Unique ID: 1485

Project #2 Report:
Design Optimization of an Unmanned Aerial Vehicle's Wing Spar

Executive Summary

The widespread applicability of unmanned aerial vehicles (UAVs) in the aerospace industry has the capability to continue to revolutionize science and technology in the near future. One objective of these aircraft that is currently being developed, through programs such as Zephyr from Airbus, is enabling UAVs to serve as long-endurance telecommunications platforms [1]. Similar to Zephyr, the UAV system of interest for this analysis seeks to fulfill a need to connect remote parts of the world through a novel concept. To maximize the technological capabilities of the telecommunications system by including more advanced components, the mass of supporting structural members, such as the UAV's wing spar, must be minimized without compromising the vehicle's structural integrity. Given a set of design parameters and specifications, constraint and objective functions can be formulated to govern a numerical design optimization methodology. Utilizing the approaches discussed in this report and nonlinear solvers incorporated in Matrix Laboratory (MATLAB), the optimal design of a wing spar was determined that minimizes the total mass of the structure. From the optimization method, a final mass of 4.9139 kilograms was determined for the spar, which proved to be approximately 63% lower than that of a nominal design. Therefore, this significant decrease in system mass attests to the benefits of this numerical optimization approach and indicates its potential for further use in the aerospace design industry.

Table of Contents

Executive Summary 1

Table of Contents 1

I) Introduction and Problem Statement 2

II) Analysis Overview 2

III) Numerical Optimization Methodology 4

IV) Results 5

V) Discussion 8

VI) Conclusion 10

VII) References 10

VIII) Appendix 11

I) Introduction and Problem Statement

Aircraft spars are considered the main structural members of wings, as they carry the majority of the static and dynamic loads that aerial vehicle experience both on the ground and during flight. However, since there is a large variety in types of aircraft and their applications, there is not one unique shape or composition for aircraft spars. Yet, for most vehicles in the aerospace industry, there is a common goal to minimize the mass of structural members, including the spar, without compromising the structural integrity of the system. This is done purposefully to allow more mass to be allocated for the vehicle's payload or carrying capacity. Thus, for this analysis of an unmanned aerial vehicle that will be utilized as a telecommunications platform, a similar objective is applied that will ultimately lead to numerical optimization of the spar's geometry.

II) Analysis Overview

Prior to completing any numerical optimization of the spar's design, a set of design parameters and specifications was provided by the organization's aerodynamics and manufacturing specialists. This information, tabulated below in Table 1, provided constraints and parameters for the numerical optimization methodology that will be discussed later in this report.

Table 1: UAV Design Parameters and Specifications

Category:	**Description:**
Wing Semi-Span	7.5 meters, which also corresponds to the spar's length
Spar Cross-Section Shape	Circular annulus over the length of the entire spar
Material	Carbon fiber composite with the following material properties: • Density (ρ) = 1600 kg/m^3 • Young's Modulus (E) = 70 GPa • Ultimate tensile/compressive strength (S_{ult}) = 600 MPa
Manufacturing Constraints	The inner and outer radii of the circular annulus cannot be less than 2.5 millimeters apart, and the bounds for the radii are: • $r_{out} \leq 5\ (cm)$ • $r_{in} \geq 1\ (cm)$
Aircraft Operational Weight	Total mass of the aircraft is 500 kg, including the spar
Loading	At a 2.5 g maneuver, the force distribution in the spanwise direction of the wing will have an approximately linear distribution, with the maximum load at the wing root and zero load at the tip

While these specifications provide the constraints that will be used for the numerical optimization of the structure, there are a few critical assumptions inherent to the analysis model. In order to determine the optimal shape for the spar, which entails specifying the inner and outer

radii along the spar's length, the load supplied by the aerodynamics specialist is assumed to be the maximum load that the UAV will experience in its flight environment. Additionally, the load due to the 2.5 g maneuver is assumed to be static with the linear distribution discussed above. As a result, any effects due to dynamic or cyclic loading conditions that the UAV would likely encounter are not taken into consideration for this design optimization. These assumptions do not preclude the method employed below from finding an optimal spar design for the static loading conditions. However, if further analysis was to be conducted, it would be of interest to explore the effects of alternate loading conditions on the numerical optimization.

The subsequent analysis is conducted by modeling the spar using Euler-Bernoulli Beam Theorey. Applying this theorem to the analysis model introduces an additional set of assumptions, which are briefly mentioned below in Table 2.

Table 2, Assumptions for Euler-Bernoulli Beam Theory

Category:	**Description:**
Planar Symmetry	The longitudinal axis of the beam is straight, and the beam's cross-section has a longitudinal plane of symmetry
Cross-Section Variation	The beam's cross-section varies smoothly along its length
Normality	Any plane sections that are normal to the longitudinal plane before bending remain normal after bending
Strain Energy	The beam's internal strain energy accounts only for bending moment deformations
Linearization	Beam deformations are small enough that any nonlinear effects are negligible
Material	The beam's material can be considered elastic and isotropic

From the Euler-Bernoulli Beam Theorem, the vertical displacement of the beam can be modeled as the following 4th order partial differential equation (PDE):

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

where w is the vertical displacement (m), $q(x)$ is the applied load (N/m), E is Young's modulus (Pa), and I_{yy} is the second-moment of area with respect to the y-axis (m^4). Treating the spar as a cantilever beam, the following boundary conditions are also applied to the analytical model:

$$w(x=0) = 0\,;\ \frac{dw}{dx}(x=0) = 0\,;\ \frac{d^2w}{dx^2}(x=L) = 0\,;\ \frac{d^3w}{dx^3}(x=L) = 0$$

Physically, the first two boundary conditions indicate that there is no vertical or angular displacement of the spar at its root, respectively.

Using numerical solving techniques, the PDE specified above can be solved for the displacements, w. With these displacements computed, the normal stress at a specific location, x, along the spar's wing can be calculated from:

$$\sigma_{xx}(x) = -z_{max}E\frac{d^2w}{dx^2},$$

where z_{max} represents the maximum height of the circular annulus cross-section at a given location. For the optimization methodology employed in this analysis, a finite-element discretization is applied to the length of the spar, allowing the 4th order PDE to be solved numerically using Hermite-cubic shape functions.

