系統最佳化實驗室基本演算法及範例使用說明

詹魁元

July 14, 2019

Contents

Chapter 1

演算法說明

1.1 Excel 規劃求解(GRG Method)

1.1.1 概述

GRG Method 演算法全名爲 Generalized Reduced Gradient Method，用等式拘束條件來減少變數的自由度，把原問題轉換成沒有拘束條件的問題求解。而 Excel 裡面的最佳化工具 (規劃求解) 使用的便是 GRG 演算法，以下示範皆採用 2007 版本 Excel。

1.1.2 檔案列表

Excel 中的規劃求解並不需要額外撰寫檔案，所有設定都在 Excel 的儲存格中指定。

1.1.3 啓用規劃求解工具

此工具放置在資料選單的分析下；如果沒有這選項，有可能是因爲沒有安裝完整版本或未選擇增益集。

如何選擇增益集：

點選其它命令，接著在增益集中點選規劃求解增益集。如圖1.1、1.2。

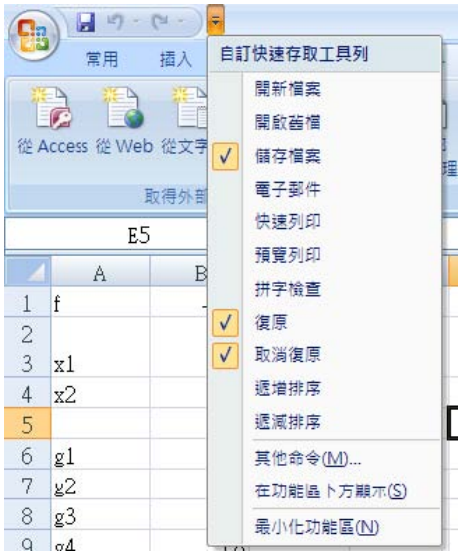


圖 1.1

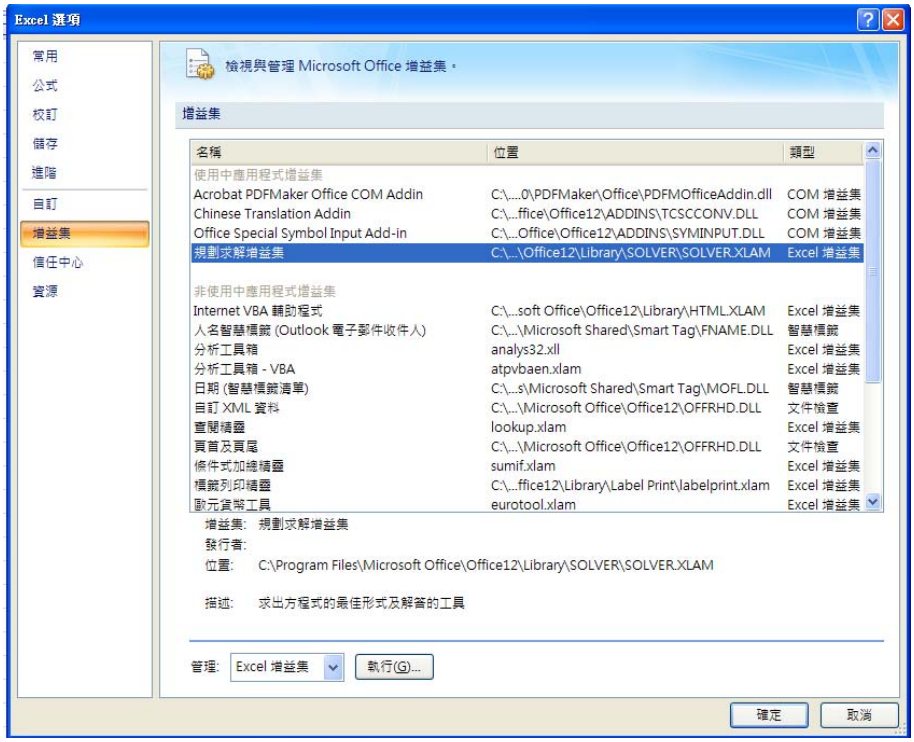


圖 1.2

1.1.4 參數設定與範例說明

下面使用一個數學範例來做練習，解決以下問題：

$$
\begin{aligned}
\min_{x_1,x_2} \quad & 2x_1^2 + x_2^2 - 48x_1 - 40x_2 \\
\text{subject to} \quad & g_1(x) = x_1 + 3x_2 \geq 0 \\
& g_2(x) = 18 - x_1 - 3x_2 \geq 0 \\
& g_3(x) = x_1 + x_2 \geq 0 \\
& g_4(x) = 8 - x_1 - x_2 \geq 0 \\
& 0 \leq x_1, x_2 \leq 6 \\
\text{起始值在} \quad & (x_1, x_2) = (0.1, 0.1)
\end{aligned}
$$

第一步：

設定儲存格，並且撰寫互相的關係 (eg: 目標函數儲存格與變數儲存格的關係)，圖1.3

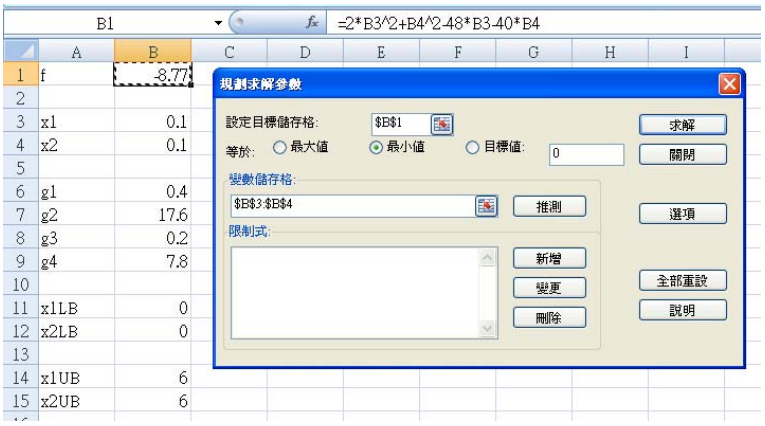
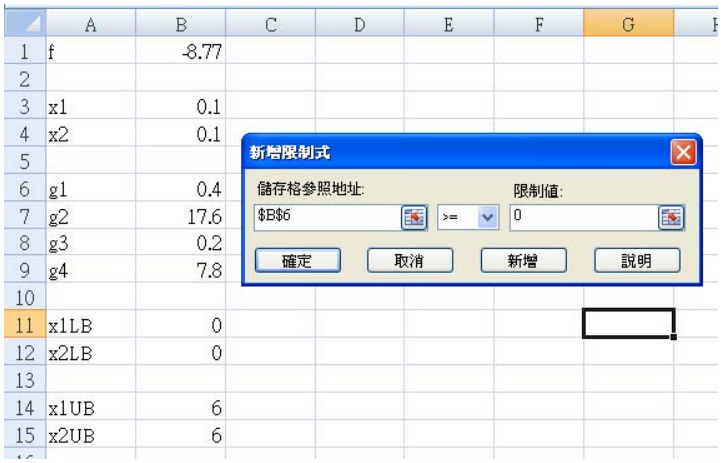


