

```

%%%%%%%%%
%TA: Raymond Ma
%Group Members: Grace, Yina, Joaquin

%% Intial Parameters for both designs and cases:
n = 1250; % Number of locations to evaluate bridge failure
L = 1250; % Length of bridge

x = linspace(1,L,n); % Define x coordinate

locationA = 1;
locationB = 1060;

%Material properties
SigT = 30;
SigC = 6;
E = 4000;
TauU = 4;
TauG = 2;
mu = 0.2;

%% Design Zero: Train
SFD_TrainLoad = zeros(1, n); % Initialize SFD(x)
BMD_TrainLoad = zeros(1, n); % Initialize BMD
SFD_TrainLoad1 = zeros(1, n); % Initialize SFD(x)
BMD_TrainLoad1 = zeros(1, n); % Initialize BMD
SFD_TrainLoad2 = zeros(1, n); % Initialize SFD(x)
BMD_TrainLoad2 = zeros(1, n); % Initialize BMD
SFD_TrainLoad3 = zeros(1, n); % Initialize SFD(x)
BMD_TrainLoad3 = zeros(1, n); % Initialize BMD
P = 400/6;

%diaphragms for Design 0
a = zeros(1,length(x));
a(1:550) = 550;
a(551:1060) = 510;
a(1061:1250) = 190;

%Cross Section
%% 2. Define cross-sections
% Design 0
dx = n/L; %length between each x coordinate
xc = [1 dx*L]; % x locations of changes in cross section
bVector = [100 1.27 1.27 10 10 80]; %Base of rectangle segments
hVector = [1.27 72.46 72.46 1.27 1.27 1.27]; %height of rectangle segments
yVector = [74.365 37.5 37.5 73.1 73.1 0.635]; % distances between local centroid of sub

I = zeros(1, length(x)); %Initialize I values
yBar = zeros(1, length(x)); %Initialize Y values
[I, yBar] = SectionProperties( I, yBar, xc(1), xc(2), bVector, hVector, yVector); %Calc

cutoffAreaCent = [ 77.46*1.27, 1.27*yBar(1), 1.27*yBar(1)]; %Calculate area from centroid

```

```

yCentCutoff = [ yBar(1)-0.635, yBar(1)/2, yBar(1)/2]; %Calculate y values from centroid

cutoffAreaGlue = [127];
yGlue = [75-yBar(1)-0.635]; % centroid of the cutoff area

Qglue = zeros(1, length(x)); %initialize Q at centriod and Q at glue
Qmax = zeros(1, length(x));
[Qmax, Qglue] = calculateQ( Qmax, Qglue, xc(1), xc(2), cutoffAreaCent, cutoffAreaGlue,

%Other constants
b = zeros(1,length(x)); %Width of Shear surface
b(:) = 2*1.27;
bGlue = zeros(1,length(x)); %Width of Shear surface of glue
bGlue(:) = 10;
t = zeros(1,length(x)); %Thickness of board (used for plate buckling)
t(:) = 1.27;
height = zeros(1,length(x)); %Height of cross section
height(:) = 75;

%% Train Load: 6 points for the 6 wheels
TLocation = 1; %Train at Start
[SFD_TrainLoad, BMD_TrainLoad] = ApplyPL(TLocation, P, x, SFD_TrainLoad, locationA, loca
[SFD_TrainLoad, BMD_TrainLoad] = ApplyPL(TLocation + 176, P, x, SFD_TrainLoad, location
[SFD_TrainLoad, BMD_TrainLoad] = ApplyPL(TLocation + 176 + 164, P, x, SFD_TrainLoad, lo
[SFD_TrainLoad, BMD_TrainLoad] = ApplyPL(TLocation + 176 + 164 + 176, P, x, SFD_TrainLo
[SFD_TrainLoad, BMD_TrainLoad] = ApplyPL(TLocation + 176 + 164 + 176 + 164, P, x, SFD_T
[SFD_TrainLoad, BMD_TrainLoad] = ApplyPL(TLocation + 176 + 164 + 176 + 164 + 176, P, x,

TLocation = 196; %Train at Start
[SFD_TrainLoad2, BMD_TrainLoad2] = ApplyPL(TLocation, P, x, SFD_TrainLoad2, locationA,
[SFD_TrainLoad2, BMD_TrainLoad2] = ApplyPL(TLocation + 176, P, x, SFD_TrainLoad2, locat
[SFD_TrainLoad2, BMD_TrainLoad2] = ApplyPL(TLocation + 176 + 164, P, x, SFD_TrainLoad2,
[SFD_TrainLoad2, BMD_TrainLoad2] = ApplyPL(TLocation + 176 + 164 + 176, P, x, SFD_Train
[SFD_TrainLoad2, BMD_TrainLoad2] = ApplyPL(TLocation + 176 + 164 + 176 + 164, P, x, SFD
[SFD_TrainLoad2, BMD_TrainLoad2] = ApplyPL(TLocation + 176 + 164 + 176 + 164 + 176, P,

TLocation = 391; %Train at Start
[SFD_TrainLoad3, BMD_TrainLoad3] = ApplyPL(TLocation, P, x, SFD_TrainLoad3, locationA,
[SFD_TrainLoad3, BMD_TrainLoad3] = ApplyPL(TLocation + 176, P, x, SFD_TrainLoad3, locat
[SFD_TrainLoad3, BMD_TrainLoad3] = ApplyPL(TLocation + 176 + 164, P, x, SFD_TrainLoad3,
[SFD_TrainLoad3, BMD_TrainLoad3] = ApplyPL(TLocation + 176 + 164 + 176, P, x, SFD_Train
[SFD_TrainLoad3, BMD_TrainLoad3] = ApplyPL(TLocation + 176 + 164 + 176 + 164, P, x, SFD
[SFD_TrainLoad3, BMD_TrainLoad3] = ApplyPL(TLocation + 176 + 164 + 176 + 164 + 176, P,

%Find applied stresses
%Shear Stress demand
ShearStress = VStress(I, b, Qmax, SFD_TrainLoad); %Shear Stress at centroid
ShearStressGlue = VStress(I, 20 + 2*1.27, Qglue, SFD_TrainLoad); %Shear Stress at Glue

%Shear Stress Capacity
ShearBuckle = VBuck(I, t, b, height, E, mu, yBar, a, Qmax); %Buckling Shear capacity

%Bending Stress
%Bending Stress Demand

