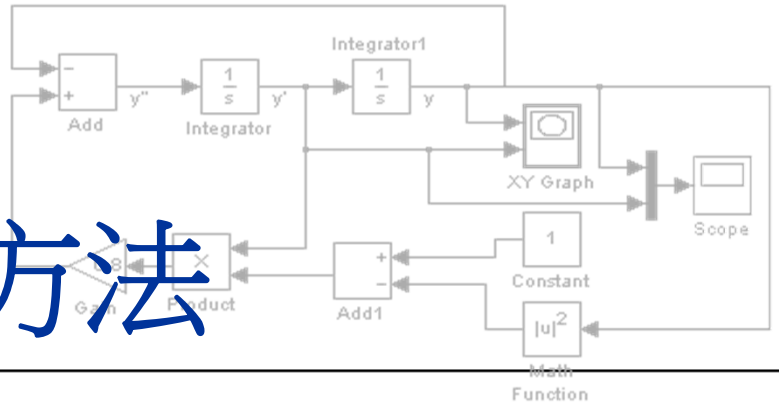


決策分析方法

求解線性聯立方程式

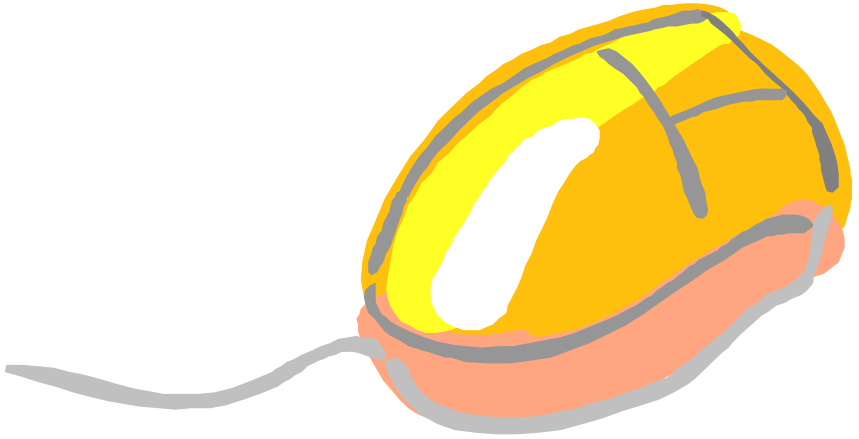
求解非線性方程式

最佳化設計的極值求解

曲線擬合與插值法

積分與微分方程式

常微分方程式



Linear Algebra Conception

- Matrix Equation
- Matrix Inverse
  - Cramers Rules
  - Determinant
- Gaussian Elimination
- LU Decomposition
- Square Root Method
  - Normal Equation

線性代數的相關運算
聯立方程式的求解

設要解聯立方程式 $\begin{bmatrix} 1 & 0 & 2 \\ 0 & 4 & 3 \\ 3 & 6 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 9 \\ 1 \\ 0 \end{bmatrix}$

這是

$AX=B$

的基本型式。要解出向量 X，只要計算 A 左除 B 即可：

```
>> A=[1 0 2;0 4 3;3 6 0]
A =
     1     0     2
     0     4     3
     3     6     0
```

```
>> B=[9;1;0]
B =
     9
     1
     0

>> X=A\B
X =
    3.5714
   -1.7857
    2.7143

>> inv(A)*B
ans =
    3.5714
   -1.7857
    2.7143
```

克拉瑪法則

- 克拉瑪法則：$AX = B$ 的唯一解為 $x_k = \frac{|A(k;B)|}{|A|}; \quad k = 1,\cdots,n$

```
>> A=[1 0 2;0 4 3;3 6 0];
>> B=[9;1;0];
>> det([B,A(:,[2 3])])/det(A)
ans =
    3.5714

>> det([A(:,1),B,A(:,3)])/det(A)
ans =
   -1.7857

>> det([A(:,[1 2]),B])/det(A)
ans =
    2.7143
```

使用linsolve解現性系統

- 線性方程式：$AX = B$

```
>> A=[4 1 2;1 2 5;2 5 1];
>> B=[4;3;1];
>> X=linsolve(A,B)

X =

    0.7901
   -0.2222
    0.5309

>>
```

$$A = \begin{bmatrix} 4 & 1 & 2 \\ 1 & 2 & 5 \\ 2 & 5 & 1 \end{bmatrix} \quad B = \begin{Bmatrix} 4 \\ 3 \\ 1 \end{Bmatrix}$$

當方程式數目不等於變數數目的情況

- over determinate：方程式的數目大於未知數的數目

- 「左除」法求解over determinate的問題，所求得的解是最小平方解。也就是找出的 X 是使得

$$\| AX - B \|^2$$

即 $AX - B$ 之範數（norm）的平方為最小。

$$2x - 4y = 6$$
$$3x - 8y = 3$$
$$6x + 7y = 14$$

```
>> A=[2 -4;3 -8;6 7]
A =
     2    -4
     3    -8
     6     7

>> B=[6;3;14]
B =
     6
     3
    14

>> X=A\B
X =
    2.0969
    0.2250
```

- under determinate：方程式的數目小於未知數的數目，此時有無窮多組解

  ✓利用左除法，可以求得所有的解向量裡，擁有元素0最多的向量

  ✓如果是以虛反矩陣法（pseudo-inverse），則所求得的是最小範數解

舉例來說，要求解如下的方程式：

$$5x+3y+3z-4u-9v=40$$
$$3x+3y+4z+2u-5v=30$$
$$7x-6y+3z+3u+2v=24$$

```
>> A=[5 3 3 -4 -9;3 3 4 2 -5;7 -6 3 3 2]
A =
     5     3     3    -4    -9
     3     3     4     2    -5
     7    -6     3     3     2

>> B=[40;30;24]
B =
    40
    30
    24
```

利用左除法：

```
>> s1=A\B
s1 =
    3.6250
         0
         0
    1.6513
   -3.1645
```

虛反矩陣法：

```
>> s2=pinv(A)*B
s2 =
    2.9836
    0.0273
    2.1179
    0.5061
   -2.2967
```

固有值與固有向量

- A的固有值可求解 $\det(\lambda I_n - A) = 0$ 而得
- 由 $AX=\lambda X$ 可求得伴隨λ的固有向量

Matlab 的 eig 函數可計算固有值與固有向量

表 11.1.2　求解固有值與固有向量的函數

函 數	說 明
ev=eig(*mat*)	計算矩陣 *mat* 的固有值
[*vect*,*dv*]=eig(*mat*)	計算矩陣 *mat* 的固有值與固有向量

```
>> A=magic(3)
A =
     8     1     6
     3     5     7
     4     9     2

>> eig(A)
ans =
   15.0000
    4.8990
   -4.8990
```

```
>> [vect,dv]=eig(A)
vect =
   -0.5774   -0.8131   -0.3416
   -0.5774    0.4714   -0.4714
   -0.5774    0.3416    0.8131
dv =
   15.0000         0         0
         0    4.8990         0
         0         0   -4.8990
```

方程式的求解
多項式的處理

- n 次多項式可以表示成如下的型式：

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x^1 + a_0$$

多項式

$$p(x) = 6x^5 + 12x^4 + 8x^3 - x + 32$$

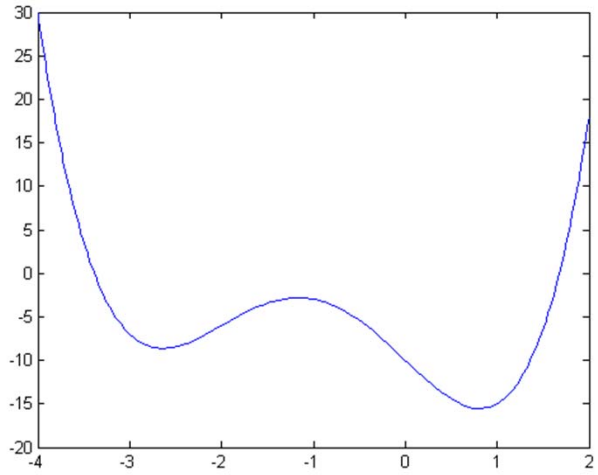
可用下面的列向量來表示：

```
p=[6 12 8 0 -1 32]
```

表 11.2.1　與多項式運算相關的函數

函 數	說 明
polyval(p,a)	計算多項式 $p(x)$ 於 $x=a$ 的值。a 可以為一個純量或向量
roots(p)	計算多項式 $p(x)$ 的根

```
>> p=[1 4 0 -10 -10];
>> fplot('x^4+4*x^3-10*x-10',[-4,2])

>> roots(p)
ans =
  -3.3851
   1.6769
  -1.1459 + 0.6698i
  -1.1459 - 0.6698i
```



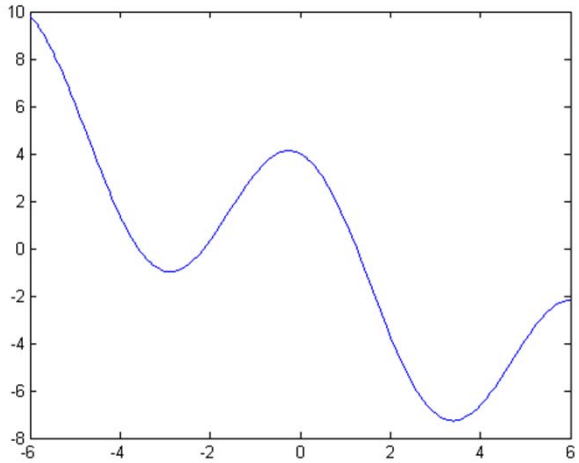
求解非線性方程式

- 如果要求解非線性方程式，可利用 fzero 函數：

表 11.2.2　求解非線性方程式的函數

函 數	說 明
`fzero(`*fun*`,`x_0`)`	以 x_0 為初始值，求解方程式 *fun*=0 的解
`fzero(`*fun*`,[`x_0`,`x_1`])`	在 `[`x_0`,`x_1`]` 區間內求解方程式 *fun*=0 的解

```
function y=func11_1(x)
y=4*cos(x)-x;

>> fzero('func11_1',3)
ans =
    1.2524

>> fzero('func11_1',-4)
ans =
   -3.5953

>> fzero('func11_1',[-4,-3])
ans =
   -3.5953
```