圖 1.3

如圖所示，我們先指定目標函數儲存格 (B1)，變數儲存格 (B3, B4)，並且在目標函數儲存格設定其與變數儲存格的關係 (i.e. 在目標儲存格打入 $= 2 * B3^2 + B4^2 - 48 * B3 - 40 * B4$)，同理拘束條件儲存格也打入相對關係式。

第二步：給定限制式

在限制式中按下“新增”，圖1.4



	A	B
1	f	-8.77
2		
3	x1	0.1
4	x2	0.1
5		
6	g1	0.4
7	g2	17.6
8	g3	0.2
9	g4	7.8
10		
11	x1LB	0
12	x2LB	0
13		
14	x1UB	6
15	x2UB	6

圖 1.4

如上圖所示，我們依序設定指定第一個拘束條件值 (B6)≥ 限制值 0，其餘 3 個拘束條件都是如此處理。同理，變數的上下界也是依照同樣觀念設定 4 條限制式。

第三步：設定演算法選項(也可不設定直接使用預設值運算)

在設定完限制式之後，設定規劃求解的選項（i.e. 設定收斂條件，精確度，最長運算時間等），設定完之後按確定，就可以進行求解；最後儲存格的值便是最佳解。選項如圖1.5：

圖 1.5

選項：

最長運算時限：

我們容許整個運算過程的時間限制，爲了避免當問題是無解，Excel 無法計算出結果持續運算的情況；最多可以輸入 32767，預設值是 100。

反覆運算次數：

我們容許整個運算過程的疊代次數，同樣也是爲了避免當問題是無解，Excel 無法計算出結果持續運算的情況；最多可以輸入 32767，預設值是 100。

精確度：

調整精確度會影響變數靠近限制式的上下界程度。當變數在可行解空間尋找最佳解，碰到重要拘束條件 (active constraints) 所造成的邊界時，此拘束條件值會很接近零，而因爲數值運算上只要小於某個值(精確度)它就會判斷其值爲零。此值只能輸入介於0 1之間的數值，小數點位數越高越精確 (eg: 0.0001 比 0.1 精準度來的高)。

誤差容忍度：

調整可行解空間的大小。在運算過程容許變數在違背拘束條件的情況下，還能接受此變數並且繼續運算，而違背的程度我們可以設定“誤差容忍度”來調整；越高的誤差容忍度會加速運算過程。

收斂：

目標函數儲存格在疊代過程中，連續兩次的值相差在我們設定的“收斂條件”以下的時候運算停止。這個值必須輸入在 0 ～ 1 之間。

採用線性模式：

當目標函數與變數間的關係，還有拘束條件跟變數之間的關係都是線性的時候，可以選擇此模式加速運算。

採用非負值：

讓所有儲存格的變動都必須在正數的條件下運算。

自動按比例縮放：

自動把輸入 (變數) 與輸出 (目標函數值與拘束條件值) 的數字大小等級 (Scale) 調整成差不多。有些情況下會因爲輸入與輸出的數字大小等級相差過大而計算失敗。

顯示反覆運算結果：

讓 Excel 每次疊代時皆暫停，且秀出當時運算結果。

估計式：

GRG 內迴圈運算時，起始點的預測方式。

正切函數：使用一階線性的預測方式，此爲原始設定 (default)。

二次方程式：使用二階線性的預測方式。

導函數/偏導式：

在尋找目標函數與拘束條件梯度的數值方法。

前推離差估計：在下一點與當前設計點兩點的資訊來計算梯度，適用在函數值變動相對較不劇烈的情況。

中央離差估計：數值上另一種找梯度的方法，是用在函數值變動快速的情況；雖然需要較多的計算量，但可以改善部分原本無法求解的問題。

搜尋：

GRG 演算法中，外迴圈尋找設計點的方法選擇。

牛頓法：相較於共軛法，牛頓法雖然每次計算量相對較多，但是需要較少的疊代次數。

共軛法： 雖然每次計算量相對較少，但需要較多的疊代次數才能達到最佳值，適合解決較龐大的問題。

1.2 fmincon

1.2.1 概要

`fmincon` 是 MATLAB Optimization Toolbox 中最常被使用的函數之一。它通常被用來解決多變數的非線性最佳化問題，其所使用的演算法爲 SQP (Sequential Quadratic Programming)。

1.2.2 檔案列表

爲了執行 `fmincon` 演算法，使用者必須製作一個 script 檔 (主程式)、兩個 function 檔 (副程式)，其內容可由下列範例說明，其中包含了如何設定必要的參數、目標函數與拘束條件。

1.2.3 範例演示與程式撰寫

以下將使用一數學範例來介紹如何使用 `fmincon`：

$$\begin{aligned} \min_{x_1,x_2} \quad & 2x_1^2 + x_2^2 - 48x_1 - 40x_2 \\ \text{subject to} \quad & g_1(x_1,x_2) = x_1 + 3x_2 \geq 0 \\ & g_2(x_1,x_2) = 18 - x_1 - 3x_2 \geq 0 \\ & g_3(x_1,x_2) = x_1 + 3x_2 \geq 0 \\ & g_4(x_1,x_2) = 8 - x_1 - x_2 \geq 0 \\ & 0 \leq x_1, x_2 \leq 6 \end{aligned}$$

1. 爲符合 `fmincon` 的輸入型式，須先將最佳化數學式轉換爲 negative null form：

$$\begin{aligned} \min_{x_1,x_2} \quad & 2x_1^2 + x_2^2 - 48x_1 - 40x_2 \\ \text{subject to} \quad & g_1(x_1,x_2) = -x_1 - 3x_2 \leq 0 \\ & g_2(x_1,x_2) = x_1 + 3x_2 - 18 \leq 0 \\ & g_3(x_1,x_2) = -x_1 - 3x_2 \leq 0 \\ & g_4(x_1,x_2) = x_1 + x_2 - 8 \leq 0 \\ & 0 \leq x_1, x_2 \leq 6 \end{aligned}$$

2. 建立主程式：

File：main.m

```
doc fmincon %利用 doc 查詢 fmincon 的使用方式
x0=[0.1;0.1]; %起始點
A=[]; %線性不等式拘束條件的係數矩陣
b=[]; %線性不等式拘束條件的係數向量 AX <= b
Aeq=[]; %線性不等式拘束條件的係數向量
beq=[]; %線性等式拘束條件的係數向量 AeqX = beq
ub=[6;6]; %設計空間的 upper bounds
lb=[0;0]; %設計空間的 lower bounds
options = optimset ('display','off','Algorithm','sqp');%演算法的參數設定
[x,fval,exitflag]=fmincon(@(x)obj(x),x0,A,b,Aeq,beq,lb,ub,...
     @(x)nonlcon(x),options);
%執行fmincon，輸出最佳解,x; 最佳目標函數值,fval; 收斂情形,exitflag
%obj爲目標函數，nonlcon 爲 (非線性) 拘束條件
```

