MANE 6710 - Numerical Design Optimization Lab 2

Human 6966

October 20 2024

Table of Contents:

Executive Summery

Wing spars are an important part of aviation design because they support the loading of airplane wings, enabling airplanes to stay in the air. These must be as light as possible as this increases the load capacity of the aircraft, which is an important consideration for most aircraft designs. This lab focuses on choosing and using optimization algorithms in Matlab (fmincon()) along with the complex stem method of approximating gradients to optimize the profile of a wing spar to minimize their mass. A two-dimensional discretized version of the Euler-Bernoulli Beam Theory was chosen to analyze the structural system, and the SQP algorithm for fmincon() was selected to optimize the profile of the wing spar. The SQP algorithm was successfully able to minimize the mass of the wing spar within the problem constraints by minimizing the thickness and outer radius of the spar.

1 Analysis Introduction

1.1 Methodology

The method used to analyze the stress in the beam was the Euler-Bernoulli Beam Bending Theory (Equation 1).

$$\frac{d^2}{dx^2}(EI_{yy}\frac{d^2w}{dx^2}) = q, \forall x \in [0, L] \tag{1}$$

Where:

- w: the vertical displacement
- q(x): the applied load
- E: the Young's modulus
- I_{yy}: Second-moment of area with respect to the y-axis

With the initial conditions:

- $w(0) = 0$: No displacement at the root
- $\frac{dw}{dx}(0) = 0$: No angular displacement at the root
- $\frac{d^2w}{dx^2}(L) = 0$: No stress at tip
- $\frac{d^3w}{dx^3}(L) = 0$: No stress at tip

Equation 1 can be solved for the magnitude of the normal stress (equation 2).

$$\sigma_{xx}(x) = |-r_{outer}E\frac{d^2w}{dx^2}| \tag{2}$$

This is discretized using the finite-element method, with the solution represented using Hermite-cubic shape functions derived from the minimization of the potential energy function. The distributed load is given by equation 3 with $q_L = 0$ and F being the total applied load.

$$q(x) = (1 - \frac{x}{L})q_0 + \frac{x}{L}q_L, q_0 = \frac{2 * F}{L} - q_L \tag{3}$$

1.2 Assumptions

To simplify the physical model and fundamental equations, the following assumptions were made:

1. Planar symmetry on the logitudinal axis
2. Smoothly varying cross section
3. Plane sections that are normal to longitudinal plane before bending remain normal after bending
4. Internal strain energy accounts only for bendding moment deforrmations
5. Deformations are small enough that nonlinear effects are negligible
6. Carbon fiber is an elastic and isotropic material and has the following propeties:
   (a) E=70 GPa
   (b) $\sigma_{ultimate}$=600 MPa
   (c) ρ=1600 kg/m^3

1.3 Limitations

These assumptions limit this analysis method to static systems with radial symmetrical cross-sections. Additionally, though it is assumed to be an isotropic material to simplify the model, composites like carbon fiber are highly anisotropic meaning the loading direction has a significant impact on the physical properties of the material (like the Young's modulus), resulting in a large error in the results. Additionally, the assumption that nonlinear effects are negligible is only viable for the elastic region of loading (yield stress). As many of these conditions would not be met in the real world, there is expected to be significant error in the result of this analysis

2 Parameterization of the Geometry

The shape of the wing spar was defined by linear interpolation between the inner and outer radii of adjacent nodes.

$$r_{in}(x^*) = (1-x) * r_{in}(i) + x * r_{in}(i+1), i = floor(\frac{N * x^*}{L}), x = \frac{x^*}{L} \tag{4}$$

$$r_{out}(x^*) = (1-x) * r_{out}(i) + x * r_{out}(i+1), i = floor(\frac{N * x^*}{L}), x = \frac{x^*}{L} \tag{5}$$

These equations were then descritized into a two-dimensional mesh.