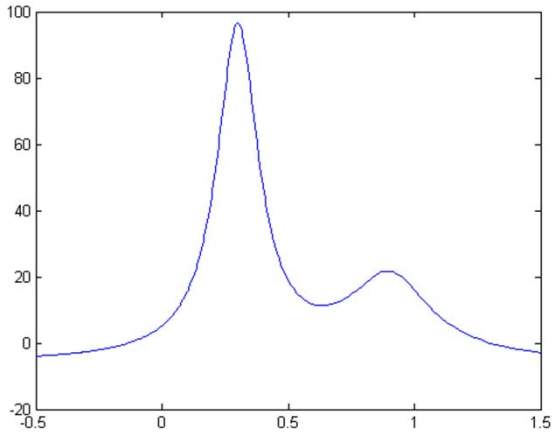


- 關於humps函數

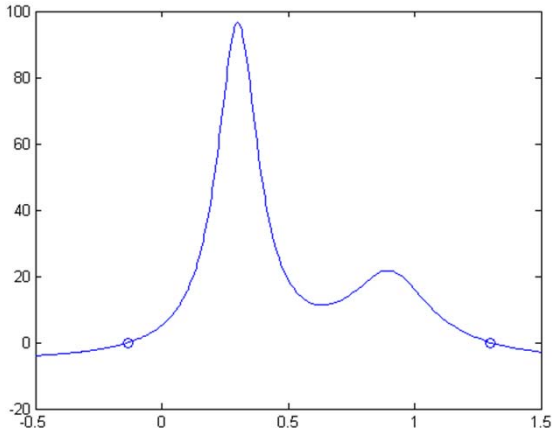
  humps 函數的數學定義式為

$$y(x) = \frac{1}{(x-0.3)^2 + 0.01} + \frac{1}{(x-0.9)^2 + 0.04} - 6$$

```
>> fplot('humps',[-0.5,1.5]); hold on
```



```
>> x1=fzero('humps',-0.5)
x1 =
   -0.1316

>> x2=fzero('humps',1.5)
x2 =
    1.2995

>> plot([x1 x2],[0 0],'o');hold off
```



顯示求解的過程

- optimset 函數可查看求解過程或改變求解精度：

表 11.2.3　optimset 函數的用法

函 數	說 明
opts=optimset('par_1','val_1','par_2','val_2',…)	依照參數 par_1 的值為 val_1，參數 par_2 的值為 val_2，建立一個選項結構，以供 Matlab 的 fzero、fminbnd 與 fminsearch 等函數使用

參 數	說 明
Display	設定 off 則不顯示輸出，設定 iter 則顯示每一個迭代的結果，設定 final 則只顯示計算的最終結果
TolX	設定所容許的誤差
FunValCheck	檢查計算過程中的函數值
OutputFcn	可用來指定一個自定的函數，只要求解過程中每迭代一次，自定的函數就會被執行一次

```
>> opts=optimset('Display','iter')

   opts =
                Display:  'iter'
           MaxFunEvals:  []
               MaxIter:  []
                TolFun:  []
                  TolX:  []
           FunValCheck:  []
             OutputFcn:  []
                     :     :
                TolPCG:  []
              TolRLPFun:  []
           TolXInteger:  []
              TypicalX:  []
```

```
>> fzero('humps',3,opts)

Search for an interval around 3 containing a sign change:
 Func-count      a         f(a)         b         f(b)        Procedure
      1          3     -5.63829          3     -5.63829  initial interval
      3    2.91515     -5.61014    3.08485     -5.66348       search
      5       2.88     -5.59749       3.12     -5.67314       search
      7    2.83029     -5.57852    3.16971      -5.6861       search
      9       2.76     -5.54928       3.24     -5.70314       search
     11    2.66059     -5.50236    3.33941     -5.72494       search
     13       2.52     -5.42219       3.48     -5.75188       search
     15    2.32118     -5.27031    3.67882     -5.78365       search
     17       2.04      -4.9243       3.96     -5.81906       search
     19    1.64235     -3.75631    4.35765     -5.85593       search
     20       1.08      9.42923    4.35765     -5.85593       search
```

```
Search for a zero in the interval [1.08, 4.3576]:
 Func-count     x          f(x)          Procedure
    20       4.35765        -5.85593    initial
    21       3.10194        -5.66823    interpolation
    22       2.09097        -5.00353    bisection
    23       1.58548        -3.43728    bisection
    24       1.33274       -0.670982    bisection
    25       1.28333        0.372862    interpolation
    26       1.30098      -0.0313704    interpolation
    27       1.29961     -0.00133688    interpolation
    28       1.29955    2.54388e-007    interpolation
    29       1.29955   -4.07603e-011    interpolation
    30       1.29955               0    interpolation
```

```
Zero found in the interval [1.08, 4.35765]
ans =
    1.2995
```

極小值的求解

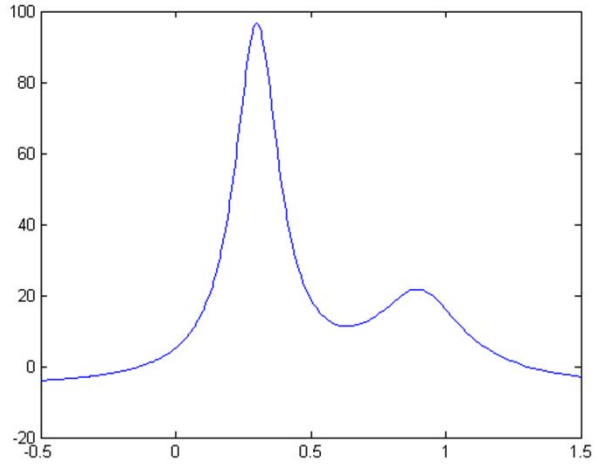
- fminbnd函數：可求解單一變數函數的極小值
- fminsearch函數：可求解多變數函數的極小值

表 11.3.1　求解函數的最小值

函 數	說 明
`fminbnd(`*fun*`,` x_1`,`x_2`)`	搜尋單變數函數 *fun* 的最小值
`fminsearch(`*fun*`,[`x`,`y`,`$\cdots$`])`	指定初始點為 $x,y,\cdots$，求解多變數函數 *fun* 的最小值

單一變數的極小值求解

```
>> fplot('humps',[-0.5,1.5])
>> fminbnd('humps',0.5,0.8)
ans =
    0.6370

>> [x,val]=fminbnd('humps',0.5,0.8)
x =
    0.6370
val =
   11.2528
```



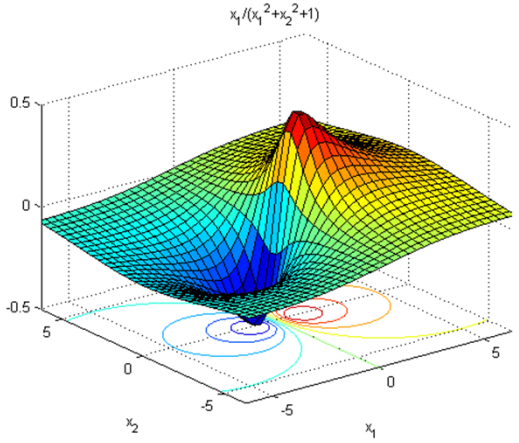
多變數函數的極小值求解

- 利用fminsearch求解函數極小值時：

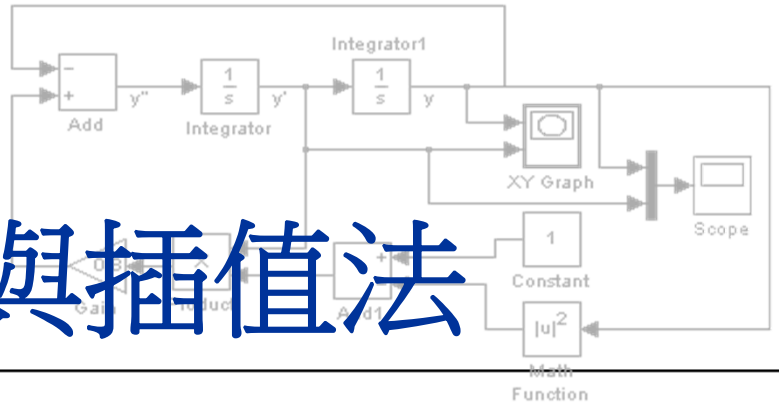
  第一個變數 x 必須定義成 $x(1)$ ，第二個變數 y 必須定義成 $x(2)$ ，多於兩個以上的變數則以此類推

```
function  y=func11_3(x)
y = x(1)/(x(1)^2+x(2)^2+1)
```

```
>> ezsurfc('x1/(x1^2+x2^2+1)',36)
```



```
>> fminsearch('func11_3',[0,4])
ans =
   -1.0000   -0.0000

>> [x,val]=fminsearch('func11_3',[0,4])
x =
   -1.0000   -0.0000
val =
   -0.5000

>> fminsearch('x(1)/(x(1)^2+x(2)^2+1)',[0,4])
ans =
   -1.0000   -0.0000
```



