



Mathematica 在最佳化上的應用

戴忠淵

cydye@stu.edu.tw

May 25, 2022

目 錄

1	微積分的基本操作	1
1.1	極限	1
1.2	微分	5
1.3	全微分	11
1.4	積分	16
1.5	級數	39
2	方程式求解	49
2.1	方程式具解析解	49
2.2	方程式不具解析解	64
3	線性規劃	79
3.1	線性規劃問題及單形法	79
3.2	對偶理論	101
4	非線性規劃	109
4.1	無限制條件下非線性規劃	109
4.1.1	梯度下降法或最陡坡降法 (Gradient descent or Steepest descent method)	110
4.1.2	牛頓法 (Newton method)	111
4.1.3	擬牛頓法 (Quasi-Newton method)	111
4.1.4	共軛梯度法 (Conjugate Gradient method)	113
4.1.5	萊文貝格-馬夸特法 (Levenberg-Marquardt method)	114
4.1.6	範例.	115
4.2	限制條件下非線性規劃	136
4.2.1	等式限制條件下非線性規劃	136
4.2.2	不等式限制條件下非線性規劃	138
4.2.3	範例.	140

第 1 章 微積分的基本操作

微積分在科學、經濟學和工程學領域有廣泛的應用，主要用途為解決那些僅依靠代數不能有效解決的問題。此外，微積分更是最佳化理論的基礎。微分是導數的計算，它使得函數、速度、加速度和曲線的斜率均能以一套通用的符號進行邏輯上的演繹或計算。積分則為定義和計算面積、體積等提供一套通用的方法。微積分基本定理指出，微分和積分互為逆運算。藉由 Mathematica 的符號運算，除了能夠有效解決微積分複雜的推導外，更提供精確的數值計算及繪圖環境。以下針對微積分主要四大類分支：極限、微分、積分和級數分別介紹。

1.1 極限

微積分是建立在實數、函數和極限的基礎上。微積分是微分和積分的總稱，『微分』可視為『無限細分』；相對的，『積分』則可視為『無限求和』，而所謂的『無限』就是『極限』。極限的思想是微積分的基礎，Mathematica 計算極限的函數為 `Limit[f, x]`，該指令的用法有以下幾種：

1. `Limit[f, x → a]`: 計算函數 f 在 $x \rightarrow a$ 的極限值。
2. `Limit[f, x → a, Direction → 1]`: 計算函數 f 在 $x \rightarrow a$ 的右極限值。
3. `Limit[f, x → a, Direction → -1]`: 計算函數 f 在 $x \rightarrow a$ 的左極限值。
4. `Limit[f, x → a, Analytic → True]`: 計算函數 f 在 $x \rightarrow a$ 的解析解。

範例 1-1. 極限運算

計算 $\lim_{x \rightarrow 0} \frac{\sin(x)}{x}$

In[1]: `Limit[Sin[x]/x, x->0]`

Out[1]: 1

$$\text{計算 } \lim_{x \rightarrow \infty} \frac{\sin(x)}{x}$$

In[2]: Limit[Sin[x]/x, x->Infinity]

Out[2]: 0

$$\text{計算 } \lim_{x \rightarrow \infty} \sin x$$

In[3]: Limit[Sin[x], x->Infinity]

Out[3]: Interval[[-1, 1]]

$$\text{計算 } \lim_{n \rightarrow \infty} \left(1 + \frac{x}{n}\right)^n$$

In[4]: Limit[(1+x/n)^n, n->Infinity]

Out[4]: e^x

$$\text{計算 } \lim_{n \rightarrow \infty} \left(1 + \frac{f(x)}{n}\right)^n$$

In[5]: Limit[(1+f[x]/n)^n, n->Infinity]

Out[5]: e^{f(x)}

$$\text{計算 } \lim_{x \rightarrow -2^+} \frac{|x + 2|}{x + 2}$$

In[6]: Limit[Abs[x+2]/(x+2), x->-2, Direction->-1]

Out[6]: 1

$$\text{計算 } \lim_{x \rightarrow -2^-} \frac{|x + 2|}{x + 2}$$

In[7]: Limit[Abs[x+2]/(x+2), x->-2, Direction->1]

Out[7]: -1

$$\text{計算 } \lim_{x \rightarrow -2} \frac{|x + 2|}{x + 2}$$

In[8]: Limit[Abs[x+2]/(x+2), x->-2]

Out[8]: 1

由極限的唯一性可知，上式中很明顯發現 Mathematica 在計算極限值時並不會考慮到極限的存在性。所以在函數極限發散或不存在時，使用 Limit 時必須小心。

範例 1-2. 微分四則運算

函數 $f(x)$ 的一階導數

```
In[1]: Limit[(f[x+h]-f[x])/h,h->0,Analytic->True]
```

```
Out[1]: f'[x]
```

函數 $f(x)$ 的二階導數

```
In[2]: Limit[(f'[x+h]-f'[x])/h,h->0,Analytic->True]
```

```
Out[2]: f''[x]
```

微分加法

```
In[3]: Limit[((f[x+h]+g[x+h])-(f[x]+g[x]))/h,h->0,Analytic->True]
```

```
Out[3]: f'[x] + g'[x]
```

微分減法

```
In[4]: Limit[((f[x+h]-g[x+h])-(f[x]-g[x]))/h,h->0,Analytic->True]
```

```
Out[4]: f'[x] - g'[x]
```

微分乘法

```
In[5]: Limit[(f[x+h]*g[x+h]-f[x]*g[x])/h,h->0,Analytic->True]
```

```
Out[5]: g[x]f'[x] + f[x]g'[x]
```

微分除法

```
In[6]: Limit[(f[x+h]/g[x+h]-f[x]/g[x])/h,h->0,Analytic->True]
```

```
Out[6]:  $\frac{g[x]f'[x] - f[x]g'[x]}{g[x]^2}$ 
```

連鎖律

In[7]: `Limit[(f[g[x+h]]-f[g[x]])/h,h->0,Analytic->True]`

Out[7]: $f'[g[x]]g'[x]$

將 $f(x) = \cos(x)$ 及 $g(x) = \sin(x)$ 帶入微分運算法則

In[8]: `{Out[3],Out[4],Out[5],Out[6],Out[7]}/.{f->Cos,g->Sin}`

Out[8]: $\{\cos[x] - \sin[x], -\cos[x] - \sin[x], \cos[x]^2 - \sin[x]^2, \csc[x]^2 (-\cos[x]^2 - \sin[x]^2), -\cos[x]\sin[\sin[x]]\}$

範例 1-3. 微積分基本定理敘述如下：令 $F(x) = \int_a^x f(x)dx$, 則 $F'(x) = f(x)$; 且若 $G'(x) = f(x)$, 則 $\int_a^b f(x)dx = G(b) - G(a)$ 。

定義函數

In[1]: `F[x_]:=Integrate[f[u],{u,a,x}]`

首先我們利用微分的定義，求解微積分第一基本定理

計算 $F'(x)$

In[2]: `Limit[(F[x+h]-F[x])/h,h->0,Analytic->True]`

Out[2]: $f(x)$

接下來我們將面積分割為 n 等份，求解微積分第二基本定理

計算 $\int_a^b f(x)dx$

In[3]: `Sum[G[a+(b-a)/n*i]-G[a+(b-a)/n*(i-1)],{i,1,n}]`

Out[3]: $G(b) - G(a)$

計算 $\int_a^b f(x)dx$

In[4]: `Sum[G[a+(b-a)/n*i]-G[a+(b-a)/n*(i-1)],{i,1,n}]/.G->F`

Out[4]: $\int_a^b f(u)du$

範例 1-4. 計算 $\int_a^b \sin(x) dx$

定義黎曼和

```
In[1]: Riemann[f_,a_,b_,n_]:=Sum[f[a+(b-a)/n*i]*(b-a)/n,{i,1,n}]//Simplify
```

計算 $\sin(x)$ 由 a 至 b 的黎曼和

```
In[2]: Riemann[Sin,a,b,n]
```

```
Out[2]: -(a - b) Csc[(a - b)/2n] Sin[(a - b)/2] Sin[(a(-1 + n) + b(1 + n))/2n]/n
```

計算 $\sin(x)$ 由 0 至 π 的黎曼和

```
In[3]: Riemann[Sin,0,Pi,n]
```

```
Out[3]: πCot[π/2n]/n
```

計算極限值

```
In[4]: Limit[Riemann[Sin,a,b,n],n->Infinity]
```

```
Out[4]: Cos[a] - Cos[b]
```

計算 $\int_0^\pi \sin(x) dx$

```
In[5]: Limit[Riemann[Sin,0,Pi,n],n->Infinity]
```

```
Out[5]: 2
```

1.2 微分

在 Mathematica 中計算函數的微分是非常方便的，其指令為 $D[f,x]$ ，用來表示函數 f 對 x 做或偏微分。該指令的用法有以下幾種：

1. $D[f, x]$: 函數 f 對 x 微分，即 $\frac{\partial f}{\partial x}$ 。
2. $D[f, x_1, x_2, \dots, x_n]$: 函數 f 對 x_1, x_2, \dots 偏微分，即 $\frac{\partial^n f}{\partial x_1 \partial x_2 \dots \partial x_n}$ 。
3. $D[f, \{x, n\}]$: 函數 f 對 x 做 n 次微分，即 $\frac{\partial^n f}{\partial x^n}$ 。
4. $D[f, \{\{x_1, x_2, \dots, x_n\}\}]$: 函數 f 分別對 x_1, x_2, \dots, x_n 做偏微分，即傳回函數 f 的梯度向量， $\left\{ \frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \dots, \frac{\partial f}{\partial x_n} \right\}$ 。

對於單變數函數的微分 Mathematica 有個方便的技巧，就是直接以 $f'[x]$ 表示即可；同理， $f''[x]$ 則表示函數 $f(x)$ 的二階導數，以此類推。

範例 1-5. $f(x) = xe^{x^2}$

定義函數 $f(x) = xe^{x^2}$

In[1]: $f[x_] := x*Exp[x^2]$

傳回一階、二階和三階導數

In[2]: $\{f'[x], f''[x], f'''[x]\}$

Out[2]: $\left\{ e^{x^2} + 2e^{x^2}x^2, 6e^{x^2}x + 4e^{x^2}x^3, 6e^{x^2} + 24e^{x^2}x^2 + 8e^{x^2}x^4 \right\}$

傳回 $x = 1$ 時的一階、二階和三階導數

In[3]: $\{f'[1], f''[1], f'''[1]\}$

Out[3]: {3 e, 10 e, 38 e}

傳回 $g(x)$ 在 $x = 0$ 的 5 階泰勒展開式

In[4]: Plus@@Table[(D[g[x], {x, n}]/.x->0)/n!*x^n, {n, 0, 5}]

Out[4]: $g[0] + x g'[0] + \frac{1}{2}x^2 g''[0] + \frac{1}{6}x^3 g^{(3)}[0] + \frac{1}{24}x^4 g^{(4)}[0] + \frac{1}{120}x^5 g^{(5)}[0]$

若將函數 $f(x) = xe^{x^2}$ 帶入上式，可得 $f(x)$ 在 $x = 0$ 的 5 階泰勒展開式

In[5]: %/.g->f

Out[5]: $x + x^3 + \frac{x^5}{2}$

以上為 Mathematica 對單變數函數微分的方式，多變數函數微分在 Mathematica 中也是非常簡單，僅需要注意微分的順序即可。

範例 1-6. $f(x, y) = xye^{xy}$

定義函數 $f(x, y) = xye^{xy}$

In[1]: f[x_,y_]:=x*y*Exp[x*y]

函數 $f(x, y)$ 對 x 做偏導數，即求 $\frac{\partial f}{\partial x}$

In[2]: D[f[x,y],x]

Out[2]: $e^{xy}xy^2 + e^{xy}y$

函數 $f(x, y)$ 對 x 做二階偏微分， $\frac{\partial^2 f}{\partial x^2}$

In[3]: D[f[x,y],x,x]

Out[3]: $e^{xy}xy^3 + 2e^{xy}y^2$

函數 $f(x, y)$ 對 x 做二階偏微分

In[4]: D[f[x,y],{x,2}]

Out[4]: $e^{xy}xy^3 + 2e^{xy}y^2$

函數 $f(x, y)$ 分別對 x, y 偏微分，傳回梯度

In[5]: {D[f[x,y],x],D[f[x,y],y]}

Out[5]: $\{e^{xy}y + e^{xy}xy^2, e^{xy}x + e^{xy}x^2y\}$

函數 $f(x, y)$ 分別對 x, y 偏微分，傳回梯度

In[6]: D[f[x,y],{{x,y}}]

Out[6]: $\{e^{xy}xy^2 + e^{xy}y, e^{xy}yx^2 + e^{xy}x\}$

以矩陣方式表示梯度

In[7]: %//MatrixForm

$$\text{Out}[7]: \begin{pmatrix} e^{xy}xy^2 + e^{xy}y \\ e^{xy}yx^2 + e^{xy}x \end{pmatrix}$$

上式結果分別再對 x, y 做偏微分即傳回海賽矩陣 (Hessian Matrix)，並以矩陣方式表示

In[8]: D[D[f[x,y],{{x,y}}],{{x,y}}]//MatrixForm

$$\text{Out}[8]: \begin{pmatrix} e^{xy}xy^3 + 2e^{xy}y^2 & e^{xy}x^2y^2 + 3e^{xy}xy + e^{xy} \\ e^{xy}x^2y^2 + 3e^{xy}xy + e^{xy} & e^{xy}yx^3 + 2e^{xy}x^2 \end{pmatrix}$$

運用函數 $D[f,\{x_1, n\}]$ 和 $D[f, \{\{x_1, x_2, \dots, x_n\}\}]$ 的概念，我們可以把程式寫得更簡潔

In[9]: D[f[x,y],{{x,y},2}]//MatrixForm

$$\text{Out}[9]: \begin{pmatrix} e^{xy}xy^3 + 2e^{xy}y^2 & e^{xy}x^2y^2 + 3e^{xy}xy + e^{xy} \\ e^{xy}x^2y^2 + 3e^{xy}xy + e^{xy} & e^{xy}yx^3 + 2e^{xy}x^2 \end{pmatrix}$$

計算海賽矩陣個階組行列式值

In[10]: Det[D[f[x,y],{{x,y},2}][[1;#1;1;#2]]]&@{1,2}//Simplify

$$\text{Out}[10]: \{e^{xy}y^2(2+xy), -e^{2xy}(1+6xy+7x^2y^2+2x^3y^3)\}$$

產生 $g(x, y)$ 的多變數麥克勞尼展開式前 3 階各項元素

In[11]: Table[Binomial[n,i]*(D[g[x,y],{x,i},{y,n-i}]/.{x->0,y->0})*
x^i*y^(n-i)/n!, {n,0,3}, {i,0,n}]//TableForm

Out[11]: g[0, 0]

$$\begin{array}{cccc} yg^{(0,1)}[0,0] & xg^{(1,0)}[0,0] & & \\ \frac{1}{2}y^2g^{(0,2)}[0,0] & xyg^{(1,1)}[0,0] & \frac{1}{2}x^2g^{(2,0)}[0,0] & \\ \frac{1}{6}y^3g^{(0,3)}[0,0] & \frac{1}{2}xy^2g^{(1,2)}[0,0] & \frac{1}{2}x^2yg^{(2,1)}[0,0] & \frac{1}{6}x^3g^{(3,0)}[0,0] \end{array}$$

將 $f(x, y)$ 帶入上式，即可求得 $f(x, y)$ 在 $(x, y) = (0, 0)$ 的 3 階泰勒展開式

In[12]: Plus@@Flatten[%/.g->f]

Out[12]: xy

以上我們可以發現 Mathematica 在計算微分的方式有許多種類，例如：Derivative、D 及 ∂ 。然而，Mathematica 在輸出皆以 Derivative 方式輸出。若想以傳統微積分書寫方式 $\frac{\partial f(x,y)}{\partial x \partial y}$ 輸出，則需透過 Defer 及 TraditionalForm 修改輸出方式。

範例 1-7. $f(x,y) = xy e^{xy}$

函數 $f(x,y)$ 對 x 做偏微分

In[1]: D[f[x,y],x]

Out[1]: $f^{(1,0)}[x,y]$

查看上式輸出語法

In[2]: %//InputForm

Out[2]: Derivative[1, 0][f][x, y]

以 Defer 配合 TraditionalForm 輸出

In[3]: Defer@D[f[x,y],x]//TraditionalForm

Out[3]: $\frac{\partial f(x,y)}{\partial x}$

函數 $f(x,y)$ 對 x 及 y 做 3 階及 2 階微分

In[4]: Defer@D[f[x,y],{x,3},{y,2}]//TraditionalForm

Out[4]: $\frac{\partial^5 f(x,y)}{\partial x^3 \partial y^2}$

函數 $xyf(x,y)$ 對 x 及 y 做 3 階及 2 階微分

In[5]: Defer@D[x*y*f[x,y],{x,3},{y,2}]//TraditionalForm

Out[5]: $\frac{\partial^5 (xyf(x,y))}{\partial x^3 \partial y^2}$

由以上輸出可以發現 Mathematica 在單一函數時可以利用 Defer 及 TraditionalForm 正確輸出成傳統書寫方式，但是在混合函數時則會無法正確輸出。

傳回 Mathematica 預設的微分格式

In[6]: D[x*y*f[x,y],{x,3},{y,2}]

Out[6]: $6f^{(2,1)}[x,y] + 3yf^{(2,2)}[x,y] + 2xf^{(3,1)}[x,y] + xyf^{(3,2)}[x,y]$

因此若要將一般混合函數的微分以傳統書寫方式輸出，則我們必須將上述的 `Derivative` 函數修改以 `Defer` 及 `TraditionalForm`。

將 `Derivative` 函數轉成以 `Defer` 配合 `TraditionalForm` 輸出

```
In[7]: pdconv[f_]:=TraditionalForm[f/.Derivative[index___][wz_][vars___]:>
Apply[Defer@D[wz[vars],##]&,Transpose[{{vars},{index}}]]]
```

函數 $xyf(x,y)$ 對 x 及 y 做 3 階及 2 階微分

```
In[8]: pdconv@D[x*y*f[x,y],{x,3},{y,2}]
```

$$\text{Out}[8]: xy \frac{\partial^5 f(x,y)}{\partial x^3 \partial y^2} + 2x \frac{\partial^4 f(x,y)}{\partial x^3 \partial y^1} + 3y \frac{\partial^4 f(x,y)}{\partial x^2 \partial y^2} + 6 \frac{\partial^3 f(x,y)}{\partial x^2 \partial y^1}$$

觀察上式輸出的結果大致上符合我們書寫方式，但在一次偏微分的寫法， $\frac{\partial^3 f(x,y)}{\partial x^2 \partial y^1}$ ，仍有些許出入。因此我們僅需將 0 次或 1 次偏微分的次數數字取消即可。

將 0 次或 1 次偏微分的次數數字取消

```
In[9]: pdconv[f_]:=TraditionalForm[f/.Derivative[index___][wz_][vars___]:>
Apply[Defer@D[wz[vars],##]&,Transpose[{{vars},{index}}]]/.
{{x_,0}:>Sequence[],{x_,1}:>{x}}]]
```

函數 $xyf(x,y)$ 對 x 及 y 做 3 階及 2 階微分

```
In[10]: pdconv@D[x*y*f[x,y],{x,3},{y,2}]
```

$$\text{Out}[10]: xy \frac{\partial^5 f(x,y)}{\partial x^3 \partial y^2} + 2x \frac{\partial^4 f(x,y)}{\partial x^3 \partial y} + 3y \frac{\partial^4 f(x,y)}{\partial x^2 \partial y^2} + 6 \frac{\partial^3 f(x,y)}{\partial x^2 \partial y}$$

驗證 Leibniz integral rule，計算 $\frac{d}{dx} \int_{u(x)}^{v(x)} \frac{\partial f(x,y)}{\partial x} dy$

```
In[11]: pdconv@D[Integrate[f[x,y],{y,u[x],v[x]}],x]
```

$$\text{Out}[11]: \int_{u(x)}^{v(x)} \frac{\partial f(x,y)}{\partial x} dy - \frac{\partial u(x)}{\partial x} f(x, u(x)) + \frac{\partial v(x)}{\partial x} f(x, v(x))$$

最後必須注意的是，經由 `Defer` 轉換後輸出的結果並無法計算，主要的原因為 `Defer` 在函數未定義時會傳回未計算的結果。因此，若要得到計算的結果則必須以 `Evaluate` 將 `Defer` 取代。

定義函數 $f(x,y) = xy e^{xy}$

```
In[12]: f[x_,y_]:=x*y*Exp[x*y]
```

傳回 Out[10] 計算的結果

In[13]: %10

Out[13]: $6\partial_{\{x,2\},\{y\}}f[x,y] + 3y\partial_{\{x,2\},\{y,2\}}f[x,y] + 2x\partial_{\{x,3\},\{y\}}f[x,y] + xy\partial_{\{x,3\},\{y,2\}}f[x,y]$

傳回 Out[10] 的第一項，可發現因為 Defer 保留未計算結果，所以並未將 $f(x,y)$ 代入計算

In[14]: %10[[1]]

Out[14]: $6*\text{Defer}[D[f[x,y], \{x, 2\}, \{y\}]]$

將 Defer 以函數 Evaluate 取代，傳回 Out[10] 計算的結果

In[15]: %10/.Defer->Evaluate//Simplify

Out[15]: $e^{xy}y(36 + 108xy + 74x^2y^2 + 16x^3y^3 + x^4y^4)$

1.3 全微分

在上一小節中，我們介紹如何用 Mathematica 來求解 $y'(x)$ ，然而並非所有的 $y(x)$ 都可以表示成 x 的封閉解 (close-form)。此時，我們稱 $f(x,y) = 0$ 為 x 與 y 的隱函數關係。相對的， $y = f(x)$ 稱為顯函數。當 y 無法表示成 x 的封閉解時，要求解 $\frac{dy}{dx}$ 則必須倚賴隱函數微分。Mathematica 中要進行隱含數微分則必須借助 Dt 這個全微分指令。全微分指令的用法有以下幾種：

1. Dt[f, x]: 函數 f 對 x 做全微分。
2. Dt[f]: 函數 f 對做全微分。
3. Dt[f, {x,n}]: 函數 f 對 x 做 n 階全微分。
4. Dt[f, {{x1, x2, ..., xn}}]: 函數 f 分別對 x_1, x_2, \dots, x_n 做全微分。

範例 1-8. 已知 $4x^2 + 2xy - xy^3 = 0$

函數 $f(x, y)$ 做全微分

In[1]: $Dt[f[x, y]]$

Out[1]: $Dt[y]f^{(0,1)}[x, y] + Dt[x]f^{(1,0)}[x, y]$

函數 $f(x, y)$ 對 x 做全微分

In[2]: $Dt[f[x, y], x]$

Out[2]: $Dt[y, x]f^{(0,1)}[x, y] + f^{(1,0)}[x, y]$

上式中 $Dt[x]$ 和 $Dt[y]$ 分別表示 dx 和 dy , $Dt[y, x]$ 則表示 $\frac{dy}{dx}$ 或 $y'(x)$; 接下來我們利用 `Solve` 來驗證隱函數微分法及隱函數定理所得到的公式。

隱函數微分法所求得之公式

In[3]: $Dt[y, x] /. Solve[Dt[f[x, y], x] == 0, Dt[y, x]][[1]]$

Out[3]: $-\frac{f^{(1,0)}[x, y]}{f^{(0,1)}[x, y]}$

隱函數定理所求得之公式

In[4]: $Dt[y] / Dt[x] /. Solve[Dt[f[x, y]] == 0, Dt[y]][[1]]$

Out[4]: $-\frac{f^{(1,0)}[x, y]}{f^{(0,1)}[x, y]}$

定義函數 $f(x, y)$

In[5]: $f[x_, y_] := 4x^2 + 2x*y - x*y^3$

函數 $f(x, y)$ 對 x 做全微分

In[6]: $Dt[f[x, y], x]$

Out[6]: $8x + 2y - y^3 + 2xDt[y, x] - 3xy^2Dt[x, y]$

利用隱函數微分法求解

In[7]: $Dt[y, x] /. Solve[Dt[f[x, y], x] == 0, Dt[y, x]][[1]]$

Out[7]: $\frac{8x + 2y - y^3}{x(-2 + 3y^2)}$

函數 $f(x, y)$ 做全微分

In[8]: $Dt[f[x, y]]$

Out[8]: $8x Dt[x] + 2y Dt[y] - y^3 Dt[x] + 2x Dt[y] - 3xy^2 Dt[y]$

利用隱函數定理求解 $\frac{\partial y}{\partial x}$

In[9]: $Dt[y]/Dt[x] /. Solve[Dt[g[x, y]] == 0, Dt[y]][[1]] /. g \rightarrow f$

Out[9]: $-\frac{8x + 2y - y^3}{2x - 3xy^2}$

在Mathematica 中， $D[f,x]$ 所有的變數都假設與 x 無關，因此 $D[f,x]$ 即表示 $\frac{\partial f}{\partial x}$ 。相對的， $Dt[f,x]$ 所有的變數都假設與 x 有關，所以在變數超過三個時，函數 Dt 可以搭配參數 `Constants` 指定特定常數變數。

範例 1-9. 已知 $z^4 + x^2z^3 + y^2 + xy = 0$

函數 $f(x, y, z)$ 做全微分

In[1]: $Dt[f[x, y, z]]$

Out[1]: $Dt[z]f^{(0,0,1)}[x, y, z] + Dt[y]f^{(0,1,0)}[x, y, z] + Dt[x]f^{(1,0,0)}[x, y, z]$

指定變數 y 為常數，利用隱函數定理求解 $\frac{\partial z}{\partial x}$

In[2]: $Dt[z, x, Constants \rightarrow \{y\}] /. Solve[Dt[f[x, y, z], x, Constants \rightarrow \{y\}] == 0,$

$Dt[z, x, Constants \rightarrow \{y\}][[1]]$

Out[2]: $-\frac{f^{(1,0,0)}[x, y, z]}{f^{(0,0,1)}[x, y, z]}$

搭配 `TagSet` 指定變數 y 與 x 無關，利用隱函數定理求解 $\frac{\partial z}{\partial x}$

In[3]: $y /: Dt[y, x] = 0; Dt[z, x] /. Solve[Dt[f[x, y, z], x] == 0, Dt[z, x]][[1]]$

Out[3]: $-\frac{f^{(1,0,0)}[x, y, z]}{f^{(0,0,1)}[x, y, z]}$

以 `Block` 語法撰寫成函數 `Partial`

```
In[4]: partial[var1_, var2_] := Block[{v1 = var1, v2 = var2},
  constant = Complement[{x, y, z}, {v1, v2}];
  If[TrueQ[v1 == v2], 1, Solve[Dt[f[x, y, z], v2, Constants \rightarrow constant] == 0,
    Dt[v1, v2, Constants \rightarrow constant]][[1, -1, -1]]]]
```

計算 $\frac{\partial z}{\partial x}, \frac{\partial z}{\partial y}$ 及 $\frac{\partial z}{\partial z}$

In[5]: `partial[z,#]&/@{x,y,z}`

Out[5]: $\left\{ -\frac{f^{(1,0,0)}[x,y,z]}{f^{(0,0,1)}[x,y,z]}, -\frac{f^{(0,1,0)}[x,y,z]}{f^{(0,0,1)}[x,y,z]}, 1 \right\}$

定義函數以傳統書寫方式輸出

```
In[6]: pdconv[f_]:=TraditionalForm[f/.Derivative[index___][wz_][vars___]:>
  Apply[Defer@D[wz[vars],##]&,Transpose[{vars}, {index}]]/>
  {{x_,0}:>Sequence[], {x_,1}:>{x}}]
```

計算 $\frac{\partial z}{\partial x}, \frac{\partial z}{\partial y}$ 及 $\frac{\partial z}{\partial z}$

In[7]: `partial[z,#]&/@{x,y,z}//pdconv`

Out[7]: $\left\{ -\frac{\frac{\partial f(x,y,z)}{\partial x}}{\frac{\partial f(x,y,z)}{\partial z}}, -\frac{\frac{\partial f(x,y,z)}{\partial y}}{\frac{\partial f(x,y,z)}{\partial z}}, 1 \right\}$

定義函數 $f(x, y, z)$

In[8]: `f[x_,y_,z_]:=x*y+y^2+x^2*z^3+z^4`

求解 $\frac{\partial z}{\partial x}, \frac{\partial z}{\partial y}$ 及 $\frac{\partial z}{\partial z}$

In[9]: `partial[z,#]&/@{x,y,z}`

Out[9]: $\left\{ \frac{-y-2xz^3}{z^2(3x^2+4z)}, \frac{-x-2y}{z^2(3x^2+4z)}, 1 \right\}$

範例 1-10. $x = r \cos(t), y = r \sin(t)$ 及 $f(r, t) = \sqrt{r} \sin\left(\frac{\sqrt{2}t^2}{3}\right)$, 求解 $\frac{df}{dx}$ 和 $\frac{df}{dy}$ 。

定義 x 、 y 及 $f(r, t)$

In[1]: `x=r*Cos[t];y=r*Sin[t];f=Sqrt[r]*Sin[Sqrt[2]*t^2/3];`

對 x 和 y 做全微分

In[2]: `Dt[{x,y}]`

Out[2]: $\{ \text{Cos}[t] \text{Dt}[r] - r \text{Dt}[t] \text{Sin}[t], r \text{Cos}[t] \text{Dt}[t] + \text{Dt}[r] \text{Sin}[t] \}$

由上式可知， dx 和 dy 可以下列二元一次聯立方程組表示：

$$\begin{bmatrix} dx \\ dy \end{bmatrix} = \begin{bmatrix} \cos(t) & -r \sin(t) \\ \sin(t) & r \cos(t) \end{bmatrix} \begin{bmatrix} dr \\ dt \end{bmatrix}$$

求解 dr 和 dt

In[3]: $\{dr, dt\} = \{dr, dt\} /. Solve[\{dx, dy\} == (Dt[\{x, y\}] /. \{Dt[r] \rightarrow dr, Dt[t] \rightarrow dt\}), \{dr, dt\}] [[1]] // Simplify$

Out[3]: $\left\{ dx \cos[t] + dy \sin[t], \frac{dy \cos[t] - dx \sin[t]}{r} \right\}$

對 f 做全微分，並分別將 dr 和 dt 代入

In[4]: $df = Dt[f] /. \{Dt[r] \rightarrow dr, Dt[t] \rightarrow dt\}$

Out[4]: $\frac{2\sqrt{2}t \cos\left[\frac{\sqrt{2}t^2}{3}\right] (dy \cos[t] - dx \sin[t])}{3\sqrt{r}} + \frac{(dx \cos[t] + dy \sin[t]) \sin\left[\frac{\sqrt{2}t^2}{3}\right]}{2\sqrt{r}}$

由於 df 很清楚可以看出為 dx 和 dy 的函數，因此我們可以很輕易的求出 $\frac{df}{dx}$ 和 $\frac{df}{dy}$ 。

根據定義求解 $\frac{df}{dx}$

In[5]: $df /. \{dx \rightarrow 1, dy \rightarrow 0\}$

Out[5]: $-\frac{2\sqrt{2}t \cos\left[\frac{\sqrt{2}t^2}{3}\right] \sin[t]}{3\sqrt{r}} + \frac{\cos[t] \sin\left[\frac{\sqrt{2}t^2}{3}\right]}{2\sqrt{r}}$

根據定義求解 $\frac{df}{dy}$

In[6]: $df /. \{dx \rightarrow 0, dy \rightarrow 1\}$

Out[6]: $\frac{2\sqrt{2}t \cos[t] \cos\left[\frac{\sqrt{2}t^2}{3}\right]}{3\sqrt{r}} + \frac{\sin[t] \sin\left[\frac{\sqrt{2}t^2}{3}\right]}{2\sqrt{r}}$

使用微分求解 $\frac{df}{dx}$

In[7]: $D[df, dx]$

Out[7]: $-\frac{2\sqrt{2}t \cos\left[\frac{\sqrt{2}t^2}{3}\right] \sin[t]}{3\sqrt{r}} + \frac{\cos[t] \sin\left[\frac{\sqrt{2}t^2}{3}\right]}{2\sqrt{r}}$

根據定義求解 $\frac{df}{dy}$

In[8]: $D[df, dy]$

$$\text{Out[8]: } \frac{2\sqrt{2}t \cos[t] \cos\left[\frac{\sqrt{2}t^2}{3}\right]}{3\sqrt{r}} + \frac{\sin[t] \sin\left[\frac{\sqrt{2}t^2}{3}\right]}{2\sqrt{r}}$$

1.4 積分

在 Mathematica 中計算函數的積分指令為 `Integrate`, 該指令的用法有以下幾種:

1. `Integrate[f, x]`: 傳回函數 f 的不定積分, 即 $\int f(x)dx$ 。
2. `Integrate[f, {x, a, b}]`: 傳回函數 f 在區間 (a, b) 間的定積分值。
3. `Integrate[f, {x1, a, b}, {x2, c, d}, ...]`: 傳回函數 f 在區間 $(a, b) \times (c, d) \times \dots$ 的定積分值。要注意的是 x_1 為最外面的積分變數, x_2 為倒數第二個的積分變數, 以此類推, 即

$$\int_a^b \int_c^d \cdots \int f(x_1, x_2, \dots, x_n) dx_n \cdots dx_2 dx_1.$$

範例 1-11. 考慮函數 $\frac{1}{x^p}$

計算 $\frac{1}{x^p}$ 的不定積分

`In[1]: Integrate[1/x^p, x]`

$$\text{Out[1]: } \frac{x^{1-p}}{1-p}$$

由於當 $p \geq 1$ 時積分會發散, 所以 Mathematica 會給於提示

`In[2]: Integrate[1/x^p, {x, 0, 1}]`

$$\text{Out[2]: } \text{ConditionalExpression}\left[\frac{1}{1-p}, \text{Re}[p] < 1\right]$$

如果要避免發生錯誤, 可以在指令後面加上一個假設參數, `Assumptions -> {0 < p < 1}`

`In[3]: Integrate[1/x^p, {x, 0, 1}, Assumptions -> {0 < p < 1}]`

$$\text{Out[3]: } \frac{1}{1-p}$$

另外一個方式為設定參數 `GenerateConditions` 為 `False`, 要求 Mathematica 不產生條件判斷式

```
In[4]: Integrate[1/x^p, {x, 0, 1}, GenerateConditions->False]
```

```
Out[4]: 1/(1 - p)
```

然而在一連串的計算中, 若每次計算都加入假設參數, 實在是造成很多不方便。如果希望 Mathematica 在未來的計算過程中都不要自動產生判斷式, 有以下兩種方式達成: 一是藉由改變 `Integrate` 函數 `GenerateConditions` 的屬性; 二則是將 $p < 1$ 放置在 Mathematica 的 `$Assumptions` 中。

要求 Mathematica 在未來所有的積分計算過程都不產生條件判斷式

```
In[5]: SetOptions[Integrate, GenerateConditions->False]
```

不需要再作任何假設, 也會產生正確的積分值

```
In[6]: Integrate[1/x^p, {x, 0, 1}]
```

```
Out[6]: 1/(1 - p)
```

將 $0 < p < 1$ 放置在 Mathematica 的 `$Assumptions` 中

```
In[7]: $Assumptions={0<p<1};
```

不需要再作任何假設, 也會產生正確的積分值

```
In[8]: Integrate[1/x^p, {x, 0, 1}]
```

```
Out[8]: 1/(1 - p)
```

將屬性及假設取消

```
In[9]: SetOptions[Integrate, GenerateConditions->True];
```

```
$Assumptions=.;
```

重做一次積分, 由於屬性及假設取消, 所以 Mathematica 會給於提示

```
In[10]: Integrate[1/x^p, {x, 0, 1}]
```

```
Out[10]: ConditionalExpression[1/(1 - p), Re[p] < 1]
```

最後，假設當 $p = 1$ 時，考慮以下定積分

$$\int_{-2}^3 \frac{1}{x} dx$$

由於函數 $\frac{1}{x}$ 在零點有奇異點，故 Mathematica 進行積分時會傳回錯誤訊息。

計算 $\int_{-2}^3 \frac{1}{x} dx$

In[11]: Integrate[1/x, {x, -2, 3}]

Out[11]: Integrate::idiv: Integral of $\frac{1}{x}$ does not converge on $\{-2, 3\}$. »

事實上此一定積分是存在的，因為

$$\begin{aligned} \int_{-2}^0 \frac{1}{x} dx + \int_0^3 \frac{1}{x} dx &= \lim_{t \rightarrow 0^+} \left[\int_{-2}^{-t} \frac{1}{x} dx + \int_t^3 \frac{1}{x} dx \right] \\ &= \lim_{t \rightarrow 0^+} \left[|\ln x|_{-2}^{-t} + |\ln x|_t^3 \right] \\ &= \lim_{t \rightarrow 0^+} [\ln t - \ln 2 + \ln 3 - \ln t] \\ &= \ln \frac{3}{2} \end{aligned}$$

故在進行此類瑕積分時可以要求 Mathematica 在計算時考慮到柯西主值（Cauchy principal Value）的存在性。

考慮柯西主值，計算 $\int_{-2}^3 \frac{1}{x} dx$

In[12]: Integrate[1/x, {x, -2, 3}, PrincipalValue -> True]

Out[12]: Log[3/2]

範例 1-12. 假設 $f(x, y) = xy e^{x^2+y^2}$

定義函數 $f(x, y)$

In[1]: f[x_, y_] := x*y*Exp[x^2+y^2]

$f(x, y)$ 對 x 做積分

In[2]: Integrate[f[x, y], x]

Out[2]: $\frac{1}{2} e^{x^2+y^2} y$

$f(x, y)$ 對 x 做積分，其中 $x \in (a, b)$

In[3]: Integrate[f[x, y], {x, a, b}]

$$\text{Out[3]: } \frac{1}{2} e^{y^2} \left(-e^{a^2} + e^{b^2} \right) y$$

$f(x, y)$ 先對 x 做積分，之後再對 y 做積分，其中 $x \in (a, b)$ 及 $y \in (c, d)$

In[4]: Integrate[f[x, y], {y, c, d}, {x, a, b}]

$$\text{Out[4]: } \frac{1}{4} \left(e^{a^2} - e^{b^2} \right) \left(e^{c^2} - e^{d^2} \right)$$

$f(x, y)$ 對 x 做積分，其中 $x \in (0, 1)$ 及 $y \in (0, 1)$

In[5]: Integrate[f[x, y], {y, 0, 1}, {x, 0, 1}]

$$\text{Out[5]: } \frac{1}{4} (-1 + e)^2$$

計算 $\int_0^1 \int_0^y f(x, y) dx dy$

In[6]: Integrate[f[x, y], {y, 0, 1}, {x, 0, y}]

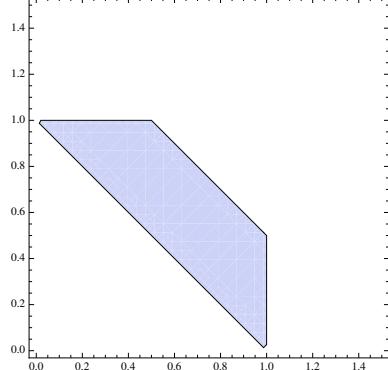
$$\text{Out[6]: } \frac{1}{8} (-1 + e)^2$$

範例 1-13. $\iint_{\Omega} 24xy dy dx$, 其中 $\Omega = \{(x, y) | 0 \leq x \leq 1, 0 \leq y \leq 1, 1 \leq x + y \leq \frac{3}{2}\}$

繪出 Ω 的平面圖

In[1]: RegionPlot[{0 <= x <= 1 && 0 <= y <= 1 && 1 <= x + y <= 3/2}, {x, 0, 3/2}, {y, 0, 1.5}]

Out[1]:



使用布爾函數來計算積分值

```
In[2]: Integrate[24*x*y*Boole[0<=x<=1&&0<=y<=1&&1<=x+y<=3/2],{x,0,1},{y,0,1}]
```

```
Out[2]: 47/16
```

上述中 Boole 為一指標函數，若 (x, y) 滿足 $0 \leq x \leq 1$ 、 $0 \leq y \leq 1$ 且 $1 \leq x + y \leq \frac{3}{2}$ 則傳回 1，否則傳為 0。因此，當積分範圍複雜時，搭配布爾函數則可不考慮複雜的積分順序。在此例中，若不使用布爾函數，則積分範圍必須拆成兩部分分開計算。

不使用布爾函數來計算積分值

```
In[3]: Integrate[24*x*y,{x,0,1/2},{y,1-x,1}]+  
Integrate[24*x*y,{x,1/2,1},{y,1-x,3/2-x}]
```

```
Out[3]: 47/16
```

範例 1-14. 計算半徑為 r 的圓週長

假設圓心為 $(0, 0)$ ，則該圓的方程式為 $x^2 + y^2 = r^2$ ，其中第一象限的圓方程式可改寫為 $y = \sqrt{r^2 - x^2}$ 。

定義函數 $f(x)$

```
In[1]: f[x_]:=y/.Solve[x^2+y^2==r^2,y][[2]]
```

```
Out[1]: \sqrt{r^2 - x^2}
```

假設函數座標為 $(x, f(x))$ ，當 x 的變動量為 Δx 時，則函數的新座標為 $(x + \Delta x, f(x + \Delta x))$ 。若函數 $f(x)$ 可微分，則 $f(x + \Delta x) \approx f(x) + f'(x)\Delta x$ ，故可求得點 $(x, f(x))$ 的移動距離。

計算 x 的變動量為 $Dt[x]$ 時，函數座標在點 $(x, f(x))$ 的移動距離

```
In[2]: EuclideanDistance[{x,f[x]},{x,f[x]]+Dt[{x,f[x]}]]
```

```
Out[2]: \sqrt{Abs[Dt[x]]^2 + 1/4 Abs[(2rDt[r] - 2xDt[x])/(r^2 - x^2)]^2}
```

上式輸出中包含有 Abs 的原因為 Mathematica 無法判斷變數平方後是否為正數所導致。此外，變數 r 為一固定常數，故 $Dt[r]$ 實際上為 0。為方便起見，我們建立一個函數將非 x 的其他變數移除。

建立函數 g 將非 x 的變數移除

```
In[3]: g[y_]:=If[y==x,1,0];
```

建立弧長公式

```
In[4]: arc[x_]:=EuclideanDistance[{x,f[x]},{x,f[x]+Dt[{x,f[x]}]]}/.
{Dt->g,Abs->Sequence}
```

計算座標 $(x, f(x))$ 在 x 的變動量為 Δx 時的移動距離

```
In[5]: arc[x]*\[CapitalDelta]
```

$$\text{Out}[5]: \sqrt{1 + \frac{x^2}{r^2 - x^2}} \Delta$$

計算圓在第一象限的週長

```
In[6]: Integrate[arc[x], {x, 0, r}, Assumptions->{r>0}]
```

$$\text{Out}[6]: \frac{\pi r}{2}$$

由以上輸出即可輕易求得半徑為 r 的圓周長為 $\frac{\pi r}{2} \times 4 = 2\pi r$ 。若假設函數為一長軸為 a , 短軸為 b 的橢圓，我們也可利用相同的方法計算橢圓週長。

定義第一象限的橢圓方程式 $f(x)$

```
In[7]: f[x_]:=y/.Solve[x^2/a^2+y^2/b^2==1,y][[2]]
```

$$\text{Out}[7]: \frac{b\sqrt{a^2 - x^2}}{a}$$

計算橢圓的週長

```
In[8]: 4Integrate[arc[x], {x, 0, a}, Assumptions->{a>b>0}]
```

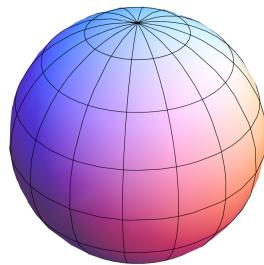
$$\text{Out}[8]: 4a \text{EllipticE}\left[1 - \frac{b^2}{a^2}\right]$$

範例 1-15. 計算半徑為 r 的球體體積

輸出半徑為 1 的球體

```
In[1]: ParametricPlot3D[{Cos[u]Sin[v], Sin[u]Sin[v], Cos[v]},  
{u, 0, 2Pi}, {v, 0, 2Pi}, PlotRange -> {{-1, 1}, {-1, 1}, {-1, 1}},  
Boxed -> False, Axes -> False]
```

