MANE 6710 - Numerical Design Optimization Lab 4

Human 6966

December 10 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. To handle uncertainty in the loading of the wing, the first four harmonic modes were added to the loading condition and the design space was limited to designs at least six standard deviations stronger than the mean loading. 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 (see lab two for more information [1]). The first four modes of loading perturbation are given by equation 1.

$$\delta_f(x, \zeta_n) = \zeta_n \cos(\frac{(2n-1)\pi x}{2L}) \tag{1}$$

with

- ζ_n being a normally distributed random variable $N(0, \frac{f_{nom(0)}}{10n})$
- n being the mode number (1, 2, 3, or 4)
- x being the position along the spar
- L being the length of the spar
- $f_{nom}(0)$ being the nominal loading at the base of the wind spar

These perturbation modes were added to the loading condition used in lab two when calculating the stresses at each point. Then the mean and standard deviation of the stresses were calculated at each node with Gauss-quadrature using equations 2 and 3

$$\mu_i = \sum_{i1=1}^{m}\sum_{i2=1}^{m}\sum_{i3=1}^{m}\sum_{i4=1}^{m} w_{i1}w_{i2}w_{i3}w_{i4}\eta(q(x_{i1}, x_{i2}, x_{i3}, x_{i4}), R)_i \tag{2}$$

$$\sigma_i = \sqrt{\eta_i^2 - \mu_i^2} \tag{3}$$

with

- η_i being the stress function at node i
- μ_i being the mean stress at node i
- w_{iy} being the adjusted Gauss quadrature weight for the yth perturbation mode
- $w_{iy} = \frac{w_x}{\sqrt{\pi}}$

- w_x being the standard Gauss quadrature weight [2]
- x_{iy} being the adjusted Gauss quadrature point for the yth perturbation mode
- $x_{iy} = \sqrt{2} x_z \frac{f_{nom}(0)}{10n}$
- x_z being the standard Gauss quadrature point [2]
- σ_i being the standard deviation of the stress at node i

1.2 Assumptions

To simplify the physical model and fundamental equations, the following assumptions were made in addition to the ones in lab two [1]:

1. The loading perturbation can be represented as a set of harmonic modes
2. The perturbation modes are independent random variables
3. The perturbation modes are normal random variables

1.3 Limitations

These assumptions limit this analysis method to cases where the loading perturbations can be represented as a Fourier series or integral (which has limited effect as many aerodynamic phenomena can be represented this way). Additionally, it may be experimentally determined that in some cases the harmonic loading perturbation modes aren't normally distributed. For these cases, a new mean and standard deviation model would have to be made as these are only valid for normally distributed random variables. Finally if the perturbation modes are proven to not be independent, a new model for the mean and standard deviation would be required to account for the interaction between random variables.

2 Parameterization of the Geometry

The geometry was parameterized the same way as in lab two [1]. A finite element mesh of N+1 nodes was used to represent the geometry with the stress calculated at each node.

3 Optimization Method and Limitations

The optimization method from lab two is the same as in this lab except for the nonlinear constraint given by equation 4. This means that all of the previous assumptions and limitations hold for this lab (Note that η in this lab is equivalent to σ in the previous lab).

$$c(i) = \frac{\mu + 6\sigma}{\eta_{ultimate}} - 1 \leq 0 \tag{4}$$

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 in addition to the ones presented in lab two [1][3][4]:

1. $\eta_{ultimate} = 600$ MPa

2. $\mu + 6\sigma \leq \eta_{ultimate}$

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. Gauss quadrature used to calculate the mean and standard deviation of the stress for the nonlinear constraint. The method chosen for the minimization problem was the SQP algorithm for the fmincon Matlab function (see lab two section 3 for more details [5]) with the gradients calculated using the complex step method.

5 Results

The convergence plot shown in Figure 1 was calculated by taking the norm of the error of $\mu + 6\sigma$ at each node using 1, 2, 3, 4, 5, and 6 Gauss quadrature points from the result of using six Gauss quadrature points. The plot shows a significant decrease in error between the first and second set of Gauss quadrature points relative to the six-point solution before remaining relatively flat near zero. This behavior makes sense as it shows the function is closely approximated by a third-order polynomial and thus can be approximated well with a higher-order polynomial but not a lower-order one. The acceptable number of Gauss quadrature points was determined to be two as it represented little error while also not resulting in a long run time.

