Table of Contents


```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Sam Alvares, Ian Landwehr, Sam Ridgely
% ME480-03: Machine Component Design
% Dr. Constans
%
% Project IV: Fatigue Analysis of Input Shaft
% Due: April 23th, 2021
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Given from Problem Statement

```
clc; clear variables; close all;
% Given Measurements            % Units
x1 = 1.5625;                    % in
x2 = 2.375;                     % in
x3 = 3.125;                     % in
x4 = 0.6875;                    % in
L_input = 2.375;                % in
N = L_input/.0001+1;            % Determine # points so that points at
 at 0.001 increments
Sut = 148000;                   % psi
```

Basic

Results of statics analysis (from Maple)

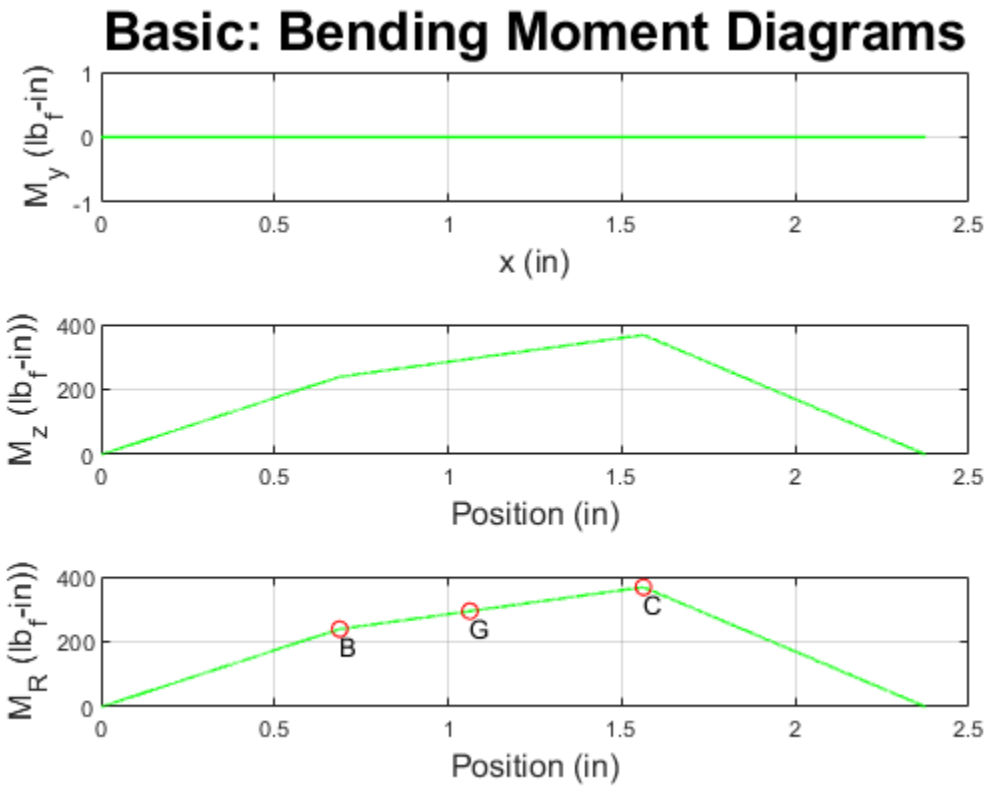
```
Ay = 0;
Az = 6600/19;
Dy = 0;
Dz = 8600/19;
Ey = 22500/19;
Ez = -7500/19;
GR2 = 0;
GR4 = 0;
GT2 = 200;
GT4 = 600;
Hy = -5400/19;
Hz = -3900/19;
```

```
% Find Bending Momemnt: y-direction
[x,My,Mz] = deal(zeros(N,1));
for i = 1:N
  x(i) = (i-1)*L_input/(N-1);
  My(i) = Ay*heaviside(x(i))*x(i)+GR2*heaviside(x(i)-x4)*(x(i)-x4)...
          -GR4*heaviside(x(i)-x1)*(x(i)-x1)+Dy*heaviside(x(i)-
x2)*(x(i)-x2);
end

% z-direction
for i = 1:N
  Mz(i) = Az*heaviside(x(i))*x(i)-GT2*heaviside(x(i)-x4)*(x(i)-x4)...
          -GT4*heaviside(x(i)-x1)*(x(i)-x1)+Dz*heaviside(x(i)-
x2)*(x(i)-x2);
end