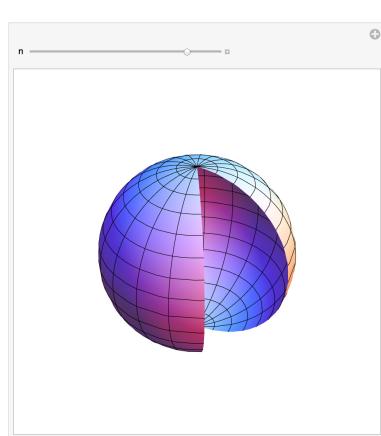
Out[1]:



Shell Method 示意圖

```
In[2]: Manipulate[  
ParametricPlot3D[{Cos[u]Sin[v], Sin[u]Sin[v], Cos[v]},  
{u, 0, n*Pi/20}, {v, 0, Pi}, PlotRange -> {{-1, 1}, {-1, 1}, {-1, 1}},  
Axes -> False, Boxed -> False, SphericalRegion -> True, RotationAction -> "Fit"],  
{n, 1, 40}]
```

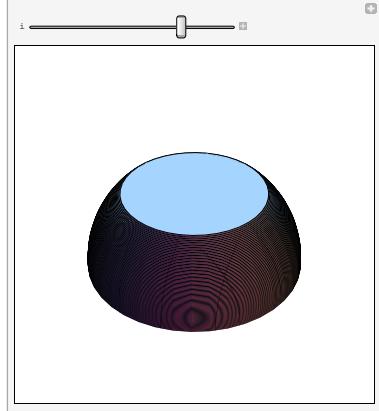
Out[2]:



Disk Method 示意圖

```
In[3]: Manipulate[Show[Graphics3D[
Cylinder[{{0, 0, (#-1)/100}, {0, 0, #/100}}, Sqrt[1-(#/100)^2]]]&/@Range[n],
Axes->False, Boxed->False, SphericalRegion->True, RotationAction->"Fit"],
{n, 1, 100}]
```

Out[3]:



以 Shell Method 計算球體體積

```
In[4]: 2Integrate[Pi*(r^2-x^2),{x,0,r}]
```

$$\text{Out[4]: } \frac{4\pi r^3}{3}$$

以 Disk Method 計算球體體積

```
In[5]: 2Integrate[2Pi*x*sqrt[(r^2-x^2)],{x,0,r}]/.(r^a_)^b_->r^(a*b)
```

$$\text{Out[5]: } \frac{4\pi r^3}{3}$$

使用布爾函數求體積

```
In[6]: Integrate[Evaluate@Boole[x^2+y^2+z^2<=r^2],{z,-r,r},{y,-r,r},
{x,-r,r},Assumptions->r>0]
```

$$\text{Out[6]: } \frac{4\pi r^3}{3}$$

然而並非所有的函數都存在反導數，例如標準常態分配的機率密度函數，

$$f(z) = \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}}, -\infty < z < \infty$$

就無法求得其不定積分，所以在求解此類問題時就必須借助數值積分。數值積分是解決定積分的另一種有效的方法，它可以求出一個近似解。特別是對於用 `Integrate` 指令無法求出的定積分，數值積分更是可以發揮巨大作用。

數值積分與傳統定積分基本概念是一樣的。傳統定積分使用對應的積分公式，而對於一般人來說不容易推導高複雜度的非線性函數定積分公式則必須藉由電腦或計算機來快速計算積分值。數值積分的概念源自於黎曼積分，將積分範圍在一個極小的區域內分割成 $[x_1, x_2, x_3, \dots, x_n]$ ，對於每一個樣本點給予適當的權重 w_i 作為 $\int_a^b f(x) dx$ 的估計值，即

$$\int_a^b f(x) dx \approx \sum_{i=1}^n w_i f(x_i)$$

最後使用電腦或電子計算機程式進行累加積分值。

範例 1-16. 假設 $f(z) = \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}}$

定義函數

In[1]: `f[z_]:=1/Sqrt[2Pi]*Exp[-z^2/2];`

由於 $f(z)$ 為標準常態分配之機率密度函數，我們也可直接以 Mathematica 內建的統計函數傳回標準常態分配的機率密度函數。

計算標準常態分配的機率密度函數

In[2]: `f[z_]:=PDF[NormalDistribution[0,1],z]`

計算 $a < z < b$ 的機率

In[3]: `Integrate[f[z],{z,a,b}]`

Out[3]: $\frac{1}{2} \left(-\text{Erf}\left[\frac{a}{\sqrt{2}}\right] + \text{Erf}\left[\frac{b}{\sqrt{2}}\right] \right)$

上式中 $\text{Erf}(x)$ 為誤差函數， $\text{Erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$ 。由於函數 $f(z)$ 不可積，所以 Mathematica 並沒有真正傳回不定積分。

定義黎曼和

In[4]: `Riemann[g_,a_,b_,n_]:=Sum[g[a+(b-a)/n*i]*(b-a)/n,{i,1,n}]/Simplify`

傳回 a 至 b 的黎曼和

In[5]: `Riemann[f, a, b, n]`

Out[5]: $\sum_{i=1}^n \frac{(-a+b)e^{-\frac{1}{2}\left(a+\frac{(-a+b)i}{n}\right)^2}}{n\sqrt{2\pi}}$

在 -5 至 1.96 劃分成 1000 等份，傳回黎曼和

In[6]: Riemann[f, -5., 1.96, 1000]

Out[6]: 0.975205

在 -5 至 5 劃分成 1000 等份，傳回黎曼和

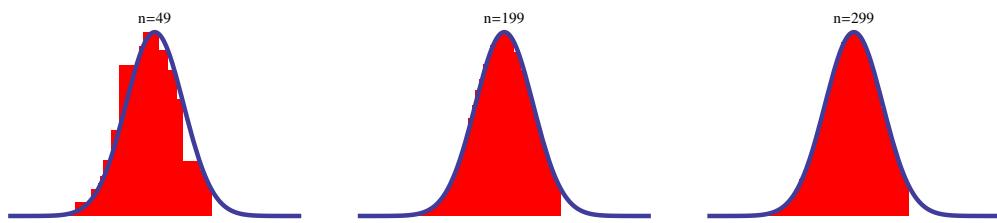
In[7]: Riemann[f, -5., 5., 1000]

Out[7]: 0.999999

分別在 -5 與 1.96 間隨機選取 49, 199, 299 的分割點繪製矩形

```
In[8]: GraphicsGrid[{{pts=RandomReal[{-5,1.96},#]//Sort;
Show[Graphics[{Red,Rectangle[{pts[[#]],0},
{pts[[#+1]],f[pts[[#+1]]]}]}]&@Range[Length[pts]-1],
Plot[f[x],{x,-5,5},PlotStyle->Thickness[0.015]],AspectRatio->0.65,
PlotLabel->"n="<>ToString[#])&/@{49,199,299}],ImageSize->800]}
```

Out[8]:



由上圖我們可以發現，只要分割點取得夠多，那矩形的面積和就會逼近真正的面積。在數值分析上，常用的數值積分的方法有梯形法則 (Trapezoidal rule) 和辛普森法則 (Simpson's rule)，這兩種方法都屬於牛頓-寇次公式 (Newton-Cotes formula)。牛頓-寇次公式的原理是以 $n + 1$ 個等距點進行插值，求得對應函數 $f(x)$ 的 Lagrange 多項式來近似原來的函數，再進行定積分。所以，當 $n = 1$ 時，即為梯形法則； $n = 2$ 時，為辛普森法則； $n = 3$ 時，為辛普森 $3/8$ 法則； $n = 4$ 時，為保爾法則。在 Mathematica 中，以 n 個點建構 $n - 1$ 次方的 Lagrange 多項式的函數為 InterpolatingPolynomial。接下來介紹如何利用 InterpolatingPolynomial 建立牛頓-寇次公式。

範例 1-17. 牛頓-寇次公式相關數值積分法

建立牛頓-寇次公式

```
In[1]: NC[n_,a_,b_]:=NC[n,a,b]=Block[{temp=n,x,f,i,aa=a,bb=b,h=(b-a)/temp},
Integrate[InterpolatingPolynomial[Table[{aa+i*h,f[aa+i*h]}, {i,0,temp}],t],{t,aa,bb}]]
```

梯形法則計算公式

In[2]: NC[1,a,b]

$$\text{Out}[2]: -\frac{1}{2}(a - b)(f[a] + f[b])$$

辛普森法則計算公式

In[3]: NC[2,a,b]

$$\text{Out}[3]: -\frac{1}{6}(a - b) \left(f[a] + f[b] + 4f \left[\frac{a+b}{2} \right] \right)$$

辛普森 3/8 法則計算公式

In[4]: NC[3,a,b]

$$\text{Out}[4]: -\frac{1}{8}(a - b) \left(f[a] + f[b] + 3 \left(f \left[\frac{1}{3}(2a + b) \right] + f \left[\frac{1}{3}(a + 2b) \right] \right) \right)$$

保爾法則計算公式

In[5]: NC[4,a,b]

$$\text{Out}[5]: -\frac{1}{90}(a - b) \left(7f[a] + 7f[b] + 4 \left(3f \left[\frac{a+b}{2} \right] + 8f \left[\frac{1}{4}(3a + b) \right] + 8f \left[\frac{1}{4}(a + 3b) \right] \right) \right)$$

將 -5 與 1.96 均分成 1000 等分

In[6]: pts=Table[-5+(1.96+5)/1000*i,{i,0,1000}];

定義函數 $f(z)$

In[7]: f[z_]:=PDF[NormalDistribution[0,1],z]

梯形法則計算公式

In[8]: fNC[1,a_,b_]:=NC[1,a,b]

$$\text{Out}[8]: -\frac{1}{2}(a - b) \left(\frac{e^{-\frac{a^2}{2}}}{\sqrt{2\pi}} + \frac{e^{-\frac{b^2}{2}}}{\sqrt{2\pi}} \right)$$

辛普森 3/8 法則計算公式

In[9]: fNC[2,a_,b_]:=NC[2,a,b]

$$\text{Out[9]: } -\frac{1}{6}(a-b) \left(2e^{-\frac{1}{8}(a+b)^2} \sqrt{\frac{2}{\pi}} + \frac{e^{-\frac{a^2}{2}}}{\sqrt{2\pi}} + \frac{e^{-\frac{b^2}{2}}}{\sqrt{2\pi}} \right)$$

保爾法則計算公式

In[10]: fNC[4, a_, b_] = NC[4, a, b]

$$\text{Out[10]: } -\frac{1}{90}(a-b) \left(\frac{7e^{-\frac{a^2}{2}}}{\sqrt{2\pi}} + \frac{7e^{-\frac{b^2}{2}}}{\sqrt{2\pi}} + 4 \left(4e^{-\frac{1}{32}(3a+b)^2} \sqrt{\frac{2}{\pi}} + 4e^{-\frac{1}{32}(a+3b)^2} \sqrt{\frac{2}{\pi}} + \frac{3e^{-\frac{1}{8}(a+b)^2}}{\sqrt{2\pi}} \right) \right)$$

分別以梯形法則、辛普森法則及保爾法則計算面積

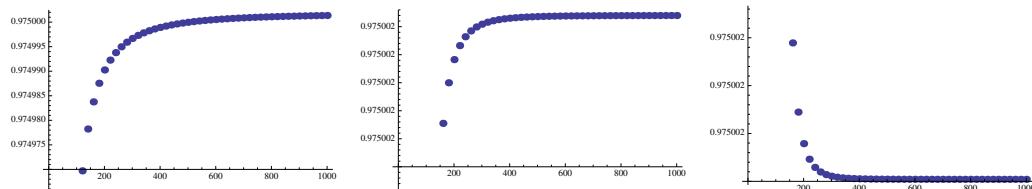
In[11]: Sum[fNC[#, pts[[i]], pts[[i+#]]], {i, 1, 1000, #}] & /@ {1, 2, 4}

Out[11]: {0.975001, 0.975002, 0.975002}

分別將 -5 與 1.96 均分成 $20, 40, \dots, 1000$ 等分再以梯形法則、辛普森法則及保爾法則計算面積。

In[12]: GraphicsGrid[{Table[ListPlot[(pts=Table[-5+(1.96+5)/##, {i, 0, #}], {#, Sum[fNC[k, pts[[i]], pts[[i+k]]], {i, 1, #, k}]}] & /@ Range[20, 1000, 20], PlotStyle -> PointSize[0.025]], {k, {1, 2, 4}}]}, ImageSize -> 1000]

Out[12]:



由輸出圖形發以發現分割越多，越趨近 0.975002 。此外，輸出結果也顯示保爾法則在近似的效果較佳。但必須注意的是，牛頓-寇次公式並非以高階的多項式進行插值近似效果就會越好。事實上，牛頓-寇次公式對於高階的多項式做為近似方式反而會有很較大的誤差，此即為龍格 (Runge) 現象。然而，上述的計算方式雖然可以達到近似的效果，但複雜的計算卻也造成計算的效率不高。為了提高收斂速度和減少計算次數，我們在此介紹一個較常用的計算方式，Romberg 積分法。Romberg 積分法是在以牛頓-寇次公式為基礎所構造出的一種加速計算積分的方法，不同於牛頓-寇次公式取 n 個等距點，Romberg 積分法則改以 2^n 個等距點，它在不增加計算

次數的前提下提高了準確度。有關 Romberg 演算法的計算公式如下：

$$\begin{cases} R(0,0) = \frac{1}{2}(b-a)(f(a)+f(b)) \\ R(n,0) = \frac{1}{2}R(n-1,0) + \frac{b-a}{2^n} \sum_{k=1}^{2^{n-1}} f\left(a + (2k-1)\frac{b-a}{2^n}\right) \\ R(n,m) = R(n,m-1) + \frac{1}{4^m - 1} [R(n,m-1) - R(n-1,m-1)] \end{cases} \quad (1-1)$$

根據 Romberg 演算法，我們可以將上述公式以迴圈的方式撰寫程式如下：

利用 Romberg 演算法求解 $\int_{-5}^{1.96} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx$

```
In[13]: Clear[R,n,m]; f[x_]:=1/Sqrt[2Pi]*Exp[-x^2/2]; a=-5;b=1.96;n=1,error=100;
R[0,0]=((b-a)/2*(f[a]+f[b]))//N;
While[error>10^-11,
  m=1;
  While[m<=n,
    R[n,0]=(R[n-1,0]/2+(b-a)/2^n*Sum[f[a+(2k-1)*(b-a)/2^n],
    {k,1,2^(n-1)}])//N;
    R[n,m]=(R[n,m-1]+(b-a)/(4^m-1)*(R[n,m-1]-R[n-1,m-1]))//N;
    error=Abs[R[n,m]-R[n,m-1]];
    m++];
  n++];
Table[R[i,j],{i,n-6,n-1},{j,n-6,i}]//TableForm
```

```
Out[13]: 0.974865
          0.974962  0.974962
          0.974992  0.974992  0.974992
          0.974999  0.974999  0.974999  0.974999
          0.975001  0.975001  0.975001  0.975001  0.975001
          0.975002  0.975002  0.975002  0.975002  0.975002  0.975002
```

將 Romberg 演算法撰寫成 romberg 函數

```
In[14]: romberg[fun_,min_,max_]:=Block[{err=10,i=1,a=min,b=max,R},
R[0,0]=(b-a)/2*(fun[a]+fun[b]);
R[n_,0]:=R[n,0]=R[n-1,0]/2+(b-a)/2^n*Sum[fun[a+(2k-1)*(b-a)/2^n],
{k,1,2^(n-1)}];
R[n_,m_]:=R[n,m]=R[n,m-1]+(b-a)/(4^m-1)*(R[n,m-1]-R[n-1,m-1]);
While[err>10^(-11),err=Abs[N[R[i,i]-R[i,i-1]]];i++];
R[i,i]]
```

以 romberg 函數計算 $\int_{-5}^{1.96} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx$

In[15]: romberg[f,-5,1.96]

Out[15]: 0.975002

上述所介紹牛頓-寇次公式相關積分近似方法皆以等距分割點，然而數值積分的分割點在理論上可任意指定，

高斯 (Karl F. Gauss) 也發現特別給定的分割點有時能大大改進數值積分之精確度。以下我們介紹如何建立高斯積分公式 (Gauss Quadrature)。

範例 1-18. 高斯積分公式，首先假設取兩個分割點 x_1 和 x_2 ，則我們的目標為

$$\int_{-1}^1 f(x) dx \approx \sum_{i=1}^2 w_i f(x_i) = w_1 f(x_1) + w_2 f(x_2)$$

其中， w_1 和 w_2 為權重係數。由於 w_1 , w_2 , x_1 和 x_2 皆為未知數，若使用待訂係數法，則我們需要假設 $f(x)$ 為一元三次方程式作為求解工具。若取三個分割點時則有六個未知數，故需要假設 $f(x)$ 為一元五次方程式；同理，當在區間 $[a, b]$ 取 n 個分割點時，則有 $2n$ 個未知數，需要假設 $f(x)$ 為一元 $2n - 1$ 次方程式。

建立取代規則

```
In[1]: myrule=a_[i_]:=ToExpression[ToString[a]<>ToString[i]];
```

建立一元 $2n - 1$ 次多項式的係數向量

```
In[2]: coef[n_]:=Table[a[i],{i,0,2n-1}]/.myrule;
```

定義變數

```
In[3]: vars[n_]:=Flatten[{Table[w[i],{i,n}],Table[x[i],{i,n}]}/.myrule];
```

建立一元 $2n - 1$ 次多項式

```
In[4]: poly[n_]:=Total@Table[a[i]x^i,{i,0,2n-1}]/.myrule;
```

計算 $\sum_{i=1}^2 w_i f(x_i) = w_1 f(x_1) + w_2 f(x_2)$

```
In[5]: rhs[n_]:=Total@Table[w[i]*poly[n]/.x->x[i],{i,n}]/.myrule;
```

計算 $\int_{-1}^1 f(x) dx$

```
In[6]: lhs[n_]:=Integrate[poly[n], {x,-1,1}];
```

計算 $n = 2$ 時的 lhs

```
In[7]: Collect[lhs[2],coef[2]]
```

```
Out[7]: 2a0 + 2a2/3
```

計算 $n = 2$ 時的 rhs

In[8]: Collect[rhs[2], coef[2]]

Out[8]: $a_0(w_1 + w_2) + a_1(w_1x_1 + w_2x_2) + a_2(w_1x_1^2 + w_2x_2^2) + a_3(w_1x_1^3 + w_2x_2^3)$

由待訂係數法可知，等號左右兩邊的係數， a_0 , a_1 , a_2 和 a_3 必須相等，故可以函數 Coefficient 取出係數後再以 Solve 求解聯立方程組即可求得 w_1 , w_2 , x_1 和 x_2 。

使用 Solve 求解未知數 w_1 , w_2 , x_1 和 x_2

In[9]: Solve[Coefficient[lhs[2], coef[2]] == Coefficient[rhs[2], coef[2]], vars[2]]

Out[9]: $\left\{ \begin{array}{l} \left\{ w_1 \rightarrow 1, w_2 \rightarrow 1, x_1 \rightarrow \frac{1}{\sqrt{3}}, x_2 \rightarrow -\frac{1}{\sqrt{3}} \right\}, \\ \left\{ w_1 \rightarrow 1, w_2 \rightarrow 1, x_1 \rightarrow -\frac{1}{\sqrt{3}}, x_2 \rightarrow \frac{1}{\sqrt{3}} \right\} \end{array} \right\}$

高斯積分近似公式

In[10]: GaussQuadrature[g_, n_] := SortBy[
Flatten@{vars[n][[1;;n]].Outer[g, vars[n][[n+1;;-1]], vars[n]]}/.
Solve[Coefficient[lhs[n], coef[n]] == Coefficient[rhs[n], coef[n]],
vars[n]], #[[{-n+1;;-1}]] &][[1]]

兩點高斯積分公式

In[11]: GaussQuadrature[f, 2]
Out[11]: $\left\{ f\left[-\frac{1}{\sqrt{3}}\right] + f\left[\frac{1}{\sqrt{3}}\right], 1, 1, \frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}} \right\}$

三點高斯積分公式

In[12]: GaussQuadrature[f, 3]
Out[12]: $\left\{ \frac{8f[0]}{9} + \frac{5}{9}f\left[-\sqrt{\frac{3}{5}}\right] + \frac{5}{9}f\left[\sqrt{\frac{3}{5}}\right], \frac{5}{9}, \frac{8}{9}, \frac{5}{9}, \sqrt{\frac{3}{5}}, 0, -\sqrt{\frac{3}{5}} \right\}$

輸出四點高斯積分公式求解的聯立方程組

```
In[13]: With[{n=4}, Thread[Coefficient[rhs[n], coef[n]]==Coefficient[lhs[n], coef[n]]]]//TableForm]
```

```
Out[13]: w1 + w2 + w3 + w4 == 2
w1x1 + w2x2 + w3x3 + w4x4 == 0
w1x1^2 + w2x2^2 + w3x3^2 + w4x4^2 == 2/3
w1x1^3 + w2x2^3 + w3x3^3 + w4x4^3 == 0
w1x1^4 + w2x2^4 + w3x3^4 + w4x4^4 == 2/5
w1x1^5 + w2x2^5 + w3x3^5 + w4x4^5 == 0
w1x1^6 + w2x2^6 + w3x3^6 + w4x4^6 == 2/7
w1x1^7 + w2x2^7 + w3x3^7 + w4x4^7 == 0
```

從上面的計算可發現，當 n 大時，要以 `Solve` 求解聯立方程式來計算分割點 x_i 和權重 w_i 是非常耗時的。

由於 $f(x)$ 為一 $2n+1$ 次的多項式，假設 $q(x)$ 為一 $n+1$ 次的多項式，且其根為 x_i , $i = 0, 1, \dots, n$ ，若

$$\int_a^b x^k q(x) w(x) dx = 0, k = 1, 2, \dots, n$$

則

$$\int_a^b f(x) w(x) dx = \int_a^b [p(x)q(x) + r(x)] w(x) dx = \int_a^b r(x) w(x) dx \approx \sum_{i=1}^n f(x_i) \int_a^b l_i(x) w(x) dx$$

因此，我們只要能找到 x^k 的正交多項式 $q(x)$ 即可輕易求出 x_i 的值。當 $a = -1$, $b = 1$ 及權重函數 $w(x) = 1$ 時，則 $q(x)$ 即為勒壤得多項式 (Legendre polynomial)，此時權重係數

$$w_i = \frac{2}{(1-x_i^2)[q'_n(x_i)]^2}$$

在 Mathematica 中也有提供一個函數 `LegendreP` 以輸出勒壤得多項式，以下我們比較兩種計算方式的效率上的差異。

使用 `GaussQuadrature` 計算五個分割點及權重

```
In[14]: AbsoluteTiming@TableForm[SortBy[Transpose@Partition[
GaussQuadrature[f, 5][[2;;-1]], 5], Last], TableHeadings->{None, {wi, xi}}]
```

```
Out[14]: {{40.780226, {{0.236927, 0.478629, 0.568889, 0.478629, 0.236927}, {-0.90618, -0.538469, 0., 0.538469, 0.90618}}}}
```

使用 `LegendreP` 計算五個分割點及權重

```
In[15]: AbsoluteTiming@TableForm[{2/((1-x^2)D[LegendreP[5,x],x]^2),x}/.
NSolve[LegendreP[5,x]==0,x], TableHeadings->{None, {xi, wi}}]
```

Out[15]:

$$\left\{ \begin{array}{cc} wi & xi \\ \hline 0.236927 & -0.90618 \\ 0.478629 & -0.538469 \\ 0.004646, & 0. \\ 0.568889 & 0. \\ 0.478629 & 0.538469 \\ 0.236927 & 0.90618 \end{array} \right\}$$

建立高斯勒壞得積分公式

In[16]: GaussLegendre[g_,n_]:=Total[#[[1]]*g#[[2]]]&/@(
 $f2/((1-x^2)D[LegendreP[n,x],x]^2),x}\).Solve[LegendreP[n,x]==0,x)];$

兩點高斯積分公式

In[17]: GaussLegendre[f,2]

Out[17]: $f\left[-\frac{1}{\sqrt{3}}\right] + f\left[\frac{1}{\sqrt{3}}\right]$

三點高斯積分公式

In[18]: GaussLegendre[f,3]

Out[18]: $\frac{8f[0]}{9} + \frac{5}{9}f\left[-\sqrt{\frac{3}{5}}\right] + \frac{5}{9}f\left[\sqrt{\frac{3}{5}}\right]$

輸出計算 20 點高斯積分公式所需時間

In[19]: First@AbsoluteTiming@N@GaussLegendre[f,20]

Out[19]: 0.024461

由於 $\int_a^b f(x)dx$ 可以經由一次轉換寫成

$$\int_a^b f(x)dx = \frac{b-a}{2} \int_{-1}^1 f\left(\frac{b-a}{2}v + \frac{b+a}{2}\right)dv$$

因此，大多數的函數皆可利用高斯積分公式進行計算。

定義標準常態機率密度函數

In[20]: g[v_]=(b-a)/2*PDF[NormalDistribution[0,1],(b-a)*v/2+(b+a)/2]

Out[20]: $\frac{(-a+b)e^{-\frac{1}{2}\left(\frac{a+b}{2}+\frac{1}{2}(-a+b)v\right)^2}}{2\sqrt{2\pi}}$

在不同分割數下計算 $\int_{-5}^{1.96} \frac{e^{-\frac{x^2}{2}}}{\sqrt{2\pi}} dx$

In[21]: GaussLegendre[g, #] /. {a -> -5, b -> 1.96} & /@ Range[2, 10] // Chop

Out[21]: {1.2345, 0.775297, 1.04122, 0.961676, 0.976426, 0.975075, 0.974932, 0.97502, 0.974999}

由上面輸出與前例使用牛頓-寇次公式計算結果相比較，我們可以發現高斯勒壤得積分公式能夠在分割數較少的情況下達到相當高的準確度，此外在計算效率上也有大幅度的提升。當然 Mathematica 中作數值積分不需要自己撰寫程式，Mathematica 有提供自己的數值積分函數。在 Mathematica 中數值積分的函數為 NIntegrate，該函數結合了 N 以及 Integrate 的用途，其用法有以下幾種：

1. NIntegrate[f, {x, a, b}]: 傳回函數 f 的數值積分
2. NIntegrate[f, {x₁, a, b}, {x₂, c, d}, ...]: 傳回函數 f 在區間 (a, b) × (c, d) × ... 的定積分值。要注意的是 x₁ 為最外面的積分變數，x₂ 為倒數第二個的積分變數
3. NIntegrate[f, {x₁, a, b}, {x₂, c, d}, ..., Method -> {Strategy, Method -> Rule, SymbolicProcessing -> k}]: 以特定方式計算函數 f 的數值積分。

此外，Mathematica 也提供了 CartesianRule, ClenshawCurtisRule, GaussKronrodRule, LobattoKronrodRule, MultidimensionalRule, MultipanelRule, NewtonCotesRule 及 TrapezoidalRule 等積分近似公式可供使用，其中，Mathematica 預設積分近似公式為 GaussKronrodRule。最後，Symbolic -Processing 則可用來指定是否先行對積分項做符號運算所允許的最大時間。另一方面，在積分近似的精確度與準確度的考量上，Mathematica 也提供了 GlobalAdaptive, LocalAdaptive, DoubleExponential, MonteCarlo, AdaptiveMonteCarlo, QuasiMonteCarlo 及 AdaptiveQuasiMonteCarlo 等策略，其中 Mathematica 在此預設使用的策略為 GlobalAdaptive。

範例 1-19. 計算 $\int_{-\infty}^{1.96} \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}} dz$

定義函數

In[1]: f[z_] := PDF[NormalDistribution[0, 1], z]

以 Integrate 計算瑕積分 $\int_{-\infty}^{1.96} f[z] dz$ ，並輸出計算時間

In[2]: Integrate[f[z], {z, -Infinity, 1.96}] // AbsoluteTiming

Out[2]: {0.153418, 0.975002}

以 NIntegrate 計算瑕積分 $\int_{-\infty}^{1.96} f[z] dz$, 並輸出計算時間

In[3]: NIntegrate[f[z], {z, -Infinity, 1.96}] // AbsoluteTiming

Out[3]: {0.004712, 0.975002}

將 NIntegrate 以 -5 到 1.96 所取的分割點取出

In[4]: data = SortBy[Reap[NIntegrate[f[z], {z, -5, 1.96}],
EvaluationMonitor :> Sow[{z, f[z]}]]][[2, 1]], First];

輸出分割點點數

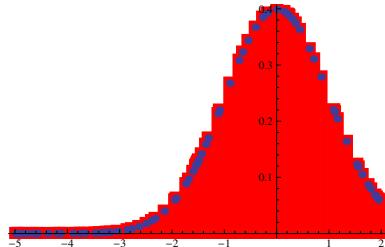
In[5]: Length@data

Out[5]: 77

分割點分佈圖

In[6]: ListPlot[data, PlotStyle -> PointSize[0.02],
Filling -> 0, FillingStyle -> {{Thickness[0.035], Red}}]

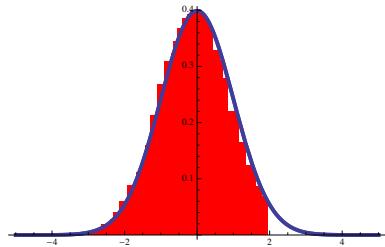
Out[6]:



將分割點以矩形方式連接

```
In[7]: Show[Plot[f[z], {z, -5, 5}, PlotStyle -> Thickness[0.01]],  
Graphics[{Red, Rectangle[{#[[1, 1]], 0}, #[[2, 1]]] & /@ Partition[data, 2, 1]}],  
Plot[f[z], {z, -5, 5}, PlotStyle -> Thickness[0.01]],  
AspectRatio -> 1/GoldenRatio]
```

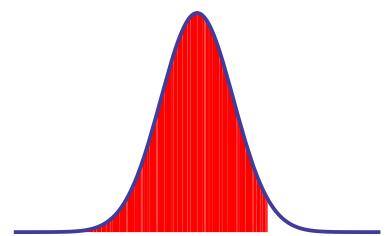
Out[7]:



將分割點以梯形方式連接

```
In[8]: Show[Plot[f[z], {z, -5, 5}, PlotStyle -> Thickness[0.01]],  
Graphics[{Red, Polygon[{{data[[#, 1]], 0}, data[[#]],  
data[[#+1]], {data[[#+1, 1]], 0}}] } & /@ Range[Length@data - 1]],  
Plot[f[z], {z, -5, 5}, PlotStyle -> Thickness[0.01]],  
AspectRatio -> 1/GoldenRatio]
```

Out[8]:



定義策略方式

```
In[9]: mymethod = {"GlobalAdaptive", "LocalAdaptive", "DoubleExponential",  
"MonteCarlo", "AdaptiveMonteCarlo", "QuasiMonteCarlo"}
```

定義積分近似公式

```
In[10]: myrule = {"CartesianRule", "ClenshawCurtisRule", "GaussKronrodRule",  
"LobattoKronrodRule", "NewtonCotesRule"};
```

取得不同的策略方式計算 $f(z)$ 在 $-5 < z < 1.96$ 的樣本點

```
In[11]: data1=Reap[NIntegrate[f[z],{z,-5,1.96},  
EvaluationMonitor:>Sow[{z,f[z]}],Method->#]]&/@mymethod;
```

輸出不同策略方式的計算結果

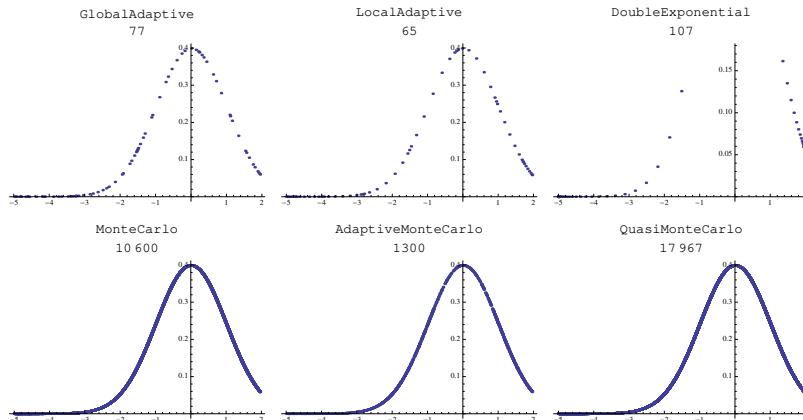
```
In[12]: data1[[All,1]]
```

```
Out[12]: {0.975002, 0.975002, 0.975002, 0.988246, 0.991901, 0.975002}
```

輸出不同策略方式的樣本數及樣本點分佈

```
In[13]: Map[Column[{mymethod[[#]],Length@data1[[#,2,1]],  
ListPlot[data1[[#,2,1]]],Center}&,  
{1,2,3},{4,5,6}],{2}]//GraphicsGrid
```

Out[13]:



取得不同的策略方式計算 $f(x)f(y)$ 在 $0 < x < y < 1.96$ 的樣本點

```
In[14]: data2=Reap[NIntegrate[f[x]*f[y],{y,0,1.96},{x,0,y},  
EvaluationMonitor:>Sow[{x,y}],Method->#]]&/@mymethod;
```

輸出不同策略方式的計算結果

```
In[15]: data2[[All,1]]
```

```
Out[15]: {0.112813, 0.112814, 0.112813, 0.114, 0.114402, 0.112803}
```

在 GlobalAdaptive 策略下，取得不同近似規則計算 $f(x)f(y)$ 在 $0 < x < y < 1.96$ 的樣本點

```
In[16]: data3=Reap[NIntegrate[f[x]*f[y],{y,0,1.96},{x,0,y},
EvaluationMonitor:>Sow[{x,y}],Method->#]]&/@myrule;
```

輸出不同積分近似公式的計算結果

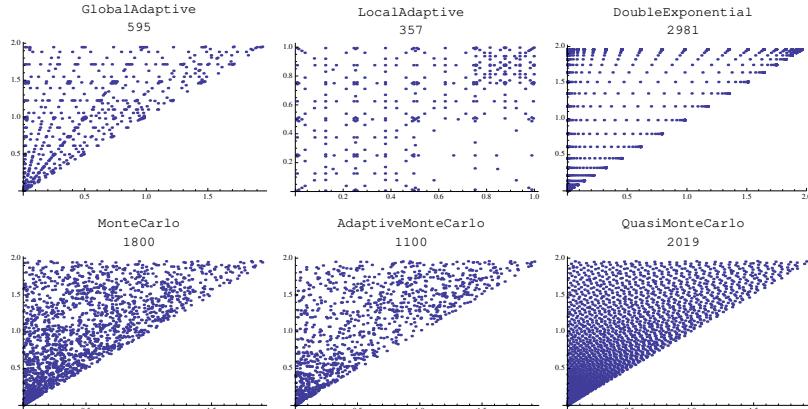
```
In[17]: data3[[All,1]]
```

```
Out[17]: {0.112813, 0.112813, 0.112813, 0.112813, 0.112813}
```

輸出不同策略方式的樣本數及樣本點分佈

```
In[18]: GraphicsGrid[Map[Column[{mymethod[[#]],Length@data2[[#,2,1]],
ListPlot[data2[[#,2,1]]]},Center]&,{1,2,3},{4,5,6},{2}]]
```

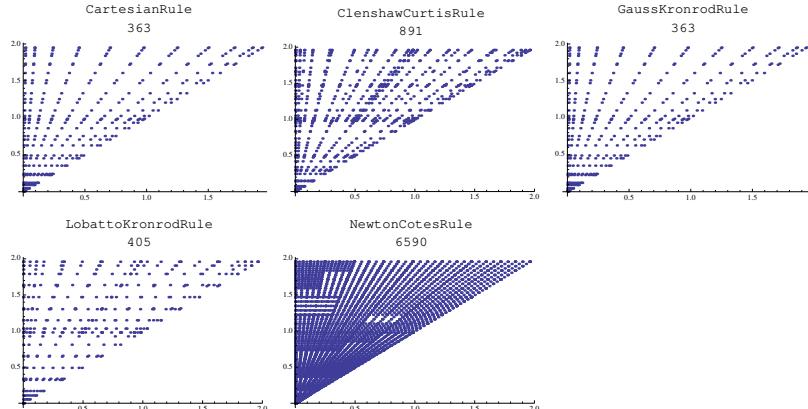
Out[18]:



輸出不同積分近似公式的樣本數及樣本點分佈

```
In[19]: Map[Column[{myrule[[#]],Length@data3[[#,2,1]],
ListPlot[data3[[#,2,1]]]},Center]&,{1,2,3},{4,5},{2}]//GraphicsGrid
```

Out[19]:



由以上的結果可以發現，MonteCarlo 的策略方式在計算上較慢且無效率；因此僅適合較低維度的積分函數。若以預設的 GlobalAdaptive 指派方式下，CartesianRule 和 GaussKronrodRule 則會有較佳的計算效率和精確度。

此外，要注意的是 Nintegrate 指令中的積分範圍必須為一給定的數值，否則 Nintegrate 會因積分範圍未知而無法運算並出現錯誤訊息。

以 NIntegrate 計算，由於 x 並沒有指定特定數值，所以無法計算

In[20]: NIntegrate[f[z], {z, -Infinity, x}]

NIntegrate::nintp : Encountered the non-number x at $z = z$. More...

Out[20]: Nintegrate $\left[\frac{e^{-\frac{z^2}{2}}}{(2\pi)^{0.5}}, \{z, -\infty, x\}\right]$

上述問題發生的原因在於 x 為一未知數，所以 Mathematica 在執行 NIntegrate 時因積分範圍未知而無法運算。解決的方法則是將上述運算式以函數表示。

定義函數 $\Phi(x)$ ，其中?NumberQ 表示指定 x 為數值型態的變數

In[21]: [x_?NumberQ]:=NIntegrate[f[z], {z, -Infinity, x}];

上式中?NumberQ 為 PatternTest，其目的為限制函數只有在參數為數值時才做計算，若參數為一符號時則不做任何計算。

因為 1.96 為一數值，故函數 Φ 會執行計算； x 並非數值，故函數 Φ 不計算

In[22]: {\Phi[1.96], \Phi[x]}

Out[22]: {0.975002, \Phi[x]}

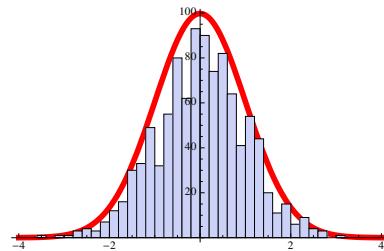
利用 $\Phi(x)$ 產生 1000 個標準常態分配的隨機變數

In[23]: Table[x/.Quiet@FindRoot[[x]==Random[],{x,1}],{1000}];

將上式得到的隨機變數以直方圖繪出

```
In[24]: Show[Plot[1000f[z], {z, -4, 4}, PlotStyle -> {Red, Thickness[0.015]}],  
Histogram[%]]
```

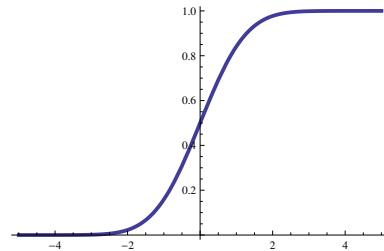
Out[24]:



常態分配累積機率密度函數

```
In[25]: Plot[[x], {x, -5, 5}, PlotStyle -> Thickness[0.01]]
```

Out[25]:



1.5 級數

函數的幕級數展開在微積分中是非常重要的一個工具，若存在一個幕級數 $\sum_{n=0}^{\infty} a_n(x - x_0)^n$ 使得在 x_0 的鄰域有 $f(x) = \sum_{n=0}^{\infty} a_n(x - x_0)^n$ ，則我們稱 $\sum_{n=0}^{\infty} a_n(x - x_0)^n$ 為函數 $f(x)$ 在 x_0 的泰勒級數 (Taylor series)。泰勒級數可將函數轉換成幕級數的形式，由於幕級數的微分和積分可以逐項進行，因此在計算級數和相對比較容易。此外，泰勒級數也可以用來近似計算函數值，其一階近似和二階近似在求解非線性規劃問題也有重要作用。Mathematica 中計算函數的泰勒級數 (Taylor series) 指令為 `Series[f, {x, a, n}]`，表示函數 $f(x)$ 在 $x = a$ 的 n 階的泰勒展開式；當 $a = 0$ 時，泰勒級數也稱為麥克勞林級數 (Maclaurin series)。有關函數 `Series` 的用法，介紹如下：

1. `Series[f,{x, a, n}]`: 函數 f 在 $x = a$ 的 n 階泰勒展開式，所傳回的運算式中 $O(x^n)$ 稱為第 n 項餘式。
2. `Series[f,{x1, a1, n1}, {x2, a2, n2}]`: 函數 f 先對 x_1 在 $x_1 = a_1$ 做 n_1 階展開後再對 x_2 在 $x_2 = a_2$ 展開 n_2 階的泰勒展開式。

另外，與級數較相關的指令有 `SeriesCoefficient`、`Coefficient` 與 `CoefficientList`，其用法如下述：

1. `SeriesCoefficient[f,n]`: 傳回多項式函數 $f(x)$ 中 x^n 項的係數。
2. `SeriesCoefficient[f,{x, x0, n}]`: 傳回多項式函數 $f(x)$ 在 $x = x_0$ 展開後第 x^n 項的係數的通式。
3. `Coefficient[f,x,n]`: 傳回多項式函數 $f(x)$ 中 x^n 項的係數。
4. `CoefficientList[f,x]`: 傳回多項式函數 $f(x)$ 中的係數向量。

範例 1-20. 假設函數 $f(x) = \sin(ax)$

計算 $f(x)$ 在 $x = a$ 的 5 階泰勒展開式

In[1]: `Series[f[x],{x,a,5}]`

Out[1]: $f[a] + f'[a](x-a) + \frac{1}{2}f''[a](x-a)^2 + \frac{1}{6}f^{(3)}[a](x-a)^3 + \frac{1}{24}f^{(4)}[a](x-a)^4 + \frac{1}{120}f^{(5)}[a](x-a)^5 + O[x-a]^6$

上式中的 $O((x-a)^6)$ 即為泰勒級數的 5 次餘項。

定義函數

In[2]: `f[x_]:=Sin[a*x]`

計算 $f(x)$ 在 $x = 0$ 的 10 階泰勒展開式

In[3]: `Series[f[x],{x, 0, 10}]`

Out[3]: $ax - \frac{a^3x^3}{6} + \frac{a^5x^5}{120} - \frac{a^7x^7}{5040} + \frac{a^9x^9}{362880} + O[x]^{11}$

將上式中餘式省略

In[4]: Normal[%]

$$\text{Out}[4]: ax - \frac{a^3 x^3}{6} + \frac{a^5 x^5}{120} - \frac{a^7 x^7}{5040} + \frac{a^9 x^9}{362880}$$

分別將 $\sin(x)$ 在 $x = 0$ 的 10 階泰勒展開式中的各項係數取出

In[5]: SeriesCoefficient[f[x] /. a -> 1, {x, 0, #}] & /@ Range[10]

$$\text{Out}[5]: \left\{ 1, 0, -\frac{1}{6}, 0, \frac{1}{120}, 0, -\frac{1}{5040}, 0, \frac{1}{362880}, 0 \right\}$$

$\sin(x)$ 的泰勒展開式第 n 項係數的通式

In[6]: SeriesCoefficient[Sin[x], {x, 0, n}]

$$\text{Out}[6]: \begin{cases} \frac{i i^n (-1 + (-1)^n)}{2n!} & n \geq 0 \\ 0 & \text{True} \end{cases}$$

利用 CoefficientList 將各項係數取出

In[7]: CoefficientList[Series[f[x], {x, 0, 10}] /. a -> 1, x]

$$\text{Out}[7]: \left\{ 1, 0, -\frac{1}{6}, 0, \frac{1}{120}, 0, -\frac{1}{5040}, 0, \frac{1}{362880}, 0 \right\}$$

在泰勒展開式中我們是以多項式函數來逼近原式。因此，隨著 n 的增加，泰勒展開式就可以更逼近原式。

以下我們以不同的 n 來近似 $\sin(x)$ 做說明。

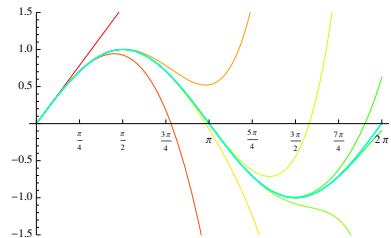
分別產生 $\sin(x)$ 的 $2, 4, 6, 8, 10, \dots, 20$ 階泰勒展開式

In[8]: y=Normal[Series[Sin[x], {x, 0, #}]] & /@ Range[2, 20, 2];

將上式產生的各展開式與 $\sin(x)$ 同時繪出

```
In[9]: Plot[Evaluate[{y, Sin[x]}], {x, 0, 2Pi}, PlotRange->{-1.5, 1.5},
PlotStyle->Table[Hue[k], {k, 0, 1, 0.05}],
Ticks->{Flatten[Table[{i}, {i, 0, 2Pi, Pi/4}]], Automatic}]
```

Out[9]:



由上圖可以發現，當 $n = 20$ 時，泰勒展開式與 $\sin(x)$ 在 $x \in (0, 2\pi)$ 之間幾乎無差別。惟利用高次方程式作近似曲線時容易產生大幅度的震盪，即為龍格效應。

範例 1-21. 證明對所有的 $x > 0$ ，函數 $f(x) = \frac{e^x - 1 - x}{x^2}$ 為一凸函數

定義函數

```
In[1]: f[x_]:= (Exp[x]-1-x)/x^2
```

函數對 x 做一階及二階微分

```
In[2]: D[f[x], {x, #}]&/@{1, 2}//FullSimplify
```

```
Out[2]: {2 + e^x(-2 + x) + x, -2(3 + x) + e^x(6 + (-4 + x)x)}
```

由上式結果我們很難判斷函數的二階導數是否為正數。為判斷 $f(x)''$ 是否為正數，我們由 $-2(3 + x) + e^x(6 + (-4 + x)x)$ 著手。

對 $-2(3 + x) + e^x(6 + (-4 + x)x)$ 做一階及二階導數

```
In[3]: D[Numerator[%[[2]]], {x, #}]&/@{1, 2}//FullSimplify
```

```
Out[3]: {-2 + e^x(2 + (-2 + x)x), e^x x^2}
```

由於 $-2(3 + x) + e^x(6 + (-4 + x)x)$ 的二階導數恆正，故其一階導數為一單調遞增函數。由 $\lim_{x \rightarrow 0} -2(3 + x) + e^x(6 + (-4 + x)x) = 0$ ，故對所有的 $x > 0$ ， $-2(3 + x) + e^x(6 + (-4 + x)x)$ 恒為正數。因此，可推得 $f(x)$ 為凸函數。

接下來，我們改以級數方式來證明 $f(x)$ 為凸函數。

以 SeriesCoefficient 取出函數 $f(x)$ 一階導數泰勒展開式係數的通式

In[4]: SeriesCoefficient[D[f[x], {x, 1}], {x, 0, n}] // FullSimplify

Out[4]:

$$\begin{cases} -\frac{2}{\text{Gamma}[4+n]} & -3 < n < -2 \\ \frac{1+n}{\text{Gamma}[4+n]} & n > -2 \\ 0 & \text{True} \end{cases}$$

由於 $n \geq 0$ 且 $\text{Gamma}[x] = (x-1)!$, 故可得 $f'(x) = \sum_{n=0}^{\infty} \frac{1+n}{(n+3)!} x^n > 0$ 即 $f(x)$ 為一單調遞增函數。上式中若不想以 Gamma 的方式顯示，則可改以 $\text{Gamma}[x] := (x-1)!$ 方式以階層方式顯示。

以 SeriesCoefficient 取出函數 $f(x)$ 二階導數泰勒展開式係數的通式

In[5]: FullSimplify@SeriesCoefficient[D[f[x], {x, 2}], {x, 0, n}] /.
 $\text{Gamma}[x_] := (x-1)!$

Out[5]:

$$\begin{cases} \frac{6}{(4+n)!} & -4 < n < -3 \\ \frac{(1+n)(2+n)}{(4+n)!} & n \geq -2 \\ -\frac{2(5+2n)}{(4+n)!} & -3 < n < -2 \\ 0 & \text{True} \end{cases}$$

然而，在這個例子中我們僅需要討論 $n \geq 0$ 的部份。所以若要 Mathematica 只列出 $n \geq 0$ 的通式，則可藉由 Select 來達成。

將上式 $n \geq 0$ 的通式取出

In[6]: FullSimplify@Select[SeriesCoefficient[D[f[x], {x, 2}], {x, 0, n}] [[1]], (# [[2]] /. n -> 0 == True &) [[1, 1]] /. $\text{Gamma}[x_] := (x-1)!$
Out[6]: $\frac{(1+n)(2+n)}{(4+n)!}$

驗算結果是否正確

In[7]: D[f[x], {x, 2}] - Sum[%*x^n, {n, 0, Infinity}] // FullSimplify

Out[7]: 0

由輸出結果可知 $f'(x) = \sum_{n=0}^{\infty} \frac{1+n}{(n+3)!} x^n > 0$ 且 $f''(x) = \sum_{n=0}^{\infty} \frac{(1+n)(2+n)}{(4+n)!} x^n > 0$, 故 $f(x)$ 為一嚴格單調遞增的凸函數。

上述我們討論的方式為先利用 $f(x)$ 一階及二階導數的泰勒級數來判斷二階導數是否恆為正數。事實上，若

我們直接對 $f(x)$ 做泰勒展開，則可以更容易的利用 Mathematica 來證明 $f(x) = \frac{e^x - 1 - x}{x^2}$ 為一嚴格凸函數。

以 SeriesCoefficient 直接取出 $f(x)$ 泰勒展開式係數的通式

In[8]: FullSimplify@SeriesCoefficient[f[x], {x, 0, n}]/.Gamma[x_]:=>(x-1)!