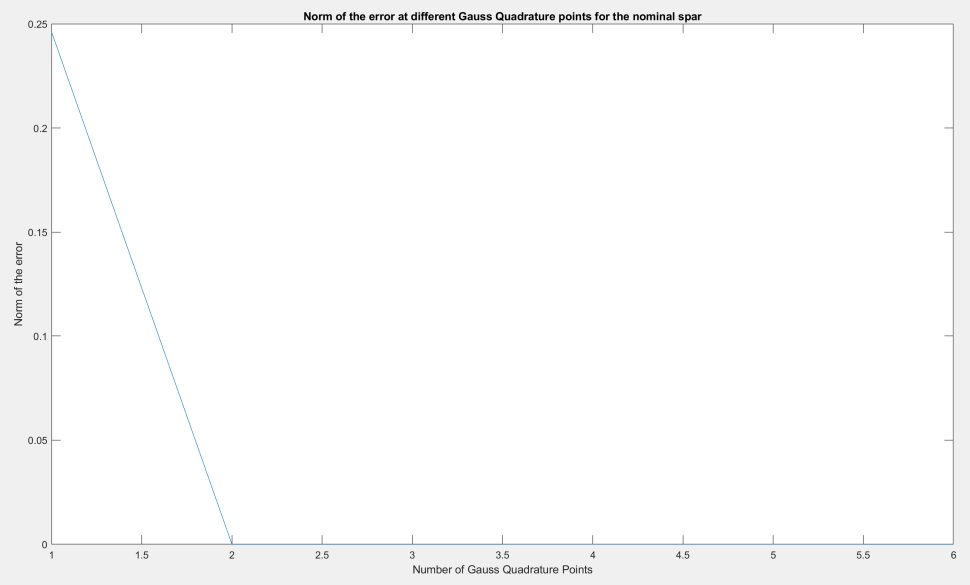

Figure 1: Plot of the stress error vs Number of Gauss Quadrature points

Once the convergence study was run, a graph of the mean and standard deviation for the nominal design was created to verify the analysis method. This was compared to a provided graph on Piazza, which was similar enough to conclude the analysis was implemented correctly.

