Toronto Metropolitan University

Department of Aerospace Engineering
Faculty of Engineering & Architectural Science

Course Code: AER404

Course Title: Introduction to Aerospace Engineering

Semester: 4

Instructor: Dr. Reza Faieghi

TA: Surrayya Mobeen

Assignment: Assignment 2

Section Number: 2

Submission Date: March 31, 2025

Due Date: March 31, 2025

Name	**Student Number** (XXXX99999)	**Signature**
Alec Tabachnik	45432	AT

By signing above you attest that you have contributed to this submission and confirm that all work you have contributed to this submission is your own work. Any suspicion of copying or plagiarism in this work will result in an investigation of Academic Misconduct and may result in a "0" on the work, an "F" in the course, or possibly more severe penalties, as well as a Disciplinary Notice on your academic record under the Student Code of Academic Conduct, which can be found online at: www.ryerson.casenate/current/pol60.pdf .

Introduction

Based on a given airfoil size, a code was written to determine the most optimal spar configuration for a length of 1.8m. The 3 possible shapes to be considered for this assignment were given as a solid rod, hollow pipe, and rectangular bar. An analysis of materials should also be conducted in order to determine the most optimal spar configuration. The 2 given materials were Aluminium and Carbon Fibre, with respective yield strengths of 55 and 131 MPa. All spars can have a maximum deflection of 5% of the wing length (1.8m). The Young's modulus', densities and costs per kg were also given for Aluminium and Carbon Fibre as 69 and 134 GPa, 2.70 and 1.60 (g/cm^3), and 2.53 and 55 ($USD/kg) respectively. The code is required to find the most optimal configuration for the spar based on price (after factoring in strength requirements).

Code Analysis

The maximum diameter chosen was 0.055m. This value was found by cross referencing the Eppler 420 airfoil on a CAD software, where it is evident that a maximum diameter of 0.055m would fit at the quarter chord length.

The approach taken to solve this problem is as follows. The code should determine all possible configurations that will fit within the airfoil, and store these values within an array (one for rod, one for pipe, one for rectangular bar). The code should then be able to filter and store the possible size configurations by yield strength, and maximum deflection. Once all of the filtering is done, the code would then need to locate the lowest price values for each of the 6 configurations, and the associated dimensions with this configuration.

There were many ways to go through each of the possible configurations for the inner/outer diameters for the pipe, and for base/height values for the rectangular rods. However the chosen approach was to make a random number generator, which would generate a number within the allowable conditions. In the rod section of the code, a line is added to ensure that the minimum thickness is withheld during the trial stages of the loop. It is important to note that more iterations would result in lowest price values (more optimal results).

Since the code allowed a couple Aluminium Rod values to work, while every other Aluminium configuration failed, I decided to increase the factor of safety to 1.5x, for a slightly more realistic data selection process, and to fully eliminate all of the Aluminium Rod values.

The code is able to display 3 graphs, 2 of which are 3D graphs. The general shape of the 3D graphs was compared to published results/graphs in order to ensure that the calculations performed by the code are being done so correctly.

This code is performing correct calculations, as a check was done by hand calculating results and cross referencing them to the code's calculations. The results also make ssense, as aluminium is not usually used in the aerospace industry due to strength limitations (as seen in the code). A comment which can be made, is that technically the correct solution for the rods might be Aluminium, even though it is a bit counterintuitive. However, the counterintuitiveness is the main reason behind the factor of safety being added into this code.