% Resultant
MR = sqrt(My.^2+Mz.^2);

% Find resultant moment at key locations
xb = x4;
xG = 1.0625;
xc = x1;
for i = 1:length(x)
    xCheck = x(i);
    if xCheck == xb
        Ib = i;
    elseif xCheck == xG
        IG = i;
    elseif xCheck == xc
        Ic = i;
    end
end

Mb = MR(Ib);
MG = MR(IG);
Mc = MR(Ic);

% Create bending moment diagrams as subplot, label points b,G,c
figure(1)
subplot(3,1,1), plot(x,My,'g','LineWidth',1), grid on
title('Basic: Bending Moment Diagrams','FontSize',20)
ylabel('M_{y} (lb_{f}-in)','FontSize',12)
xlabel('x (in)','FontSize',12)
subplot(3,1,2), plot(x,Mz,'g','LineWidth',1), grid on
ylabel('M_{z} (lb_{f}-in))','FontSize',12)
xlabel('Position (in)','FontSize',12)
subplot(3,1,3), plot(x,MR,'g','LineWidth',1), grid on
ylabel('M_{R} (lb_{f}-in))','FontSize',12)
xlabel('Position (in)','FontSize',12)
hold on
plot(xb,Mb,'ro')
label = 'B';
text(xb,Mb,label,'VerticalAlignment','top','HorizontalAlignment','left')
```

```
plot(xG,MG,'ro')
label = 'G';
text(xG,MG,label,'VerticalAlignment','top','HorizontalAlignment','left')
plot(xc,Mc,'ro')
label = 'C';
text(xc,Mc,label,'VerticalAlignment','top','HorizontalAlignment','left')

% Run optimization problem
x0 = [7/16;5/8];
A = [-1  0;
      0 -1
      1 -1];
b = [0;0;-1/16];

xSol = fmincon(@funObj,x0,A,b,[],[],[],
[],@(xVec)funcNL(xVec,Mb,MG,Mc));

% Solve for results with optimized value - rounded up to 1/16
 increment
DB = ceil(xSol(1) * 16) / 16;
DC = ceil(xSol(2) * 16) / 16;
Cload = 1;
CsizeB = 0.869*DB^(-0.097);
CsizeG = CsizeB;
CsizeC = 0.869*DC^(-0.097);
Csurf = 2.7*148^-.265;
Ctemp = 1;
Creliab = 0.814;
Sep = 0.5*Sut;
SeB = Cload*CsizeB*Csurf*Ctemp*Creliab*Sep;
SeG = Cload*CsizeG*Csurf*Ctemp*Creliab*Sep;
SeC = Cload*CsizeC*Csurf*Ctemp*Creliab*Sep;

% find fatigue stress conectration factors at G
[Kt_bending,Kt_torsion] = interpFatigue(DC,DB);
q = 0.862;
Kf_bending = 1+q*(Kt_bending-1);
Kf_torsion = 1+q*(Kt_torsion-1);
sigma_G = Kf_bending*((32*MG)/(pi*DB^3));
T = 600;
tau_torsion_G = Kf_torsion*((16*T)/(pi*DB^3));
sigma_m_G = sqrt(3)*tau_torsion_G;
sigma_m_B = 0;
sigma_m_C = (16*T*sqrt(3))/(pi*DC^3);
sigma_B = (32*Mb)/(pi*DB^3);
sigma_C = (32*Mc)/(pi*DC^3);

FOSB = 1/(sigma_m_B/Sut + sigma_B/SeB);
FOSG = 1/(sigma_m_G/Sut + sigma_G/SeG);
FOSC = 1/(sigma_m_C/Sut + sigma_C/SeC);

fprintf('\n--------------- BASIC ---------------')
fprintf('\nOptimzation Solution: DB = %1.4f, DC = %1.4f
\n',xSol(1),xSol(2))
```

```
fprintf('Round up to nearest 1/16": DB = %1.4f, DC = %1.4f\n',DB,DC)
fprintf('FOSB = %1.4f\nFOSG = %1.4f\nFOSC = %1.4f\n',FOSB,FOSG,FOSC)
```

```
Local minimum found that satisfies the constraints.