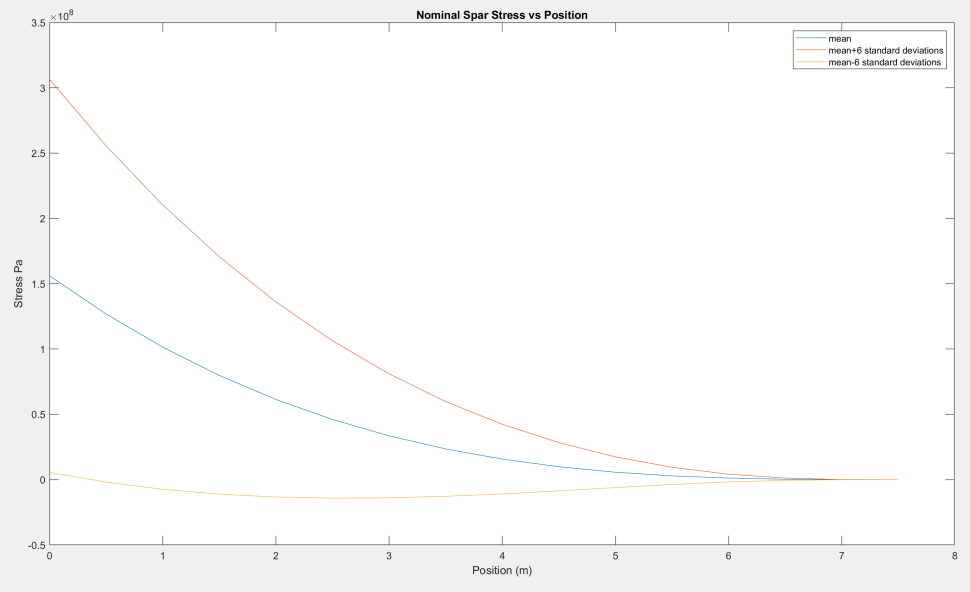

Figure 2: Stress vs Position in the Nominal Spar

The optimized shape of the wing spar with the chosen parameterization, number of elements (80)[1], and number of Gauss quadrature points (2) is shown in Figure 3. 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. This is similar to the result from lab two shown in Figure 4, except starting with a thicker initial portion at the root. This causes the rest of the design to be shifted towards the spar tip, which makes sense. This is because by adding uncertainty to the spar loading and designing for six standard deviations above the mean instead of designing for the mean, the effective applied force is increased resulting in a thicker spar with the same behavior.