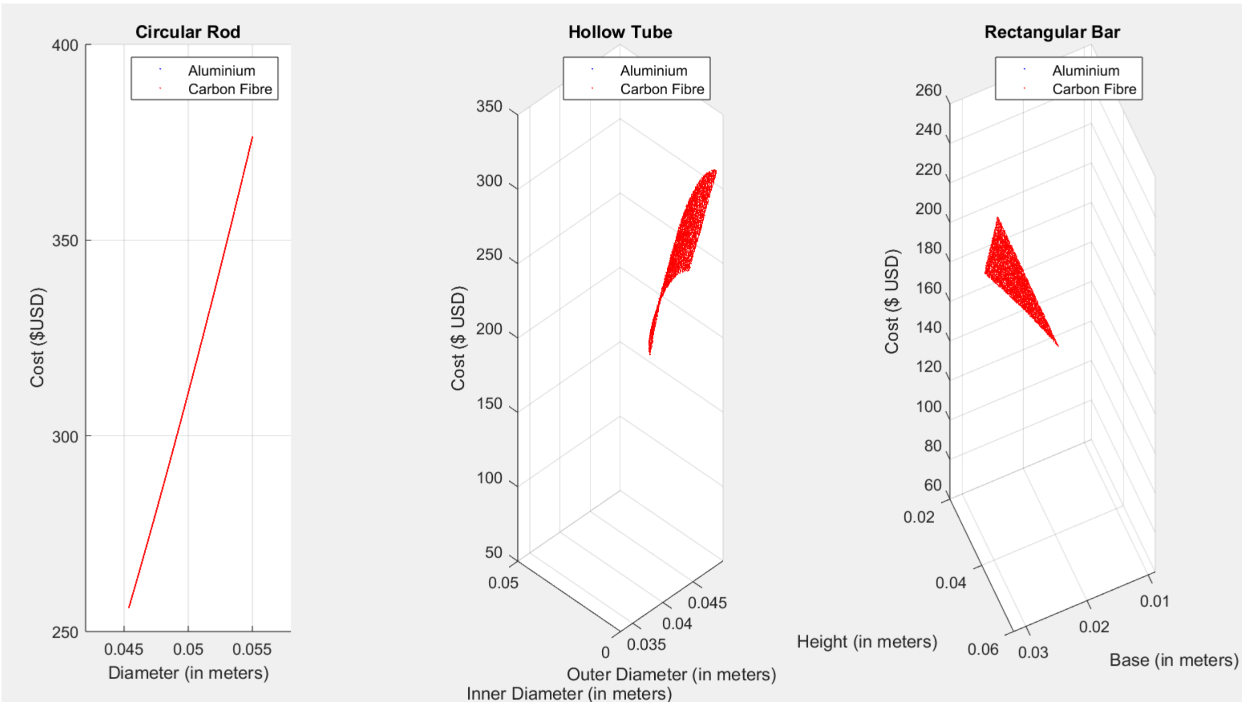

Figure 1: Output Graphs

Conclusion

In conclusion, a code was made which is able to sort through all the possible configurations of sports that would fit within the airfoil. The code was then able to sort through the possible configurations based on strength and deflection. Finally, all of the information got sorted, and the lowest prices and their respective dimensions were listed on the matlab code.