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


--------------- BASIC ---------------
Optimzation Solution: DB = 0.6320, DC = 0.6945
Round up to nearest 1/16": DB = 0.6875, DC = 0.7500
FOSB = 5.2079
FOSG = 1.9098
FOSC = 3.1800
```



Medium

Results of statics analysis (from Maple)

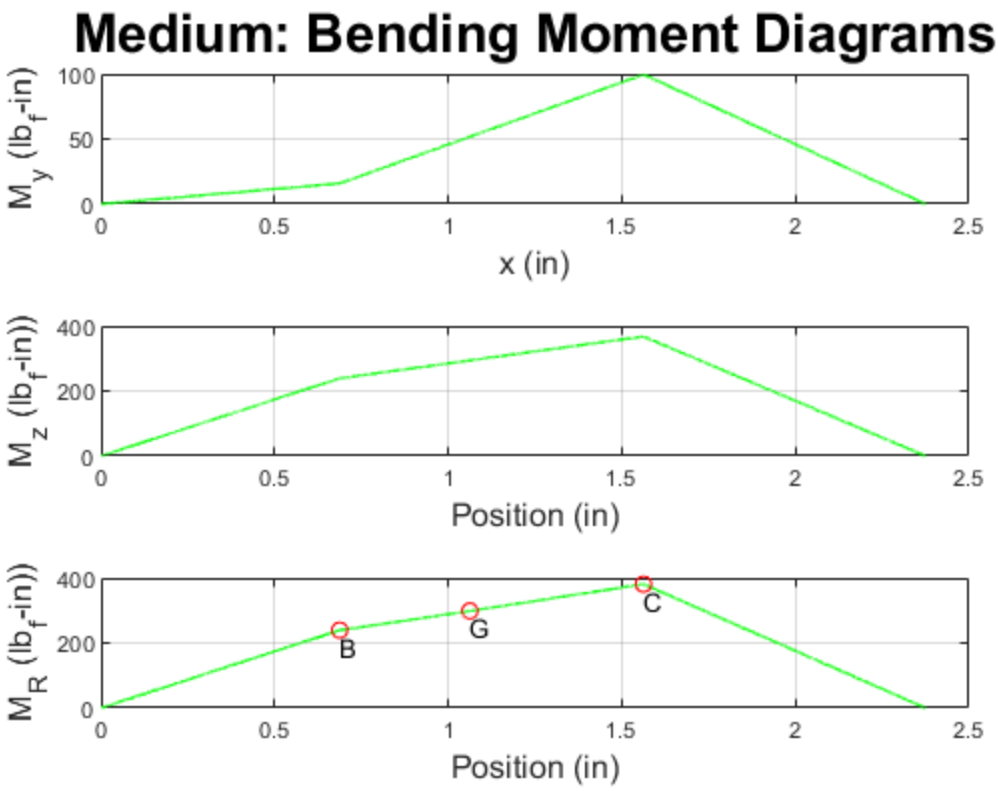
```
Ay = 22.98759375;
Az = 347.3684211;
```

```
Dy = 122.6005000;
Dz = 452.6315789;
Ey = 1040.538065;
Ez = -394.7368421;
GR2 = 72.79404686;
GR4 = 218.3821406;
GT2 = 200.;
GT4 = 600.;
Hy = -358.9202060;
Hz = -205.2631579;

% Find Bending Momemnt: y-direction
[x,My,Mz] = deal(zeros(N,1));
for i = 1:N
  x(i) = (i-1)*L_input/(N-1);
  My(i) = Ay*heaviside(x(i))*x(i)+GR2*heaviside(x(i)-x4)*(x(i)-x4)...
          -GR4*heaviside(x(i)-x1)*(x(i)-x1)+Dy*heaviside(x(i)-
x2)*(x(i)-x2);
end

% z-direction
for i = 1:N
  Mz(i) = Az*heaviside(x(i))*x(i)-GT2*heaviside(x(i)-x4)*(x(i)-x4)...
          -GT4*heaviside(x(i)-x1)*(x(i)-x1)+Dz*heaviside(x(i)-
x2)*(x(i)-x2);
end

% Resultant
MR = sqrt(My.^2+Mz.^2);