曲線擬合與插值法

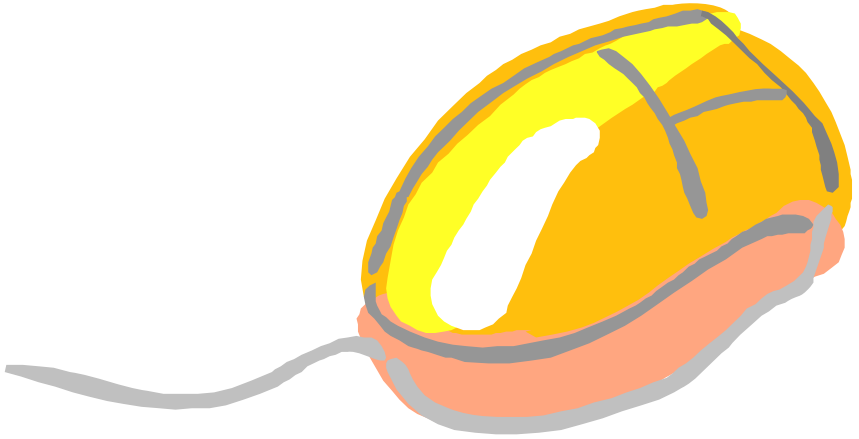
認識曲線擬合與插值法

n 階多項式的擬合方法

各種插值方法與其應用

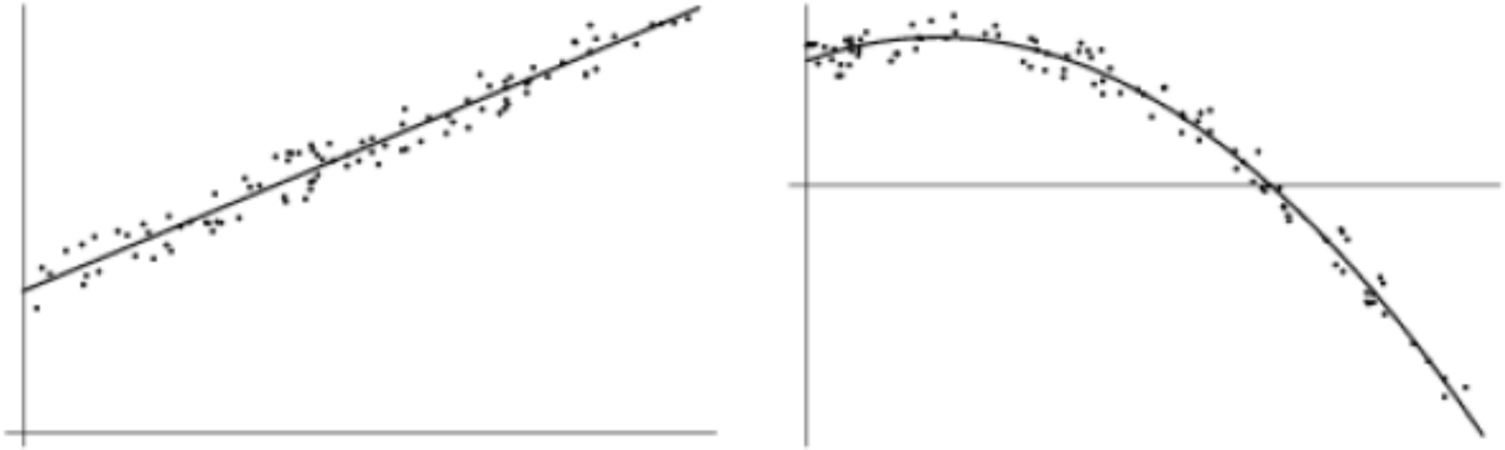
散佈式資料點的插值法

克利金(Kriging)法

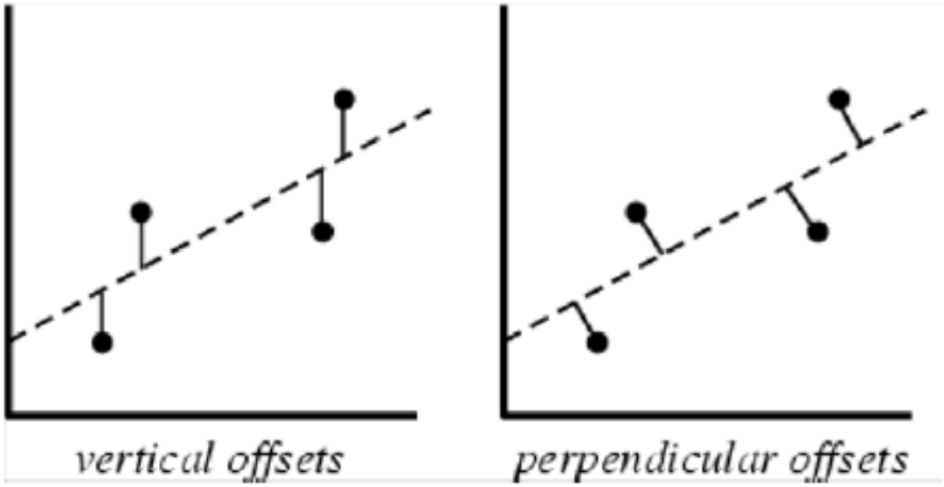


Least Squares Fitting

- Finding the best-fitting curve to a given set of points



- Minimizing the sum of the squares of the offsets ("the residuals") of the points from the curve



Mathematical References

- Linear regression: $\hat{y} = a + b\hat{x}$
  - Perpendicular offsets
  - Correlation coefficients
- Logarithmic regression: $\hat{y} = a + b \ln \hat{x}$
- Power-Law regression: $\hat{y} = A\hat{x}^B$
- Exponential regression: $\hat{y} = Ae^{B\hat{x}}$
- Polynomial regression: $\hat{y} = a_0 + a_1\hat{x} + \ldots + a_n\hat{x}^n$
- Nonlinear regression:
  - Ex: Gaussion functions $Y = A\exp(Bx)$

曲線擬合

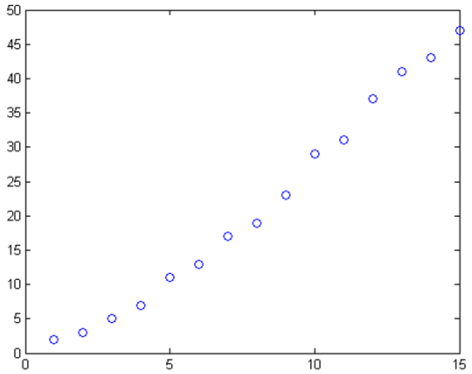
- 曲線擬合（curve fitting）：
  利用最小平方法求出最能表示出資料趨勢的函數

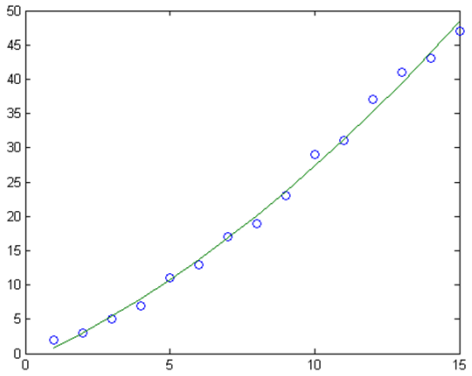
- Matlab是以ployfit函數來進行多項式的曲線擬合：

表 12.1.1　多項式曲線擬合函數

函 數	說 明
`polyfit(`x`,`y`,`n`)`	分別以橫座標與縱座標所組成的向量 x 與 y，進行 n 階多項式的擬合
`polyval(`p`,`a`)`	計算多項式 $p(x)$ 於 $x=a$ 的值

```
>> y=primes(50);
>> x=1:length(y);
>> plot(x,y,'o')
>> p2=polyfit(x,y,2)
p2 =
    0.0907    1.9517   -1.2484

>> f2=polyval(p2,x);
>> plot(x,y,'o',x,f2,'-')
```





插值法
一維的插值法

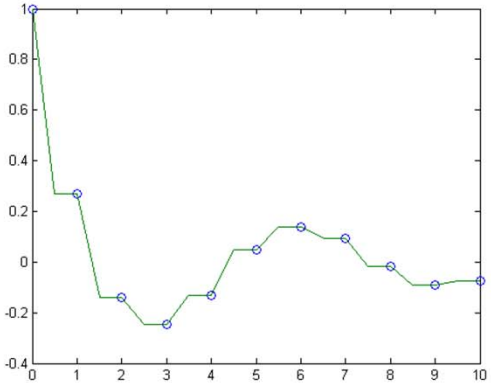
- 如果要處理一維資料點的插值，可用interp1函數：

表 12.2.1　一維插值法

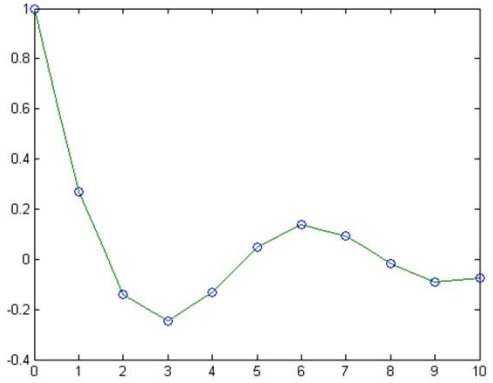
函 數	說 明
`interp1(`x`,`y`,`x_i`,'`*method*`')`	計算插值，其中 *method* 可為 `nearest:` 鄰近點插值法 `linear:` 線性插值法（預設值） `spline:` spline（雲形線）插值法 `cubic:` 三次多項式插值法
`interp1(`x`,`y`,`x_i`,'`*method*`','`*extrap*`')`	同上，但以外插來計算

```
>> interp1([1 5],[4 6],2,'nearest')
ans =
     4

>> interp1([1 5],[4 6],2,'linear')
ans =
    4.5000

>> interp1([1 5],[4 6],5.1,'linear','extrap')
ans =
    6.0500
```