Appendix

```
clear all
clc
close all
%% VARIABLES %%
x = 0.1; %1/4 of chord length in m
L = 1.8; %spar length in m
dens_Al = 2700; %density in kg/m^3
dens_CF = 1600; %density in kg/m^3
E_Al = 69000000000; %youngs modulus in Pa
E_CF = 134000000000; %youngs modulus in Pa
cost_Al = 2.53; %$usd per kg
cost_CF = 55; %$usd per kg
max_defl = 0.05 * L; %max allowable deflection
mind = 0.001; %estimate based off graph
maxd = 0.055; %estimate based off graph
nmax = 200000; %number of iterations (more = more accurate results)
FS = 1.5; %factor of safety
i = 0; %count start at 0
%% Initialize Arrays %%
%rodstuff
dvalues = linspace(mind, maxd, nmax); %space out iterations
rodarrcostAl = zeros(1,nmax); %array for rod cost Al
rodarrcostCF = zeros(1,nmax); %array for rod cost CF
rodarrstrAl = zeros(1,nmax); %array for Al strength
rodarrstrCF = zeros(1,nmax); %array for CF strength
rodarrdvalsAl = zeros(1,nmax); %array for Al rod diameter values
rodarrdvalsCF = zeros(1,nmax); %array for CF rod diameter values
%pipe stuff
maxdout = 0.055; %max pipe outer diameter value
mindout = 0.001; %min pipe outer diameter value
maxdin = 0.053; %max pipe inner diameter value
mindin = 0.003; %min pipe inner diameter value
pipeinnervals = zeros(1,nmax); %array for pipe inner diamter values
pipeoutervals = zeros(1,nmax); %array for pipe outer diameter values
pipearrcostAl = zeros (1,nmax); %array for pipe cost Al
pipearrcostCF = zeros (1,nmax); %array for pipe cost CF
pipearrdoutAl = zeros (1,nmax); %array for pipe outer diameter Al
pipearrdinAl = zeros (1,nmax); %array for pipe inner diameter Al
pipearrdoutCF = zeros (1,nmax); %array for pipe outer diameter CF
pipearrdinCF = zeros (1,nmax); %array for pipe inner diameter CF
%rectangular stuff
maxb = 0.032; %max base value (4/10) cuz scaled from given pic in doc
minb = 0.001; %minimum b value
maxh = 0.05; %max height value (4/10) cuz scaled from given pic in doc
minh = 0.001; %minimum h value
```

```
rectarrcostAl = zeros (1,nmax); %array for rectangular cost Al
rectarrcostCF = zeros (1,nmax); %array for rectangular cost CF
rectbvalAl = zeros(1,nmax); %array for base values Al
recthvalAl = zeros(1,nmax); %array for height values Al
rectbvalCF = zeros(1,nmax); %array for base values CF
recthvalCF = zeros(1,nmax); %array for height values CF
%% Sorting by Yeild Strength and Deflection %%
%Rod
for d = dvalues %keeps running until goes through all iterations possible with
the conditions
i = i + 1; %count
mass_Al = dens_Al * pi * (d/2)^2 * L; %mass of aluminium rod
mass_CF = dens_CF * pi * (d/2)^2 * L; %mass of CF rod
price_Al = mass_Al * cost_Al; %total $ price for current Al config
price_CF = mass_CF * cost_CF; %total $ price for current CF config
I_rod = (pi * d^4) / 64; %have to make mass update in loop
   if (445 * (L^3)) / (3 * E_Al * I_rod) <= (max_defl) && (445 * L * d/2) /
I_rod <= (55000000 / FS) %filter by deflection and yeild strength + factor of
safety (1.5)
   rodarrcostAl(i) = price_Al; %record acceptable price values
   rodarrdvalsAl(i) = d; %record acceptable d values
   end
   if (445 * (L^3)) / (3 * E_CF * I_rod) <= (max_defl) && (445 * L * d/2) /
I_rod <= (131000000 / FS) %filter by deflection and yeild strength + factor of
safety (1.5)
   rodarrcostCF(i) = price_CF; %record acceptable price values
   rodarrdvalsCF(i) = d; %record acceptable d values
   end
end
for  n = 1:nmax %number of iterations
   d_outer = mindout + (maxdout - mindout) * rand(); %get random outer diameter
within range
   d_inner = mindin + (maxdin - mindin) * rand(); %get random inner diameter