```

```

BendingStressT = MStressT( I,yBar, height, BMD_TrainLoad); %Tension Stress
BendingStressC = MStressC( I,yBar, height, BMD_TrainLoad); %Compression Stress

%Shear Stress Capacity %Bending Plate Buckling capacity stress
BendingBuck1 = MBuck( I, t, (100-80)/2, E, mu, 2);
BendingBuck2 = MBuck( I, t, 80-2*1.27, E, mu, 1 );
BendingBuck3 = MBuck( I, t, 75-41.701-1.27, E, mu, 3 );
BendingBuck4 = MBuck( I, t, (100-80)/2, E, mu, 2 );
BendingBuck5 = MBuck( I, t, 75-41.701-1.27, E, mu, 3 );
BendingBuck6 = MBuck( I, t, 41.701-1.27, E, mu, 3 );
BendingBuck7 = MBuck( I, t, 80-2*1.27, E, mu, 1 );
BendingBuck8 = MBuck( I, t, 41.701-1.27, E, mu, 1 );

%Calculate FOS
FOSBridge = FOSCalc(ShearStress, ShearStressGlue, BendingStressT, BendingStressC, SigT, SigC);

%Visualization

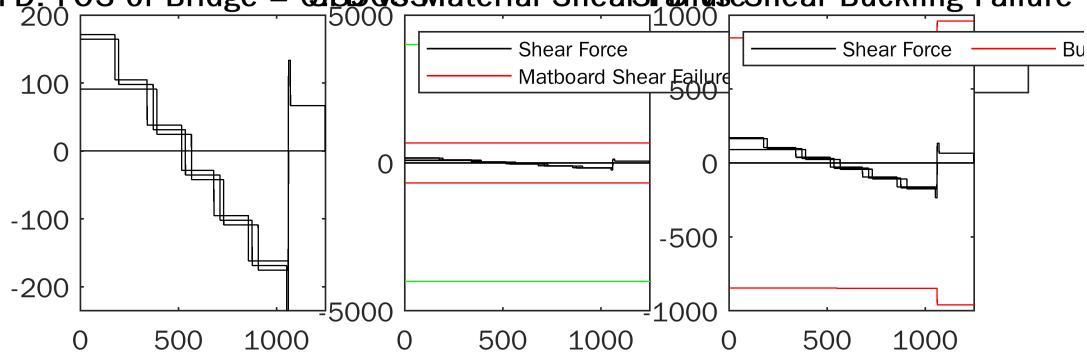
%Find max Shear Forces and Bending Moments
V_Mat = Vfail(I, b, Qmax, TauU); %Max Shear
V_Buck = VfailBuck( I, t, b, height, E, mu, yBar, a, TauU, Qmax); %Max Shear caused by buckling
V_Glue = Vfail(I, bGlue, Qglue, TauU); %Max Shear at glue

M_MatT = MfailMatT(I, yBar, height, SigT, BMD_TrainLoad); %Max tension
M_MatC = MfailMatC(I, yBar, height, SigC, BMD_TrainLoad); %Max compression
M_Buck1 = MfailBuck(I, t, (100-80)/2, E, mu, BMD_TrainLoad, 2, yBar, height, SigC); %Max Buckling
M_Buck2 = MfailBuck(I, t, 80-2*1.27, E, mu, BMD_TrainLoad, 1, yBar, height, SigC);
M_Buck3 = MfailBuck(I, t, 75-41.701-1.27, E, mu, BMD_TrainLoad, 3, yBar, height, SigC);
M_Buck4 = MfailBuck(I, t, (100-80)/2, E, mu, BMD_TrainLoad, 2, yBar, height, SigC);
M_Buck5 = MfailBuck(I, t, 75-41.701-1.27, E, mu, BMD_TrainLoad, 1, yBar, height, SigC);
M_Buck6 = MfailBuck(I, t, 41.701-1.27, E, mu, BMD_TrainLoad, 3, yBar, height, SigC);
M_Buck7 = MfailBuck(I, t, 80-2*1.27, E, mu, BMD_TrainLoad, 1, yBar, height, SigC);
M_Buck8 = MfailBuck(I, t, 41.701-1.27, E, mu, BMD_TrainLoad, 3, yBar, height, SigC);

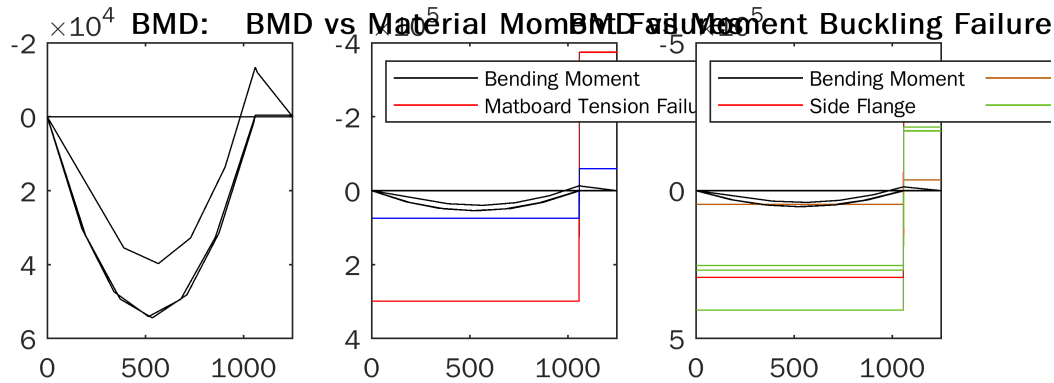
%% Visualization
VisualizeTL(x, SFD_TrainLoad, BMD_TrainLoad, SFD_TrainLoad2, BMD_TrainLoad2, SFD_TrainLoad2, BMD_TrainLoad2);

