Lecture 8

第7章多變數非線性函數

大綱

- 非線性方程組
  - *牛頓法*
  - *正割法*
  - *固定點迭代*
- 最小化
  - *牛頓法(Newton-Raphson)*
  - *最大陡降法 (Steepest descent)*
  - *準牛頓法(Quasi-Newton)*
- 進階問題

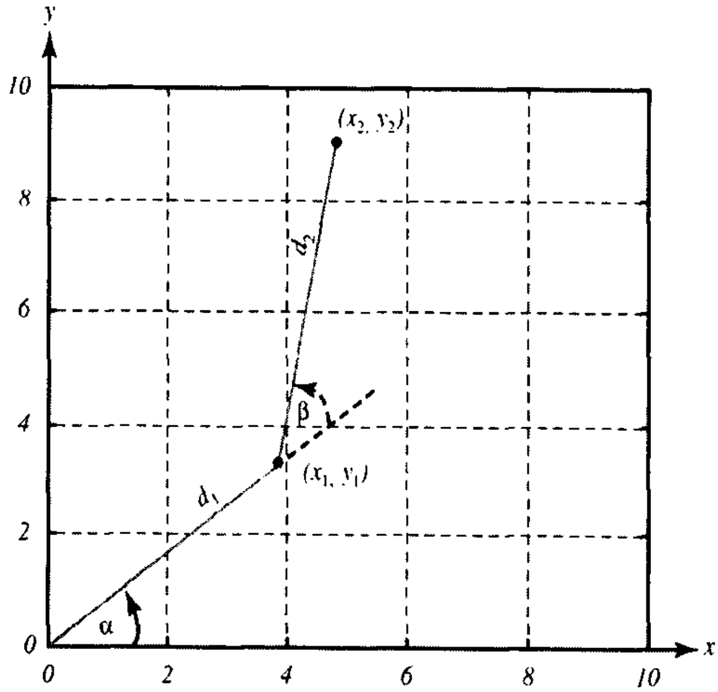
概述

- 在第2章所介紹的求單變數非線性函數零點的各種方法中，有數種可擴充至多變數非線性函數。
- 對於兩個變數的非線性函數 $z=f(x,y)$ 和 $z=g(x,y)$，要求得此方程組的零點，必須求出曲線 $f(x,y)=0$ 和 $g(x,y)=0$ 的交點。

問題7-A 雙節式機械手臂的位置

- 一個雙節式機械手臂的位置，可以用第一節與水平的夾角，和第二節與第一節的夾角，兩個角度來描述。
- 在本例中，我們假設兩節機械臂的長度分別為 d_1 和 d_2；第一節和水平軸夾角為 α，第二節和第一節所指方向的夾角為 β。我們的問題就是找出 α 和 β ，讓第二節的端點可落於指定之位置，即座標值 (p_1,p_2)。其配置如圖7.1所示。

問題7-A 雙節式機械手臂的位置



問題7-A 雙節式機械手臂的位置

- **這些要求可化為以下方程式:**

第一節端點 (x_1,y_1)：

$$x_1 = d_1 \cos(\alpha)$$

$$y_1 = d_1 \sin(\alpha)$$

第二節端點位置爲 (x_2,y_2)：

$$x_2 = x_1 + d_2 \cos(\alpha + \beta)$$

$$y_2 = y_1 + d_2 \sin(\alpha + \beta)$$

因此我們必須解

$$p_1 = d_1 \cos(\alpha) + d_2 \cos(\alpha + \beta)$$

$$p_2 = d_1 \sin(\alpha) + d_2 \sin(\alpha + \beta)$$

以獲得未知數 α 和 β。

在例題7.2中會解此問題。

問題7-B 最小總距離

- 已知平面上三點(x_1,y_1)，(x_2,y_2)及(x_3,y_3) ，希望求出點p(x,y)的位置，使得點p到三已知點距離的平方和為最小。即要求以下函數的最小值

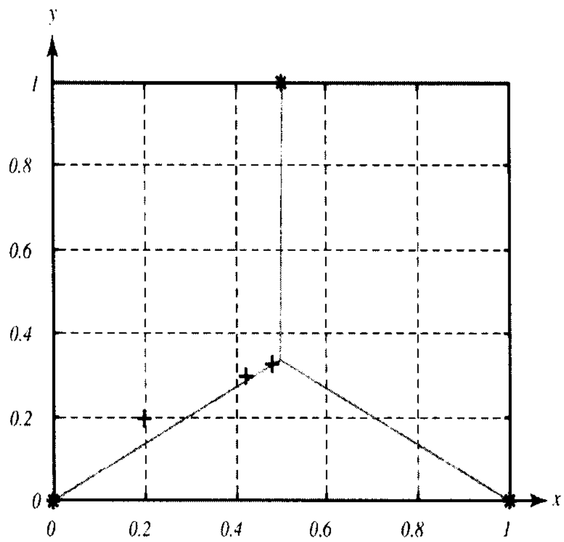
$$f(x,y)=(x-x_1)^2+(y-y_1)^2+(x-x_2)^2+(y-y_2)^2+(x-x_3)^2+(y-y_3)^2 \text{。}$$

所須的導數爲

$$f_x(x,y)=2(x-x_1)+2(x-x_2)+2(x-x_3)=6x-2x_1-2x_2-2x_3$$
$$f_y(x,y)=2(y-y_1)+2(y-y_2)+2(y-y_3)=6y-2y_1-2y_2-2y_3$$

問題7-B 最小總距離

- 初始估計值(0.2,0.2)和頭二次更新的結果顯示於圖 7.2，三個已知點 (0,0)、(1,0)和(1/2,1)同時顯示於圖中。



問題7-B 最小總距離

- 以下顯示MATLAB函數fminsearch的用法，它無須用到要求最小值之函數上的導數。函數fminsearch求得f的最小值出現在z=[0.5000 0.3333]。

```
f=inline('x(1)^2+x(2)^2+(x(1)-1)^2+x(2)^2+(x(1)-0.5)^2+(x(2)-1)^2')
z = fminsearch(f, [0.2, 0.2])
```

- 相關的問題包括對全距(不僅是距離平方)或者是某一與距離有關之量的總成本的最小化。

非線性方程組

- 解非線性方程組的問題，本質上就比解單一方程式困難得多。對於包含兩個變數的兩個聯立方程式，它的解是兩個函數的零等值線的交點，它們可能交於多點，或不相交。
- 在更高維度時，各種可能性變得更複雜。在此先討論牛頓法的多維形式，然後再考慮一些相關的方法。

解非線性方程組之方法

- 牛頓法
- 正割法
- 固定點迭代

牛頓法

- 在第2章所介紹的求單一非線性方程式之根的牛頓法，可以擴展到解非線性方程組。在以下例題中，用上標 (加括弧)表示迭代，因為經常要用到向量符號，用下標來表示未知向量的分量。在不需要保留所有的迭代數的情況下，用x_{old}代表既有估計值，用x_{new}代表下個估計值。

牛頓法

$$x_{new} = x_{old} - J^{-1}(x)F(x)$$

其中，$F(x)$為求解方程組，J為$Jacobian$矩陣

$$J(x_1,\cdots,x_n) = \begin{vmatrix} \partial f_1/\partial x_1 & \cdots & \partial f_1/\partial x_n \\ \vdots & \ddots & \vdots \\ \partial f_n/\partial x_1 & \cdots & \partial f_n/\partial x_n \end{vmatrix}$$

例題7.1 圓和拋物線的交點

- 首先考慮的是有兩個方程式的方程組，它們代表一個單位圓(以原點為圓心)和一條拋物線 (以原點為頂點)的交點。兩曲線顯示於圖7.3。尋找以下兩函數的共同零點：

$$\begin{cases} f(x,y) = x^2 + y^2 - 1 \\ g(x,y) = x^2 - y \end{cases}$$

例題7.1 圓和拋物線的交點

- 以共同解的初始估計值(x_0,y_0)為起點。在(x x_0,y_0,F(x_0,y_0))處正切於函數z=f(x,y)之平面的方程式為

$$z - f(x_0, y_0) = f_x(x_0, y_0)(x - x_0) + f_y(x_0, y_0)(y - y_0)$$

$$z - g(x_0, y_0) = g_x(x_0, y_0)(x - x_0) + g_y(x_0, y_0)(y - y_0)$$

其中

$f_x(x_0, y_0)$、$f_y(x_0, y_0)$、$g_x(x_0, y_0)$、$g_y(x_0, y_0)$分別為$f(x, y)$及$g(x, y)$在(x_0, y_0)處相對於x和y的偏導數。

