix MKC = new MKCMatrix(M, K, C);
            ReMatrix A = new ReMatrix(MKC.Matrix);

            EIG eig = new EIG(A);
            CxMatrix D = eig.CxMatrixD;
            CxMatrix V = eig.CxVector;
            CxMatrix Q = eig.CxMatrixQ;

            // @ t = 10
            CxToHexp Hexp10 = new CxToHexp(D, Q, 10);
            CxMatrix HexpTime10 = Hexp10.GetCxMatrix;
            RowSlice RSlice10 = new RowSlice(HexpTime10, 2, 1);
            CxMatrix Mat10 = RSlice10.GetCxMatrix;
            // @ t = 35
            CxToHexp Hexp35 = new CxToHexp(D, Q, 35);
            CxMatrix HexpTime35 = Hexp35.GetCxMatrix;
            RowSlice RSlice35 = new RowSlice(HexpTime35, 2, 1);
            CxMatrix Mat35 = RSlice35.GetCxMatrix;
```

```

// 將矩陣Mat15與矩陣Mat35垂直合并。即使用運算子 "|"。
CxMatrix Mat = Mat10 | Mat35;
// 向量係數d。
CxMatrix d = ~Mat * y0;

// 列印系統狀態參數。
Console.WriteLine("\n***{0, 12} 狀{0, 7} 態{0, 7} 參{0, 7} 數{0, 12}***\n", "");
Console.WriteLine("\n***{0, 5} 系統特徵值V{0, 5}***\n{1}\n", "", new PR(V));
Console.WriteLine("\n***{0, 5} 系統特徵向量Q{0, 5}***\n{1}\n", "", new
PR(Q));
Console.WriteLine("\n***{0, 5} 係數向量d{0, 5}***\n{1}\n", "", new PR(d));
    }
}
}

```

/*輸出結果如下：

```

***          狀          態          參          數          ***
***    系統特徵值V    ***
      -0.11666  +          1.40933i
      -0.11666  -          1.40933i
      -0.08334  +          0.70221i
      -0.08334  -          0.70221i
***    系統特徵向量Q    ***
0.57745 + 0.00000i, 0.57745 + 0.00000i, 0.25800 + 0.00000i,
0.25800 + 0.00000i
-0.57707 - 0.01365i, -0.57707 + 0.01365i, 0.51648 - 0.00609i,
0.51648 + 0.00609i
-0.03369 - 0.40695i, -0.03369 + 0.40695i, -0.04300 - 0.36231i,
-0.04300 + 0.36231i
0.02405 + 0.40748i, 0.02405 - 0.40748i, -0.09463 - 0.72428i,
-0.09463 + 0.72428i
***    係數向量d    ***
      0.00481  +          0.00025i
      0.00481  -          0.00025i
     -0.01087  +          1.37916i
     -0.01087  -          1.37916i
*/

```

4.7 第七群組類別：

精銳矩陣計算求解器類別庫共有兩百多個類別，除了以上所列舉之外，其他的輔助類別部分，譬如 `Iden`、`Zero`、`Remainder`，... 等等。

實例 1

```
using System;
using Matrix_0;

namespace ConsoleApp33
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Iden id = new Iden(3);
            ReMatrix A = id.GetMatrix;
            Console.WriteLine("\n 單位矩陣 :\n{0}", new PR(A));

            Zero ze = new Zero(3);
            ReMatrix B = ze.GetMatrix;
            Console.WriteLine("\n 零矩陣 :\n{0}", new PR(B));
        }
    }
}

/*輸出結果如下：
單位矩陣：
    1.00000    0.00000    0.00000
    0.00000    1.00000    0.00000
    0.00000    0.00000    1.00000
零矩陣：
    0.00000    0.00000    0.00000
    0.00000    0.00000    0.00000
    0.00000    0.00000    0.00000
*/
```

實例 2：


```

using System;
using Matrix_0;

namespace ConsoleApp33
{
    internal class Program
    {
        static void Main(string[] args)
        {
            // (被除數139.89) 除 (除數11.47897) = 商 ... 餘數
            Remainder Rem = new Remainder(139.89, 11.47897);
            double remVal = Rem.Value;
            Console.WriteLine("\n 1 : 循環餘數 = {0,10:F5}\n", remVal);

            // 除數的預設值是 2 * PI
            Rem = new Remainder(139.89);
            remVal = Rem.Value;
            Console.WriteLine("\n 2 : 循環餘數 = {0,-10:F5}\n\n", remVal);
        }
    }
}

/*輸出結果如下:
 1 : 循環餘數 =    2.14236
 2 : 循環餘數 = 1.65992
*/

```