```

SFD: FOS of Bridge = 3.5083 Material Shear Failure vs Shear Buckling Failure



BMD: BMD vs Material Moment Buckling Failure



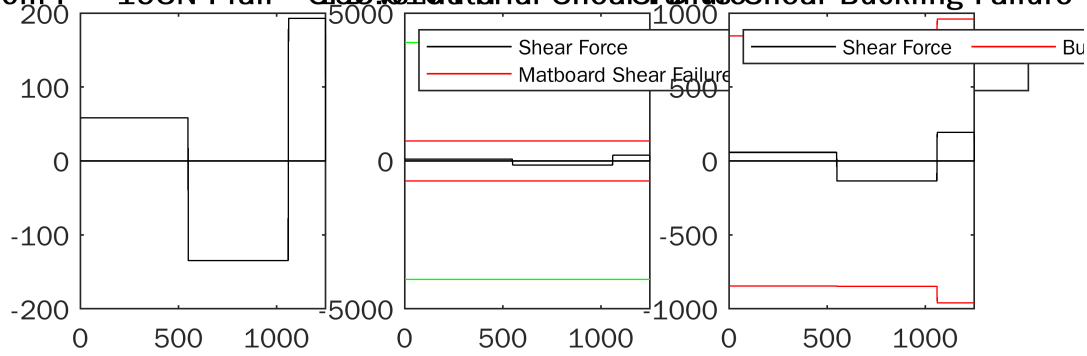
```
%Design 0 Point Loads
SFD_PL = zeros(1, n);           % Initialize SFD(x)
BMD_PL = zeros(1, n);
P = 1; %P = 1 * whatever value, so I can set p to 1

%% 1. Point Loading Analysis (SFD, BMD) Two loads for two points
[SFD_PL, BMD_PL] = ApplyPL(550, P, x, SFD_PL, locationA, locationB); % Construct SFD, BMD
[SFD_PL, BMD_PL] = ApplyPL(L, P, x, SFD_PL, locationA, locationB);   % Construct SFD, BMD

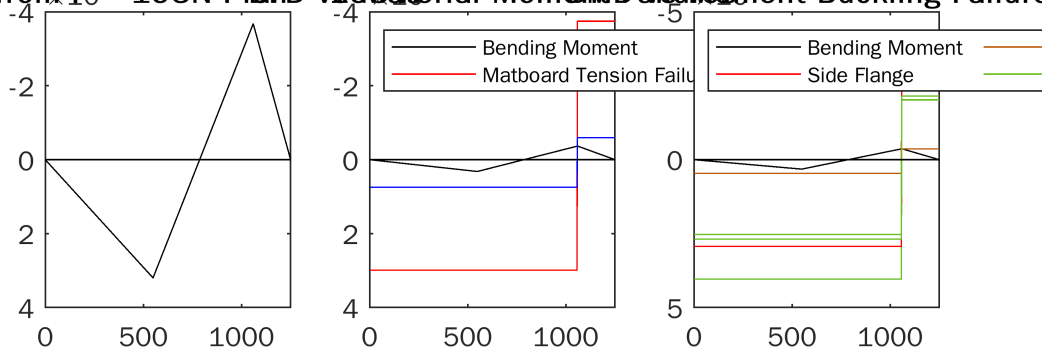
%%Calculate Failure Load
Pf = FailLoad( SFD_PL, BMD_PL, V_Mat, V_Glue, V_Buck, M_MatT, M_MatC, M_Buck1, M_Buck2, ...

%% Visualization
SFD_FAIL = zeros(1,length(x));
BMD_FAIL = zeros(1,length(x));
[SFD_FAIL, BMD_FAIL] = ApplyPL(550, Pf, x, SFD_FAIL, locationA, locationB); % Construct SFD, BMD
[SFD_FAIL, BMD_FAIL] = ApplyPL(L, Pf, x, SFD_FAIL, locationA, locationB);   % Construct SFD, BMD
VisualizePL(x, SFD_FAIL, BMD_FAIL, V_Mat, V_Glue, V_Buck, M_MatT, M_MatC, M_Buck1, M_Buck2, ...
```

FD from $P = 193\text{N}$ $P_{fail} = 993.6104\text{N}$ SFD vs SFD Material Shear Failure vs Shear Buckling Failure



