第七章 實例測試與程式碼解說

作者撰寫 SMS-2029 精銳矩陣計算器時，一面撰寫程式碼同時一面測試，兩者同時進行，自我測試程式碼是一件重要的工作。

但有關動態微分方程式的測試，較為複雜，多自由度(m-DOF)且多階數(n-Order)而且有解答的實例並不多見。題材本身就是一個大問題，一般微分方程、線性代數、或是矩陣計算的書籍，其題材大部分是一個自由度(m=1)多階(n<=3)的問題，或是一個自由度多階非線性問題，但我們希望是多自由度且多階的微分方程式。上述題材都是無特別解。如果有外力時，即包含特別解，更增加問題的複雜性。

作者目前找到四個較具代表性的測試題才 :
比對測試的結果與課本提供的數據，完全相同，也表示類別庫程式碼的可靠性，而且四個測試的程式碼都很類似，讀者主要輸入已知的初始條件，或是邊界條件，就可得求得結果。讀者甚至直接用四個測試題材作測試。

第 0 個測試 :
J. L. Humar, "Dynamics of Structures" 第 502-504 頁。
第 1 個測試 :
Ray W. Clough & Joseph Penzien, "Dynamics of Structures" 第 202-203 頁。
第 2 個測試 :
William E. Boyce / Richard C. DiPrima, "Elementary Differential Equations and Boundary Value Problems 10th ED." 第 413 頁-416 頁。
第 3 個測試 :
Dennis G. Zill, "Differential Equations with Boundary-Value Problems 9th ED." 第 322 頁。

7.1 第 0 測試之已知條件和數學式響應

由已知 M、K、C、初始值、和數學式響應

$$M = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix} \quad K = \begin{pmatrix} 3 & -1 \\ -1 & 1 \end{pmatrix} \quad C = \begin{pmatrix} 0.4 & -0.5 \\ -0.05 & 0.2 \end{pmatrix}$$

$$\begin{pmatrix} \dot{y}(0) \\ y(0) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \setminus \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

$$\dot{y}_0(t) = e^{-0.08334t}\{ -0.00558 * \cos(0.70221t) - 0.71164 * \sin(0.70221t) \} + e^{-0.11677t} \{ 0.00559 * \cos(1.40933t) - 0.000066 * \sin(1.40933t) \}$$

$$\dot{y}_1(t) = e^{-0.08334t}\{-0.00558 * \cos(0.70221t) - 1.42474 * \sin(0.70221t) \} + e^{-0.11677t} \{ 0.00559 * \cos(1.40933t) + 0.000193 * \sin(1.40933t) \}$$

$$y_0(t) = e^{-0.08334t}\{-1.00028 * \cos(0.70221t) + 0.11076 * \sin(0.70221t) \} + e^{-0.11677t}\{ -0.00028 * \cos(1.40933t) + 0.00395 * \sin(1.40933t) \}$$

$$y_1(t) = e^{-0.08334t}\{1.99983 * \cos(0.70221t) + 0.24528 * \sin(0.70221t) \} + e^{-0.11677t}\{ 0.00019 * \cos(1.40933t) - 0.00395 * \sin(1.40933t) \}$$

$$\ddot{y}_0(t) = N / A$$
$$\ddot{y}_1(t) = N / A$$

由上式 ODE(t, m=2, n=2)，即使是很簡單的微分方程式和初始條件如下 ：

$$M * \ddot{y} + C * \dot{y} + K * y = 0$$

$$\begin{pmatrix} \dot{y}(0) \\ y(0) \end{pmatrix} = \left(\dot{y}(0) \setminus y(0)\right) = \left(0, 0, 1, 2\right) t$$

M, C, 和 K 是 mxm(即 2x2)的任何實數矩陣，但仍然須繁複計算，才可求得複雜的空間狀態響應數學式，但若是 m>2 或是 n>2，幾乎不容易由手算求得此複雜的響應數學式。另經作者實際測試，使用 SMS-2029 矩陣類別庫計算機程式，很容易求得的結果並與上式完全相同，故使用計算機程式求解是必需。以下是說明第 0 的測試的程式碼。