3 Optimization Method and Limitations

The SQP (Sequential Quadratic Programming) Algorithm was chosen for fmincon() (Matlab's nonlinear optimization problem solver [1]) to solve the minimization problem. The SQP algorithm uses a quasi-Newton update method to approximate the Hessian, which is used to solve the quadratic subproblems, which are linearized around the current state. Then the algorithm does a line search to determine the step size in the direction of the solution for each node in the mesh. The algorithm iterates through the updated states until the solutions to the quadratic subproblems converge into a smooth gradient. To perform this analysis, the following assumptions were made:

1. Constraints are continuously twice differentiable

2. The objective function is continuously twice differentiable
3. There is a 'good' initial guess
4. There is a 'good' mesh

The methodology and assumptions impose the following limitations on this analysis method:

1. Due to the derivative assumptions, smooth objective and model functions are required
2. Due to ending conditions, a smooth mesh is required
   - No discontinuities can be in the mesh
   - The mesh must be fine enough to capture the gradient
3. Due to the gradient-driven update method, the solution may not recover from a step outside the constraints
   - Initial guess should start solver inside constraints (algorithm may or may not work otherwise)
   - A 'bad' step can put algorithm outside constraints
4. The algorithm won't know if the minimum it found is a local or global minimum
5. If the objective function behavior cannot be adequately captured in a localized linear model, this can create 'bad' steps due to deviation between actual and linearized models

The complex step method was used to generate the gradients for fmincon for the nonlinear constraint and the objective functions using equation 6, and the nonlinear stress constraint given by equation 7.

$$\frac{df}{dx} \approx \frac{imag(f(x+hi)))}{h}, h = 1e-30 \tag{6}$$

$$c(i) = \frac{\sigma}{\sigma_{ultimate}} - 1 \leq 0 \tag{7}$$

4 Optimization Problem

The objective of this lab is to apply the principles learned in lecture to optimize the shape of a wing spar modeled within the following criteria and constraints[2]:

1. The design is to be two-dimensional
2. Has a width of L = 7.5m
3. The spar should survive under 2.5gs of acceleration ($F = 2.5 * \frac{m}{2} * 9.81$)

The objective of this project is to optimize the shape of the wing spar to minimize the weight. The model geometry was parameterized and discretized (see section 2 for more details) and applied to a discretized version of the Euler-Bernoulli Beam Theory to analyze the problem. The method chosen for the minimization problem was the SQP algorithm for the fmincon Matlab function (see section 3 for more details) with the gradients calculated using the complex step method.

5 Results

The convergence plot shown in Figure 1 was calculated for the stress at the spar tip with an initial shape function r(x). The plot shows an exponential decrease towards 0 as the number of segments increases. This behavior makes sense because as the number of variables increases, the error from the volume and discretization calculations decreases. The acceptable number of elements was calculated as 50 as it represented a 6.5 orders of magnitude decrease from the maximum normalized stress while also not resulting in a long run time, which is similar to the conditions for the first-order optimality of the iteration solutions.

