# Orientation Homework Report

10-bar truss optimization

徐若瑄

August 24, 2021

# 1 問題描述

十桿衍架 (10-bar truss) 由十個長桿件組成並且有 6 個端點，其中第 5、6 號端點為固定端，桿件的配置如圖 1所示。

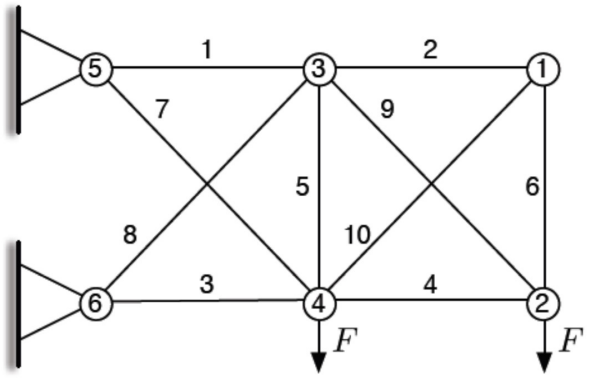


Figure 1: 10-bar truss

所有桿件的截面皆為圓形，桿件 1 到桿件 6 的截面半徑同為 $r1$ 且長度為 $9.14\ m$；桿件 7 到桿件 8 的截面半徑同為 $r2$。桿件所使用的材料為剛，相關的材料性質有：

1. 密度 $\rho = 7860\ kg/m^3$
2. 楊氏模數 (Young's Modulus) $E = 200\ GPa$
3. 降伏係數 (Yield stress) $\sigma_y = 250\ MPa$

端點 4 及端點 5 有一向下的外力 $F = 1.0x10^{-}7\ N$，在所有桿件的應力不超過降伏應力且端點 2 的位移小於 $0.02\ m$ 的條件下，求桿件截面半徑 $r1$ 與 $r2$ 的值，使得十桿衍架整體重量為最輕。

# 2 計算方法

計算十桿衍架最小重量主要使用兩個方法，首先利用有限元素法 (Finite element method) 計算出所有桿件的應力值、各個端點的位移量，接著進行最佳化即可得到最佳值與最佳解。

以下所有計算過程皆使用 Matlab 程式輔助計算。

## 2.1 有限元素分析

有限元素分析的關鍵步驟如下：

1. 建立元素表 (Element Table)：元素表內包含各桿件的長度、截面積、角度等
2. 建立剛性矩陣 (Stiffness Matrix)：利用元素表計算出剛性矩陣 $K$，表示出 6 個端點的 12 個自由度上下位移與受力之間的關係
3. 計算各個節點的位移：透過 $F = KQ$ 的關係式求出各點在 x,y 兩方向的位移量
4. 計算各桿件的應力：透過 $\sigma = E\varepsilon$ 的關係式求出各桿件的應力
5. 計算反作用力：同樣使用 $F = KQ$ 的關係式求出固定端點 5、6 的反作用力
6. 最佳化：將問題條件轉換為數學式，最佳化後得到最佳值與最佳解

### 2.1.1 建立端點與桿件參數

為了快速建立元素表，首先將桿件長度轉換為各端點的座標位置 (以端點 6 為原點) 並存於 `nodeTable`

```
%% Create Node Table
nodeTable = [length*2 length;
             length*2 0;
             length*1 length;
             length*1 0;
             0 length;
             0 0];
```

接著設定桿件 1 到桿件 10 對應到的左右兩截點編號。利用 `nodeInfo` 紀錄各端點連接的桿件編號，再將桿件兩端點的編號存於 `elementToNode`

```
%% Create element to node array
nodeInfo = {[2 6 10],[4 6 9],[1 2 5 8 9],[3 4 5 7 10],[1 7],[3
8]};
elementToNode = zeros(10,2);
index = ones(10,1);

for i = 1:6
    for j = 1:size(nodeInfo{i},2)
        element = nodeInfo{i}(j);
        elementToNode(element,index(element)) = i;
        index(element) = index(element) + 1;
    end
end
```

### 2.1.2 建立元素表

元素表 `elementTable` 為 10x4 的矩陣，每一欄依序紀錄各桿件的

1. 長度 $L = \sqrt{(x_{nodej} - x_{nodei})^2 + (y_{nodej} - y_{nodei})^2}$
2. 截面積 $A = \pi \times r^2$
3. 餘弦 $cos\theta_e = \dfrac{(x_{nodej} - x_{nodei})}{L}$
4. 正弦 $sin\theta_e = \dfrac{(y_{nodej} - y_{nodei})}{L}$