BMD from $P = 193\text{N}$ $P_{fail} = 993.6104\text{N}$ BMD vs BMD Material Moment Failure vs Moment Buckling Failure



```
%Our Design
%Train:

%diaphragms
a = zeros(1,length(x)); %Our design has 3 different diaphragm spacings
a(1:550) = 104;
a(551:1060) = 120;
a(1060:1250) = 53;

%% 2. Define cross-sections
% Cross Section A
dx = n/L;
xc = [1 dx*350 dx*738 dx*838 dx*L]; % x locations of changes in cross section
I = zeros(1, length(x));
yBar = zeros(1, length(x));
Qglue = zeros(1, length(x));
Qmax = zeros(1, length(x));

%Cross Section A
bVector = [100 1.27 1.27 10 10];
hVector = [1.27 120 120 1.27 1.27];
```

```

yVector = [120.635 60 60 119.365 119.365]; % distances between local centroid of subcom
[I, yBar] = SectionProperties( I, yBar, xc(1), xc(2), bVector, hVector, yVector); %This
[I, yBar] = SectionProperties( I, yBar, xc(3)+1, xc(4), bVector, hVector, yVector);
cutoffAreaCent = [ 1.27*yBar(1), 1.27*yBar(1)];
yCentCutoff = [ yBar(1)/2, yBar(1)/2];
cutoffAreaGlue = [127];
yGlue = [121.27-yBar(1)-0.635]; % centroid of the cutoff area
[Qmax, Qglue] = calculateQ( Qmax, Qglue, xc(1), xc(2), cutoffAreaCent, cutoffAreaGlue,
[Qmax, Qglue] = calculateQ( Qmax, Qglue, xc(3)+1, xc(4), cutoffAreaCent, cutoffAreaGlue

% Cross Section B
bVector = [100 1.27 1.27 10 10 10 10 70 70];
hVector = [1.27 117.46 117.46 1.27 1.27 1.27 1.27 1.27 1.27];
yVector = [120.635 61.27 61.27 119.365 119.365 3.175 3.175 1.905 0.635]; % distances be
[I, yBar] = SectionProperties( I, yBar, xc(2)+1, xc(3), bVector, hVector, yVector); %Th
[I, yBar] = SectionProperties( I, yBar, xc(4)+1, xc(5), bVector, hVector, yVector);
cutoffAreaCent = [ 127 12.7 12.7 (120-yBar(900))*1.27 (120-yBar(900))*1.27];
yCentCutoff = [ 121.27-yBar(900)-0.635 120-yBar(900)-0.635 120-yBar(900)-0.635 120-yBar
cutoffAreaGlue = [127];
yGlue = [121.27-yBar(1)-0.635]; % centroid of the cutoff area
[Qmax, Qglue] = calculateQ( Qmax, Qglue, xc(2)+1, xc(3), cutoffAreaCent, cutoffAreaGlue
[Qmax, Qglue] = calculateQ( Qmax, Qglue, xc(4)+1, xc(5), cutoffAreaCent, cutoffAreaGlue

%Other constants
b = zeros(1,length(x));
b(:) = 2*1.27;
bGlue = zeros(1,length(x));
bGlue(:) = 10;
t = zeros(1,length(x));
t(:) = 1.27;
height = zeros(1,length(x));
height(:) = 120;

% It was unnecessary to calculate the SFD and BMD again since it was already
% done for Design 0

%Find applied stresses
%Shear Stress demand
ShearStress = VStress(I, b, Qmax, SFD_TrainLoad);
ShearStressGlue = VStress(I, 20+ 2*1.27, Qglue, SFD_TrainLoad);

%Shear Stress Capacity
ShearBuckle = VBuck(I, t, b, height, E, mu, yBar, a, Qmax);

%Bending Stress
%Bending Stress Demand
BendingStressT = MStressT( I,yBar, height, BMD_TrainLoad);
BendingStressC = MStressC( I,yBar, height, BMD_TrainLoad);

%Shear Stress Capacity
%Cross Section A
BendingBuck1 = MBuck( I, t, 15, E, mu, 2);
BendingBuck2 = MBuck( I, t, 70-2*(1.27), E, mu, 1);
BendingBuck3 = MBuck( I, t, 39.859, E, mu, 3 );