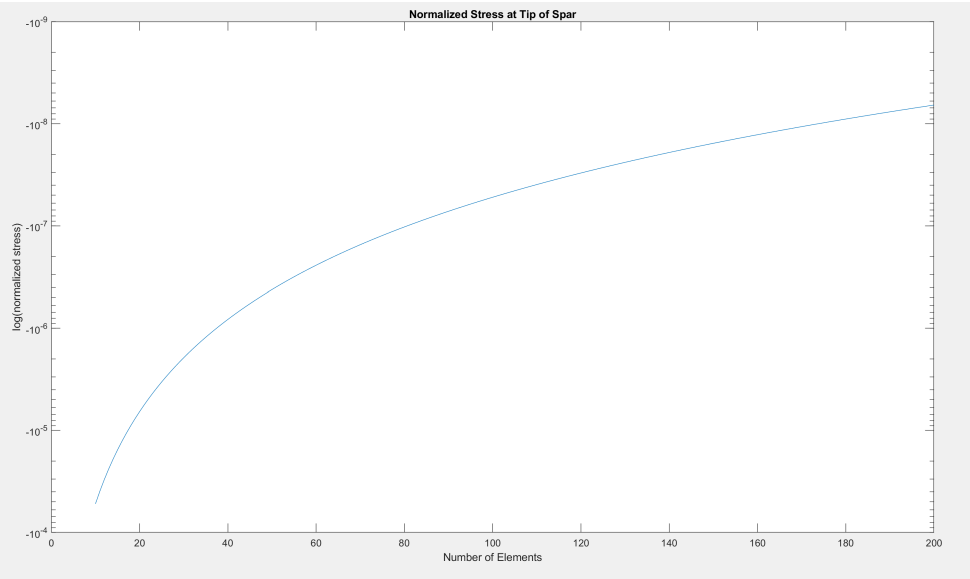

Figure 1: Plot of the stress vs N

The optimized shape of the wing spar with the chosen parameterization and number of elements (50) is shown in Figure 2. This result makes sense because it shows a similar inner and outer radii to the provided nominal design which gets thinner before the outer radii decrease until the design hits the lower bound of the design space, which would result in a limited mass of the spar

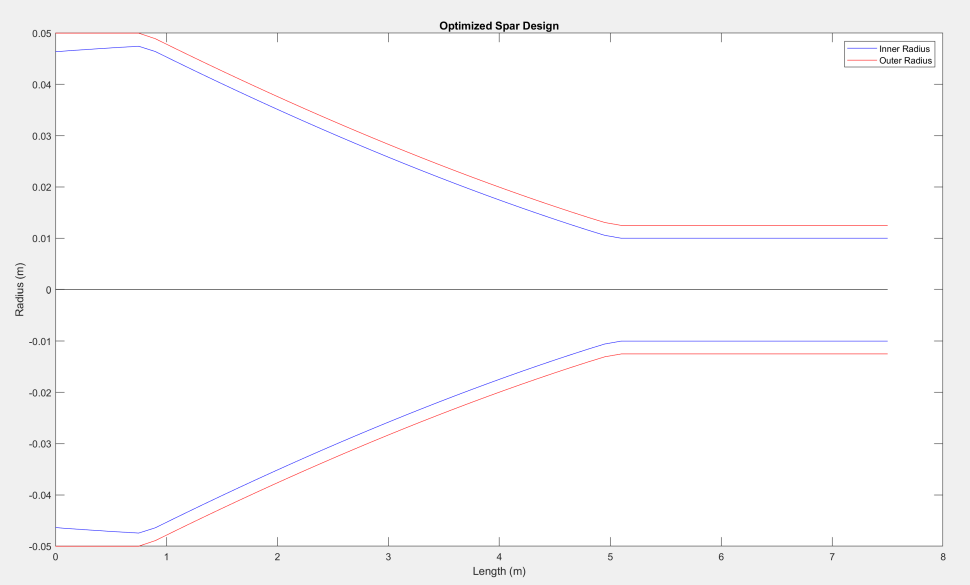

Figure 2: Plot of the Wing Spar Cross Section

The feasibility graph shows that the feasibility score of the current solution decreased rapidly over the first iteration and then stabilized over the remaining iterations as the First-Order Optimality improved. The First-Order Optimality graph shows a slow improvement over the iterations until the solver converged on a minimum causing a rapid improvement (see Figure 3 both Feasibility and First Order Optimality decreased by sixteen orders of magnitude over the iterations). (Note: the feasibility score for both the general and complex step method spends most of the optimization at 0, so the semilog plot didn't work for them see 4 for the semi-log plot of the first order optimization).