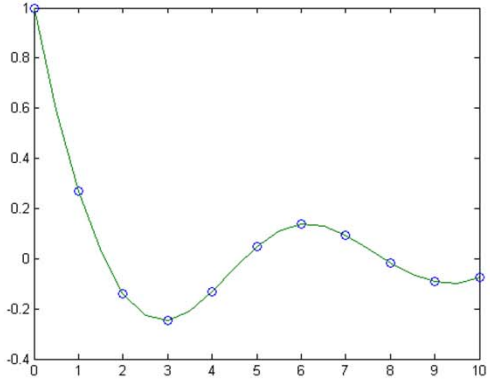
鄰近點插值法



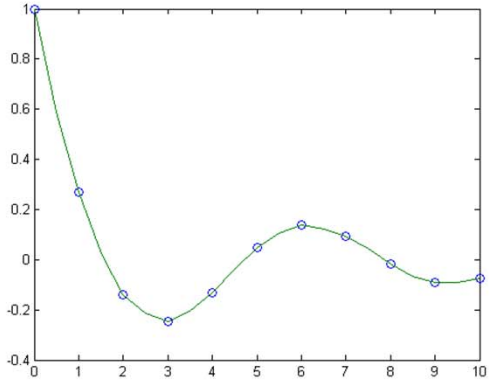
線性插值法



spline 內插



cubic 內插



二維平面的插值法

- interp2函數可用來進行二維平面插值：

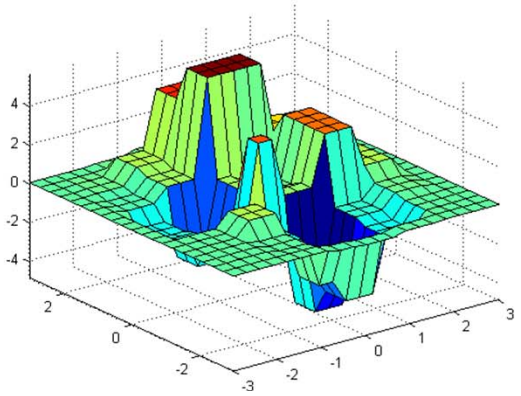
表 12.2.2　二維插值法

函 數	說 明
`interp2(`x`,`y`,`z`,`x_i`,`y_i`,'`*method*`')`	計算二維插值，其中 *method* 可為 `nearest:` 鄰近點插值法 `linear:` 線性插值法（預設） `spline:` spline 插值法 `cubic:` 三次多項式插值法
`interp2(`x`,`y`,`z`,`x_i`,`y_i`,'`*method*`',`*val*`)`	當插點落在數據範圍之外時，則結果設定為 *val*

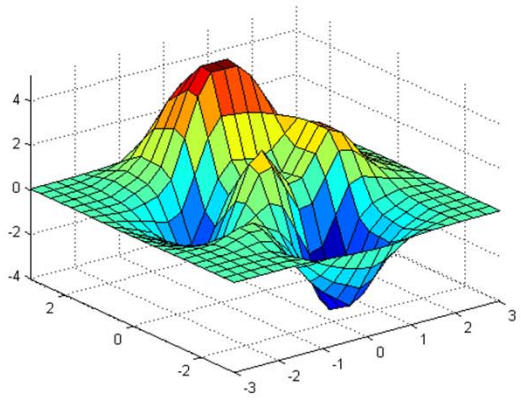
- 下面的範例是以peaks函數來示範如何進行二維平面的插值法：

```
>> [x,y,z]=peaks(8);
>> zz=peaks(1,1)
zz =
    2.4338

>> interp2(x,y,z,1,1,'linear')
ans =
    2.7986

>> interp2(x,y,z,[-0.5,0.2],[-0.4,1],'spline')
ans =
       3.3018   4.2221
```

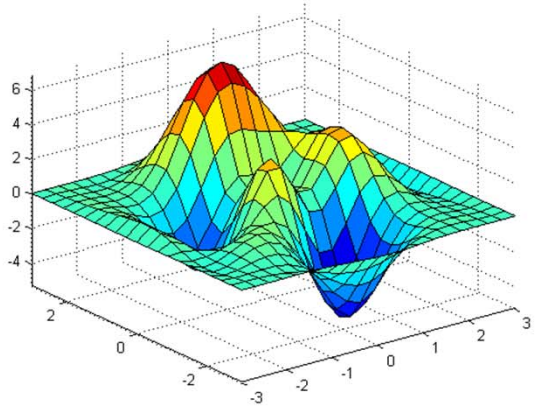
鄰近點插值法



線性插值法



spline 內插



多維插值法

- 多維插值法包含了三維與三維以上的多維插值，其語法如下表所列：

表 12.2.3　三維與三維以上的多維插值法

函 數	說 明
`interp3(`x,y,z,w,x_i,y_i,z_i`,'`*method*`')`	求解 $x = x_i$ 、 $y = y_i$ 與 $z = z_i$ 時所相對應的 w_i
`interp3(`x,y,z,w,x_i,y_i,z_i`,'`*method*`',`*val*`)`	插點的值超出範圍時，則以 *val* 的值當成插值
`interpn(`$x_1,x_2,\cdots,w,\ y_1,y_2,\cdots$`,'`*method*`')`	求解 $x_1 = y_1$ ， $x_2 = y_2$ ， $\cdots$ 時所相對應的 w_i
`interpn(`$x_1,x_2,\cdots,w,\ y_1,y_2,\cdots$`,'`*method*`',`*val*`)`	插點的值超出範圍時，則以 *val* 的值當成插值

- 三維插值法的結果可利用slice函數來觀看：

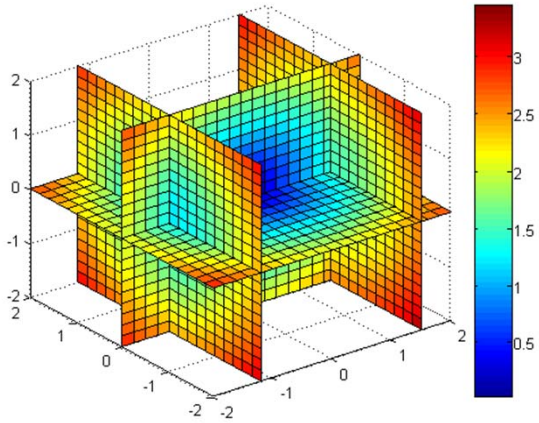
表 12.2.4　三維的 slice 繪圖指令

函 數	說 明
`slice(`x,y,z,w,s_x,s_y,s_z`)`	繪出交 x 軸於 s_x，交 y 軸於 s_y，以及交 z 軸於 s_z 之平面

- 下面是以 $w(x,y,z)=\sqrt{x^2+y^2+z^2}$ 為例，說明如何使用inperp3與slice函數。

```
>> w=sqrt(x1.^2+x2.^2+x3.^2);
>> x=-2:0.2:2; y=-2:0.2:2; z=-2:0.2:2;
>> [x1,x2,x3]=meshgrid(x,y,z);
```

```
>> slice(x1,x2,x3,w,[-1.2,1.5],0,0);colorbar
```




```
>> sqrt(0.35^2+0.17^2+0.64^2)
ans =
    0.7490

>> interp3(x1,x2,x3,w,0.35,0.17,0.64,'spline')
ans =
    0.7490
```

散佈式資料點插值法
二維的散佈點內插

- 當資料點散佈於平面或空間上時，可以採用散佈式插值法計算
- 散佈式插值法只能計算內插，不能計算外插
- Matlab以griddata函數進行二維的散佈點內插：

表 12.3.1　二維散佈式資料點插值法

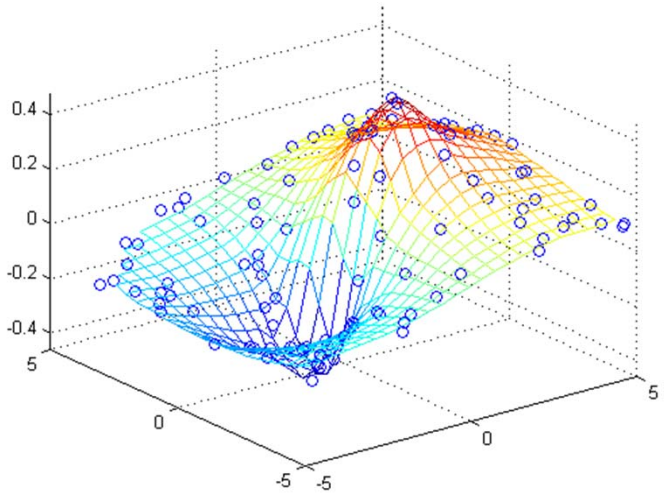
函 數	說 明
`griddata(`x`,`y`,`z`,`x_i`,`y_i`,'`*method*`')`	二維的散佈點內插，其中 *method* 可為
	`nearest:` 鄰近點插值法
	`linear:` 線性插值法（預設）
	`cubic:` 三次多項式插值法