其中 optimset 會在1.2.4章節中說明。

3. 建立目標函數檔：

File：obj.m

```
function f=object(x)
f = 2*x(1,:).^2+x(2,:).^2-48*x(1,:)-40*x(2,:); %目標函數
```

此檔的檔名爲 obj.m，與 main.m 相呼應，使得目標函數可以被呼叫。

File：nonlcon.m

```
function [g,geq]=nonlcon(x)
g(1) = -x(1,:)-3*x(2,:); %(非線性)拘束條件，g(1)<=0
g(2) = -18+x(1,:)+3*x(2,:); %g(2)<=0
g(3) = -x(1,:)-x(2,:); %g(3)<=0
g(4) = -8+x(1,:)+x(2,:); %g(4)<=0
geq=[];%等式拘束條件，在本問題中沒有等式拘束條件，故爲空集合
```

此檔的檔名爲 nonlcon.m，與 main.m 相呼應，使得非線性拘束條件可以被呼叫。之後的結果在 main.m 中的 `x` 與 `fval` 中可以得到。

1.2.4 optimset

`optimset` 爲 optimization toolbox 中的參數設定選項，可根據不同的最佳化問題，做細部的調整。

在 command window 中輸入 `optimset` 即可顯示所有可調整之參數，其中常調整的參數有：

- Display：決定函數被呼叫時中間結果的顯示方式，其中 off 爲不顯示， iter 表示每步都顯示， final 只顯示最終結果。
- MaxFunEvals：允許最大的函數計算次數。
- MaxIter：允許最大的疊代次數。
- TolFun：當連續兩個設計點之間的差距小於 TolFun 值時，函數即終止。
- TolX：當連續兩個設計點之間的 **x** 差距小於 TolX 值時，函數即終止。
- Algorithm：選擇要使用的演算法，其中有：active-set, interior-point, interior-point-convex, levenberg-marquardt, simplex, sqp, trust-region-dogleg, trust-region-reflective。

1.2.5 輸出結果

`x`：最佳值
`fval`：最佳目標函數值
`exitflag`：收斂情形(見下表)

Exitflag值	收斂狀況
1	一階最佳化量值和違反拘束條件的最大程度小於預設值
2	變數的變化量小於預設值
3	目標函數值的變化量小於預設值
4	搜尋方向值和違反拘束條件的最大程度小於預設值
5	搜尋方向的方向導數值和違反拘束條件的最大程度小於預設值
0	疊代次數或函數計算數超出預設值
-1	輸出函數結束演算法
-2	無可行解

1.3 Genetic Algorithm

1.3.1 概述

Genetic Algorithm (GA) 在某一族群中作運算，某一族群爲設計空間中的點的集合。演算法一開始選用的族群爲隨機也可指定，而某一族群繁衍的下一代族群，則因問題的不同而有不同的下一代族群產生。因此可處理線性與非線性的目標函數，線性和非線性的拘束條件問題。其數學形式可由下列表示:

$$\begin{aligned} \min_{x} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{g}(\mathbf{x}) \leq 0 \end{aligned}$$

1.3.2 檔案列表

爲了執行 GA，使用者必須先製作一個 main.m 檔及一至二個 function 檔 (目標函數與非線性的拘束條件)，其內容可由下列範例說明，其中包含如何訂定變數邊界，目標函數與拘束條件。

1.3.3 參數設定與範例說明

下面使用一個數學範例來做練習，解決以下問題：

$$
\begin{aligned}
\min_{x_1,x_2} \quad & 2x_1^2 + x_2^2 - 48x_1 - 40x_2 \\
\text{subject to} \quad & g_1(x) = x_1 + 3x_2 \geq 0 \\
& g_2(x) = 18 - x_1 - 3x_2 \geq 0 \\
& g_3(x) = x_1 + x_2 \geq 0 \\
& g_4(x) = 8 - x_1 - x_2 \geq 0 \\
& 0 \leq x_1, x_2 \leq 6 \\
\text{起始值在} \quad & (x_1, x_2) = (0.1, 0.1)
\end{aligned}
$$

File:main.m

```
clc
clear
%----------GA algorithm Demo----------%
n=2; % 設計變數的個數
A=[-1,-3; % 輸入線性不等式拘束條件的係數矩陣A與B
1, 3; % A*X<B
-1,-1;
1, 1];
B=[0;18;0;8];
Aeq=[]; % 輸入線性等式的拘束條件係數矩陣Aeq與Beq
Beq=[]; % Aeq*X=Beq
LB=[0;0]; % 設計變數的下限
UB=[6;6]; % 設計變數的上限
options=gaoptimset; % 演算法的參數使用內設值
[x,fval,exitflag]=ga(@f,n,A,B,Aeq,Beq,LB,UB,@nonlcon,options)
% x:最佳解
```

% fval:目標函數值

% exitflag: 收斂情形，當 exitflag>0 爲收斂，其餘顯示沒有收斂

% f: 呼叫檔名爲f的m檔，在此 f 爲目標函數

% nonlcon: 呼叫檔名爲 nonolcon 的 m 檔，在此爲非線性的等式與不等式拘束條件

% options: 調整演算法的內的參數與收斂條件。

File:object.m

```
function obj=f(x)
obj=2*x(1)^2+x(2)^2-48*x(1)-40*x(2); % 輸入目標函數
```

此檔的檔名爲 object.m，與 main.m 相呼應，使得目標函數可以被呼叫。

File:nonlcon.m

```
function [c,ceq]=nonlcon(x)
c=[]; % 輸入非線性不等式拘束條件
ceq=[]; % 輸入非線性等式拘束條件
```

此檔的檔名爲 nonlcon.m，與 Script 相呼應，使得非線性拘束條件可以被呼叫。而解題後的結果在 main.m 中的 x 與 fval 中可以得到。

1.3.4 gaoptimset參數設定

欲達到改變演算法參數與收斂條件參數，可藉由 main 中的 Option 來調整。

```
options=gaoptimset; % 演算法的參數使用內設值
[x,fval,exitflag]=ga(@f,n,A,B,Aeq,Beq,LB,UB,@nonlcon,options)
```

常用的建議調整如下:

1. 當想要檢視一代一代的過程時。

   ```
   options=gaoptimset('Display','iter'); % 顯示每次的過程結果
   ```

2. exitflag 出現 0 時，會試者 options 更改成下圖，來觀察得到較好的結果。

   ```
   options=gaoptimset('generation',200); % 增加 generation 的次數
   ```

3. exitflag -1

   ```
   options=gaoptimset; %恢復內定值，將畫繪圖與 output 取消
   ```

4. exitflag -4

   ```
   options=gaoptimset('StallTimeLimit',30); % 增加 stall time
   ```

5. 自動繪圖：繪出每一次 generation 中最好的答案。

   ```
   options=gaoptimset('PlotFcns',@gaplotbestf');
   ```

1.4 DIRECT(gclsolve)

1.4.1 概述

DIRECT 是不需要梯度計算而且可以找到全域最佳值的演算法，它首先會在設計空間的正中央計算，接下來演算法開始把設計空間切割成較小的矩形 (Rectangles)，再計算矩形中央的值，因此這類演算法才會命名爲 DIvided RECTangles。