- 要求得近似解，要找出這兩個切面與xy平面(即z=0的平面)的交集。

例題7.1 圓和拋物線的交點

定義$r = (x - x_0)$及$s = (y - y_0)$，另外$z = 0$，則方程組變成

$$f_x(x_0, y_0)r + f_y(x_0, y_0)s = -f(x_0, y_0)$$

$$g_x(x_0, y_0)r + g_y(x_0, y_0)s = -g(x_0, y_0)$$

其中

r及s為交會點相對於點(x_0, y_0)的位移量，亦即

$(x, y) = (r + x_0, s + y_0)$或

$$\begin{cases} x = x_0 + r \\ y = y_0 + s \end{cases}$$

例題7.1 圓和拋物線的交點

$$\begin{cases} f(x,y)=x^2+y^2-1 \\ g(x,y)=x^2-y \end{cases}$$

$f_x=2x,\ f_y=2y, g_x=2x$且$g_y=-1$

若初始值$(x_0,y_0)=(0.5,0.5)$，則

$f_x=1,\ f_y=1, g_x=1, g_y=-1,$及

$f(0.5,0.5)=-0.5$

$g(0.5,0.5)=-0.25$

例題7.1 圓和拋物線的交點

- 由此得待解之線性方程組為

$$\begin{cases} r+s=0.5 \\ r-s=0.25 \end{cases} \Rightarrow \text{可得} \begin{cases} r=3/8 \\ s=1/8 \end{cases}$$

因此第二次迭代值(x_1,y_1)為

$$\begin{cases} x_1=x_0+r=0.5+3/8=7/8 \\ y_1=y_0+s=0.5+1/8=5/8 \end{cases}$$

重複前面步驟

例題7.1 圓和拋物線的交點

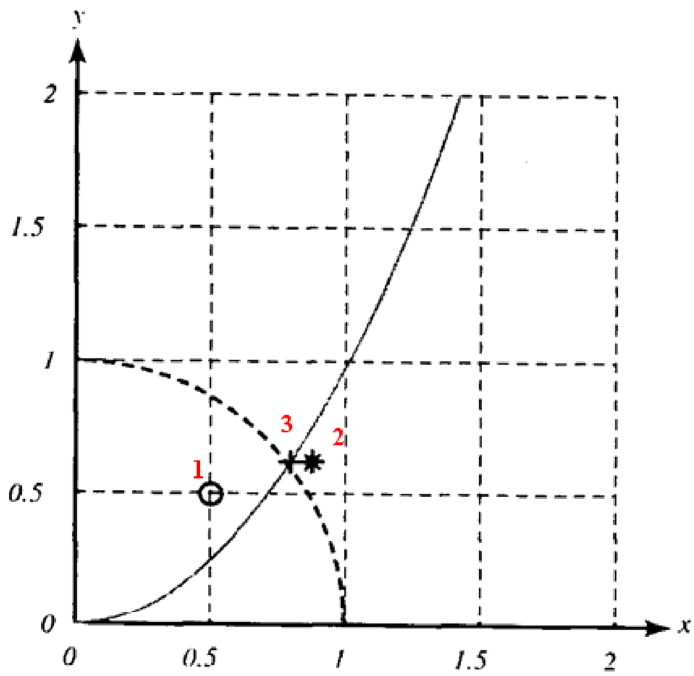
- 兩次迭代的結果：

Step	x	y	$\|\Delta x\|$
0	0.5	0.5	
1	0.875	0.625	0.39528
2	0.79067	0.61806	0.084611

- 其中，最後一行是每一步所得解向量之變量的歐幾里得範數。
- 其真實的解為(x,y)=(0.78615,0.61803)

例題7.1 圓和拋物線的交點

- 牛頓法的初始估計值和兩次迭代



例題7.1 圓和拋物線的交點

- 在每一次迭代，由既有的近似解向量，求得**新的近似解x_{new}**，其迭代方程式為(牛頓法)

$$x_{new} = x_{old} - J^{-1}(x_{old})F(x_{old})$$

其中，J為*Jacobian*矩陣

$$J(x_1,\cdots,x_n) = \begin{vmatrix} \partial f_1/\partial x_1 & \cdots & \partial f_1/\partial x_n \\ \vdots & \ddots & \vdots \\ \partial f_n/\partial x_1 & \cdots & \partial f_n/\partial x_n \end{vmatrix}$$

例題7.1 圓和拋物線的交點

- 但求Jacobian矩陣的逆矩陣是一件非常昂貴的計算工作，所以實務上是解同義的線性方程組

$$J(x_{old})y = -F(x_{old}) \Rightarrow y = -F(x_{old})/J(x_{old})$$

其中

$y = x_{new} - x_{old}$，再更新x進行迭代，得

$$x_{new} = x_{old} + y$$

- 在下列所給的MATLAB程式，以m-file或內嵌函數的方式定義函數F和J。

例題7.1 圓和拋物線的交點

$$\begin{cases} f(x,y) = x^2 + y^2 - 1 \\ g(x,y) = x^2 - y \end{cases}$$

其$Jacobian$為

$$J = \begin{vmatrix} f_x & f_y \\ g_x & g_y \end{vmatrix} = \begin{vmatrix} 2x & 2y \\ 2x & -1 \end{vmatrix}$$

例題7.1 圓和拋物線的交點

- **主程式：Main_Newton_Ex71.m**
- **Newton程式：NewtonSys.m**

```
clear all;
close all;

F=inline('[x(1)^2+x(2)^2-1;x(1)^2-x(2)]');
J=inline('[2*x(1),2*x(2);2*x(1),-1]');
x0=[0.5,0.5]; tol=1e-5;kmax=10;
x=NewtonSys(F,J,x0,tol,kmax)
```

例題7.1 圓和拋物線的交點

- 結果：

i	x1	x2	dx
1.0000	0.8750	0.6250	0.3953
2.0000	0.7907	0.6181	0.0846
3.0000	0.7862	0.6180	0.0045
4.0000	0.7862	0.6180	0.0000
5.0000	0.7862	0.6180	0.0000

```
Newton method has converged

x =  0.7862    0.6180
```


例題7.2 機械臂定位

- 考慮一個兩節式機械手臂，如應用問題7-A所述。令第一節的長度為5且第二節長度為6。希望找到兩個角度α及β，使得機械臂移到點(10,4)，若起始角度為α =0.7及β =0.7。
- 方程組為

$$\begin{cases} 5\cos(\alpha)+6\cos(\alpha+\beta)-10=0 \\ 5\sin(\alpha)+6\sin(\alpha+\beta)-4=0 \end{cases}$$

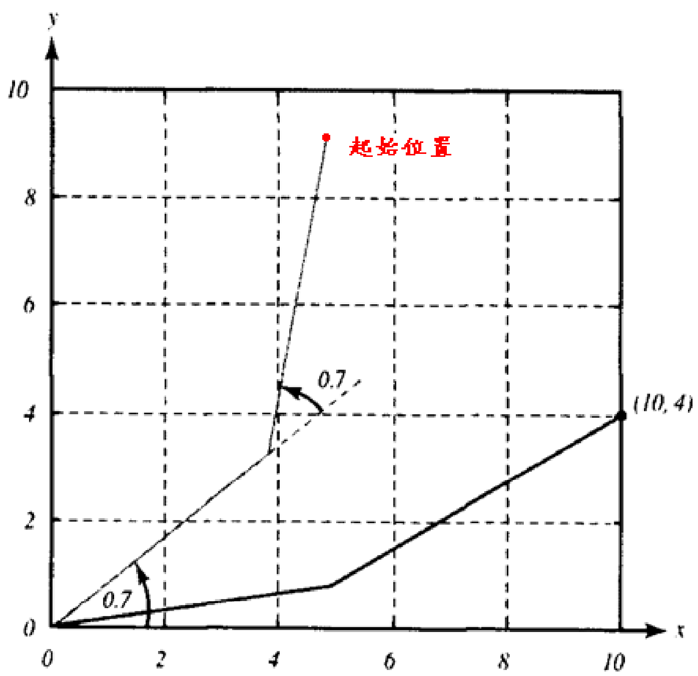
例題7.2 機械臂定位

• 使用牛頓法六次迭代的結果：

Step	α	β	$\|\|Dx\|\|$
0	0.7000	0.7000	
1	−0.5986	1.8339	1.724
2	−0.1078	0.8999	1.0551
3	0.0869	0.5389	0.4101
4	0.1479	0.4260	0.12837
5	0.1558	0.4114	0.016621
6	0.1560	0.4111	0.00029053