7.2　SMS-2029 程式碼與解說

使用 SMS-2029 類別庫，ConsoleApp6K 程式碼和相關的說明如下 ：

```
using System;
using Matrix_0;

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

double[,] y0Start = { { 0 }, { 0 }, { 1 }, { 2 } };

double[,] M = { { 2, 0 }, { 0, 1 } };
double[,] K = { { 3, -1 }, { -1, 1 } };
double[,] C = { { 0.4, -0.05 }, { -0.05, 0.2 } };
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 Hexp = new CxToHexp(D, Q, 0);
CxMatrix MatTemp = Hexp.GetCxMatrix;
CxMatrix d = ~MatTemp * y0Start;

// 列印空間相依狀態參數。
Console.Write("\n***{0,12}空{0,7}間{0,7}狀{0,7}態{0,7}參{0,7}
  數{0,12}***\n", "");
Console.Write("\n***{0,5}系統特徵值V{0,5}***\n{1}\n", "", new
PR(V));
Console.Write("\n***{0,5}系統特徵向量Q{0,5}***\n{1}\n", "",
  new PR(Q));
```

```
32    Console.Write("\n***{0,5}係數向量d{0,5}***\n{1}\n", "", new
        PR(d));
33
34    double step = 0.5;
35    int iRow = (int)(50 / step + 1);
36    int iCol = M.GetLength(1) + 1;
37    ReMatrix Disp = new ReMatrix(iRow, iCol);
38    ReMatrix Vel = new ReMatrix(iRow, iCol);
39    ReMatrix Acc = new ReMatrix(iRow, iCol);
40
41    for (int i = 0; i != iRow; i++)
42    {
43        double t = step * i;
44
45        Hexp = new CxToHexp(D, Q, t);
46        MatTemp = Hexp.GetCxMatrix;
47        CxMatrix yh_Cx = MatTemp * d;
48        ReMatrix yh_Re = (ReMatrix)yh_Cx;
49
50        Vel.Matrix[i, 0] = t;
51        Vel.Matrix[i, 1] = yh_Re.Matrix[0, 0];
52        Vel.Matrix[i, 2] = yh_Re.Matrix[1, 0];
53        Disp.Matrix[i, 0] = t;
54        Disp.Matrix[i, 1] = yh_Re.Matrix[2, 0];
55        Disp.Matrix[i, 2] = yh_Re.Matrix[3, 0];
56
57        CxMatrix yhDot_Cx = A * yh_Cx;
58        ReMatrix yhDot_Re = (ReMatrix)yhDot_Cx;
59        Acc.Matrix[i, 0] = t;
60        Acc.Matrix[i, 1] = yhDot_Re.Matrix[0, 0];
61        Acc.Matrix[i, 2] = yhDot_Re.Matrix[1, 0];
62
63        }
64
65    // 列印節點的變位，速度，和加速。
66    Console.Write("\n***{0,12}空{0,7}間{0,7}狀{0,7}態{0,7}響{0,7}
        應{0,12}***\n", "");
67    Console.Write("\n{0,5}***位移反應量***{0,5}\n{0,8}時間(秒)" +
```

```
68        "{0,8}第0點位移{0,8}第1點位移\n\n{1}", "", new PR(Disp));
69      Console.Write("\n{0,5}***速度反應量***{0,5}\n{0,8}時間(秒)" +
70        "{0,8}第0點速度{0,8}第1點速度\n\n{1}", "", new PR(Vel));
71      Console.Write("\n***{0,5}加速度反應量{0,5}***\n{0,8}時間(秒)"
          + 72 "{0,8}第0點加速度{0,7}第1點加速度\n\n{1}", "", new
          PR(Acc));
73
74                }
75          }
76      }
```

程式碼說明 :

L2 : 加入名稱空間(namespace), Matrix_0 組態檔。

L11 : t = 0 的初始條件

L17 : 建構系統矩陣 A

L20 - L22 : 特徵值 V 和特徵向量 Q

L24 - L26 : t = 0, 求得 係數向量 d

L29 - L32 : 空間相依狀態參數, 即特徵值 V, 特徵向量 Q, 和系統向量 d

L34 - L36 : 頭尾時間總個數 iRow, 響應總變數 iCol

L37 - L39 : 建構三個矩陣 Disp, Vel, 和 Acc, 其尺寸為 iRow 和 iCol

L41 - L63 : 在時間間隔內, 將響應的資料放入 Disp、Vel、和 Acc 的矩陣內。

L66 - L76 : 列印空間狀態響應資料, 即變位(Disp)、速度(Vel)、和加速度(Acc)