本說明所使用的程式碼 (gclsolve.m)，只能處理不等式拘束條件，另外它最後只會在函數計算次數 (Function evaluations) 或是疊代次數 (Iterations) 達到我們指定的值才會停止，因此理論上計算越多次達案越接近最佳解。

1.4.2 檔案列表

使用 gclsolve.m 進行解決最佳化問題需要製作一個主程式檔 (main.m)、一個拘束條件檔 (nonlcon.m)、一個目標函數檔 (object.m)。

1.4.3 參數設定與範例說明

下面使用一個數學範例來做練習，解決以下問題：

$$
\begin{aligned}
\min_{x_1,x_2} \quad & 2x_1^2 + x_2^2 - 48x_1 - 40x_2 \\
\text{subject to} \quad & g_1(x) = x_1 + 3x_2 \geq 0 \\
& g_2(x) = 18 - x_1 - 3x_2 \geq 0 \\
& g_3(x) = x_1 + x_2 \geq 0 \\
& g_4(x) = 8 - x_1 - x_2 \geq 0 \\
& 0 \leq x_1, x_2 \leq 6 \\
\text{起始值在} \quad & (x_1, x_2) = (0.1, 0.1)
\end{aligned}
$$

使用 gclsolve 來處理這個最佳化問題，只需要撰寫目標函數檔以及拘束條件檔，接著在主程式設定一些參數之後就可以進行運算，詳述如下：

File:main.m

```
clear all
clc
GLOBAL.MaxIter=0; % 設定0爲最大值，即不倚靠疊代次數做停止條件
GLOBAL.MaxEval=1000; % 設定當函數計算1000次時停止運算。
vlb=[0 0]; % 變數下界
vub=[6 6]; % 變數上界
I=[ ]; % 設定離散變數，此問題沒有離散變數。
nc=2; % 非線性拘束條件的數目
c_L=-inf*ones(nc,1); % 非拘束條件值的下界 (-infinity)
c_U=zeros(nc,1); % 非拘束條件值的上界 (zero)
A=[ ]; b_L=[ ]; b_U=[ ]; % 設定線性拘束條件，此問題沒有線性拘束條件
xopt = gclSolve(' object_function ', ' constraints ',...
 vlb, vub, A, b_L, b_U, c_L, c_U, I, GLOBAL)
```

File:object.m

```
Function [ f ] = object_function(x)
f = 2*x(1)^2+x(2)^2-48*x(1)-40*x(2);
```

File:nonlcon.m

```
function [ c ] = constraints(x)
c(1) = -x(1)-3*x(2);
c(2) = -18+x(1)+3*x(2);
c(3) = -x(1)-x(2);
c(4) = -8+x(1)+x(2);
```

* 注意：拘束條件檔只輸出函數值，它並沒有 c(x)0 的基本設定，要達到這目的，必須在主程式裡給定拘束函數值的上下界。

此演算法計算到最後會產生一個結構狀的輸出 (xopt)，其中最佳解與最佳值如下：

```
optimal = xopt.x_k % 最佳解
optimal = xopt.f_k % 最佳值
```

DIRECT是個沒有收斂條件的演算法，因此理論上計算的次數越多就越接近最佳解，也因此容易有接續上次計算結果繼續計算的需求，只要在主程式一開始加入以下一行程式碼，就可以繼續計算：

```
GLOBAL = xopt.GLOBAL;
```

1.5 NOMAD

1.5.1 概述

NOMAD 演算法，是使用一種使用數值方法 Mesh Adaptive Direct Search (MADS) filter 演算法，可以用來解決設計變數爲連續或是不連續，拘束條件與目標函數可爲非線性或是線性的問題。NOMAD 藉由使用者介面，來輔助解題。數學形式爲下:

$$\begin{aligned} \min_{x} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{g}(\mathbf{x}) \leq 0 \end{aligned}$$

1.5.2 檔案列表

NOMAD 演算法需要三個 function.m 檔：jobname, jobname_x0, jobname_Omega，其內容可由下列範例說明，其中包含了如何設定必要的參數、目標函數與拘束條件。因爲藉由使用者介面，不需製作 main.m 檔，製作 function 檔即可。

1.5.3 參數設定與範例說明

下面使用一個數學範例來做練習，解決以下問題：

$$\begin{aligned} \min_{x_1,x_2} \quad & 2x_1^2 + x_2^2 - 48x_1 - 40x_2 \\ \text{subject to} \quad & g_1(x) = x_1 + 3x_2 \geq 0 \\ & g_2(x) = 18 - x_1 - 3x_2 \geq 0 \\ & g_3(x) = x_1 + x_2 \geq 0 \\ & g_4(x) = 8 - x_1 - x_2 \geq 0 \\ & 0 \leq x_1, x_2 \leq 6 \\ \text{起始值在} \quad & (x_1, x_2) = (0.1, 0.1) \end{aligned}$$

File: Jobname.m

```
function [fx,cx] =Demo(x)
fx=2*x(1)^2+x(2)^2-48*x(1)-40*x(2); %目標函數
cx=[-x(1)-3*x(2); %拘束條件
-18+x(1)+3*x(2);
-x(1)-x(2)
-8+x(1)+x(2)]
return
```

File: jobname_x0.m

```
function iterate = Demo_x0
iterate(1).x =[0.1;0.1]; %初始設計變數的值
iterate(1).p = {}; %初始設計參數
return;
```

File: jobname_Omega.m

```
function [A,l,u] = Demo_Omega(n);
% Set upper and lower bounds
A = eye(n); %線性拘束條件(A*x<B)與上下限的係數矩陣
l = [0;0]; %Lower Bound of Design Variable(A*X>l)
u = [6;6]; %Upper Bound of Design Variables(A*X<u)
return
```