Out[8]:

$$\begin{cases} -1 + \frac{1}{(2+n)!} & n == -2 \text{||} n == -1 \\ \frac{1}{(2+n)!} & -2 < n < -1 \text{||} n > -1 \\ 0 & \text{True} \end{cases}$$

因此，對所有的 $x > 0$ ， $f(x)$ 可改寫成

$$f(x) = \sum_{n=0}^{\infty} \frac{x^n}{(2+n)!}$$

及

$$f''(x) = \sum_{n=2}^{\infty} \frac{n(n-1)x^{n-2}}{(2+n)!} > 0$$

故可輕易推得 $f(x) = \frac{e^x - 1 - x}{x^2}$ 為一嚴格凸函數。

以上我們所探討的皆為單變數下 Mathematica 中計算級數的方法；接下來我們介紹函數 Series 在多變數的情況下的應用。

範例 1-22. 假設函數 $f(x, y) = \sin(x + y)$

定義函數 $f(x, y) = \sin(x + y)$

In[1]: f[x_, y_] := Sin[x+y]

先將 $f(x, y) = \sin(x + y)$ 以 $x = 0$ 求 3 階泰勒展開式，接著在以 $y = 0$ 展開 3 階泰勒展開式

In[2]: Series[f[x, y], {x, 0, 3}, {y, 0, 3}]

Out[2]: $\left(y - \frac{y^3}{6} + O[y]^4\right) + \left(1 - \frac{y^2}{2} + O[y]^4\right)x + \left(-\frac{y}{2} + \frac{y^3}{12} + O[y]^4\right)x^2 + \left(-\frac{1}{6} + \frac{y^2}{12} + O[y]^4\right)x^3 + O[x]^4$

我們必須注意上式並非 $f(x, y)$ 以 $(x, y) = (0, 0)$ 展開的泰勒展開式。而是先將 $f(x, y)$ 以 $x = 0$ 求 3 階展開式，之後再對 x 次方項的係數 y 求 3 階的泰勒展開式，說明如下：

先將 $f(x, y) = \sin(x + y)$ 以 $x = 0$ 求 3 階泰勒展開式

In[3]: Series[f[x, y], {x, 0, 3}]

Out[3]: $\text{Sin}[y] + \text{Cos}[y]x - \frac{1}{2}\text{Sin}[y]x^2 - \frac{1}{6}\text{Cos}[y]x^3 + O[x]^4$

將上式的係數項取出

```
In[4]: mycoff=CoefficientList[%,x]
Out[4]: {Sin[y],Cos[y],-Sin[y]/2,-Cos[y]/6}
```

分別將各係數項對 y 取 3 階泰勒展開式

```
In[5]: Map[Series[#, {y,0,3}]&,mycoff]
Out[5]: {y - y^3/6 + O[y]^4, 1 - y^2/2 + O[y]^4, -y/2 + y^3/12 + O[y]^4, -1/6 + y^2/12 + O[y]^4}
```

我們在此可以發現，將%5 的結果代回%3 中可得到與%2 相同的結果。所以，Mathematica 中函數 Series 在多變數中所求得的展開式並非泰勒展開式。若要求得 $\sin(x+y)$ 之 n 階泰勒展開式方式如下：

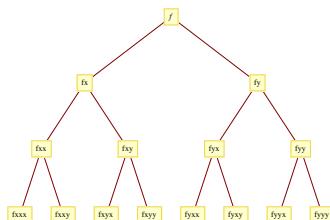
建立一個 3 階泰勒展開式的關係圖

```
In[6]: diff=Rule@@@Flatten[NestList[Flatten[Map[{f[[2]],#[[2]]<>"x"},{#[[2]],#[[2]]<>"y"}]&,#],1]&,{{"f","f"},3},1][[2;;-1]];
```

輸出 3 階泰勒展開式樹狀圖

```
In[7]: TreePlot[diff,VertexLabeling->True]
```

Out[7]:



建立一個以 $(x,y) = (a,b)$ 展開 n 階泰勒展開式的公式

```
In[8]: mytaylor[f_,a_,b_,k_]:=Plus@@Flatten@Table[Binomial[n,i]*D[f[x,y],{x,i},{y,n-i}]/.{x->a,y->b}*(x-a)^i*(y-b)^(n-i)/n!,{n,0,k},{i,0,n}];
```

傳回 $(x,y) = (0,0)$ 展開 3 階泰勒展開式

```
In[9]: mytaylor[f,0,0,3]
Out[9]: x - x^3/6 + y - x^2y/2 - xy^2/2 - y^3/6
```

傳回 $(x, y) = (0, 0)$ 展開 5 階泰勒展開式

In[10]: mytaylor[f, 0, 0, 5]

$$\text{Out}[10]: x - \frac{x^3}{6} + \frac{x^5}{120} + y - \frac{x^2 y}{2} + \frac{x^4 y}{24} - \frac{x y^2}{2} + \frac{x^3 y^2}{12} - \frac{y^3}{6} + \frac{x^2 y^3}{12} + \frac{x y^4}{24} + \frac{y^5}{120}$$

以 SeriesCoefficient 取出函數 $f(x, y)$ 泰勒展開式 $x^n y^m$ 係數的通式

In[11]: SeriesCoefficient[f[x, y], {x, 0, n}, {y, 0, m}]

$$\text{Out}[11]: \begin{cases} \frac{\cos[\frac{1}{2}(-1+m+n)\pi]}{m!n!} & n \geq 0 \& m \geq 0 \\ 0 & \text{True} \end{cases}$$

因此 $f(x, y) = \sin(x + y)$ 可改寫為

$$\sin(x + y) = \sum_{n=0}^{\infty} \sum_{m=0}^{\infty} \frac{\cos(\frac{1}{2}(-1+m+n)\pi)}{m!n!} x^n y^m$$

定義函數 g

In[12]: g=Out[11][[1,1,1]]

$$\text{Out}[12]: \frac{\cos[\frac{1}{2}(-1+m+n)\pi]}{m!n!}$$

驗算結果是否正確

In[13]: Sum[%[[1,1,1]]x^n*y^m,{n,0,Infinity},{m,0,Infinity}]

-Sin[x+y]//FullSimplify

Out[13]: 0

使用 SeriesCoefficient 傳回 $(x, y) = (0, 0)$ 展開 3 階泰勒展開式

In[14]: Total[Flatten@Table[g*x^n*y^m,{n,0,3},{m,0,3-n}]]

$$\text{Out}[14]: x - \frac{x^3}{6} + y - \frac{x^2 y}{2} - \frac{x y^2}{2} - \frac{y^3}{6}$$

使用 SeriesCoefficient 傳回 $(x, y) = (0, 0)$ 展開 5 階泰勒展開式

In[15]: Total[Flatten@Table[g*x^n*y^m,{n,0,5},{m,0,5-n}]]

$$\text{Out}[15]: x - \frac{x^3}{6} + \frac{x^5}{120} + y - \frac{x^2 y}{2} + \frac{x^4 y}{24} - \frac{x y^2}{2} + \frac{x^3 y^2}{12} - \frac{y^3}{6} + \frac{x^2 y^3}{12} + \frac{x y^4}{24} + \frac{y^5}{120}$$

範例 1-23. 假設 X 服從常態分配 $N(\mu, \sigma)$, 計算 X 之 k 階動差。

定義常態分配機率密度函數

```
In[1]: f[x_]:=PDF[NormalDistribution[,],x]
Out[1]: 
$$\frac{e^{-\frac{(x-\mu)^2}{2\sigma^2}}}{\sqrt{2\pi}\sigma}$$

```

由定義計算常態分配的 k 階動差

```
In[2]: Integrate[x^#*f[x],{x,-Infinity,Infinity},
Assumptions->{>0}]&/@Range[4]
Out[2]: {μ, μ² + σ², μ³ + 3μσ², μ⁴ + 6μ²σ² + 3σ⁴}
```

由定義計算標準常態分配的 k 階動差

```
In[3]: Integrate[((x-)/)^#*f[x],{x,-Infinity,Infinity},
Assumptions->{>0}]&/@Range[4]
Out[3]: {0, 1, 0, 3}
```

計算常態分配的動差生成函數

```
In[4]: M[t_]:=Integrate[Exp[t*x]*f[x],{x,-Infinity,Infinity},
Assumptions->>0]
Out[4]: e^{t\mu+\frac{t^2\sigma^2}{2}}
```

定義常態分配的 k 階動差

```
In[5]: EX[k_]:=Derivative[k][M][0];
```

計算常態分配的 $1 \sim 4$ 階動差

```
In[6]: EX[#]&/@{1,2,3,4}
Out[6]: {μ, μ² + σ², μ³ + 3μσ², μ⁴ + 6μ²σ² + 3σ⁴}
```

計算標準常態分配的偏態係數

```
In[7]: Expand[(X-)^3/^3]//.{X^n_:>EX[n],X:>EX[1]}
Out[7]: 
$$\frac{2\mu^3}{\sigma^3} - \frac{3\mu(\mu^2 + \sigma^2)}{\sigma^3} + \frac{\mu^3 + 3\mu\sigma^2}{\sigma^3}$$

```

化簡上式

In[8]: FullSimplify[%]

Out[8]: 0

計算標準常態分配的峰態係數

In[9]: Expand[(X-)^4/^4]//.{X^n_:>EX[n], X:>EX[1]}

Out[9]: $-\frac{3\mu^4}{\sigma^4} + \frac{6\mu^2(\mu^2 + \sigma^2)}{\sigma^4} - \frac{4\mu(\mu^3 + 3\mu\sigma^2)}{\sigma^4} + \frac{\mu^4 + 6\mu^2\sigma^2 + 3\sigma^4}{\sigma^4}$

化簡上式

In[10]: FullSimplify[%]

Out[10]: 3

Mathematica 在 8.0 之後也提供了一系列的機率相關函數以供使用，其中 Moment 和 ExpectedValue 函數則可分別用來求解指定機率密度的動差生成函數及期望值。

使用 Moment 函數計算常態分配的前四階動差

In[11]: Moment[NormalDistribution[, #] & /@ Range[4]]

Out[11]: $\{\mu, \mu^2 + \sigma^2, \mu(\mu^2 + 3\sigma^2), \mu^4 + 6\mu^2\sigma^2 + 3\sigma^4\}$

使用 ExpectedValue 函數計算標準常態分配的平均數、變異數、偏態係數及峰態係數

In[12]: ExpectedValue[((x-)/)^#, NormalDistribution[, x] & /@ Range[4]]

Out[12]: {0, 1, 0, 3}

第 2 章 方程式求解

求解方程式或方程組為數學上常遇到的問題。在 Mathematica 中求解方程式或方程組的方式有很多種方式，大致可分為求解析解及數值解兩種類型。當方程式或方程組具有解析解時，我們可以用 `Solve`、`Reduce` 及 `Roots` 等指令來整理出一些代數公式。另一方面，當方程式或方程組無解析解時則只能藉助數值方法求其數值解，而對於這類問題我們可以用 `FindRoot` 指令來求解。

2.1 方程式具解析解

若方程式具解析解，則表示方程式的解可藉由有限次常見運算的組合出。在 Mathematica 中函數求解方程式或聯立方程組最基本的指令有 `Solve`、`Reduce` 及 `Roots`。這些指令用法介紹如下：

1. `Solve[lhs == rhs, x]`: 以 x 為變數求解方程式。
2. `Solve[{eqn1, eqn2, ...}, {x1, x2, ...}]`: 以 x_1, x_2, \dots 為變數求解聯立方程組。
3. `Reduce[lhs == rhs, x]`: 以 x 為變數化簡方程式。
4. `Reduce[{eqn1, eqn2, ...}, {x1, x2, ...}]`: 以 x_1, x_2, \dots 為變數化簡聯立方程組。
5. `Reduce[lhs == rhs, x, domain]`: 在 x 的定義域上化簡方程式。
6. `Reduce[{eqn1, eqn2, ...}, {x1, x2, ...}]`: 在 x_1, x_2, \dots 的定義域上化簡聯立方程組。
7. `Roots[lhs == rhs, x]`: 以 x 為變數求解多項式函數。

以上這些指令雖然都可作為求解的工具，但目的卻不盡相同。`Solve` 主要的目的是用來求解方程式和方程組；`Reduce` 雖然可以用來作為求解的工具，但其主要目的是化簡等式和不等式，藉以釐清變數間複雜關係；`Roots`

的應用則較小，僅能用來求解多項式函數的根。

首先，我們來看一個常見的一元二次方程式 $ax^2 + bx + c = 0$ 。由過去的經驗我們知道一元二次方程式 $ax^2 + bx + c = 0$ ，在 $a \neq 0$ 時的公式解為 $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ 。

範例 2-1. 求解 $ax^2 + bx + c = 0$

以 Reduce 化簡方程式

```
In[1]: Reduce[a*x^2+b*x+c==0,x]
Out[1]: ((a != 0 && (x == -(b - Sqrt[b^2 - 4 a c])/2 a || x == (b + Sqrt[b^2 - 4 a c])/2 a)) ||
(a == 0 && b != 0 && x == -c/b) || (c == 0 && b == 0 && a == 0))
```

使用 Resolve 與 Exists 求解方程式有實根的條件

```
In[2]: Resolve[Exists[x, a*x^2+b*x+c==0], Reals]
Out[2]: (a == 0 && b != 0) || (a == 0 && c == 0) || (a != 0 && -b^2 + 4 a c <= 0)
```

由以上輸出可以了解到一元二次方程式的性質：當 $a = 0$ 時， $x = -\frac{c}{b}$ ；當 $a \neq 0$ 時，則解即為我們熟悉的一元二次方程式的公式解。

以 Roots 求解

```
In[3]: Roots[a*x^2+b*x+c==0,x]
Out[3]: x == -(b - Sqrt[b^2 - 4 a c])/2 a || x == (b + Sqrt[b^2 - 4 a c])/2 a
```

以 Solve 求解

```
In[4]: Solve[a*x^2+b*x+c==0,x]
Out[4]: {x -> -(b - Sqrt[b^2 - 4 a c])/2 a, x -> (b + Sqrt[b^2 - 4 a c])/2 a}
```

由以上兩式輸出可以發現，使用 Solve 及 Roots 在計算時 Mathematica 會假設方程式 $ax^2 + bx + c = 0$ 的公式解存在，即 $a \neq 0$ ；而 Reduce 則會輸出所有可能情況下的解。

使用 Reduce 求解 $6x^2 + 13x + 5 = 0$

```
In[5]: Reduce[6*x^2+13*x+5==0,x]
Out[5]: x == -5/3 || x == -1/2
```

傳回 Reduce 化簡所得的解

In[6]: Reduce[$6x^2 + 13x + 5 == 0$, x] /. {x_ == y_ :> y, Or -> List}

Out[6]: $\left\{-\frac{5}{3}, -\frac{1}{2}\right\}$

使用 Roots 求解 $6x^2 + 13x + 5 = 0$

In[7]: Roots[$6x^2 + 13x + 5 == 0$, x] /. {x_ == y_ :> y, Or -> List}

Out[7]: $\left\{-\frac{5}{3}, -\frac{1}{2}\right\}$

使用 Solve 求解 $6x^2 + 13x + 5 = 0$

In[8]: x /. Solve[$6x^2 + 13x + 5 == 0$, x]

Out[8]: $\left\{-\frac{5}{3}, -\frac{1}{2}\right\}$

範例 2-2. 將方程式 $x^2 + bx + 1 = 0, -2 \leq b \leq 2$ 所有的根繪製在複數平面上。

以 Solve 求解 $x^2 + bx + 1 = 0$

In[1]: Solve[$x^2 + b*x + 1 == 0$, x]

Out[1]: $\left\{\frac{1}{2} \left(-b - \sqrt{-4 + b^2}\right), \frac{1}{2} \left(-b + \sqrt{-4 + b^2}\right)\right\}$

以 Reduce 求解 $x^2 + bx + 1 = 0$

In[2]: Reduce[$x^2 + b*x + 1 == 0$, x, Reals] // Simplify

Out[2]: $(b == -2 \& \& x == 1) \parallel (b == 2 \& \& x == -1) \parallel ((b < -2 \parallel b > 2) \& \& (b + \sqrt{-4 + b^2} + 2x == 0 \parallel \sqrt{-4 + b^2} == b + 2x))$

以 Resolve 與 Exists 求解 b 的範圍

In[3]: Resolve[Exists[x, x^2 + b*x + 1 == 0], Reals]

Out[3]: $b \leq -2 \parallel b \geq 2$

由以上輸出可發現方程式 $x^2 + bx + 1 = 0$ 有實數根的區間為 $|b| \geq 2$ 。

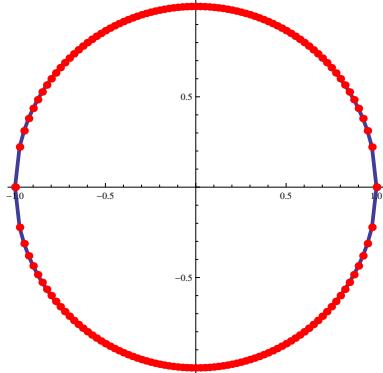
傳回 $b = -2, -1.95, \dots, 1.95, 2$ 所對應根的實部及虛部

In[4]: data = Flatten[{Re[x], Im[x]}] /. Solve[x^2 + # x + 1 == 0, x] & /@ Range[-2, 2, 0.05], 1];

將 data 繪製在複數平面上

```
In[5]: ListLinePlot[Flatten[{data[[Range[1,Length@data,2]]],  
Reverse@data[[Range[2,Length@data,2]]]},1],  
AspectRatio->1,PlotStyle->Thickness[0.01],  
Epilog->{Red,PointSize[0.02],Point[data]}]
```

Out[5]:



以上的一元二次方程式的解法很早就為人所熟悉。接下來我們利用 Mathematica 來求解一般化的一元高次方程式的公式解。

範例 2-3. 求解 $ax^3 + bx^2 + cx + d = 0$

使用 Resolve 與 Exists 求解方程式 $ax^3 + bx^2 + cx + d = 0$ 有實根的條件

```
In[1]: Resolve[Exists[x,a*x^3+b*x^2+c*x+d==0],Reals]//Simplify]
```

```
Out[1]: a ≠ 0||(c == 0&&((d ≥ 0&&b < 0)|| (b > 0&&d ≤ 0)))||  
(c ≠ 0&&4bd ≤ c^2) ||d == 0
```

假設 $y = x - \frac{b}{3a}$, 重新化簡 $ax^3 + bx^2 + cx + d = 0$

```
In[2]: Collect[a*x^3+b*x^2+c*x+d/.x->y-b/(3a),y,Simplify]
```

```
Out[2]:  $\frac{2b^3}{27a^2} - \frac{bc}{3a} + d + \left(-\frac{b^2}{3a} + c\right)y + ay^3$ 
```

將上式 y^3 的係數 a 削去

```
In[3]: Collect[%/a,y,Simplify]
```

```
Out[3]:  $\frac{2b^3 - 9abc + 27a^2d}{27a^3} - \frac{(b^2 - 3ac)y}{3a^2} + y^3$ 
```

所以一個標準型的一元三次方程式可以藉由 $x = y - \frac{b}{3a}$ 轉換成 $y^3 + py + q = 0$ 。

以 Solve 求解 $y^3 + py + q = 0$

In[4]: `Solve[y^3+p*y+q==0,y]//Simplify`

Out[4]: $\left\{ \begin{array}{l} y \rightarrow \frac{-2 \times 3^{1/3} p + 2^{1/3} (-9q + \sqrt{12p^3 + 81q^2})^{2/3}}{6^{2/3} (-9q + \sqrt{12p^3 + 81q^2})^{1/3}} \\ y \rightarrow \frac{2(3i + \sqrt{3}) p + i \times 2^{1/3} 3^{1/6} (i + \sqrt{3}) (-9q + \sqrt{12p^3 + 81q^2})^{2/3}}{22^{2/3} 3^{5/6} (-9q + \sqrt{12p^3 + 81q^2})^{1/3}} \\ y \rightarrow \frac{2(-3i + \sqrt{3}) p + 2^{1/3} 3^{1/6} (-1 - i\sqrt{3}) (-9q + \sqrt{12p^3 + 81q^2})^{2/3}}{2 \times 2^{2/3} 3^{5/6} (-9q + \sqrt{12p^3 + 81q^2})^{1/3}} \end{array} \right\}$

產生 26 個英文字母

In[5]: `ToExpression@CharacterRange["a", "z"]`

Out[5]: {a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z}

產生一元 n 次方程式

In[6]: `f[n_?IntegerQ /; n >= 1] := Table[x^i, {i, 0, n}] . Reverse@ToExpression@CharacterRange["a", "z"] [[1;;n+1]]`

輸出一元 4,5,6 次方程式

In[7]: `Map[f,{4,5,6}]//TableForm`

Out[7]: $e + dx + cx^2 + bx^3 + ax^4$
 $f + ex + dx^2 + cx^3 + bx^4 + ax^5$
 $g + fx + ex^2 + dx^3 + cx^4 + bx^5 + ax^6$

求解一元四次方程式

In[8]: `Solve[f[4]==0,x]//Short`

Out[8]: $\{\langle\langle 1 \rangle\rangle\}$

求解一元五次方程式

In[9]: `Solve[f[5]==0,x]//Short`

Out[9]: $\{\{x \rightarrow \text{Root}[f + e\#1 + d\#1^2 + c\#1^3 + b\#1^4 + a\#1^5 \&, 1]\},$
 $\langle\langle 3 \rangle\rangle, \{x \rightarrow \text{Root}[f + e\#1 + d\#1^2 + c\langle\langle 1 \rangle\rangle + b\#1^4 + a\#1^5 \&, 5]\}\}$

求解一元六次方程式

In[10]: `Solve[f[6]==0,x]//Short`

Out[10]: $\{\{x \rightarrow \text{Root}[g + f\#1 + e\#1^2 + d\#1^3 + c\#1^4 + b\#1^5 + a\#1^6\&, 1]\}, \langle 4 \rangle, \{x \rightarrow \text{Root}[g + f\#1 + \langle 3 \rangle + b\#1^5 + a\#1^6\&, 6]\}\}$

上式中 `Root` 指令表示回多項式 $f(x) = 0$ 的第 k 個根，其用法為 `Root[f[x],k]`。由以上的結果我們可以了解一般的一元 n 次方程式在 $n = 1, 2, 3, 4$ 時有公式解，但在 $n \geq 5$ 時則無統一的公式解。因此在 Mathematica 中對於一般 5 次以上的一元方程式僅能以 `Root` 方式求得數值解。

範例 2-4. 求解 $x^5 - 4x + 2 = 0$

以 `Solve` 求解

In[1]: `Solve[x^5-4*x+2==0,x]`

Out[1]: $\{\{x \rightarrow \text{Root}[2 - 4\#1 + \#1^5\&, 1]\}, \{x \rightarrow \text{Root}[2 - 4\#1 + \#1^5\&, 2]\}, \{x \rightarrow \text{Root}[2 - 4\#1 + \#1^5\&, 3]\}, \{x \rightarrow \text{Root}[2 - 4\#1 + \#1^5\&, 4]\}, \{x \rightarrow \text{Root}[2 - 4\#1 + \#1^5\&, 5]\}\}$

由於方程式 $x^5 - 4x + 2 = 0$ 並無解析解，故 Mathematica 在此僅能以 `Root` 指令提供數值解。所以要傳回%1 的數值解有以下方式。

使用 `N`，要求將 `Solve` 所得到的結果轉為數值格式

In[2]: `sol=x/.Solve[x^5-4*x+2==0,x]//N`

Out[2]: $\{-1.51851, 0.508499, 1.2436, -0.116792 - 1.43845i, -0.116792 + 1.43845i\}$

使用 `NSolve` 輸出數值解

In[3]: `sol=x/.NSolve[x^5-4*x+2==0,x]`

Out[3]: $\{-1.51851, -0.116792 - 1.43845i, -0.116792 + 1.43845i, 0.508499, 1.2436\}$

上兩式所使用的 `N@Solve` 與 `NSolve` 會輸出相同的結果，其差異僅是精確度不同而已。若想控制精確度，可使用指令 `NSolve[方程式, 變數, 準確度]`。

使用 `NSolve` 輸出數值解，精確度設定為小數點後第四位

In[4]: `x/.NSolve[x^5-4*x+2==0,x,5]`

Out[4]: $\{-1.5185, -0.1168 - 1.4384i, -0.1168 + 1.4384i, 0.50850, 1.2436i\}$

以 Solve 方式傳回第 2 個根

In[5]: `Solve[x^5-4*x+2==0,x][[2]]//N`

Out[5]: `{x → 0.508499}`

以 Root 方式傳回第 2 個根

In[6]: `Root[x^5-4*x+2,x,2]//N`

Out[6]: `{x → 0.508499}`

以 Root 方式傳回 $x^5 - 4x + 2 = 0$ 所有根

In[7]: `N@Root[x^5-4*x+2,x,#]&/@{1,2,3,4,5}`

Out[7]: `{-1.51851, 0.508499, 1.2436, -0.116792 - 1.43845i, -0.116792 + 1.43845i}`

上式中可以發現方程式 $x^5 - 4x + 2 = 0$ 分別有 3 個實根以及 2 個虛根。若只要傳回上式的實根，可利用以下幾種篩選的方式：Cases, Select 以及 DeleteCases。

以 Cases 傳回實數根

In[8]: `Cases[sol,_Real]`

Out[8]: `{-1.51851, 0.508499, 1.2436}`

以 Cases 傳回正的實數根

In[9]: `Cases[sol,x_/_;Element[x,Reals]&&x>0]`

Out[9]: `{0.508499, 1.2436}`

以 Select 傳回實數根

In[10]: `Select[sol,Element[#,Reals]&]`

Out[10]: `{-1.51851, 0.508499, 1.2436}`

以 Select 傳回正的實數根

In[11]: `Select[sol,Element[#,Reals]&&#>0&]`

Out[11]: `{-1.51851, 0.508499, 1.2436}`

以 DeleteCases 將虛根移除

In[12]: DeleteCases[sol, _Complex]

Out[12]: {−1.51851, 0.508499, 1.2436}

以 DeleteCases 將負數根或虛根移除

In[13]: DeleteCases[sol, x_ /; Negative[x] || Head[x] == Complex] |

Out[13]: {0.508499, 1.2436}

直接以 Reduce 篩選定義域內的根

In[14]: Reduce[{x^5 - 4*x + 2 == 0, x > 0}, x, Reals] /. {Equal[x_, y_] :> N@y, Or -> List}

Out[14]: {0.508499, 1.2436}

Mathematica8.0 以後的版本，Solve 及 NSolve 也提供變數定義域的選項，使得在不等式或限制式的求解更為便利。

以 Solve 求解正的實數根

In[15]: x /. N@Solve[{x^5 - 4*x + 2 == 0, x > 0}, x, Reals]

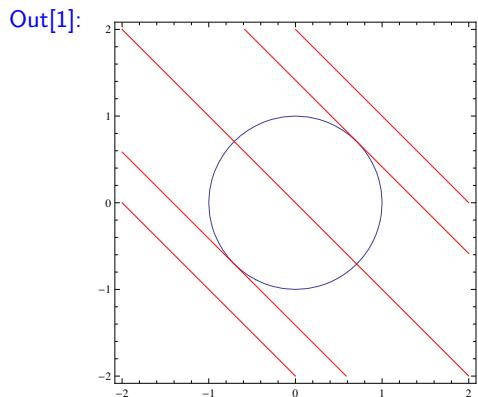
Out[15]: {0.508499, 1.2436}

範例 2-5. 求解聯立方程組

$$\begin{cases} x^2 + y^2 = 1 \\ x + y = a \end{cases}$$

輸出函數圖

```
In[1]: Show[ContourPlot[x^2+y^2==1,{x,-2,2},{y,-2,2}],
ContourPlot[x+y==#, {x,-2,2},{y,-2,2},
ContourStyle->Red]&/@{-2,-Sqrt[2],0,Sqrt[2],2}]
```



以 Reduce 化簡聯立方程組

```
In[2]: Reduce[{x+y==a,x^2+y^2==1},{x,y}]
Out[2]: \left(x == \frac{1}{2} \left(a - \sqrt{2 - a^2}\right) \wedge x == \frac{1}{2} \left(a + \sqrt{2 - a^2}\right)\right) \& \& y == a - x
```

以 Reduce 化簡聯立方程組，並輸出 a 的定義域

```
In[3]: Reduce[Exists[{x,y},x+y>=a,x^2+y^2==1],a]
Out[3]: a \leq \sqrt{2}
```

以 Reduce 化簡聯立方程組，並輸出 a 的定義域

```
In[4]: Reduce[Exists[{x,y},x+y<=a,x^2+y^2==1],a]
Out[4]: a \geq -\sqrt{2}
```

由以上輸出的圖形及 Reduce 化簡結果顯示當 $a^2 = 2$ 時，直線 $x + y = a$ 與圓 $x^2 + y^2 == 1$ 相切於一點；當 $a^2 < 2$ 時，直線 $x + y = a$ 會過圓 $x^2 + y^2 = 1$ 且與圓相交於兩點；當 $a^2 > 2$ 時，直線 $x + y = a$ 與圓 $x^2 + y^2 = 1$ 則無交點。

傳回 Reduce 化簡所得的解

```
In[5]: Reduce[{x+y==a,x^2+y^2==1},{x,y}]/.{x==z_:>z,Or->List,
And->List}//Thread
Out[5]: \left\{\left\{\frac{1}{2} \left(a - \sqrt{2 - a^2}\right), a - x\right\}, \left\{\frac{1}{2} \left(a + \sqrt{2 - a^2}\right), a - x\right\}\right\}
```

上式中可發現 y 的值 Reduce 並未將 $x = \frac{1}{2}(a - \sqrt{2 - a^2})$ 代入化簡。若想直接傳回所有可能的解，則可藉由參數 Backsubstitution 將化簡的的函數代回來達成。

Reduce 使用 Backsubstitution 參數可直接傳回所有可能的解

```
In[6]: Reduce[{x+y==a,x^2+y^2==1},{x,y},Backsubstitution->True]/.
{x_==z_:>z,Or->List,And->List}
Out[6]: { {1/2 (a - Sqrt[2 - a^2]), 1/2 (a + Sqrt[2 - a^2])}, {1/2 (a + Sqrt[2 - a^2]), 1/2 (a - Sqrt[2 - a^2])} }
```

傳回 Reduce 化簡 $x > 0$ 時所得的整數解

```
In[7]: Reduce[{x+y==a,x^2+y^2==1,x>0},{x,y},Integers]
Out[7]: a == 1 && x == 1 && y == 0
```

以 Solve 求解

```
In[8]: sol={x,y}/.Solve[{x+y==a,x^2+y^2==1},{x,y}]//Simplify
Out[8]: { {1/2 (a - Sqrt[2 - a^2]), 1/2 (a + Sqrt[2 - a^2])}, {1/2 (a + Sqrt[2 - a^2]), 1/2 (a - Sqrt[2 - a^2])} }
```

範例 2-6. 設一正方形 ABCD，其中 AC 兩端點在 $x + y = 4$ ，BD 兩端點在 $y = x^2$ ，求此正方形面積？

定義距離函數，在此為方便起見，我們以不考慮根號

```
In[1]: distance[pt1_List,pt2_List]:=Sqrt[(pt2-pt1).(pt2-pt1)];
```

定義斜率函數

```
In[2]: slope[pt1_List,pt2_List]:=(pt2[[2]]-pt1[[2]])/(pt2[[1]]-pt1[[1]]);
```

假設正方形 ABCD 的四個端點分別為 (x_1, y_1) , (x_2, y_2) , (x_3, y_3) 和 (x_4, y_4) ，由於 AC 兩端點在直線 $x + y = 4$ 上，故 AC 兩點座標可表示為 $(x_1, 4 - x_1)$ 和 $(x_3, 4 - x_3)$ ；又 BD 兩端點在曲線 $y = x^2$ 上，故 BD 兩點座標可表示為 (x_2, x_2^2) 和 (x_4, x_4^2) 。

定義正方形 ABCD 的四個端點

```
In[3]: pts={{x1,4-x1},{x2,x2^2},{x3,4-x3},{x4,x4^2}};
```

正方形 ABCD 的四個邊長需相等

```
In[4]: eq1={distance[#[[1]],#[[2]]]==distance[#[[2]],#[[3]]]}&/@  
Partition[Insert[pts,pts[[1]],-1],3,1]  
Out[4]: {Sqrt[(-x1+x2)^2+(-4+x1+x2^2)^2]==Sqrt[(4-x2^2-x3)^2+(-x2+x3)^2],  
Sqrt[(4-x2^2-x3)^2+(-x2+x3)^2]==Sqrt[(-x3+x4)^2+(-4+x3+x4^2)^2],  
Sqrt[(-x3+x4)^2+(-4+x3+x4^2)^2]==Sqrt[(x1-x4)^2+(4-x1-x4^2)^2]}
```

正方形 ABCD 的對角線必須垂直，由於 AC 對角線斜率為 -1 ，故可知 BD 對角線斜率為 1

```
In[5]: eq2=slope[pts[[2]],pts[[4]]]==1  
Out[5]: -x2^2+x4^2 == 1 -x2 + x4
```

正方形 ABCD 的對角線必須相等

```
In[6]: eq3=distance[pts[[1]],pts[[3]]]==distance[pts[[2]],pts[[4]]]  
Out[6]: (x1-x3)^2+(-x1+x3)^2==(-x2+x4)^2+(-x2^2+x4^2)^2
```

由於正方形 ABCD 中，假設頂點 C 的 x 座標值必須大於頂點 A 的 x 座標值及頂點 D 的 x 座標值必須大於頂點 B 的 x 座標值，故我們可假設以下條件： $x_3 > x_1$ 和 $x_4 > x_2$ 方便 Mathematica 求解。

正方形 ABCD 的座標條件

```
In[7]: eq4={pts[[4,1]]>pts[[2,1]],pts[[3,1]]>pts[[1,1]]}  
Out[7]: {x4 > x2, x3 > x1}
```

使用 Reduce 化簡限制式 eq1、eq2、eq3 及 eq4 並輸出

```
In[8]: ans={{x1,4-x1},{x2,x2^2},{x3,4-x3},{x4,x4^2}}/.  
(Reduce[Flatten[{eq1,eq2,eq3,eq4}],{x1,x2,x3,x4},  
Backsubstitution->True]/.{Equal->Rule,Or->List,And->List})//Simplify  
Out[8]: {{1/2 (1 - Sqrt[13]), 1/2 (7 + Sqrt[13])}, {1/2 (1 - Sqrt[13]), 1/2 (7 - Sqrt[13])},  
{{1/2 (1 + Sqrt[13]), 1/2 (7 - Sqrt[13])}, {1/2 (1 + Sqrt[13]), 1/2 (7 + Sqrt[13])}}
```

計算正方形 ABCD 之邊長

In[9]: `distance@@ans[[1;;2]]//Simplify`

Out[9]: $\sqrt{13}$

計算正方形 ABCD 之面積

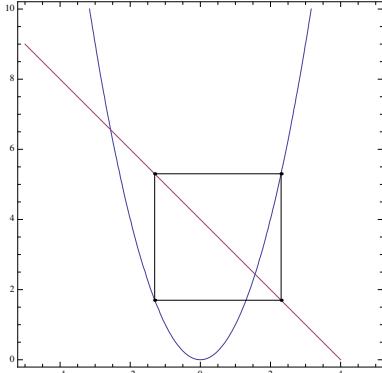
In[10]: `(distance@@ans[[1;;2]])^2//Simplify`

Out[10]: 13

輸出函數及正方形 ABCD 圖形

In[11]: `ContourPlot[{y==x^2,y==4-x},{x,-5,5},{y,0,10},Epilog->{Point[ans],Line[Insert[ans,ans[[1]],-1]]}]`

Out[11]:



範例 2-7. 求解三元一次聯立方程組

$$\begin{cases} x + 2y + 6z = 23 \\ 5x - 3y + z = 2 \\ 2x + 7y - 3z = 13 \end{cases}$$

定義矩陣係數

In[1]: `A={{1,2,6},{5,-3,1},{2,7,-3}};`
`B={23,2,13};`
`X={x,y,z};`

以 Reduce 化簡聯立方程組

In[2]: `Reduce[A.X==B,X]/.{Equal->Rule,And->List}`

Out[2]: $\left\{x \rightarrow \frac{67}{47}, y \rightarrow \frac{123}{47}, z \rightarrow \frac{128}{47}\right\}$

以 Solve 求解聯立方程組

In[3]: `Solve[A.X==B,X]`

Out[3]: $\left\{\left\{x \rightarrow \frac{67}{47}, y \rightarrow \frac{123}{47}, z \rightarrow \frac{128}{47}\right\}\right\}$

利用 Fold 函數列出代入消去法計算過程

In[4]: `FoldList[Eliminate,A.X==B,{z,y}]//TableForm`

Out[4]: $\{x + 2y + 6z, 5x - 3y + z, 2x + 7y - 3z\} == \{23, 2, 13\}$
 $17x == 19 + 2y \quad \& \quad 47y == 123$
 $47x == 67$

對於一些較大規模的聯立方程組，Mathematica 也提供了矩陣的求解函數。

建立 Gaussian-Jordan 消去法所需的增廣矩陣

In[5]: `gj=Insert[A[[#]],B[[#]],-1]&/@{1,2,3};
gj//MatrixForm`

Out[5]:
$$\begin{pmatrix} 1 & 2 & 6 & 23 \\ 5 & -3 & 1 & 2 \\ 2 & 7 & -3 & 13 \end{pmatrix}$$

以 Gaussian-Jordan 消去法求解聯立方程組

In[6]: `RowReduce[gj]//MatrixForm`

Out[6]:
$$\begin{pmatrix} 1 & 0 & 0 & \frac{67}{47} \\ 0 & 1 & 0 & \frac{123}{47} \\ 0 & 0 & 1 & \frac{128}{47} \end{pmatrix}$$

以 LinearSolve 求解聯立方程組

In[7]: `LinearSolve[A,B]`

Out[7]: $\left\{\frac{67}{47}, \frac{123}{47}, \frac{128}{47}\right\}$

直接由矩陣運算求解聯立方程組

In[8]: Inverse[A].B

Out[8]: $\left\{ \frac{67}{47}, \frac{123}{47}, \frac{128}{47} \right\}$

以上範例由於係數矩陣為一非奇異矩陣 (Nonsingular Matrix)，因此具有唯一的反矩陣來求解。接下來我們介紹當線性方程組具無窮多組解的範例。

範例 2-8. 求解三元一次聯立方程組

$$\begin{cases} x + 2y + 6z = 23 \\ 5x - 3y + z = 2 \\ 10x - 6y + 2z = 4 \end{cases}$$

定義矩陣係數

In[1]: A={{1,2,6},{5,-3,1},{10,-6,2}};
B={23,2,4};
X={x,y,z};

以 Reduce 化簡聯立方程組

In[2]: Reduce[A.X==B,X]
Out[2]: $y == \frac{11}{20} + \frac{29x}{20} \& \& z == \frac{73}{20} - \frac{13x}{20}$

由聯立方程組很清楚可以看出方程式 $5x - 3y + z = 2$ 與 $10x - 6y + 2z = 4$ 重疊，因此我們無法求得唯一解。所以由上式輸出的結果可知當 x 給定後，則三元一次聯立方程組會退化為一二元一次聯立方程組。