III) Numerical Optimization Methodology

At its core, the purpose of this analysis is to minimize the weight of a spar for a UAV without allowing the structure to exceed the ultimate strength value dictated by its material properties. The design variables selected for this optimization approach are the outer and inner radii of the spar at a set of nodes along the spar's length. Since the outer and inner radii are being changed directly during the optimization, there is no need for a geometric parameterization for this model. As a finite-element discretization is utilized for the analytical model, the user inputs the number of elements desired prior to conducting the design optimization. With this input, the nodes for the optimization are generated through a uniform spacing of the spar's length divided by the total number of elements. Then, the design variable array containing the inner and outer radii at each node is constructed in the following form:

$$radii = [r_{out1}, r_{in1}, r_{out2}, r_{in2}, \dots, r_{outN}, r_{inN}],$$

where *N* represents the total number of nodes along the spar's length. It is important to note that the total length of this array is equivalent to *2*(Nelem+1)*, as there is both an outer and inner radii value at each node. Once the final values for the outer and inner radii are determined through the optimization, the volume for each element can be computed as the difference between two hollow tapered cylinders, where the volume of a tapered cylinder is given by the following formula:

$$V_{cyl,i} = \frac{1}{3}\pi(r_i^2 + r_i r_{i+1} + r_{i+1}^2),$$

where *i* is a value from one to the total number of elements and *r* represents either the outer or inner radii value at a particular node. Given that the spar will be made of a carbon fiber composite material, the density of the material is known and remains constant. Therefore, summing the elemental volumes and multiplying by the density yields the total mass of the spar, which serves as the objective for this numerical optimization, contained in the spar_obj.m MATLAB function.

With the objective function setup, the next step prior to conducting the optimization is defining the constraints that limit the feasible design space for the spar's final shape. From the provided manufacturing specifications, lower and upper bounds were constructed for the combined radii array, such that the minimum and maximum allowable values for both the outer and inner radii were defined at each node. Additionally, a linear inequality constraint of the following form was specified for the thickness of the circular annulus cross section at each node:

$$r_{out} - r_{in} \geq 2.5\ (mm)$$

While the format was adjusted slightly to comply with the expected input structure for fmincon, the solver-based nonlinear optimization function employed for this optimization, the thickness_constraint.m script composes this linear inequality constraint. The final constraint that must be passed into fmincon during the numerical optimization pertains to the magnitude of the normal stresses generated from the loading condition. The formula shown in the previous section

to compute the normal stress at a given location along the spar's length is nonlinear, which necessitates the formation of a nonlinear constraint in MATLAB. Documented in the stress_constraint.m script, this function takes in the design variables and some of the input parameters, computes the second-moment of area and displacement values at each node, and returns an array of stress inequality constraints of the form:

$$c_{ineq,i} = \frac{\sigma_i}{S_{ult}} - 1,$$

where σ_i is the stress value at a specific node calculated in CalcBeamStress.m. These constraints establish the feasible design space, which subsequently guide the fmincon optimization solver to find a minimum for the objective function.

Incorporated into both the objective and nonlinear constraint scripts, the complex-step method is utilized to approximate the gradients for each of these functions. Although fmincon computes the gradients when it is run, its default method utilizes finite-differencing techniques. Therefore, implementing the complex step method, which does not require differences of function values in its derivation, eliminates error due to round-off and increases the accuracy of the optimized design.

The final step before the optimization approach can be implemented is to define an initial guess for the design variables, the spar's outer and inner radii at each node. The selection for this preliminary design is governed by the imposed manufacturing constraints, such that all values for the outer and inner radii are 5-centimeters and 1-centimeter, respectively. After the initial guess, constraints, and objective function are defined, the algorithm selected for fmincon's optimization is Sequential Quadratic Programming (SQP). The SQP algorithm, which has many similarities to Newton's method for constrained optimization, is considered to be the state-of-the-art methodology in nonlinear programming methods [2]. Thus, utilizing this algorithm and the previously discussed constraints, the optimization script can be completed with fmincon, leading to a design that optimally satisfies the constraints and objective function.

IV) Results

Upon initial implementation of the numerical optimization methodology, it was noted that varying the input value for *Nelem* can have an impact on the operational behavior of the solver, including the required computational time and final optimal design. Given these variations, a mesh convergence study was conducted to determine the impact that altering *Nelem* had on the approach. Specifically, the final mass in kilograms and the time to conduct the optimization in seconds was determined for a total of 10 trials, corresponding to evenly spaced values of *Nelem* from 10 to 100. Figure 1 on the subsequent page illustrates how these variables change with *Nelem*.

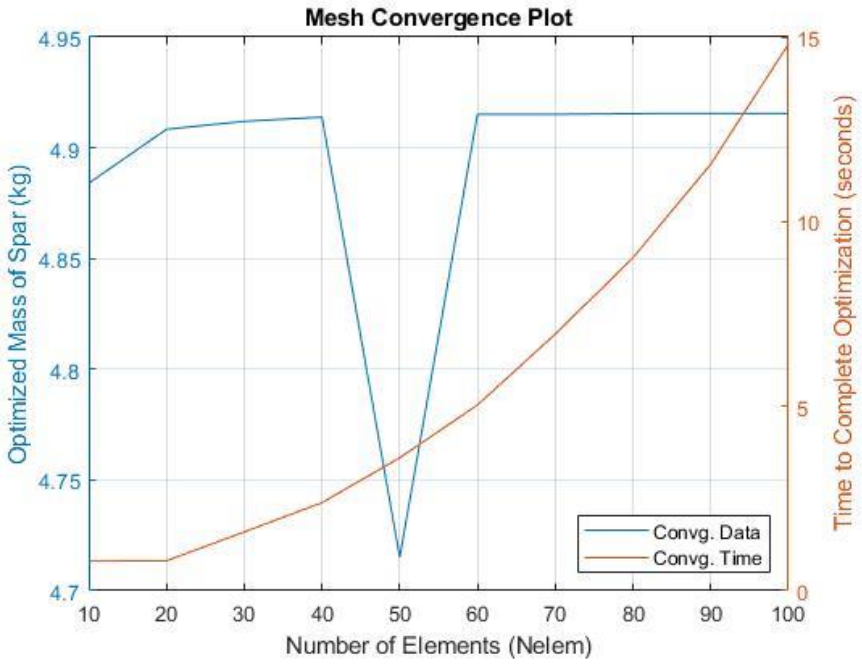

Figure 1: Mesh Convergence Trade Study

Following the completion of this trade study, it was determined to conduct the remaining optimization trials with 40 total elements along the length of the spar. After fixing *Nelem*, the main optimization script was run again, yielding the optimal outer and inner radii at each node. Figure 2 depicts how the radii vary along the spar's total length. A three-dimensional rendering of the optimized wing spar was also created and is shown in Figure 3 on the next page.