within range
   I_hrod = (pi * (d_outer^4 - d_inner^4))/64; %area moment of inertia for
hollow rod
   if d_outer - d_inner >= 0.004 %min thickness of 0.002m
       if ((445 * (L^3)) / (3 * E_Al * I_hrod)) <= (max_defl) && (445 * L *
(d_outer/2) / I_hrod) <= (55000000 / FS) %filter by deflection and yeild
strength + factor of safety (1.5)
       pipearrcostAl(n) = ((dens_Al * pi * (d_outer/2)^2 * L) - (dens_Al * pi *
(d_inner/2)^2 * L)) * cost_Al; %record acceptable values
       pipearrdoutAl(n) = d_outer; %outer diameter array for Al
       pipearrdinAl(n) = d_inner; %inner diameter array for Al
       end
       if ((445 * (L^3)) / (3 * E_CF * I_hrod)) <= (max_defl) && (445 * L *
(d_outer/2) / I_hrod) <= (131000000 / FS) %filter by deflection and yeild
strength + Factor of safety (1.5)
```

```
        pipearrcostCF(n) = ((dens_CF * pi * (d_outer/2)^2 * L) - (dens_CF * pi *
(d_inner/2)^2 * L)) * cost_CF; %record acceptable values
        pipearrdoutCF(n) = d_outer; %outer diameter array for CF
        pipearrdinCF(n) = d_inner; %inner diameter array for CF
        end
    end
end
%Rectangular Rod
for  n = 1:nmax %number of iterations
   b = minb + (maxb - minb) * rand(); %get random base within range
   h = minh + (maxh - minh) * rand(); %get random height within range
   I_rect = (b * (h^3)) / 12;
   if (445 * (L^3)) / (3 * E_Al * I_rect) <= (max_defl) && (445 * L * h/2) /
I_rect <= (55000000 / FS) %filter by yeild strength and defleciton + factor of
safety (1.5)
   rectarrcostAl(n) = b * h * L * dens_Al * cost_Al; %store cost in array
   rectbvalAl(n) = b; %array for base values Al
   recthvalAl(n) = h; %array for height values Al
   end

   if (445 * (L^3)) / (3 * E_CF * I_rect) <= (max_defl) && (445 * L * h/2) /
I_rect <= (131000000 / FS) %filter by yeild strength and deflection + factor of
safety (1.5)
   rectarrcostCF(n) = b * h * L * dens_CF * cost_CF; %store cost in array
   rectbvalCF(n) = b; %array for base values CF
   recthvalCF(n) = h; %array for height values CF
   end
end
%% Graph Results %%
figure;
%plot for rod stuff
subplot(1,3,1) %so all on same figure
hold on; %to show both on same graph
plot(rodarrdvalsAl, rodarrcostAl, 'bo', 'MarkerSize', 0.5); %blue for Al
plot(rodarrdvalsCF, rodarrcostCF, 'ro', 'MarkerSize', 0.5); %red for CF
hold off;
ylim([250, 400]);
xlim([0.042, 0.058]);
xlabel('Diameter (in meters)'); %X-axis label
ylabel('Cost ($USD)'); %y-axis label
title('Circular Rod'); %Title
legend('Aluminium', 'Carbon Fibre'); %Legend
grid on; %grid for reference
view(2);
%plot for pipe stuff
subplot(1,3,2) %so all on same figure
hold on; %so both Al and CF on same graph
plot3(pipearrdoutAl, pipearrdinAl, pipearrcostAl, 'bo', 'MarkerSize', 0.5);
%blue for Al
```

```
plot3(pipearrdoutCF, pipearrdinCF, pipearrcostCF, 'ro', 'MarkerSize', 0.5);
%red for CF
hold off;
xlim([0.033 0.05]); %limit so no empty space
xlabel('Outer Diameter (in meters)'); %X-axis label
ylabel('Inner Diameter (in meters)'); %Y-axis label
zlabel('Cost ($ USD)'); %Z-axis label
title('Hollow Tube'); %Title
legend('Aluminium', 'Carbon Fibre'); %Legend
grid on; %Grid for reference
view(3); %3D graph
%plot for rectangle stuff
subplot(1,3,3) %So all on same figure
hold on; %So both Al and CF on same graph
plot3(rectbvalAl, recthvalAl, rectarrcostAl, 'bo', 'MarkerSize', 0.5); %blue
for Al
plot3(rectbvalCF, recthvalCF, rectarrcostCF, 'ro', 'MarkerSize', 0.5); %Red for
CF
hold off;
ylim([0.02 0.06]); %Y-limit so not much empty space
xlabel('Base (in meters)'); %X-axis label
ylabel('Height (in meters)'); %Y-axis label