因此在這個範例中，jobname 就是 Demo，至此就完成了檔案的設定。

1.5.4 操作與結果

完成了 function 檔的設定後，程式操作與檢視結果步驟如下：

步驟一：

在 command window 打“nomadm”呼叫使用者介面，如圖1.6。

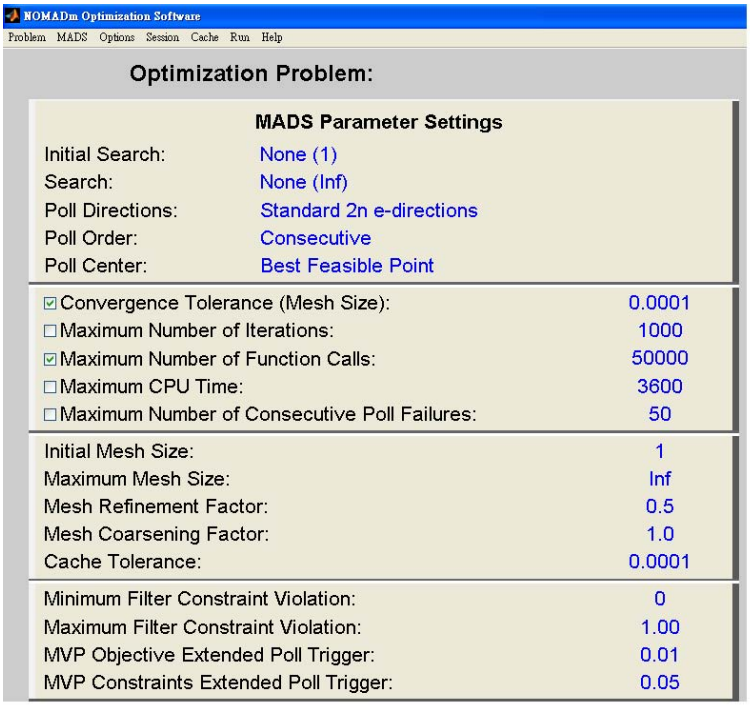


圖 1.6

步驟二：

在主選單 Problem → New Optimization Problem → Jobname，即 Load 檔案。

步驟三：

在主選單 Run → Execute Next Run，即完成了解題。

步驟四：

在主選單 Result → View Run，可以檢視所得到的解果。

*注意：改變 nomad 演算法的參數，主選單 MADS → Edit Terminate Parameter 去改變收斂參數或是演算法中的參數。

P.S: MATLAB 版本：建議 2007a，而2007b、 2008a 使用會有不相容的問題。使用前，使用者需先至網路上下載 nomad 的 Matlab 的 m 檔。

1.6 SuperEGO

1.6.1 概述

SuperEGO 演算法爲 EGO 演算法的一種，主要是針對目標函數或拘束條件是經由電腦模擬或實驗等花費較多時間的方式而取得，所寫的一種非線性全域最佳化演算法，其數學形式可由下列表示：

$$\begin{aligned} \min_{x} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{g}(\mathbf{x}) \leq 0 \end{aligned}$$

1.6.2 檔案列表

爲了執行 superEGO 演算法，使用者必需製作一個 main 檔、一或二個 function 檔，其內容可由下列範例說明，其中包含了如何設定必要的參數、目標函數與拘束條件。

1.6.3 參數設定與範例說明

下面使用一個數學範例來做練習；解決以下問題

$$\begin{aligned} \min_{x_1,x_2} \quad & 2x_1^2 + x_2^2 - 48x_1 - 40x_2 \\ \text{subject to} \quad & g_1(x) = x_1 + 3x_2 \geq 0 \\ & g_2(x) = 18 - x_1 - 3x_2 \geq 0 \\ & g_3(x) = x_1 + x_2 \geq 0 \\ & g_4(x) = 8 - x_1 - x_2 \geq 0 \\ & 0 \leq x_1, x_2 \leq 6 \\ \text{起始值在} \quad & (x_1, x_2) = (0.1, 0.1) \end{aligned}$$

File: main.m

```
fileInfo.expFile = 'myfun_exp'; % expensive function 函數檔名稱
fileInfo.cheapFile = 'myfun_cheap'; % inexpensive function 函數檔名稱
fileInfo.expObjFun = 1; % 設定目標函數是否爲 expensive
lb = [-50, -50]; % 設計空間的 lower bounds
ub = [50, 50]; % 設計空間的 upper bounds
options.maxfCount = 75; % 函數計算次數上限
options.conTol = 1e-5; % 拘束條件的 tolerance
options.display = 1; % 是否顯示迭代過程
options.saveFile = 'myfile'; % 結果儲存名稱
results = superego(fileInfo, lb, ub, options); % 開始執行 superEGO
```

其中目標函數與拘束條件，依照其性質是否爲 expensive，可分別寫成 myfun_exp 與 myfun_cheap 等 function 檔。可由以下範例說明：

File:myfun_exp.m or myfun_cheap.m

```
function [f, g] = KS224(x)
f = 2*x(:,1).^2+x(:,2).^2-48*x(:,1)-40*x(:,2); % 目標函數
g(1) = -x(:,1)-3*x(:,2); % 拘束條件
g(2) = -18+x(:,1)+3*x(:,2);
g(3) = -x(:,1)-x(:,2);
g(4) = -8+x(:,1)+x(:,2);
```

此 function 檔名稱爲 KS224，若是將此處的拘束條件認定爲 expensive function，則將 script 檔中的 myfun_exp 改爲 KS224，以此類推。(注意：目標函數不管寫在哪一個檔，均可以在 fileInfo.expObjFun 處自行定義性質)。

而在 myresult 中將儲存演算過後的結果，其詳細內容與意義可由下表所示：

xBest	找到的最佳位置
fBest	找到的最佳解
gbest	在最佳位置的拘束條件情形
lb	變數的lower bounds
ub	變數的upper bounds
fCount	function的計算次數
fCountCheap	cheap function的計算次數
iter	迭代次數
history	迭代過程紀錄
inputs	迭代過程中輸入的變數
outputs	迭代過程中輸出的值
kparam	kriging model的參數
infill	迭代過程中的ISC與其計算值
flag	superEGO收斂與否
prob	問題定義

Chapter 2

工程範例

2.1 Wheelchair Brake

Problem Definition

Figure 2.1 shows a wheelchair brake in its retracted position with a tension spring exerting a force of 4 N, which holds the handle against the pin stop. When the handle is moved clockwise, pivot A drops below the axis of the spring, and the spring acts to hold the brake shoe against the tire. Available space limits the outside diameter of the spring to 10mm. Determine a optimal combination of diameter of spring D, diameter of wire d, number of active coils of spring N, and free length l_0 of the coiled section with minimal material volume.

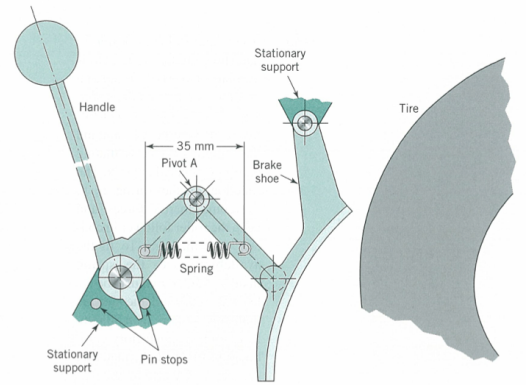


圖 2.1: Wheelchair brake system

- **Requirements**
  - Minimize the volume of the spring $f(d, D, N)$;
  - Number of coils of the spring N must be a positive integer.
- **Given conditions**
  - $G = 79\text{GPa}$;
  - Max force $F_{\max}$ occurs at $l = 35\sqrt{2}$;
  - Safety factor $F_{\text{s}} = 2$;
  - Initial point (l_0,d,D,N)=(11.5, 0.5, 9, 20).