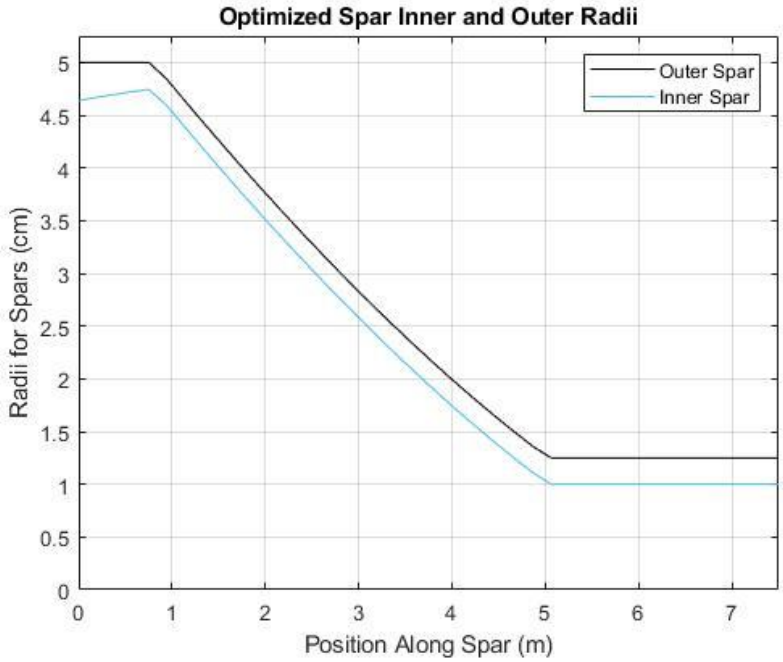

Figure 2: Optimized Radii Along Spar's Length for 40-elements

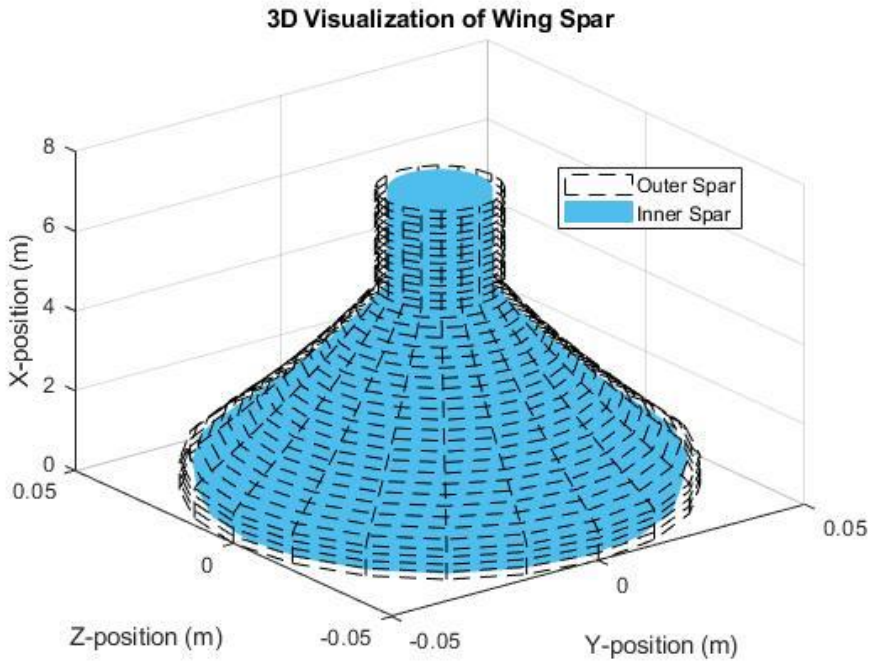

Figure 3, 3D Rendering of Optimized Spar

While the optimized shape of the spar is of interest, it is also significant to consider the nodal displacements and stresses of the resulting design. This data is plotted below in Figure 4 as a function of the position along the spar for the 40-element optimization trial.

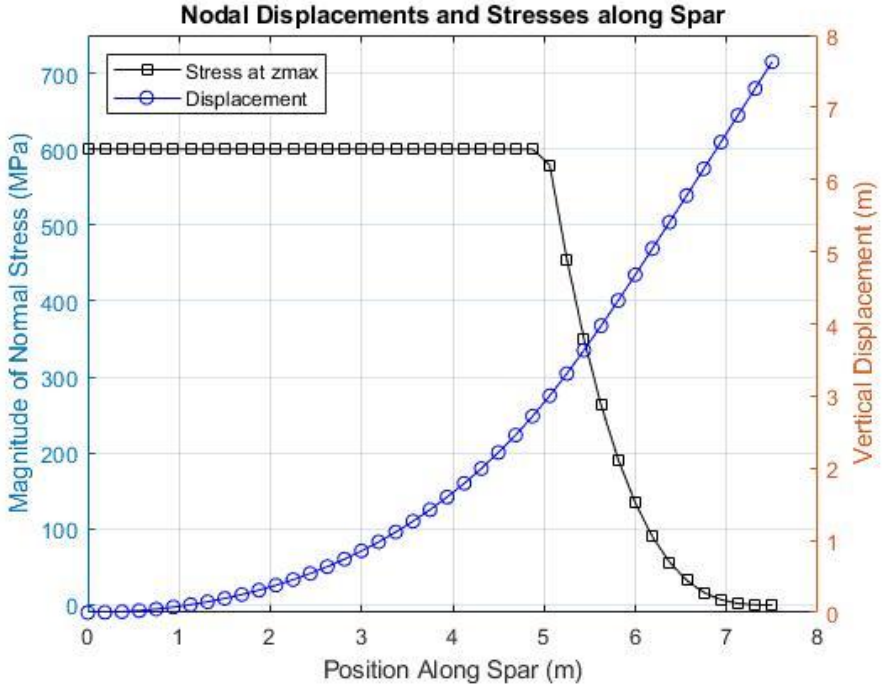

Figure 4: Nodal Displacements and Stresses for 40-element Optimization

Completing this numerical optimization ultimately yielded a spar mass of approximately 4.9139 kilograms. However, it was necessary to confirm that the numerical optimizer was able to achieve a minimizer that satisfies the objective and constraint functions. To address this, a plot, shown below in Figure 5, was generated that displays the optimization history for the first-order optimality and spar mass values during the 40-element optimization analysis.