- 下面是二維的散佈點內插的範例

```
>> x=10*rand(1,100)-5;
>> y=10*rand(1,100)-5;
>> z=x./(x.^2+y.^2+1);
>> griddata(x,y,z,1,1)
ans =
    0.2759
```

$$z(x,y) = x/(x^2 + y^2 + 1)$$

資料點與擬合之後的圖形：



三維的散佈點內插

- 如果散佈的資料點是三維，則可以利用griddata3函數來進行內插：

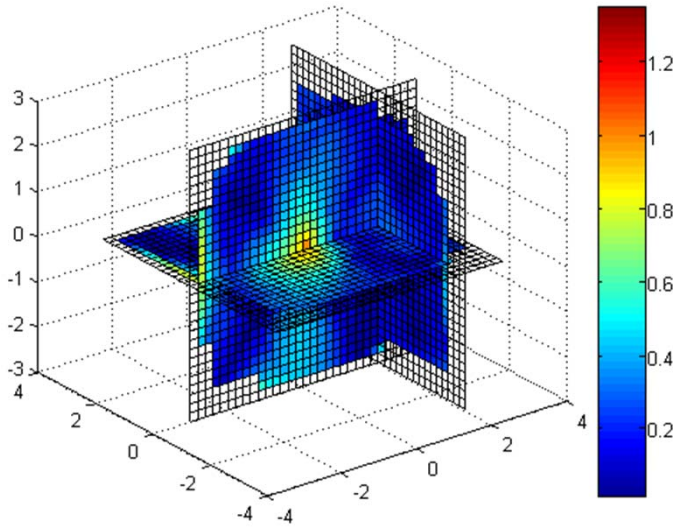
表 12.3.2　三維散佈式資料點插值法

函 數	說 明
`griddata3(`*x*`,`*y*`,`*z*`,`*w*`,`x_i`,`y_i`,`z_i`,'`*method*`')`	三維的散佈點內插，其中 *method* 可為
	`nearest:`　鄰近點插值法
	`linear:`　二維線性插值法（預設）

- 下面是三維散佈式資料點內插的作法：

```
>> x=6*rand(1,100)-3;
>> y=6*rand(1,100)-3;
>> z=6*rand(1,100)-3;
>> w=sqrt(x.^2+y.^2+z.^2);

>> sqrt(2^2+2^2+2^2)
ans =
    3.4641

>> griddata3(x,y,z,w,2,2,2)
ans =
    3.6999
```

$$w(x,y,z)=\sqrt{x^2+y^2+z^2}$$



上圖顯示了除了在原點附近之外，多半的誤差均小於0.2

空間資料分析之插值法
克利金法(Kriging technique)

- 克利金法又名地質統計法(Geostatistics technique)
- 此技術是以南非採礦地質學家D. G. Krige來命名
- 根據資料在空間中分佈的統計特性，推測線性內插係數的一種技術，以簡化的最小自成演算法為基礎，使用變異圖(Variogram)或半變異圖(Semi-variogram)為權重函數
- 基本假設為期望值與變異數只和隨機變數的距離有關，而與其所在空間無關。
- 通常用於從已知點的高程取得表面高程的推測值，可應用於任一現象自點資料產生一個表面

數學上不同趨勢條件假設下的克利金法

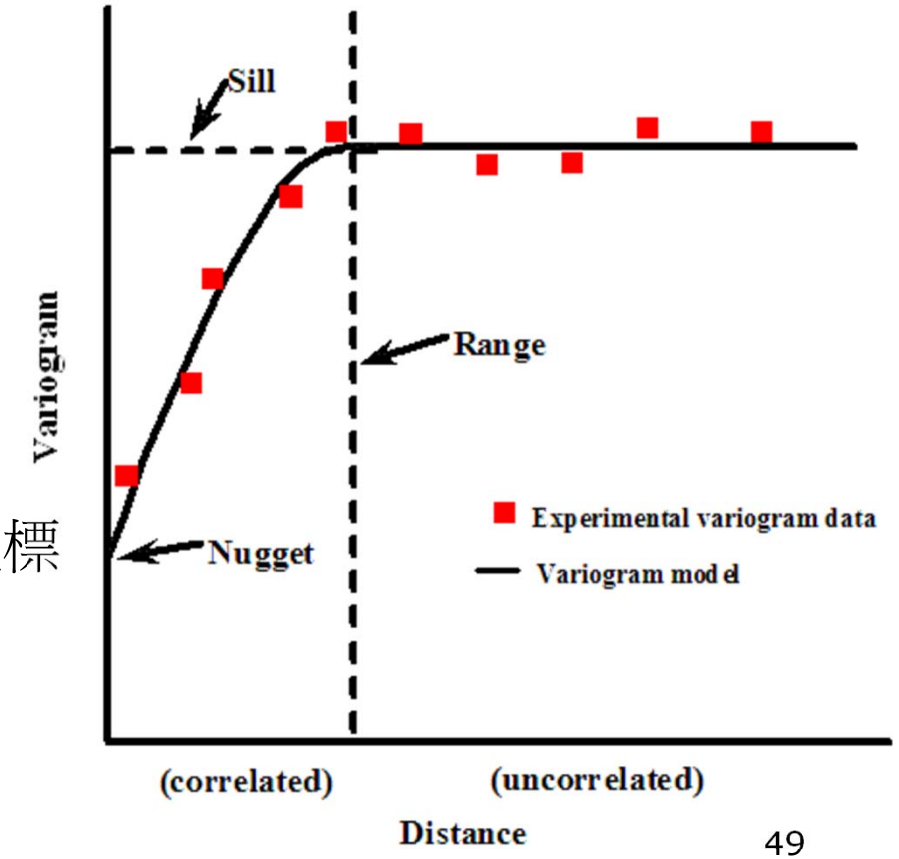
- 簡單克利金法(simple Kriging)：
  - 不包含「不偏估條件」(unbiased estimation)
- 普通克利金法(ordinary Kriging)：
  - 包含「平均值為常數」(constant mean)假設的「不偏估條件」
- 通用克利金法(universal Kriging)：
  - 包含「平均值為某種空間趨勢函數」(local trend)假設的「不偏估條件」
- 其他，如：區域克利金法、聯合克利金法及分離克利金法

克利金法推估步驟

- 結構分析(structural analysis)
  - 由歷史資料迴歸統計協變異數(covariance)隨距離變化的情形
  - 決定變異圖或半變異圖函數
- 最佳線性不偏估估計(Best Linear Unbiased Estimation-BLUE)
  - 假設估計值為已知值的線性權重平均
  - 根據不偏估和最小估計誤差變異數兩項原則
  - 利用變異圖或半變異圖導出權重係數值。
- 函數工具: Matlab, Python, R等均有提供

變異圖或半變異圖函數

- 變異圖(Variogram)函數
  - 觀測樣本資料的協變異數隨距離變化情形
    - Nugget: 碎塊效應
    - Sill: 臨界變異值
    - Range: 影響範圍
  - 函數定義
    - $2\gamma(d_{ij}) = E[(Z(u_i) - Z(u_j))^2]$
    - u(x, y)為空間位置座標
    - Z(u)為空間隨機變數
    - d_{ij}為i及j點距離
    - $\gamma(d_{ij})$為半變異圖函



半變異圖模式

- 模式說明
  - 半變異圖模式須滿足半變異結構及維度條件，為決定$\gamma(d_{ij})$需選用已滿足正定條件的模式
  - $\gamma(d_{ij})$決定後，即可提供Kriging進行最佳推估
- 配套模式
  - Spherical
  - Exponential
  - Power
  - Gaussian

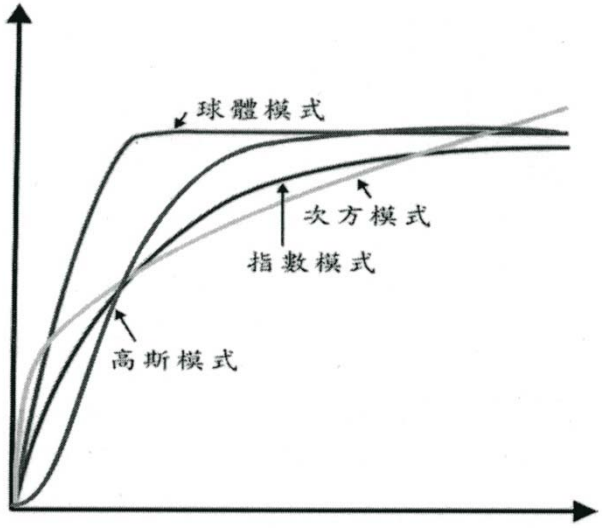


圖 3-3 常用理論半變異元套配模式

Kriging Example

```
% define the spatial geometry with random autocorrelation pts
[X,Y] = meshgrid(0:500);
Z = randn(size(X));      % Y, Y, Z all are a 500x500 matrix
Z = imfilter(Z,fspecial('gaussian',[40 40],8));
% sample the field: create 500 pts randomly in x, y axes and
% interpolated into z axis
n = 500;
% x, y are 500x1 vectors, 0<=x[i], y[i]<=500
x = rand(n,1)*500;
y = rand(n,1)*500;
% use linear interpolation ('linear', 'cubic','nearest','spline')
z = interp2(X,Y,Z,x,y);
% plot the random field by projecting 3D surf to 2D plan
% at the upper-left corner of the figure
subplot(2,2,1)
imagesc(X(1,:),Y(:,1),Z); axis image; axis xy
hold on
plot(x,y,'.k')
title('random field with sampling locations')
```

```
% calculate the sample variogram by variogram.m
v = variogram([x y],z,'plotit',false,'maxdist',100);
% and fit a spherical variogram by variogramfit.m
% plot at the upper-right corner of the figure
subplot(2,2,2)
[dum,dum,dum,vstruct] =
variogramfit(v.distance,v.val,[],[],[],'model','stable');
title('variogram')
% now use the sampled locations in a Kriging: (x,y,z) is
% observation, (X,Y) is prediction
% the processing bar will show as computing the Kriging
[Zhat,Zvar] = kriging(vstruct,x,y,z,X,Y);
% plot the prediction at the lower-left corner of the figure
subplot(2,2,3)
imagesc(X(1,:),Y(:,1),Zhat); axis image; axis xy
title('kriging predictions')
% plot the contour of prediction data, the contour is plotted at
% the lower-right corner
subplot(2,2,4)
contour(X,Y,Zvar); axis image
title('kriging variance')
```