例題7.2 機械臂定位

• 機械臂的初始及最終位置圖：



例題7.2 機械臂定位

$$\begin{cases} f(\alpha,\beta)=5\cos(\alpha)+6\cos(\alpha+\beta)-10 \\ g(\alpha,\beta)=5\sin(\alpha)+6\sin(\alpha+\beta)-4 \end{cases}$$

其Jacobian為

$$J=\begin{vmatrix} f_\alpha & f_\beta \\ g_\alpha & g_\beta \end{vmatrix}=\begin{vmatrix} -5\sin(\alpha)-6\sin(\alpha+\beta) & -6\sin(\alpha+\beta) \\ 5\cos(\alpha)+6\cos(\alpha+\beta) & 6\cos(\alpha+\beta) \end{vmatrix}$$

例題7.2 機械臂定位

- **主程式：Main_Newton_Ex72.m**
- **Newton程式：NewtonSys.m**

```
clear all;
close all;

F=inline('[5*cos(x(1))+6*cos(x(1)+x(2))-10;5*sin(x(1))+6*sin(x(1)+x(2))-4]');
J=inline('[-5*sin(x(1))-6*sin(x(1)+x(2)),-
6*sin(x(1)+x(2));5*cos(x(1))+6*cos(x(1)+x(2)),6*cos(x(1)+x(2))]');
x0=[0.7,0.7]; tol=1e-5;kmax=10;
x=NewtonSys(F,J,x0,tol,kmax)
```

例題7.2 機械臂定位

- 結果：

i	x1	x2	dx
1.0000	-0.5985	1.8339	1.7240
2.0000	-0.1078	0.8999	1.0551
3.0000	0.0869	0.5389	0.4101
4.0000	0.1479	0.4260	0.1284
5.0000	0.1558	0.4114	0.0166
6.0000	0.1560	0.4111	0.0003
7.0000	0.1560	0.4111	0.0000

```
Newton method has converged

x =  0.1560    0.4111
```


例題7.3　用於三方程式之方程組的牛頓法

- 以下三個方程式的共同解，它們分別代表了一個以原點為中心的**單位球體**、一個半徑為1/2以x_2軸為軸心的**圓柱體**和一個繞x_3軸的**拋物面體**，即

$$F(x)=\begin{bmatrix} x_1^2+x_2^2+x_3^2-1 \\ x_1^2+x_3^2-1/4 \\ x_1^2+x_2^2-4x_3 \end{bmatrix} \begin{matrix} (\text{球體}) \\ (\text{圓柱體}) \\ (\text{拋物面體}) \end{matrix}$$

其$Jacobian$為

$$\mathrm{J}=\begin{vmatrix} f_{x_1} & f_{x_2} & f_{x_3} \\ g_{x_1} & g_{x_2} & g_{x_3} \\ h_{x_1} & h_{x_2} & h_{x_3} \end{vmatrix}=\begin{vmatrix} 2x_1 & 2x_2 & 2x_3 \\ 2x_1 & 0 & 2x_3 \\ 2x_1 & 2x_2 & -4 \end{vmatrix}$$

例題7.3 用於三方程式之方程組的牛頓法

```
F = inline('[x(1)^2+x(2)^2+x(3)^2-1;
             x(1)^2+x(3)^2-0.25;
             x(1)^2+x(2)^2-4*x(3)]');
J = inline('[2*x(1), 2*x(2), 2*x(3);
             2*x(1), 0,      2*x(3);
             2*x(1), 2*x(2), -4   ]');
x0 = [ 1, 1, 1];    tol = 0.00001;    maxit = 10
x = NewtonSys(F, J, x0, tol, maxit)
```

結果爲：

```
    iter       x(1)      x(2)      x(3)      dx
    1.0000     0.7917    0.8750    0.3333    0.7096
    2.0000     0.5237    0.8661    0.2381    0.2846
    3.0000     0.4473    0.8660    0.2361    0.0764
    4.0000     0.4408    0.8660    0.2361    0.0065
    5.0000     0.4408    0.8660    0.2361    0.0000
    6.0000     0.4408    0.8660    0.2361    0.0000
Newton method has converged
x =
    0.4408    0.8660    0.2361
```


例題7.3 用於三方程式之方程組的牛頓法

- 主程式：Main_Newton_Ex73.m
- Newton程式：NewtonSys.m

```
clear all;
close all;

F=inline('[x(1)^2+x(2)^2+x(3)^2-1;x(1)^2+x(3)^2-0.25;x(1)^2+x(2)^2-4*x(3)]');
J=inline('[2*x(1),2*x(2),2*x(3);2*x(1),0,2*x(3);2*x(1),2*x(2),-4]');
x0=[1,1,1]; tol=1e-5;kmax=10;
x=NewtonSys(F,J,x0,tol,kmax)
```

例題7.3 用於三方程式之方程組的牛頓法

- 結果：

i	x1	x2	x3	dx
1.0000	0.7917	0.8750	0.3333	0.7096
2.0000	0.5237	0.8661	0.2381	0.2846
3.0000	0.4473	0.8660	0.2361	0.0764
4.0000	0.4408	0.8660	0.2361	0.0065
5.0000	0.4408	0.8660	0.2361	0.0000
6.0000	0.4408	0.8660	0.2361	0.0000

```
Newton method has converged

x =     0.4408    0.8660    0.2361
```


討論

- 一個向量函數g(x)存在有固定點的條件，類似於第1章所介紹的，單變數函數存在有固定點的條件。考慮將R_1中的區間$a \leqq x \geqq b$擴展到R^n，假設
  - *g(x)是定義於n維矩形D之上，它是滿足$a_i \leqq x_i \geqq b_i$的所有點$(x_1, x_2, \ldots, x_n)$所成的集合，其中$a_i = a_1, a_2, \ldots, a_n$及$b_i = b_1, b_2, \ldots, b_n$為未定常數。*
  - *Jacobian矩陣G(x)的所有分量，在D上是連續的。*

R^n的固定點收斂定理

- 若g(x)將映射至D，則g在D中有固定點。亦即，若只要x是在D中，g(x)就在D中，則在D中會有某一點p可使得p=g(p)。

  若

  $$\|G(p)\|_\infty < 1$$

  則定義

  $$x^{(k+1)} = g(x^{(k)})$$

  的固定點近似值數列會收斂，只要起始點$x^{(0)}$夠接近固定點p。矩陣範數$\|G(p)\|_\infty$是G最大的列總和(Row sum)。

R^n的固定點收斂定理

- 由固定點收斂定理，可推論出以下結果。若存在有常數K<1，對所有屬於D的x，

  $$\left|\frac{\partial g_i(x)}{\partial x_j}\right| \le \frac{K}{n}$$

- 對每個i=1,…,n和每個j=1,…,n都成立，則對任何起始點$x^{(0)} \in D$固定點的近似值數列會收斂。

R^n的固定點收斂定理

- 在第m步時的誤差界限為

$$\left\| x^{(m)} - p \right\|_{\infty} = \frac{K^m}{1-K} \left\| x^{(1)} - x^{(0)} \right\|_{\infty}$$