令 $x = t$ ，以參數式表示

In[3]: {t,y,z}/. (Reduce[A.X==B,X]/.{x->t, Equal->Rule, And->List})
Out[3]: $\left\{ t, \frac{11}{20} + \frac{29t}{20}, \frac{73}{20} - \frac{13t}{20} \right\}$

上式中的解之表示為參數式，其中 t 為參數，其實也可以令 $y = t$ 為參數或 $z = t$ 為參數，雖然會得出不同形式的解集合，但這些不同形式的集合都是一樣。

建立 Gaussian-Jordan 消去法所需的增廣矩陣

In[4]: gj=Insert[A[[#]],B[[#]],-1]&/@{1,2,3};
gj//MatrixForm

Out[4]:
$$\begin{pmatrix} 1 & 2 & 6 & 23 \\ 5 & -3 & 1 & 2 \\ 10 & -6 & 2 & 4 \end{pmatrix}$$

以 Gaussian-Jordan 消去法求解聯立方程組

In[5]: RowReduce[gj]//MatrixForm

Out[5]:
$$\begin{pmatrix} 1 & 0 & \frac{20}{13} & \frac{73}{13} \\ 0 & 1 & \frac{29}{13} & \frac{113}{13} \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

由於 Gaussian-Jordan 消去法最後結果為一列梯形，所以聯立方程組並無唯一解。

令 $z = t$, 以參數式表示

In[6]: {x,y,t}/.Solve[RowReduce[gj][[{1,2},1;;3]],{x,y,t}]==
RowReduce[gj][[{1,2},-1]],{x,y}][[1]]

Out[6]: $\left\{ \frac{1}{13}(73 - 20t), \frac{1}{13}(113 - 29t), t \right\}$

以 LinearSolve 求解聯立方程組只會傳回 $z = 0$ 的特解

In[7]: LinearSolve[A,B]

Out[7]: $\left\{ \frac{73}{13}, \frac{113}{13}, 0 \right\}$

範例 2-9. 考慮狀態空間 $F = \{0, 1, 2\}$ 的馬可夫鏈，已知其轉移矩陣為：

$$\begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{6} & \frac{1}{3} & \frac{1}{2} \end{bmatrix}$$

求此馬可夫鏈的穩定狀態。

定義轉移矩陣

In[1]: T={{1/3,1/3,1/3},{1/4,1/2,1/4},{1/6,1/3,1/2}};

利用 `Solve` 求解穩定狀態

```
In[2]: Solve[{x,y,z}.T=={x,y,z},{x,y,z}]
Solve::svrs: Equations may not give solutions for all "solve" variables. »
Out[2]: {y -> 5x/3, z -> 3x/2}
```

由於機率總和必須為 1，加上此條件後即可求解穩定狀態的機率。

利用 `Solve` 求解穩定狀態機率

```
In[3]: Solve[Flatten@{{x,y,z}.T=={x,y,z},x+y+z==1},{x,y,z}]
Out[3]: {x -> 6/25, y -> 2/5, z -> 9/25}
```

利用 `Reduce` 求解穩定狀態機率

```
In[4]: Reduce[Flatten@{{x,y,z}.T=={x,y,z},x+y+z==1},
{x,y,z}]/.{Equal->Rule,And->List}
Out[4]: {x -> 6/25, y -> 2/5, z -> 9/25}
```

利用 `NestWhile` 求解穩定機率

```
In[5]: NestWhile[#, T &, NCQ[{1/3, 1/3, 1/3}], Unequal, All]
Out[5]: {0.24, 0.4, 0.36}
```

利用 `NestWhile` 求解穩定狀態

```
In[6]: NestWhile[#, T &, NCQ[T], Unequal, All]
Out[6]: {{0.24, 0.4, 0.36}, {0.24, 0.4, 0.36}, {0.24, 0.4, 0.36}}
```

2.2 方程式不具解析解

一般而言，大多數的方程式都不具解析解，在此種情況下該方程式只能利用數值逼近法來求出根。假設聯立方程組

$$f_i(x) = 0, i = 1, 2, \dots, m$$

其中 f 為一連續可微分函數。由泰勒展開式可知， x 在 x^k 的一階泰勒展開式為

$$f_i(x) \approx f_i(x^k) + \nabla f_i(x^k)(x - x^k), i = 1, 2, \dots, m \quad (2-1)$$

由於 $f_i(x) = 0, i = 1, 2, \dots, m$, 故可知

$$f_i(x^k) + \nabla f_i(x^k)(x - x^k) = 0 \quad (2-2)$$

經移項整理後，可求得

$$x = x^k - [\nabla f_i(x^k)]^{-1} f_i(x^k) \quad (2-3)$$

由上式可分別求得單變數與多變數牛頓法 (Newton Method) 的迭代計算過程間遞迴關係式如下：

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \quad (2-4)$$

及

$$x^{k+1} = x^k - [\nabla f_i(x^k)]^{-1} f_i(x^k) \quad (2-5)$$

其中 $\nabla f_i(x^k)$ 為 Jacobian,

$$\nabla f_i(x^k) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}$$

由於 Newton Method 使用到一階導數；若一階導數難以求得，我們可以差分法， $\frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}$ ，作為 $f'(x_k)$ 的近似值，則 (2-4) 可改寫為

$$x_{k+1} = x_k - f(x_k) \times \frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})} \quad (2-6)$$

此法則稱為正切法 (Secant method)。

在Mathematica 中函數求數值解的指令為 FindRoot, FindRoot 預設以 Newton Method 做為預設的搜尋方法。由於牛頓法的收斂速度與起始值的設定有關，為避免迭代過程發散，求解之前可先繪製函數圖形以利設定起始值，其用法有以下幾種：

1. `FindRoot[lhs == rhs, {x, a}]`: 以 $x = a$ 為起始值，利用 Newton method 尋找左右兩方程式交點的數值解。
2. `FindRoot[f, {x, a}]`: 以 $x = a$ 為起始值，利用 Newton method 尋找方程式 f 的數值解。必須注意的是，該指令預設為求解 $f = 0$ ，若要求解其他數值可以 $f == c$ 定義。
3. `FindRoot[lhs == rhs, {x, a,b}]`: 以區間 (a, b) 為起始值，利用 Secant method 尋找左右兩方程式交點的數值解。
4. `FindRoot[{f1, f2, ...}, {{x, a1}, {y, a2}, ...}]`: 以 (x, y, \dots) 為起始值，利用 Newton method 尋找聯立方程組 (f_1, f_2, \dots) 的數值解。

範例 2-10. 求解 $e^x - x^2 = 0$

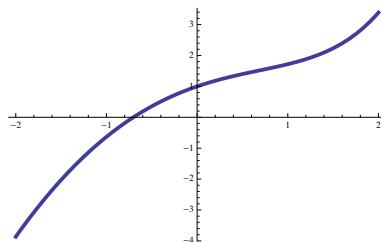
定義函數

In[1]: `f[x_] := Exp[x] - x^2;`

繪製圖形以方便設定起始值

In[2]: `Plot[f[x], {x, -2, 2}, PlotStyle -> Thickness[0.01]]`

Out[2]:



以 While 自行編寫 Newton method 程式

```
In[3]: Block[{x, init=2., err=10, sol={}}, x=init;
  While[err>10^(-6), x=N[x-f[x]/f'[x]]; err=Abs[f[x]]; AppendTo[sol, x]];
  sol]
```

Out[3]: {1., -1.39221, -0.835088, -0.709834, -0.703483, -0.703467}

由 (2-4) 我們可以發現

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

為一個遞迴函數，因此，我們在這邊可以 NestList 或 FixedPoint 來將程式寫得更簡潔。

以 NestList 傳回 6 次迭代計算的結果

In[4]: NestList[N[#-f[#]/f'[#]]&, 2., 6]

Out[4]: {2, 1., -1.39221, -0.835088, -0.709834, -0.703483, -0.703467}

以 NestWhile 傳回最後計算結果

In[5]: NestWhile[N[#-f[#]/f'[#]]&, 2., Unequal, All]

Out[5]: -0.703467

以 FixedPoint 傳回最後計算結果

In[6]: FixedPoint[N[#-f[#]/f'[#]]&, 2.]

Out[6]: -0.703467

利用 NestWhileList 來輸出整個迭代計算的過程

In[7]: data=N@{#,f[#]}&/@NestWhileList[N[#-f[#]/f'[#]]&, 2., Unequal, All];

輸出 data

In[8]: TableForm@data

Out[8]:

2	3.38906
1.	1.71828
-1.39221	-1.68973
-0.835088	-0.263535
-0.709834	-0.0121387
-0.703483	-0.0000303943
-0.703467	-1.92216×10^{-10}
-0.703467	0.
-0.703467	0.

利用 FixedPointList 來輸出整個迭代計算的過程

```
In[9]: TableForm[N@{#,f[#]}&/@FixedPointList[N[ #-f[#]/f'[#]]&,2.]]
```

```
Out[9]: 2 3.38906
         1. 1.71828
        -1.39221 -1.68973
       -0.835088 -0.263535
      -0.709834 -0.0121387
     -0.703483 -0.0000303943
    -0.703467 -1.92216 × 10-10
   -0.703467 0.
  -0.703467 0.
```

Secant method 以區間 $(-2, 2)$ 進行求解

```
In[10]: FixedPointList[N@{#[[2]],#[[2]]}
  -f#[[2]]*(#[[2]]-#[[1]])/(f#[[2]]-f#[[1]])}&,
  {-2,2},SameTest->(Abs[#[1[[1]]]-#[2[[1]]]]<10-5&)]//TableForm
```

```
Out[10]: -2 2
         2. 0.131135
        0.131135 -0.794935
      -0.794935 -0.666807
     -0.666807 -0.702161
    -0.702161 -0.703487
   -0.703487 -0.703467
  -0.703467 -0.703467
 -0.703467 -0.703467
```

Regula Falsi method (假位法) 以區間 $(-2, 2)$ 進行求解

```
In[11]: Block[{c},c[{x_,y_}]=(x*f[y]-y*f[x])/ (f[y]-f[x])//N;
  FixedPointList[If[Sign[c@#[[1;;2]]]==Sign[f#[[2]]]],
  #[[1]],c@#[[1;;2]],c@#[[1;;2]],{c@#[[1;;2]],#[[2]],c@#[[1;;2]]}]&,
  {-2,2,0},SameTest->(Abs[#[1[[3]]]-#[2[[3]]]]<10-5&)]//TableForm
```

```
Out[11]: -2 2 0
         -2 0.131135 0.131135
        -0.348677 0.131135 -0.348677
      -0.868704 0.131135 -0.868704
     -0.638882 0.131135 -0.638882
    -0.730767 0.131135 -0.730767
   -0.692284 0.131135 -0.692284
  -0.708109 0.131135 -0.708109
 -0.701551 0.131135 -0.701551
 -0.70426 0.131135 -0.70426
 -0.70314 0.131135 -0.70314
 -0.703603 0.131135 -0.703603
 -0.703411 0.131135 -0.703411
 -0.703491 0.131135 -0.703491
 -0.703458 0.131135 -0.703458
 -0.703471 0.131135 -0.703471
 -0.703466 0.131135 -0.703466
```

Bisection method (二分法) 以區間 $(-2, 2)$ 進行求解

```
In[12]: FixedPointList[N@If[Sign[f[Mean@#[[1;;2]]]==Sign[f#[[2]]],  
#[[1]],Mean@#[[1;;2]],Mean@{#[[1]],Mean@#[[1;;2]]}],  
{Mean@#[[1;;2]],#[[2]],Mean@{#[[2]],Mean@#[[1;;2]]}}]&,  
{-2,2,0},SameTest->(Abs[#1[[3]]-#2[[3]]]<10^-6&)]//TableForm
```

```
Out[12]: -2 2 0  
-2. 0. -1.  
-1. 0. -0.5  
-1. -0.5 -0.75  
-0.75 -0.5 -0.625  
-0.75 -0.625 -0.6875  
-0.75 -0.6875 -0.71875  
-0.71875 -0.6875 -0.703125  
-0.71875 -0.703125 -0.710938  
-0.710938 -0.703125 -0.707031  
-0.707031 -0.703125 -0.705078  
-0.705078 -0.703125 -0.704102  
-0.704102 -0.703125 -0.703613  
-0.703613 -0.703125 -0.703369  
-0.703613 -0.703369 -0.703491  
-0.703491 -0.703369 -0.70343  
-0.703491 -0.70343 -0.703461  
-0.703491 -0.703461 -0.703476  
-0.703476 -0.703461 -0.703468  
-0.703468 -0.703461 -0.703465  
-0.703468 -0.703465 -0.703466  
-0.703468 -0.703466 -0.703467
```

利用 FindRoot 函數以 2 為起始值進行求解

```
In[13]: FindRoot[f[x],{x,2}]
```

```
Out[13]: {x → -0.703467}
```

FindRoot 以 2 為起始值，利用 Secant method 進行求解

```
In[14]: FindRoot[f[x],{x,-2,2},Method→"Secant"]
```

```
Out[14]: {x → -0.703467}
```

FindRoot 以 2 為起始值，利用 Brent method 進行求解

```
In[15]: FindRoot[f[x],{x,-2,2},Method→"Brent"]
```

```
Out[15]: {x → -0.703467}
```

若要了解函數 FindRoot 的收斂過程，我們也可藉由參數 EvaluationMonitor 或 StepMonitor 來要求將計算過程輸出。

利用 Reap 與 Sow 函數傳回 Newton method 的迭代步驟

In[16]: `Reap[FindRoot[f[x], {x, 2}, StepMonitor :> Sow[x]]]`

Out[16]: $\{\{x \rightarrow -0.703467\}, \{1., -1.39221, -0.835088, -0.709834, -0.703483, -0.703467, -0.703467\}\}$

利用 Reap 與 Sow 函數傳回 Secant method 的迭代步驟

In[17]: `Reap@FindRoot[f[x], {x, -2, 2}, Method -> "Secant", StepMonitor :> Sow[x]]`

Out[17]: $\{\{x \rightarrow -0.703467\}, \{0.131135, -0.794935, -0.666807, -0.702161, -0.703487, -0.703467, -0.703467\}\}$

利用 Reap 與 Sow 函數傳回 Brent method 的迭代步驟

In[18]: `Reap@FindRoot[f[x], {x, -2, 2}, Method -> "Brent", StepMonitor :> Sow[x]]`

Out[18]: $\{\{x \rightarrow -0.703467\}, \{2., 0.131135, -0.586436, -0.732718, -0.702079, -0.703451, -0.703467\}\}$

Mathematica 也有提供一個繪製 FindRoot 迭代步驟圖的指令 `FindRootPlot`。`FindRootPlot` 所傳回資料陣列包含三個元素，依序為收斂值、求解資訊及收斂步驟圖。

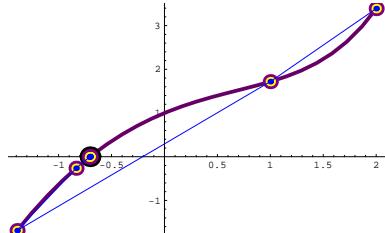
引入最佳化套件繪製求解步驟圖

In[19]: `<<Optimization`UnconstrainedProblems``

傳回收斂步驟圖

In[20]: `FindRootPlot[f[x], {x, 2}, PlotStyle -> {Thickness[.01], RGBColor[0.4, 0, 0.4]}][[3]]`

Out[20]:



由圖形可知，收斂過程為 $x_0 = 2 \Rightarrow x_1 = 1 \Rightarrow x_2 \Rightarrow -1.39221 \Rightarrow \dots \Rightarrow x_7 = -0.703467$ 。

傳回求解資訊

```
In[21]: FindRootPlot[f[x],{x,2},
PlotStyle->{Thickness[.01],RGBColor[0.4,0,0.4]}][[{1,2}]]
```

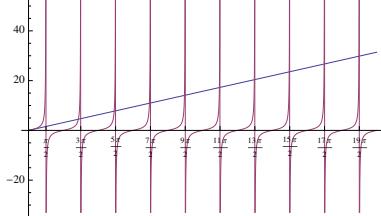
```
Out[21]: {{x → -0.703467},{Steps → 8,Residual → 8,Jacobian → 7}}
```

範例 2-11. 求解 $\tan(x) = x$

輸出函數 $\tan(x)$ 與 x 的圖形

```
In[1]: Plot[{x,Tan[x]},{x,0,10Pi},
Ticks->{{{-Pi/2+##*Pi}&}/@Range[0,10]//Flatten}]
```

Out[1]:



由輸出圖形可知方程式 $\tan(x) = x$ 存在無窮多組解；且在每個區間 $(-\frac{\pi}{2} + k\pi, -\frac{\pi}{2} + (k+1)\pi)$, $k = 1, 2, \dots$, 皆存在唯一解。

以 $-\frac{\pi}{2} + (k + \frac{1}{4})\pi$ 為起始值，利用 FindRoot 求解 $x \geq 0$ 的前五個解

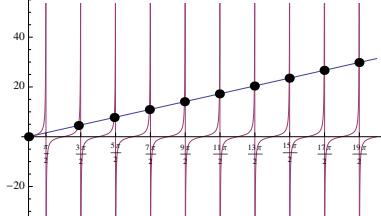
```
In[2]: x/.FindRoot[Tan[x]-x,{x,-Pi/2+(#+1/4)Pi}]&/@Range[0,4]
```

```
Out[2]: {0.,4.49341,7.72525,10.9041,14.0662}
```

輸出函數 $\tan(x)$ 與 x 的圖形

```
In[3]: Plot[{x,Tan[x]},{x,0,10Pi},
Ticks->{{{-Pi/2+##*Pi}&}/@Range[0,10]//Flatten},
Epilog->{PointSize[0.02],Point[{x,Tan[x]}]/.
FindRoot[Tan[x]-x,{x,-Pi/2+(#+1/4)Pi}]&/@Range[0,10]}]
```

Out[3]:



搜尋特定區間的解

```
In[4]: ft[a_?NumberQ,b_?NumberQ]:=Select[Round[x,0.000001]/
FindRoot[Tan[x]-x,{x,-Pi/2+(#+1/4)Pi}]&/@Range[Floor[(a+Pi/2)/Pi],
Floor[(b+Pi/2)/Pi]],a<=#<=b&]
```

輸出 $0 \leq x \leq 20$ 的所有解

```
In[5]: ft[0,20]
```

```
Out[5]: {0, 4.49341, 7.72525, 10.9041, 14.0662, 17.2208}
```

範例 2-12. 求解聯立方程組

$$\begin{cases} x^2 + y^2 = 1 \\ y - xe^x = 0 \end{cases}$$

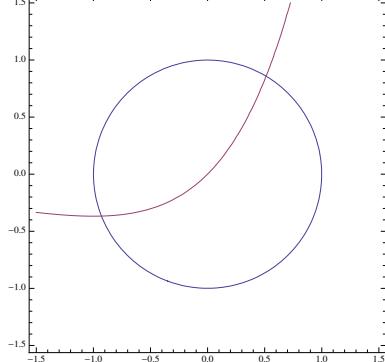
定義函數

```
In[1]: f[x_,y_]:=x^2+y^2-1;g[x_,y_]:=y-x*Exp[x];
```

輸出函數圖形，可發現交點有兩處

```
In[2]: ContourPlot[{f[x,y],g[x,y]},{x,-1.5,1.5},{y,-1.5,1.5}]
```

Out[2]:



接下來我們利用牛頓法求解聯立方程組

計算 Jacobian

```
In[3]: J[x_,y_]:=D[{f[x,y],g[x,y]},{x,y}]
```

```
Out[3]: {{2x, 2y}, {-e^x - e^x x, 1}}
```

定義 $[\nabla f_i(\mathbf{X}^k)]^{-1} f_i(\mathbf{X}^k)$

In[4]: `delta[x_,y_]:=Inverse[J[x,y]].{f[x,y],g[x,y]}/Simplify`

Out[4]: $\left\{ \frac{-1+x^2+2e^xxy-y^2}{2(x+e^x(1+x)y)}, \frac{2xy+e^x(-1+y^2+x(-1+(-1+x)x+y^2))}{2(x+e^x(1+x)y)} \right\}$

分別以 $(1, 0.5)$ 和 $(-1, 0.5)$ 為起始值利用 NestWhile 傳回數值解

In[5]: `NestWhile[N[#-delta@@#]&,#,Unequal,All]&/@{{1,0.5},{-1,0.5}}`

Out[5]: $\{ \{0.513489, 0.858096\}, \{-0.930244, -0.366942\} \}$

分別以 $(1, 0.5)$ 和 $(-1, 0.5)$ 為起始值利用 FixedPoint 傳回數值解

In[6]: `FixedPoint[N[#-delta@@#]&,#]&/@{{1,0.5},{-1,0.5}}`

Out[6]: $\{ \{0.513489, 0.858096\}, \{-0.930244, -0.366942\} \}$

將 NestWhile 的迭代計算過程輸出

In[7]: `data=NestWhileList[N[#-delta@@#]&,#,Unequal,All]&/@{{1,0.5},{-1,0.5}};`

輸出 data1

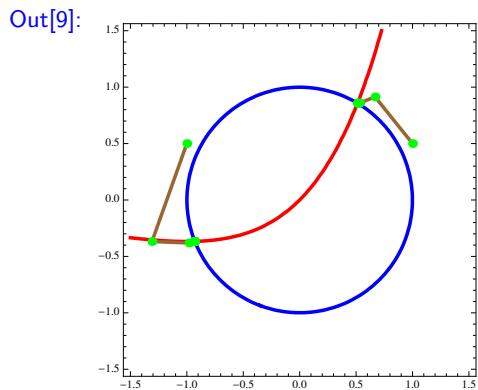
In[8]: `Grid[{data[[1]]//TableForm,"",data[[2]]//TableForm}],Alignment->Top]`

Out[8]:

1	0.5	-1	0.5
0.668088	0.913823	-1.30894	-0.367879
0.531751	0.859521	-0.9811	-0.38091
0.51377	0.858118	-0.93146	-0.367461
0.513489	0.858096	-0.930245	-0.366942
0.513489	0.858096	-0.930244	-0.366942
0.513489	0.858096	-0.930244	-0.366942
		-0.930244	-0.366942

繪出最佳解迭代過程圖

```
In[9]: ContourPlot[{f[x,y],g[x,y]},{x,-1.5,1.5},{y,-1.5,1.5},
ContourStyle->{{Blue,Thickness[0.01]},{Red,Thickness[0.01]}},
Epilog->{Brown,Thickness[0.01],Line[data[[1]]],Line[data[[2]]],
Green,PointSize[0.025],Point[data[[1]]],Point[data[[2]]]}]
```



分別以 $(1, 0.5)$ 和 $(-1, 0.5)$ 為起始值，利用 FindRoot 求解聯立方程組

```
In[10]: FindRoot[{f[x,y],g[x,y]},{x,#},{y,0.5}]&/@{1,-1}
```

```
Out[10]: {{x \rightarrow 0.513489, y \rightarrow 0.858096}, {x \rightarrow -0.930244, y \rightarrow -0.366942}}
```

使用 Reap 及 Sow 將以 $(1, 0.5)$ 及 $(-1, 0.5)$ 為起始值的迭代過程以表格輸出。

```
In[11]: Grid[{TableForm[Reap[FindRoot[{f[x,y],g[x,y]},{x,#},{y,0.5}],
StepMonitor:>Sow[{x,y}]]][[2,1]]]&/@{1,-1}],Alignment->Top]
```

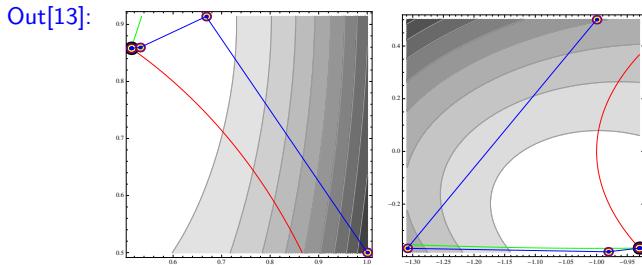
```
Out[11]: 0.668088 0.913823 -1.30894 -0.367879
0.531751 0.859521 -0.9811 -0.38091
0.51377 0.858118 -0.93146 -0.367461
0.513489 0.858096 -0.930245 -0.366942
0.513489 0.858096 -0.930244 -0.366942
0.513489 0.858096 -0.930244 -0.366942
```

引入最佳化套件繪製求解步驟圖

```
In[12]: <<OptimizationUnconstrainedProblems`
```

分別輸出以 $(1, 0.5)$ 及 $(1, 0.5)$ 為起始值的求解的步驟圖

In[13]: `GraphicsGrid[{FindRootPlot[{f[x,y],g[x,y]},{{x,#},{y,0.5}}][[3]]&/@{1,-1}]}]`



輸出以 $(1, 0.5)$ 為起始值的求解資訊

In[14]: `FindRootPlot[{f[x,y],g[x,y]},{{x,1},{y,0.5}}][[{1,2}]]`

Out[14]: `{x->0.513489,y->0.858096},{Steps->7,Residual->7,Jacobian->6}`

以上我們所介紹的範例中 `FindRoot` 均是以個別變數為求解的標的，事實上 `FindRoot` 也以矩陣為變數作為求解的標的。接下來我們以求解特徵向量及反矩陣為例。

範例 2-13. 假設

$$\mathbf{A} = \begin{pmatrix} 1 & 2 & -1 \\ 1 & 0 & 1 \\ 4 & -4 & 5 \end{pmatrix}$$

求 \mathbf{A} 之特徵值與特徵向量。

輸入資料

In[1]: `A={{1,2,-1},{1,0,1},{4,-4,5}};`

建立特徵方程式

In[2]: `char=Det[A-*IdentityMatrix[3]]`

Out[2]: $6 - 11\lambda + 6\lambda^2 - \lambda^3$

使用 `Solve` 計算特徵值

In[3]: `eigvalue=/.Solve[char==0,]`

Out[3]: `{1, 2, 3}`

由於 $\mathbf{A}\mathbf{x} = \lambda_i \mathbf{x}$, 所以我們可藉由簡單的列運算來求得特徵向量 \mathbf{x}

建立 $\lambda = i$ 的增廣矩陣

```
In[4]: pmatrix[i_]:=Insert[(A-eigvalue[[i]]*IdentityMatrix[3])[[#]],0,-1]&/@Range[3]
```

對 $\lambda = i$, $i = 1, 2, 3$, 增廣矩陣進行 Gaussian-Jordan 消去法

```
In[5]: MatrixForm[RowReduce[pmatrix[#]]]&/@Range[3]
```

$$\text{Out[5]: } \left\{ \begin{pmatrix} 1 & 0 & \frac{1}{2} & 0 \\ 0 & 1 & -\frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 & \frac{1}{2} & 0 \\ 0 & 1 & -\frac{1}{4} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 & \frac{1}{4} & 0 \\ 0 & 1 & -\frac{1}{4} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \right\}$$

由上式輸出可知 λ_i 所對應的特徵向量並無唯一解，故僅能求解一般化的參數式。

將 $z = t$ 代入求解一般化參數式

```
In[6]: {x,y,t}/.Solve[RowReduce[pmatrix[#]][[{1,2},1;;3]].{x,y,t}=={0,0}, {x,y}][[1]]&/@Range[3]
```

$$\text{Out[6]: } \left\{ \left\{ -\frac{t}{2}, \frac{t}{2}, t \right\}, \left\{ -\frac{t}{2}, \frac{t}{4}, t \right\}, \left\{ -\frac{t}{4}, \frac{t}{4}, t \right\} \right\}$$

由輸出可知 λ_1 所對應的特徵向量為 $\{-t, t, 2t\}$; λ_2 所對應的特徵向量為 $\{-2t, t, 4t\}$ 及 λ_3 所對應的特徵向量為 $\{-t, t, 4t\}$, 其中 t 為任意常數。由於特徵向量有無窮多組，所以為了有唯一解，我們可利用函數 Normalize 將特徵向量標準化。

將特徵向量做標準化轉換

```
In[7]: TableForm[N@Refine[Normalize[#], t>0]&/%, TableHeadings->{{P1,P2,P3},None}]
```

$$\text{Out[7]: } \begin{array}{c|ccc} & P1 & P2 & P3 \\ \hline P1 & -0.408248 & 0.408248 & 0.816497 \\ P2 & 0.436436 & -0.218218 & -0.872872 \\ P3 & -0.235702 & 0.235702 & 0.942809 \end{array}$$

使用 FindRoot 求解特徵向量

```
In[8]: x/.FindRoot[{A.x==eigvalue[[#]]*x},{x,RandomReal[1,3]}]&/@Range[3]
```

$$\text{Out[8]: } \left\{ \{-0.214811, 0.214811, 0.429622\}, \{-0.00913344, 0.00456672, 0.0182669\}, \{-0.0441798, 0.0441798, 0.176719\} \right\}$$

根據特徵值計算對應之特徵向量

In[9]: TableForm[Normalize/@%,TableHeadings->{{P1,P2,P3},None}]

Out[9]:

	P1	-0.408248	0.408248	0.816497
P2	0.436436	-0.218218	-0.872872	
P3	-0.235702	0.235702	0.942809	

範例 2-14. 假設

$$\mathbf{S} = \begin{bmatrix} 1. & 0.94379 & 0.93254 & -0.01804 & -0.07196 \\ 0.94379 & 1. & 0.77026 & -0.04713 & -0.07802 \\ 0.93254 & 0.77026 & 1. & 0.03639 & -0.04724 \\ -0.01804 & -0.04713 & 0.03639 & 1. & 0.68457 \\ -0.07196 & -0.07802 & -0.04724 & 0.68457 & 1. \end{bmatrix}$$

求 \mathbf{S} 之反矩陣。

輸入資料

In[1]: S={{1.,0.94379,0.93254,-0.01804,-0.07196}, {0.94379,1.,0.77026,-0.04713,-0.07802}, {0.93254,0.77026,1.,0.03639,-0.04724}, {-0.01804,-0.04713,0.03639,1.,0.68457}, {-0.07196,-0.07802,-0.04724,0.68457,1.}};

求 \mathbf{S} 的反矩陣

In[2]: invS=x/.FindRoot[S.x==IdentityMatrix[5],{x,RandomReal[1,{5,5}]}];

以 MatrixForm 輸出 invS

In[3]: invS//MatrixForm

Out[3]:

$$\begin{pmatrix} 191.235 & -105.639 & -97.0979 & 2.57112 & -0.827656 \\ -105.639 & 60.8496 & 51.7052 & -1.1999 & 0.409705 \\ -97.0979 & 51.7052 & 51.8078 & -1.60692 & 0.594325 \\ 2.57112 & -1.1999 & -1.60692 & 1.95291 & -1.32141 \\ -0.827656 & 0.409705 & 0.594325 & -1.32141 & 1.90508 \end{pmatrix}$$

計算 $\mathbf{S}.\text{invS}$

In[4]: S.invS//MatrixForm

Out[4]:
$$\begin{pmatrix} 1. & 0 & 0 & 0 & 0 \\ 0 & 1. & 0 & 0 & 0 \\ 0 & 0 & 1. & 0 & 0 \\ 0 & 0 & 0 & 1. & 0 \\ 0 & 0 & 0 & 0 & 1. \end{pmatrix}$$

第 3 章 線性規劃

線性規劃 (Linear Programming) 是最佳化理論中的重要領域之一，其目的在解決在有限的資源情況下，如何有效地做出最佳分配以發揮資源最高效用。舉例來說，在商業管理領域中，企業決策者必須將有限的資源做最有效率配置，以尋求最佳的解決方案藉以達成利潤極大化或生產過程的成本極小化的企業目標。事實上，很多運籌學中的實際問題都可以線性規劃來描述，例如：運輸、指派及轉運問題。

3.1 線性規劃問題及單形法

假設企業面臨一個牽涉 n 的變數的利潤決策問題，若目標函數及 m ($n > m$) 個資源限制式皆為線性函數，則企業就面臨一個線性規劃問題，可以下列數學模式表示：

$$\max \quad c_1x_1 + c_2x_2 + \cdots + c_nx_n \quad (3-1)$$

subject to

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &\leq b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &\leq b_2 \\ a_{31}x_1 + a_{32}x_2 + \cdots + a_{3n}x_n &\leq b_3 \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &\leq b_m \\ x_1, x_2, \dots, x_n &\geq 0 \end{aligned} \quad (3-2)$$

由於線性規劃的可行解集合若非空集合，則最佳解必存在頂點或可行解區域的邊界上（線性規劃基本定理，Fundamental theorem of linear programming）。因此，對於一個具有唯一最佳解的線性規劃問題，若能列舉出所有的頂點，則可藉由相互比較找出解答。然而，在 n 維空間中每一個頂點均由 n 個超平面相交而成，因為上述問題最多有 $\binom{n+m}{n}$ 個頂點。顯然，當 n 或 m 大時，想以列舉法找所有頂點並非是一件容易的事。

為求解上述線性規劃問題，一般多以 George Dantzig 於 1947 年所提出的代數求解方法（單形法，Simplex method）為主。單形法利用多面體的頂點構造一個可能的解，然後沿著多面體的邊走到目標函數值更高的另一個頂點，直至到達最優解為止。相較於圖解法僅能在兩個變數的線性規劃問題，單形法不論對於小型的問題或是

成千上萬個變數與限制式的大型問題均能有效地求解。為方便單形法計算處理，我們必須將 (3-2) 為單形法的標準形式：

$$\max \quad c_1x_1 + c_2x_2 + \cdots + c_nx_n + 0 \times s_1 + 0 \times s_2 + \cdots + 0 \times s_m$$

subject to

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n + s_1 &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n + s_2 &= b_2 \\ a_{31}x_1 + a_{32}x_2 + \cdots + a_{3n}x_n + s_3 &= b_3 \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n + s_n &= b_m \end{aligned} \quad (3-3)$$

上式可以矩陣形式改寫為

$$\begin{aligned} \min \quad & c^T x \\ \text{subject to} \quad & [A, I]x = b, \quad x \geq 0 \end{aligned}$$

其中， $A = [a_{ij}]$ 為一 $m \times n$ 階矩陣， I 為一 $m \times m$ 階單位矩陣。 $b = [b_i]$ 為一 $1 \times m$ 階矩陣， $c = [c_i]$ 為一 $1 \times n$ 階矩陣， $x = [x_i]$ 則是一 $1 \times n$ 階矩陣，且 x 的每一元皆為非負數。

由於 (3-3) 為一包含 $n+m$ 個變數及 m 個方程式組成的聯立方程組，其自由度為 $n-m$ 。欲求解此一方程組，我們必須令 $n-m$ 個變數為 0 (非基本變數)，再利用其餘 m 個變數 (基本變數) 來求解方程組 (3-3)。此時所得到的 n 個變數的值我們稱為基本解，即為可行解集合的一個頂點。若此一頂點的目標函數值直接來得大，則此一頂點即為最佳解；否則我們便移動到其他的鄰近頂點。我們將單形法的步驟敘述如下：

步驟 1：先將原使線性規劃問題 (3-2) 轉化為標準形式 (3-3)。轉換規則如下：

- 小於 ($<$) 或等於 (\leq) 限制式：將不等式加入閒置變數 (Slack variable)。
- 等號 ($=$) 限制式：在等式限制式中加入人工變數 (Artificial variable)。
- 大於 ($>$) 或等於 (\geq) 限制式：將不等式減去差額變數 (Surplus variable)，之後再人工變數。

步驟 2：選擇一組起始的可行解。

步驟 3：移動到下一個相鄰的頂點，方法如下：

1. 若 c_j 為 c 中係數最大者，則 x_i 做為進入基本變數的非基本變數。
2. 由於僅有 m 個限制式，故需從現有的基本變數中選擇一個做為離去的基本變數，準則為 $\frac{a_{ij}}{b_i}$ 中最小的非負數所對應的基本變數。
3. 利用高斯消去法計算基本變數及目標函數值。

步驟 4：判斷是否為最佳解，若為最佳解則停止，否則返回步驟 3。

單形法的應用上一般會以原點作為起始基本變數，即令 s_1, s_2, \dots, s_m 為基本變數， x_1, x_2, \dots, x_n 為非基本變數。倘若原點並非可行解，則可利用兩階段法（Two Stage method）或大 M 法（Big M method）以求出原始問題的起始可行解。

此外，不難發現單形法終止有兩種情況：第一種情況為對於所有的非基本變數對應的 c_j 均非正數。因為若非基本變數的 c_j 為非正數，則增加 x_j 將使得目前的目標函數變小。第二種情況為對於做為進入基本變數的非基本變數 x_j 其所對應的 a_{ij} 均為非正數。當 a_{ij} 均為非正數時，由於 x_j 的任何變動均在可行解區域上，若令 $x_j \rightarrow \infty$ ，很容易證明此時目標函數值趨近無窮大。

Mathematica 在最佳化套件中所提供之線性規劃函數為 LinearProgramming，其使用方式介紹如下：

1. `LinearProgramming[c, A, b]`：求解在 $A \cdot x \geq b$ 及 $x \geq 0$ 條件下，使得 $c \cdot x$ 有最小值的 x 。
2. `LinearProgramming[c, A, {{b1, z1}, {b2, z2}...}]`：求解在 $A \cdot x \geq b$ 及 $x \geq 0$ 條件下，使得 $c \cdot x$ 有最小值的 x ，其中 $z_i = 1$ 表 $A_i \cdot x \geq b_i$ ， $z_i = 0$ 表 $A_i \cdot x = b_i$ 及 $z_i = -1$ 表 $A_i \cdot x \leq b_i$ 。
3. `LinearProgramming[c, A, b, l]`：求解在 $A \cdot x \geq b$ 及 $x \geq l$ 條件下，使得 $c \cdot x$ 有最小值的 x 。
4. `LinearProgramming[c, A, b, {{l1, u1}, {l2, u2}...}]`：求解在 $A \cdot x \geq b$ 及 $l_i \leq x_i \leq u_i$ 條件下，使得 $c \cdot x$ 有最小值的 x 。
5. `LinearProgramming[c, A, b, lu, dom]`：求解在 $A \cdot x \geq b$ ， $l_i \leq x_i \leq u_i$ 及指定定義域條件下，使得 $c \cdot x$ 有最小值的 x 。

由於函數 `LinearProgramming` 無輸出單形法迭代計算過程的選項，可改由附錄之 `mySimplex` 函數輸出。`mySimplex` 函數用法如下：

1. `mySimplex[x, xB, xNB, A, c, b, 0]`：以 Simplex method 求解極大化線性規劃問題。
2. `LPNormalize[objective, constrains, variables]`：傳回線性規劃問題標準形式資料。
3. `$mySimplexIteration`：傳回各階段的計算過程。
4. `$myVariableIteration`：傳回各階段變數的解。
5. `$mySimplexIterationTable`：傳回各階段的樞紐計算表。

範例 3-1.

$$\max \quad 4x_1 + 5x_2$$

subject to

$$6x_1 + 4x_2 \leq 24$$

$$x_1 + 2x_2 \leq 6$$

$$-x_1 + x_2 \leq 1$$

$$x_2 \leq 2$$

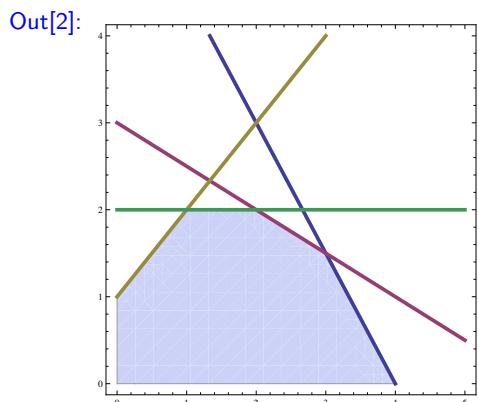
$$x_1, x_2 \geq 0$$

定義目標函數及限制式

```
In[1]: obj=4x1+5x2;
constraints={6x1+4x2<=24,x1+2x2<=6,-x1+x2<=1,x2<=2};
```

輸出可行區域

```
In[2]: Show[RegionPlot[And@@constraints,{x1,0,5},{x2,0,4}],
ContourPlot[Evaluate[constraints/.{LessEqual->Equal}],{x1,0,5},{x2,0,4}]/.{Hue[a_,b_,c_],Line[pts_]}:>{Hue[a,b,c],
Thickness[0.01],Line[pts]},PlotRangePadding->{Scaled[0.03],Scaled[0.03]}]
```



傳回所有可能的頂點

```
In[3]: vertex=Flatten[Solve[#, {x1, x2}]&@
(Subsets[Flatten[{constraints, x1>=0, x2>=0}], {2}]/.
{LessEqual|GreaterEqual->Equal}), 1];
```

傳回所有可行的頂點

```
In[4]: feasiblevertex=Select[vertex,
Times@@Boole[Flatten[{constrains,x1>=0,x2>=0}]/.#]==1&]
Out[4]: {x1→3,x2→3/2},{x1→4,x2→0},{x1→2,x2→2},{x1→1,x2→2},
{x1→0,x2→1},{x1→0,x2→0}
```

比較所有可行的頂點傳回最佳解

```
In[5]: Sort[{obj/.#, #}&@feasiblevertex][[-1]]
Out[5]: {39/2,{x1→3,x2→3/2}}
```

因為 LinearProgramming 函數所求解之問題必須為極小化問題，所以必須對目標函數及限制式做轉換。

定義目標函數係數

```
In[6]: objcoef=Coefficient[-obj,#]&/@{x1,x2}
Out[6]: {-4,-5}
```

定義限制式左方係數

```
In[7]: concoef=Table[-Coefficient[constrains[[z,1]],#]&/@{x1,x2},
{z,Length@constrains}]
Out[7]: {{-6,-4},{-1,-2},{1,-1},{0,-1}}
```

定義限制式右方係數

```
In[8]: bcoef=-constrains[[All,2]]
Out[8]: {-24,-6,-1,-2}
```

使用 LinearProgramming 求解

```
In[9]: LinearProgramming[objcoef,concoef,bcoef]
Out[9]: {3,3/2}
```

計算目標函數之最佳解

```
In[10]: obj/.Rule@@@Transpose@{{x1,x2},%}
```

Out[10]: $\frac{39}{2}$

使用 Maximize 求解

In[11]: Maximize[Flatten@{obj, constraints, x1>=0, x2>=0}, {x1, x2}]

Out[11]: $\left\{\frac{39}{2}, \left\{x_1 \rightarrow 3, x_2 \rightarrow \frac{3}{2}\right\}\right\}$

使用 FindMaximum 求解

In[12]: FindMaximum[Flatten@{obj, constraints, x1>=0, x2>=0}, {x1, x2}]

Out[12]: {19.5, {x1 → 3., x2 → 1.5}}

由於 LinearProgramming 無輸出迭代計算過程選項，以下使用 mySimplex 函數輸出單形法計算步驟。首先將資料依據單形法步驟轉換為標準形式如下：

$$\begin{array}{rcl} 6x_1 + 4x_2 + s_1 & = & 24 \\ x_1 + 2x_2 + s_2 & = & 6 \\ -x_1 + x_2 + s_3 & = & 1 \\ x_2 + s_4 & = & 2 \end{array}$$

定義基本變數

In[13]: bvar={s1, s2, s3, s4};

定義基本變數

In[14]: nonbvar={x1, x2};

定義變數

In[15]: vars=Flatten[{nonbvar, bvar}];

定義目標函數係數矩陣

In[16]: objcoef={4, 5, 0, 0, 0, 0};

定義非基本變數矩陣

```
In[17]: nonbmatrix=-concoef;
```

定義基本變數矩陣

```
In[18]: bmatrix=IdentityMatrix[Length@bvar];
```

定義限制條件係數矩陣

```
In[19]: sol=-bcoef;
```

將限制條件轉換為正規標準型

```
In[20]: simplexmatrix=MapThread[Flatten[{#1,#2}]&,{nonbmatrix,bmatrix}];
```

使用 mySimplex 函數線性規劃問題之極大值

```
In[21]: mySimplex[vars,bvar,nonbvar,simplexmatrix,objcoef,sol,0]
```

$$\text{Out}[21]: \left\{ \frac{39}{2}, \left\{ x_1 \rightarrow 3, x_2 \rightarrow \frac{3}{2} \right\} \right\}$$

我們也可利用函數 LPNormalize 傳回線性規劃問題標準形式資料再進行求解。

利用 LPNormalize 傳回線性規劃問題標準形式資料

```
In[22]: LPNormalize[obj,constraints,{x1,x2}][[4]]//MatrixForm
```

$$\text{Out}[22]: \begin{pmatrix} 6 & 4 & 1 & 0 & 0 & 0 \\ 1 & 2 & 0 & 1 & 0 & 0 \\ -1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