以 `elementToNode` 搭配 `nodeTable` 取得各端點的 $x, y$ 座標值，根據上方提供的公式計算各項數值

```
%% Create the elementTable using elementToNode
%%% (10,4) = [A L cos sin]
elementTable = zeros(10,4);

for i = 1:10
    % Node of Element
    nodei_x = nodeTable(elementToNode(i,1),1);
    nodei_y = nodeTable(elementToNode(i,1),2);
    nodej_x = nodeTable(elementToNode(i,2),1);
    nodej_y = nodeTable(elementToNode(i,2),2);

    % Area
    if i < 7
        elementTable(i,1) = pi * r1.^2;
    else
        elementTable(i,1) = pi * r2.^2;
    end

    % Length of element
    elementTable(i,2) = sqrt(power(nodej_x - nodei_x
,2)+power(nodej_y - nodei_y ,2));

    % cos
    elementTable(i,3) = (nodej_x - nodei_x)/elementTable(i,2);

    % sin
    elementTable(i,4) = (nodej_y - nodei_y)/elementTable(i,2);
end
```

### 2.1.3 剛性矩陣

剛性矩陣 (Stiffness Matrix) 為表示各端點的位移與受力關係的矩陣，在數學式中以 $K$ 表示。各桿件皆存在左右兩端點，因此其剛性矩陣大小為 4x4，表示出 4 個自由度 (Degree of freedom,DOF) 位移與受力的關係，其公式如下：

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

十桿衍架有 6 個端點且每個端點皆有 2 個 DOF，端點 1 的 $x$ 方向 DOF 以 1 表示；端點 1 的 $y$ 方向 DOF 以 2 表示，依序到第 12 個 DOF 表示端點 6 的 $y$ 方向。以桿件 2 及桿件 6 的剛性矩陣為範例，其矩陣各行及各列所表示的 DOF 依序為左端點的 $x$ 方向、左端點的 $y$ 方向、右端點的 $x$ 方向及右端點的 $y$ 方向：

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

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

將所有桿件的剛性矩陣加總即可得到十桿衍架整體的剛性矩陣：

$$\mathbf{K} \leftarrow \sum_{e=1}^{10} \mathbf{k}^e$$

`elementTable` 中提供計算剛性矩陣所需的 $L_e, A_e, cos, sin$，以 `elementToNode` 取得桿件的 4 個 DOF 值，並依序將計算的結果加總至 `stiffnessMatrix` 中對應的 DOF 欄位

```
    stiffnessMatrix = zeros(12,12);
    globalDOF = zeros(1,4);

    for i = 1:10
        globalDOF(1,1) = elementToNode(i,1) * 2 -1;
        globalDOF(1,2) = elementToNode(i,1) * 2;
        globalDOF(1,3) = elementToNode(i,2) * 2 -1;
        globalDOF(1,4) = elementToNode(i,2) * 2;

        cos = elementTable(i,3);
        sin = elementTable(i,4);

        tmpMatrix = [cos;sin;-cos;-sin] * [cos sin -cos -sin];

        for j = 1:4
            for k = 1:4
                stiffnessMatrix(globalDOF(j),globalDOF(k)) =...
                stiffnessMatrix(globalDOF(j),globalDOF(k)) + ...

   E*elementTable(i,1)*tmpMatrix(j,k)/elementTable(i,2);
            end
        end
    end
```

### 2.1.4 節點位移

使用前一步驟所得到的剛性矩陣及已知的受力，即可求得各節點的位移：

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

由於端點 5、端點 6 為固定點，因此 $x, y$ 方向位移皆為 0：

$$Q_9 = Q_{10} = Q_{11} = Q_{12} = 0$$

也因此可將剛性矩陣簡化為 8x8 的矩陣：

$$\mathbf{F}_{8\times1} = \mathbf{K}_{8\times8} \times \mathbf{Q}_{8\times1}$$

等式兩邊同乘 $K^{-1}$：

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

將原為 12x12 的 `stiffnessMatrix` 複製到 `stiffness_disp` 並轉換為 8x8 的矩陣，利用上述公式計算 DOF1 DOF8 的位移量

```
%% Displacement
force = zeros(8,1);
force(4) = -1E+07;  % 1.0 x 10^7 N
force(8) = -1E+07;

% Duplicate the 1~8rows&column of stiffnessMatrix
stiffness_disp = zeros(8);
for i = 1:8
    for j = 1:8
        stiffness_disp(i,j) = stiffnessMatrix(i,j);
    end
end

% stiffness * disp = force
disp = inv(stiffness_disp) * force;
```

### 2.1.5 桿件應力

楊式係數 (Young's Modulus) 的定義為應力與應變的比值：

$$E = \frac{\sigma}{\varepsilon}$$

其中應變 $\varepsilon$ 可由上一步驟所得到的位移轉換而成：

$$\varepsilon = \frac{\delta}{L} = \frac{1}{L}\begin{bmatrix} -c & -s & c & s \end{bmatrix} \times \mathbf{Q}$$

因此各桿件所受到的應力大小為：

$$\sigma_e = \frac{E_e}{L_e}\begin{bmatrix} -c & -s & c & s \end{bmatrix} \times \mathbf{Q}$$