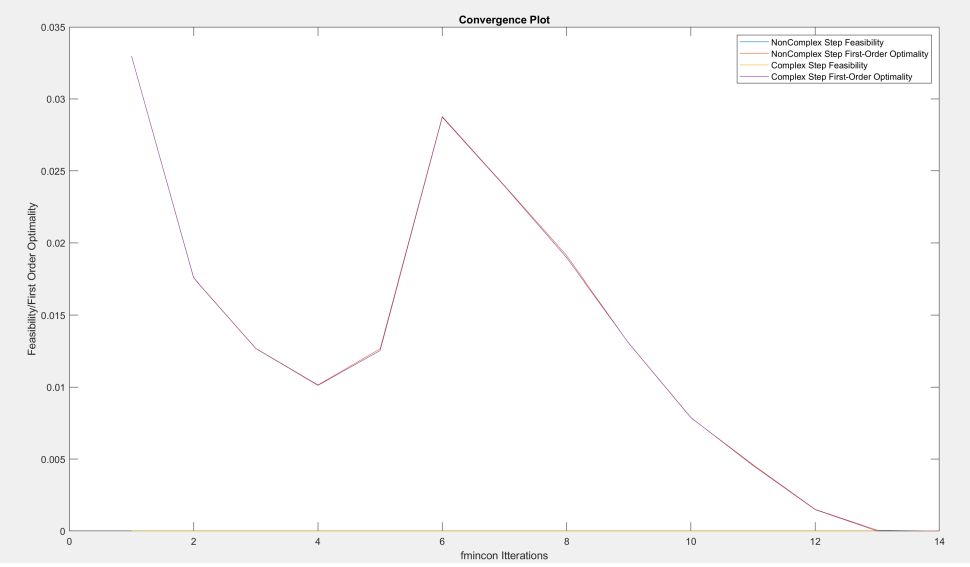

Figure 3: The Feasibility and Convergence Plots

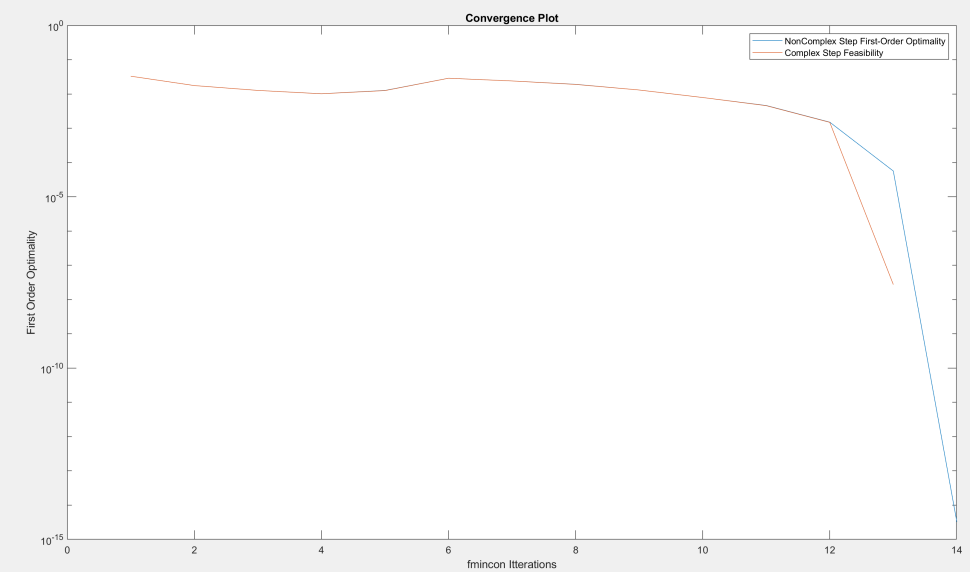

Figure 4: The Convergence Plots

6 Conclusions

The implementation of the SQP optimization algorithm with the complex step gradient method for this lab was successful. The resulting shape function for the optimized wing spar section profile made sense because it minimized the mass by minimizing the thickness and outer radius of the spar. The resulting calculated mass from the optimized wing spar was less than forty percent of the baseline design ($.4 * 13.26 = 5.3020 > 5.0707$ kg) while remaining within the problem constraints. Complex step and the normal fmincon optimizations resulted in the same optimized design with the complex step method running noticeably faster, especially when a higher number of elements was used.