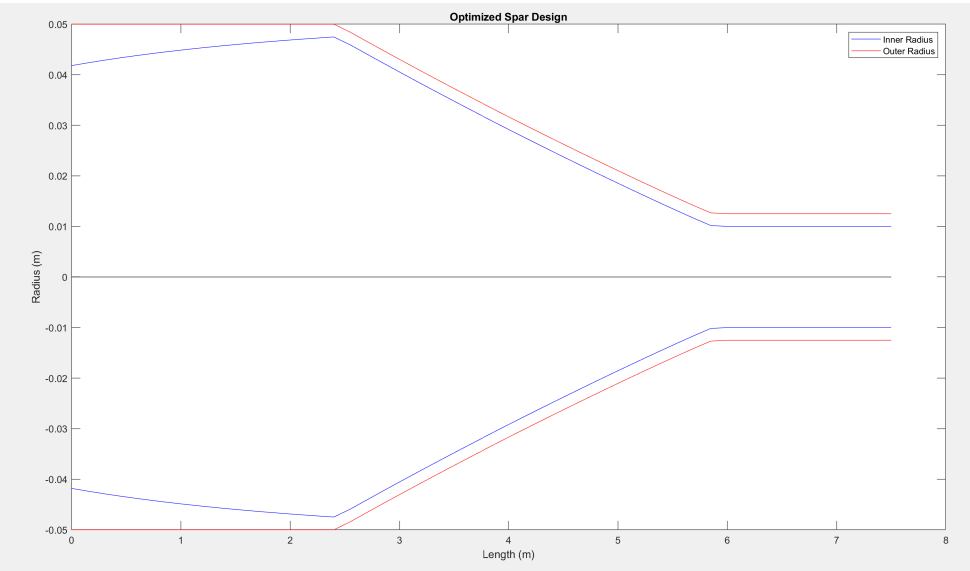

Figure 3: Plot of the Wing Spar Cross Section

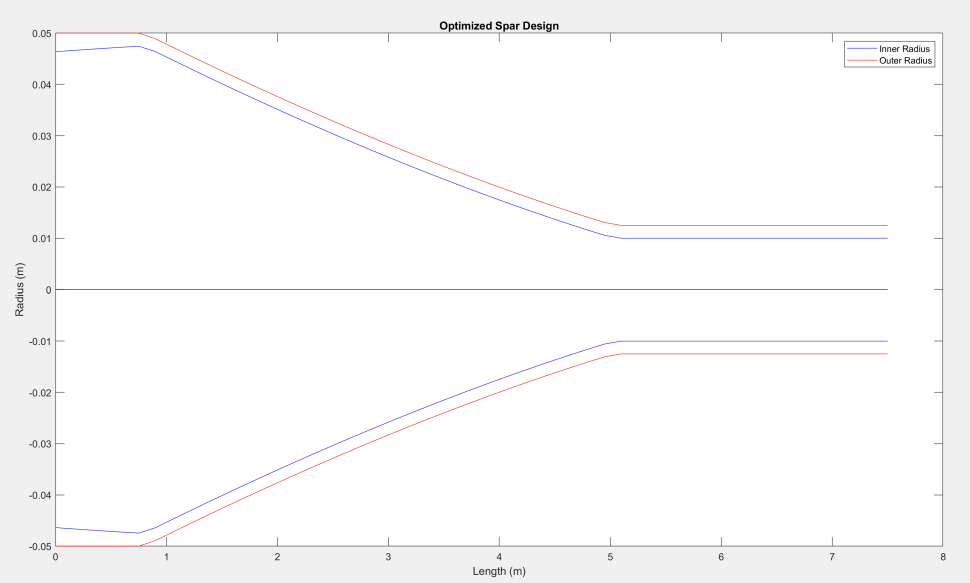

Figure 4: Plot of lab two's Wing Spar Cross Section

The convergence plot shows that the First-Order Optimality score of the current solution decreased gradually over the first 12 iterations before rapidly decreasing as the optimization algorithm closed in on the solution (see Figure 5 the First Order Optimality decreased by six orders of magnitude over the iterations). (Note: the feasibility score for the optimization is 0 at every point, so the semilog plot didn't work for it). Also included is a of the mean and standard deviations of the stress at each node for the optimal design (figure 6) (Note: the mean+6 standard deviations line behaves like the stress curve from lab 2 [1]).