zlabel('Cost ($ USD)'); %Z-axis label
title('Rectangular Bar'); %Title
legend('Aluminium', 'Carbon Fibre'); %legend
grid on; %Grid for reference
view(3); %3D graph
%% Filter and Write Out Results %%
%findLowestNonzero function
findLowestNonzero = @(arr) struct('value', min(arr(arr > 0)), 'index', find(arr
== min(arr(arr > 0)), 1, 'first'));
% Rods
valid_rod_Al = rodarrcostAl > 0; %make sure Al rod cost is above 0 (valid
configuration)
if any(valid_rod_Al) %if valid config find cost and d
   [min_val_Al, min_idx_Al] = min(rodarrcostAl(valid_rod_Al));
   lowest_rod_Al = struct('value', min_val_Al, 'index',
find(cumsum(valid_rod_Al) == min_idx_Al, 1)); %store cost
   lowest_rod_d_Al = rodarrdvalsAl(lowest_rod_Al.index); %store d based of cost
index
else
   lowest_rod_Al = struct('value', NaN, 'index', NaN); %disregard irrelevant
configurations
   lowest_rod_d_Al = NaN;
end
valid_rod_CF = rodarrcostCF > 0; %make sure CF rod cost is above 0 (valid
configuration)
if any(valid_rod_CF) %if valid config find cost and d
   [min_val_CF, min_idx_CF] = min(rodarrcostCF(valid_rod_CF));
```

```
    lowest_rod_CF = struct('value', min_val_CF, 'index',
find(cumsum(valid_rod_CF) == min_idx_CF, 1)); %store cost
    lowest_rod_d_CF = rodarrdvalsCF(lowest_rod_CF.index); %store d based of cost
index
else
    lowest_rod_CF = struct('value', NaN, 'index', NaN); %disregard irrelevant
configurations
    lowest_rod_d_CF = NaN;
end
% For pipes
valid_pipe_Al = pipearrcostAl > 0; %make sure Al pipe cost above 0 (valid
config)
if any(valid_pipe_Al) %if valid config find infos
    [min_val_Al, min_idx_Al] = min(pipearrcostAl(valid_pipe_Al));
    lowest_pipe_Al = struct('value', min_val_Al, 'index',
find(cumsum(valid_pipe_Al) == min_idx_Al, 1)); %store cost
    lowest_pipe_out_d_Al = pipearrdoutAl(lowest_pipe_Al.index); %store outer d
based off cost index
    lowest_pipe_in_d_Al = pipearrdinAl(lowest_pipe_Al.index); %store inner d
based off cost index
else
    lowest_pipe_Al = struct('value', NaN, 'index', NaN); %disregard irrelevant
configurations
    lowest_pipe_out_d_Al = NaN;
    lowest_pipe_in_d_Al = NaN;
end
valid_pipe_CF = pipearrcostCF > 0; %check that CF rod cost above 0 (valid
config)
if any(valid_pipe_CF) %if valid config, store infos
    [min_val_CF, min_idx_CF] = min(pipearrcostCF(valid_pipe_CF));
    lowest_pipe_CF = struct('value', min_val_CF, 'index',
find(cumsum(valid_pipe_CF) == min_idx_CF, 1)); %store cost
    lowest_pipe_out_d_CF = pipearrdoutCF(lowest_pipe_CF.index); %store outer
diameter based off cost index
    lowest_pipe_in_d_CF = pipearrdinCF(lowest_pipe_CF.index); %store inner
diameter based off cost index
else
    lowest_pipe_CF = struct('value', NaN, 'index', NaN); %disregard irrelevant
configurations
    lowest_pipe_out_d_CF = NaN;
    lowest_pipe_in_d_CF = NaN;
end
% For rectangles
valid_rect_Al = rectarrcostAl > 0; %check that Al rectangle cost above 0 (valid
config)
if any(valid_rect_Al) %if valid config, store infos
    [min_val_Al, min_idx_Al] = min(rectarrcostAl(valid_rect_Al));
    lowest_rect_Al = struct('value', min_val_Al, 'index',
find(cumsum(valid_rect_Al) == min_idx_Al, 1)); %store cost
```

```
   lowest_rect_b_Al = rectbvalAl(lowest_rect_Al.index); %store b based off cost
index
   lowest_rect_h_Al = recthvalAl(lowest_rect_Al.index); %store h ased off cost
index
else
   lowest_rect_Al = struct('value', NaN, 'index', NaN); %disregard irrelevant
configs
   lowest_rect_b_Al = NaN;
   lowest_rect_h_Al = NaN;
end
valid_rect_CF = rectarrcostCF > 0; %check that CF rectangle cost above 0 (valid