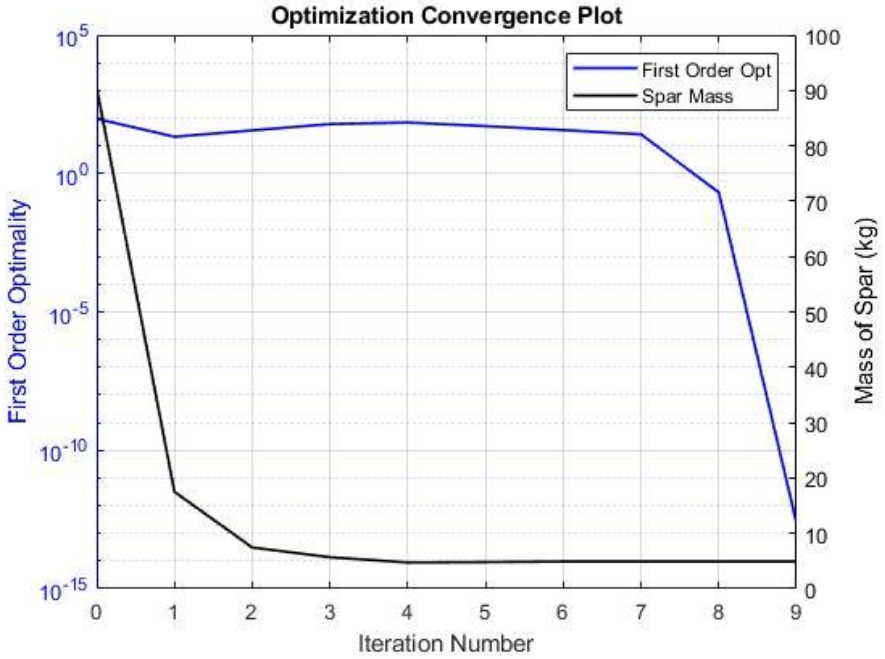

Figure 5: Wing Spar Optimization Convergence Study for 40-elements

V) Discussion

Before utilizing the optimization approach to generate any of the final radii values corresponding to the optimal design presented above, the mesh convergence trade study provided valuable insight into the relationship between the total number of elements, spar mass, and computational time. From inspection of Figure 1, it appears that the optimized value for the spar's mass quickly plateaus after the number of elements exceeds 30. One exception to this trend appears when *Nelem* is equal to 50 elements, as there is a relatively stark decrease in the mass value with this number of elements. While this smaller mass value is favorable, inspection of the optimized radii for 50-elements illustrated sharp changes in radii values around 0.75 meters, which would be nonsensical for a physical spar. Therefore, treating this point as an outlier, it is readily apparent that the mass value is converging to a value of approximately 4.915 kilograms. Conversely, the computational time required to solve the optimization problem increases quadratically as the number of elements increases. This differing behavior poses an interesting trade-off between model accuracy and time to compute a solution. Comparing the mass values for the 40-element and 90-element trials, which are 4.9139 kilograms and 4.9155 kilograms, respectively, there is about a 0.03% difference in their optimal designs. However, the percent difference in the computational times for these two element trials (2.3870 seconds and 11.5574 seconds) is

drastically larger at nearly 125%. From this analysis, it can therefore be concluded that increasing the number of elements does not impact minimization of the objective function enough that it is worth the significant increase in computational time. For this reason, the subsequent optimization studies were conducted with 40-elements, as this was deemed to be sufficiently accurate and computationally beneficial.

With the value of *Nelem* now constant for the remaining numerical analysis, the outer and inner radii were optimized by fmincon to generate the design shown in Figure 2 and Figure 3. Examining the nodal stress and displacement data in Figure 4 expounds the success of the optimization approach. Regarding the Euler-Bernoulli Beam Theorem discussed previously, which was used as part of the analytical model for the spar's optimization, several boundary conditions were introduced. The displacement data in Figure 4 illustrates that there is zero vertical displacement at the wing root, and there is approximately zero normal stress for the node located at the end of the spar's length. This behavior adheres to the boundary conditions imposed by the Euler-Bernoulli Beam Theorem, validating that the model was accurately implemented. It is also significant to highlight that the constraints supplied to fmincon remain satisfied throughout the optimization process. Specifically, at no point along the spar's length do the radii values defy the manufacturing constraints, nor does the magnitude of the normal stress at any point exceed that of the ultimate tensile and compressive strength for the carbon fiber composite.

From optimizing the objective function defined previously for the spar, one of the primary outputs of this numerical analysis is the structural member's total mass. In particular, the mass generated from this optimization is to be compared to that of a nominal design with 20-elements, where the outer and inner radii remain constant along the spar's length at 5-centimeters and 4.635-centimeters, respectively. Compared to the nominal value, the percent decrease in the optimized design's mass is given by:

$$Percent\ Decrease\ = \frac{mass_{nominal} - mass_{optimized}}{mass_{nominal}} * 100\% = \frac{13.26kg - 4.9139kg}{13.26kg} * 100\%,$$

which yields a percent decrease of approximately 62.94%. With an initial goal of reducing the spar's weight relative to the nominal by 60%, the optimization methodology is evidently successful in its purpose.

Although surpassing the optimized mass threshold relative to the nominal value suggests that the nonlinear optimizer achieved a local minimizer, it is important to verify this through examination of the convergence history. Shown in Figure 5, the first-order optimality measure and spar mass in kilograms are plotted as a function of the number of optimizer iterations. For both of these data, as the number of iterations increases, the first-order optimality and values for the spar's mass are seen to decrease until reaching a minimum value. At the ninth iteration, the algorithm was stopped because the relative first-order optimality measure (2.5060e-15) and relative maximum constraint violation (4.5297e-13) were both less than their tolerance values of 1e-6 [3]. Consequently, the numerical optimizer successfully found a local minimizer that satisfied the constraints, culminating in an optimal design for the UAV's spar.

VI) Conclusion

The numerical optimization methodology applied to the analysis of the UAV's wing spar ultimately resulted in approximately a 63% decrease in total mass relative to the nominal design value. Utilizing computational resources to complete a mesh convergence study granted baseline knowledge on the trade-off between objective accuracy and elapsed time for the nonlinear optimizer. This information led to an informed determination regarding the required number of elements for sufficient optimization accuracy while adhering to the provided design constraints. When considering the displacement plot shown previously in Figure 5, one additional note can be made regarding the magnitude of the displacement values seen for the optimized design. While the wing's semi-span is 7.5 meters in length, the displacement value seen at the wing tip exceeds the length of the semi-span at approximately 7.63 meters. Though modern aircraft wings are designed to endure significant vertical deflection before serious structural damage can occur, the values that result from this optimization are larger than anticipated. This may be a function of the limitations inherent in the current analytical model, requiring more detailed studies to be done to gather accurate information on the true structural response of the UAV's spar given the prescribed loading conditions. Therefore, incorporating more elements in the analytical model, such as a maximum vertical displacement constraint and the effect of cyclic loads, may give further clarity on the spar's optimal shape. Nevertheless, the numerical optimization conducted in this analysis provides a valuable foundation for ensuring that the UAV's structural design enables it to achieve its objectives as a long-endurance aircraft capable of serving as a future telecommunications platform.