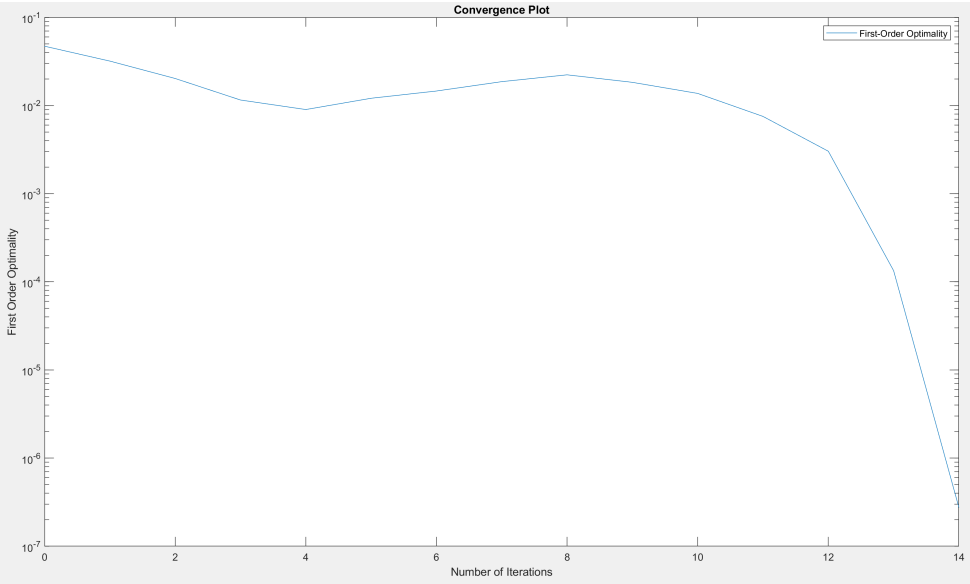

Figure 5: First order Optimality Convergence Plot

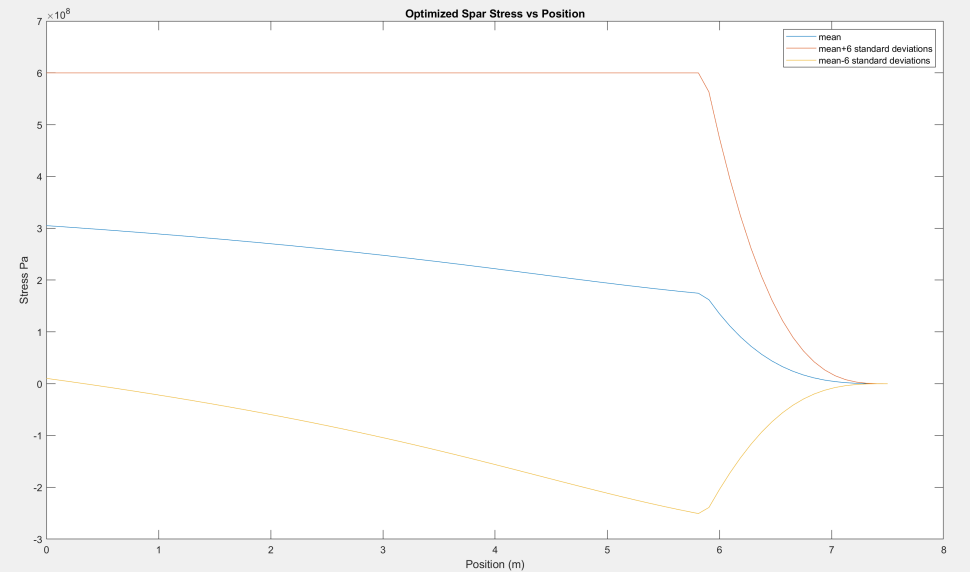

Figure 6: Stress vs Position in the Optimized Spar

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 thirty percent of the baseline design ($0.3 * 29.32 = 8.796 > 8.784$ kg) while remaining within the problem constraints.

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. However, many engineering problems display uncertainty caused by noise in measured data. This lab demonstrated effective methods for handling these uncertainties while using optimization algorithms to prevent the creation of designs that only work in theory due to their lack of robustness.

References

[1] H. 6966, *Mane 6710: Numerical design optimization lab 2*, MANE 6710 Course LMS Website, October, 2024.

[2] *Abscissas and weights of gauss-hermite integration*, Website, 2024. [Online]. Available: `https://www.efunda.com/math/num%5C_integration/findgausshermite.cfm`..

[3] D. J. Hicken, *Mane 4280/6710: Numerical design optimization lab 4*, MANE 6710 Course LMS Website, November, 2024.

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

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

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=80;
rho=1600;
E=70000000000;
sigmaUlt=600000000;

R=putR(.01*ones([N+1,1]),.05*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,2);

%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,ngq)
% 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
%   ---------------------------------
    function [bob]=stressX(q,x,N,L,r,E)
    % calculate the nonlinear inequality constraint
% Inputs:
%   r- the design variable array
%   L- the length of the spar
%   N- the number of elements
%   q-the nominal loding condition
%   E- he young's modulus of the spar material
%   x- the position along the spar
% Outputs:
%   bob - the uncertain spar loading
%   ---------------------------------
    for i=1:N+1
        q(i)=q(i)+x(1)*cos(pi*(i-1)/2/(N))+x(2)*cos(3*pi*(i-1)/2/(N))+x(3)
            *cos(5*pi*(i-1)/2/(N))+x(4)*cos(7*pi*(i-1)/2/(N));
    end
    %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
    [bob] = CalcBeamStress(L, E, rout, u, N);
    bob;
end
```

```
48      %get the distributed load array
49      q=getq(N, 2.5*m*9.81,0,L);
50      %gauss quadrature points
51      sigf=[q(1)/10,q(1)/20,q(1)/30,q(1)/40];
52      if ngq == 1
53          xi = [0.0];
54          wts = [1.77245]./sqrt(pi);% adjusted weights !
55      elseif ngq ==2
56          xi = [-0.707107; 0.707107];
57          wts = [0.886227; 0.886227]./sqrt(pi);
58      elseif ngq == 3
59          xi = [ -1.22474487139; 0.0; 1.22474487139];
60          wts = [0.295408975151; 1.1816359006; 0.295408975151]./ sqrt(pi);
61      elseif ngq == 4
62          xi = [-1.65068; -0.524648; 0.524648; 1.65068];
63          wts = [0.0813128; 0.804914; 0.804914; 0.0813128]./sqrt(pi);
64      elseif ngq == 5
65          xi = [-2.02018; -0.958572; 0.0; 0.958572; 2.02018];
66          wts = [0.0199532; 0.393619; 0.945309; 0.393619; 0.0199532]./sqrt(
                pi);
67      elseif ngq == 6
68          xi = [-2.35061; -1.33585; -0.436077; 0.436077; 1.33585; 2.35061];
69          wts = [0.00453001; 0.157067; 0.72463; 0.72463; 0.157067;
                0.00453001]./sqrt(pi);
70      end
71      mean=0;
72      mean2=0;
73      %calculate mean, mean squared using gauss quadrature
74      for i1=1:ngq
75          p1=sqrt(2)*sigf(1)*xi(i1);
76          for i2=1:ngq
77              p2=sqrt(2)*sigf(2)*xi(i2);
78              for i3=1:ngq
79                  p3=sqrt(2)*sigf(3)*xi(i3);
80                  for i4=1:ngq
81                      p4=sqrt(2)*sigf(4)*xi(i4);
82                      temp=stressX(q,[p1,p2,p3,p4],N,L,r,E);
83                      mean=mean+wts(i1)*wts(i2)*wts(i3)*wts(i4)*temp;
84                      mean2=mean2+wts(i1)*wts(i2)*wts(i3)*wts(i4)*temp.*temp
                            ;
85                  end
86              end
87          end
88      end
89      %calculate standard deviation
90      sigma=mean+6*sqrt((mean2-mean.*mean));
91      %create c
92      %plot mean +/- 6stddev
93     % x=0:L/N:L;
94      %plot(x,mean)
```