利用 LPNormalize 傳回資料進行單形法求解

```
In[23]: mySimplex@@LPNormalize[obj,constraints,{x1,x2}]
```

$$\text{Out}[23]: \left\{ \frac{39}{2}, \left\{ x_1 \rightarrow 3, x_2 \rightarrow \frac{3}{2} \right\} \right\}$$

傳回單形法各階段的基變數

```
In[24]: $mySimplexIteration[[All,{2,6}]]
```

Out[24]: $\left\{ \left\{ \{s1, s2, x2, s4\}, \{20, 4, 1, 1\} \right\}, \left\{ \{s1, s2, x2, x1\}, \{10, 1, 2, 1\} \right\}, \left\{ \{s1, s3, x2, x1\}, \{4, 1, 2, 2\} \right\}, \left\{ \{s4, s3, x2, x1\}, \left\{ \frac{1}{2}, \frac{5}{2}, \frac{3}{2}, 3 \right\} \right\} \right\}$

傳回單形法各階段變數的解

In[25]: \$myVariableIteration

Out[25]: $\left\{ \{0, 0\}, \{0, 1\}, \{1, 2\}, \{2, 2\}, \left\{ 3, \frac{3}{2} \right\} \right\}$

傳回單形法初始及最後階段的樞紐計算表

In[26]: Column@\$mySimplexIterationTable

Out[26]:

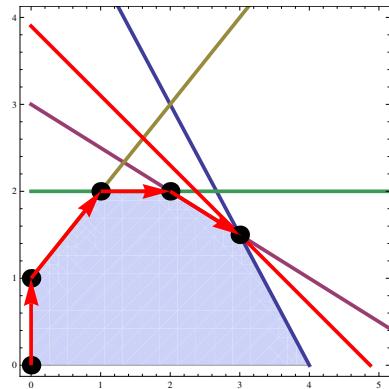
	x1	x2	s11	s12	s13	s14	Solution
c-z	-4	-5	0	0	0	0	0
s11	6	4	1	0	0	0	24
s12	1	2	0	1	0	0	6
s13	-1	1	0	0	1	0	1
s14	0	1	0	0	0	1	2

	x1	x2	s11	s12	s13	s14	Solution
c-z	0	0	$\frac{3}{8}$	$\frac{7}{4}$	0	0	$\frac{39}{2}$
s14	0	0	$\frac{1}{8}$	$-\frac{3}{4}$	0	1	$\frac{1}{2}$
s13	0	0	$\frac{3}{8}$	$-\frac{5}{4}$	1	0	$\frac{5}{2}$
x2	0	1	$-\frac{1}{8}$	$\frac{3}{4}$	0	0	$\frac{3}{2}$
x1	1	0	$\frac{1}{4}$	$-\frac{1}{2}$	0	0	3

輸出單形法的求解路徑

```
In[27]: Show[RegionPlot[And@@constraints,{x1,0,5},{x2,0,4}],  
ContourPlot[Evaluate[constraints/.{LessEqual->Equal}],  
{x1,0,5},{x2,0,4}]/.{Hue[a_,b_,c_],Line[pts_]}:>{Hue[a,b,c],  
Thickness[0.01],Line[pts]},  
ContourPlot[objcoef.vars==39/2,{x1,0,5},{x2,0,4},  
ContourStyle->{Red,Thickness[0.01]}],  
Epilog->{PointSize[0.05],Point[$myVariableIteration],  
Red,Thickness[0.01],Arrowheads[{0,0.075}],  
Arrow/@Partition[$myVariableIteration,2,1]},  
PlotRangePadding->{Scaled[0.03],Scaled[0.03]}]
```

Out[27]:



由以上輸出可發現單純法乃沿著可行解集合的邊界由一個頂點移動到鄰近的下一個頂點，進而求得最佳解。

範例 3-2.

$$\max \quad 3x_1 + 2x_2 + x_3$$

subject to

$$x_1 - x_2 + x_3 \leq 4$$

$$2x_1 + x_2 + 3x_3 \leq 6$$

$$-x_1 + 2x_3 \leq 3$$

$$x_1 + x_2 + x_3 \leq 8$$

$$x_i \geq 0, i = 1, 2, 3$$

定義目標函數及限制式

```
In[1]: obj=3x1+2x2+x3;  
constraints={x1-x2+x3<=4,2x1+x2+3x3<=6,-x1+2x3<=3,x1+x2+x3<=8};
```

傳回所有可能的頂點

```
In[2]: vertex=Flatten[Solve[#, {x1,x2,x3}]&/@
  (Subsets[Flatten[{constrains,x1>=0,x2>=0,x3>=0}],{3}]/.
  {LessEqual|GreaterEqual->Equal}),1];
```

傳回所有可行的頂點

```
In[3]: feasiblevertex=Select[vertex,
  Times@@Boole[Flatten[{constrains,x1>=0,x2>=0,x3>=0}]/.#]==1&]
Out[3]: { $\left\{x_1 \rightarrow 0, x_2 \rightarrow \frac{3}{2}, x_3 \rightarrow \frac{3}{2}\right\}, \left\{x_1 \rightarrow \frac{3}{7}, x_2 \rightarrow 0, x_3 \rightarrow \frac{12}{7}\right\},$ 
 $\{x_1 \rightarrow 0, x_2 \rightarrow 6, x_3 \rightarrow 0\}, \{x_1 \rightarrow 3, x_2 \rightarrow 0, x_3 \rightarrow 0\},$ 
 $\left\{x_1 \rightarrow 0, x_2 \rightarrow 0, x_3 \rightarrow \frac{3}{2}\right\}, \{x_1 \rightarrow 0, x_2 \rightarrow 0, x_3 \rightarrow 0\}\}$ 
```

比較所有可行的頂點傳回最佳解

```
In[4]: Sort[{obj/.#, #]&@feasiblevertex][[-1]]
Out[4]: {12,{x1 → 0, x2 → 6, x3 → 0}}
```

由於 LinearProgramming 所求解之問題必須為極小化問題，所以將目標函數及限制式轉換。

定義目標函數係數

```
In[5]: objcoef=Coefficient[-obj,#]&/@{x1,x2,x3}
Out[5]: {-3,-2,-1}
```

定義限制式左方係數

```
In[6]: concoef=Table[-Coefficient[constrains[[z,1]],#]&/@{x1,x2,x3},
{z,Length@constrains}]
Out[6]: {{-1,1,-1},{-2,-1,-3},{1,0,-2},{-1,-1,-1}}
```

定義限制式右方係數

```
In[7]: bcoef=-constrains[[All,2]]
Out[7]: {-4,-6,-3,-8}
```

使用 LinearProgramming 求解

```
In[8]: LinearProgramming[objcoef, concoef, bcoef]
```

```
Out[8]: {0, 6, 0}
```

計算目標函數之最佳解

```
In[9]: -objcoef.%
```

```
Out[9]: 12
```

使用 Maximize 求解

```
In[10]: Maximize[Flatten@{obj, constrains, x1>=0, x2>=0, x3>=0}, {x1, x2, x3}]
```

```
Out[10]: {12, {x1 → 0, x2 → 6, x3 → 0}}
```

使用 FindMaximum 求解

```
In[11]: FindMaximum[Flatten@{obj, constrains, x1>=0, x2>=0, x3>=0}, {x1, x2, x3}]
```

```
Out[11]: {12., {x1 → 0., x2 → 6., x3 → 0.}}
```

利用 mySimplex 函數求解

```
In[12]: mySimplex@@LPNormalize[obj, constrains, {x1, x2, x3}]
```

```
Out[12]: {12, {x1 → 0, x2 → 6, x3 → 0}}
```

傳回單形法各階段的基變數

```
In[13]: $mySimplexIteration[[A11, {2, 6}]]
```

```
Out[13]: {{{s11, x1, s13, s14}, {1, 3, 6, 5}}, {{s11, x2, s13, s14}, {10, 6, 3, 2}}}
```

傳回單形法各階段變數的解

```
In[14]: $myVariableIteration
```

```
Out[14]: {{0, 0, 0}, {3, 0, 0}, {0, 6, 0}}
```

傳回單形法各階段的樞紐計算表

In[15]: Column@\$mySimplexIterationTable

Out[15]:

	x1	x2	x3	s11	s12	s13	s14	Solution
c-z	-3	-2	-1	0	0	0	0	0
s11	1	-1	1	1	0	0	0	4
s12	2	1	3	0	1	0	0	6
s13	-1	0	2	0	0	1	0	3
s14	1	1	1	0	0	0	1	8

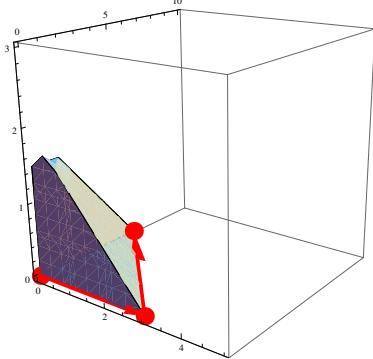
	x1	x2	x3	s11	s12	s13	s14	Solution
c-z	0	$-\frac{1}{2}$	$\frac{7}{2}$	0	$\frac{3}{2}$	0	0	9
s11	0	$-\frac{3}{2}$	$-\frac{1}{2}$	1	$-\frac{1}{2}$	0	0	1
x1	1	$\frac{1}{2}$	$\frac{3}{2}$	0	$\frac{1}{2}$	0	0	3
s13	0	$\frac{1}{2}$	$\frac{7}{2}$	0	$\frac{1}{2}$	1	0	6
s14	0	$\frac{1}{2}$	$-\frac{1}{2}$	0	$-\frac{1}{2}$	0	1	5

	x1	x2	x3	s11	s12	s13	s14	Solution
c-z	1	0	5	0	2	0	0	12
s11	3	0	4	1	1	0	0	10
x2	2	1	3	0	1	0	0	6
s13	-1	0	2	0	0	1	0	3
s14	-1	0	-2	0	-1	0	1	2

輸出 Simplex method 的求解路徑

```
In[16]: Show[RegionPlot3D[And@@constraints,
{x1,0,5},{x2,0,10},{x3,0,3},PlotStyle->Opacity[0.5],Mesh->None],
Graphics3D[{Red,PointSize[0.05],Point[$myVariableIteration],
Thickness[0.01],Arrow/@Partition[$myVariableIteration,2,1]}]]
```

Out[16]:



範例 3-3.

$$\min \quad 4x_1 + 5x_2$$

subject to

$$3x_1 + x_2 \leq 27$$

$$x_1 + x_2 = 12$$

$$3x_1 + 2x_2 \geq 30$$

$$x_i \geq 0, i = 1, 2$$

定義目標函數及限制式

```
In[1]: obj=4x1+5x2;
constraints={3x1+x2<=27,x1+x2==12,3x1+2x2>=30};
```

定義目標函數係數

```
In[2]: objcoef=Coefficient[obj,#]&/@{x1,x2}
```

Out[2]: {4, 5}

此例中由於限制式的符號並非同為 \leq ，故不需做符號的轉換。

定義限制式左方係數

```
In[3]: concoef=Table[Coefficient[constraints[[z,1]],#]&/@{x1,x2},
{z,Length@constraints}]
```

Out[3]: {{3, 1}, {1, 1}, {3, 2}}

定義限制式右方係數

In[4]: `bcoef=constraints[[A11,2]]`

Out[4]: {27, 12, 30}

使用 LinearProgramming 求解

In[5]: `LinearProgramming[objcoef,concoef,`

`{bcoef[[1]],-1},{bcoef[[2]],0},{bcoef[[3]],1}}]`

Out[5]: $\left\{ \frac{15}{2}, \frac{9}{2} \right\}$

計算目標函數之最佳解

In[6]: `-objcoef.%`

Out[6]: $\frac{105}{2}$

利用 mySimplex 函數求解

In[7]: `mySimplex@@LPNormalize[-obj,constraints,{x1,x2}]`

Out[7]: $\left\{ -\frac{105}{2}, \left\{ x1 \rightarrow \frac{15}{2}, x2 \rightarrow \frac{9}{2} \right\} \right\}$

傳回單形法迭代計算步驟

In[8]: `$myVariableIteration`

Out[8]: $\left\{ \{0, 0\}, \{9, 0\}, \{8, 3\}, \left\{ \frac{15}{2}, \frac{9}{2} \right\} \right\}$

傳回單形法初始及最後階段的樞紐計算表

In[9]: `$mySimplexIterationTable[{{1,-1}}]`

Out[9]:

	x1	x2	sul	s11	R1	R2	Solution
c-z	4 - 4 BM	5 - 3 BM	BM	0	0	0	- 42 BM
s11	3	1	0	1	0	0	27
R1	1	1	0	0	1	0	12
R2	3	2	-1	0	0	1	30

	x1	x2	sul	s11	R1	R2	Solution
c-z	0	0	0	$\frac{1}{2}$	$-\frac{11}{2} + \text{BM}$	BM	$-\frac{105}{2}$
x1	1	0	0	$\frac{1}{2}$	$-\frac{1}{2}$	0	$\frac{15}{2}$
sul	0	0	1	$\frac{1}{2}$	$\frac{3}{2}$	-1	$\frac{3}{2}$
x2	0	1	0	$-\frac{1}{2}$	$\frac{3}{2}$	0	$\frac{9}{2}$

以上輸出之單形法初始階段樞紐計算表包含 BM 的原因為使用大 M 法，故轉換為標準形式後包含人工變數 R₁ 和 R₂ 所導致。

範例 3-4. 利用範例4-6的資料，利用最小絕對值法以 $y = a_1 + a_2x + a_3x^2$ 擬合資料。

輸入資料

```
In[1]: data={{4,2},{4,10},{7,4},{7,22},{8,16},{9,10},{10,18},{10,26},{10,34},{11,17},
{11,28},{12,14},{12,20},{12,24},{12,28},{13,26},{13,34},{13,34},{13,46},
{14,26},{14,36},{14,60},{14,80},{15,20},{15,26},{15,54},{16,32},{16,40},
{17,32},{17,40},{17,50},{18,42},{18,56},{18,76},{18,84},{19,36},{19,46},
{19,68},{20,32},{20,48},{20,52},{20,56},{20,64},{22,66},{23,54},{24,70},
{24,92},{24,93},{24,120},{25,85}};
```

假設 $z_i \geq |a_1 + a_2x_i + a_3x_i^2 - y_i|$ ，則以最小絕對值法估計 a_1 , a_2 及 a_3 可以線性規劃求解下列模式：

$$\min \sum_{i=1}^n z_i$$

subject to

$$z_i \geq a_1 + a_2x_i + a_3x_i^2 - y_i, i = 1, 2, \dots, 50$$

$$z_i \geq -a_1 - a_2x_i - a_3x_i^2 + y_i, i = 1, 2, \dots, 50$$

$$z_i \geq 0, i = 1, 2, \dots, n$$

$$a_1, a_2, a_3 \text{ 不限正負}$$

由於線性規劃假設所有變數皆為非負數，上述限制式可改寫為

$$\begin{aligned} z_i &\geq (a_{11} - a_{12}) + (a_{21} - a_{22})x_i + (a_{31} - a_{32})x_i^2 - y_i, i = 1, 2, \dots, 50 \\ z_i &\geq -(a_{11} - a_{12}) - (a_{21} - a_{22})x_i - (a_{31} - a_{32})x_i^2 + y_i, i = 1, 2, \dots, 50 \\ z_i &\geq 0, i = 1, 2, \dots, n \\ a_{ij} &\geq 0 \end{aligned}$$

建立變數變化規則

```
In[2]: myrule={z[i_]:=ToExpression["z"<>ToString[i]]};
```

建立決策變數矩陣

```
In[3]: vars1=Flatten@{a11,a12,a21,a22,a31,a32,z[#]&/@Range[Length@data]}/.myrule;
```

建立目標函數係數矩陣

```
In[4]: c1=Flatten@{{0,0,0,0,0,0},Table[1,{Length@data}]};
```

建立限制式左方係數矩陣

```
In[5]: m1=Flatten[{Coefficient[z[#]-(a11-a12)-(a21-a22)*data[[#,1]]-  
-(a31-a32)*data[[#,1]]^2/.myrule,vars1]&/@Range[Length@data],  
Coefficient[z[#]+(a11-a12)+(a21-a22)*data[[#,1]]+(a31-a32)*data[[#,1]]^2/.myrule,vars1]&/@Range[Length@data]},1];
```

建立限制式右方係數矩陣

```
In[6]: b=Flatten[{-data[[All,2]],data[[All,2]]},1];
```

利用 LinearProgramming 求解最小絕對值法估計參數

```
In[7]: myans1=LinearProgramming[c1,m1,b];
```

傳回 myans1 所求得之參數估計值

```
In[8]: myans1[[1;;6]]
```

```
Out[8]: {135/14, 0, 0, 131/280, 39/280, 0}
```

傳回最小絕對值法之參數估計值

```
In[9]: #[[1]]-#[[2]]&/@Partition[myans1[[1;;6]],2]
```

$$\text{Out[9]: } \left\{ \frac{135}{14}, -\frac{131}{280}, \frac{39}{280} \right\}$$

建立 mySimplex 函數使用的限制式

```
In[10]: constrains=ExpandAll@Flatten@{Thread[-vars1[[7;;-1]] + ((a11-a12)+(a21-a22)#[[1]]+(a31-a32)#[[1]]^2&/@data)<=data[[A11,2]]], Thread[vars1[[7;;-1]]+((a11-a12)+(a21-a22)#[[1]]+(a31-a32)#[[1]]^2&/@data)>=data[[A11,2]]]};
```

利用 mySimplex 函數求解最小絕對值法估計參數

```
In[11]: mySimplex@@LPNormalize[-Total@vars1[[7;;-1]],constrains,vars1];
```

傳回最小絕對值法之參數估計值

```
In[12]: vars1[[1;;6]]/.%[[2]]
```

$$\text{Out[12]: } \left\{ \frac{135}{14}, 0, 0, \frac{131}{280}, \frac{39}{280}, 0 \right\}$$

傳回單形法迭代計算次數

```
In[13]: $myVariableIteration//Length
```

```
Out[13]: 132
```

LinearProgramming 函數雖然預設決策變數皆為非負數，但也提供選項方便處理不限制決策變數定義域的線性規劃問題。在此例中 a_1 , a_2 及 a_3 不限正負數，故需分別建立對應的決策變數及其定義域。

建立決策變數矩陣

```
In[14]: vars2=Flatten@{a1,a2,a3, z[#]&/@Range[Length@data]/.z[i_]:>ToExpression["z"<>ToString[i]]};
```

建立目標函數係數矩陣

```
In[15]: c2=Flatten@{{0,0,0},Table[1,{Length@data}]};
```

建立限制式左方係數矩陣

```
In[16]: m2=Flatten[{Coefficient[z[#]-a1-a2*data[[#,1]]-a3*data[[#,1]]]^2/.
z[i_]:=ToExpression["z"<>ToString[i]],vars2]&/@Range[Length@data],  
Coefficient[z[#]+a1+a2*data[[#,1]]+a3*data[[#,1]]]^2/.  
z[i_]:=ToExpression["z"<>ToString[i]],vars2]&/@Range[Length@data]},1];
```

建立決策變數的下界矩陣

```
In[17]: l=Flatten@{-Infinity,-Infinity,-Infinity,Table[0,{Length@data}]};
```

利用 LinearProgramming 求解最小絕對值法估計參數

```
In[18]: myans2=LinearProgramming[c2,m2,b,l];
```

傳回最小絕對值法之參數估計值

```
In[19]: myans2[[1;;6]]
```

$$\text{Out[19]: } \left\{ \frac{135}{14}, -\frac{131}{280}, \frac{39}{280} \right\}$$

利用 Minimize 求解最小絕對值法估計參數

```
In[20]: myans3=Minimize[{c2.vars2,Thread[m2.vars2>=b],Thread[vars2>=1]},vars2];
```

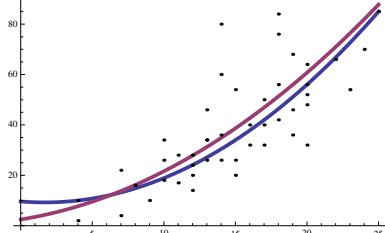
利用 Minimize 求解最小平方法估計參數

```
In[21]: myans4=Minimize[Total[((a1,a2,a3).{1,#[[1]],#[[1]]^2}-#[[2]])^2]&@  
data],{a1,a2,a3}];
```

輸出最小絕對值法與最小平方法所估計的曲線及散佈圖

```
In[22]: Plot[{{1,x,x^2}.myans2[[1;;3]],a1+a2*x+a3*x^2/.myans4[[2]]},  
{x,0,25},PlotStyle->Thickness[0.01],Epilog->Point@data]
```

Out[22]:



範例 3-5.

$$\max \frac{2x_1 + 3x_2}{x_1 + 2x_2 + 1}$$

subject to

$$x_1 + 3x_2 \leq 2$$

$$3x_1 + 2x_2 \leq 3$$

$$x_i \geq 0, i = 1, 2$$

令 $z_i = x_i * t, i = 1, 2$ 及 $(x_1 + 2x_2 + 1)t = 1$, 原始問題可改寫為

$$\max 2z_1 + 3z_2$$

subject to

$$z_1 + 3z_2 - 2t \leq 0$$

$$3z_1 + 2z_2 - 3t \leq 0$$

$$z_1 + 2z_2 + t = 1$$

$$z_1, z_2, t \geq 0$$

定義目標函數及限制式

```
In[1]: obj=2z1+3z2;
constraints={z1+3z2-2t<=0,3z1+2z2-3t<=0,z1+2z2+t==1};
```

定義目標函數係數

```
In[2]: objcoef=Coefficient[obj,#]&/@{z1,z2,t}
```

```
Out[2]: {2, 3, 0}
```

定義限制式左方係數

```
In[3]: concoef=Table[Coefficient[constraints[[y,1]],#]&/@{z1,z2,t},
{y,Length@constraints}]
```

```
Out[3]: {{1, 3, -2}, {3, 2, -3}, {1, 2, 1}}
```

定義限制式右方係數

```
In[4]: bcoef=constraints[[All,2]]
```

```
Out[4]: {0, 0, 1}
```

使用 LinearProgramming 求解

In[5]: `LinearProgramming[-objcoef, concoef, {{bcoef[[1]], -1}, {bcoef[[2]], -1}, {bcoef[[3]], 0}}]`

Out[5]: $\left\{ \frac{5}{18}, \frac{1}{6}, \frac{7}{18} \right\}$

計算目標函數之最佳解

In[6]: `-objcoef.%`

Out[6]: $-\frac{19}{18}$

使用 mySimplex 函數求解

In[7]: `mySimplex@@LPNormalize[obj, {constraints[[1;;2]], constraints[[3;;3]], {}}, {z1, z2, t}]`

Out[7]: $\left\{ \frac{19}{18}, \left\{ z1 \rightarrow \frac{5}{18}, z2 \rightarrow \frac{1}{6}, t \rightarrow \frac{7}{18} \right\} \right\}$

傳回原始變數 x_1 和 x_2 的最佳解

In[8]: `{z1/t, z2/t}/.%[[2]]`

Out[8]: $\left\{ \frac{5}{7}, \frac{3}{7} \right\}$

傳回單形法初始及最後階段的樞紐計算表

In[9]: `Column@$mySimplexIterationTable[[{1, -1}]]`

Out[9]:

	z_1	z_2	t	s_{11}	s_{12}	R_1	Solution
$c-z$	$-2 - BM$	$-3 - 2 BM$	$-BM$	0	0	0	$-BM$
s_{11}	1	3	-2	1	0	0	0
s_{12}	3	2	-3	0	1	0	0
R_1	1	2	1	0	0	1	1

	z_1	z_2	t	s_{11}	s_{12}	R_1	Solution
$c-z$	0	0	0	$\frac{1}{9}$	$\frac{5}{18}$	$\frac{19}{18} + BM$	$\frac{19}{18}$
z_2	0	1	0	$\frac{1}{3}$	$-\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$
z_1	1	0	0	$-\frac{4}{9}$	$\frac{7}{18}$	$\frac{5}{18}$	$\frac{5}{18}$
t	0	0	1	$-\frac{2}{9}$	$-\frac{1}{18}$	$\frac{7}{18}$	$\frac{7}{18}$

範例 3-6.

$$\min \{x_1 + 2x_2 + 3x_3, 2x_1 + x_2 + 3x_3, x_1 + 3x_2 + 2x_3\}$$

subject to

$$2x_1 + 3x_2 + 4x_3 \leq 12$$

$$x_i \geq 0, i = 1, 2, 3$$

假設 $z = \min \{x_1 + 2x_2 + 3x_3, 2x_1 + x_2 + 3x_3, x_1 + 3x_2 + 2x_3\}$, 故可得

$$x_1 + 2x_2 + 3x_3 \geq z$$

$$2x_1 + x_2 + 3x_3 \geq z$$

$$x_1 + 3x_2 + 2x_3 \geq z$$

由於 $x_i \geq 0$, 所以 $z \geq 0$ 。因此, 原始問題可改寫為

$$\max z$$

subject to

$$x_1 + 2x_2 + 3x_3 \geq z$$

$$2x_1 + x_2 + 3x_3 \geq z$$

$$x_1 + 3x_2 + 2x_3 \geq z$$

$$2x_1 + 3x_2 + 4x_3 \leq 12$$

$$x_i, z \geq 0, i = 1, 2, 3$$

定義目標函數及限制式

```
In[1]: obj=z;
constraints={2x1+3x2+4x3<=12,x1+2x2+3x3-z>=0,2x1+x2+3x3-z>=0,x1+3x2+2x3-z>=0};
```

定義目標函數係數

```
In[2]: objcoef=Coefficient[obj,#]&/@{x1,x2,x3,z}
Out[2]: {0,0,0,1}
```

定義限制式左方係數

```
In[3]: concoef=Table[Coefficient[constraints[[y,1]],#]&/@{x1,x2,x3,z},
{y,Length@constraints}]
```

```
Out[3]: {{2,3,4,0},{1,2,3,-1},{2,1,3,-1},{1,3,2,-1}}
```

定義限制式右方係數

In[4]: `bcoef=constraints[[A11,2]]`

Out[4]: {12, 0, 0, 0}

使用 LinearProgramming 求解

In[5]: `LinearProgramming[-objcoef,concoef,{bcoef[[1]],-1},{bcoef[[2]],-1},{bcoef[[3]],0}]`

Out[5]: $\left\{ \frac{4}{3}, \frac{4}{3}, \frac{4}{3}, 8 \right\}$

計算目標函數之最佳解

In[6]: `-objcoef.%`

Out[6]: 8

使用 mySimplex 函數求解

In[7]: `mySimplex@@LPNormalize[obj,{constraints[[1;;1]],{},constraints[[2;;4]]},{x1,x2,x3,z}]`

Out[7]: $\left\{ 8, \left\{ x_1 \rightarrow \frac{4}{3}, x_2 \rightarrow \frac{4}{3}, x_3 \rightarrow \frac{4}{3}, z \rightarrow 8, su1 \rightarrow 0, su2 \rightarrow 0, su3 \rightarrow 0 \right\} \right\}$

傳回單形法初始及最後階段的樞紐計算表

In[8]: `$mySimplexIterationTable[[{1,-1}]]`

Out[8]:

	x_1	x_2	x_3	z	s_{u1}	s_{u2}	s_{u3}	s_{l1}	R_1	R_2	R_3	Solution
$c-z$	-4 BM	-6 BM	-8 BM	$-\frac{1}{3} +$ BM	BM	BM	BM	0	0	0	0	0
s_{l1}	2	3	4	0	0	0	0	1	0	0	0	12
R_1	1	2	3	-1	-1	0	0	0	1	0	0	0
R_2	2	1	3	-1	0	-1	0	0	0	1	0	0
R_3	1	3	2	-1	0	0	-1	0	0	0	1	0
	x_1	x_2	x_3	z	s_{u1}	s_{u2}	s_{u3}	s_{l1}	R_1	R_2	R_3	Solution
$c-z$	0	0	0	0	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{2}{3}$	$-\frac{1}{3} +$ BM	$-\frac{1}{3} +$ BM	$-\frac{1}{3} +$ BM	8
z	0	0	0	1	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{2}{3}$	$-\frac{1}{3}$	$-\frac{1}{3}$	$-\frac{1}{3}$	8
x_3	0	0	1	0	$-\frac{7}{9}$	$\frac{2}{9}$	$\frac{5}{9}$	$\frac{1}{9}$	$\frac{7}{9}$	$-\frac{2}{9}$	$-\frac{5}{9}$	$\frac{4}{3}$
x_1	1	0	0	0	$\frac{11}{9}$	$-\frac{7}{9}$	$-\frac{4}{9}$	$\frac{1}{9}$	$-\frac{11}{9}$	$\frac{7}{9}$	$\frac{4}{9}$	$\frac{4}{3}$
x_2	0	1	0	0	$\frac{2}{9}$	$\frac{2}{9}$	$-\frac{4}{9}$	$\frac{1}{9}$	$-\frac{2}{9}$	$-\frac{2}{9}$	$\frac{4}{9}$	$\frac{4}{3}$

3.2 對偶理論

一個線性規劃問題可以從兩個不同的角度來分析，其一為根據每一產品的獲利情況做有效的資源分配，進而使得利潤極大化；另一為根據生產資源所願付出的成本做有效的規劃，進而使得成本極小化。「資源有限，求最高利潤」及「指定產量，求最低成本」

考慮範例 3-1，

$$\max Z = 4x_1 + 5x_2$$

subject to

$$6x_1 + 4x_2 \leq 24$$

$$x_1 + 2x_2 \leq 6$$

$$-x_1 + x_2 \leq 1$$

$$x_2 \leq 2$$

$$x_1, x_2 \geq 0$$

由於 x_1 和 x_2 皆為非負數，利用限制式的線性組合可得以下結果：

$$Z \leq \text{限制式 } 1 \times \frac{5}{4} = \frac{30}{4}x_1 + 5x_2 \leq 30$$

$$Z \leq \text{限制式 } 1 \times \frac{2}{3} + \text{限制式 } 2 + \text{限制式 } 3 = 4x_1 + \frac{17}{3}x_2 \leq 23$$

$$Z \leq \text{限制式 } 1 \times \frac{1}{2} + \text{限制式 } 2 + \text{限制式 } 3 + \text{限制式 } 4 = 4x_1 + 6x_2 \leq 21$$

由以上簡單的運算可以發現原始問題上界逐漸縮小。因此，我們可以藉由限制式組成適當的線性組合， $y_1b_1 + y_2b_2 + y_3b_3 + y_4b_4$ ，以求得原始問題中任一可行解之上限。又由於 $\mathbf{Ax} \leq \mathbf{b}$ ，可得

$$\begin{aligned} W(y_1, y_2, y_3, y_4) &\equiv y_1b_1 + y_2b_2 + y_3b_3 + y_4b_4 \\ &\geq y_1(6x_1 + 4x_2) + y_2(x_1 + 2x_2) + y_3(-x_1 + x_2) + y_4(x_2) \\ &= x_1(6y_1 + y_2 - y_3) + x_2(4y_1 + 2y_2 + y_3 + y_4) \end{aligned}$$

不難發現上式成立的條件為 $6y_1 + y_2 - y_3 \geq 4$ 及 $4y_1 + 2y_2 + y_3 + y_4 \geq 5$ 。故原始問題可改寫成下列對偶問題：

$$\min W = 24y_1 + 6y_2 + y_3 + 3y_4$$

subject to

$$\begin{aligned} 6y_1 + y_2 - y_3 &\geq 4 \\ 4y_1 + 2y_2 + y_3 + y_4 &\geq 5 \\ y_i &\geq 0, i = 1, 2, 3, 4 \end{aligned}$$

根據以上說明可知，對於一個線性規劃問題（Primal problem，記為 \mathbf{P} ）

$$\max c_1x_1 + c_2x_2 + \cdots + c_nx_n$$

subject to

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &\leq b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &\leq b_2 \\ a_{31}x_1 + a_{32}x_2 + \cdots + a_{3n}x_n &\leq b_3 \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &\leq b_m \\ x_1, x_2, \dots, x_n &\geq 0 \end{aligned}$$

必存在一個對偶問題（Dual problem，記為 \mathbf{D} ）如下：

$$\min b_1y_1 + b_2y_2 + \cdots + b_my_m \quad (3-4)$$

subject to

$$\begin{aligned}
 a_{11}y_1 + a_{21}y_2 + \cdots + a_{m1}y_m &\geq c_1 \\
 a_{12}y_1 + a_{22}y_2 + \cdots + a_{m2}y_m &\geq c_2 \\
 a_{13}y_1 + a_{23}y_2 + \cdots + a_{m3}y_m &\geq c_3 \\
 &\vdots \\
 a_{1n}x_1 + a_{2n}x_2 + \cdots + a_{mn}y_m &\geq c_n \\
 y_1, y_2, \dots, y_m &\geq 0
 \end{aligned} \tag{3-5}$$

可以矩陣表示如下：

原始問題 (P)	對偶問題 (D)
$\max \quad \mathbf{c}^\top \mathbf{x}$	$\min \quad \mathbf{b}^\top \mathbf{y}$
subject to	subject to
$\mathbf{A}\mathbf{x} \leq \mathbf{b}$	$\mathbf{A}^\top \mathbf{y} \geq \mathbf{c}$
$\mathbf{x} \geq \mathbf{0}$	$\mathbf{y} \geq \mathbf{0}$

此外，我們也可發現原始問題與對偶問題間有以下重要關係：

1. 對稱性 (Symmetric duality)：對偶問題的對偶問題即為原始問題。
2. 弱對偶定理 (Weak Duality theorem)：若 \mathbf{x} 及 \mathbf{y} 分別為 **P** 與 **D** 的可行解，則 $\mathbf{c}^\top \mathbf{x} \leq \mathbf{y}^\top \mathbf{A}\mathbf{x} = \mathbf{b}^\top \mathbf{y}$ ，即 $Z(\mathbf{x}) \leq W(\mathbf{y})$ 。由此可推得若 **P** 的最佳解為無限大，則 **D** 無可行解；反之，若 **P** 無可行解，則 **D** 的最佳解為無限大。
3. 強對偶定理 (Strong Duality theorem)：若 \mathbf{x}^* 及 \mathbf{y}^* 分別為 **P** 與 **D** 的最佳解，則 $\mathbf{c}^\top \mathbf{x}^* = \mathbf{b}^\top \mathbf{y}^*$ ，即 $Z(\mathbf{x}^*) = W(\mathbf{y}^*)$ 。
4. 互補差餘定理 (Complementary slackness theorem)：假設 \mathbf{u} 和 \mathbf{v} 分別為 **P** 與 **D** 所對應的鬆弛變數與差額變數，故 $\mathbf{A}\mathbf{x} + \mathbf{u} = \mathbf{b}$ 及 $\mathbf{A}^\top \mathbf{y} - \mathbf{v} = \mathbf{c}$ ，其中 $\mathbf{u} \geq \mathbf{0}$ 及 $\mathbf{v} \geq \mathbf{0}$ 。由強對偶定理可知，若 \mathbf{x}^* 和 \mathbf{y}^* 分別為原始問題 **P** 與對偶問題 **D** 的最佳解，則 $\mathbf{c}^\top \mathbf{x}^* = \mathbf{b}^\top \mathbf{y}^*$ 。故可推得 $\mathbf{y}^{*\top} (\mathbf{A}\mathbf{x}^* + \mathbf{u}) - \mathbf{x}^{*\top} (\mathbf{A}^\top \mathbf{y}^* - \mathbf{v}) = \mathbf{u}^\top \mathbf{y}^* + \mathbf{v}^\top \mathbf{x}^* = 0$ 。又 $\mathbf{y} \geq \mathbf{0}$ 且 $\mathbf{x} \geq \mathbf{0}$ ，故可得上式成立的必要條件為 $\mathbf{u}^\top \mathbf{y}^* = 0$ 且 $\mathbf{v}^\top \mathbf{x}^* = 0$ 。

範例 3-7. 承範例 3-1，試建立對偶問題線性規劃模式及求解。

定義目標函數及限制式

```
In[1]: obj=4x1+5x2;
constraints={6x1+4x2<=24,x1+2x2<=6,-x1+x2<=1,x2<=2};
```

使用 LinearProgramming 求解

In[2]: `LinearProgramming[-{4, 5}, -{{6, 4}, {1, 2}, {-1, 1}, {0, 1}}, -{24, 6, 1, 2}]`

Out[2]: $\left\{3, \frac{3}{2}\right\}$

使用 mySimplex 函數求解，並將單形法各階段的樞紐計算表輸出為 table1

In[3]: `mySimplex@@LPNormalize[obj, constraints, {x1, x2}]`

`table1=$mySimplexIterationTable;`

Out[3]: $\left\{\frac{39}{2}, \left\{x_1 \rightarrow 3, x_2 \rightarrow \frac{3}{2}\right\}\right\}$

接下來我們利用 myDualProblem 來輸出對偶問題的線性規劃模型。myDualProblem 函數的輸出包含兩個部分，第一部分為對偶問題線性規劃模型，第二部分輸出為 mySimplex 函數所需參數。

傳回對偶問題線性規劃模型

In[4]: `Column@Flatten@myDualProblem[obj, constraints, {x1, x2}][[1, {1, 2}]]`

Out[4]:
 $24y_1 + 6y_2 + y_3 + 2y_4$
 $6y_1 + y_2 - y_3 \geq 4$
 $4y_1 + 2y_2 + y_3 + y_4 \geq 5$
 $y_1 \geq 0$
 $y_2 \geq 0$
 $y_3 \geq 0$
 $y_4 \geq 0$

對偶問題的對偶問題即為原始問題

In[5]: `Column@Flatten@(myDualProblem@@myDualProblem[obj, constraints, {x1, x2}][[2]])`
`[[2, {1, 2}]] /. {y1->x1, y2->x2}`

Out[5]:
 $-4x_1 - 5x_2$
 $6x_1 + 4x_2 \leq 24$
 $x_1 + 2x_2 \leq 6$
 $-x_1 + x_2 \leq 1$
 $x_2 \leq 2$

使用 DualLinearProgramming 函數求解

In[6]: `DualLinearProgramming[-{4, 5}, -{{6, 4}, {1, 2}, {-1, 1}, {0, 1}}, -{24, 6, 1, 2}]`

Out[6]: $\left\{\left\{3, \frac{3}{2}\right\}, \left\{\frac{3}{8}, \frac{7}{4}, 0, 0\right\}, \{0, 0\}, \{0, 0\}\right\}$

In[7]: DualLinearProgramming[-{4,5},{{6,4},{1,2},{-1,1},{0,1}},
 {{24,-1},{6,-1},{1,-1},{2,-1}}]

Out[7]: $\left\{\left\{3, \frac{3}{2}\right\}, \left\{-\frac{3}{8}, -\frac{7}{4}, 0, 0\right\}, \{0, 0\}, \{0, 0\}\right\}$

使用 mySimplex 函數求解對偶問題，並將單形法各階段的樞紐計算表輸出為 table2

In[8]: mySimplex@@LPNormalize@@myDualProblem[obj,constraints,{x1,x2}][[2]]
 table2=\$mySimplexIterationTable;

Out[8]: $\left\{-\frac{39}{2}, \left\{y1 \rightarrow \frac{3}{8}, y2 \rightarrow \frac{7}{4}, y3 \rightarrow 0, y4 \rightarrow 0\right\}\right\}$

輸出原始問題及對偶問題單形法最後階段的樞紐計算表

In[9]: Column@{table1[[-1]],table2[[-1]]}

Out[9]:

	x1	x2	s11	s12	s13	s14	Solution
c-z	0	0	$\frac{3}{8}$	$\frac{7}{4}$	0	0	$\frac{39}{2}$
s14	0	0	$\frac{1}{8}$	$-\frac{3}{4}$	0	1	$\frac{1}{2}$
s13	0	0	$\frac{3}{8}$	$-\frac{5}{4}$	1	0	$\frac{5}{2}$
x2	0	1	$-\frac{1}{8}$	$\frac{3}{4}$	0	0	$\frac{3}{2}$
x1	1	0	$\frac{1}{4}$	$-\frac{1}{2}$	0	0	3

	y1	y2	y3	y4	s11	s12	R1	R2	Solution
c-z	0	0	$\frac{5}{2}$	$\frac{1}{2}$	3	$\frac{3}{2}$	$-3 + BM$	$-\frac{3}{2} + BM$	$-\frac{39}{2}$
y1	1	0	$-\frac{3}{8}$	$-\frac{1}{8}$	$-\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{4}$	$-\frac{1}{8}$	$\frac{3}{8}$
y2	0	1	$\frac{5}{4}$	$\frac{3}{4}$	$\frac{1}{2}$	$-\frac{3}{4}$	$-\frac{1}{2}$	$\frac{3}{4}$	$\frac{7}{4}$

In[10]: constraints[[All,2]]-constraints[[All,1]]/.{x1->3,x2->3/2}

Out[10]: $\left\{0, 0, \frac{5}{2}, \frac{1}{2}\right\}$

```
In[11]: sen[i_,delta_]:=Block[{temp},temp=constraints;
temp[[All,2]]=temp[[All,2]]+Table[If[j==i,delta,0],{j,Length@temp}];
mySimplex@@LPNormalize[obj,temp,{x1,x2}]]
```

```
In[12]: sen[#,1/10][[1]]-primal[[1]]&/@Range[Length@constraints]
Out[12]: {3/80, 7/40, 0, 0}
```

範例 3-8.

```
In[1]: obj=4x1+5x2;
constraints={3x1+x2<=27,x1+x2==12,3x1+2x2>=30};
```

```
In[2]: primal=mySimplex@@LPNormalize[-obj,constraints,{x1,x2}]
table1=$mySimplexIterationTable;
Out[2]: {-105/2,{x1 -> 15/2,x2 -> 9/2}}
```

```
In[3]: dualproblem=myDualProblem[-obj,constraints,{x1,x2}];
```

```
In[4]: Column@Flatten@dualproblem[[1,{1,2}]]
```

```
Out[4]: 27y1 + 12y2 - 30y3
3y1 + y2 - 3y3 ≥ -4
y1 + y2 - 2y3 ≥ -5
y1 ≥ 0
y3 ≥ 0
```

```
In[5]: Column@Flatten@dualproblem[[2,{1,2}]]
```

Out[5]:

$$\begin{aligned} -27y_1 - 12y_{21} + 12y_{22} - 30y_3 \\ -3y_1 - y_{21} + y_{22} + 3y_3 \leq 4 \\ -y_1 - y_{21} + y_{22} + 2y_3 \leq 5 \end{aligned}$$

In[6]: DualLinearProgramming[{4,5},{{3,1},{1,1},{3,2}},{{27,-1},{12,0},{30,1}}]

Out[6]: $\left\{ \left\{ \frac{15}{2}, \frac{9}{2} \right\}, \left\{ -\frac{1}{2}, \frac{11}{2}, 0 \right\}, \{0, 0\}, \{0, 0\} \right\}$

In[7]: Minimize@@dualproblem[[1]]

Out[7]: $\left\{ -\frac{105}{2}, \left\{ y_1 \rightarrow \frac{1}{2}, y_2 \rightarrow -\frac{11}{2}, y_3 \rightarrow 0 \right\} \right\}$

In[8]: dual=mySimplex@@LPNormalize@@myDualProblem[-obj,constraints,{x1,x2}][[2]]

table2=\$mySimplexIterationTable;

Out[8]: $\left\{ \frac{105}{2}, \left\{ y_1 \rightarrow \frac{1}{2}, y_{21} \rightarrow 0, y_{22} \rightarrow \frac{11}{2}, y_3 \rightarrow 0 \right\} \right\}$

輸出原始問題及對偶問題單形法最後階段的樞紐計算表

In[9]: Column@{table1[[-1]],table2[[-1]]}

Out[9]:

	x1	x2	s11	s11	R1	R2	Solution
c-z	0	0	0	$\frac{1}{2}$	$-\frac{11}{2} + BM$	BM	$-\frac{105}{2}$
x1	1	0	0	$\frac{1}{2}$	$-\frac{1}{2}$	0	$\frac{15}{2}$
s11	0	0	1	$\frac{1}{2}$	$\frac{3}{2}$	-1	$\frac{3}{2}$
x2	0	1	0	$-\frac{1}{2}$	$\frac{3}{2}$	0	$\frac{9}{2}$

	y1	y21	y22	y3	s11	s12	Solution
c-z	0	0	0	$\frac{123}{2}$	$\frac{15}{2}$	$\frac{9}{2}$	$\frac{105}{2}$
y22	0	-1	1	$\frac{3}{2}$	$-\frac{1}{2}$	$\frac{3}{2}$	$\frac{11}{2}$
y1	1	0	0	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$

```
In[10]: sen[i_,delta_]:=Block[{temp},temp=constraints;
temp[[All,2]]=temp[[All,2]]+Table[If[j==i,delta,0],{j,Length@temp}];
mySimplex@@LPNormalize[-obj,temp,{x1,x2}]]
```

```
In[11]: -sen[#,1/10][[1]]-(-primal[[1]])&/@Range[Length@constraints]
Out[11]: ⎧-1/20, 11/20, 0⎫
```