Solution

1. 最佳化數學表示式：

$$\min \quad f(d, D, N) = \frac{1}{4}\pi^2 d^2 DN$$

$$\begin{aligned} \text{s.t.} \quad & F_s \cdot \tau(l_0, d, D) \leq \tau_{\max}(d) \\ & D + d \leq 10 \\ & l_0 = (N+3)d \\ & k = \frac{F}{\Delta x} = \frac{d^4 G}{8D^3 N} \end{aligned}$$

$$\text{w.r.t.} \quad l_0, d, D, N$$

$$\text{where} \quad \tau = \frac{8\,F_{\max} D}{\pi^3 d}\left(1 + \frac{0.615\,d}{D}\right)$$

$$\tau_{\max}(d) = 0.4 \cdot 2060 d^{-0.163}$$

2. 使用 fmincon 進行最佳化後，可得最佳值與最佳解：

$$(l_0, d, D, N) = (19.16, 0.7023, 7.3180, 24.2784) \qquad f = 216.2251$$

表 2.1: Unit of Variables

Variable	D	d	F	G	l_0	τ	$\tau_{\max}$
Unit	mm	mm	N	MPa	mm	MPa	MPa

3. N must be a positive integer

$$\text{set} \quad l_0 = 22.4 \quad d = 0.8 \quad D = 7.9877 \quad N = 25$$

$$f = 315.342$$

2.2 Truss

定義上，桁架 (truss) 是以多個長桿由端點互相連接而成的結構，因爲桿件的數量和尺寸等規格皆可以變動，可以組合出的幾何型態更是非常多變，很容易適用於各種工程應用的場域，因此桁架工程應用甚廣，是房屋、橋梁等建築非常常見的結構。

處理桁架問題時所套用的物理模型，通常使用以下假設：

- 外力與內力均作用於桿件端點上；
- 各桿件間的連接簡化爲銷釘連接 (pin connection)；
- 忽略桿件本身重量之影響，因此各桿件視爲二力構件 (two-force member)；
- 定義各桿件之受力正方向爲拉伸，負方向爲壓縮。

本章節將以二維的十桿桁架 (ten-bar truss) 和三維的二十五桿桁架 (twenty-five-bar truss) 爲例，藉由有限元素法 (finite element method，FEM) 將每一桿件視爲個別的元素，計算其受外力的反應。

2.2.1 Ten-bar Truss

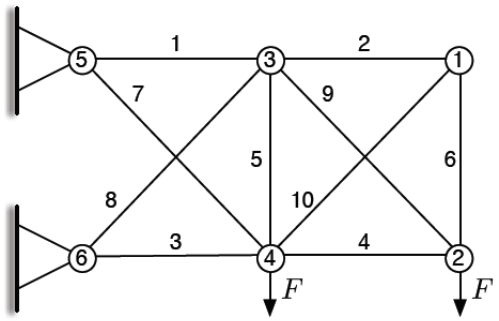


圖 2.2: 十桿桁架結構示意圖

十桿桁架 (ten-bar truss) 是典型的桁架結構之一，如圖 2.2 所示。此節以十桿桁架範例，演示如何使用有限元素法解決桁架的問題。

Problem Definition

在以下的已知條件下，給定桿件截面半徑，試求各桿件的**位移**、**應力**與**反作用力**：

- 整體架構處在靜力平衡的情況下
- 所有桿件截面皆爲圓形
- 材料爲鋼，楊氏係數 $E = 200$ GPa，密度 $\rho = 7860$ kg/m^3，降伏強度 $\sigma_y = 250$ MPa
- 平行桿件與鉛直桿件（桿件1至桿件6）長度皆爲 9.14 m
- 桿件 1 至桿件 6 截面半徑相同爲 r_1，桿件 7 至桿件 10 截面半徑相同爲 r_2
- 所有桿件半徑的最佳化範圍爲 0.001 至 0.5 m 之間
- 在節點 2 和節點 4 上的負載 F 皆爲 1.0×10^7 N 向下

Solution

1. 最佳化數學表示式：

$$\begin{aligned} \min_{r_1,\, r_2} \quad & f(r_1, r_2) = \sum_{i=1}^{6} m_i(r_1) + \sum_{i=7}^{10} m_i(r_2) \\ \text{subject to} \quad & |\boldsymbol{\sigma}_i| \leq \sigma_y \\ & Q \leq 0.02 \\ \text{where} \quad & f : \text{所有桿件的質量} \\ & Q : \text{node 2 的位移} \\ & \sigma_y : \text{降伏應力} \\ & \boldsymbol{\sigma}_i : \text{所有桿件的應力} \end{aligned}$$

2. 利用有限元素法分析 10-bar truss problem：
   將有限元素法應用在桁架，各桿件視爲元素 (element)、元素間的連接爲節點 (node)，求解流程如圖2.3。首先建立元素表格 (element table)，利用表格中的資料計算出剛性矩陣 (stiffness matrix)，再以力、剛性和位移三者之間的關係找出所求的位移，最後應

力和反作用力也能透過位移計算而得。

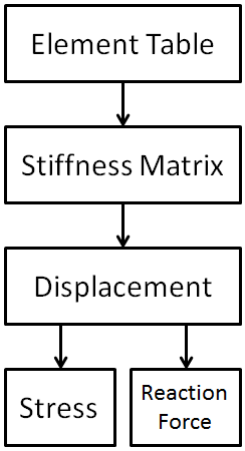


圖 2.3: 有限元素法流程

以下提供有限元素法的步驟，運算的過程可以利用 MATLAB 軟體輔助，將此方法撰寫成程式執行。

Step 1 Element Table

將各元素和節點編號後 (本節以圖 2.2 中的編號爲主)，先記錄各節點的座標，方便計算元素表格使用：

表 2.2: 節點座標

node	x	y
1	18.28	9.14
2	18.28	0
3	9.14	9.14
4	9.14	0
5	0	9.14
6	0	0

首先在表格中填入各元素兩端的節點，再計算面積、長度、角度等資料：

彈性係數 E 爲題目已知，截面積以半徑 r_1 或 r_2 透過圓面積公式計算：

$$A = \pi r^2 \tag{2.1}$$

表 2.3: 元素表格

Element	node i	node j	E	A	L	cos	sin
1	3	5					
2	1	3					
3	4	6					
4	2	4					
5	3	4					
6	1	2					
7	4	5					
8	3	6					
9	2	3					
10	1	4					

利用各元素兩端的節點座標，以畢氏定理計算出桿長：

$$L = \sqrt{(x_{\mathrm{node\,j}} - x_{\mathrm{node\,i}})^2 + (y_{\mathrm{node\,j}} - y_{\mathrm{node\,i}})^2} \tag{2.2}$$