```
95      %hold on
96      %plot(x,mean+6*sqrt((mean2-mean.*mean)))
97      %plot(x,mean-6*sqrt((mean2-mean.*mean)))
98      %xlabel("Position (m)")
99      %ylabel("Stress Pa")
100     %legend(["mean","mean+6 standard deviations","mean-6 standard
            deviations"])
101     %title("Nominal Spar Stress vs Position")
102     %hold off
103     frac=sigma/sigmaUlt;
104     end
105     %set up complex step
106     h=1e-30;
107     g=zeros([2*(N+1),N+1]);
108     %calculate jacobian
109     for ir=1:2*(N+1)
110     comp=R;
111     comp(ir)=R(ir)+complex(0,h);
112     g(ir,:)= imag(subs(comp))/h;
113     end
114     frac=subs(R);
115     %calculate outputs
116     c=frac-ones(size(frac));
117     ceq=[];
118     jeq=[];
119     j=g;
120     %run convergence history
121     %frac=[[],[],[],[],[],[],[]];
122     %for asd=1:6
123      %    ngq=asd;
124      %    frac(asd,:)=subs(R);
125     %end
126     %calculate error
127     %error=[0,0,0,0,0,0];
128     %for asd=1:6
129     %     error(asd)=norm(frac(asd)-frac(6));
130     %end
131     %plot errors
132     %plot(1:6,error)
133     %xlabel("Number of Gauss Quadrature Points")
134     %ylabel("Norm of the error")
135     %title("Norm of the error at different Gauss Quadrature points for the
            nominal spar")
136 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 , 4.712e-02 ; 0.000e+00 , 3.193e-02 ; 0.000e+00 , 2.027e-02 ; 0.000e+00 , 1.155e-02 ; 0.000e+00 , 9.005e-03 ; 0.000e+00 , 1.209e-02 ; 0.000e+00 , 1.461e-02 ; 0.000e+00 , 1.866e-02 ; 0.000e+00 , 2.231e-02 ; 0.000e+00 , 1.833e-02 ; 0.000e+00 , 1.372e-02 ; 0.000e+00 , 7.524e-03 ; 0.000e+00 , 3.030e-03 ; 1.532e-06 , 1.337e-04 ; 6.503e-11 , 2.741e-07 ];

comstep=[ 0.000e+00 , 3.299e-02 ; 0.000e+00 , 1.757e-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 ; 0.000e+00 , 7.883e-03 ; 0.000e+00 , 4.550e-03 ; 0.000e+00 , 1.485e-03 ; 3.843e-11 , 2.745e-08 ];
%

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

semilogy(0:14,feasoptinocomstep(:,2))
hold on
%plot(1:13,comstep(:,1))
%semilogy(1:13,comstep(:,2))
xlabel("fmincon Itterations")
title("Convergence Plot")
ylabel("First Order Optimality")
xlabel("Number of Iterations")
legend( "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