```

```

BendingBuck4 = MBuck( I, t, 80.1411, E, mu, 3 );

%Cross Section B
BendingBuck5 = MBuck( I, t, 56.4858-2*1.27, E, mu, 3 );
BendingBuck6 = MBuck( I, t, 117.46-56.4858, E, mu, 3 );
BendingBuck7 = MBuck( I, 2*t, 70-2*1.27, E, mu, 1 );
BendingBuck8 = MBuck( I, t, 56.4858-2*1.27, E, mu, 3 );

FOSBridge = FOSCalc(ShearStress, ShearStressGlue, BendingStressT, BendingStressC, SigT, SigC);

%Visualization

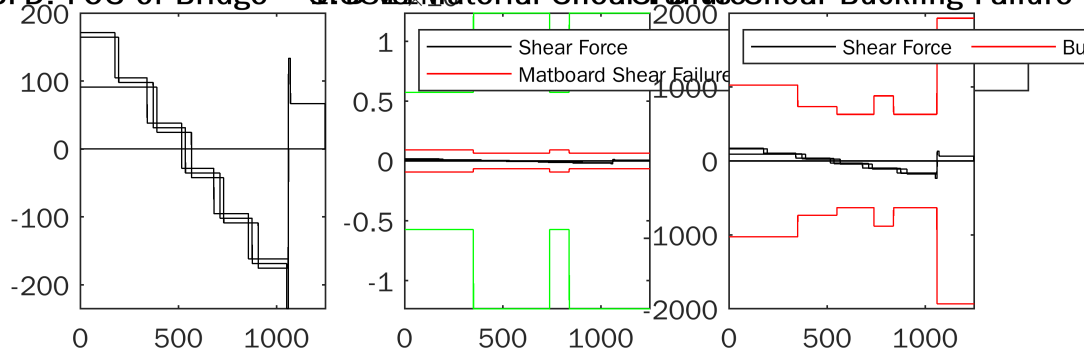
V_Mat = Vfail(I, b, Qmax, TauU);
V_Buck = VfailBuck( I, t, b, height, E, mu, yBar, a, TauU, Qmax);
V_Glue = Vfail(I, bGlue, Qglue, TauU);

M_MatT = MfailMatT(I, yBar, height, SigT, BMD_TrainLoad);
M_MatC = MfailMatC(I, yBar, height, SigC, BMD_TrainLoad);
M_Buck1 = MfailBuck(I, t, 15, E, mu, BMD_TrainLoad, 2, yBar, height, SigC);
M_Buck2 = MfailBuck(I, t, 70-2*(1.27), E, mu, BMD_TrainLoad, 1, yBar, height, SigC);
M_Buck3 = MfailBuck(I, t, 39.859, E, mu, BMD_TrainLoad, 3, yBar, height, SigC);
M_Buck4 = MfailBuck(I, t, 80.1411, E, mu, BMD_TrainLoad, 3, yBar, height, SigC);
M_Buck5 = MfailBuck(I, t, 56.4858-2*1.27, E, mu, BMD_TrainLoad, 3, yBar, height, SigC);
M_Buck6 = MfailBuck(I, t, 117.46-56.4858, E, mu, BMD_TrainLoad, 3, yBar, height, SigC);
M_Buck7 = MfailBuck(I, 2*t, 70-2*1.27, E, mu, BMD_TrainLoad, 1, yBar, height, SigC);
M_Buck8 = MfailBuck(I, t, 56.4858-2*1.27, E, mu, BMD_TrainLoad, 3, yBar, height, SigC);

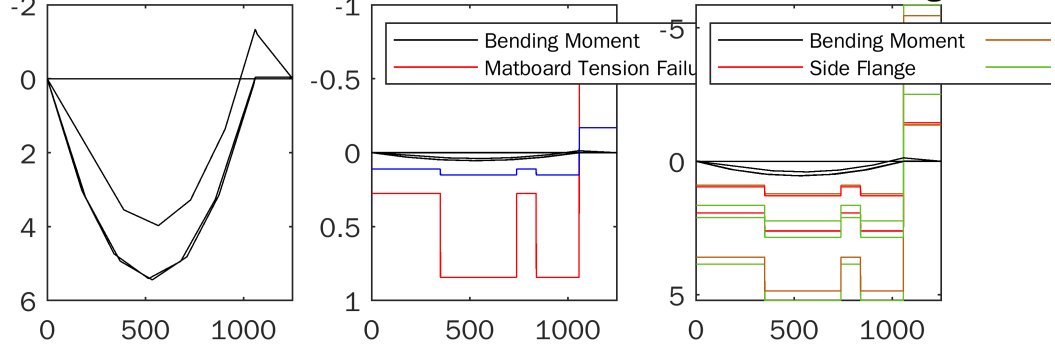
%% Visualization
VisualizeTL(x, SFD_TrainLoad, BMD_TrainLoad, SFD_TrainLoad2, BMD_TrainLoad2, SFD_TrainLoad2, BMD_TrainLoad2);

```

SFD: FOS of Bridge = 3.8814 Material Shear Failure vs Shear Buckling Failure



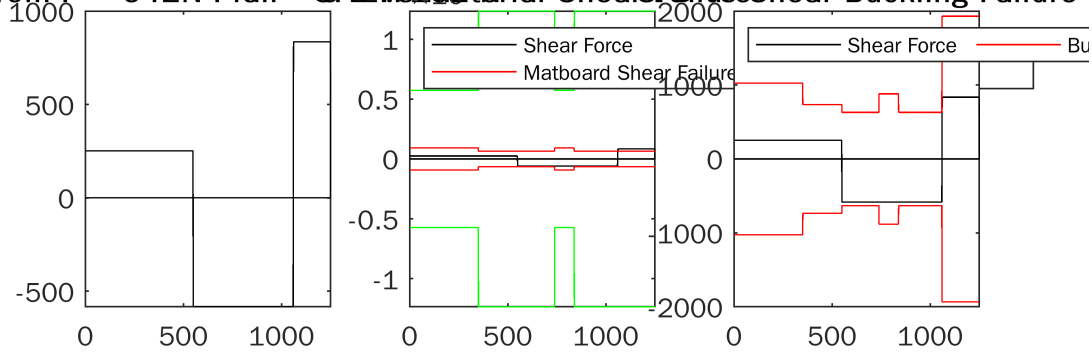
BMD: BMD vs Material Moment Buckling Failure