以長度除兩端節點的 x 座標變量即角度的 cos 值，除 y 座標變量即角度的 sin 值：

$$\cos\theta_e = \frac{(x_{\mathrm{node\,j}} - x_{\mathrm{node\,i}})}{L} \tag{2.3}$$

$$\sin\theta_e = \frac{(\mathrm{y}_{\mathrm{node\,j}} - y_{\mathrm{node\,i}})}{L} \tag{2.4}$$

將式 2.1 至 2.4 計算出所有元素的規格和位置統整至表 2.3，便可得到完整的元素表格，此表囊括下一步驟計算剛性矩陣所需的所有數據。

Step 2 Stiffness Matrix

每個元素末端連接兩個節點，節點又分別有 x 和 y 兩個方向的自由度 (degree of freedom，DOF)，以 4×4 的剛性矩陣 (stiffness matrix，數學式中以 **k** 表示) 表

示元素中4個自由度上位移和受力之間的相互關係：

$$\mathbf{k}^e = \frac{EA_e}{L_e}\begin{bmatrix} c^2 & cs & -c^2 & -cs \\ cs & s^2 & -cs & -s^2 \\ -c^2 & -cs & c^2 & cs \\ -cs & -s^2 & cs & s^2 \end{bmatrix} \tag{2.5}$$

其中 c 表示 $\cos\theta_e$；s 表示 $\sin\theta_e$。

桁架整體結構中有 6 個節點，總共有 $2 \times 6 = 12$ 個自由度，以 node1 在 x 方向爲 DOF1、 node1 在 y 方向爲 DOF2、node2 的 x 方向爲 DOF3.....依 node1 到 node6 的順序和 x、y 的順序編號 12 個自由度。透過元素表格揭露的數據代入式 2.5，建立各元素的剛性舉陣，以元素 2 及元素 6 爲例：

$$\mathbf{k}^2 = \frac{EA_2}{9.14}\begin{matrix} \\ \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{matrix}\begin{matrix} \overset{1\quad 2\quad 5\quad 6}{\leftarrow\downarrow \text{Global DOF}} \\ \begin{matrix} 1 \\ 2 \\ 5 \\ 6 \end{matrix} \end{matrix} \tag{2.6}$$

$$\mathbf{k}^6 = \frac{EA_6}{9.14}\begin{matrix} \\ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix} \end{matrix}\begin{matrix} \overset{1\quad 2\quad 3\quad 4}{\leftarrow\downarrow \text{Global DOF}} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} \end{matrix} \tag{2.7}$$

10 個元素皆參考元素表格代入 2.5 推導出剛性矩陣後，按照所對應的自由度，整合成 12×12 的整體剛性矩陣 $\mathbf{K}$：

$$\mathbf{K} \leftarrow \sum_e \mathbf{k}^e \tag{2.8}$$

Step 3 Displacement

爲了利用力、剛性和位移三者之間的關係：

$$\mathbf{F} = \mathbf{KQ} \tag{2.9}$$

承 Step2 設定整體系統的 12 個自由度，建立對應 12 個自由度的 12×1 力向量和 12×1 位移向量。

$$\mathbf{F} = \begin{bmatrix} F_1 & F_2 & ... & F_{12} \end{bmatrix}^T \tag{2.10}$$

$$\mathbf{Q} = \begin{bmatrix} Q_1 & Q_2 & ... & Q_{12} \end{bmatrix}^T \tag{2.11}$$

套用邊界條件，node5 和 node6 爲固定端，在 x、y 方向均無位移：

$$Q_9 = Q_{10} = Q_{11} = Q_{12} = 0 \tag{2.12}$$

由於 DOF9 至 DOF12 無位移，可以將剛性矩陣 $\mathbf{K}$、力向量 $\mathbf{F}$、位移向量 $\mathbf{Q}$ 中相對應的行與列消除。剛性矩陣 $\mathbf{K}$ 截取第 1 行到第 8 行、第 1 列到第 8 列，簡化成 8×8 的 $\mathbf{K}_{reduced}$；力向量 $\mathbf{F}$ 和移向量 $\mathbf{Q}$ 簡也只取前 8 個，化簡成 8×1 的 $\mathbf{F}_{reduced}$ 和 $\mathbf{Q}_{reduced}$：

$$\mathbf{K}_{\text{reduced}} = \begin{bmatrix} K_{1,1} & K_{1,2} & \cdots & K_{1,8} \\ K_{2,1} & K_{2,2} & \cdots & K_{2,8} \\ \vdots & \vdots & \ddots & \vdots \\ K_{4,1} & K_{8,2} & \cdots & K_{8,8} \end{bmatrix} \tag{2.13}$$

$$\mathbf{F}_{\text{reduced}} = \begin{bmatrix} F_1 & F_2 & ... & F_8 \end{bmatrix}^T \tag{2.14}$$

$$\mathbf{Q}_{\text{reduced}} = \begin{bmatrix} Q_1 & Q_2 & ... & Q_8 \end{bmatrix}^T \tag{2.15}$$

利用式 2.9，同乘 $\mathbf{K}^{-1}_{reduced}$ 後，即可計算出各節點在x 和 y 方向上位移：

$$\begin{aligned} \mathbf{K}_{\text{reduced}}\mathbf{Q}_{\text{reduced}} &= \mathbf{F}_{\text{reduced}} \\ \mathbf{Q}_{\text{reduced}} &= \mathbf{K}^{-1}_{\text{reduced}}\mathbf{F}_{\text{reduced}} \end{aligned} \tag{2.16}$$

總而言之，Q_1 至 Q_8 由式 2.16 求得，Q_9 到 Q_{12} 爲零。

Step 4 Stress

應力與應變的關係如下：

$$\sigma = E \times \varepsilon \tag{2.17}$$

其中，σ 爲應力；ε 爲應變；E 爲楊氏係數。

應變的定義爲：

$$\varepsilon = \frac{\delta}{L} \tag{2.18}$$

其中，δ 爲長度變量。

元素的長度變量可以透過節點在各方向的位移作三角函數的轉換得到：

$$\boldsymbol{\sigma} = \frac{E_e}{l_e}\begin{bmatrix} -c & -s & c & s \end{bmatrix}\mathbf{Q} \tag{2.19}$$

Step 5 Reaction Force

在此結構上，反作用力會在固定端 node5 和 node6 產生，其對應到的自由度爲 DOF9 到 DOF12，可以透過系統所有節點的位移計算而得。因此，在剛性矩陣中我們取第 9 列到第 12 列：

$$\mathbf{K}_{\text{reaction}} = \begin{bmatrix} K_{9,1} & K_{9,2} & \cdots & K_{9,12} \\ K_{10,1} & K_{10,2} & \cdots & K_{10,12} \\ K_{11,1} & K_{11,2} & \cdots & K_{11,12} \\ K_{12,1} & K_{12,2} & \cdots & K_{12,12} \end{bmatrix} \tag{2.20}$$

將式 2.20 的矩陣乘位移矩陣 $\mathbf{Q}$ 即可得到反作用力：

$$\begin{Bmatrix} R_9 \\ R_{10} \\ R_{11} \\ R_{12} \end{Bmatrix} = \mathbf{K}_{\text{reaction}}\mathbf{Q} \tag{2.21}$$