Basic optimization algorithms, such as the ones used in this lab, can be powerful tools for understanding the complex interactions between different design variables in engineering design problems and balancing the design variables to arrive at an optimal solution. Understanding how these algorithms work and the trade-offs between accuracy and run time shown in the convergence study is important when utilizing these algorithms to solve problems.

References

[1] *Fmincon*, MathWorks Help Website, 2023.

[2] D. J. Hicken, *Mane 4280/6710: Numerical design optimization lab 2*, MANE 6710 Course LMS Website, October, 2024.

7 Appendix

7.1 getq.m

```
function [q]=getq(N, F,qmin,L)
% Get array that describes the distributed load (assumes linear
% distribution
% Inputs:
%   L - length of the beam
%   N - number of elements
%   F - total  force of the
%   qmin - the minimum value of the distributed load
% Outputs:
%   q- the distributed load array
%   ------------------------------
%   calculate force in the triangular part of the distribution
    f=F-qmin*L;
    %calculate the max value of the distributed load
    qmax=2*f/L+qmin;
    %create the array of values for the distributed load
    q=linspace(qmax,qmin,N+1);
end
```

7.2 run.m

```
%initialize the design constants
L=7.5;
m=500/2;
N=50;
rho=1600;
E=70000000000;
sigmaUlt=600000000;

R=putR(.015*ones([N+1,1]),.035*ones([N+1,1]));
%[rin,rout]=getR(R);
%initialize function
vol=@(R) calcVol(R,L,N);
%initialize c nonlcon
chat=@(R) getC(R,L,m,N,E,sigmaUlt);

%set fmincon options
options = optimoptions('fmincon','Display','iter','Algorithm','sqp','
    SpecifyConstraintGradient',true,'SpecifyObjectiveGradient',true);

%initialize lb,ub, a,b

lb=putR(.01*ones([N+1,1]),.0125*ones([N+1,1]));
ub=putR(.0475*ones([N+1,1]),.05*ones([N+1,1]));
bineq=-.0025*ones([N+1,1]);
Aineq=zeros([N+1,2*(N+1)]);
for i=1:length(bineq)
    Aineq(i,2*i-1)=1;
    Aineq(i,2*i)=-1;
end
%run optimization function
a=fmincon(vol,R,Aineq,bineq,[],[],lb,ub,chat,options);
%plot output design function
[ain,aout]=getR(a);
x=0:L/N:L;
plot(x,ain,'b')
hold on
plot(x,aout,'r')
plot(x,-1*aout,'r')
plot(x,-1*ain,'b')
plot(x,zeros(size(x)),"-k")
xlabel("Length (m)")
ylabel("Radius (m)")
legend('Inner Radius','Outer Radius')
title("Optimized Spar Design")
hold off
```

7.3 getR.m

```
function [rin,rout]=getR(R)
%Splits a design variable into rin and rout
% Inputs:
%   R- the design variable array
% Outputs:
%   rin - the array of inner radii
%   rout - the array of outer radii
%   ------------------------------
%
l=max(size(R))/2;
    rout=zeros([l,1]);
    rin=zeros([l,1]);
    for i=1:l
        rin(i)=R(2*i-1);
        rout(i)=R(2*i);
    end
end
```

7.4 getI.m

```
function [Iyy]=getI(R)
    %Calculates Iyy for each element in the model
% Inputs:
%   R- the design variable array
% Outputs:
%   Iyy - the array of the second moment of inertia
%   ------------------------------
%get rin, rout from design variable array
    [rin,rout]=getR(R);
    Iyy=zeros(length(rin));
%calculate Iyy for each element
    for i=1:length(rin)
        Iyy(i)=pi*(rout(i)^4-rin(i)^4)/4;
    end
end
```