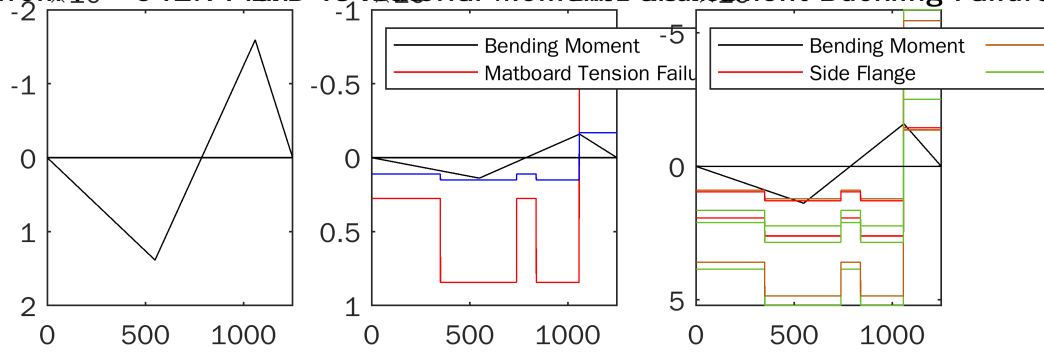
%Point Load our Design:

```
Pf = FailLoad( SFD_PL, BMD_PL, V_Mat, V_Glue, V_Buck, M_MatT, M_MatC, M_Buck1, M_Buck2,
[SFD_FAIL, BMD_FAIL] = ApplyPL(550, Pf, x, SFD_FAIL, locationA, locationB); % Construct
[SFD_FAIL, BMD_FAIL] = ApplyPL(L, Pf, x, SFD_FAIL, locationA, locationB); % Construct
VisualizePL(x, SFD_FAIL, BMD_FAIL, V_Mat, V_Glue, V_Buck, M_MatT, M_MatC, M_Buck1, M_Bu
```


FD from P = 642N Pfail = 542.7102N Material Shear Failure vs Shear Buckling Failure



BMD from P = 642N Pfail = 643.46N Material Moment Failure vs Moment Buckling Failure



```
% Train failure Functions:
```

```
function [ ShearFail ] = VStress(I, b, Qcent, SFD )
%Calculates the Shear Stress at Applied load for every x value

    ShearFail = ((SFD .* Qcent) ./ (I .* b));
end

function [ ShearBuckle ] = VBuck( I, t, b, height, E, mu, y, a, Qcent )
% Calculates the Shear Buckle capacity at every x value

    %pi
    pi = 355/113; %Pi approximation

    for i = 1:length(I)
        ShearBuckle(i) = ((5*E*pi^2)/(12*(1-mu^2))) * ((t(i)/height(i))^2 + (t(i)/a(i)))
    end

end

function [ BendingTensionStress ] = MStressT( I,y, height, BMD ) %Tension
% Calculates Tension Stress at every x value based on applied load
```

```

BendingTensionStress = zeros(1,length(I));

for i = 1 : length(I)
    ybot = y(i);
    ytop = height(i) - y(i);

    if BMD(i) >= 0 % If the moment is positive, the tension failure will be at the
        BendingTensionStress(i) = (BMD(i) * ybot) / I(i);
    elseif BMD < 0 % If the moment is negative, the tension failure will be at the
        BendingTensionStress(i) = (BMD(i) * ytop) / I(i);
    end
end

end

function [ BendingCompressionStress ] = MStressC( I,y, height, BMD ) %Compression
% % Calculates Compression Stress at every x value based on applied load

BendingCompressionStress = zeros(1,length(I));

for i = 1:length(I)

    ybot = y(i);
    ytop = height(i) - y(i);

    if BMD(i) >= 0 % If the moment is positive, the compression failure will be at
        BendingCompressionStress(i) = (BMD(i) * ytop) / I(i);
    elseif BMD(i) > 0 % If the moment is negative, the compression failure will be
        BendingCompressionStress(i) = (BMD(i) * ybot) / I(i);
    end
end

end

function [ MBuckStress ] = MBuck( I, t, bBuck, E, mu, caseNum )
% Calculates Bending Capacity based on plate buckling for every x value

    if caseNum == 1
        k = 4;
    end
    if caseNum == 2
        k = 0.425;
    end
    if caseNum == 3
        k = 6;
    end

    for i = 1:length(I)
        MBuckStress(i) = ((k*E*pi^2)/(12*(1-mu^2))) * (t(i)/bBuck)^2;
    end

end

```

```

function [ FOS ] = FOSCalc( ShearFail, GlueFail, BendingTensionStress, BendingCompressionStress, ...
    %Finds FOS based on every capacity and every demand -> FOS =
    %Capacity/Demand
    ShearFOS = min(TauCapacity ./ abs(ShearFail));
    GlueFOS = min(GlueCapacity ./ abs(GlueFail));
    ShearBucklingFOS = min(TauBuckling ./ abs(ShearFail));
    TensionFOS = min(TensionCapacity ./ abs(BendingTensionStress));
    CompressionFOS = min(CompressionCapacity ./ abs(BendingCompressionStress));
    CompressionBucklingFOS1 = min(CompressionBucklingCapacity1 ./ abs(BendingCompressionStress));
    CompressionBucklingFOS2 = min(CompressionBucklingCapacity2 ./ abs(BendingCompressionStress));
    CompressionBucklingFOS3 = min(CompressionBucklingCapacity3 ./ abs(BendingCompressionStress));
    CompressionBucklingFOS4 = min(CompressionBucklingCapacity4 ./ abs(BendingCompressionStress));
    CompressionBucklingFOS5 = min(CompressionBucklingCapacity5 ./ abs(BendingCompressionStress));
    CompressionBucklingFOS6 = min(CompressionBucklingCapacity6 ./ abs(BendingCompressionStress));
    CompressionBucklingFOS7 = min(CompressionBucklingCapacity7 ./ abs(BendingCompressionStress));
    CompressionBucklingFOS8 = min(CompressionBucklingCapacity8 ./ abs(BendingCompressionStress));

    %Finds the minimum of the FOS
    FOSes = [ShearFOS GlueFOS ShearBucklingFOS TensionFOS CompressionFOS CompressionBucklingFOS1 ...
        CompressionBucklingFOS2 ... CompressionBucklingFOS8];
    FOS = min(FOSes);
end

%Cross Section Functions
function [I, yBar] = SectionProperties( I, yBar, xStart, xEnd, bVector, hVector, yVector)
    % Input: Geometric Inputs. Format will depend on user
    % Output: Sectional Properties at every value of x. Each property is a 1-D array of

    % finding global centroid
    for i = xStart : xEnd
        sumAY = 0;
        sumA = 0;

        for j = 1:length(bVector)
            sumAY = sumAY + bVector(j) * hVector(j) * yVector(j);
            sumA = sumA + bVector(j) * hVector(j);
        end

        yBar(i) = sumAY/sumA;

        %finding I
        for j = 1:length(bVector)
            I(i) = I(i) + (bVector(j)*hVector(j)^3)/12 + bVector(j)*hVector(j)*(yVector(j)-yBar(i))^2;
        end
    end
end

end

function [Qmax, Qglue] = calculateQ( Qmax, Qglue, xStart, xEnd, cutoffAreaCent, cutoffArea)
    for i = xStart : xEnd
        %finding Q at centroid