以上爲有限元素法於十桿桁架的流程，在給定設計變數 r_1 和 r_2 的條件，透過 5 個步驟中提供的公式，可計算出各桿件的位移、應力和反作用力的方法，式 2.16 所得的位移、式 2.19 所得的應力以及 2.21 所得的反作用力即爲所求。

3. 使用 `fmincon` 進行最佳化後，可得其最佳值與最佳解：

$$(r_1, r_2) = (0.3, 0.2663) \qquad f = 212410$$

其設計空間、可行解空間與目標函數值表示如圖2.4。

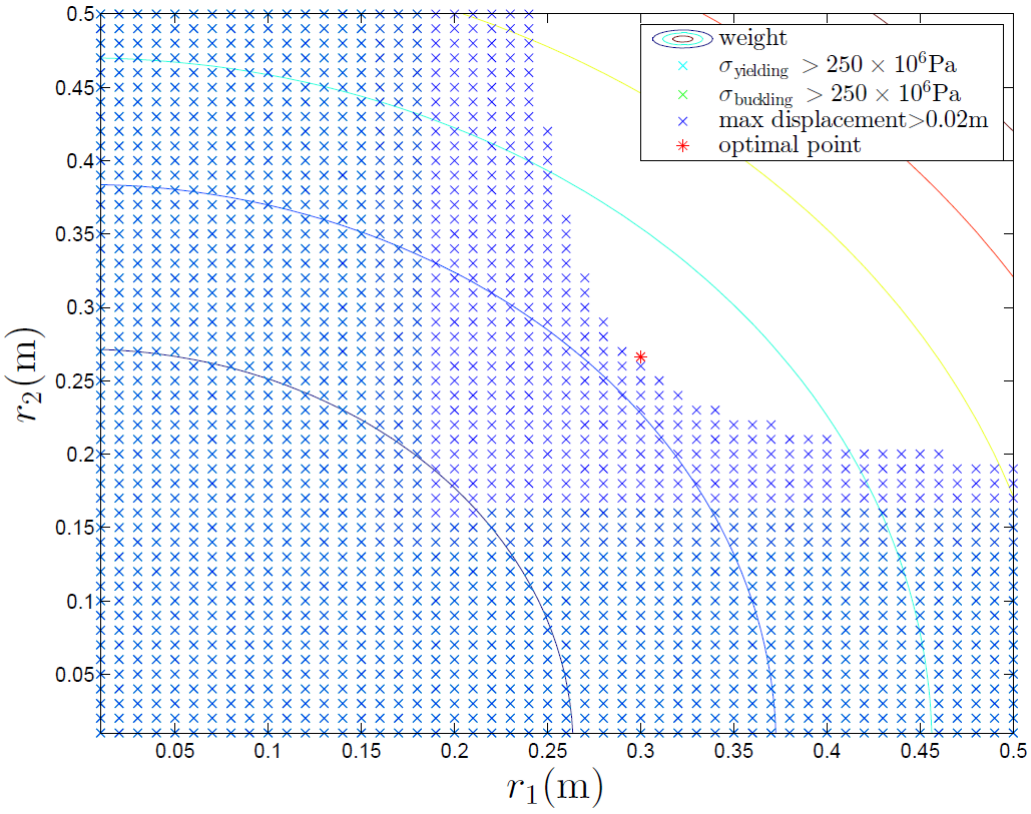


圖 2.4: 10-bar truss 設計空間

2.2.2 Twenty-five-bar Truss

Problem Definition

This is a twenty-five-bar truss structure problem.

- Assumptions:
  - 整體架構處在靜力平衡的情況下；
  - 忽略桿件本身重量之影響；
  - 定義各桿件之受力 F 之正方向爲拉伸方向。
- Given:
  - 給定了各個桿件的位置與長度。

Solution

利用有限元素法來作分析步驟與 10 桿件相同，因爲這爲一個三維的空間衍架，因此會比平面二維的衍架多增加一個自由 (D.O.F.) 因此 nodal coordinate data 將會增加 z 方向，因此 element table 會增加一個 n 來代表其自由度。所以 element stiffness matrix 也會改變形態。但 global stiffness matrix 算法還是一樣。

1. The nodal coordinate data:

node	x	y	z
1	x_1	y_1	z_1
⋮	⋮	⋮	⋮
10	x_10	y_10	z_10

2. The element table:

Element	node i	node j	E	A	l_e	l	m	n
1	$Node_{1i}$	$Node_{1j}$						
$\vdots$	$\vdots$	$\vdots$						
25	$Node_{25i}$	$Node_{25j}$						

where

$$l_e = \sqrt{(x_{node\_i} - x_{node\_j})^2 + (y_{node\_i} - y_{node\_j})^2 + (z_{node\_i} - z_{node\_j})^2}$$

$$l = \frac{(x_{node\_i} - x_{node\_j})}{l_e}$$

$$m = \frac{(y_{node\_i} - y_{node\_j})}{l_e}$$

$$n == \frac{(z_{node\_i} - z_{node\_j})}{l_e}$$

3. Element stiffness matrix:

$$\mathbf{k}^e = \frac{E_i A_i}{L_i} \begin{bmatrix} l^2 & lm & ln & -l^2 & -lm & -ln \\ lm & m^2 & mn & -lm & -m^2 & -mn \\ ln & mn & n^2 & -ln & -mn & -n^2 \\ -l^2 & -lm & -ln & l^2 & lm & ln \\ -lm & -m^2 & -mn & lm & m^2 & mn \\ -ln & -mn & -n^2 & ln & mn & n^2 \end{bmatrix}$$

4. Global stiffness matrix:

$$\mathbf{K} \leftarrow \sum_e \mathbf{k}^i$$

由十桿件上所述的方法組合成一 30×30 之 $\mathbf{K}$ 矩陣。

5. Set boundary condition:

$$\mathbf{Q} = \begin{bmatrix} Q_1 & Q_2 & \dots & Q_{30} \end{bmatrix}^T$$

圖中，節點 7、8、9、10 爲固定端，在 x、y、z 方向均無位移，故

$$Q_{19} = Q_{20} = \ldots = Q_{30} = 0$$

6. Reduce stiffness matrix:
   由於$Q_{19} = Q_{20} = \ldots = Q_{30} = 0$，可將 $\mathbf{K}$ 矩陣之 19 至 30 行列消除，形成一 18×18 矩陣。

7. Solve:

$$\mathbf{KQ} = \mathbf{F}$$
$$\mathbf{Q} = \mathbf{K}^{-1}\mathbf{F}$$

最後輸入力、截面積、楊式係數、位移中的其中三個就可以找出另一個的解。

Bibliography

[1] Tirupathi R. Chandrupatla, Ashok D. Belegundu, Introduction to Finite Elements in Engineering, 3rd ed. Pearson Education, 2002.