相關的 PDF 檔案、程式碼文件檔、執行後的文件檔案還包含空間狀態參數, 即特徵值、特徵向量和係數向量。時間軸上的空間狀態響應, 即變位、速度、和加速度。

使用 SMS-2029 類別庫, ConsoleApp6L 程式碼和相關的說明如下 :

```
1     using System;
2     using Matrix_0;
3
4     namespace ConsoleApp6L
5     {
6         internal class Program
```

```
7       {
8           static void Main(string[] args)
9           {
10
11  double[,] y0Start = { { 0 }, { 0 }, { 1 }, { 2 } };
12
13  double[,] M = { { 2, 0 }, { 0, 1 } };
14  double[,] K = { { 3, -1 }, { -1, 1 } };
15  double[,] C = { { 0.4, -0.05 }, { -0.05, 0.2 } };
16  MKCMatrix MKC = new MKCMatrix(M, K, C);
17  ReMatrix A = new ReMatrix(MKC.Matrix);
18
19  EIG eig = new EIG(A);
20  CxMatrix D = eig.CxMatrixD;
21  CxMatrix V = eig.CxVector;
22  CxMatrix Q = eig.CxMatrixQ;
23
24  CxToHexp Hexp = new CxToHexp(D, Q, 0);
25  CxMatrix MatTemp = Hexp.GetCxMatrix;
26  CxMatrix d = ~MatTemp * y0Start;
27
28  double step = 0.5;
29  int iRow = (int)(50 / step + 1);
30  int iCol = M.GetLength(1) + 1;
31  ReMatrix Disp = new ReMatrix(iRow, iCol);
32  ReMatrix Vel = new ReMatrix(iRow, iCol);
33  ReMatrix Acc = new ReMatrix(iRow, iCol);
34
35  for (int i = 0; i != iRow; i++)
36  {
37      double t = step * i;
38
39      Hexp = new CxToHexp(D, Q, t);
40      MatTemp = Hexp.GetCxMatrix;
41      CxMatrix yh_Cx = MatTemp * d;
42      ReMatrix yh_Re = (ReMatrix)yh_Cx;
43
44      Vel.Matrix[i, 0] = t;
```

```
45          Vel.Matrix[i, 1] = yh_Re.Matrix[0, 0];
46          Vel.Matrix[i, 2] = yh_Re.Matrix[1, 0];
47          Disp.Matrix[i, 0] = t;
48          Disp.Matrix[i, 1] = yh_Re.Matrix[2, 0];
49          Disp.Matrix[i, 2] = yh_Re.Matrix[3, 0];
50
51          CxMatrix yhDot_Cx = A * yh_Cx;
52          ReMatrix yhDot_Re = (ReMatrix)yhDot_Cx;
53          Acc.Matrix[i, 0] = t;
54          Acc.Matrix[i, 1] = yhDot_Re.Matrix[0, 0];
55          Acc.Matrix[i, 2] = yhDot_Re.Matrix[1, 0];
56
57          }
58
59          // 時間 :
60          Console.Write("\n 時間 : \n\n");
61          for( int i = 0; i != iRow; i++)
62          {
63              Console.Write("{0,10:F2}", Disp.Matrix[i, 0]);
64          }
65          Console.Write("\n\n");
66
67          // 位移
68          Console.Write("\n 位移 :\n");
69          Console.Write("\n 第 0 點 \n");
70          for(int i = 0; i != iRow; i++)
71          {
72              Console.Write("{0,10:F4}", Disp.Matrix[i, 1]);
73          }
74          Console.Write("\n 第 1 點 \n");
75          for(int i = 0; i != iRow; i++)
76          {
77              Console.Write("{0,10:F4}", Disp.Matrix[i, 2]);
78          }
79          Console.Write("\n\n");
80
81          // 速度
82          Console.Write("\n 速度 :\n");
```

```
83          Console.Write("\n 第 0 點 \n");
84          for (int i = 0; i != iRow; i++)
85          {
86              Console.Write("{0,10:F4}", Vel.Matrix[i, 1]);
87          }
88          Console.Write("\n 第 1 點 \n");
89          for (int i = 0; i != iRow; i++)
90          {
91              Console.Write("{0,10:F4}", Vel.Matrix[i, 2]);
92          }
93          Console.Write("\n\n");
94
95          // 加速度
96          Console.Write("\n 加速度 :\n");
97          Console.Write("\n 第 0 點 \n");
98          for (int i = 0; i != iRow; i++)
99          {
100             Console.Write("{0,10:F4}", Acc.Matrix[i, 1]);
101         }
102         Console.Write("\n 第 1 點 \n");
103         for (int i = 0; i != iRow; i++)
104         {
105             Console.Write("{0,10:F4}", Acc.Matrix[i, 2]);
106         }
107         Console.Write("\n\n\n");
108
109             }
110         }
111     }
```