- 此結果是收縮映射定理(Contraction mapping theorem)的一個特例。

牛頓法的困難點及解決之道

- 牛頓法的缺點是它需要很好的初始估計值，以及Jacobian矩陣(一次偏導數)。為克服這些困難，發展出數種改進方式。
- 在多維問題中，常使用下降法(Gradient descent)求數值解，相較於使用完整的Jacobian矩陣或其它更全面的數值近似法，此種方法是一個計算效率較好的替代方法。

正割法

- 用以取代牛頓法且**不需計算Jacobian矩陣**的方法叫做正割法。此方法是由Broyden所提出的。
- 用於單變數函數之正割法，其基本觀念是用差分代替微分，所以原來的$f'(x_1)(x_1-x_0)$換成$f(x_1)-f(x_0)$。在多變數函數的情形下，則是用一個滿足$A(x_1-x_0)=F(x_1)-F(x_0)$的矩陣A來取代Jacobian矩陣$J(x_1)$。

正割法

- 和一維的情形不同，這樣並無法唯一決定A。**有幾種不同的方法可決定Jacobian的近似矩陣A**，但一般的要求是，選取新的Jacobian矩陣A_{new}以使得與前一步的變化為最小，與基本方程式一致。得定義為

$$A_{new} = A_{old} + (y - A_{old} * s) * s' / norm(s)^2$$

其中$s = x_{new} - x_{old}$且$y = F(x_{new}) - F(x_{old})$

牛頓法：

$$x_{new} = x_{old} - J^{-1}(x_{old})F(x_{old})$$

例題7.1 使用正割法求圓和拋物線的交點

- 求下列圓和拋物線的交點

$$\begin{cases} f(x,y) = x^2 + y^2 - 1 \\ g(x,y) = x^2 - y \end{cases}$$

例題7.1 使用正割法求圓和拋物線的交點

- **主程式：Main_Broyden_Ex71.m**
- **正割法程式：Broyden.m**

```
clear all;
close all;

F=inline('[x(1)^2+x(2)^2-1;x(1)^2-x(2)]');
J=inline('[2*x(1),2*x(2);2*x(1),-1]');
x0=[0.5; 0.5]; tol=1e-5; kmax=10;
xnew = [0.8; 0.6];

x = Broyden(F,J,x0, xnew,tol, kmax)
```

例題7.1 使用正割法求圓和拋物線的交點

- 結果：

i	x1	x2	dx
1.0000	0.7829	0.6202	0.3074
2.0000	0.7861	0.6180	0.0038
3.0000	0.7862	0.6180	0.0000
4.0000	0.7862	0.6180	0.0000

```
Broyden method has converged

x =    0.7862    0.6180
```


正割法

- 對 Broyden 方法的下一個改良，則是利用 Shermanan-Morrison公式，以由前一個Jacobian逆矩陣求新的Jacobian近似逆矩陣代替，因此減少求更新方程式所須的計算 。
- 假設A 是一個可逆的方陣，u、v 是向量；若

  $1+v^T A^{-1}u \neq 0$ 則Shermanan-Morrison公式為

$$(A+uv^T)^{-1} = A^{-1} - \frac{A^{-1}uv^T A^{-1}}{1+v^T A^{-1}u}$$

- 其中 uv^T 稱為向量 u 與 v 的 Dyadic 積 (Dyadic product：兩相同維度向量之張量積(Tensor product))。

例題7.3 用於三方程式之方程組的正割法

- 以下三個方程式的共同解，它們分別代表了一個以原點為中心的**單位球體**、一個半徑為1/2以x_2軸為軸心的**圓柱體**和一個繞x_3軸的**拋物面體**，即

$$F(x)=\begin{bmatrix} x_1^2+x_2^2+x_3^2-1 \\ x_1^2+x_3^2-1/4 \\ x_1^2+x_2^2-4x_3 \end{bmatrix}\begin{matrix}\text{(球體)}\\ \text{(圓柱體)}\\ \text{(拋物面體)}\end{matrix}$$

其$Jacobian$為

$$J=\begin{vmatrix} f_{x_1} & f_{x_2} & f_{x_3} \\ g_{x_1} & g_{x_2} & g_{x_3} \\ h_{x_1} & h_{x_2} & h_{x_3} \end{vmatrix}=\begin{vmatrix} 2x_1 & 2x_2 & 2x_3 \\ 2x_1 & 0 & 2x_3 \\ 2x_1 & 2x_2 & -4 \end{vmatrix}$$

例題7.3 用於三方程式之方程組的正割法

- 主程式：Main_Broyden_SM_Ex73.m
- 正割法程式：Broyden_SM.m

```
clear all;
close all;

F = inline('[ x(1)^2+x(2)^2+x(3)^2-1;x(1)^2+x(3)^2-0.25;x(1)^2+x(2)^2-
4*x(3)]');
J=inline('[2*x(1),2*x(2),2*x(3);2*x(1),0,2*x(3);2*x(1),2*x(2),-4]');
tol = 1e-5; kmax=10; x0 = [ 1;1;1];
%J = [ 2  2   2; 2  0   2; 2  2  -4 ] ;

x = Broyden_SM(F,J,x0,tol, kmax)
```

例題7.3　用於三方程式之方程組的正割法

- 結果：

i	x(1)	x(2)	x(3)
0	1	1	1
0.5000	0.7917	0.8750	0.3333
1.0000	0.5870	0.8656	0.2441
2.0000	0.4553	0.8653	0.2260
3.0000	0.4311	0.8659	0.2327
4.0000	0.4347	0.8660	0.2350
5.0000	0.4425	0.8660	0.2368
6.0000	0.4411	0.8660	0.2362
7.0000	0.4407	0.8660	0.2360
8.0000	0.4408	0.8660	0.2361
9.0000	0.4408	0.8660	0.2361

Broyden with SM method has converged

x =   0.4408   0.8660   0.2361

固定點迭代

- 固定點迭代除是一個功能很強的分析工具，它也可做為實際的解法。在第1章中，有一個例題是用固定點迭代求一個單變數非線性函數的零點。將此觀念推廣到方程組是很簡單的。

例題7.4　兩個聯立函數的固定點迭代

- 求以下方程組零點的問題

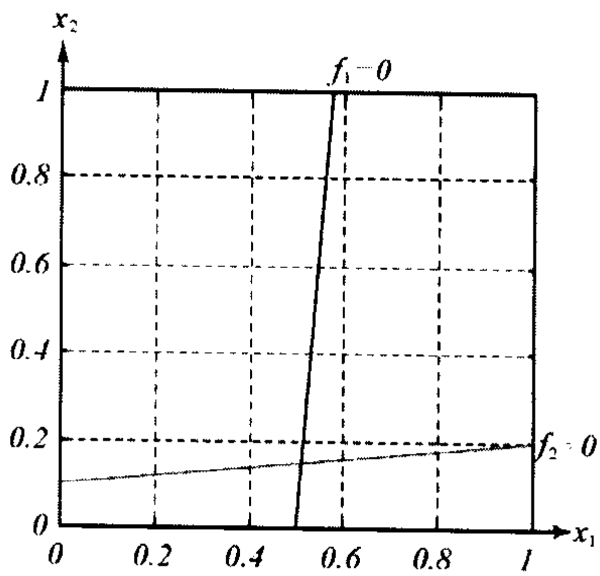
$$\begin{cases} f_1(x_1,x_2)=x_1^3+10x_1-x_2-5=0 \\ f_2(x_1,x_2)=x_1+x_2^3-10x_2+1=0 \end{cases}$$

- 將這兩個方程式轉換成迭代方程組

$$\begin{cases} x_1=g_1(x_1,x_2)=-0.1x_1^3+0.1x_2+0.5 \\ x_2=g_2(x_1,x_2)=0.1x_1+0.1x_2^3+0.1 \end{cases}$$

例題7.4　兩個聯立函數的固定點迭代

- 兩個聯立非線性方程式圖形



例題7.4　兩個聯立函數的固定點迭代

- 執行結果：

Step	x_1	x_2	$\|\Delta x\|$
0	0.6000	0.6000	
1	0.5384	0.1816	0.42291
2	0.50255	0.15444	0.044975
3	0.50275	0.15062	0.0038204
4	0.50235	0.15062	0.00039661
5	0.50238	0.15058	4.9381e-05