```

```

        for j = 1:length(cutoffAreaCent)
            Qmax(i) = Qmax(i) + cutoffAreaCent(j)*yCentCutoff(j);
        end

        %finding Q at glue
        for j = 1:length(cutoffAreaGlue)
            Qglue(i) = Qglue(i) + cutoffAreaGlue(j)*yGlueCutoff(j);
        end

    end

end

function [] = VisualizeTL(x, SFD_PL, BMD_PL, SFD2, BMD2, SFD3, BMD3, V_Mat, V_Glue, V_F
    %Plots all outputs of design process
    figure;

    axis = zeros(1,length(x));
    hold on;
    %shear force diagram
    subplot(2, 3, 1)
    plot(x, SFD_PL, "k")
    hold on
    plot(x, SFD2, "k")
    hold on
    plot(x, SFD3, "k")
    hold on
    plot(x,axis,"k")
    hold off
    title1 = strcat("SFD: FOS of Bridge = ", num2str(FOS));
    title(title1)

    %shear force vs mat shear fail
    subplot(2, 3, 2)
    plot(x, SFD_PL, "k")
    hold on
    plot(x, V_Mat, "r")
    hold on
    plot(x, V_Glue, "g")
    hold on
    plot(x, -V_Mat, "r")
    hold on
    plot(x, -V_Glue, "g")
    hold on
    plot(x, SFD2, "k")
    hold on
    plot(x, SFD3, "k")
    hold on
    plot(x,axis,"k")
    hold off
    legend("Shear Force", "Matboard Shear Failure", "Glue Shear Failure", 'Location', 'nor
    title("SFD vs Material Shear Failure")

```

```

subplot(2, 3, 3)
plot(x, SFD_PL, "k")
hold on
plot(x, V_Buck, "r")
hold on
plot(x, -V_Buck, "r")
hold on
plot(x, SFD2, "k")
hold on
plot(x, SFD3, "k")
hold on
plot(x,axis,"k")
hold off
legend("Shear Force", "Buckling Failure", 'Location','northwest','NumColumns',2)
title("SFD vs Shear Buckling Failure")

%BMD Diagrams
subplot(2, 3, 4)
plot(x, BMD_PL, "k")
hold on
plot(x, BMD2, "k")
hold on
plot(x, BMD3, "k")
hold on
plot(x,axis,"k")
hold on
title2 = strcat("BMD:");
title(title2)
set(gca, 'YDir', 'reverse')
set(gca, 'YDir', 'reverse')
hold off

subplot(2, 3, 5)
plot(x, BMD_PL, "k")
hold on
plot(x, M_MatT, "r")
hold on
plot(x, M_MatC, "b")
hold on
plot(x, BMD2, "k")
hold on
plot(x, BMD3, "k")
hold on
plot(x,axis,"k")
hold on
legend("Bending Moment", "Matboard Tension Failure", "Matboard Compression Failure",
title("BMD vs Material Moment Failures")
set(gca, 'YDir', 'reverse')
hold off

subplot(2, 3, 6)
plot(x, BMD_PL, "k")
hold on
plot(x, M_Buck1, "-r")

```

```

hold on
plot(x, M_Buck2, 'Color', [0.74, 0.36, 0.04])
hold on
plot(x, M_Buck3, 'Color', [0.4, 0.75, 0.13])
hold on
plot(x, M_Buck4, "-r")
hold on
plot(x, M_Buck5, 'Color', [0.4, 0.75, 0.13])
hold on
plot(x, M_Buck6, 'Color', [0.4, 0.75, 0.13])
hold on
plot(x, M_Buck7, 'Color', [0.74, 0.36, 0.04])
hold on
plot(x, M_Buck8, 'Color', [0.4, 0.75, 0.13])
hold on
plot(x, BMD2, "k")
hold on
plot(x, BMD3, "k")
hold on
plot(x,axis,"k")
hold on
%legend("Bending Moment", "Mid Flange Buckling", "Side Flange Buckling", "Web Compr
title("BMD vs Moment Buckling Failures")
legend("Bending Moment", "Side Flange", "Mid Flange", "Web Flange", 'Location','north
set(gca, 'YDir', 'reverse')
hold off