第 4 章 非線性規劃

非線性規劃是在研究一個目標函數在多組等式或不等式的限制條件下的最佳化問題。非線性規劃的應用十分廣泛，其模式主要特徵為目標函數或限制條件中至少有一個是非線性函數。在 Mathematica 中進行最佳化求解有很多方式，大致可分為兩種：間接搜尋法與直接搜尋法。間接搜尋法必須使用到微積分做為基礎，依據函數的梯度設定起始值的搜尋方向，並以此方向找出極值所在的位置。而直接搜尋法則不需要使用到任何的微積分，僅需以函數值做運算即可。Mathematica 在以梯度為基礎的所提供的最佳化指令有 `Minimize`、`Maximize`、`FindMinimize` 和 `FindMaximum`；而直接搜尋法所提供的指令有 `NMinimize` 和 `NMaximize`。若和線性規劃問題所使用的單純法（Simplex method）和內點法（Interior-point algorithm）相比，非線性規劃並沒有適用於各種問題的一般演算法，各個方法都有自己特定的適用範圍。在本章中，我們僅針對 Mathematica 中以梯度為基礎之相關迭代演算法函數說明。

4.1 無限制條件下非線性規劃

假設 $\mathbf{x} = \{x_1, x_2, \dots, x_n\}$ ，考慮下列問題

$$\min_{\mathbf{x}} f(\mathbf{x}), \mathbf{x} \in \mathbb{R}^n,$$

其中 f 為一連續可微分函數。在微積分中我們知道，若存在一個點 $\mathbf{x}_0 = (x_1, x_2, \dots, x_n)$ 使得目標函數 $f(\mathbf{x}_0)$ 有局部極小值，則表示對於所有的 $\mathbf{h} (\neq \mathbf{0})$ 皆使得

$$f(\mathbf{x}_0 + \mathbf{h}) \leq f(\mathbf{x})$$

相對的，若存在一個點 $\mathbf{x}_0 = (x_1, x_2, \dots, x_n)$ 使得目標函數 $f(\mathbf{x})$ 有局部極小值，則表示對於所有的 $\mathbf{h} (\neq \mathbf{0})$ 皆使得

$$f(\mathbf{x}_0 + \mathbf{h}) \geq f(\mathbf{x})$$

此外，若極值存在，必發生在

$$\nabla f(\mathbf{x}) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \dots, \frac{\partial f}{\partial x_n} \right) = \mathbf{0} \quad (4-1)$$

然而如何判斷極值為局部極大或是局部極小？關於這一點則必須藉由判斷海賽矩陣（Hessian Matrix），的正負定來決定，其中

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix} \quad (4-2)$$

對於任意非 $\mathbf{0}$ 向量 \mathbf{x} ，若 $\mathbf{x}^\top \mathbf{H} \mathbf{x} > 0$ ，則我們稱 \mathbf{H} 為一正定矩陣；相反地，若 $\mathbf{x}^\top \mathbf{H} \mathbf{x} < 0$ ，則我們稱 \mathbf{H} 為一負定矩陣。由泰勒展開式可知，

$$f(\mathbf{x}_0 + \Delta \mathbf{x}) \approx f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^\top \Delta \mathbf{x} + \frac{1}{2} \Delta \mathbf{x}^\top \mathbf{H} \Delta \mathbf{x} \quad (4-3)$$

若極值存在，則 $\nabla f(\mathbf{x}_0) = \mathbf{0}$ ，故可推得

$$f(\mathbf{x}_0 + \Delta \mathbf{x}) - f(\mathbf{x}_0) = \frac{1}{2} \Delta \mathbf{x}^\top \mathbf{H} \Delta \mathbf{x} \quad (4-4)$$

由上式不難發現若 \mathbf{H} 為一正定矩陣，則 $f(\mathbf{x}_0 + \Delta \mathbf{x}) - f(\mathbf{x}_0) > 0$ ，故得 $f(\mathbf{x}_0)$ 為局部極小值；相對的，若 \mathbf{H} 為一負定矩陣，則 $f(\mathbf{x}_0 + \Delta \mathbf{x}) - f(\mathbf{x}_0) < 0$ ，故得 $f(\mathbf{x}_0)$ 為局部極大值。

在 Mathematica 中，當 $\nabla f(\mathbf{x}) = \mathbf{0}$ 有解析解時，我們可以用 Solve、Reduce 或 Root 等函數來求得臨界點的解析解；反之，若 $\nabla f(\mathbf{x}) = \mathbf{0}$ 只有數值解，則必須使用 FindRoot 來求得臨界點。然而當目標函數是包含多變數時，求解聯立方程組 $\nabla f(\mathbf{x}) = \mathbf{0}$ 並不是一件容易的事，所以我們需要一些有效率的演算法來幫助求解。Mathematica 在非線性規劃中所使用的演算法有 Newton、Quasi Newton、Conjugate Gradient、LevenbergMarquardt 及 PrincipalAxis，分別介紹如下：

4.1.1 梯度下降法或最陡坡降法（Gradient descent or Steepest descent method）

由一階泰勒展開式可知

$$f(\mathbf{x} + \Delta \mathbf{x}) \approx f(\mathbf{x}) + (\nabla f(\mathbf{x}))^\top \Delta \mathbf{x} \quad (4-5)$$

假設 $\|\Delta \mathbf{x}\| = 1$ ，則

$$-1 \leq \cos \theta = \frac{\nabla f(\mathbf{x})^\top \Delta \mathbf{x}}{\|\nabla f(\mathbf{x})\| \|\Delta \mathbf{x}\|} = \frac{\nabla f(\mathbf{x})^\top \Delta \mathbf{x}}{\|\nabla f(\mathbf{x})\|} \leq 1$$

移項化簡得 $\nabla f(\mathbf{x})^\top \Delta \mathbf{x} = \cos \theta \times \|\nabla f(\mathbf{x})\|$ 。因此當梯度向量與搜尋方向夾角 $\theta = \pi$ 時有最小值，即 $\Delta \mathbf{x} = -\nabla f(\mathbf{x})$ 。由於第 $k+1$ 階段搜尋方向 $\Delta \mathbf{x} = \mathbf{x}_{k+1} - \mathbf{x}_k$ ，其中 \mathbf{x}_k 為第 k 階段的迭代值，故可推得

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \nabla f(\mathbf{x}_k) \quad (4-6)$$

此法我們所選擇的搜尋方向為在單位距離內下降最快的方向，故稱為梯度下降法或最速下降法。由於梯度下降法所採用的為線性函數，因此當起始值選擇離最佳解較遠時收斂速度比較慢，很難到達真正的局部極小值。

4.1.2 牛頓法 (Newton method)

由二階泰勒展開式可知

$$f(\mathbf{x} + \Delta\mathbf{x}) \approx f(\mathbf{x}) + (\nabla f(\mathbf{x}))^\top \Delta\mathbf{x} + \frac{1}{2} \Delta\mathbf{x}^\top \mathbf{H} \Delta\mathbf{x} \quad (4-7)$$

欲求 $f(\mathbf{x})$ 極值，必要條件為

$$\frac{\partial f(\mathbf{x} + \Delta\mathbf{x})}{\partial \Delta\mathbf{x}} \approx \nabla f(\mathbf{x}) + \mathbf{H} \Delta\mathbf{x} = 0 \quad (4-8)$$

由於第 $k + 1$ 階段搜尋方向 $\Delta\mathbf{x} = \mathbf{x}_{k+1} - \mathbf{x}_k$ ，其中 \mathbf{x}_k 為第 k 階段的迭代值，故可推得

$$\mathbf{x}_{k+1} - \mathbf{x}_k = \Delta\mathbf{x}_k = -\mathbf{H}_k^{-1} \nabla f(\mathbf{x}_k)$$

所以第 $k + 1$ 階段的迭代值可改寫為

$$\mathbf{x}_{k+1} = \mathbf{x}_k - 1 \times \mathbf{H}_k^{-1} \nabla f(\mathbf{x}_k) \quad (4-9)$$

以上的搜尋方式稱為牛頓法 (Newton's Method)， $-\mathbf{H}^{-1} \nabla f(\mathbf{x}_k)$ 則稱為牛頓方向。在這個推導中步長 1，但實際演算法進行時容易造成發散。為避免發生上述情況，我們以 α_k 取代 1 並求得使得 $f(\mathbf{x})$ 有極大或極小的 α_k 值作為下一階段的迭代值，即

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{H}_k^{-1} \nabla f(\mathbf{x}_k) \quad (4-10)$$

這種作法稱為修正牛頓法 (Modified Newton's Method)。由 (4-10) 可知，當 $\mathbf{H} = \mathbf{I}$ 時，

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k) \quad (4-11)$$

因此 \mathbf{x}_{k+1} 會朝梯度向量的反方向前進，此時牛頓法即為梯度下降法。

由以上的介紹我們可以發現當 \mathbf{x}_k 已知時， $\Delta\mathbf{x}_k = -\mathbf{H}_k^{-1} \nabla f(\mathbf{x}_k)$ 亦為已知數，故 $f(\mathbf{x}_{k+1}) = f(\mathbf{x}_k + \alpha_k \Delta\mathbf{x}_k)$ 為 α_k 的一元方程式。因此，我們可以很輕易的求得使得 $f(\mathbf{x}_{k+1})$ 最小的 α_k 值，即求

$$\frac{df(\mathbf{x}_k + \alpha_k \Delta\mathbf{x}_k)}{d\alpha_k} = \nabla f(\mathbf{x}_k + \alpha_k \Delta\mathbf{x}_k)^\top \Delta\mathbf{x}_k = 0 \quad (4-12)$$

上式可發現第 k 階段的搜尋方向會與下一階段的梯度方向 $\nabla f(\mathbf{x}_k + \alpha_k \Delta\mathbf{x}_k)$ 垂直。由於搜尋方向是沿著直線 $\mathbf{x}_k + \alpha_k \Delta\mathbf{x}_k$ 進行搜索，故此類求解方法也稱為直線搜尋法 (Line search Methods)。

4.1.3 擬牛頓法 (Quasi-Newton method)

牛頓法在收斂的速度上雖然快速，但由 (4-9) 及 (4-10) 可發現牛頓法在每次迭代計算都必須重新計算海賽矩陣。然而海賽矩陣並非在每一次的迭代計算都會存在，如果海賽矩陣是奇異矩陣 (singular matrix) 或狀況不佳的矩陣 (ill-conditioned matrix)，求反矩陣時會產生數值問題造成演算法失效。為解決此一困擾，我們可使用前幾次迭代的梯度去建構一個近似於海賽矩陣之反矩陣，此一方法稱為準牛頓法 (Quasi-Newton's Method)。

由泰勒展開式可知梯度函數在第 k 階段及第 $k + 1$ 階段的一階泰勒展開式可寫為

$$\nabla f(\mathbf{x}_{k+1}) \approx \nabla f(\mathbf{x}_0) + \mathbf{H}_{k+1}(\mathbf{x}_{k+1} - \mathbf{x}_0)$$

和

$$\nabla f(\mathbf{x}_k) \approx \nabla f(\mathbf{x}_0) + \mathbf{H}_k(\mathbf{x}_k - \mathbf{x}_0)$$

由前兩式可得

$$\nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k) \approx \mathbf{H}_{k+1}(\mathbf{x}_{k+1} - \mathbf{x}_k) = \mathbf{H}_{k+1}\Delta\mathbf{x}_k \quad (4-13)$$

故可推得

$$\Delta\mathbf{x}_k = \mathbf{H}_{k+1}^{-1}(\nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k)) = \mathbf{B}_{k+1}\mathbf{y}_k \quad (4-14)$$

其中 $\mathbf{B}_{k+1} = \mathbf{H}_{k+1}^{-1}$ 及 $\mathbf{y}_k = \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k)$ 。我們稱 (4-14) 為 Quasi-Newton condition。假設第 $k + 1$ 階段海賽矩陣的反矩陣 \mathbf{B}_{k+1} 可藉由第 k 階段的資訊所求得，則 \mathbf{B}_{k+1} 可寫為

$$\mathbf{B}_{k+1} = \mathbf{B}_k + \Delta\mathbf{B}_k = \mathbf{B}_k + c_1\mathbf{z}_1\mathbf{z}_1^T + c_2\mathbf{z}_2\mathbf{z}_2^T \quad (4-15)$$

由 Quasi-Newton condition 可知

$$\begin{aligned} \Delta\mathbf{x}_k &= \mathbf{B}_{k+1}\mathbf{y}_k \\ &= (\mathbf{B}_k + c_1\mathbf{z}_1\mathbf{z}_1^T + c_2\mathbf{z}_2\mathbf{z}_2^T)\mathbf{y}_k \\ &= \mathbf{B}_k\mathbf{y}_k + c_1\mathbf{z}_1\mathbf{z}_1^T\mathbf{y}_k + c_2\mathbf{z}_2\mathbf{z}_2^T\mathbf{y}_k \end{aligned}$$

由於 $\mathbf{z}_1^T\mathbf{y}_k$ 和 $\mathbf{z}_2^T\mathbf{y}_k$ 分別為一純量，且 \mathbf{z}_1 和 \mathbf{z}_2 並非唯一解，觀察上式可簡單假設 $\mathbf{z}_1 = \Delta\mathbf{x}_k$ 、 $\mathbf{z}_2 = -\mathbf{B}_k\mathbf{y}_k$ 、 $c_1 = \frac{1}{\mathbf{z}_1^T\mathbf{y}_k}$ 及 $c_2 = -\frac{1}{\mathbf{z}_2^T\mathbf{y}_k}$ 。將以上代入 (4-15) 可推得

$$\begin{aligned} \mathbf{B}_{k+1} &= \mathbf{B}_k + c_1\mathbf{z}_1\mathbf{z}_1^T + c_2\mathbf{z}_2\mathbf{z}_2^T \\ &= \mathbf{B}_k + \frac{\Delta\mathbf{x}_k\Delta\mathbf{x}_k^T}{\Delta\mathbf{x}_k^T\mathbf{y}_k} - \frac{\mathbf{B}_k\mathbf{y}_k\mathbf{y}_k^T\mathbf{B}_k}{\mathbf{y}_k^T\mathbf{B}_k\mathbf{y}_k} \end{aligned} \quad (4-16)$$

以上海賽矩陣更新公式稱為 Davidon–Fletcher–Powell (DFP) method。

此外，由 (4-15) 可知，海賽矩陣近似的方式有很多種，一般常用的更新方法為 Broyden–Fletcher–Goldfarb–Shanno (BFGS) method，其更新公式如下：

$$\mathbf{B}_{k+1} = \left(\mathbf{I} - \frac{\Delta\mathbf{x}_k\mathbf{y}_k^T}{\mathbf{y}_k^T\Delta\mathbf{x}_k} \right)^T \mathbf{B}_k \left(\mathbf{I} - \frac{\mathbf{y}_k\Delta\mathbf{x}_k^T}{\mathbf{y}_k^T\Delta\mathbf{x}_k} \right) + \frac{\Delta\mathbf{x}_k\Delta\mathbf{x}_k^T}{\mathbf{y}_k^T\Delta\mathbf{x}_k} \quad (4-17)$$

其中 $\mathbf{H}_0 = \mathbf{B}_0 = \mathbf{I}$ 及 $\Delta\mathbf{x}_0 = -\mathbf{H}_0^{-1}\nabla f(\mathbf{x}_0) = -\nabla f(\mathbf{x}_0)$ 。

由公式 (4-16) 和 (4-17) 可發現在迭代的過程中，我們並沒真正的計算海賽矩陣，而是利用過去幾次迭代的一次微分資訊去建構一個近似於原目標函數的海賽矩陣及其反矩陣，即利用 \mathbf{B}_k 做為 \mathbf{H}_k^{-1} 的近似值。

4.1.4 共軛梯度法 (Conjugate Gradient method)

由於 $\Delta \mathbf{X}_k = -\mathbf{H}_k \nabla f(\mathbf{x}_k)$, 故 $\nabla f(\mathbf{x}_k) = -\mathbf{H}_k \Delta \mathbf{X}_k$ 。由直線搜尋法可知, (4-11) 可改寫為

$$\nabla f(\mathbf{x}_k + \alpha_k \Delta \mathbf{x}_k)^\top \Delta \mathbf{x}_k = -\Delta \mathbf{x}_{k+1}^\top \mathbf{H}_{k+1} \Delta \mathbf{x}_k = \mathbf{0} \quad (4-18)$$

假設 $\Delta \mathbf{x}_0 = -\nabla f(\mathbf{x}_0)$, 由於 $\Delta \mathbf{x}_0$ 與 $\Delta \mathbf{x}_1$ 共軛, 所以 $(\Delta \mathbf{x}_1)^\top \mathbf{H} \Delta \mathbf{x}_0 = -(\Delta \mathbf{x}_1)^\top \nabla f(\mathbf{x}_0) = 0$ 。令 $\Delta \mathbf{x}_1 = -\nabla f(\mathbf{x}_1) + w_0 \Delta \mathbf{x}_0$, 可推得

$$(\Delta \mathbf{x}_1)^\top \mathbf{H} \Delta \mathbf{x}_0 = -\nabla f(\mathbf{x}_1)^\top \mathbf{H} \Delta \mathbf{x}_0 + w_0 (\Delta \mathbf{x}_0)^\top \mathbf{H} \Delta \mathbf{x}_0 = 0$$

上式經化簡後得

$$w_0 = \frac{\nabla f(\mathbf{x}_1)^\top \mathbf{H} \Delta \mathbf{x}_0}{(\Delta \mathbf{x}_0)^\top \mathbf{H} \Delta \mathbf{x}_0} = -\frac{\nabla f(\mathbf{x}_1)^\top \mathbf{H} \nabla f(\mathbf{x}_0)}{(\nabla f(\mathbf{x}_0))^\top \mathbf{H} \nabla f(\mathbf{x}_0)}$$

又由 (4-13), 令 $\nabla f(\mathbf{x}_1) = \nabla f(\mathbf{x}_0) - s \mathbf{H} \nabla f(\mathbf{x}_0)$, 可推得

$$\begin{aligned} 0 &= \nabla f(\mathbf{x}_1)^\top \nabla f(\mathbf{x}_0) \\ &= \nabla f(\mathbf{x}_0)^\top \nabla f(\mathbf{x}_0) - s \nabla f(\mathbf{x}_0)^\top \mathbf{H} \nabla f(\mathbf{x}_0) \end{aligned}$$

化簡後得

$$s = \frac{\nabla f(\mathbf{x}_0)^\top \nabla f(\mathbf{x}_0)}{\nabla f(\mathbf{x}_0)^\top \mathbf{H} \nabla f(\mathbf{x}_0)}$$

或

$$\mathbf{H} \nabla f(\mathbf{x}_0) = \frac{\nabla f(\mathbf{x}_1) - \nabla f(\mathbf{x}_0)}{s}$$

將上式帶入 w_0 , 得

$$\begin{aligned} w_0 &= -\frac{\nabla f(\mathbf{x}_1)^\top \mathbf{H} \nabla f(\mathbf{x}_0)}{(\nabla f(\mathbf{x}_0))^\top \mathbf{H} \nabla f(\mathbf{x}_0)} \\ &= -\frac{\nabla f(\mathbf{x}_1)^\top \nabla f(\mathbf{x}_1) - \nabla f(\mathbf{x}_1)^\top \nabla f(\mathbf{x}_0)}{\nabla f(\mathbf{x}_0)^\top \nabla f(\mathbf{x}_1) - \nabla f(\mathbf{x}_0)^\top \nabla f(\mathbf{x}_0)} \\ &= -\frac{\nabla f(\mathbf{x}_1)^\top \nabla f(\mathbf{x}_1)}{\nabla f(\mathbf{x}_0)^\top \nabla f(\mathbf{x}_0)} \end{aligned}$$

因此, 由梯度下降法, $\Delta \mathbf{x}_0 = -\nabla f(\mathbf{x}_0)$,

$$\Delta \mathbf{x}_1 = -\nabla f(\mathbf{x}_1) + \frac{\nabla f(\mathbf{x}_1)^\top \nabla f(\mathbf{x}_1)}{\nabla f(\mathbf{x}_0)^\top \nabla f(\mathbf{x}_0)} \Delta \mathbf{x}_0 \quad (4-19)$$

$$= -\nabla f(\mathbf{x}_1) - \frac{\nabla f(\mathbf{x}_1)^\top \nabla f(\mathbf{x}_1)}{\nabla f(\mathbf{x}_0)^\top \nabla f(\mathbf{x}_0)} \nabla f(\mathbf{x}_0) \quad (4-20)$$

同理, 可求得第 k 階段之搜尋方向為

$$\Delta \mathbf{x}_k = -\nabla f(\mathbf{x}_k) - \frac{\nabla f(\mathbf{x}_k)^\top \nabla f(\mathbf{x}_k)}{\nabla f(\mathbf{x}_{k-1})^\top \nabla f(\mathbf{x}_{k-1})} \nabla f(\mathbf{x}_{k-1})$$

以上公式為 Fletcher 和 Reeves 所提出，由於第 k 階段與第 $k+1$ 階段的搜尋方向共軛，在求解下一階段的搜尋方向時，我們僅需利用前一階段所求得的梯度訊息，因此免除掉海賽矩陣是奇異矩陣或狀況不佳矩陣的困擾。這種方式讓所有搜尋方向皆互為共軛方向，因而被稱為共軛梯度演算法。之後，Polak 和 Ribiere 將 (4.1.4) 修改為

$$\Delta \mathbf{x}_k = -\nabla \mathbf{f}(\mathbf{x}_k) - \frac{\nabla \mathbf{f}(\mathbf{x}_k)^T (\nabla \mathbf{f}(\mathbf{x}_k) - \nabla \mathbf{f}(\mathbf{x}_{k-1}))}{\nabla \mathbf{f}(\mathbf{x}_{k-1})^T \nabla \mathbf{f}(\mathbf{x}_{k-1})} \nabla \mathbf{f}(\mathbf{x}_{k-1}) \quad (4-21)$$

由於

$$\frac{\nabla \mathbf{f}(\mathbf{x}_k)^T (\nabla \mathbf{f}(\mathbf{x}_k) - \nabla \mathbf{f}(\mathbf{x}_{k-1}))}{\nabla \mathbf{f}(\mathbf{x}_{k-1})^T \nabla \mathbf{f}(\mathbf{x}_{k-1})}$$

可能為負數，因此一般將 (4-21) 修改為

$$\Delta \mathbf{x}_k = -\nabla \mathbf{f}(\mathbf{x}_k) - \max \left\{ 0, \frac{\nabla \mathbf{f}(\mathbf{x}_k)^T (\nabla \mathbf{f}(\mathbf{x}_k) - \nabla \mathbf{f}(\mathbf{x}_{k-1}))}{\nabla \mathbf{f}(\mathbf{x}_{k-1})^T \nabla \mathbf{f}(\mathbf{x}_{k-1})} \right\} \nabla \mathbf{f}(\mathbf{x}_{k-1}) \quad (4-22)$$

4.1.5 萊文貝格-馬夸特法 (Levenberg-Marquardt method)

假設目標函數可寫成

$$f(\mathbf{x}) = \frac{(r_1(\mathbf{x}))^2}{2} + \frac{(r_2(\mathbf{x}))^2}{2} + \cdots + \frac{(r_m(\mathbf{x}))^2}{2} = \frac{1}{2} \sum_{i=1}^m (r_i(\mathbf{x}))^2$$

此種特定形式時，函數 $f(\mathbf{x})$ 對 \mathbf{x} 做一階及二階微分可得

$$\nabla f(\mathbf{x}) = \sum_{i=1}^m r_i(\mathbf{x}) \nabla r_i(\mathbf{x}) = \mathbf{J}^T \mathbf{r}(\mathbf{x})$$

及

$$\mathbf{H} = \nabla^2 f(\mathbf{x}) = \mathbf{J}^T \mathbf{J} + \sum_{i=1}^m r_i(\mathbf{x}) \nabla^2 r_i(\mathbf{x}) \approx \mathbf{J}^T \mathbf{J}$$

其中

$$\mathbf{J} = \begin{bmatrix} \frac{\partial r_1}{\partial x_1} & \cdots & \frac{\partial r_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial r_m}{\partial x_1} & \cdots & \frac{\partial r_m}{\partial x_n} \end{bmatrix}$$

故由 (4-10)，第 $k+1$ 階段的迭代值 \mathbf{x}_{k+1} 可改寫為

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k (\mathbf{J}^T \mathbf{J})^{-1} \nabla \mathbf{f}(\mathbf{x}_k) \quad (4-23)$$

以上作法我們稱為高斯牛頓法 (Gauss-Newton method)。Levenberg 發現上式若改寫為

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k (\mathbf{J}^T \mathbf{J} + \lambda_k \mathbf{I})^{-1} \nabla \mathbf{f}(\mathbf{x}_k) \quad (4-24)$$

則可結合牛頓法和梯度下降法。當 λ 小時，有牛頓法的快速收斂性；另一方面當 λ 大時，也有梯度下降法的全局搜索特性。如同前面所述，當 λ 很大時， $(J^\top J + \lambda_k I)^{-1}$ 不一定存在。為解決此一困擾，Marquardt 將 Levenberg method 修改為

$$x_{k+1} = x_k - \alpha_k (J^\top J + \lambda_k \text{diag}(J^\top J))^{-1} \nabla f(x_k) \quad (4-25)$$

以上作法稱為 Levenberg-Marquardt Method。

4.1.6 範例

在Mathematica 最佳化套件中所提供之梯度有關的無限制最佳化函數包含有 Minimize, Maximize, FindMinimum 和 FindMaximum。Minimize 和 Maximize 適合用在變數較少及最佳解具解析解的情況下，也就是最佳解可經由有限次常見運算的組合的方式求得。然而一般情況下，多數的最佳化問題都只能透過數值分析的方法求解近似值，此時就必須改用 FindMinimum 和 FindMaximum。以上四個函數的使用方式大致介紹如下：

1. Minimize[f(x), x]: 以 x 為變數求解函數 $f(x)$ 的全域極小值。
2. Maximize[f(x), x]: 以 x 為變數求解函數 $f(x)$ 的全域極大值。
3. Minimize[f(x₁, x₂, ..., x_n), {x₁, x₂, ..., x_n}]: 以 x_1, x_2, \dots, x_n 為變數求解函數 $f(x_1, x_2, \dots, x_n)$ 的全域極小值。
4. Maximize[f(x₁, x₂, ..., x_n), {x₁, x₂, ..., x_n}]: 以 x_1, x_2, \dots, x_n 為變數求解函數 $f(x_1, x_2, \dots, x_n)$ 的全域極大值。
5. FindMinimum[f(x), x]: 以 x 為變數求解函數 $f(x)$ 的局部極小值。
6. FindMinimum[f(x), {x, x₀}]: 以 $x = x_0$ 為起始值求解函數 f 的局部極小值。
7. FindMinimum[f(x₁, x₂, ..., x_n), {x₁, x₂, ..., x_n}]: 以 x_1, x_2, \dots, x_n 為變數求解 f 的局部極小值。
8. FindMinimum[f(x₁, x₂, ..., x_n), {x₁, x₁₀}, ..., {x_n, x_{n0}}]: 以 (x_{10}, \dots, x_{n0}) 為起始值求解 f 的局部極小值。
9. FindMaximum[f(x), x]: 以 x 為變數求解函數 $f(x)$ 的局部極大值。
10. FindMaximum[f(x), {x, x₀}]: 以 $x = x_0$ 為起始值求解函數 f 的局部極大值。

11. `FindMaximum[f(x1, x2, ..., xn), {x1, x2, ..., xn}]`: 以 x_1, x_2, \dots, x_n 為變數求解 f 的局部極大值。
12. `FindMaximum[f(x1, x2, ..., xn), {x1, x10}, ..., {xn, xn0}]`: 以 (x_{10}, \dots, x_{n0}) 為起始值求解 f 的局部極大值。

此外，上述指令用法均會傳回最佳解及最佳解所對應的函數值，若只想傳回最佳解或對應的函數值，Mathematica 在 7.0 後新增了一些對應的指令：

1. `FindArgMin`, `FindArgMax`: 傳回 `FindMinimum` 或 `FindMaximum` 求得的最佳解。
2. `FindMinValue`, `FindMaxValue`: 傳回 `FindMinimum` 或 `FindMaximum` 求得最佳解的對應函數值。

Mathematica 在進行 `FindMinimum` 或 `FindMaximum` 時也可以指定以下幾種搜尋方式求解：Newton, QuasiNewton, LevenbergMarquardt, ConjugateGradient 及 PrincipalAxis。Mathematica 在計算時會自動判斷函數的屬性決定適當的演算法，若不指定計算方式，預設以共軛梯度法（ConjugateGradient）進行求解；當目標函數含有平方和時，Mathematica 則改以 LevenbergMarquardt 法進行；而當變數指定兩個起始值或函數無法微分時，則會改以使用主軸法（PrincipalAxis）進行求解；若以準牛頓法（Quasi-Newton Method）進行求解，Mathematica 會以 BFGS 作為海賽矩陣更新的近似方法。

範例 4-1. 求 $f(x) = 3x^4 - 28x^3 + 84x^2 - 96x + 42$ 之極小值

定義函數 $f(x)$

In[1]: `f[x_] := 3x^4 - 28x^3 + 84x^2 - 96x + 42`

計算臨界點

In[2]: `{x, f[x]} /. Solve[f'[x] == 0, x]`

Out[2]: `\{\{1, 5\}, \{2, 10\}, \{4, -22\}\}`

計算臨界點的二階條件

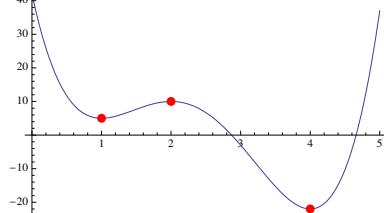
In[3]: `f''[x] /. Solve[f'[x] == 0, x]`

Out[3]: {36, -24, 72}

計算臨界點並繪出函數圖

In[4]: Plot[f[x], {x, 0, 5}, Epilog -> {Red, PointSize[Large], Point[%]}]

Out[4]:



由二階條件和圖形可知，函數 $f(x)$ 的極小值為 $f(4) = -22$; $f(1)$ 和 $f(2)$ 分別為函數 $f(x)$ 的局部極小與局部極大值。然而，由於最佳解具解析解，若改以函數 Minimize 求解此例，則不需做任何臨界點比較。

使用 Minimize 求解函數最大值

In[5]: Minimize[f[x], x]

Out[5]: {-22, {x → 4}}

使用 FindMinimum 求解，並輸出所有的計算過程

In[6]: Reap@FindMinimum[f[x], x, EvaluationMonitor :> {Sow[x]}]

FindMinimum::fmgz: Encountered a gradient that is effectively zero. The result returned may not be a minimum;
it may be a maximum or a saddle point. »

Out[6]: {{5., {x → 1.}}, {{1.}}}

上式輸出傳回錯誤訊息的原因為 $x = 1$ 在此例為臨界點，由於臨界點的一階導數為 0，因此牛頓法使用，故導致警告訊息產生。

改以 $x = 0.1$ 為起始值，輸出 FindMinimum 求解所有的計算過程

In[7]: Reap@FindMinimum[f[x], {x, 1}, EvaluationMonitor :> {Sow[x]}]

Out[7]: {{5., {x → 1.}}, {{0.1, 0.6, 0.79979, 0.935152, 0.986551, 0.998928, 0.999981, 1., 1.}}}

使用不同的起始值利用 FindMinimum 求解

In[8]: TableForm[{#, x, f[x]}/. Quiet@FindMinimum[f[x], {x, #}] [[2]] &/@
Range[0, 4, 0.5], TableHeadings -> {None, {"Initial", "x*", "f*"}}]

Out[8]:	Initial	x^*	f^*
	0.	1.	5.
	0.5	1.	5.
	1.	1.	5.
	1.5	1.	5.
	2.	2.	10.
	2.5	4.	-22.
	3.	4.	-22.
	3.5	4.	-22.
	4.	4.	-22.

使用不同的起始值利用 FindArgMin 和 FindMinValue 求解

```
In[9]: TableForm[Quiet@{#,FindArgMin[f[x],{x,#}],FindMinValue[f[x],{x,#}]}&@  
Range[0,4,0.5],TableHeadings->{None,{"Initial","x*","f*"}}]
```

Out[9]:	Initial	x^*	f^*
	0.	1.	5.
	0.5	1.	5.
	1.	1.	5.
	1.5	1.	5.
	2.	2.	10.
	2.5	4.	-22.
	3.	4.	-22.
	3.5	4.	-22.
	4.	4.	-22.

由以上輸出可發現只有當起始值大於 2.5 時，FindMinimum 才可求得最佳解。否則，當起始值小於 2.5 時大多數的演算法均會落如局部極大值或局部極小值。

範例 4-2. 求平面上一點 (x_0, y_0) 到直線 $ax + by + c = 0$ 的最短距離。

計算過點 (x_0, y_0) 且斜率為 m 的方程式與 $ax + by + c = 0$ 的交點座標

```
In[1]: pt={x,y}/.Solve[{a*x+b*y+c==0,y==m*(x-x0)+y0},{x,y}][[1]]
```

$$\text{Out[1]: } \left\{ -\frac{-bmx_0 + by_0 + c}{a + bm}, -\frac{amx_0 - ay_0 + cm}{a + bm} \right\}$$

點 (x_0, y_0) 與 pt 的距離平方

```
In[2]: distancesquare=(x0,y0)-pt).(x0,y0)-pt)//Simplify
```

$$\text{Out[2]: } \frac{(m^2 + 1)(ax_0 + by_0 + c)^2}{(a + bm)^2}$$

使用 Solve 求解最短距離

```
In[3]: Sqrt[distancesquare]/.Solve[D[distancesquare,m]==0,m][[1]]//Simplify
```

$$\text{Out[3]: } \sqrt{\frac{(ax_0 + by_0 + c)^2}{a^2 + b^2}}$$

求解 $ax + by + c = 0$ 上與點 (x_0, y_0) 最短距離的座標

In[4]: pt/.Solve[D[distancesquare,m]==0,m][[1]]//Simplify

Out[4]: $\left\{ \frac{b^2x_0 - a(by_0 + c)}{a^2 + b^2}, \frac{a^2y_0 - b(ax_0 + c)}{a^2 + b^2} \right\}$

從一個角度，我們也可以利用直線方程式 $ax + by + c = 0$ 將 y 改寫為 x 的函數，直接求解直線 $ax + by + c = 0$ 上與點 (x_0, y_0) 最短距離的 x 座標。

求解 $ax + by + c = 0$ 上與點 (x_0, y_0) 最短距離的座標

In[5]: {x,-(a*x+c)/b}/.Solve[D[({x0,y0})-{x,-(a*x+c)/b}].

{({x0,y0})-{x,-(a*x+c)/b}},x]==0,x][[1]]//Simplify

Out[5]: $\left\{ \frac{b^2x_0 - a(c + by_0)}{a^2 + b^2}, \frac{-b(c + ax_0) + a^2y_0}{a^2 + b^2} \right\}$

計算最短距離

In[6]: Sqrt[({x0,y0})-%].({x0,y0})%]]//Simplify

Out[6]: $\sqrt{\frac{(c + ax_0 + by_0)^2}{a^2 + b^2}}$

由於具有解析解，我們可利用 Minimize 作為求解工具。

使用 Minimize 求解兩點間的最短距離平方

In[7]: Assuming[{b!=0},

Minimize[SquaredEuclideanDistance[{x0,y0},{x,-(a*x+c)/b}].

Abs[z_]:=z,x]]//Simplify]

Out[7]: $\left\{ \frac{(ax_0 + by_0 + c)^2}{a^2 + b^2}, \left\{ x \rightarrow \frac{b^2x_0 - a(by_0 + c)}{a^2 + b^2} \right\} \right\}$

求解 $ax + by + c = 0$ 上與點 (x_0, y_0) 最短距離的座標

In[8]: {x,-(a*x+c)/b}/.%[[2]]//Simplify

Out[8]: $\left\{ \frac{b^2x_0 - a(by_0 + c)}{a^2 + b^2}, \frac{a^2y_0 - b(ax_0 + c)}{a^2 + b^2} \right\}$

範例 4-3. 求空間上一點 (x_0, y_0, z_0) 到平面 $ax + by + cz + d = 0$ 的最短距離。

平面 $ax + by + cz + d = 0$ 一點 $\left(x, y, -\frac{d + ax + by}{c}\right)$ 與點 (x_0, y_0, z_0) 的距離平方

In[1]: distancesquare=(({x0,y0,z0})-{x,y,-(d+a*x+b*y)/c}).

{({x0,y0,z0})-{x,y,-(d+a*x+b*y)/c}}

Out[1]: $(x_0 - x)^2 + (y_0 - y)^2 + \left(z_0 - \frac{-ax - by + d}{c} \right)^2$

使用 Solve 求解最短距離

In[2]: $\text{Sqrt}[d\text{istancesquare}] /. \text{Solve}[D[d\text{istancesquare}, \{x, y\}] == \{0, 0\}, \{x, y\}] [[1]] // \text{Simplify}$

Out[2]: $\sqrt{\frac{(ax_0 + by_0 + cz_0 + d)^2}{a^2 + b^2 + c^2}}$

求解 $ax + by + cz + d = 0$ 上與點 (x_0, y_0, z_0) 最短距離的座標

In[3]: $\{x, y, -(d + a*x + b*y)/c\} /. \text{Solve}[D[d\text{istancesquare}, \{x, y\}] == \{0, 0\}, \{x, y\}] [[1]] // \text{Simplify}$

Out[3]: $\left\{ \frac{(b^2 + c^2)x_0 - a(d + by_0 + cz_0)}{a^2 + b^2 + c^2}, \frac{(a^2 + c^2)y_0 - b(d + ax_0 + cz_0)}{a^2 + b^2 + c^2}, \frac{-c(d + ax_0 + by_0) + (a^2 + b^2)z_0}{a^2 + b^2 + c^2} \right\}$

範例 4-4. 求解 $\min_{x_1, x_2} f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$

定義函數 $f(x_1, x_2)$

In[1]: $f[x1_, x2_] := 100*(x2 - x1^2)^2 + (1 - x1)^2$

定義梯度

In[2]: $\text{grad}[x1_, x2_] = D[f[x1, x2], \{x1, x2\}]$

Out[2]: $\{-2(1 - x) - 400x(-x^2 + y), 200(-x^2 + y)\}$

求解臨界點

In[3]: $\text{sol} = \text{Solve}[\text{grad}[x1, x2] == \{0, 0\}, \{x1, x2\}] [[1]]$

Out[3]: $\{x1 \rightarrow 1, x2 \rightarrow 1\}$

計算海賽矩陣

In[4]: $\text{hessian}[x1_, x2_] = D[f[x1, x2], \{x1, x2\}, 2]$

Out[4]: $\{2 + 800x1^2 - 400(-x1^2 + x2), -400x1\}, \{-400x1, 200\}$

計算海賽矩陣各階行列值

```
In[5]: Table[Det@hessian[x1,x2]/.sol][[1;;z,1;;z]],{z,2}]
```

```
Out[5]: {802, 400}
```

由海賽矩陣行列值可知 \mathbf{H} 為一正定矩陣，故 $f(x_1, x_2)$ 在 $\{x_1, x_2\} = \{1., 1.\}$ 有極小值 0。若變數較多時，我們也可使用 Mathematica 提供的函數 PositiveDefiniteMatrixQ 來判斷矩陣是否為正定矩陣。

使用函數 PositiveDefiniteMatrixQ 判斷海賽矩陣是否為正定矩陣

```
In[6]: PositiveDefiniteMatrixQ@hessian[x1,x2]/.sol
```

```
Out[6]: True
```

接下來，我們利用修正牛頓法進行求解。

定義更新矩陣

```
In[7]: delta[x1_,x2_] == a*Inverse[hessian[x1,x2]].D[f[x1,x2],{{x1,x2}}]//  
FullSimplify
```

```
Out[7]: ⎧ a(-1+x1) ⎫  
⎩ -1-200x1^2+200x2 , a ⎨ x1^2 - 2(-1+x1)x1 ⎬ - x2 ⎭  
⎩ 1+200x1^2-200x2 ⎾
```

NestWhileList 以 $(0, 0)$ 為起始值，利用牛頓法求解

```
In[8]: solNW1=NestWhileList[#+delta@@#/Quiet@FindRoot[D[f@@(#+delta@@#),a],  
{a,0}]&,{0,0},Unequal,All];
```

FixedPointList 以 $(0, 0)$ 為起始值，利用牛頓法求解

```
In[9]: solNW2=FixedPointList[#+delta@@#/Quiet@FindRoot[D[f@@(#+delta@@#),a],  
{a,0}]&,{0,0}];
```

輸出 solNW1

```
In[10]: TableForm[Flatten[{#, solNW1[[#]]}] & /@ Range[Length@solNW1],
TableHeadings -> {None, {"i", "x1", "x2"}}]
```

Out[10]:

i	x1	x2
1	0	0
2	0.161262	0
3	0.367189	0.106009
4	0.630157	0.380126
5	0.807286	0.639082
6	0.976581	0.951533
7	0.997991	0.996209
8	1.00003	1.00006
9	1.	1.
10	1.	1.
11	1.	1.

輸出 solNW2

```
In[11]: TableForm[Flatten[{#, solNW2[[#]]}] & /@ Range[Length@solNW2],
TableHeadings -> {None, {"i", "x1", "x2"}}]
```

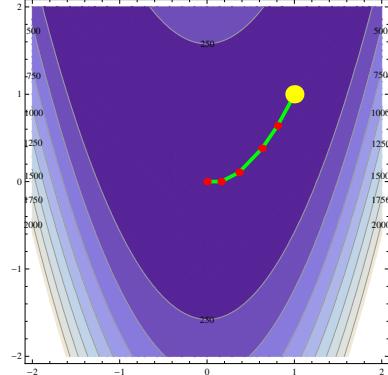
Out[11]:

i	x1	x2
1	0	0
2	0.161262	0
3	0.367189	0.106009
4	0.630157	0.380126
5	0.807286	0.639082
6	0.976581	0.951533
7	0.997991	0.996209
8	1.00003	1.00006
9	1.	1.
10	1.	1.
11	1.	1.
12	1.	1.