例題7.4　兩個聯立函數的固定點迭代

- 主程式：Main_FixedPtSys_Ex74.m
- 固定點迭代法程式：FixedPtSys.m

```
G=inline('[-0.1*x(1)^3+0.1*x(2)+0.5;0.1*x(1)+0.1*x(2)^3+0.1]');
x0=[0.6,0.6]; tol=1e-5;kmax=10;
x=FixedPtSys(G,x0,tol,kmax)
```

例題7.4　兩個聯立函數的固定點迭代

- 結果：

```
       i           x1          x2          dx
       0         0.6000      0.6000
    1.0000       0.5384      0.1816      0.4229
    2.0000       0.5026      0.1544      0.0450
    3.0000       0.5028      0.1506      0.0038
    4.0000       0.5024      0.1506      0.0004
    5.0000       0.5024      0.1506      0.0000

Fixed-point iteration converged

x = 0.5024    0.1506
```


最小化

- 求一個多變數純量函數的最小值的問題。在許多不同應用中，最小化問題都相當重要。方法有：
- 牛頓法(Newton-Raphson)
- 最大陡降法 (Steepest descent)
- 準牛頓法(Quasi-Newton)
  - *Davidon-Fletcher-Powell (DFP) 法 (或稱 Fletcher-Powell)*
  - *Broyden-Fletcher-Goldfarb-Shano(BFGS)法*

牛頓法(Newton-Raphson)

最小化：$f(x)$

牛頓法迭代公式：$x_{new} = x_{old} - H^{-1}(x_{old})\nabla f(x_{old})$

其中，$f(x)$為欲最小化之方程式，$\nabla f(x)$為f之梯度$(Gradient)$

$$\nabla f(x_1,\cdots,x_n) = \begin{vmatrix} \partial f / \partial x_1 \\ \vdots \\ \partial f / \partial x_n \end{vmatrix}$$

$H(x)$為$Hessian$矩陣

$$\mathrm{H}(x_1,\cdots,x_n) = \nabla^2 f(x_1,\cdots,x_n) = \begin{vmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_1 \partial x_n} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{vmatrix}$$

習題P7.12 使用牛頓法求$f^2+g^2+h^2$之最小值

$$f(x,y,z) = x^2 + 20x + y^2 + z^2 - 20$$

$$g(x,y,z) = x^2 + 20y + z^2 - 20$$

$$h(x,y,z) = x^2 + y^2 - 40z$$

最小化$F(x) = f^2 + g^2 + h^2$

習題P7.12 使用牛頓法求f²+g²+h²之最小值

$$\nabla f(X)=\begin{bmatrix}\partial f/\partial x_1\\ \vdots\\ \partial f/\partial x_n\end{bmatrix}=\begin{bmatrix}2f(2x+20)+2g(2x)+2h(2x)\\ 2f(2g)+2g(20)+2h(2y)\\ 2f(2z)+2g(2z)+2h(-40)\end{bmatrix}$$

$$\mathrm{H}(X)=\nabla^2 f(X)=\begin{bmatrix}\frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n}\\ \vdots & \ddots & \vdots\\ \frac{\partial^2 f}{\partial x_1 \partial x_n} & \cdots & \frac{\partial^2 f}{\partial x_n^2}\end{bmatrix}$$

$$=\begin{bmatrix}2(2x+20)^2+2f(2)+2(2x)^2+2g(2)+2(2x)^2+2h(2) & 2(2y)(2x+20)+2(20)(2x)+2(2y)(2x) & 2(2z)(2x+20)+2(2z)(2x)+2(-40)(2x)\\ 2(2x+20)(2y)+40(2x)+2(2y)(2x) & 2(2y)^2+2f(2)+2(20)(20)+2(2y)^2+4h & 2(2z)(2y)+2(2z)(20)+2(-40)(2y)\\ 2(2x+20)(2z)+2(2x)(2z)+(-80)(2x) & 2(2y)(2z)+2(20)(2z)-80(2y) & 2(2z)^2 4f+2(2z)^2+4g+(-80)(-40)\end{bmatrix}$$

習題P7.12 使用牛頓法求f²+g²+h²之最小值

- **主程式： Main_Newton_Raphson_Q712.m**
- **牛頓法程式：Newton_Raphson.m**

```
clear all;
close all;

%x0=rand(1,3);
x0=[0.2 0.2 0.2];
tol=1e-10; kmax=10000;
xmin = Newton_Raphson(@Q712, @grad_Q712, @hessian_Q712, x0, tol, kmax)
```

習題P7.12 使用牛頓法求$f^2+g^2+h^2$之最小值

```
function d=Q712(x)

f=x(1)^2+20*x(1)+x(2)^2+x(3)^2-20;
g=x(1)^2+20*x(2)+x(3)^2-20;
h=x(1)^2+x(2)^2-40*x(3);
d=f^2+g^2+h^2;
```

```
function gg=grad_Q712(x)

f=x(1)^2+20*x(1)+x(2)^2+x(3)^2-20;
g=x(1)^2+20*x(2)+x(3)^2-20;
h=x(1)^2+x(2)^2-40*x(3);
gg=[2*f*(2*x(1)+20)+4*g*x(1)+4*h*x(1)    2*f*(2*x(2))+40*g+4*h*x(2)
            2*f*(2*x(3))+4*g*x(3)-80*h];
```


習題P7.12 使用牛頓法求$f^2+g^2+h^2$之最小值

```
function hh=hessian_Q712(x)

f=x(1)^2+20*x(1)+x(2)^2+x(3)^2-20;
g=x(1)^2+20*x(2)+x(3)^2-20;
h=x(1)^2+x(2)^2-40*x(3);

df11=2*(2*x(1)+20)^2+4*f+8*x(1)^2+4*g+8*x(1)^2+4*h;
df12=4*x(2)*(2*x(1)+20)+80*x(1)+8*x(1)*x(2);
df13=4*x(3)*(2*x(1)+20)+8*x(1)*x(3)-160*x(1);

df21=4*(2*x(1)+20)*x(2)+80*x(1)+8*x(1)*x(2);
df22=8*x(2)^2+4*f+800+8*x(2)^2+4*h;
df23=8*x(2)*x(3)+80*x(3)-160*x(2);

df31=4*(2*x(1)+20)*x(3)+8*x(1)*x(3)-160*x(1);
df32=8*x(2)*x(3)+80*x(3)-160*x(2);
df33=8*x(3)^2+4*f+8*x(3)^2+4*g+3200;

hh=[df11 df12 df13; df21 df22 df23; df31 df32 df33];
```

習題P7.12 使用牛頓法求$f^2+g^2+h^2$之最小值

- 結果：

i	x1	x2	x3	dx
1.0000	1.1425	1.0819	0.0113	1.3044
2.0000	0.9237	0.9629	0.0426	0.2509
3.0000	0.9124	0.9583	0.0438	0.0123
4.0000	0.9124	0.9583	0.0438	0.0000
5.0000	0.9124	0.9583	0.0438	0.0000
6.0000	0.9124	0.9583	0.0438	0.0000

Fun_Value = 1.9722e-031

Newton-Raphson method has converged

xmin = 0.9124 0.9583 0.0438

Gauss-Newton法

$$x_{new} = x_{old} - (J^T J)^{-1} J^T f(x_{old})$$

其中，$f(x)$為欲求最小值之目標函數，

J為*Jacobian*矩陣

$$J(x_1, \cdots, x_n) = \begin{bmatrix} \partial f / \partial x_1 \\ \vdots \\ \partial f / \partial x_n \end{bmatrix}$$

習題P7.12 使用Gauss-Newton法求$f^2+g^2+h^2$之最小值

$$f(x,y,z) = x^2 + 20x + y^2 + z^2 - 20$$

$$g(x,y,z) = x^2 + 20y + z^2 - 20$$

$$h(x,y,z) = x^2 + y^2 - 40z$$

最小化$F(x) = f^2 + g^2 + h^2$

習題P7.12 使用Gauss-Newton法求$f^2+g^2+h^2$之最小值

$$J(X) = \nabla f(X) = \begin{bmatrix} \partial f / \partial x_1 \\ \vdots \\ \partial f / \partial x_n \end{bmatrix} = \begin{bmatrix} 2f(2x+20) + 2g(2x) + 2h(2x) \\ 2f(2g) + 2g(20) + 2h(2y) \\ 2f(2z) + 2g(2z) + 2h(-40) \end{bmatrix}$$