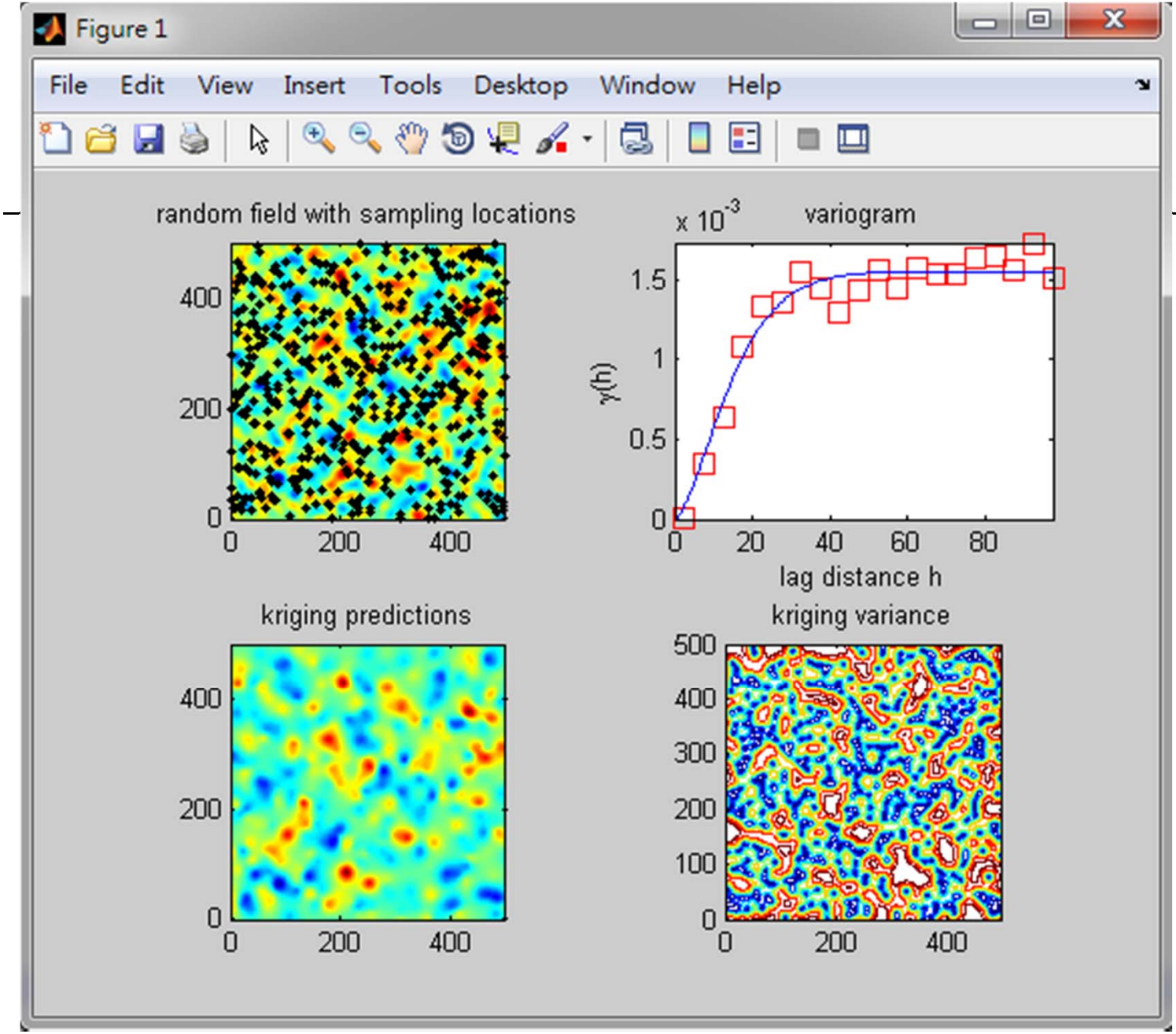
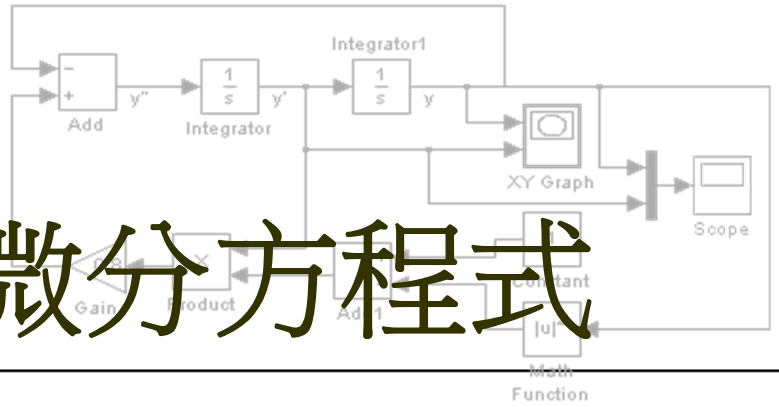


Figure 1
File Edit View Insert Tools Desktop Window Help
random field with sampling locations
0 200 400
0 200 400
variogram
x 10^-3
1.5
1
0.5
0
γ(h)
0 20 40 60 80
lag distance h
kriging predictions
0 200 400
0 200 400
kriging variance
500 400 300 200 100 0
0 200 400



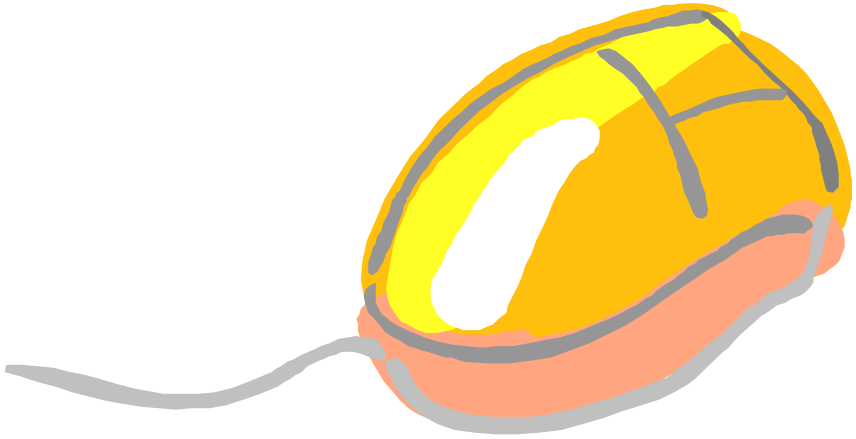
微積分與微分方程式

數值微分

數值積分及積分常見的問題

微分方程式的求解方法

剛性系統及解法



微分的運算
梯度與微分的運算

- 要計算數值微分，可利用gradient函數：

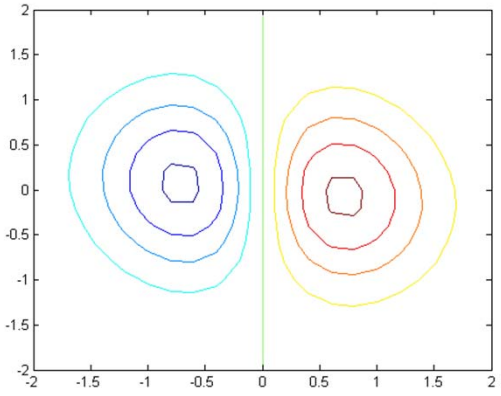
表 13.1.1　梯度運算函數

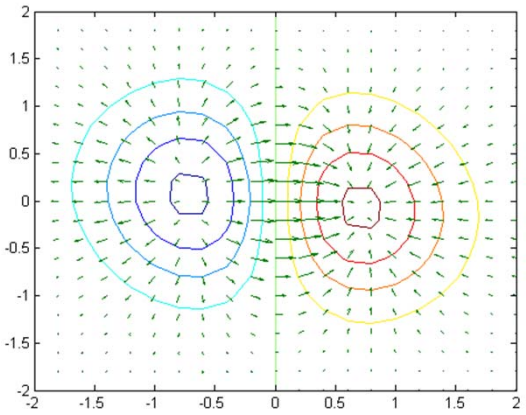
函 數	說 明
dy=gradient(*vect*,*dx*)	計算一維向量 *vect* 每一個元素所在位置之梯度
[*dAx*,*dAy*]=gradient(*A*,*dx*,*dy*)	計算矩陣 *A* 裡每一個元素於所在位置之 *x* 與 *y* 方向的梯度
[*dAx*,*dAy*,*dAz*]=gradient(*V*,*dx*,*dy*,*dz*)	計算三維陣列 *V* 裡每一個元素於所在位置之 *x*、*y* 與 *z* 方向的梯度

```
>> x=0:2:8
x =
     0     2     4     6     8

>> y=sqrt(x)
y =
    0  1.4142  2.0000  2.4495  2.8284

>> gradient(y,x)
ans =
    0.7071  0.5000  0.2588  0.2071  0.1895
```