利用 solNW1 輸出修正牛頓法求解的步驟圖

```
In[12]: ContourPlot[f[x1, x2], {x1, -2, 2}, {x2, -2, 2}, ContourLabels -> True,
Epilog -> {Green, Thickness[0.01], Line[solNW1], Red, PointSize[Large],
Point[solNW1], Yellow, PointSize[0.05], Point[solNW1[[-1]]]}]
```

Out[12]:



定義擬牛頓法的起始值

```
In[13]: x0={0,0};init={x0,x0+a*grad@@x0,-grad@@x0,
IdentityMatrix[Length@x0]}/.FindRoot[D[f@@(x0+a*grad@@x0),a],{a,0}];
```

定義 DFP 海賽矩陣近似函數

```
In[14]: DFP[x1_,x2_,d_,B_]:=B+Outer[Times,d,d]/(d.(grad@@x2-grad@@x1))-B.Outer[Times,grad@@x2-grad@@x1,grad@@x2-grad@@x1].B/((grad@@x2-grad@@x1).B.(grad@@x2-grad@@x1));
```

定義 BFGS 海賽矩陣近似函數

```
In[15]: BFGS[x1_,x2_,d_,B_]:=Transpose[IdentityMatrix[Length@x1]-Outer[Times,d,grad@@x2-grad@@x1]/((grad@@x2-grad@@x1).d)].B.(IdentityMatrix[Length@x1]-Outer[Times,grad@@x2-grad@@x1,d]/((grad@@x2-grad@@x1).d))+Outer[Times,d,d]/((grad@@x2-grad@@x1).d);
```

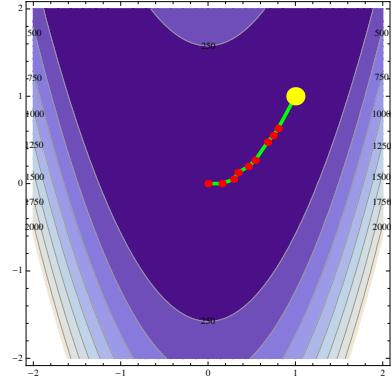
擬牛頓法以 DFP method 及 {0, 0} 為起始值進行求解

```
In[16]: solDFP=NestWhileList[(direction=-(DFP@@#).(grad@@#[[2]]);{#[[2]],#[[2]]+a*direction,direction,
DFP@@{#[[2]],#[[2]]+a*direction,direction,DFP@@#}}/.Quiet@FindRoot[D[f@@(#[[2]]+a*direction),a],{a,0}]&,
init,Norm#[[1]]-#[[2]]]>10^-6&];
```

利用 solDFP 輸出 DFP 法求解的步驟圖

```
In[17]: ContourPlot[f[x1,x2],{x1,-2,2},{x2,-2,2},ContourLabels->True,
Epilog->{Green,Thickness[0.01],Line[solDFP[[All,1]]],
Red,PointSize[Large],Point[solDFP[[All,1]]],
Yellow,PointSize[0.05],Point[solDFP[[-1,1]]]}]
```

Out[17]:



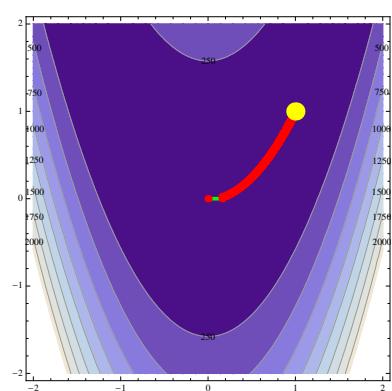
擬牛頓法以 BFGS method 及 $\{0, 0\}$ 為起始值進行求解

```
In[18]: solBFGS=NestWhileList[(direction=-({BFGS@@#[[2]]}).(grad@@#[[2]]));
  #[[2]], #[[2]]+a*direction,a*direction,
  BFGS@@{#[[2]],#[[2]]+a*direction,a*direction,BFGS@@#[[2]]}/.
  Quiet@FindRoot[D[f@@({#[[2]]+a*direction},a],{a,0}]&,
  init,Norm#[[1]]-#[[2]]]>10^-6&];
```

利用 solBFGS 輸出 DFP 法求解的步驟圖

```
In[19]: ContourPlot[f[x1,x2],{x1,-2,2},{x2,-2,2},ContourLabels->True,
 Epilog->{Green,Thickness[0.01],Line[solBFGS[[All,1]]],
 Red,PointSize[Large],Point[solBFGS[[All,1]]],
 Yellow,PointSize[0.05],Point[solBFGS[[-1,1]]]}]
```

Out[19]:



定義共軛梯度法的起始值

```
In[20]: x0={0,0};init={x0,x0+a*grad@@x0,-grad@@x0}/.FindRoot[
 D[f@@(x0+a*grad@@x0),a],{a,0}];
```

Out[20]:

```
{ {0, 0}, {0.161262, 0}, {2, 0} }
```

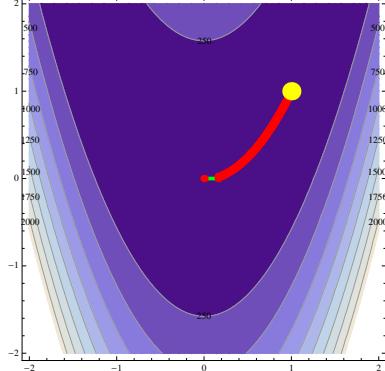
共軛梯度法以 $\{0, 0\}$ 為起始值進行求解

```
In[21]: solCG=NestWhileList[(direction=-grad@@#[[2]]+Max[0,#[[2]].(grad@@#[[2]]-grad@@#[[1]])/(grad@@#[[1]].grad@@#[[1]])]*#[[3]]; #[[2]],#[[2]]+a*direction,a*direction]/. Quiet@FindRoot[D[f@@(#[[2]]+a*direction),a],{a,0}]&, init, Norm[#[[1]]-#[[2]]]>10^-6&];
```

利用 `solCG` 輸出共軛梯度法求解的步驟圖

```
In[22]: ContourPlot[f[x1,x2],{x1,-2,2},{x2,-2,2},ContourLabels->True, Epilog->{Green,Thickness[0.01],Line[solCG[[All,1]]], Red,PointSize[Large],Point[solCG[[All,1]]], Yellow,PointSize[0.05],Point[solCG[[-1,1]]]}]
```

Out[22]:



由於目標函數 $f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$ 為平方和，因此可利用萊文貝格-馬夸特法來求解。

定義 Jacobian 矩陣

```
In[23]: J[x1_,x2_]:=D[#,{{x1,x2}}]&/@{Sqrt[200]*(x2-x1^2),Sqrt[2](1-x1)}
```

```
Out[23]: \{\{-20\sqrt{2}x1, 10\sqrt{2}\}, \{-\sqrt{2}, 0\}\}
```

定義高斯牛頓法的更新矩陣

```
In[24]: GNdelta[x1_,x2_]=-a*Inverse[Transpose[J[x1,x2]].J[x1,x2]].D[f[x1,x2],{{x1,x2}}]//FullSimplify
```

```
Out[24]: \{a - ax1, -a((x1 - 2)x1 + x2)\}
```

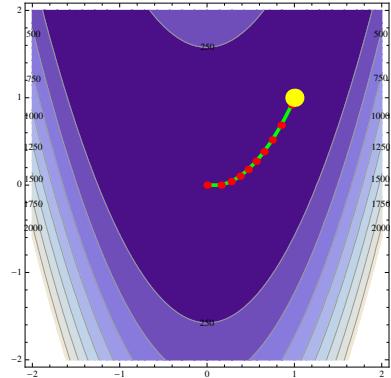
(0, 0) 為起始值利用高斯牛頓法求解

```
In[25]: solGN=NestWhileList[#+GNdelta@@#/Quiet@FindRoot[
D[f@@(#+GNdelta@@#),a],{a,0}]&,{0,0},Unequal,All];
```

利用 solGN 輸出高斯牛頓法求解的步驟圖

```
In[26]: ContourPlot[f[x1,x2],{x1,-2,2},{x2,-2,2},ContourLabels->True,
Epilog->{Green,Thickness[0.01],Line[solGN],Red,PointSize[Large],
Point[solGN],Yellow,PointSize[0.05],Point[solGN[[-1]]]}]
```

Out[26]:



假設 $\lambda = 1/10$, 定義萊文貝格-馬夸特法的更新矩陣

```
In[27]: LMdelta[x1_,x2_]=-a*Inverse[Transpose[J[x1,x2]].J[x1,x2]
+1/10*IdentityMatrix[2]].D[f[x1,x2],{{x1,x2}}]//FullSimplify
```

```
Out[27]: {20 a (-2001 + 200 x1^3 + x1 (2001 - 200 x2)) / (42021 + 8000 x1^2), 2000 a (x1 (-40 + 19 x1) + 21 x2) / (42021 + 8000 x1^2)}
```

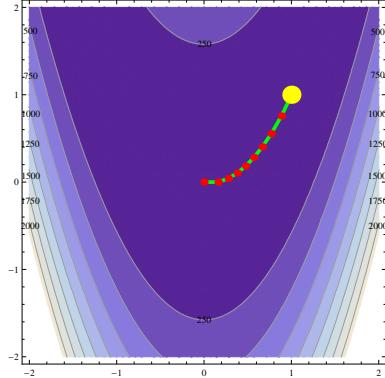
(0, 0) 為起始值利用萊文貝格-馬夸特法求解

```
In[28]: solLM=NestWhileList[#+LMdelta@@#/Quiet@FindRoot[
D[f@@(#+LMdelta@@#),a],{a,0}]&,{0,0},Unequal,All];
```

利用 solLM 輸出萊文貝格-馬夸特法求解的步驟圖

```
In[29]: ContourPlot[f[x1,x2],{x1,-2,2},{x2,-2,2},ContourLabels->True,
Epilog->{Green,Thickness[0.01],Line[solLM],Red,PointSize[Large],
Point[solLM],Yellow,PointSize[0.05],Point[solLM[[-1]]]}]
```

Out[29]:



範例 4-5. 求解 $\min_{\mathbf{x}} f(\mathbf{x}) = \sum_{i=1}^{N-1} [(1-x_i)^2 + 100(x_{i+1}-x_i^2)^2]$

定義函數 $f(\mathbf{x})$

```
In[1]: f[n_]:=Sum[(1-x[i])^2+100(x[i+1]-x[i]^2)^2,{i,n-1}][[1]]/.
x[i_]:>ToExpression["x"<>ToString[i]];
```

定義變數 \mathbf{x}

```
In[2]: vars[n_]:=Table[x[i],{i,n}]/.x[i_]:>ToExpression["x"<>ToString[i]];
```

輸出 $f[2], f[3], f[4]$

```
In[3]: f/@{2,3,4}//Column
```

```
Out[3]: 100 (x2 - x1^2)^2 + (1 - x1)^2
100 (x2 - x1^2)^2 + (1 - x1)^2 + 100 (x3 - x2^2)^2 + (1 - x2)^2
100 (x2 - x1^2)^2 + (1 - x1)^2 + 100 (x3 - x2^2)^2 + (1 - x2)^2 + 100 (x4 - x3^2)^2 + (1 - x3)^2
```

FindMinimum 以 $(0,0)$ 為起始值求解 $f[2]$

```
In[4]: FindMinimum[f[2],Transpose@{vars[2],Table[0,{Length@vars[2]}]}]
```

```
Out[4]: {8.34935 × 10-27, {x1 → 1., x2 → 1., x3 → 1.}}
```

FindMinimum 以 $(0,0,0)$ 為起始值求解 $f[3]$

```
In[5]: FindMinimum[f[3], Transpose@{vars[3]}, Table[0, {Length@vars[3]}]]
```

```
Out[5]: {8.34935 × 10-27, {x1 → 1., x2 → 1., x3 → 1.}}
```

若要了解 FindMinimum 或 FindMaximum 的迭代計算的過程，則可藉由指定參數 EvaluationMonitor 或 StepMonitor 來要求將收斂過程輸出至指定字串。

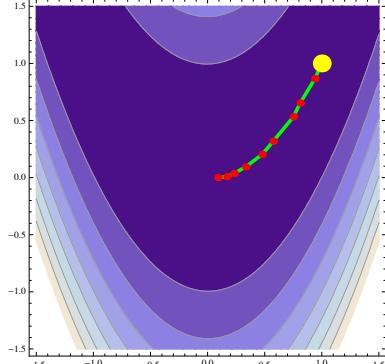
輸出 FindMinimum 求解 $f[2]$ 的迭代計算過程

```
In[6]: solf2=Reap@FindMinimum[f[2], Transpose@{vars[2]}, Table[0, {Length@vars[2]}]],  
StepMonitor:>Sow[vars[2]]];
```

輸出求解步驟圖

```
In[7]: ContourPlot[f[2], {x1, -1.5, 1.5}, {x2, -1.5, 1.5},  
Epilog -> {Green, Thickness[0.01], Line[solf2[[2, 1]]],  
Red, PointSize[Large], Point[solf2[[2, 1]]],  
Yellow, PointSize[0.05], Point[solf2[[2, 1, -1]]]}]
```

Out[7]:



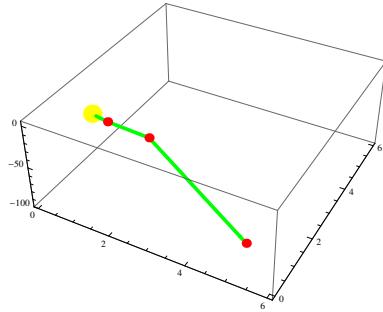
輸出 FindMinimum 求解 $f[3]$ 的迭代計算過程

```
In[8]: solf3=Reap@FindMinimum[f[3], Transpose@{vars[3]},  
Table[10, {Length@vars[3]}]], StepMonitor:>Sow[vars[3]]];
```

輸出求解步驟圖

```
In[9]: Show[ListPointPlot3D[solf3[[2, 1]]],  
Graphics3D[{Red, PointSize[0.025], Point[solf3[[2, 1]]]},  
{Green, Thickness[0.01], Line[solf3[[2, 1]]]},  
{Yellow, PointSize[0.05], Point[solf3[[2, 1, -1]]]}],  
PlotRange -> {{0, 6}, {0, 6}, {-100, 6}}]
```

Out[9]:



為了方便比較不同演算法異同，我們將上述寫成一個函數

函數 solf

```
In[10]: solf[n_,method_]:=Reap@FindMinimum[f[n],
Transpose@{vars[n],Table[0,{Length@vars[n]}]},EvaluationMonitor:>Sow[vars[n]],Method->method]
```

定義 FindMinimum 使用的演算法

```
In[11]: method={Automatic,"Gradient","ConjugateGradient","Newton","QuasiNewton",
"LevenbergMarquardt"};
```

使用不同演算法分別進行求解

```
In[12]: solf3method=solf[3,#]&/@method;
```

輸出不同演算法計算結果

```
In[13]: TableForm[Flatten@{#[[1,1]],[x1,x2,x3]/.#[[1,2]],
Length@#[[2,1]]}&/@solf3method,
TableHeadings->{method,{"optimal","x1","x2","x3","Iterations"}}]
```

Out[13]:

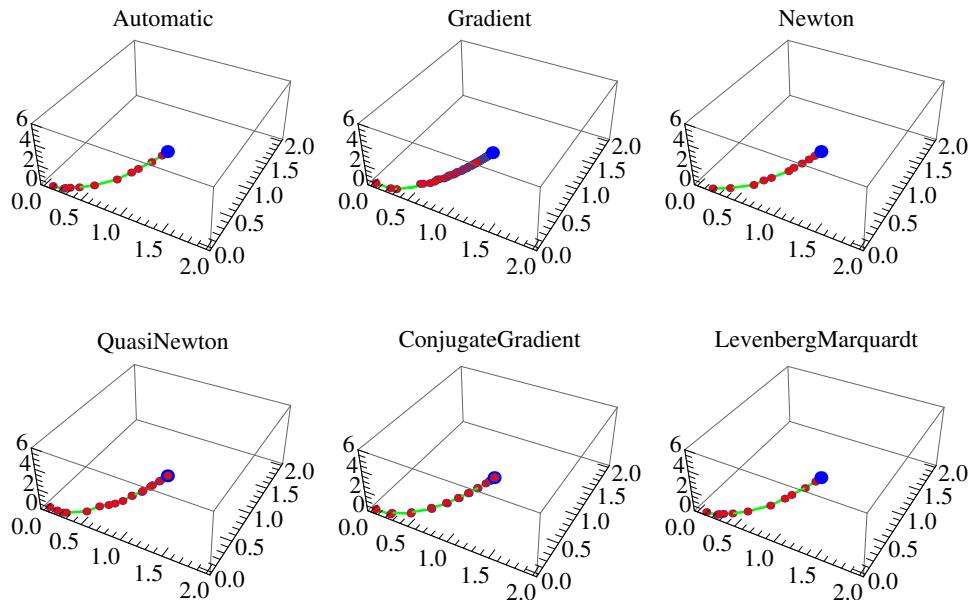
	optimal	x1	x2	x3	Iterations
Automatic	8.34935×10^{-27}	1.	1.	1.	14
Gradient	0.00058663	0.989183	0.978805	0.957742	100
Newton	2.47752×10^{-30}	1.	1.	1.	16
QuasiNewton	5.74481×10^{-21}	1.	1.	1.	27
ConjugateGradient	2.042291×10^{-19}	1.	1.	1.	47
LevenbergMarquardt	8.34935×10^{-27}	1.	1.	1.	14

由於目標函數包含平方和，故 FindMinimum 預設是以 LevenbergMarquardt 法進行求解。由輸出資料可知經由 14 次迭代計算後得 $(x_1^*, x_2^*, x_3^*) = (1, 1, 1)$ 。

輸出不同演算法的求解步驟圖

```
In[14]: GraphicsGrid[Partition[Labeled[#, [[2]], #, [Top]] & /@ Transpose[{method, Flatten@{Show[ListPointPlot3D[#[[2,1]], Graphics3D[{Red, PointSize[0.025], Point[#[[2,1]]], Green, Thickness[0.01], Line[#[[2,1]]], Blue, PointSize[0.05], Point[#[[2,1,-1]]]}], PlotRange -> {{0,2},{0,2},{0,6}}] & /@ solf3method]}], 3]]
```

Out[14]:



$n = 2, 3, \dots, 10$ 時 $f(x)$ 的計算結果

```
In[15]: TableForm[Flatten@solf[#,Automatic][[1]]&/@Range[2,10]/.Rule[x_,y_]:>y,  
TableHeadings->{Range[2,10],Flatten@>{"optimal",vars[10]}}]
```

範例 4-6. 以 $y = a + bx$ 及 $y = a + bx + cx^2$ 擬合下列資料

x	4	4	7	7	8	9	10	10	10	11	11	12	12	12	13	13	13	14		
y	2	10	4	22	16	10	18	26	34	17	28	14	20	24	28	26	34	46	26	
x	14	14	14	15	15	15	16	16	17	17	17	18	18	18	19	19	19	20	20	
y	36	60	80	20	26	54	32	40	32	40	50	42	56	76	84	36	46	68	32	48
x	20	20	20	22	23	24	24	24	24	25										
y	52	56	64	66	54	70	92	93	120	85										

輸入資料

```
In[1]: data={{4,2},{4,10},{7,4},{7,22},{8,16},{9,10},{10,18},{10,26},{10,34},
{11,17},{11,28},{12,14},{12,20},{12,24},{12,28},{13,26},{13,34},
{13,34},{13,46},{14,26},{14,36},{14,60},{14,80},{15,20},{15,26},
{15,54},{16,32},{16,40},{17,32},{17,40},{17,50},{18,42},{18,56},
{18,76},{18,84},{19,36},{19,46},{19,68},{20,32},{20,48},{20,52},
{20,56},{20,64},{22,66},{23,54},{24,70},{24,92},{24,93},{24,120},
{25,85}};
```

建立 $y = a + bx$ 的最小平方法目標函數

```
In[2]: obj1=Total[(a+b#[[1]]-#[[2]])^2&/@data];
```

建立 $y = a + bx$ 的最小絕對值法目標函數

```
In[3]: obj2=Total[Abs[(a+b#[[1]]-#[[2]])]&/@data];
```

使用微分求解點 (x_0, y_0) 到直線 $y = a + bx$ 的最短距離平方

```
In[4]: ({x0,y0}-{x, y}).({x0,y0}-{x, y})/.
```

```
Solve[{-b*x+y-a==0,y-y0==(x-x0)*(-1/b)}, {x,y}] [[1]]//Simplify
```

```
Out[4]: 
$$\frac{(a + bx_0 - y_0)^2}{1 + b^2}$$

```

使用 Minimize 求解點 (x_0, y_0) 到直線 $y = a + bx$ 的最短距離平方

```
In[5]: Minimize[SquaredEuclideanDistance[{x0,y0},{x,a+b*x}]/.
```

```
Abs[z_]:=z,x] [[1]]//Simplify
```

```
Out[5]: 
$$\frac{(a + bx_0 - y_0)^2}{1 + b^2}$$

```

建立 $y = a + bx$ 的最短距離法目標函數

```
In[6]: obj3=Total[(a+b#[[1]]-#[[2]])^2/(1+b^2)&/@data];
```

建立 $y = a + bx + cx^2$ 最小平方法目標函數

```
In[7]: obj4=Total[(a+b#[[1]]+c#[[1]]^2-#[[2]])^2&/@data];
```

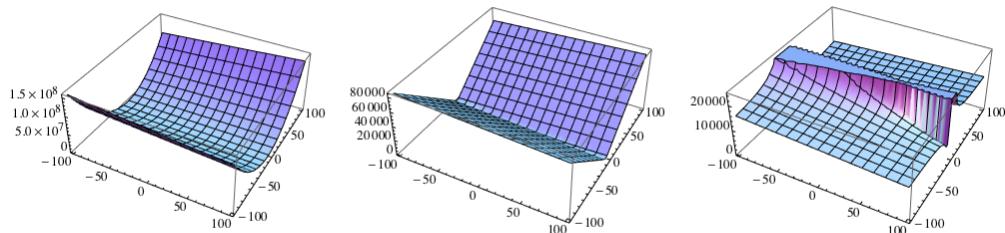
建立 $y = a + bx + cx^2$ 最小絕對值法目標函數

```
In[8]: obj5=Total[Abs[a+b#[[1]]+c#[[1]]^2-#[[2]]]&/@data];
```

輸出 obj1、obj2 和 obj3 之 3D 圖

```
In[9]: GraphicsGrid[{Plot3D[#, {a, -100, 100}, {b, -100, 100}]&/@{obj1,obj2,obj3}},  
ImageSize->750]
```

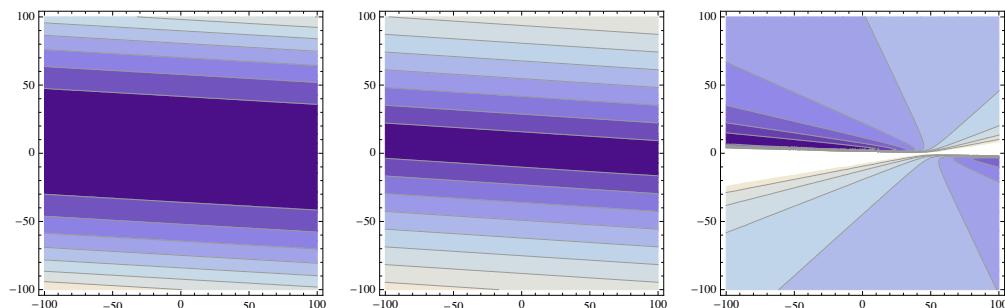
Out[9]:



輸出 obj1、obj2 和 obj3 之等高線圖

```
In[10]: GraphicsGrid[{ContourPlot[#, {a, -100, 100}, {b, -100, 100}]&@  
{obj1,obj2,obj3}},ImageSize->750]
```

Out[10]:



由 obj1、obj2 及 obj3 的等高線圖可發現 obj1 和 obj2 具有唯一的臨界點，而 obj3 有兩個臨界點。

使用 `Solve` 求解目標函數 `obj1` 的臨界點

`In[11]: Solve[D[obj1, {{a, b}}] == {0, 0}, {a, b}]`

`Out[11]:` $\left\{ \left\{ a \rightarrow -\frac{301042}{17125}, b \rightarrow \frac{26937}{6850} \right\} \right\}$

計算海賽矩陣各階行列式值

`In[12]: Det[D[obj1, {{a, b}}, 2] [[1; ; #, 1; ; #]] &/@{1, 2}]`

`Out[12]:` {100, 274000}

由於海賽矩陣各階行列式值階為正數，故可知以最小平方法估計的目標函數 `obj1` 有全域域最小值。

使用 `Solve` 求解目標函數 `obj3` 的臨界點

`In[13]: Solve[D[obj3, {{a, b}}] == {0, 0}, {a, b}]`

`Out[13]:` $\left\{ \left\{ a \rightarrow \frac{7(-67069 - 11\sqrt{33567951521})}{299300}, b \rightarrow \frac{173161 + \sqrt{33567951521}}{59860} \right\}, \left\{ a \rightarrow \frac{7(-67069 + 11\sqrt{33567951521})}{299300}, b \rightarrow \frac{173161 - \sqrt{33567951521}}{59860} \right\} \right\}$

計算海賽矩陣各階行列式值

`In[14]: (Det[D[obj3, {{a, b}}, 2] [[1; ; #, 1; ; #]] &/@{1, 2}) /. N@%`

`Out[14]:` {{2.74392, 136.264}, {97.2561, -6.06758 × 10⁶}}

由傳回各臨界點的海賽矩陣可知，以最短距離法估計的目標函數 `obj3` 有全域域最小值，此外第二個臨界點並非極值而是鞍點。

使用 `Minimize` 求解 `obj1`

`In[15]: Minimize[obj1, {a, b}]`

`Out[15]:` $\left\{ \frac{194429048}{17125}, \left\{ a \rightarrow -\frac{301042}{17125}, b \rightarrow \frac{26937}{6850} \right\} \right\}$

使用 `Minimize` 求解 `obj3`

`In[16]: Minimize[obj3, {a, b}]`

`Out[16]:` $\left\{ \frac{1}{100} (1695449 - 9\sqrt{33567951521}), \left\{ a \rightarrow -\frac{7(67069 + 11\sqrt{33567951521})}{299300}, b \rightarrow \frac{59860}{\sqrt{33567951521} - 173161} \right\} \right\}$

由於 obj2 中含有絕對值在某些點上可能無法微分，因此 obj2 設定以 PrincipalAxis 法計算。

以 FindMinimum 估計 obj1、obj2 和 obj3

```
In[17]: TableForm[Flatten@#,FindMinimum[ToExpression@#,If[#= "obj2",
{ {a,-10,10},{b,-10,10}},{{a,0},{b,0}}]]]&/@{"obj1","obj2","obj3"}]
```

```
Out[17]: obj1 11353.5 a → -17.5791 b → 3.93241
          obj2 563.856 a → -11.712 b → 3.40467
          obj3 465.087 a → -48.7039 b → 5.9535
```

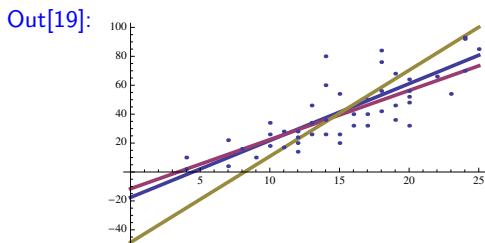
輸出迴歸方程式

```
In[18]: myfit1=a+b*x/.%[[All,{3,4}]]
```

```
Out[18]: {-17.5791 + 3.93241x, -11.6 + 3.4x, -48.7039 + 5.9535x}
```

輸出迴歸線與散佈圖

```
In[19]: Plot[Tooltip[myfit1],{x,0,25},PlotStyle->Thickness[0.01],Epilog->Point[data]]
```



使用 Solve 求解目標函數 obj4 的臨界點

```
In[20]: Solve[D[obj4,{ {a,b,c}}]=={0,0,0},{a,b,c}]
```

```
Out[20]: { {a → 746242097/302105454, b → 413863754/453158181, c → 90594751/906316362}}
```

計算海賽矩陣各階行列式值

```
In[21]: Det[D[obj4,{ {a,b,c}},2][[{1;#1;#2}]]]&/@{1,2,3}
```

```
Out[21]: {100, 274000, 29002123584}
```

由於海賽矩陣各階行列式值階為正數，故可知以最小平方法估計的目標函數 obj4 也具有全域域最小值。

以 Minimize 進行 $y = a + bx + cx^2$ 的最小平方法參數估計

```
In[22]: Minimize[obj4,{a,b,c}]
```

Out[22]: $\left\{ \frac{9810617141123}{906316362}, \left\{ a \rightarrow \frac{746242097}{302105454}, b \rightarrow \frac{413863754}{453158181}, c \rightarrow \frac{90594751}{906316362} \right\} \right\}$

以 `FindMinimum` 估計 `obj4` 和 `obj5`

In[23]: `TableForm[Flatten@#, FindMinimum[ToExpression@#, If[#=="obj5", {{a,-10,10},{b,-10,10},{c,-10,10}},{{{a,0},{b,0},{c,0}}}]]&/@{"obj4","obj5"}]`

Out[23]:

obj4	10824.7	a → 2.47014	b → 0.913288	c → 0.0999593
obj5	5733.36	a → -1225.08	b → 164.957	c → -5.09999

事實上，上述 `obj5` 所得之計算結果並非最佳解。為了避免起始值導致 `FindMinimum` 計算收斂到局部極值，我們利用以下模擬的方式來求得最佳解。

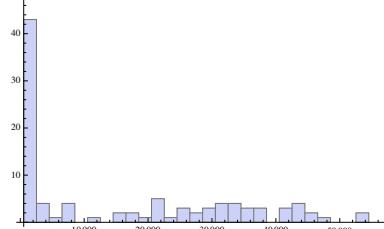
利用隨機的起始值，重複求解 100 次

In[24]: `obj5sim=Sort[With[{z:=RandomReal[{-100,100}]}, FindMinimum[obj5,{a,z,z},{b,z,z},{c,z,z}]]&/@Range[100]];`

輸出模擬資料的直方圖

In[25]: `Histogram[obj5sim[[All,1]], {Min@obj5sim[[All,1]],Max@obj5sim[[All,1]],2000}]`

Out[25]:



輸出模擬資料所得之最佳解

In[26]: `obj5sim[[1]]`

Out[26]: `{538.479, {a → 9.64291, b → -0.467873, c → 0.139286}}`

輸出迴歸方程式

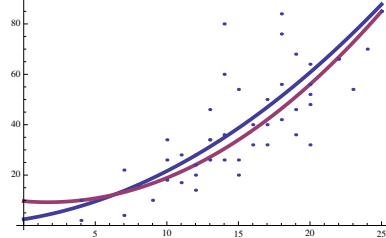
In[27]: `myfit2=a+b*x+c*x^2/.{N@Out[22][[2]],obj5sim[[1,2]]}`

Out[27]: `{0.0999593x^2 + 0.913288x + 2.47014, 0.139286x^2 - 0.467873x + 9.64291}`

輸出迴歸線與散佈圖

In[28]: Plot[Tooltip[myfit2], {x, 0, 25}, PlotStyle -> Thickness[0.01], Epilog -> Point[data]]

Out[28]:



4.2 限制條件下非線性規劃

4.2.1 等式限制條件下非線性規劃

假設 $\mathbf{x} = \{x_1, x_2, \dots, x_n\}$, 考慮以下問題

$$\min \text{ or } \max f(\mathbf{x})$$

受制於

$$h_i(\mathbf{x}) = 0, i = 1, 2, \dots, m$$

拉格朗日 (Joseph Lagrange) 提出將一個有 n 個變數與 m 個限制條件的最佳化問題轉換為一個有 $n + m$ 個變數的不受限制最佳化問題。由於此法引入一個未知的純量 (拉格朗日乘數)，故稱為拉格朗日乘數法。利用拉格朗日乘數法 (Lagrange multiplier method)，上述目標函數可改寫為無限制式之最佳化問題如下：

$$L(\mathbf{x}, \lambda) = f(\mathbf{x}) - \sum_{j=1}^m \lambda_j h_j(\mathbf{x}) \quad (4-26)$$

因此，極大或極小值的一階必要條件 (First Order Necessary Conditions, FONC) 為

$$\frac{\partial L(\mathbf{x}, \lambda)}{\partial x_i} = \frac{\partial f(\mathbf{x})}{\partial x_i} - \sum_{j=1}^m \lambda_j \frac{\partial h_j(\mathbf{x})}{\partial x_i} = 0, i = 1, 2, \dots, n \quad (4-27)$$

及

$$\frac{\partial L(\mathbf{x}, \lambda)}{\partial \lambda_i} = -h_i(\mathbf{x}) = 0, i = 1, 2, \dots, m \quad (4-28)$$

(4-27) 和 (4-28) 可以矩陣表示可改寫為

$$\frac{\partial L(\mathbf{x}, \lambda)}{\partial \mathbf{x}} = \nabla f(\mathbf{x}) - \boldsymbol{\lambda} \nabla h(\mathbf{x}) = \mathbf{0} \quad (4-29)$$

及

$$\frac{\partial L(x, \lambda)}{\partial \lambda} = -h(x) = 0 \quad (4-30)$$

由 (4-29) 可發現，最佳解 x^* 的梯度 $\nabla f(x^*)$ 與限制式 $h(x^*)$ 的梯度 $\nabla h(x^*)$ 必然是平行關係。此外，由於 $h(x^* + \varepsilon) \approx h(x^*) + \varepsilon^\top \nabla h(x^*)$ ，故當 $\varepsilon \rightarrow 0$ 時， $\varepsilon^\top \nabla h(x^*) = 0$ ，即 $h(x^*)$ 在 x^* 的瞬間行進方向與其梯度垂直。又 $\nabla f(x^*)$ 與 $\nabla h(x^*)$ 平行，因此 h 在 x^* 的瞬間行進方向也與 $\nabla f(x^*)$ 垂直，即為梯度垂直定理 (The orthogonal gradient theorem)。

在求得極值後，極值為局部極大值或是局部極小值則必須藉由以下鑲邊海賽矩陣 (Bordered Hessian matrix) 的正定或負定矩陣來決定

$$H_B = \nabla^2 L(\lambda, x) = \begin{bmatrix} 0 & P \\ P^T & Q \end{bmatrix}_{(n+m) \times (n+m)} \quad (4-31)$$

其中

$$P = \begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \frac{\partial h_1}{\partial x_2} & \cdots & \frac{\partial h_1}{\partial x_n} \\ \frac{\partial h_2}{\partial x_1} & \frac{\partial h_2}{\partial x_2} & \cdots & \frac{\partial h_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial h_m}{\partial x_1} & \frac{\partial h_m}{\partial x_2} & \cdots & \frac{\partial h_m}{\partial x_n} \end{bmatrix} = \frac{\partial h(x)}{\partial x} = \nabla h(x)$$

及

$$Q = \begin{bmatrix} \frac{\partial^2 L}{\partial x_1^2} & \frac{\partial^2 L}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 L}{\partial x_1 \partial x_n} \\ \frac{\partial^2 L}{\partial x_2 \partial x_1} & \frac{\partial^2 L}{\partial x_2^2} & \cdots & \frac{\partial^2 L}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 L}{\partial x_n \partial x_1} & \frac{\partial^2 L}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 L}{\partial x_n^2} \end{bmatrix}$$

由 H_B 可以發現，由於前 $2m - 1$ 階的子行列式皆為 0，而第 $2m$ 階的子矩陣並未包含有函數 $f(x)$ 的資訊，故要探討 H_B 為正定或負定矩陣，僅需就後面的 $n + m - 2m = n - m$ 個子矩陣討論即可。若 H_B 最後 $n - m$ 個子行列式符號皆為 $(-1)^{m+1}$ ，則 H_B 為一負定矩陣，故此極值為局部極大值；相對的若 H_B 最後 $n - m$ 個子行列式符號皆為 $(-1)^m$ ，則 H_B 為一正定矩陣，故此極值為局部極小值。

然而上述的方式對於大規模的問題在充分條件的檢驗過於繁瑣，我們也可利用正負定矩陣的性質來做為檢驗二階條件的工具。假設

$$\Delta = \begin{bmatrix} 0 & P \\ P^T & Q - uI \end{bmatrix} \quad (4-32)$$

若方程式 $\det(\Delta) = 0$ 的根皆為負數，則此極值為局部極大值；相反的，若方程式 $\det(\Delta) = 0$ 的根皆為正數，則此極值為局部極小值。

4.2.2 不等式限制條件下非線性規劃

假設 $\mathbf{x} = \{x_1, x_2, \dots, x_n\}$, 考慮以下問題

$$\min \text{ or } \max f(\mathbf{X})$$

受制於

$$g_i(\mathbf{X}) \leq 0, i = 1, 2, \dots, p$$

$$h_i(\mathbf{X}) = 0, i = 1, 2, \dots, q$$

將不等限制式 $g(x)$ 加上鬆弛變數 (Slack variable), 利用拉格朗日乘數法, 目標函數可改寫為無限制式之最佳化問題如下:

$$L(x, \lambda, u, S) = f(x) - \sum_{i=1}^p \lambda_i [g_i(x) + S_i^2] - \sum_{j=1}^q u_j h_j(x)$$

因此, 極值的一階必要條件 (First Order Necessary Conditions, FONC) 為

$$\frac{\partial L(x, \lambda, u, S)}{\partial x_i} = \frac{\partial f(x)}{\partial x_i} - \sum_{i=1}^p \lambda_i \frac{\partial g_i(x)}{\partial x_i} - \sum_{i=1}^q u_i \frac{\partial h_i(x)}{\partial x_i} = 0, i = 1, 2, \dots, n \quad (4-33)$$

$$\frac{\partial L(x, \lambda, u, S)}{\partial \lambda_i} = -[g_i(x) + S_i^2] = 0, i = 1, 2, \dots, p \quad (4-34)$$

$$\frac{\partial L(x, \lambda, u, S)}{\partial u_i} = -h_i(x) = 0, i = 1, 2, \dots, q \quad (4-35)$$

及

$$\frac{\partial L(x, \lambda, u, S)}{\partial S_i} = -2\lambda_i S_i = 0, i = 1, 2, \dots, p \quad (4-36)$$

由 (4-36) 可知 λ_i 與 S_i 不可同時為 0, 故可將 (4-33)–(4-36) 改寫為

$$\nabla f(x) - \lambda^\top \nabla g(x) - u^\top \nabla h(x) = \mathbf{0} \quad (4-37)$$

$$h(x) = \mathbf{0} \quad (4-38)$$

$$g(x) \leq 0 \quad (4-39)$$

$$\lambda_i g_i(x) = 0, i = 1, 2, \dots, p \quad (4-40)$$

$$\begin{cases} \lambda_i \geq 0, i = 1, 2, \dots, p & \text{局部極大值} \\ \lambda_i \leq 0, i = 1, 2, \dots, p & \text{局部極小值} \end{cases} \quad (4-41)$$

我們稱以上條件為 Karush-Kuhn-Tucker 最佳化條件, 簡稱 KKT 條件。(4-38) 和 (4-39) 表示滿足 KKT 條件的解必是一組可行解; (4-37) 則表示若 x 為一最佳解, $\nabla f(x)$ 必須是 $\nabla g(x)$ 和 $\nabla h(x)$ 之線性組合; (4-40) 則為互補差餘定理 (Complementary slackness Theorem)。最後, 由於 λ 表示資源 $g(x)$ 對目標函數的影響, 當資源 $g(x)$ 增加時, 由於可行區域變大導致函數 $f(x)$ 可能增加, 故對最大化問題而言, λ_i 必須為非負數, 即 $\lambda \geq 0$; 同理, 對最小化問題而言, $\lambda \leq 0$ 。而等號限制式 $h_i(x)$ 並無方向性, 故 u 符號不限制。

若假設 s_i 必須為非負數，利用對數障礙函數及拉格朗日乘數法，原題可改寫為

$$L(x, \lambda_g, \lambda_h, s) = f(x) - u \sum_{i=1}^p \ln s_i - \sum_{i=1}^p \lambda_{g_i} [g_i(x) + s_i] - \sum_{i=1}^q \lambda_{h_i} h_i(x)$$

其中 $u > 0$ 為障礙參數。故極值的一階必要條件為

$$\frac{\partial L(x, \lambda_g, \lambda_h, S)}{\partial x_i} = \frac{\partial f(x)}{\partial x_i} - \sum_{i=1}^p \lambda_{g_i} \frac{\partial g_i(x)}{\partial x_i} - \sum_{i=1}^q \lambda_{h_i} \frac{\partial h_i(x)}{\partial x_i} = 0, i = 1, 2, \dots, n \quad (4-42)$$

$$\frac{\partial L(x, \lambda_g, \lambda_h, S)}{\partial \lambda_{g_i}} = -[g_i(x) + s_i] = 0, i = 1, 2, \dots, p \quad (4-43)$$

$$\frac{\partial L(x, u, \lambda, S)}{\partial \lambda_{h_i}} = -h_i(x) = 0, i = 1, 2, \dots, q \quad (4-44)$$

$$\frac{\partial L(x, \lambda_g, \lambda_h, S)}{\partial s_i} = -\frac{u}{s_i} - \lambda_{g_i}, i = 1, 2, \dots, p \quad (4-45)$$

(4-42)–(4-45) 可以矩陣表示可改寫為

$$\nabla f(x) - \lambda_g^\top \nabla g(x) - \lambda_h^\top \nabla h(x) = \mathbf{0} \quad (4-46)$$

$$-(g(x) + s) = \mathbf{0} \quad (4-47)$$

$$-h(x) = \mathbf{0} \quad (4-48)$$

$$-ue - S\lambda_g = \mathbf{0} \quad (4-49)$$

其中 S 為對角線為 s_i 之矩陣且 e 為所有元素為 1 的矩陣。(4-46)–(4-49) 為一非線性聯立方程組，將 (4-49) 改寫 $-ue - S\lambda_g = \mathbf{0}$ 後，我們可用牛頓法求解步長如下：

$$\begin{bmatrix} H(x, \lambda_g, \lambda_h) & -\nabla g(x) & -\nabla h(x) & \mathbf{0} \\ -\nabla g(x) & \mathbf{0} & \mathbf{0} & I \\ -\nabla h(x) & \mathbf{0} & \mathbf{0} & 0 \\ \mathbf{0} & -S & \mathbf{0} & -\Lambda_g \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda_g \\ \Delta \lambda_h \\ \Delta s \end{bmatrix} = -\begin{bmatrix} \nabla f(x) - \lambda_g^\top \nabla g(x) - \lambda_h^\top \nabla h(x) \\ -(g(x) + s) \\ -h(x) \\ -ue - S\lambda_g \end{bmatrix} \quad (4-50)$$

其中 $H(x, \lambda_g, \lambda_h) = \nabla^2 f(x) - \lambda_g^\top \nabla^2 g(x) - \lambda_h^\top \nabla^2 h(x)$ 及 Λ_g 為對角線為 λ_i 之矩陣。又 $\Delta s = -u\Lambda_g^{-1}e - \Lambda_g^{-1}S\lambda_g - \Lambda_g^{-1}S\Delta x$ ，故 (4-50) 可改寫為

$$\begin{bmatrix} H(x, \lambda_g, \lambda_h) & -\nabla g(x) & -\nabla h(x) \\ -\nabla g(x) & \mathbf{0} & \mathbf{0} \\ -\nabla h(x) & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda_g \\ \Delta \lambda_h \end{bmatrix} = -\begin{bmatrix} \nabla f(x) - \lambda_g^\top \nabla g(x) - \lambda_h^\top \nabla h(x) \\ -(g(x) + s) + \Delta s \\ -h(x) \end{bmatrix} \quad (4-51)$$

在求得 Δx 、 $\Delta \lambda_g$ 、 $\Delta \lambda_h$ 和 Δs 後，即可以直線搜尋法求得 x 、 λ_g 、 λ_h 和 s 下一步的位置，即

$$x_{k+1} = x_k + \alpha_x \Delta x_k$$

$$\lambda_{g,k+1} = \lambda_{g,k} + \alpha_{\lambda_g} \Delta \lambda_{g,k}$$

$$\lambda_{h,k+1} = \lambda_{h,k} + \alpha_{\lambda_h} \Delta \lambda_{h,k}$$

$$s_{k+1} = s_k + \alpha_s \Delta s_k$$

以上這種種結合障礙函數和 Karush-Kuhn-Tucker 最佳化條件，並利用牛頓法求解的限制條件下最佳化方法稱為內點法 (Interior Point method)，也是 Mathematica 求解受限制最佳化問題時的預設方法。

然而，事實上 Karush-Kuhn-Tucker 最佳化條件一般主要的用途並非作為求解限制條件下非線性規劃的主要方法，通常求解還是以數值搜尋法為主。主要原因是求解 KKT 條件的非線性聯立方程組通常是一個相當困難的問題。因此，Karush-Kuhn-Tucker 最佳化條件反而是在以數值搜尋法求得到一數值解之後，用來檢驗此數值解是否為最佳解的一種方法。

4.2.3 範例

Mathematica 求解受限制最佳化函數同樣可以使用 Minimize, Maximize, FindMinimum 和 FindMaximum。Minimize 和 Maximize 適合用在變數較少及最佳解具解析解的情況下，此函數通常僅在線性或為多項式函數時才可求得。在一般情況下，多數的受限制最佳化問題都只能透過數值分析的方法求解近似值，此時就必須改用 FindMinimum 和 FindMaximum。以上四個最佳化求解函數的使用方式介紹如下：

1. `Minimize[{f(x), 限制式}, x]`: 以 x 為變數求解函數 $f(x)$ 在限制條件下的全域極小值。
2. `Maximize[{f(x), 限制式}, x]`: 以 x 為變數求解函數 $f(x)$ 在限制條件下的全域極大值。
3. `Minimize[{f(x1, x2, ..., xn), 限制式}, x]`: 以 x_1, x_2, \dots, x_n 為變數求解函數 $f(x)$ 在限制條件下的全域極小值。
4. `Maximize[{f(x1, x2, ..., xn), 限制式}, x]`: 以 x_1, x_2, \dots, x_n 為變數求解函數 $f(x)$ 在限制條件下的全域極大值。
5. `FindMinimum[{f(x), 限制式}, x]`: 以 x 為變數求解函數 $f(x)$ 的局部極小值。
6. `FindMinimum[{f(x), 限制式}, {x, x0}]`: 以 $x = x_0$ 為起始值求解函數 f 的局部極小值。
7. `FindMinimum[{f(x1, x2, ..., xn), 限制式}, {x1, x2, ..., xn}]`: 以 x_1, \dots, x_n 為變數求解 f 的局部極小值。
8. `FindMinimum[{f(x1, x2, ..., xn), 限制式}, {x1, x10}, ..., {xn, xn0}]`: 以 (x_{10}, \dots, x_{n0}) 為起始值求解 f 的局部極小值。
9. `FindMaximum[{f(x), 限制式}, x]`: 以 x 為變數求解函數 $f(x)$ 的局部極大值。
10. `FindMaximum[{f(x), 限制式}, {x, x0}]`: 以 $x = x_0$ 為起始值求解函數 f 的局部極大值。