7.5 getC.m

```
function [c,ceq,j,jeq]=getC(R,L,m,N,E,sigmaUlt)
% calculate the nonlinear inequality constraint and the nonlinear
%   inequality constraint gradient for the spar
% Inputs:
%   R– the design variable array
%   L– the length of the spar
%   N– the number of elements
%   m–the mass of the plane
%   E– he young's modulus of the spar material
%   sigmaUlt – the ultimate tensile strength of the material
% Outputs:
%   c – the normalized nonlinear inequality constraint
%   j – the jacobian for the nonlinear inequality constraint calculated
%       using the complex step method
%   ----------------------------------
    function [frac]=subs(r)
        % calculate the nonlinear inequality constraint
% Inputs:
%   r– the design variable array
% Outputs:
%   c – the normalized nonlinear inequality constraint
%   ----------------------------------
    %get the distributed load array
    q=getq(N, 2.5*m*9.81,0,L);
    %get rin, rout
    [rin,rout]=getR(r);
    %get the second moment of Inertia array
    Iyy=getI(r);
    %get the beam displacements
    [u] = CalcBeamDisplacement(L, E, Iyy, q, N);
    %get element stresses
    [sigma] = CalcBeamStress(L, E, rout, u, N);
    %create c
    frac=sigma/sigmaUlt;

    end
    %set up complex step
    h=1e-30;
    g=zeros([2*(N+1),N+1]);
    %calculate jacobian
    for i=1:2*(N+1)
    comp=R;
    comp(i)=R(i)+complex(0,h);
    g(i,:)= imag(subs(comp))/h;
    end
    frac=subs(R);
    %calculate outputs
    c=frac-ones(size(frac));
```

```
49          ceq = [];
50          jeq = [];
51          j=g;
52
53
54  end
```

7.6 putR.m

```
function [R]=putR(rin, rout)
%Integrate rin and rout into a R array
% Inputs:
%   rin - the array of inner radii
%   rout - the array of outer radii
% Outputs:
%   R- the design variable array
%   ------------------------------
    R=zeros([length(rin)*2,1]);
    for i=1:length(rin)
        R(2*i-1)=rin(i);
        R(2*i)=rout(i);
    end
end
```

7.7 calcVol.m

```
function [vol,g]= calcVol(R,L,N)
% calculate the volume and the volume gradient for the spar
% Inputs:
%   R- the design variable array
%   L- the length of the spar
%   N- the number of elements
% Outputs:
%   vol - the volume of the spar
%   g - the gradient fo the volume calculated using the complex step
    method
%   ----------------------------

    function [vol]=subs(r)
        %calculate the volume and the volume gradient for the spar
% Inputs:
%   r- the design variable array
% Outputs:
%   vol - the volume of the spar
%   ----------------------------
    [rin,rout]=getR(r);
    vol=0;
    for i=1:length(rin)
        vol=vol+(pi*rout(i)^2-pi*rin(i)^2)*L/(N);
    end
    end
 %set up complex step
    h=1e-30;
    g=zeros(size(R));
    %compute gradient
    for j=1:length(g)
        comp=R;
    comp(j)=R(j)+complex(0,h);
    g(j)=imag(subs(comp))/h;
    end
%compute volume
vol=subs(R);
%g=imag(subs(comp))/h;
end
```