習題P7.12 使用Gauss-Newton法求$f^2+g^2+h^2$之最小值

- 主程式： Main_GaussNewton_Q712.m
- Gauss-Newton程式：GaussNewton.m

```
clear all;
close all;

%x0=rand(1,3);
x0=[0.2 0.2 0.2];
tol=1e-10; kmax=10000;
xmin = GaussNewton (@Q712, @grad_Q712, x0, tol, kmax)
```


習題P7.12 使用Gauss-Newton法求$f^2+g^2+h^2$之最小值

```
function d=Q712(x)

f=x(1)^2+20*x(1)+x(2)^2+x(3)^2-20;
g=x(1)^2+20*x(2)+x(3)^2-20;
h=x(1)^2+x(2)^2-40*x(3);
d=f^2+g^2+h^2;
```

```
function gg=grad_Q712(x)

f=x(1)^2+20*x(1)+x(2)^2+x(3)^2-20;
g=x(1)^2+20*x(2)+x(3)^2-20;
h=x(1)^2+x(2)^2-40*x(3);
gg=[2*f*(2*x(1)+20)+4*g*x(1)+4*h*x(1)    2*f*(2*x(2))+40*g+4*h*x(2)
            2*f*(2*x(3))+4*g*x(3)-80*h];
```

習題P7.12 使用Gauss-Newton法求$f^2+g^2+h^2$之最小值

- 結果：

i	x1	x2	x3	dx
1.0000	0.5045	0.4994	-0.0777	0.5094
2.0000	0.7010	0.6932	0.0588	0.3079
3.0000	0.8105	0.8071	0.0083	0.1658
4.0000	0.8644	0.8696	0.0451	0.0904
5.0000	0.8930	0.9076	0.0339	0.0489
6.0000	0.9054	0.9286	0.0444	0.0266
----------	----------	----------	----------	----------
47.0000	0.9124	0.9583	0.0438	0.0000
48.0000	0.9124	0.9583	0.0438	0.0000

Fun_Value = 8.0080e-018

Gauss-Newton method has converged

xmin = 0.9124 0.9583 0.0438

最大陡降法 (Steepest gradient descent)

- 一個多變數函數的梯度(Gradient)，是一個指向增加最快之方向的向量(Ascent)，負的梯度(Descent)則為減少最快的方向。
- 使用最大陡降法的主要問題是，沿著負梯度方向要走多遠(步距)。在它最基本的形式中，我們沿著負梯度方向前進一指定的距離，求得新的近似解，然後將新位置的函數值和舊位置的做比較。如果新的值並沒有比較小，則減小沿著負梯度方向前進的距離。

最大陡降法 (Steepest gradient descent)

- 有一種較複雜的方法可用來決定步距(沿梯度方向前進的距離)，基本上它用二次式來近似要最小化的函數g，用梯度方向上三個相異的x值，也就是參數a的三個不同的值，其中新的x將是x=x-α*g。

問題7-B 使用最大陡降法求最小總距離

- 已知平面上三點(x_1,y_1)，(x_2,y_2)及(x_3,y_3) ，希望求出點p(x,y)的位置，使得點p到三已知點距離的平方和為最小。即要求以下函數的最小值

$$f(x,y)=(x-x_1)^2+(y-y_1)^2+(x-x_2)^2+(y-y_2)^2+(x-x_3)^2+(y-y_3)^2$$ 。

所須的導數爲

$$f_x(x,y)=2(x-x_1)+2(x-x_2)+2(x-x_3)=6x-2x_1-2x_2-2x_3$$
$$f_y(x,y)=2(y-y_1)+2(y-y_2)+2(y-y_3)=6y-2y_1-2y_2-2y_3$$

問題7-B 使用最大陡降法求最小總距離

- 主程式：Main_Steepest_Descent_Ex7B.m
- 最大陡降法程式：Steepest_Descent.m

```
clear all;
close all;

x0=[0.2,0.2]; tol=1e-5; kmax=10;
xmin = Steepest_Descent(@sum_dist, @grad_dist, x0, tol, kmax)
```


問題7-B 使用最大陡降法求最小總距離

```
function d=sum_dist(x)

p1=[0,0]; p2=[1,0]; p3=[1/2,1];
x1=p1(1); y1=p1(2); x2=p2(1); y2=p2(2); x3=p3(1); y3=p3(2);
d=(x(1)-x1)^2+(x(2)-y1)^2+(x(1)-x2)^2+(x(2)-y2)^2+(x(1)-x3)^2+(x(2)-y3)^2;
```

```
function gg=grad_dist(x)

p1=[0,0]; p2=[1,0]; p3=[1/2,1];
x1=p1(1); y1=p1(2); x2=p2(1); y2=p2(2); x3=p3(1); y3=p3(2);
gg=[2*(x(1)-x1)+2*(x(1)-x2)+2*(x(1)-x3)  2*(x(2)-y1)+2*(x(2)-y2)+2*(x(2)-y3)];
```

問題7-B 使用最大陡降法求最小總距離

- 程式問題：

```
………………..
while f3 >= f1
    a3 = a3/2;      f3 = feval(my_f, x-a3*g);
    if a3 < tol/2
      disp('No improvement likely');
      xmin = x;
      return;
    end
  end
…………………
```


問題7-B 使用最大陡降法求最小總距離

- 結果：

```
  iter      x(l)       x(2)
   0       0.2000     0.2000
1.0000     0.5000     0.3333

No improvement likely

xmin =    0.5000    0.3333
```

習題P7.12 使用最大陡降法求$f^2+g^2+h^2$之最小值

$$f(x,y,z) = x^2 + 20x + y^2 + z^2 - 20$$
$$g(x,y,z) = x^2 + 20y + z^2 - 20$$
$$h(x,y,z) = x^2 + y^2 - 40z$$

最小化$F(x) = f^2 + g^2 + h^2$

習題P7.12 使用最大陡降法求$f^2+g^2+h^2$之最小值

- 主程式：Main_Steepest_Descent_Q712.m
- 最大陡降法程式：Steepest_Descent1.m

```
clear all;
close all;

x0=[0.2,0.2,0.2]; tol=1e-5; kmax=100;
xmin = Steepest_Descent1(@Q712, @grad_Q712, x0, tol, kmax)
```

習題P7.12 使用最大陡降法求$f^2+g^2+h^2$之最小值

```
function d=Q712(x)

f=x(1)^2+20*x(1)+x(2)^2+x(3)^2-20;
g=x(1)^2+20*x(2)+x(3)^2-20;
h=x(1)^2+x(2)^2-40*x(3);
d=f^2+g^2+h^2;
```

```
function gg=grad_Q712(x)

f=x(1)^2+20*x(1)+x(2)^2+x(3)^2-20;
g=x(1)^2+20*x(2)+x(3)^2-20;
h=x(1)^2+x(2)^2-40*x(3);
gg=[2*f*(2*x(1)+20)+4*g*x(1)+4*h*x(1)    2*f*(2*x(2))+40*g+4*h*x(2)
          2*f*(2*x(3))+4*g*x(3)-80*h];
```


習題P7.12 使用最大陡降法求$f^2+g^2+h^2$之最小值

- 結果：

iter	x(l)	x(2)	x(3)
0	0.2000	0.2000	0.2000
1.0000	0.6149	0.6079	-0.1783
2.0000	0.7407	0.7347	0.0881
3.0000	0.8486	0.8492	-0.0169
4.0000	0.8784	0.8878	0.0557
5.0000	0.9027	0.9222	0.0274
6.0000	0.9082	0.9346	0.0472
……………			
36.0000	0.9124	0.9583	0.0438
37.0000	0.9124	0.9583	0.0438

No improvement likely

Fun_Value = 2.1921e-019

xmin = 0.9124 0.9583 0.0438

最大陡降法 (Steepest gradient descent)