10. `FindMaximum[{f(x1, x2, ..., xn), 限制式}, {x1, x2, ..., xn}]`: 以 x₁, ..., x_n 為變數求解 f 的局部極大值。

11. `FindMaximum[{f(x1, x2, ..., xn), 限制式}, {x1, x10}, ..., {xn, xn0}]`: 以 (x₁₀, ..., x_{n0}) 為起始值求解 f 的局部極大值。

範例 4-7. 假設 $x + y + z = 10$, $xyz - 23x = 40$ 且 $x \geq y \geq z$, 求 $|x| + |y| + |z|$ 之極小值。

利用 `Solve` 化簡 y 和 z

`In[1]: Solve[{x+y+z==10, x*y*z-23x==40}, {y, z}] // FullSimplify`

`Out[1]:` $\left\{ \begin{array}{l} y \rightarrow 5 - \frac{x}{2} - \frac{\sqrt{(-20+x)(8+x^2)}}{2\sqrt{x}}, z \rightarrow 5 - \frac{x}{2} + \frac{\sqrt{(-20+x)(8+x^2)}}{2\sqrt{x}} \\ y \rightarrow 5 - \frac{x}{2} + \frac{\sqrt{(-20+x)(8+x^2)}}{2\sqrt{x}}, z \rightarrow 5 - \frac{x}{2} - \frac{\sqrt{(-20+x)(8+x^2)}}{2\sqrt{x}} \end{array} \right\}$

因為 x, y 和 z 皆為實數, 由輸出結果可知 $x \geq 20$ ($x < 20$ 不合)。又由限制條件可知, $y + z < 0$ 且 $yz > 0$, 故可知 y 和 z 需皆為負數。我們也可直接利用 `Reduce` 求解可行解區域。

利用 `Reduce` 化簡 y 和 z

`In[2]: FullSimplify@Reduce[{x+y+z==10, x*y*z-23x==40, x>=y, y>=z}, {y, z},`

`Real, Backsubstitution->True] /. {And->List, Or->List, Equal->Rule}`

`Out[2]:` $\left\{ x \geq 20, y \rightarrow \frac{1}{2} \left(10 - x + \sqrt{\frac{(-20+x)(8+x^2)}{x}} \right), z \rightarrow 5 - \frac{x}{2} - \frac{1}{2} \sqrt{\frac{(-20+x)(8+x^2)}{x}} \right\}$

由以上結果可知, 目標函數可改寫為 $x - y - z$ 。

定義 Lagrange 函數

`In[3]: L=x-y-z+\lambda [Lambda]1(x+y+z-10)+\lambda [Lambda]2(x*y*z-23x-40)`

利用 `Solve` 求解最佳解

`In[4]: ans=Solve[D[L, {{\lambda [Lambda]1, \lambda [Lambda]2, x, y, z}}]=={0, 0, 0, 0, 0},`

`{\lambda [Lambda]1, \lambda [Lambda]2, x, y, z}, Real] [[1]] // FullSimplify`

`Out[4]:` $\left\{ \left\{ \lambda_1 \rightarrow -\frac{49}{51}, \lambda_2 \rightarrow -\frac{1}{51}, x \rightarrow 20, y \rightarrow -5, z \rightarrow -5 \right\} \right\}$

因為變數個數 $n = 3$ 和限制條件個數 $m = 2$, 所以只需檢測鑲邊海賽矩陣最後一個主行列式之行列值符號。

傳回鑲邊海賽矩陣最後一個主行列式之行列值

In[5]: `Det@D[L, {{\Lambda1, \Lambda2, x, y, z}, 2}] /. ans`

Out[5]: 8160

最後一個主行列式之行列值為正號,由於 $(-1)^m = (-1)^2 = 1$,故全域極小值 $x-y-z = 20 - (-5) - (-5) = 30$ 。

利用 `FindMinimum` 求解最佳解

In[6]: `FindMinimum[{Abs[x] + Abs[y] + Abs[z], x + y + z == 10, x*y*z - 23x == 40}, {x, y, z}]`

Out[6]: {30., {x → 20., y → -5., z → -5.}}

利用 `Minimize` 求解最佳解

In[7]: `Minimize[{Abs[x] + Abs[y] + Abs[z], x + y + z == 10, x*y*z - 23x == 40, x >= y >= z}, {x, y, z}]`

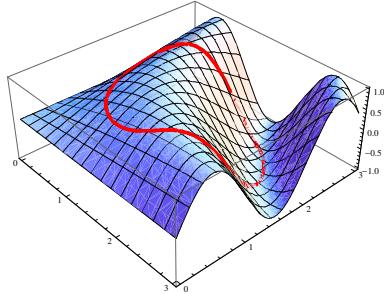
Out[7]: {30, {x → 20, y → -5, z → -5}}

範例 4-8. 求解 $f(x, y) = \sin(xy)$ 在 $(x - 3/2)^2 + (y - 3/2)^2 = 1$ 上的局部極值。

輸出函數及限制式圖形

In[1]: `Show[Plot3D[Evaluate[Sin[x*y]], {x, 0, 3}, {y, 0, 3}], Graphics3D[{Thickness[0.01], Red, Line /@ Partition[{{#[[1]], #[[2]], Sin[#[[1]]*#[[2]]]} & @ ContourPlot[(x - 3/2)^2 + (y - 3/2)^2 == 1, {x, 0, 3}, {y, 0, 3}][[1, 1]], 2, 1]}]]`

Out[1]:



由圖形可知本題所求即為求解紅色曲線上的局部極值。因為限制條件 $\left(x - \frac{3}{2}\right)^2 + \left(y - \frac{3}{2}\right)^2 = 1$ 可將 y 化簡為 x 的函數,故可將原本兩變數的問題化簡為單變數求極值的問題。

由限制條件化簡 y

In[2]: `yx=Flatten@Solve[(x-3/2)^2+(y-3/2)^2==1,y]`

Out[2]: $\left\{ y \rightarrow \frac{1}{2} \left(3 - \sqrt{-5 + 12x - 4x^2} \right), y \rightarrow \frac{1}{2} \left(3 + \sqrt{-5 + 12x - 4x^2} \right) \right\}$

所以目標函數可改寫為

$$f(x) = \begin{cases} \sin \left(\frac{1}{2}x \left(3 - \sqrt{-5 + 12x - 4x^2} \right) \right), & y = \frac{1}{2} \left(3 - \sqrt{-5 + 12x - 4x^2} \right) \\ \sin \left(\frac{1}{2}x \left(3 + \sqrt{-5 + 12x - 4x^2} \right) \right), & y = \frac{1}{2} \left(3 + \sqrt{-5 + 12x - 4x^2} \right) \end{cases}$$

計算 x 的可行區域

In[3]: `Reduce[-5+12x-4x^2>=0,x]`

Out[3]: $\frac{1}{2} \leq x \leq \frac{5}{2}$

計算上式中 x 的臨界點

In[4]: `ansx=N@Reduce[{D[Sin[x*y/.#],x]==0,1/2<=x<=5/2},x]/.`

`{Or->List,Equal->Rule}]&/@yx`

Out[4]: $\{x \rightarrow 0.792893, x \rightarrow 2.14203\}, \{x \rightarrow 2.20711, x \rightarrow 1.9842, x \rightarrow 2.37496, x \rightarrow 0.733321\}$

計算臨界點座標

In[5]: `ans=Flatten[Table[{x,y/.yx[[k]]}/.#&@ansx[[k]],{k,1,2}],1]`

Out[5]: $\{0.792893, 0.792893\}, \{2.14203, 0.733321\}, \{2.20711, 2.20711\}, \{1.9842, 2.37496\}, \{2.37496, 1.9842\}, \{0.733321, 2.14203\}$

將臨界點代入計算最佳解

In[6]: `sol=Sin[#[[1]]*#[[2]]]&/@ans`

Out[6]: $\{0.588077, 1., -0.987397, -1., -1., 1.\}$

計算最佳解的二階條件

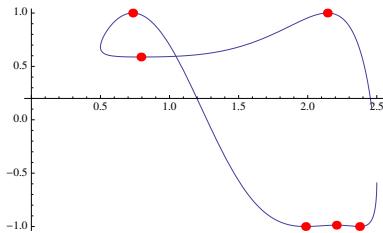
In[7]: `soc=Flatten@Table[D[Sin[x*y/.yx[[k]]],{x,2}]/.#&@ansx[[k]],{k,2}]`

Out[7]: $\{0.196249, -6.38622, -1.30451, 1.6305, 5.3241, -9.10666\}$

輸出目標函數及臨界點圖形

```
In[8]: Show[Plot[Sin[x*y]/.yx[[#]],{x,0,3]}&/@{1,2},PlotRange->All,
Epilog->{Red,PointSize[Large],Point[Transpose@{ans[[All,1]],sol}]}]
```

Out[8]:



輸出所有臨界點及對應的充分條件

```
In[9]: TableForm[Flatten[#]&@Transpose@{sol,ans,soc,
If[#[#<0,"Local Max","Local Min"]&@soc},
TableHeadings->{None,{Sol,x,y,SOC,"MinorMax"}}]
```

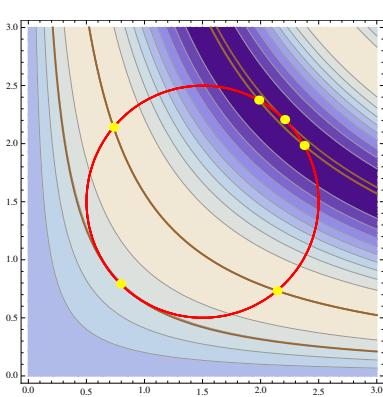
Out[9]:

Sol	x	y	SOC	Min or Max
0.588077	0.792893	0.792893	0.196249	Local Min
1.	2.14203	0.733321	-6.38622	Local Max
-0.987397	2.20711	2.20711	-1.30451	Local Max
-1.	1.9842	2.37496	1.6305	Local Min
-1.	2.37496	1.9842	5.3241	Local Min
1.	0.733321	2.14203	-9.10666	Local Max

輸出函數 $\sin(xy)$ 及限制條件之等高線圖

```
In[10]: Show[ContourPlot[Sin[x*y],{x,0,3},{y,0,3}],
ContourPlot[{(x-3/2)^2+(y-3/2)^2==1,x*y==#},{x,0,3},{y,0,3},
ContourStyle->{{Red,Thickness[0.005]},{Brown,Thickness[0.005]}}]&@(
If[#[#<0,#+2Pi,#]&@ArcSin@sol),
Epilog->{Yellow,PointSize[0.025],Point[ans]}]
```

Out[10]:



以上我們利用限制式將目標函數轉換為單變數最佳化問題方便求解，接下來我們將利用 Lagrange 乘數法

求解。

定義 Lagrange 函數

```
In[11]: L=Sin[x*y]+\[Lambda]*((x-3/2)^2+(y-3/2)^2-1);
```

傳回滿足一階條件的所有臨界點

```
In[12]: Lans=N@Reduce[Flatten@{Thread[D[L,{x,y,\[Lambda]}]==0],1/2< x<5/2,1/2< y<5/2},\{x,y,\[Lambda]\},Backsubstitution->True]/.{Or->List,And->List,Equal->Rule}]
```

```
Out[12]: {{x → 0.792893, y → 0.792893, λ → 0.453465}, {x → 2.20711, y → 2.20711, λ → -0.246995}, {x → 1.9842, y → 2.37496, λ → 0.}, {x → 2.37496, y → 1.9842, λ → 0.}, {x → 0.733321, y → 2.14203, λ → 0.}, {x → 2.14203, y → 0.733321, λ → 0.}}
```

最後，由於變數個數 $n = 2$ 和限制條件數 $m = 1$ ，所以只需檢測鑲邊海賽矩陣最後一個主行列式之行列值符號。若最後一個主行列式之行列值為正號， $(-1)^{m+1} = 1$ ，則為局部極大值；若行列式符號為負號， $(-1)^m = 1$ ，則為局部極小值。

傳回鑲邊海賽矩陣最後一個主行列式之行列值

```
In[13]: Lhessian=Det/@(D[L,{\\[Lambda]},x,y],2])/.Lans)
```

```
Out[13]: {-0.392498, 2.60901, -4.99291, -4.99291, 15.0152, 15.0152}
```

輸出臨界點等彙總

```
In[14]: TableForm[{L,x,y,\[Lambda]},Det[D[L,{\\[Lambda]},x,y],2]], If[Det[D[L,{\\[Lambda]},x,y],2]]>0,"Local Max","Local Min"]/.Lans, TableHeadings->{None,{"Sol",x,y,\[Lambda],SOC,"Max or Min"}}]
```

Sol	x	y	λ	Hessian	MaxorMin
0.588077	0.792893	0.792893	0.453465	-0.392498	Local Min
-0.987397	2.20711	2.20711	-0.246995	2.60901	Local Max
-1.	1.9842	2.37496	0.	-4.99291	Local Min
-1.	2.37496	1.9842	0.	-4.99291	Local Min
1.	0.733321	2.14203	0.	15.0152	Local Max
1.	2.14203	0.733321	0.	15.0152	Local Max

定義 delta

```
In[15]: delta=D[L,{\\[Lambda]},x,y],2]-Normal@SparseArray[{{2,2}->u,{3,3}->u}];
```

以矩陣輸出 delta

In[16]: Simplify@delta//MatrixForm

$$\text{Out}[16]: \begin{pmatrix} 0 & -3 + 2x & -3 + 2y \\ -3 + 2x & -u + 2\lambda - y^2 \sin(xy) & \cos(xy) - xy \sin(xy) \\ -3 + 2y & \cos(xy) - xy \sin(xy) & -u + 2\lambda - x^2 \sin(xy) \end{pmatrix}$$

傳回各臨界點對應的 delta 行列式實根

In[17]: Solve[Det[delta]==0,u][[1]]/.Lans

Out[17]: {{u → 0.0981245}, {u → -0.652253}, {u → 1.24823}, {u → 1.24823}, {u → -3.7538}, {u → -3.7538}}

輸出臨界點等彙總

In[18]: TableForm[{L,x,y,\[Lambda],u/.Solve[Det[delta]==0,u][[1]],
If[(u/.Solve[Det[delta]==0,u][[1]])<0,"LocalMax","LocalMin"]}/.Lans,
TableHeadings→{None,{"Sol",x,y,\[Lambda],u,"MaxorMin"}}]

Sol	x	y	λ	u	Max or Min
0.588077	0.792893	0.792893	0.453465	0.0981245	Local Min
-0.987397	2.20711	2.20711	-0.246995	-0.652253	Local Max
-1.	1.9842	2.37496	0.	1.24823	Local Min
-1.	2.37496	1.9842	0.	1.24823	Local Min
1.	0.733321	2.14203	0.	-3.7538	Local Max
1.	2.14203	0.733321	0.	-3.7538	Local Max

使用 FindMaximum 和 FindMinimum 求解局部極值

In[19]: #[{Sin[x*y],(x-3/2)^2+(y-3/2)^2==1},{x,1.5},{y,1.5}]\&@
{FindMinimum,FindMaximum}

Out[19]: {{-0.987397,{x → 2.20711,y → 2.20711}},{0.588077,{x → 0.792893,y → 0.792893}}}

與 Out[17] 結果相比較可以發現，Out[17] 中使用 FindMaximum 和 FindMinimum 所得之計算結果並非全域最佳解。由於 FindMinimum 並非以全域作為求解目標，因此容易因為起始值的設定不佳導致收斂到局部極值。為了避免起始值導致 FindMinimum 計算收斂到局部極值，我們利用以下模擬的方式來求得最佳解。

利用隨機的起始值，重複求解 100 次

```
In[20]: sim=Quiet@Sort@Table[Reap@#[{Sin[x*y],(x-3/2)^2+(y-3/2)^2==1},  
{{x,RandomReal[{0,3}]},{y,RandomReal[{0,3}]}},  
StepMonitor:>Sow[{x,y,Sin[x*y]}]]&/@{FindMinimum,FindMaximum},{100}];
```

輸出全域極小值及全域極大值

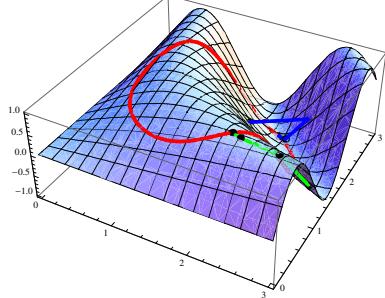
```
In[21]: {sim[[1,1,1]],sim[[1,2,1]]}
```

```
Out[21]: {{-1.,{x→1.9842,y→2.37496}},{1.,{x→0.733321,y→2.14203}}}
```

輸出 FindMaximum 和 FindMinimum 所得之收斂步驟

```
In[22]: Show[Plot3D[Sin[x*y],{x,0,3},{y,0,3},PlotStyle->Opacity[0.5]],  
Graphics3D[{Thickness[0.01],Red,  
Line[{#[[1]],#[[2]],Sin#[[1]]*#[[2]]}]&/@  
ContourPlot[(x-3/2)^2+(y-3/2)^2==1,{x,0,3},{y,0,3}][[1,1]]],  
Blue,Line[sim[[1,1,2,1]]],Green,Line[sim[[1,2,2,1]]],  
Black,PointSize[Large],Point[sim[[1,2,2,1]]],Point[sim[[1,2,2,1]]]]]
```

Out[22]:



範例 4-9. 求解下列極小化問題：

$$\min x_1^2 + x_2^2 + x_3^2$$

受制於

$$\begin{aligned} 2x_1 + x_2 - 5 &\leq 0 \\ x_1 + x_3 - 2 &\leq 0 \\ 1 - x_1 &\leq 0 \\ 2 - x_2 &\leq 0 \\ -x_3 &\leq 0 \end{aligned}$$

定義取代規則

```
In[1]: myrule={z_[i_]:>ToExpression[ToString[z]<>ToString[i]]};
```

定義目標函數

```
In[2]: obj=x1^2+x2^2+x3^2;
```

定義限制條件

```
In[3]: cons={2x1+x2-5,x1+x3-2,1-x1,2-x2,-x3};
```

傳回 Lagrange multiplier

```
In[4]: lambda=\[Lambda] [#]&/@Range[Length@cons]/.myrule
```

```
Out[4]: {λ1, λ2, λ3, λ4, λ5}
```

建立 KKT 必要條件

```
In[5]: kkt=Flatten@{Thread[D[obj-lambda.cons,{x1,x2,x3}]==0],  
Thread[cons<=0],Thread[lambda*cons==0],Thread[lambda<=0]};
```

傳回 KKT 必要條件

```
In[6]: kkt//TableForm
```

```
Out[6]: 2x1 - 2λ1 - λ2 + λ3 == 0  
2x2 - λ1 + λ4 == 0  
2x3 - λ2 + λ5 == 0  
-5 + 2x1 + x2 ≤ 0  
-2 + x1 + x3 ≤ 0  
1 - x1 ≤ 0  
2 - x2 ≤ 0  
-x3 ≤ 0  
(-5 + 2x1 + x2)λ1 == 0  
(-2 + x1 + x3)λ2 == 0  
(1 - x1)λ3 == 0  
(2 - x2)λ4 == 0  
-x3λ5 == 0  
λ1 ≤ 0  
λ2 ≤ 0  
λ3 ≤ 0  
λ4 ≤ 0  
λ5 ≤ 0
```

計算臨界點

```
In[7]: ans=Reduce[kkt,Flatten@{x1,x2,x3,lambda},Backsubstitution->True]/.  
{And->List,Or->List,Equal->Rule}
```

```
Out[7]: {x1 → 1, x2 → 2, x3 → 0, λ1 → 0, λ2 → 0, λ3 → -2, λ4 → -4, λ5 → 0}
```

輸出符合 KKT 必要條件的最佳解

In[8]: `obj/.ans,ans`

Out[8]: `{5,{x1 → 1,x2 → 2,x3 → 0,λ1 → 0,λ2 → 0,λ3 → -2,λ4 → -4,λ5 → 0}}`

利用 FindMinimum 求解

In[9]: `FindMinimum[{obj,Thread[cons<=0]}, {x1,x2,x3}]`

Out[9]: `{5.,{x1 → 1.,x2 → 2.,x3 → 0.000842009}}`

比較以上輸出結果，可發現 FindMinimum 所得到的結果與真正的最佳解仍有些微誤差，此乃 Mathematica 預設的計算精度及容忍誤差僅至小數點後第 5 位所導致。故要提高求解的準確度，可 FindMinimum 中提高 AccuracyGoal 的設定值，降低計算的絕對誤差；或是指定 WorkingPrecision 的設定值提高計算的精度。

使用 FindMinimum 求解，並指定計算的準確度至小數點後第 10 位

In[10]: `FindMinimum[{obj,Thread[cons<=0]}, {x1,1}, {x2,1}, {x3,1},
AccuracyGoal→10]`

Out[10]: `{5.,{x1 → 1.,x2 → 2.,x3 → 5.72272 × 10^-6}}`

使用 FindMinimum 求解，並指定計算的精度至小數點後第 30 位

In[11]: `FindMinimum[{obj,Thread[cons<=0]}, {x1,1}, {x2,1}, {x3,1},
WorkingPrecision→30]`

Out[11]: `{5.00000000003516811603758677727,{x1 → 1.00000000000061188121055347971,
x2 → 2.000000000000030594059926089653,x3 → 5.7201915369536153083919327647 × 10^-6}}`

範例 4-10. 求解下列極小化問題：

$$\min 10(x_1^2 - x_2)^2 + (x_1 + 3x_2 - 4)^2$$

受制於

$$x_1^4 - x_1 x_2^3 = 0$$

$$3x_1 + 4x_2^2 - 8 \leq 0$$

$$-10 \leq x_1 \leq 20$$

$$-15 \leq x_2 \leq 10$$

定義取代規則

```
In[1]: myrule={z_[i_]:>ToExpression[ToString[z]<>ToString[i]]};
```

定義目標函數

```
In[2]: obj=10(x1^2-x2)^2+(-4+x1+3*x2)^2;
```

定義限制式

```
In[3]: eqn=x1^4-x1*x2^3;
ineqn={3*x1+4*x2^2-8,-x1-10,x1-20,-x2-15,x2-10};
```

傳回 Lagrange multiplier

```
In[4]: lambda=\[Lambda][#]&/@Range[Length@ineqn]/.myrule
```

```
Out[4]: {λ1, λ2, λ3, λ4, λ5}
```

建立 KKT 必要條件

```
In[5]: kkt=Flatten@{Thread[D[obj-lambda.ineqn-u*eqn,{x1,x2}]==0],
Thread[ineqn<=0],Thread[lambda*ineqn==0],eqn==0,Thread[lambda<=0]};
```

計算臨界點

```
In[6]: ans=Reduce[kkt,Flatten@{x1,x2,u,lambda},Backsubstitution->True]/.
{And->List,Or->List,Equal->Rule}
```

```
Out[6]: {x1 → 0, x2 → 12/19, u → 1805/108, λ1 → 0, λ2 → 0, λ3 → 0, λ4 → 0, λ5 → 0},  

{x1 → 1, x2 → 1, u → 0, λ1 → 0, λ2 → 0, λ3 → 0, λ4 → 0, λ5 → 0}}
```

輸出最小值

```
In[7]: SortBy[{obj,x1,x2}/.ans,First][[1]]
```

```
Out[7]: {0, 1, 1}
```

由於最佳解具解析解，因此我們也可改以 Minimize 求解。

利用 Minimize 求解

```
In[8]: Minimize[{obj, eqn==0, Thread[ineqn<=0]}, {x1, x2}]
```

```
Out[8]: {0, {x1 → 1, x2 → 1}}
```

利用 FindMinimum 求解

```
In[9]: FindMinimum[Flatten@{obj, eqn==0, Thread[ineqn<=0]}, {x1, x2}]
```

```
Out[9]: {1.25523 × 10-14, {x1 → 1., x2 → 1.}}
```

範例 4-11. 求解下列極大化問題：

$$\max -8x_1 + x_1^2 - 2x_2 + x_2^2 - 3x_3 + 3x_3^2 - x_4 + 4x_4^2 - 2x_5 + 2x_5^2$$

受制於

$$0 \leq x_i \leq 99, i = 1, 2, 3, 4, 5$$

$$x_1 + x_2 + x_3 + x_4 + x_5 \leq 400$$

$$x_1 + 2x_2 + 2x_3 + x_4 + 6x_5 \leq 800$$

$$2x_1 + x_2 + 6x_3 \leq 200$$

$$x_3 + x_4 + 5x_5 \leq 200$$

定義取代規則

```
In[1]: myrule={z_[i_]:>ToExpression[ToString[z]<>ToString[i]]};
```

定義目標函數

```
In[2]: obj=x1^2+x2^2+3*x3^2+4*x4^2+2*x5^2-8*x1-2*x2-3*x3-x4-2*x5;
```

定義限制式

```
In[3]: cons={x1-99,x2-99,x3-99,x4-99,x5-99,-x1,-x2,-x3,-x4,-x5,
x1+x2+x3+x4+x5-400,x1+2*x2+2*x3+x4+6*x5-800,2*x1+x2+6*x3-200,
x3+x4+5*x5-200};
```

定義決策變數

```
In[4]: objvars=Variables@obj
```

Out[4]: {x1, x2, x3, x4, x5}

傳回 Lagrange multiplier

In[5]: lambda=\[Lambda] [#]&/@Range[Length@cons]/.myrule

Out[5]: {λ1, λ2, λ3, λ4, λ5, λ6, λ7, λ8, λ9, λ10, λ11, λ12, λ13, λ14}

建立 KKT 必要條件

In[6]: kkt=Flatten@{Thread[D[obj-lambda.cons,{objvars}]==0], Thread[cons<=0], Thread[lambda*cons==0], Thread[lambda>=0]};

計算局部極大值的臨界點

In[7]: ans=Reduce[kkt,Flatten@{objvars,lambda},Backsubstitution->True]/. {And->List,Or->List,Equal->Rule};

輸出臨界點的個數

In[8]: ans//Length

Out[8]: 207

將所有臨界點代入求解，並輸出最大值

In[9]: SortBy[{obj/.#, #}&@ans,First][[-1]]

Out[9]: $\left\{ \frac{5162993}{100}, \left\{ x_1 \rightarrow \frac{101}{2}, x_2 \rightarrow 99, x_3 \rightarrow 0, x_4 \rightarrow 99, x_5 \rightarrow \frac{101}{5}, \lambda_1 \rightarrow 0, \lambda_2 \rightarrow \frac{299}{2}, \right. \right.$
 $\left. \lambda_3 \rightarrow 0, \lambda_4 \rightarrow \frac{19381}{25}, \lambda_5 \rightarrow 0, \lambda_6 \rightarrow 0, \lambda_7 \rightarrow 0, \lambda_8 \rightarrow \frac{7444}{25}, \lambda_9 \rightarrow 0, \lambda_{10} \rightarrow 0, \right.$
 $\left. \lambda_{11} \rightarrow 0, \lambda_{12} \rightarrow 0, \lambda_{13} \rightarrow \frac{93}{2}, \lambda_{14} \rightarrow \frac{394}{25} \right\} \right\}$

使用 FindMaximum 求解

In[10]: FindMaximum[{obj,Thread[cons<=0]},objvars]

Out[10]: {42677.6, {x1 → 8.31059 × 10⁻¹¹, x2 → 9.98507 × 10⁻¹¹, x3 → 33.3333, x4 → 99., x5 → 13.5333}}

由 Out[8] 可以發現，滿足 KKT 條件的局部極大值共 207 組。在利用 FindMaximum 求解時，若是起始值取得不好，FindMaximum 很容易收斂到局部極值。因此，為了避免起始值的選擇導致 FindMaximum 收斂

到局部極值，我們利用以下模擬的方式來求得最佳解。

利用隨機的起始值，重複求解 100 次

```
In[11]: simans=SortBy[FindMaximum[{obj, Thread[cons<=0]}, Transpose@{objvars, RandomReal[-100, 100], Length@objvars}]]&/@Range[100], First];
```

輸出模擬資料的最大值

```
In[12]: simans[[-1]]
```

```
Out[12]: {51629.9, {x1 → 50.5, x2 → 99., x3 → 1.51651 × 10-12, x4 → 99., x5 → 20.2}}
```

建立局部極小值的 KKT 必要條件

```
In[13]: kktmin=Flatten@{Thread[D[obj-lambda.cons,{objvars}]]==0, Thread[cons<=0], Thread[lambda*cons==0], Thread[lambda<=0]};
```

計算局部極小值的臨界點

```
In[14]: ansmin=Reduce[kktmin, Flatten@{objvars, lambda}, Backsubstitution→True]/. {And→List, Or→List, Equal→Rule}
```

```
Out[14]: {x1 → 4, x2 → 1, x3 → 1/2, x4 → 1/8, x5 → 1/2, λ1 → 0, λ2 → 0, λ3 → 0, λ4 → 0, λ5 → 0, λ6 → 0, λ7 → 0, λ8 → 0, λ9 → 0, λ10 → 0, λ11 → 0, λ12 → 0, λ13 → 0, λ14 → 0}
```

輸出最小值

```
In[15]: obj/.ansmin
```

```
Out[15]: -293/16
```

使用 FindMinimum 求解

```
In[16]: FindMinimum[{obj, Thread[cons<=0]}, objvars]
```

```
Out[16]: {-18.3125, {x1 → 4., x2 → 1., x3 → 0.5, x4 → 0.125, x5 → 0.5}}
```

由以上兩個範例可以發現到若目標函數和限制式皆為多項式函數，可由函數 Reduce 來求得所有滿足 Karush-Kuhn-Tucker 最佳化條件的局部極值。為方便求解，我們可將上述範例的求解步驟利用 Block 寫成單一函數。

範例 4-12. 建立 Karush-Kuhn-Tucker 最佳化條件求解函數

使用 Block 撰寫 KKT 求解程序

```
In[1]: KKTMinimize[obj_,eqns_,ineqns_,variables_]:=Block[{myrule,lambda,u,\[Lambda],Lagrange,eqnu,ineqnlam,eqnus,ineqnlams,kkteqns,kktvars,kktans},myrule={z_[i_]:>ToExpression[ToString[z]<>ToString[i]]};eqnu=u[#]&/@Range[Length@eqns];ineqnlam=\[Lambda][#]&/@Range[Length@ineqns];eqnus=If[Length@eqns>=1,eqns.eqnu,0];ineqnlams=If[Length@ineqns>=1,ineqns.ineqnlam,0];kktvars=Flatten@{variables,eqnu,ineqnlam}/.myrule;Lagrange=obj-eqnus-ineqnlams;kkteqns=Flatten@{Thread[D[Lagrange,{variables}]==0],Thread[eqns==0],Thread[ineqns<=0],Thread[ineqns*ineqnlam==0],Thread[ineqnlam<=0]}/.myrule;If[MemberQ[PolynomialQ[#,kktvars]&/@kkteqns[[All,1]],False],Print["KKT限制式均需為多項式。"],kktns=Reduce[kkteqns,kktvars,Backsubstitution->True]/.{And->List,Or->List,Equal->Rule};If[Length@Dimensions@kktns==1,{obj/.kktns,kktns},{obj/.#,#}&/@kktns]]];
```

求解範例 4-9

```
In[2]: KKTMinimize[10(x1^2-x2)^2+(-4+x1+3*x2)^2,{x1^4-x1*x2^3}, {3*x1+4*x2^2-8,-x1-10,x1-20,-x2-15,x2-10},{x1,x2}]
```

```
Out[2]: \left\{\left\{\frac{160}{19},\left\{x1\rightarrow 0,x2\rightarrow \frac{12}{19},u1\rightarrow \frac{1805}{108},\lambda1\rightarrow 0,\lambda2\rightarrow 0,\lambda3\rightarrow 0,\lambda4\rightarrow 0,\lambda5\rightarrow 0\right\}\right\},\left\{0,\{x1\rightarrow 1,x2\rightarrow 1,u1\rightarrow 0,\lambda1\rightarrow 0,\lambda2\rightarrow 0,\lambda3\rightarrow 0,\lambda4\rightarrow 0,\lambda5\rightarrow 0\}\right\}\right\}
```

求解範例 4-10

```
In[3]: KKTMinimize[x1^2+x2^2+x3^2,{}, {2x1+x2-5,x1+x3-2,1-x1,2-x2,-x3}, {x1,x2,x3}]
```

```
Out[3]: {5,{x1\rightarrow 1,x2\rightarrow 2,x3\rightarrow 0,\lambda1\rightarrow 0,\lambda2\rightarrow 0,\lambda3\rightarrow -2,\lambda4\rightarrow -4,\lambda5\rightarrow 0}}
```

求解範例 4-11

```
In[4]: obj=x1^2+x2^2+3*x3^2+4*x4^2+2*x5^2-8*x1-2*x2-3*x3-x4-2*x5;
cons={x1-99,x2-99,x3-99,x4-99,x5-99,-x1,-x2,-x3,-x4,-x5,
x1+x2+x3+x4+x5-400,x1+2*x2+2*x3+x4+6*x5-800,2*x1+x2+6*x3-200,
x3+x4+5*x5-200};
ans=KKTMinimize[-obj,{},cons,{x1,x2,x3,x4,x5}];
SortBy[ans,First][[1]]
```

Out[4]: $\left\{ -\frac{5162993}{100}, \left\{ x_1 \rightarrow \frac{101}{2}, x_2 \rightarrow 99, x_3 \rightarrow 0, x_4 \rightarrow 99, x_5 \rightarrow \frac{101}{5}, \lambda_1 \rightarrow 0, \lambda_2 \rightarrow -\frac{299}{2}, \lambda_3 \rightarrow 0, \lambda_4 \rightarrow -\frac{19381}{25}, \lambda_5 \rightarrow 0, \lambda_6 \rightarrow 0, \lambda_7 \rightarrow 0, \lambda_8 \rightarrow -\frac{7444}{25}, \lambda_9 \rightarrow 0, \lambda_{10} \rightarrow 0, \lambda_{11} \rightarrow 0, \lambda_{12} \rightarrow 0, \lambda_{13} \rightarrow -\frac{93}{2}, \lambda_{14} \rightarrow -\frac{394}{25} \right\} \right\}$

範例 4-13. 求解下列極大化問題:

$$\max \sum_{i=1}^n \left\{ [\ln(x_i - 2)]^2 + [\ln(10 - x_i)]^2 \right\} - \left\{ \prod_{i=1}^n x_i \right\}^{\frac{1}{5}}$$

受制於

$$2 < x_i < 10, \quad i = 1, 2, \dots, n$$

定義取代規則

```
In[1]: myrule={x[i_]:=ToExpression[ToString[x]<>ToString[i]]};
```

定義目標函數

```
In[2]: obj[n_]:=Sum[Log[x[i]-2]^2+Log[10-x[i]]^2,{i,n}]-  
(Times@@Table[x[i],{i,n}])^(1/5)/.myrule;
```

定義限制式

```
In[3]: cons[n_]:=Flatten[{2-x[#],x[#]-10}&/@Range[n]]/.myrule;
```

定義變數

```
In[4]: vars[n_]:=x[#]&/@Range[n]/.myrule;
```

定義 Lagrange multiplier

```
In[5]: lambda[n_]:=\[Lambda][#]&/@Range[Length@cons[n]]/.
{\[Lambda][i_]:>ToExpression[ToString[\[Lambda]]<>ToString[i]]};
```

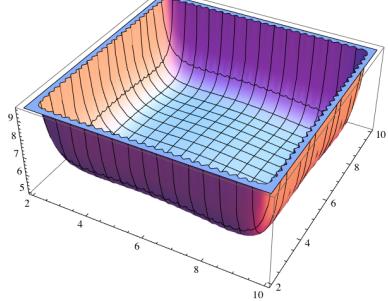
定義 KKT 最佳化條件

```
In[6]: KKT[n_]:=Flatten@{Thread[D[obj[n]-lambda[n].cons[n],{vars[n]}]==0],
Thread[lambda[n]*cons[n]==0],Thread[cons[n]<=0]};
```

輸出 $n = 2$ 時的目標函數圖形

```
In[7]: Plot3D[Evaluate@obj[2],{x1,2,10},{x2,2,10}]
```

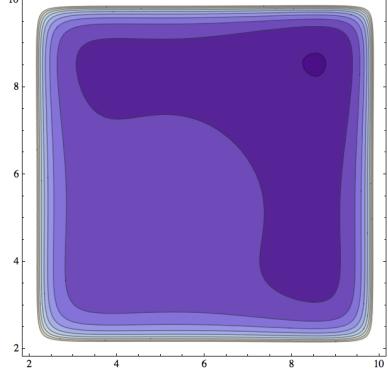
Out[7]:



輸出 $n = 2$ 時的目標函數等高線圖

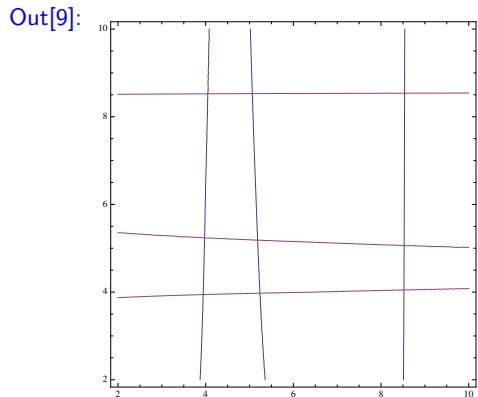
```
In[8]: ContourPlot[Evaluate@obj[2],{x1,2,10},{x2,2,10}]
```

Out[8]:



輸出所有局部極值位置

```
In[9]: ContourPlot[Evaluate@Thread[D[obj[2],{vars[2]}]==0],
{x1,2,10},{x2,2,10},Contours->20]
```



使用預設的起始值，利用使 `FindMinimum` 求解

```
In[10]: Reap@FindMinimum[Flatten@{obj[2], Thread[cons[2]<0]},  
vars[2], EvaluationMonitor:>Sow[vars[2]]]
```

```
Out[10]: {{5.64002, {x1 → 3.93526, x2 → 3.93526}}, {{4.01404, 4.01404}, {4.00082, 4.00082},  
{3.93517, 3.93517}, {3.93526, 3.93526}, {3.93526, 3.93526}, {3.93526, 3.93526}}}}
```

使用隨機的起始值，利用使 `FindMinimum` 求解

```
In[11]: Reap@FindMinimum[Flatten@{obj[2], Thread[cons[2]<0]},  
Transpose@{vars[2], Table[RandomReal[{2, 10}], {Length@vars[2]}]},  
EvaluationMonitor:>Sow[vars[2]]]
```

```
Out[11]: {{4.98151, {x1 → 8.53879, x2 → 8.53879}}, {{8.04546, 7.54566}, {8.56141, 9.47117},  
{8.30343, 8.50841}, {8.56939, 8.53356}, {8.54145, 8.53865}, {8.53883, 8.53879},  
{8.53879, 8.53879}, {8.53879, 8.53879}}}}
```

由以上輸出結果比較，可以發現當 $n = 2$ 時使用 `FindMinimum` 所求得的解並非全域極小值。事實上當 $n = 2$ 時，這個問題滿足 KKT 必要條件的臨界點共九個。由於 Mathematica 以 $(4.01404, 4.01404)$ 為起始值，故只收斂到局部極小值。

使用 `FindRoot` 求解所有局部極值

```
In[12]: local=FindRoot[KKT[2][[1;;Length[vars[2]]+Length[lambda[2]]]], #]&/@  
Flatten[Table[Transpose[{Flatten@{vars[2], lambda[2]},  
Flatten@{i, j, -1, -1, -1, -1}}], {i, {4, 5, 9}}, {j, {4, 5, 9}}]], 1];
```

輸出所有局部極值

```
In[13]: TableForm[Flatten@{obj[2], vars[2], Round[\lambda[2], 0.00001]}/.local,
TableHeadings -> {Range[Length@local], Flatten@{obj, vars[2], \lambda[2]}}]
```

Out[13]:

	obj	x1	x2	λ_1	λ_2	λ_3	λ_4
1	5.64002	3.93526	3.93526	0.	0.	0.	0.
2	5.66942	3.97015	5.24082	0.	0.	0.	0.
3	5.33423	4.04463	8.52445	0.	0.	0.	0.
4	5.66942	5.24082	3.97015	0.	0.	0.	0.
5	5.69321	5.18623	5.18623	0.	0.	0.	0.
6	5.34842	5.06588	8.52861	0.	0.	0.	0.
7	5.33423	8.52445	4.04463	0.	0.	0.	0.
8	5.34842	8.52861	5.06588	0.	0.	0.	0.
9	4.98151	8.53879	8.53879	0.	0.	0.	0.

輸出 KKT 最佳化條件

```
In[14]: TableForm[{{TableForm[Round[KKT[2][[All, 1]][[1, ; -5]]/.local, 0.00001]], 
TableForm[Round[KKT[2][[All, 1]][[-4, ; -1]]/.local, 0.00001]]}}, 
TableHeadings -> {None, {"f - \Lambda g = 0", "g = 0"}}, 
TableAlignments -> Center]
```

Out[14]:

$\nabla f - \lambda \nabla g$						$g \leq 0$		
0.	0.	0.	0.	0.	0.	-1.93526	-6.06474	-1.93526
0.	0.	0.	0.	0.	0.	-1.97015	-6.02985	-3.24082
0.	0.	0.	0.	0.	0.	-2.04463	-5.95537	-6.52445
0.	0.	0.	0.	0.	0.	-3.24082	-4.75918	-1.97015
0.	0.	0.	0.	0.	0.	-3.18623	-4.81377	-3.18623
0.	0.	0.	0.	0.	0.	-3.06588	-4.93412	-6.52861
0.	0.	0.	0.	0.	0.	-6.52445	-1.47555	-2.04463
0.	0.	0.	0.	0.	0.	-6.52861	-1.47139	-3.06588
0.	0.	0.	0.	0.	0.	-6.53879	-1.46121	-6.53879

由 Out[13] 與 Out[14] 的結果得知，Out[13] 所得到的臨界點均能滿足 KKT 的最佳化條件，因此均為局部極值。再比較後所有的局部極值後可得 $(x_1^*, x_2^*) = (8.53879, 8.53879)$ ，此時有全域極小值為 4.98151。此外，由 Out[13] 的結果也可發現各組解的 $\lambda_i = 0$, $i = 1, 2, 3, 4$ ，即表示此限制條件均為無效的限制式。

利用隨機的起始值重複求解 100 次，計算 $n = 2, 3, \dots, 10$ 的全域極大值

```
In[15]: simuans=ParallelTable[FindMinimum[{obj[#], Thread[cons[#]<=0]}, 
Transpose[{vars[#], RandomReal[{2, 10}, #]}]], {100}]&/@Range[2, 10];
```

輸出 $n = 2, 3, \dots, 10$ 的全域極大值

```
In[16]: TableForm[Table[Flatten[Round[{#[[1]], vars[z+1]/.#[[2]]}, 0.0001]&@
{Sort[simuans[[z]]][[1]]}], {z, Length@simuans}], 
TableHeadings -> {Range[2, 10], Flatten@{obj, vars[10]}}]
```

```
Out[16]:    obj      x1      x2      x3      x4      x5      x6      x7      x8      x9      x10
 2  4.9815  8.5388  8.5388
 3  7.386   8.5896  8.5896  8.5896
 4  9.1001  8.6564  8.6564  8.6564  8.6564
 5  9.7305  8.7407  8.7407  8.7407  8.7407  8.7407
 6  8.6456  8.8424  8.8424  8.8424  8.8424  8.8424  8.8424
 7  4.8216  8.9592  8.9592  8.9592  8.9592  8.9592  8.9592  8.9592
 8  -3.4119  9.0869  9.0869  9.0869  9.0869  9.0869  9.0869  9.0869  9.0869
 9  -18.7878 9.2195  9.2195  9.2195  9.2195  9.2195  9.2195  9.2195  9.2195  9.2195
10  -45.7785 9.3503  9.3503  9.3503  9.3503  9.3503  9.3503  9.3503  9.3503  9.3503
```