Iter	Func-count	Fval	Feasibility	Step Length	Norm of step	First-order optimality
0	1	5.725553e-02	0.000e+00	1.000e+00	0.000e+00	2.945e-02
1	5	2.711195e-02	1.553e-02	4.900e-01	1.288e-01	2.096e-02
2	10	1.912566e-02	1.873e-02	3.430e-01	4.927e-02	1.404e-02
3	19	1.783243e-02	1.909e-02	8.235e-02	1.089e-02	1.262e-02
4	29	1.696060e-02	1.933e-02	5.765e-02	8.104e-03	1.179e-02
5	40	1.639642e-02	1.922e-02	4.035e-02	5.415e-03	1.126e-02
6	53	1.613462e-02	1.900e-02	1.977e-02	2.573e-03	1.101e-02
7	66	1.587947e-02	1.879e-02	1.977e-02	2.537e-03	1.078e-02
8	79	1.563083e-02	1.858e-02	1.977e-02	2.501e-03	1.055e-02
9	92	1.538848e-02	1.838e-02	1.977e-02	2.468e-03	1.033e-02
10	105	1.515220e-02	1.818e-02	1.977e-02	2.436e-03	1.011e-02
11	118	1.492188e-02	1.799e-02	1.977e-02	2.404e-03	9.892e-03
12	131	1.469730e-02	1.780e-02	1.977e-02	2.375e-03	9.678e-03
13	144	1.447827e-02	1.762e-02	1.977e-02	2.347e-03	9.465e-03
14	157	1.426469e-02	1.744e-02	1.977e-02	2.319e-03	9.255e-03
15	170	1.405637e-02	1.726e-02	1.977e-02	2.293e-03	9.047e-03
16	183	1.385313e-02	1.709e-02	1.977e-02	2.269e-03	8.841e-03
17	196	1.365486e-02	1.692e-02	1.977e-02	2.245e-03	8.636e-03
18	209	1.346141e-02	1.675e-02	1.977e-02	2.223e-03	8.433e-03
19	222	1.327262e-02	1.659e-02	1.977e-02	2.202e-03	8.231e-03
20	235	1.308834e-02	1.643e-02	1.977e-02	2.182e-03	8.029e-03
21	248	1.290849e-02	1.628e-02	1.977e-02	2.163e-03	7.828e-03
22	261	1.273292e-02	1.612e-02	1.977e-02	2.145e-03	7.627e-03
23	274	1.256149e-02	1.597e-02	1.977e-02	2.128e-03	7.426e-03
24	287	1.239409e-02	1.583e-02	1.977e-02	2.112e-03	7.225e-03
25	300	1.223063e-02	1.568e-02	1.977e-02	2.097e-03	7.025e-03
26	313	1.207098e-02	1.554e-02	1.977e-02	2.082e-03	6.860e-03
27	326	1.191503e-02	1.540e-02	1.977e-02	2.069e-03	6.697e-03
28	339	1.176266e-02	1.526e-02	1.977e-02	2.057e-03	6.536e-03
29	352	1.161380e-02	1.512e-02	1.977e-02	2.045e-03	6.376e-03
Iter	Func-count	Fval	Feasibility	Step Length	Norm of step	First-order optimality
30	365	1.146844e-02	1.499e-02	1.977e-02	2.031e-03	6.219e-03
31	369	1.583878e-02	3.557e-04	7.000e-01	9.998e-02	2.509e-02
32	373	1.005952e-02	1.814e-04	4.900e-01	4.818e-02	2.030e-02
33	378	8.001468e-03	1.192e-04	3.430e-01	2.218e-02	1.017e-02
34	383	7.000361e-03	6.079e-05	4.900e-01	1.687e-02	6.461e-03
35	387	6.242388e-03	2.771e-05	7.000e-01	1.544e-02	1.444e-02
36	392	5.938455e-03	4.067e-05	4.900e-01	6.460e-03	1.068e-02
37	396	5.665623e-03	4.767e-05	7.000e-01	8.508e-03	5.859e-03
38	400	5.563878e-03	5.412e-06	7.000e-01	5.436e-03	5.610e-03
39	404	5.524566e-03	4.261e-05	7.000e-01	4.331e-03	4.655e-03
40	408	5.506057e-03	1.564e-05	7.000e-01	4.200e-03	4.138e-03
41	411	5.496491e-03	7.722e-06	1.000e+00	2.859e-03	1.175e-03
42	413	5.496435e-03	3.249e-07	1.000e+00	1.883e-06	2.656e-05
43	415	5.496435e-03	8.336e-12	1.000e+00	1.029e-09	2.704e-09

Figure 7: The Outpt of fmincon