- 雖然求參數a需要一些額外的計算，但比起基本形式只是將步距減半，直到找到適當值為止，現在的方法收斂快多了，對於應用問題7-B，此函數一次迭代就找到最小值。
- 任何形式的最大陡降法都有一個基本缺點，每一步求得的梯度都垂直於上一步的梯度，使它以Z字形的方式趨近最小值。

準牛頓法 (Quasi-Newton Methods)

- 在解非線性方程組，和求多變數非線性函數最小值的問題之間，有很強的關聯性。以牛頓法求最小值的方法，就建立在這樣一種關聯上。由此引出了一組很重要的求最小值的方法，稱做準牛頓法。
- 在使用牛頓法求多變數函數(可緊緻的表示為g(x))最小值時，在最小值的位置，未知數向量的所有分量都有 $\partial g / \partial x_i = 0$ 。這樣就得到非線性方程組 $\nabla g(x) = 0$ 。

準牛頓法 (Quasi-Newton Methods)

- 主要的兩種準牛頓法為Davidon-Fletcher-Powell(DFP) 法 (或稱 Fletcher-Powell) 及 Broyden-Fletcher-Goldfarb-Shanno(BFGS)法。
- 用一個二次函數來近似要求最小值的函數f(x)，亦即 $f(x) \approx c - bx + \frac{1}{2} xAx$ 。但是並不知道A。計畫是建立一序列逆Hessian矩陣的近似解 $H_i \to A^{-1}$。

準牛頓法 (Quasi-Newton Methods)

- 如果x_i是第i步時，可使f為最小的向量x的近似解，且g(x)為f的梯度，則可寫成(到二階精度)

  $$f(x) = f(x_i) + (x - x_i) g(x_i) + \frac{1}{2}(x - x_i) A (x - x_i)$$

  可得

  $$g(x) = g(x_i) + A(x - x_i)$$

- 如果已經知道A^{-1}，就可以像標準牛頓法一樣設g(x)=0並求解$(x-x_i)$。不過使用現有的A^{-1}的近似解，而通常這會比使用真實的Hessian矩陣效果還好。會出現這種矛盾現象是因為如果距離最小值甚遠，無法保證Hessian矩陣是正定的，但在牛頓法中它必須是正定的才能指出讓函數值減小的方向。所建構之一序列近似值H_i必定都是正定矩陣。

準牛頓法 (Quasi-Newton Methods)

- 將準牛頓法用於此方程組，其更新方程式為

$$x_{new} = x - H^{-1}(x)\nabla f(x)$$

其中H為$Hessian$矩陣，$H = \partial^2 f / (\partial x_i \partial x_j)$

準牛頓法是使用A(x)來取代$H^{-1}(x)$

$$A^{(k+1)} = A^{(k)} + \alpha E^{(k)}$$

(1) $DFP(Davidon - Fletcher - Powell)$公式：$g = \nabla f(x)$

$$E^{(k)} = \frac{\Delta X^{(k)}[\Delta X^{(k)}]^T}{[\Delta X^{(k)}]^T \Delta g^{(k)}} - \frac{A^{(k)}\Delta g^{(k)}[\Delta g^{(k)}]^T [A^{(k)}]^T}{[\Delta g^{(k)}]^T A^{(k)} \Delta g^{(k)}}$$

(2) $BFGS(Broyden - Fletxher - Goldfarb - Shanno)$公式：

$$E^{(k)} = \frac{1}{[\Delta X^{(k)}]^T \Delta g^{(k)}} \{\Delta X^{(k)}[\Delta X^{(k)}]^T + \frac{\Delta X^{(k)}[\Delta X^{(k)}]^T \cdot [\Delta g^{(k)}]^T A^{(k)} \Delta g^{(k)}}{[\Delta X^{(k)}]^T \Delta g^{(k)}}$$
$$- A^{(k)}\Delta g^{(k)}[\Delta X^{(k)}]^T - \Delta X^{(k)}[\Delta g^{(k)}]^T A^{(k)}\}$$

問題7-B 使用準牛頓法求最小總距離

- 已知平面上三點(x_1,y_1)，(x_2,y_2)及(x_3,y_3) ，希望求出點p(x,y)的位置，使得點p到三已知點距離的平方和為最小。即要求以下函數的最小值

$$f(x,y) = (x-x_1)^2 + (y-y_1)^2 + (x-x_2)^2 + (y-y_2)^2 + (x-x_3)^2 + (y-y_3)^2 \text{ 。}$$

所須的導數爲

$$f_x(x,y) = 2(x-x_1) + 2(x-x_2) + 2(x-x_3) = 6x - 2x_1 - 2x_2 - 2x_3$$
$$f_y(x,y) = 2(y-y_1) + 2(y-y_2) + 2(y-y_3) = 6y - 2y_1 - 2y_2 - 2y_3$$

問題7-B 使用準牛頓法求最小總距離

- 主程式：Main_DFP_Ex7B.m
- 準牛頓法程式：DFP.m

```
clear all;
close all;

x0=[0.2,0.2]; tol=1e-5; kmax=10;
xmin = DFP(@sum_dist, @grad_dist, x0, tol, kmax)
```


問題7-B 使用準牛頓法求最小總距離

```
function d=sum_dist(x)

p1=[0,0]; p2=[1,0]; p3=[1/2,1];
x1=p1(1); y1=p1(2); x2=p2(1); y2=p2(2); x3=p3(1); y3=p3(2);
d=(x(1)-x1)^2+(x(2)-y1)^2+(x(1)-x2)^2+(x(2)-y2)^2+(x(1)-x3)^2+(x(2)-y3)^2;
```

```
function g=grad_dist(x)

p1=[0,0]; p2=[1,0]; p3=[1/2,1];
x1=p1(1); y1=p1(2); x2=p2(1); y2=p2(2); x3=p3(1); y3=p3(2);
g=[2*(x(1)-x1)+2*(x(1)-x2)+2*(x(1)-x3)  2*(x(2)-y1)+2*(x(2)-y2)+2*(x(2)-y3)];
```

問題7-B 使用準牛頓法求最小總距離

- 結果：

iter	x(l)	x(2)	f_new
1.0000	0.5000	0.3333	1.1667

No improvement likely

xmin =
0.5000
0.3333

進階問題

- Levenbefg-Marquardt法
  - *特別適用於解最小平方問題*
- Nelder-Mead形式的單體搜尋法，它是MATLAB內建函數fminsearch所用的方法。此方法也被叫做下坡單體法(Downhill simplex method)

Levenberg-Marquardt法

- Levenberg-Marquardt(L-M)演算法主要是用於將一個模型(此模型取決於數個參數)與一組數據做擬合的問題。為求得最佳擬合(Best-fit)的參數值，針對一個可以衡量數據與模型之吻合度的函數，依特定的參數值將此函數最小化。此一函數，通常叫做「優點函數(Merit function)」，它在設計上，是當某些參數值能夠讓模型與數據有較佳的吻合度時，函數值變小。因為最常使用的優點函數就是模型值與數據值差的平方和，所以此種問題通常被叫做最小平方問題。

Levenberg-Marquardt法

- L-M法是使用兩種方法求優點函數的最小值，並使得兩種方法能平順的轉換：最陡下降法 (當近似解與所要答案相差尚遠時使用)及逆Hessian法(在接近優點函數最小值時使用)。L-M使用經修改的Hessian矩陣，它可記為$M=[m_{ij}]$，其中

$$m_{ij} = \begin{cases} \frac{1}{2}\frac{\partial^2 f}{\partial x_i \partial x_j}, & i \neq j \\ \frac{1}{2}\frac{\partial^2 f}{\partial x_i \partial x_j}(1+\lambda), & i = j \end{cases}$$

Levenberg-Marquardt法

- 定義向量b，使得 $b_i = -\partial f / \partial x_i$。其基本觀念是，由向量x的初始估計值開始，解線性方程組Md=b以得到修正向量d，然後依據f(x+d)比f(x)大或小，以決定λ值的增減。