- 若資料是二維，則gradient傳回兩個引數，分別代表 x 與 y 方向的梯度：

```
>> [xx,yy]=meshgrid(-2:0.2:2,-2:0.2:2);
>> zz=xx.*exp(-xx.^2-(yy+0.1*xx).^2);
>> contour(xx,yy,zz)
>> [px,py]=gradient(zz,0.2,0.2);
>> hold on;quiver(xx,yy,px,py);
```





差分的運算

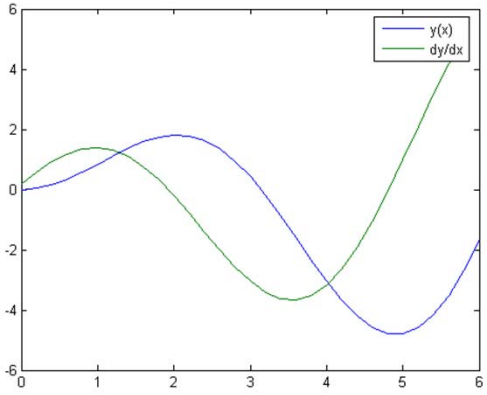
- 元素與元素之間的差值可利用diff指令進行差分運算：

表 13.1.2　差分運算函數

函 數	說 明
`diff(`*x*`)`	計算向量 *x* 的差分
`diff(`*mat*`,`*n*`)`	沿著矩陣 *mat* 的第 *n* 個維度進行差分的運算

- 差分運算也可以用來估算數據資料的梯度：

```
>> x=0:0.2:6;
>> y=x.*sin(x);
>> dy=diff(y)./diff(x);
>> plot(x,y,x(1:end-1),dy);
```



積分運算
單一變數之定積分運算

- Matlab提供了trapz與quad函數來計算數值積分：

表 13.2.1　單一變數的定積分運算函數

函 數	說 明
`trapz(`*y*`)`	間距為 1，以梯形法計算一維向量 *y* 的積分值。
`trapz(`*x*`,`*y*`)`	由 $y = f(x)$ 的關係式，以梯形法計算積分
`quad('`*func*`',`*a*`,`*b*`,`*tol*`)`	利用適應性辛普森法則對函數 *func* 做積分

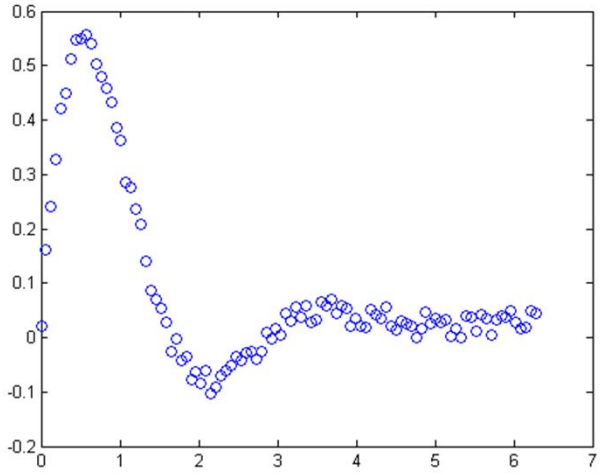
- 下面的範例是利用trapz來計算數值積分：

```
>> x=linspace(0,pi,50);
>> y=sin(x);

>> trapz(x,y)
ans =
    1.9993
```

$$\int_0^{\pi} \sin(x)\, dx$$

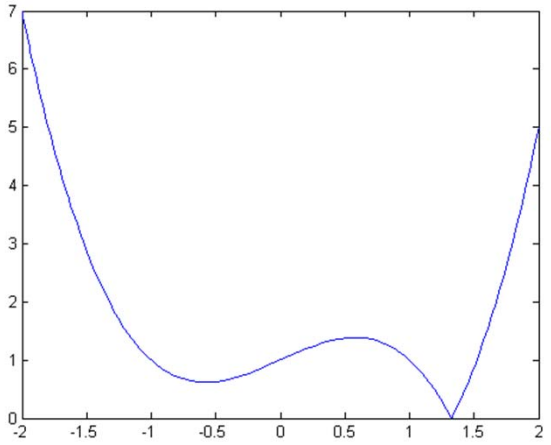
- 下面是利用梯形積分法對離散的資料點積分：

```
>> f=@(x) sin(2*x).*exp(-x);
>> x=linspace(0,2*pi,100);
>> y=f(x)+0.05*rand(1,100);
>> plot(x,y,'o')

>> trapz(x,y)
ans=
    0.5763
```



- 如果被積分函數是一個數學函數，則可利用quad函數來積分：

```
>> f=@(x) abs(x.^3-x-1);
>> fplot(f,[-2,2])
>> quad(f,-2,2)
ans =
    6.8645
```



多重積分

- 如要進行多重積分，可利用dblquad與triplequad函數：

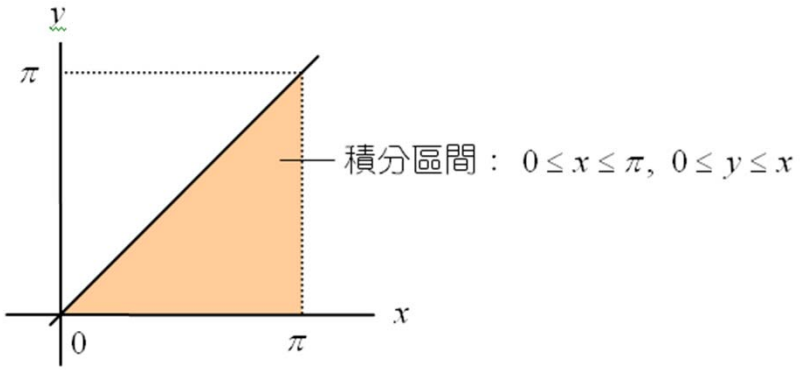
表 13.2.2　多變數的定積分運算函數

函 數	說 明
`dblquad('`*func*`',`x_1`,`x_2`,`y_1`,`y_2`,`*tol*`)`	計算 $\int_{x_1}^{x_2}\int_{y_1}^{y_2} func\ dy\,dx$
`triplequad('`*func*`',`x_1`,`x_2`,`y_1`,`y_2`,`z_1`,`z_2`,`*tol*`)`	計算 $\int_{x_1}^{x_2}\int_{y_1}^{y_2}\int_{z_1}^{z_2} func\ dz\,dy\,dx$

- 下面是積分區間是一個矩形（或立方體）的例子：

```
>> dblquad('x*y.^2',0,2,0,3)
ans =
   18.0000

>> triplequad('x+y.*z',0,6,0,2,1,4)
ans =
   198
```

- 如果區間不是矩形或立方體，可利用邏輯運算子&來定義積分的區間。



例如積分式

$$\int_0^{\pi}\int_0^{x} \cos(x+2y)\,dy\,dx$$

的被積分函數可定義成

```
cos(x+2*y).*(x>=0 & x<=pi & y>=0 & y<=x)
```

- 下面是一些多重積分的範例：

$$\int_0^2 \int_0^{4-2x} 8 - x - y \, dy \, dx$$

```
>> dblquad('(8-x-y).*(y<=4-2*x)',0,2,0,4)
ans =
   24.0000
```

$$\int_0^1 \int_x^1 x^2 \sqrt{1+y^4} \, dy \, dx$$

```
>> dblquad('x.^2.*sqrt(1+y.^4).*(y-x>=0)',0,1,0,1)
```

微分方程式的運算
微分方程式的解題器

- Matlab提供了7 種解題器
- 所有解題器的語法都相同，只是解題的演算法不同

表 13.3.1　微分方程式求解函數

函 數	說 明
$[t,y]=solver(odefun,[t_0,t_f],ini)$	區間從 t_0 到 t_f，初值為 *ini*，以 *solver* 方法解微分方程式 *odefun*
$[t,y]=solver(odefun,[t_0,t_1,t_2,\cdots,t_f],ini)$	同上，但是只解出特定時間 $t_0,t_1,t_2,\cdots,t_f$的值

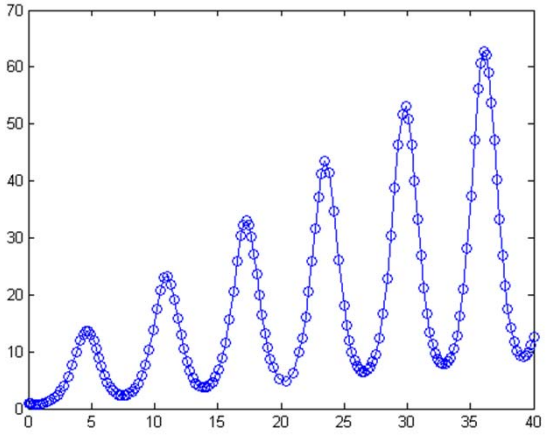
上表中，solver代表ode45、ode23、ode113、ode15s、ode23s、ode23t與 ode23tb 等 7 個函數的任意一個。