同樣使用 `elementTable` 取得 $L_e, cos, sin$ 的值，並搭配 `elementToNode` 將各桿件的相關 DOF 位移存於 `disp_element`，最後利用上述公式求得桿件應力

```
%% Stress
stress = zeros(10,1);
trans = zeros(1,4);
disp_element = zeros(4,1);
for i = 9:12
    disp(i,1) = 0;
end

for i = 1:10
    cos = elementTable(i,3);
    sin = elementTable(i,4);

    trans(1,1) = -cos;
    trans(1,2) = -sin;
    trans(1,3) = cos;
    trans(1,4) = sin;

    disp_element(1,1) = disp(elementToNode(i,1) * 2 -1,1);
    disp_element(2,1) = disp(elementToNode(i,1) * 2,1);
    disp_element(3,1) = disp(elementToNode(i,2) * 2 -1,1);
    disp_element(4,1) = disp(elementToNode(i,2) * 2,1);
```

```
        stress(i) = E*(trans*disp_element)/elementTable(i,2);

    end
```

### 2.1.6 反作用力

由於端點 5、端點 6 為固定點，因此在此會有反作用力的產生。與 2.1.4 中提到的計算方式相同，因為只計算 DOF9 DOF12 的作用力，因此將剛性矩陣調整為 4x12 的大小：

$$\mathbf{F}_{4\times1} = \mathbf{K}_{4\times12} \times \mathbf{Q}_{12\times1}$$

將原為 12x12 的 `stiffnessMatrix` 複製到 `stiffness_reac` 並轉換為 4x12 的矩陣，利用上述公式計算反作用力數值

```
    %% Reaction Force
    % Duplicate the 9~12rows&column of stiffnessMatrix
    stiffness_reac = zeros(4,12);
    for i = 1:4
        for j = 1:12
            stiffness_reac(i,j) = stiffnessMatrix(i+8,j);
        end
    end

    % R = K * Q
    reactionForce = stiffness_reac * disp;
```

## 2.2 最佳化

最佳化數學表示式為：

$$\begin{aligned} \min_{r1,r2} \quad & f\left(r1,r2\right) = \sum_{i=1}^{6} m_i(r_1) + \sum_{i=7}^{10} m_i(r_2) \\ \text{subject to} \quad & |\sigma_i| \leq \sigma_y \\ & \Delta s_2 \leq 0.02 \end{aligned}$$

於 Matlab 中使用 `fmincon` 進行最佳化運算，將計算最小值運算式存於 **`obj.m`**

```
function f = obj(r)

    % Constants
    global length density

    f = 0;

    % Element 1~6
    for i = 1:6
        f = f + density * pi * r(1,:).^2 * length;
    end

    % Element 7~10
    for i = 1:4
        f = f + density * pi * r(2,:).^2 * sqrt(length.^2*2);
    end

end
```

限制條件則於 **`nonlcon.m`** 中進行設定，並於此副函式執行 **`finiteElementMethod.m`** 的有限元素分析以取得位移及應力數值

```
function [ g,geq ] = nonlcon(r)

    %Variable
    global yieldStress

    % Finite Element Method
    [disp, stress] = finiteElementMethod(r(1,:),r(2,:));
```

```
    % Q(2) <= 0.02
    % g(1) = sqrt(Q(2).x^2 + Q(2).y^2)-0.02 <= 0
    g(1) = sqrt(disp(3).^2 + disp(4).^2) - 0.02;

    % abs(stress) <= YieldStress = 250 MPa
    % g(2~11) = abs(stress(1~10)) - yield <= 0
    for i = 1:10
        g(i + 1) = abs(stress(i,:)) - yieldStress;
    end

    geq = [];

end
```

```
function [ disp, stress] = finiteElementMethod( r1,r2 )
```

主函式 **Hw.m** 則用來執行 **fmincon** 計算最佳值、最佳解；同時於主函式中存取常數 (楊式係數、密度、長度、降伏應力) 的值並設為廣域變數，日後修改參數及程式維護較便利，及函式間的取值也較不易有錯誤產生

```
% Variable setting of the fmicon function
r0=[0.1;0.1];    %°_©l I
A=[];            %½u© £µ¥¦¡© ¥Y½Ux°}
b=[];            %½u© £µ¥¦¡© ¥Y¼V¶q AX <= b
Aeq=[];          %½u© £µ¥¦¡© ¥Y¼V¶q
beq=[];          %½u©¥¦¡© ¥Y¼V¶q AeqX = beq
ub=[0.5;0.5];        %³]pªŶ¡ªºupper bounds
lb=[0.001;0.001];        %³]pªŶ¡ªºlower bounds
options = optimset('display','off','Algorithm','sqp');

[r,fval,exitflag] = fmincon(@(r)obj(r),r0,A,b,Aeq,beq,lb,ub,...
                            @(r)nonlcon(r),options);
```

```
global length E density yieldStress

length = 9.14;          % 9.14 m
E = 200*10.^9;          % 200 GPa
density = 7860;         % 7860 kg * m^-3
yieldStress = 2.5E+08;  % 250 MPa
```

# 3 結論