VII) References

[1] "Zephyr: Pioneering the Stratosphere." Airbus, Airbus S.A.S., 2021, https://www.airbus.com/defence/uav/zephyr.html#introduction.

[2] "Constrained Nonlinear Optimization Algorithms." MathWorks Help Center, The MathWorks, Inc., 2021, https://www.mathworks.com/help/optim/ug/constrained-nonlinear-optimization-algorithms.html.

[3] "First-Order Optimality Measure." MathWorks Help Center, The Mathworks, Inc., 2021, https://www.mathworks.com/help/optim/ug/first-order-optimality-measure.html.

VIII) Appendix

Source Code

Note: MATLAB functions inherent to the optimization script that were provided (i.e. CalcBeamStress.m, CalcBeamDisplacement.m, etc.) have not been included in this appendix, as no modifications were made to the original scripts.

proj2_opt.m

```
% Script that conducts the optimization of the wing spar.
% Inputs:
%   No inputs, but parameters are passed in with the parameters function.
% Outputs:
%   1) If mesh convergence study is desired, outputs the final mass values
%   and optimization time for a range of values for Nelem (10:10:100); if
%   this is not the output of interest, then 2-4 are returned
%   2) Final mass of the optimized spar and the percent decrease relative
%   to the nominal value
%   3) Plot of the outer and inner radii as a function of the position
%   along the spar
%   4) Plot that shows the nodal displacements and stresses for the
%   optimized spar
%-------------------------------------------------------------------------
clear all
close all
clc

% Control to determine if mesh convergence study should be conducted
convg = false; % true or false

% Call parameters() to initialize parameters for optimization
[L, Nelem, m, maneuver, upper, lower, thickness, rho, E, S_ult] = ...
    parameters();

if convg
    % Define fmincon options for mesh convergence study note: specify
    % whether 'CheckGradients' should be true or false to check
    % whether complex-step method is working
    options = optimoptions(@fmincon,'Display', 'final',...
        'Algorithm','sqp',...
        'CheckGradients', false,...
        'SpecifyObjectiveGradient', true,...
        'SpecifyConstraintGradient', true,...
        'PlotFcn', {@optimplotfirstorderopt, @optimplotfval});
    Nelem = [10:10:100]';
    mass_opt = zeros(size(Nelem,1),1);
    time_fmincon = zeros(size(Nelem,1),1);
    % Loop through the number of Nelems to get optimized mass values for
    % varying number of elements along spar's length
    for i = 1:size(Nelem,1)
        % Define initial guess for r_out and r_in
        r0_out = zeros(Nelem(i)+1, 1) + 0.05;
        r0_in = zeros(Nelem(i)+1, 1) + .01;

        % Create initial radii array from r0_out and r0_in
        radii0 = zeros(2*(Nelem(i, 1)+1),1);
```

```
        for j = 1:size(r0_out, 1)
            radii0(1+2*(j-1)) = r0_out(j);
            radii0(2*j) = r0_in(j);
        end

        % Create radii upper and lower bound constraints
        [lb, ub] = radii_bounds(Nelem(i), upper, lower, thickness);

        % Create linear inequality constraints
        [Aineq, bineq] = thickness_constraint(Nelem(i), thickness);

        % Use anonymous function to get nonlinear stress constraint function
        noncon_stress = @(radii) ...
            stress_constraint(radii, Nelem(i), L, E, m, maneuver, S_ult);

        % Define objective function using anonymous function
        m_spar = @(radii) spar_obj(radii, L, Nelem(i), rho);

        startTime = tic;
        % Use fmincon to optimize mass of spar
        [radii_opt, fval, exitflag, output] = fmincon(m_spar, radii0,...
            Aineq, bineq, [], [], lb, ub, noncon_stress, options);

        time_fmincon(i) = toc(startTime);

        % Store optimized mass value for given Nelem in mass_opt array
        mass_opt(i) = fval;
    end
    % Create plot to show optimized mass value vs Nelem
    figure(2)
    plot(Nelem, mass_opt)
    xlabel('Number of Elements (Nelem)')
    ylabel('Optimized Mass of Spar (kg)')
    title('Mesh Convergence Plot')
    grid on
else
    % Define fmincon options for regular optimization script; note: specify
    % whether 'CheckGradients' should be true or false to check
    % whether complex-step method is working
    options = optimoptions(@fmincon,'Display', 'iter','Algorithm','sqp',...
        'CheckGradients', false,...
        'FiniteDifferenceType', 'central',...
        'SpecifyObjectiveGradient', true,...
        'SpecifyConstraintGradient', true,...
        'PlotFcn', {@optimplotfirstorderopt, @optimplotfval});

    % Define initial guess for r_out and r_in
    r0_out = zeros(Nelem+1, 1) + 0.05;
    r0_in = zeros(Nelem+1, 1) + .01;

    % Create initial radii array from r0_out and r0_in
    radii0 = zeros(2*(Nelem+1),1);
    for i = 1:size(r0_out, 1)
        radii0(1+2*(i-1)) = r0_out(i);
        radii0(2*i) = r0_in(i);
    end
```

```
% Create radii upper and lower bound constraints
[lb, ub] = radii_bounds(Nelem, upper, lower, thickness);

% Create linear inequality constraints
[Aineq, bineq] = thickness_constraint(Nelem, thickness);

% Use anonymous function to get nonlinear stress constraint function
noncon_stress = @(radii) ...
    stress_constraint(radii, Nelem, L, E, m, maneuver, S_ult);

% Define objective function using anonymous function
m_spar = @(radii) spar_obj(radii, L, Nelem, rho);

% Start timer to determine time for fmincon to complete
startTime = tic;

% Use fmincon to optimize mass of spar
[radii_opt, fval, exitflag, output] = fmincon(m_spar, radii0, Aineq, ...
    bineq, [], [], lb, ub, noncon_stress, options);
time_fmincon = toc(startTime);
sprintf('The optimization took %g seconds to complete.', time_fmincon)

% Extract values for optimized outer and inner radii
[r_out_opt, r_in_opt] = extract_radii(radii_opt, Nelem);

% Finding stress at each node with optimized values
q = load_distr(L, Nelem, m, maneuver);
Iyy_opt = second_moment_area(radii_opt, Nelem);
disp_opt = CalcBeamDisplacement(L, E, Iyy_opt, q, Nelem);
sigma_opt = CalcBeamStress(L, E, r_out_opt, disp_opt, Nelem);

% Get x value locations for nodes along spar's length
x = nodes(L, Nelem);

% Plot optimized radii for spar over its length
figure(2)
plot(x, r_out_opt*100, 'k-')
hold on
plot(x, r_in_opt*100, 'Color', [0.3010 0.7450 0.9330], ...
    'LineStyle', '-')
xlim([0, L]);
ylim([0, 5.25]);
xlabel('Position Along Spar (m)')
ylabel('Radii for Spars (cm)')
legend('Outer Spar', 'Inner Spar')
title('Optimized Spar Inner and Outer Radii')
grid on

% Plot the stress and displacement at each node along the spar's length
figure(3)
yyaxis left
plot(x, sigma_opt/10^6, 'ks-')
ylabel('Magnitude of Normal Stress (MPa)')
ylim([-10, 750])
yyaxis right
plot(x, disp_opt(1:2:end), 'bo-')
ylabel('Vertical Displacement (m)')
```

```
    xlabel('Position Along Spar (m)')
    legend('Stress at zmax', 'Displacement', 'Location', 'northwest')
    title('Nodal Displacements and Stresses along Spar')
    grid on

    % Determine the percent decrease/increse in spar's total mass
    % Define the nomainal mass value for comparison
    mass_nominal = 13.26;
    mass_opt = fval;
    if mass_opt < mass_nominal
        per_decr = (mass_nominal - mass_opt)/mass_nominal*100;
        str = ['The optimized mass of the spar, %.4f kg, is approximately'...
                ' a %.2f percent decrease relative to the nominal value.'];
        sprintf(str, mass_opt, per_decr)
    elseif mass_opt > mass_nominal
        str = ['The optimized mass of the spar, %.4i kg, is not an'...
            ' improvement on the nominal value of the spar.'];
        sprintf(str, mass_opt)
    end
end
```