ConsoleApp6L 的程式碼，是依據時間軸、變位、速度、和加速度分別輸出，這種輸出方式，利用 Python 的 Matplotlib 套件，方便繪製視覺化的圖表。讀者可利用提供的 TXT 資料數據自行測試。

使用 SMS-2029 類別庫，ConsoleApp6M 程式碼和相關的說明如下 ：

```
1   using System;
2   using Matrix_0;
3
4   namespace ConsoleApp6M
5   {
6      internal class Program
7      {
8          static void Main(string[] args)
9          {
10
11     // 已知陣列 M, K, and C
12     double[,] M = { { 2, 0 }, { 0, 1 } };
13     double[,] K = { { 3, -1 }, { -1, 1 } };
14     double[,] C = { { 0.4, -0.05 }, { -0.05, 0.2 } };
15
16     // 建構系統矩陣A :
17     ReMatrix Mi = ~(ReMatrix)M;
18     Iden iden = new Iden(2);
19     ReMatrix I = iden.Matrix;
20     Zero zero = new Zero(2);
21     ReMatrix O = zero.Matrix;
22     ReMatrix A = ((-1 * Mi * C) & (-1 * Mi * K)) | (I & O);
23
24     EIG eig = new EIG(A);
25     CxMatrix D = eig.CxMatrixD;
26     CxMatrix V = eig.CxVector;
27     CxMatrix Q = eig.CxMatrixQ;
28
29     // 邊界值 @t = 4.5
30     double[,] y0 = { { -0.68877 }, { -1.37729 } };
31     // 邊界值 @t = 16.5
32     double[,] y1 = { { 0.11710 }, { 0.23071 } };
33     ReMatrix BVal = (ReMatrix)y0 | y1;
34
35     // 建構係數向量
36     CxToHexp Hexp = new CxToHexp(D, Q, 4.5);
37     CxMatrix MatTemp = Hexp.GetCxMatrix;
```

```
38      RowSlice rowSlice = new RowSlice(MatTemp, 2, 1);
39      CxMatrix Mat1 = rowSlice.GetCxMatrix;
40      Hexp = new CxToHexp(D, Q, 16.5);
41      MatTemp = Hexp.GetCxMatrix;
42      rowSlice = new RowSlice(MatTemp, 2, 1);
43      CxMatrix Mat2 = rowSlice.GetCxMatrix;
44      CxMatrix Mat = Mat1 | Mat2;
45
46      CxMatrix d = ~ Mat * BVal;
47
48      // 列印空間相依狀態參數。
49      Console.Write("\n***{0,10}空{0,5}間{0,5}狀{0,5}態{0,5}參{0,5}
          數{0,10}***\n", "");
50      Console.Write("\n***{0,5}特徵值V{0,5}***\n{1}\n", "", new
          PR(V));
51      Console.Write("\n***{0,5}特徵向量矩陣Q{0,5}***\n{1}\n", "",
         new  PR(Q));
52      Console.Write("\n***{0,5}係數向量d{0,5}***\n{1}\n", "", new
          PR(d));
53
54          }
55      }
56  }
```

ConsoleApp6M 的程式碼，是利用邊界條件(BVC，Boundary Value Conditions)求得係數向量 d，與 ConsoleApp6K 是利用初始條件(IBC，Initial Value Conditions)求得係數向量，雖然程式碼或許較複雜。但求得的係數向量 d 完全相同。作者使用 ConsoleApp6K 的輸出結果，在 t = 4.5 秒時，第 0 點和第 1 點的變位 y，分別是 [-0.68877，-1.37729]，在 t = 16.5 秒時，則為 [0.11710, 0.23071]。 換言之相同的系統條件，得到相同的空間相依狀態參數(Spatial Dependency State)，特徵值(V)、特徵向量(Q)、和係數向量(d)，也就是有相同的空間相依狀態響應(Response)。作者有提供的 TXT 文件檔數據，請自行測試。

7.3　第 1 測試至第 3 測試部分

第 1 測試至第 3 測試的程式碼，幾乎與第 0 測試的程式碼完全相同，讀者可由提供的已知數據條件和程式碼，請自行測試，其結果與提供的數據是否相符。

至於資料的視覺化(輸出結果)，讀者可以將結果在 Excel 上繪製圖表。或是在 Visual Studio IDE 上執行 Python 專案，繪製圖表，也可以在 Visual Studio Code 上，執行 Python 程式，繪製圖表。