7.8 plots.m

```
feasoptinocomstep=[      0.000e+00   ,        3.299e-02 ;     0.000e+00      ,
        1.759e-02   ;      0.000e+00    ,       1.269e-02   ;     0.000e+00    ,
         1.015e-02   ; 0.000e+00      ,    1.266e-02   ; 0.000e+00     ,
    2.878e-02   ;     0.000e+00       ,  2.398e-02 ;      0.000e+00      ,
    1.913e-02 ;     0.000e+00      ,    1.304e-02   ;    0.000e+00     ,
    7.870e-03   ;       0.000e+00,          4.605e-03 ; 0.000e+00      ,
    1.504e-03   ; 3.650e-07    ,   5.631e-05   ;    4.770e-18,         3.239e-15
     ];
comstep=[  0.000e+00      ,      3.299e-02   ;        0.000e+00     ,      1.757
    e-02   ;       0.000e+00      ,      1.268e-02   ;       0.000e+00     ,
    1.012e-02   ;      0.000e+00      ,     1.254e-02   ;      0.000e+00      ,
        2.873e-02   ;       0.000e+00      ,     2.395e-02   ;          0.000e
    +00     ,     1.898e-02   ;         0.000e+00     ,    1.305e-02   ;      0.000
    e+00     ,     7.883e-03   ;      0.000e+00    ,      4.550e-03   ;     0.000e
    +00     ,     1.485e-03   ;      3.843e-11      ,     2.745e-08   ];
close all


%plot  feasibility graph
%plot(1:14,feasoptinocomstep(:,1))

semilogy(1:14,feasoptinocomstep(:,2))
hold on
%plot(1:13,comstep(:,1))
semilogy(1:13,comstep(:,2))
xlabel("fmincon Itterations")
title("Convergence Plot")
ylabel("First Order Optimality")
legend( "NonComplex Step First-Order Optimality","Complex Step Feasibility
    ", "Complex Step First-Order Optimality")
hold off

%%
% calculate convergence study

%initiialize the design constants
L=7.5;
m=500/2;
%N=50;
rho=1600;
E=70000000000;
sigmaUlt=600000000;
start=10;
stop=200;

x=start:stop;
q=zeros(size(x));
z=1;
```

```
for i=start:stop
    R=putR(.015*ones([i+1,1]),.05*ones([i+1,1]));
    [c,j,a,b]=getC(R,L,m,i,E,sigmaUlt);
    q(z)=c(i+1)+1;
    z=z+1;
end

%%
%plot convergence study

semilogy(x,q)
xlabel("Number of Elements")
ylabel("log(normalized stress)")
title("Normalized Stress at Tip of Spar")
```

7.9 fmincon Output

```
>> run
 Iter  Func-count           Fval   Feasibility   Step Length       Norm of    First-order
                                                                      step     optimality
    0           1    2.403318e-02     0.000e+00     1.000e+00     0.000e+00     3.299e-02
    1           5    1.190725e-02     0.000e+00     4.900e-01     6.014e-02     1.757e-02
    2          11    9.481606e-03     0.000e+00     2.401e-01     1.930e-02     1.268e-02
    3          18    8.212558e-03     0.000e+00     1.681e-01     1.310e-02     1.012e-02
    4          23    7.400609e-03     0.000e+00     4.900e-01     2.751e-02     1.254e-02
    5          27    4.984801e-03     0.000e+00     4.900e-01     2.549e-02     2.873e-02
    6          32    4.372973e-03     0.000e+00     4.900e-01     1.628e-02     2.395e-02
    7          37    3.870200e-03     0.000e+00     4.900e-01     1.953e-02     1.898e-02
    8          42    3.546220e-03     0.000e+00     4.900e-01     1.558e-02     1.305e-02
    9          46    3.287804e-03     0.000e+00     7.000e-01     1.537e-02     7.883e-03
   10          50    3.204750e-03     0.000e+00     7.000e-01     5.744e-03     4.550e-03
   11          54    3.179851e-03     0.000e+00     7.000e-01     1.802e-03     1.485e-03
   12          57    3.169193e-03     3.804e-07     1.000e+00     7.844e-04     5.760e-05
   13          59    3.169193e-03     3.843e-11     1.000e+00     1.682e-08     2.745e-08

Local minimum found that satisfies the constraints.

Optimization completed because the objective function is non-decreasing in
feasible directions, to within the value of the optimality tolerance,
and constraints are satisfied to within the value of the constraint tolerance.

<stopping criteria details>
```

Figure 5: The Outpt of fmincon