config)
if any(valid_rect_CF) %if valid config store infos
   [min_val_CF, min_idx_CF] = min(rectarrcostCF(valid_rect_CF));
   lowest_rect_CF = struct('value', min_val_CF, 'index',
find(cumsum(valid_rect_CF) == min_idx_CF, 1)); %store cost
   lowest_rect_b_CF = rectbvalCF(lowest_rect_CF.index); %store b based off cost
index
   lowest_rect_h_CF = recthvalCF(lowest_rect_CF.index); %store h based off cost
index
else
   lowest_rect_CF = struct('value', NaN, 'index', NaN); %disregard irrelevant
values
   lowest_rect_b_CF = NaN;
   lowest_rect_h_CF = NaN;
end
fprintf('\n---- Optimal Configurations ----\n\n'); %title block
% Rods
if ~isnan(lowest_rod_Al.value) %check if valid config exists, if does print the
following
   fprintf('ALUMINIUM ROD\n');
   fprintf('  Cost:           $%.2f\n', lowest_rod_Al.value);
   fprintf('  Mass:           %.5f kg\n', lowest_rod_Al.value/cost_Al);
   fprintf('  Diameter:       %.5f m\n\n', lowest_rod_d_Al);
else
   fprintf('No valid Aluminium Rod configurations found\n\n'); %if no valid
config exists print this
end
if ~isnan(lowest_rod_CF.value) %check if valid config exists, if does print the
following
   fprintf('CARBON FIBER ROD\n');
   fprintf('  Cost:           $%.2f\n', lowest_rod_CF.value);
   fprintf('  Mass:           %.5f kg\n', lowest_rod_CF.value/cost_CF);
   fprintf('  Diameter:       %.5f m\n\n', lowest_rod_d_CF);
else
   fprintf('No valid Carbon Fiber Rod configurations found\n\n'); %if no valid
config exists print this
end
% Pipes
```

```
if ~isnan(lowest_pipe_Al.value) %check if valid config exists, if does print
the following
   fprintf('ALUMINIUM PIPE\n');
   fprintf('  Cost:            $%.2f\n', lowest_pipe_Al.value);
   fprintf('  Mass:            %.5f kg\n', lowest_pipe_Al.value/cost_Al);
   fprintf('  Outer Diameter: %.5f m\n', lowest_pipe_out_d_Al);
   fprintf('  Inner Diameter: %.5f m\n\n', lowest_pipe_in_d_Al);
else
   fprintf('No valid Aluminium Pipe configurations found\n\n'); %if no valid
config exists print this
end
if ~isnan(lowest_pipe_CF.value) %check if valid config exists, if does print
the following
   fprintf('CARBON FIBER PIPE \n');
   fprintf('  Cost:            $%.2f\n', lowest_pipe_CF.value);
   fprintf('  Mass:            %.5f kg\n', lowest_pipe_CF.value/cost_CF);
   fprintf('  Outer Diameter: %.5f m\n', lowest_pipe_out_d_CF);
   fprintf('  Inner Diameter: %.5f m\n\n', lowest_pipe_in_d_CF);
else
   fprintf('No valid Carbon Fiber Pipe configurations found\n\n'); %if no valid
config exists print this
end
% Rectangles
if ~isnan(lowest_rect_Al.value) %check if valid config exist, if does print the
following:
   fprintf('ALUMINIUM RECTANGLE\n');
   fprintf('  Cost:            $%.2f\n', lowest_rect_Al.value);
   fprintf('  Mass:            %.5f kg\n', lowest_rect_Al.value/cost_Al);
   fprintf('  Base:            %.5f m\n', lowest_rect_b_Al);
   fprintf('  Height:          %.5f m\n\n', lowest_rect_h_Al);
else
   fprintf('No valid Aluminium Rectangle configurations found\n\n'); %if no
valid config exist print this
end
if ~isnan(lowest_rect_CF.value) %check if valid config exist, if does print the
following:
   fprintf('CARBON FIBER RECTANGLE\n');
   fprintf('  Cost:            $%.2f\n', lowest_rect_CF.value);
   fprintf('  Mass:            %.5f kg\n', lowest_rect_CF.value/cost_CF);
   fprintf('  Base:            %.5f m\n', lowest_rect_b_CF);
   fprintf('  Height:          %.5f m\n\n', lowest_rect_h_CF);
else
   fprintf('No valid Carbon Fiber Rectangle configurations found\n\n'); %if no
valid config exist print this
end
```