Levenberg-Marquardt法

- 依此方法持續迭代，直到f減小的量遠小於1為止。此程序可表示成以下演算法：

```
   Compute f(x), M(x), b(x)
   Set L = 0.001                        % lambda初始化
** Begin iteration loop
   Solve M d = b,    Compute f(x + d)
   If f(x + d) >= f(x)                  % 加大lambda
       Set L = 10 L;  return to ** and continue iterations
   If f(x + d) < f(x)                   % 減小lambda，或停止
       If f(x) - f(x+d) < 0.01, stop
       Otherwise, set L = 0.1 L;  return to ** and continue iterations
```

Newton-Raphson_LM

最小化：$f(x)$

牛頓法迭代公式：$x_{new} = x_{old} - [H(x_{old}) + \lambda I]^{-1} \nabla f(x_{old})$

其中，$f(x)$為欲最小化之方程式，$\nabla f(\mathrm{x})$為f之梯度(*Gradient*)

$$\nabla f(x_1,\cdots,x_n) = \begin{vmatrix} \partial f / \partial x_1 \\ \vdots \\ \partial f / \partial x_n \end{vmatrix}$$

$H(x)$為*Hessian*矩陣

$$\mathrm{H}(x_1,\cdots,x_n) = \nabla^2 f(x_1,\cdots,x_n) = \begin{vmatrix} \dfrac{\partial^2 f}{\partial x_1^2} & \cdots & \dfrac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial^2 f}{\partial x_1 \partial x_n} & \cdots & \dfrac{\partial^2 f}{\partial x_n^2} \end{vmatrix}$$

習題P7.12 使用牛頓_LM法求f²+g²+h²之最小值

$$f(x,y,z) = x^2 + 20x + y^2 + z^2 - 20$$

$$g(x,y,z) = x^2 + 20y + z^2 - 20$$

$$h(x,y,z) = x^2 + y^2 - 40z$$

最小化$F(x) = f^2 + g^2 + h^2$

習題P7.12 使用牛頓_LM法求$f^2+g^2+h^2$之最小值

$$\nabla f(X)=\begin{bmatrix}\partial f/\partial x_1\\ \vdots\\ \partial f/\partial x_n\end{bmatrix}=\begin{bmatrix}2f(2x+20)+2g(2x)+2h(2x)\\ 2f(2g)+2g(20)+2h(2y)\\ 2f(2z)+2g(2z)+2h(-40)\end{bmatrix}$$

$$\mathrm{H}(X)=\nabla^2 f(X)=\begin{bmatrix}\frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1\partial x_n}\\ \vdots & \ddots & \vdots\\ \frac{\partial^2 f}{\partial x_1\partial x_n} & \cdots & \frac{\partial^2 f}{\partial x_n^2}\end{bmatrix}$$

$$=\begin{bmatrix}2(2x+20)^2+2f(2)+2(2x)^2+2g(2)+2(2x)^2+2h(2) & 2(2y)(2x+20)+2(20)(2x)+2(2y)(2x) & 2(2z)(2x+20)+2(2z)(2x)+2(-40)(2x)\\ 2(2x+20)(2y)+40(2x)+2(2y)(2x) & 2(2y)^2+2f(2)+2(20)(20)+2(2y)^2+4h & 2(2z)(2y)+2(2z)(20)+2(-40)(2y)\\ 2(2x+20)(2z)+2(2x)(2z)+(-80)(2x) & 2(2y)(2z)+2(20)(2z)-80(2y) & 2(2z)^2 4f+2(2z)^2+4g+(-80)(-40)\end{bmatrix}$$

習題P7.12 使用牛頓_LM法求$f^2+g^2+h^2$之最小值

- 主程式： Main_Newton_LM_Q712.m
- 牛頓_LM法程式：Newton_LM.m

```
clear all;
close all;

%x0=rand(1,3);
x0=[0.2 0.2 0.2];
tol=1e-10; kmax=100;
xmin = Newton_LM(@Q712, @grad_Q712,@hessian_Q712, x0, tol, kmax)
```

習題P7.12 使用牛頓_LM法求$f^2+g^2+h^2$之最小值

```
function d=Q712(x)

f=x(1)^2+20*x(1)+x(2)^2+x(3)^2-20;
g=x(1)^2+20*x(2)+x(3)^2-20;
h=x(1)^2+x(2)^2-40*x(3);
d=f^2+g^2+h^2;
```

```
function gg=grad_Q712(x)

f=x(1)^2+20*x(1)+x(2)^2+x(3)^2-20;
g=x(1)^2+20*x(2)+x(3)^2-20;
h=x(1)^2+x(2)^2-40*x(3);
gg=[2*f*(2*x(1)+20)+4*g*x(1)+4*h*x(1)    2*f*(2*x(2))+40*g+4*h*x(2)
            2*f*(2*x(3))+4*g*x(3)-80*h];
```


習題P7.12 使用牛頓_LM法求$f^2+g^2+h^2$之最小值

```
function hh=hessian_Q712(x)

f=x(1)^2+20*x(1)+x(2)^2+x(3)^2-20;
g=x(1)^2+20*x(2)+x(3)^2-20;
h=x(1)^2+x(2)^2-40*x(3);

df11=2*(2*x(1)+20)^2+4*f+8*x(1)^2+4*g+8*x(1)^2+4*h;
df12=4*x(2)*(2*x(1)+20)+80*x(1)+8*x(1)*x(2);
df13=4*x(3)*(2*x(1)+20)+8*x(1)*x(3)-160*x(1);

df21=4*(2*x(1)+20)*x(2)+80*x(1)+8*x(1)*x(2);
df22=8*x(2)^2+4*f+800+8*x(2)^2+4*h;
df23=8*x(2)*x(3)+80*x(3)-160*x(2);

df31=4*(2*x(1)+20)*x(3)+8*x(1)*x(3)-160*x(1);
df32=8*x(2)*x(3)+80*x(3)-160*x(2);
df33=8*x(3)^2+4*f+8*x(3)^2+4*g+3200;

hh=[df11 df12 df13; df21 df22 df23; df31 df32 df33];
```

習題P7.12 使用牛頓_LM法求$f^2+g^2+h^2$之最小值

- 結果：

i	x1	x2	x3	dx
1.0000	1.1425	1.0819	0.0113	1.3044
2.0000	0.9237	0.9629	0.0426	0.2509
3.0000	0.9124	0.9583	0.0438	0.0123
4.0000	0.9124	0.9583	0.0438	0.0000
5.0000	0.9124	0.9583	0.0438	0.0000
6.0000	0.9124	0.9583	0.0438	0.0000
7.0000	0.9124	0.9583	0.0438	0.0000

Fun_Value = 1.8216e-017

Gauss-Newton method has converged

xmin = 0.9124 0.9583 0.0438

Nelder-Mead單體搜尋法

- 要求一個多變數函數的最小值，MATLAB的內建函數用的是Nelder-Mead形式單體搜尋。先對此方法作一概述，然後列出使用此MATLAB函數的各種選項。
- 在二維的情況下，一個單體就是三個點 (頂點)以及連接這些點的所有線段，所構成的幾何形體一一換句話說，就是三角形。在三維的時候，一個單體四面體，亦即，四個頂點和連接這些點的多邊形。在R^n中，一個單體包含n+1個點，和連接這些點的超平面。

Nelder-Mead單體搜尋法

- 為求簡化只限於說明二維單體。此方法由一個初始單體 (三角形)開始，並找出可以使該函數減小幅度最大的一個頂點(叫它「壞的」)。單體搜尋法在每一次迭代，都會將單體加以轉換，轉換方式結合反射、膨脹及收縮。
- 反射是將單體「壞的」頂點，投影到單體對面的中點。如果新點比較好 (亦即，用「新」點求得的函數值較小)，此方法會將「新」點沿著投影線移動，使得單體膨脹，直到無法獲得進一步改進為止。
- 其它的移動還有，將「壞」點沿投影線縮向對邊中點，以及將所有頂點 (除最好的一個之外)向最好點收縮。

Nelder-Mead單體搜尋法

- MATLAB的fminsearch函數即應用單體搜尋法。此函數基本呼叫方式為

  x=fminsearch('F',x0)