% Find resultant moment at key locations
xb = x4;
xG = 1.0625;
xc = x1;
for i = 1:length(x)
    xCheck = x(i);
    if xCheck == xb
        Ib = i;
    elseif xCheck == xG
        IG = i;
    elseif xCheck == xc
        Ic = i;
    end
end

Mb = MR(Ib);
MG = MR(IG);
Mc = MR(Ic);

% Create bending moment diagrams as subplot, label points b,G,c
figure(2)
subplot(3,1,1), plot(x,My,'g','LineWidth',1), grid on
title('Medium: Bending Moment Diagrams','FontSize',20)
ylabel('M_{y} (lb_{f}-in)','FontSize',12)
```

```
xlabel('x (in)','FontSize',12)
subplot(3,1,2), plot(x,Mz,'g','LineWidth',1), grid on
ylabel('M_{z} (lb_{f}-in))','FontSize',12)
xlabel('Position (in)','FontSize',12)
subplot(3,1,3), plot(x,MR,'g','LineWidth',1), grid on
ylabel('M_{R} (lb_{f}-in))','FontSize',12)
xlabel('Position (in)','FontSize',12)
hold on
plot(xb,Mb,'ro')
label = 'B';
text(xb,Mb,label,'VerticalAlignment','top','HorizontalAlignment','left')
plot(xG,MG,'ro')
label = 'G';
text(xG,MG,label,'VerticalAlignment','top','HorizontalAlignment','left')
plot(xc,Mc,'ro')
label = 'C';
text(xc,Mc,label,'VerticalAlignment','top','HorizontalAlignment','left')

% Run optimization problem
x0 = [7/16;5/8];
A = [-1  0;
      0 -1
      1 -1];
b = [0;0;-1/16];

xSol = fmincon(@funObj,x0,A,b,[],[],[],
[],@(xVec)funcNL(xVec,Mb,MG,Mc));

% Solve for results with optimized value - rounded up to 1/16
 increment
DB = ceil(xSol(1) * 16) / 16;
DC = ceil(xSol(2) * 16) / 16;
Cload = 1;
CsizeB = 0.869*DB^(-0.097);
CsizeG = CsizeB;
CsizeC = 0.869*DC^(-0.097);
Csurf = 2.7*148^-.265;
Ctemp = 1;
Creliab = 0.814;
Sep = 0.5*Sut;
SeB = Cload*CsizeB*Csurf*Ctemp*Creliab*Sep;
SeG = Cload*CsizeG*Csurf*Ctemp*Creliab*Sep;
SeC = Cload*CsizeC*Csurf*Ctemp*Creliab*Sep;

% find fatigue stress conectration factors at G
[Kt_bending,Kt_torsion] = interpFatigue(DC,DB);
q = 0.862;
Kf_bending = 1+q*(Kt_bending-1);
Kf_torsion = 1+q*(Kt_torsion-1);
sigma_G = Kf_bending*((32*MG)/(pi*DB^3));
T = 600;
tau_torsion_G = Kf_torsion*((16*T)/(pi*DB^3));
sigma_m_G = sqrt(3)*tau_torsion_G;
sigma_m_B = 0;
```

```
sigma_m_C = (16*T*sqrt(3))/(pi*DC^3);
sigma_B = (32*Mb)/(pi*DB^3);
sigma_C = (32*Mc)/(pi*DC^3);

FOSB = 1/(sigma_m_B/Sut + sigma_B/SeB);
FOSG = 1/(sigma_m_G/Sut + sigma_G/SeG);
FOSC = 1/(sigma_m_C/Sut + sigma_C/SeC);

fprintf('\n--------------- MEDIUM ---------------')
fprintf('\nOptimzation Solution: DB = %1.4f, DC = %1.4f
\n',xSol(1),xSol(2))
fprintf('Round up to nearest 1/16": DB = %1.4f, DC = %1.4f\n',DB,DC)
fprintf('FOSB = %1.4f\nFOSG = %1.4f\nFOSC = %1.4f\n',FOSB,FOSG,FOSC)
```

```
Local minimum found that satisfies the constraints.

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


--------------- MEDIUM ---------------
Optimzation Solution: DB = 0.6345, DC = 0.6970
Round up to nearest 1/16": DB = 0.6875, DC = 0.7500
FOSB = 5.1965
FOSG = 1.8882
FOSC = 3.0985
```



Published with MATLAB® R2020a