proj2_plots.m

```
% Script that creates plots to support mesh convergence and optimization
% convergence studies for wing spar optimization
% Inputs:
%   - No inputs besides loading .mat files that contain relevant plotting
%   data
% Outputs:
%   1) If mesh_convg = true, returns a plot that shows how the
%   computational time and optimized mass value vary as the number of
%   elements varies from 10 to 100 with a step size of 10 elements
%   2) If revolve = true, returns a 3D visualization of the optimized shape
%   of the wing spar for a 40-element optimization trial
%   3) If opt_convg = true, returns a plot that illustrates the
%   optimization convergence history, specifically the mass values and
%   first order optimality, for a 40-element optimization trial
%-------------------------------------------------------------------------
clear all
close all
clc

% Define controls to determine which plots to create
mesh_convg = true; %true/false
revolve = false; %true/false
opt_convg = false; %true/false

% If mesh_convg is true, create the plot to show mesh convergence
if mesh_convg
    % Load mesh convergence data
    load('Mesh_Convergence_Study_10_10_100.mat')
    yyaxis left
    plot(Nelem, mass_opt)
    ylabel('Optimized Mass of Spar (kg)')
    yyaxis right
    plot(Nelem, time_fmincon)
    ylabel('Time to Complete Optimization (seconds)')
```

```
    xlabel('Number of Elements (Nelem)')
    title('Mesh Convergence Plot')
    legend('Convg. Data','Convg. Time', 'location', 'southeast')
    grid on
end

if revolve
    % Load 40 element optimized radii data
    load('Opt_Radii_40elem.mat')
    [x_out, y_out, z_out] = cylinder(r_out_opt);
    [x_in, y_in, z_in] = cylinder(r_in_opt);

    out = surf(x_out, y_out, z_out*7.5, 'EdgeColor', 'k', ...
        'Facecolor', 'none', 'LineStyle', '--');
    hold on
    in = surf(x_in, y_in, z_in*7.5, 'EdgeColor', [0.3010 0.7450 0.9330], ...
        'Facecolor', [0.3010 0.7450 0.9330], 'LineStyle', '-');
    xlabel('Y-position (m)')
    ylabel('Z-position (m)')
    zlabel('X-position (m)')
    title('3D Visualization of Wing Spar')
    legend('Outer Spar', 'Inner Spar')
    grid on
end

if opt_convg == true
    load('opt_convg_plot_hist.mat')
    fig = figure;
    left_color = [0 0 1];
    right_color = [0 0 0];
    set(fig,'defaultAxesColorOrder',[left_color; right_color]);
    yyaxis left
    plot(iterations, firstorderopt, 'b', 'linewidth', 1.25)
    ylabel('First Order Optimality')
    set(gca, 'YScale', 'log')
    yyaxis right
    plot(iterations, fval_opt, 'k', 'linewidth', 1.25)
    ylabel('Mass of Spar (kg)')
    xlabel('Iteration Number')
    title('Optimization Convergence Plot')
    legend('First Order Opt', 'Spar Mass')
    grid on
end
```

spar_obj.m

```
function [m_spar, grad_m_spar] = spar_obj(radii, L, Nelem, rho)
% Computes the objective function, the mass of the spar, and its gradient
% for the wing spar design optimization problem
% Inputs:
%   radii - size(2*(Nelem+1, 1) array that contains the inner and outer
%   radii information at each node along the spar's length
%   L - total length of the spar in the x-direction in meters
%   Nelem - total number of elements that the spar will be divided into
%   rho - density of the material used for wing spar
% Outputs:
%   m_spar - objective function, the total mass of the spar, which is to be
```

```
%   optimized
%   grad_m_spar - size(2*(Nelem+1), 1) array that represents the gradient
%   of the objective function at each value in the input radii array
%-------------------------------------------------------------------------
% Compute the objective function using the the subfunction, spar_subobj
m_spar = spar_subobj(radii);

% Initialize the size of the gradient
grad_m_spar = zeros(size(radii,1), 1);

% Define h, complex-step size
h = 1e-60;

% Loop through based on number of design variables in radii to compute the
% gradient of the objective using the complex-step method
for j = 1:size(radii,1)
    radii_c = radii;
    radii_c(j) = radii_c(j) + complex(0, h);
    grad_m_spar(j) = imag(spar_subobj(radii_c))/h;
end

    % Define subfunction to compute the objective function, the mass of the
    % spar
    function [m_spar_fval] = spar_subobj(dv)
        % Subfunction that is used to compute the objective function for
        % the optimization problem, the mass of the wing's spar
        % Inputs:
        %   dv - design variables (the outer and inner radii) used to
        %   compute the objective function
        % Outputs:
        %   m_spar_fval - objective function, the total mass of the spar,
        % which is to be optimized
        %-----------------------------------------------------------------
        % Extract the outer and inner radii values from input radii array
        [r_out, r_in] = extract_radii(dv, Nelem);

        % Initialize volume array that represents the volume for each element
        volume = zeros(Nelem, 1);

        % Define length of segments
        L_element = L/Nelem;

        % Loop by number of elements to compute the volume for each
element/segment
        for i = 1:size(volume, 1)
            v_out =
pi*L_element*(r_out(i+1)^2+r_out(i+1)*r_out(i)+r_out(i)^2)/3;
            v_in = pi*L_element*(r_in(i+1)^2+r_in(i+1)*r_in(i)+r_in(i)^2)/3;

            volume(i) = v_out - v_in;
        end

        % Compute the total spar volume by summing the volume elements
        spar_volume = sum(volume);

        % Compute the mass of the spar using the known density and computed
volume
```

```
        m_spar_fval = spar_volume*rho;
    end
end
```