求解微分方程式

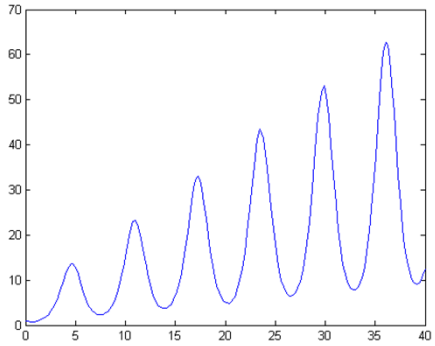
- 一階微分方程式可表示成 $y'(t) = f(y(t), t)$

下面是求解一階微分方程式的範例：

```
function dy=func13_1(t,y)
dy=sin(t)-y*cos(t);

>> ode45('func13_1',[0,40],1)
```



```
>> [t,y]=ode45('func13_1',[0,40],1);
>> plot(t,y)
```



- 下面的範例是利用匿名函數來定義微分方程式：

```
>> dy=@(t,y) cos(t)+y*cos(t/3);
>> [t,y]=ode45(dy,[0,40],0);
```

- n 階微分方程式可表示成

$$y^{(n)}(t) = f(y^{(n-1)}(t), y^{(n-2)}(t), \cdots, y'(t), \ t)$$

- 在定義 n 階微分方程式時，必須先把它改寫成 n 個一階的微分方程式

- 改寫之後，方程式便有 n 個狀態變數，此時必須以向量來定義它

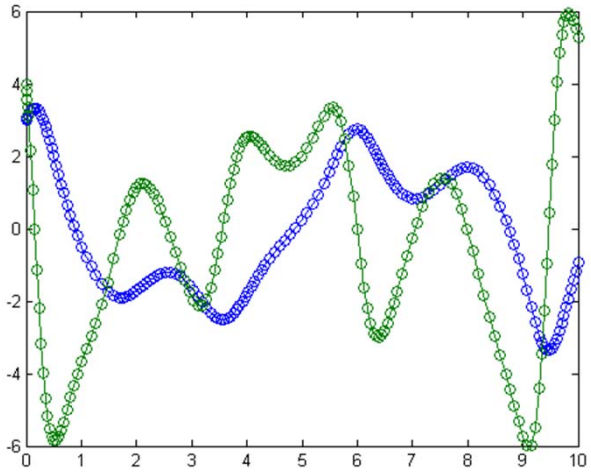
例如

$$y'' = 7.5\cos t - 0.05y' - y^3$$

可以改寫成下面的標準型式，即二個一階的微分方程式：

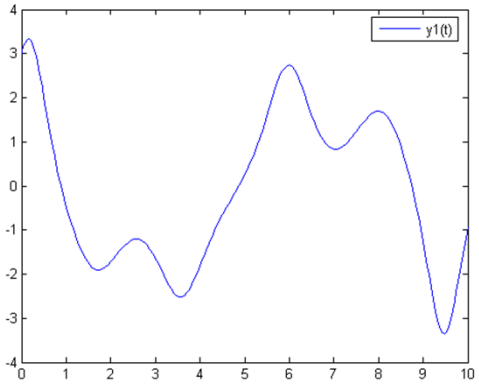
$$\begin{cases} y_1' = y_2 \\ y_2' = 7.5\cos t - 0.05y_2 - y_1^3 \end{cases}$$

```
function dy=func13_2(t,y)
dy=[y(2);
    7.5*cos(t)-0.05*y(2)-y(1)^3];

>> ode45('func13_2',[0,10],[3,4]) %y(0)=3, y'(0)=4
```

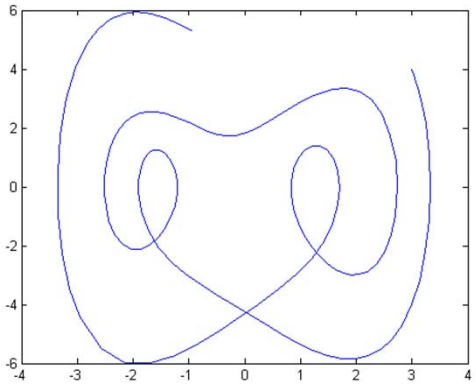


$$\begin{cases} y_1' = y_2 \\ y_2' = 7.5\cos t - 0.05 y_2 - y_1^3 \end{cases}$$

```
>> [t,y]=ode45('func13_2',[0,10],[3,4]);
>> plot(t,y(:,1));legend('y1(t)')
```




```
>> plot(y(:,1),y(:,2))
```



剛性系統的求解

- 剛性（stiff）方程式：方程式解的變化率差異過大
- 以 ode45 解剛性方程式時，會因為解題的步長無限的縮小，導致無法順利的解完整個方程式。
- 剛性方程式可以改用ode15s或ode23s等函數解之。

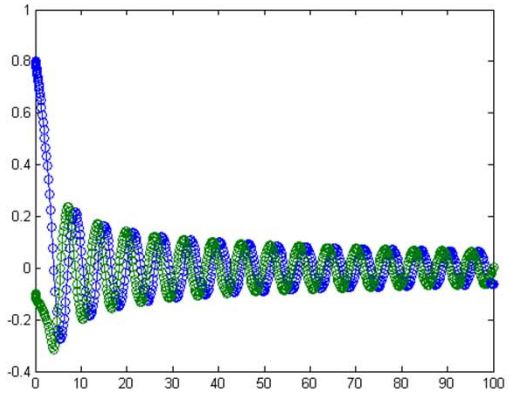
- 例如微分方程組

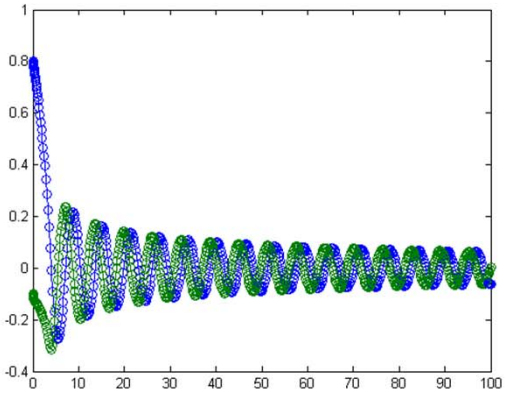
$$\begin{cases} y_1' = y_2 \\ y_2' = -y_1 - \mu\, y_1^2\, y_2 \end{cases}$$

當 μ值變得很大的時候（例如 μ =150) ，則為 stiff 系統

$\mu = 10$ 時的方程式（非剛性系統）：

```
>> dy=@(t,y) [y(2);-y(1)-10*y(1)^2*y(2)];
>> ode45(dy,[0,100],[1.3,-0.1]);
>> ode15s(dy,[0,100],[1.3,-0.1]);
```

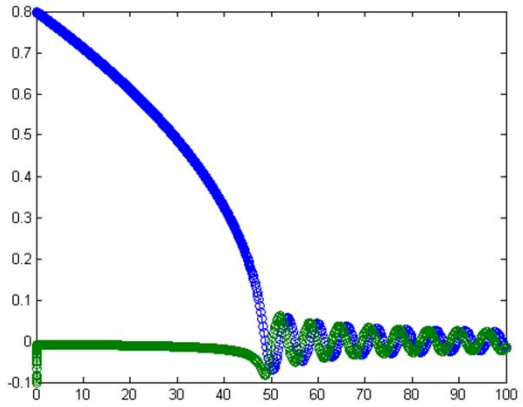


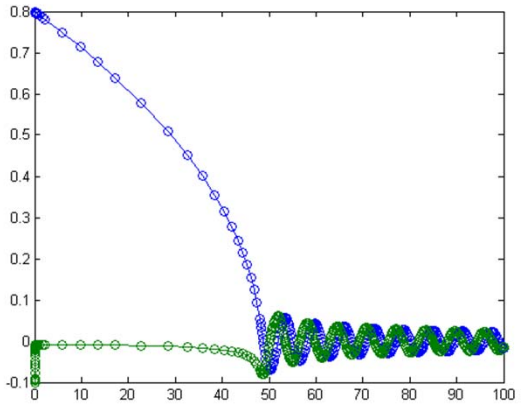


※ ode15s的解題速度較ode45來得慢上許多。因此對於非剛性的微分方程組而言，選擇ode45會有較好的執行效能。

$\mu = 150$ 時的方程式（剛性系統）：

```
>> dy=@(t,y) [y(2);-y(1)-150*y(1)^2*y(2)];
>> ode45(dy,[0,100],[0.8,-0.1]);
>> ode15s(dy,[0,100],[0.8,-0.1]);
```





※ 當μ=150時，ode15s比起ode45要快上許多。由此可知選對 ODE 的解題器將會大幅的影響到求解的效率。

解微分方程式的選項

- odeset可產生一個特殊的結構，讓ODE解題器可以依這個結構求值：

表 13.3.2　ODE 解題器的選項函數

函 數	說 明
opts=odeset('par_1','val_1','par_2','val_2',…)	依照參數 par_1 的值為 val_1，參數 par_2 的值為 val_2，建立一個選項結構

表 13.3.3　odeset 常用的參數

參 數	說 明
RelTol	相對誤差容許值，預設值為 10^{-3}
AbsTol	絕對誤差容許值，預設值為 10^{-6}
Refine	設定 Refine 為 *n* 可以用內插的方式產生 *n* 倍數目的輸出點
OutputFcn	如果沒有輸出引數，則在解題完畢後，Matlab 會自動呼叫此一函數

```
>> dy=@(t,y) [y(2); 7.5*cos(t)-0.05*y(2)-y(1)^3];
>> opts=odeset('RelTol',10^-4);
>> [t,y]=ode45(dy,[0,20],[3,4.2],opts);
>> plot(y(:,1),y(:,2),'-',y(:,1),y(:,2),'r.')
```