stress_constraint.m

```
function [cineq, ceq, Jineq, Jeq] = ...
    stress_constraint(radii, Nelem, L, E, m, maneuver, S_ult)
% Defines the nodal stress constraints for the spar optimization
% problem in the form of a nonlinear inequality constraint
% Inputs:
%   radii - size(2*(Nelem+1, 1) array that contains the inner and outer
%   radii information at each node along the spar's length
%   Nelem - total number of elements that the spar will be divided into
%   L - total length of the spar in the x-direction in meters
%   E - Young's modulus for the wing spar's material in Pa
%   m - total mass of the aircraft in kg
%   maneuver - number that designates the maneuver in question for the
%   aircraft
%   S_ult - ultiamte tensile/compressive strength for the spar's material
%   in Pa
% Outputs:
%   cineq - size(Nelem+1, 1) array that contains the nonlinear stress
%   inequality constraints at each node along the wing spar
%   ceq - empty array, as there are no nonlinear equality constraints for
%   the wing spar optimization problem
%   Jineq - constraint Jacobian for the nonlinear stress constraint, such
%   that the constraint gradients are contained in the columns of the
%   matrix; computed using the complex-step method
%   Jeq - empty array, as there are no nonlinear equality constraints for
%   the wing spar optimization problem, so no equalty constraint
%   Jacobian is required
%-------------------------------------------------------------------------
% Initialize the nonlinear stress constraint using the subfunction
[cineq, ceq, Jineq, Jeq] = stress_constraint_sub(radii);

% Initialize the size for the Jineq matrix based on the number of
% constraints and design variables
Jineq = zeros(size(radii, 1), size(cineq, 1));

% Define h, complex-step size
h = 1e-30;

% Utilizing loops, define the stress (nonlinear) inequality constraint
% Jacobian using the complex_step method
% Loop through the rows (number of constraints) of the Jineq matrix
for r = 1:size(radii,1)
    radii_cmplx = radii;
    radii_cmplx(r) = radii_cmplx(r) + complex(0, h);
    Jineq(r,:) = imag(stress_constraint_sub(radii_cmplx))/h; % need to
utilize some function here, but don't know what
end

    function [sub_cineq, sub_ceq, sub_Jineq, sub_Jeq] = ...
            stress_constraint_sub(dv)
        % Subfunction that is used to compute the nonlinear equality and
        % inequality constraints for the wing spar optimization.
```

```
        % Inputs:
        %   dv - design variables (outer and inner radii), which are used
        %   to define the stress constraints
        % Outputs:
        %   sub_cineq - nonlinear inequality constraint array for the
        %   subfunction
        %   sub_ceq - nonlinear equality constraint array for the
        %   subfunction, which is an empty array for this optimization
        %   sub_Jineq - constraint Jacobian for the nonlinear inequalty
        %   constraints
        %   sub_Jeq - constraint Jacobian for the nonlinear equality
        %   constraints, which is an empty array for this optimization
        %------------------------------------------------------------------
        % Since there are not equality constraints, return ceq and Jeq as
        % nothing
        sub_ceq = [];
        sub_Jeq = [];

        % First, extract r_out and r_in from radii input array
        [r_out, ~] = extract_radii(dv, Nelem);

        % Compute values for Iyy at each node using input radii
        Iyy = second_moment_area(dv, Nelem);

        % Call load_distr to get force per unit length at each node along
        % spar
        q = load_distr(L, Nelem, m, maneuver);

        % Call CalcBeamDisplacement to get displacements at each node
        u = CalcBeamDisplacement(L, E, Iyy, q, Nelem);

        % Call CalcBeamStress to get stress at each node
        sigma = CalcBeamStress(L, E, r_out, u, Nelem);

        % Loop through by number of nodes to define nonlinear inequality
        % constraint at each node
        sub_cineq = zeros(Nelem+1, 1);
        for i = 1:Nelem+1
            sub_cineq(i) = sigma(i)/S_ult - 1;
        end
        sub_Jineq = [];
    end
end
```

thickness_constraint.m

```
function [Aineq, bineq] = thickness_constraint(Nelem, thickness)
% Defines the manufacturing thickness constraint for the spar optimization
% problem in the form of a linear inequality constraint
% Inputs:
%   Nelem - total number of elements that the spar will be divided into
%   thickness - minimum distance that can occur between the outer and inner
%   radii at any given point along the spar's length
% Outputs:
%   Aineq - real matrix of size m by n, where m is the number of
%   inequalities and n is the number of design variables in a
%   bineq - real vector of size m by 1, where m is the number of
```

```
%   inequalities
%-------------------------------------------------------------------------
% Initialize sizes of Aineq and bineq
Aineq = zeros(Nelem+1, 2*(Nelem+1));
bineq = zeros(Nelem+1, 1);

% Define bineq column vector based on specified Nelem
bineq = bineq - thickness;

% Loop by number of inequalities
for i = 1:Nelem+1
    % Define coefficients for Aineq matrix
    Aineq(i, 1+2*(i-1)) = -1.0;
    Aineq(i, 2*i) = 1.0;
end
end
```

extract_radii.m

```
function [r_out, r_in] = extract_radii(radii, Nelem)
% Function that extracts the inner and outer radii at each node along the
% length of the spar from the all-inclusive radii array
% Inputs:
%   radii - size(2*(Nelem+1), 1) array that contains the inner and outer
%   radii information at each node along the spar's length
%   Nelem - total number of elements that the spar will be divided into
% Outputs:
%   r_out - size(Nelem+1, 1) array that contains the outer radii
%   information at each node along the spar's length
%   r_in - size(Nelem+1, 1) array that contains the inner radii information
%   at each node along the spar's length
%-------------------------------------------------------------------------
if size(radii, 1) == 2*(Nelem+1)
    % Initialize arrays for r_out and r_in based on Nelem
    r_out = zeros(Nelem+1, 1);
    r_in = zeros(Nelem+1, 1);

    % Loop through radii array to create r_out and r_in
    for i = 1:size(radii, 1)/2
        r_out(i, 1) = radii(1+2*(i-1));
        r_in(i, 1) = radii(2*i);
    end
else
    disp('The size of the radii input array does not correspond to Nelem.')
end
end
```

second_moment_area.m

```
function [Iyy] = second_moment_area(radii, Nelem)
% Computes the second moment of area for the circular annulus cross section
% at each node along the length of the spar.
% Inputs:
%   radii - size(2*(Nelem+1), 1) array that contains the inner and outer
%   radii information at each node along the spar's length
%   Nelem - total number of elements that the spar will be divided into
% Outputs:
```

```
%   Iyy = size(Nelem+1, 1) array that contains the second moment of area
%   value at each node along the spar's length
%-------------------------------------------------------------------------
% Extract the outer and inner radii values from input radii array
[r_out, r_in] = extract_radii(radii, Nelem);

% Initailize the Iyy vector based on Nelem
Iyy = zeros(size(r_out));

% Define minimum threshold for Iyy values
Iyy_thresh = 1e-12;

% Loop through by each node to compute the second moment of area
for i = 1:size(Iyy, 1)
    Iyy(i) = pi*(r_out(i)^4 - r_in(i)^4)/4;
    if Iyy(i) < Iyy_thresh
        Iyy(i) = Iyy_thresh;
    else
        continue
    end
end
end
```

radii_bounds.m

```
function [lb, ub] = radii_bounds(Nelem, upper, lower, thickness)
% Defines the lower and upper bound manufacturing constraints for the wing
% spar optimization problem.
% Inputs (in meters):
%   upper - upper value that the outer radii can take along the length
%   lower - lower value that the inner radii can take along the length
%   thickness - minimum distance that can occur between the outer and inner
%   radii at any given point along the spar's length
% Outputs:
%   lb - size(2*(Nelem+1), 1) array that contains the minimum acceptable
%   values for the inner and outer radii at any point along the spar
%   ub - size(2*(Nelem+1), 1) array that contains the maximum acceptable
%   values for the inner and outer radii at any point along the spar
%-------------------------------------------------------------------------
% Initialize the arrays for lb and ub based on Nelem
lb = zeros(2*(Nelem+1), 1);
ub = zeros(2*(Nelem+1), 1);

% Loop through the two arrays based on Nelem to define lb and ub
for i = 1:size(lb, 1)
    % Check to see if index is odd or even to fill lb and ub
    if mod(i, 2) ~= 0
        lb(i) = lower + thickness;
        ub(i) = upper;
    else
        lb(i) = lower;
        ub(i) = upper - thickness;
    end
end
end
```

load_distr.m

```
function [q] = load_distr(L, Nelem, m, maneuver)
% Returns the force per unit length for the wing spar optimization problem
% at each of the nodes along the x direction
% Inputs:
%   L - total length of the spar in the x-direction in meters
%   Nelem - total number of elements that the spar will be divided into
%   m - total mass of the aircraft in kg
%   maneuver - number that designates the maneuver in question for the
%   aircraft
% Outputs:
%   q - size(Nelem+1, 1) array that contains the force per unit length
%   value at each node along the spar's length
%-------------------------------------------------------------------------
% Find the total force from the load distribution
F = total_force(m, maneuver);

% Compute the height of the triangular distributed load
h = 2*F/L;

% Define slope of triangular distributed load
slope = -h/L;

% Find the node locations based on the input parameters L and Nelem
x = nodes(L, Nelem);

% Compute q, force per unit length, at each node in x
q = h + slope.*x;
end
```

total_force.m

```
function [F] = total_force(m, maneuver)
% Computes the total force applied to one wing of the aircraft given the
% aircraft's total weight and the maneuver that results in max stress.
% Inputs:
%   m - total mass of the aircraft in kg
%   maneuver - number that designates the maneuver in question for the
%   aircraft
% Outputs:
%   F - total force applied on one of the aircraft's wings
%-------------------------------------------------------------------------
% Compute F
F = 9.8*m*maneuver/2;
end
```

nodes.m

```
function [x] = nodes(L, Nelem)
% Defines the location of each node for the wing spar optimization.
% Inputs:
%   L - total length of the spar in the x-direction in meters
%   Nelem - total number of elements that the spar will be divided into
% Outputs:
%   x - size(Nelem+1, 1) array that contains the x location for each node
%   to be considered in the optimization problem
%-------------------------------------------------------------------------
```

```
% Compute the x array with defined length of spar and Nelem
x = 0:L/Nelem:L;
x = x';
end
```

parameters.m

```
function [L, Nelem, m, maneuver, upper, lower, thickness, rho,...
    E, S_ult] = parameters()
% Function that outputs the input parameters for the main optimization
% script. The user can alter these parameters as needed to adjust the
% optimization of the spar's weight.
% Inputs:
%   No inputs for this function.
% Outputs:
%   L - total length of the spar in the x-direction in meters
%   Nelem - total number of elements that the spar will be divided into
%   m - total mass of the aircraft in kg
%   maneuver - number that designates the maneuver in question for the
%   aircraft
%   upper - upper value that the outer radii can take along the length
%   lower - lower value that the inner radii can take along the length
%   thickness - minimum distance that can occur between the outer and inner
%   radii at any given point along the spar's length
%   rho - density of the material used for wing spar
%   E - Young's modulus for the wing spar's material in Pa
%   S_ult - ultiamte tensile/compressive strength for the spar's material
%   in Pa
%-------------------------------------------------------------------------
% Input parameters that can be adjusted by the user
Nelem = 40;

% Constant parameters
L = 7.5;
m = 500;
maneuver = 2.5;
E = 70*10^9;
S_ult = 600*10^6;
rho = 1600;

% Manufacturing Constraints
upper = 0.05;
lower = 0.01;
thickness = 0.0025;
end
```