```

end

```

%%Point Load Functions:
function [ SFD, BMD ] = ApplyPL( xP, P, x, SFD, locationA, locationB ) %(Location, Force)
% Constructs SFD and BMD for every x value
% Can be iterated since SFD and BMD diagrams can be added together

%Calculate reaction forces
SupportB = ((xP * P) / (locationB)); %Using Sum of Moments = 0
SupportA = (P - SupportB); %Using sum of vertical force = 0

%Fill out SFD diagram as if by hand
SFDLocal = zeros(1, length(SFD));
if xP < locationB %Force is closer to the left side than support B
    for i = 1:(xP-1) %SFD before new load
        SFDLocal(i) = SupportA;
    end

    for i = xP:locationB %SFD after new load
        SFDLocal(i) = (SupportA - P);
    end

    for i = 1:length(SFD) %add the new SFD with the old ones

```

```

        SFD(i) = SFD(i) + SFDLocal(i);
    end

    else %Force is farther from left side than support B
        for i = 1:(locationB-1) %SFD before new load
            SFDLocal(i) = SupportA;
        end

        for i = locationB:xP-1 %SFD after new load
            SFDLocal(i) = (SupportA + SupportB);
        end

        for i = xP:length(SFD)
            SFDLocal(i) = 0;
        end

        for i = 1:length(SFD) %add the new SFD with the old ones
            SFD(i) = SFD(i) + SFDLocal(i);
        end
    end

    %form BMD:
    dx = x(2) - x(1);
    BMD = cumsum((SFD * dx));
end

function [ V_fail ] = Vfail(I, b, Qcent, TauU )
% Calculates Shear Fail based on Tau Ultimate

    V_fail = TauU .* I .* b ./ Qcent;
end

function [ V_Buck ] = VfailBuck( I, t, b, height, E, mu, y, a, TauU, Qcent )
% Calculates Shear Fail based on Buckling

    %pi
    pi = 355/113;

    for i = 1:length(I)
        sigma(i) = ((5*E*pi^2)/(12*(1-mu^2))) * ((t(i)/height(i))^2 + (t(i)/a(i))^2);
    end

    %Calculate buckle fail by using the above formula
    V_Buck = Vfail(I, b, Qcent, sigma);

end

function [ M_MatT ] = MfailMatT( I,y, height, SigT, BMD )
% Calculates Tension Fail

    for i = 1 : length(I)

        ybot = y;
        ytop = height(i) - y; %I need somesort of height vector
    end
end

```

```

        if BMD(i) > 0 % If the moment is positive, the tension failure will be at the b
            M_MatT(i) = (SigT * I(i)) / ybot(i);
        elseif BMD(i) < 0 % If the moment is negative, the tension failure will be at t
            M_MatT(i) = (-SigT * I(i)) / ytop(i);
        end
    end
end

function [ M_MatC ] = MfailMatC( I,y, height, SigC, BMD )
% Calculates Compression Fail

ybot = y;
ytop = height - y;

for i = 1 : length(I)
    if BMD(i) > 0 % If the moment is positive, the compression failure will be at t
        M_MatC(i) = (SigC * I(i)) / ytop(i);
    elseif BMD(i) < 0 % If the moment is negative, the compression failure will be
        M_MatC(i) = (-SigC * I(i)) / ybot(i);
    end
end
end

function [ M_Buck ] = MfailBuck( I, t, bBuck, E, mu, BMD, caseNum, y, height, SigC )
% Calculates Bending Fail based on buckling

%Assign a k value based on the case
if caseNum == 1
    k = 4;
end
if caseNum == 2
    k = 0.425;
end
if caseNum == 3
    k = 6;
end

for i = 1:length(I)
    sigma(i) = ((k*E*pi^2)/(12*(1-mu^2))) * (t(i)/bBuck)^2;
end

%Calculate buckle fail using the above function
M_Buck = MfailMatC(I,y, height, min(sigma), BMD);

end

function [ Pf ] = FailLoad( SFD, BMD, V_Mat, V_Glue, V_Buck, M_MatT, M_MatC, M_buck1, M
%Calculate the point load that causes the failure

%Flexural Stress
%Because P = 1, the BMD and SFD are simply the coefficients of P. By
%dividing them, we are finding P despite the fact that there is no actual P
%variable.

```



```

CompressionFailure = abs(M_MatC) ./ abs(BMD);
TensionFailure = abs(M_MatT) ./ abs(BMD);
BendingBucklingFailure1 = abs(M_buck1) ./ abs(BMD);
BendingBucklingFailure2 = abs(M_buck2) ./ abs(BMD);
BendingBucklingFailure3 = abs(M_buck3) ./ abs(BMD);
BendingBucklingFailure4 = abs(M_buck4) ./ abs(BMD);
BendingBucklingFailure5 = abs(M_buck5) ./ abs(BMD);
BendingBucklingFailure6 = abs(M_buck6) ./ abs(BMD);
BendingBucklingFailure7 = abs(M_buck7) ./ abs(BMD);
BendingBucklingFailure8 = abs(M_buck8) ./ abs(BMD);

%Shear Stress
ShearFailureMat = V_Mat ./ abs(SFD);
ShearFailureGlue = V_Glue ./ abs(SFD);
ShearBucklingFailure = V_Buck ./ abs(SFD);

%Find the lowest Pf
Pforces = [min(CompressionFailure) min(TensionFailure) min(BendingBucklingFailure1) min(BendingBucklingFailure2) min(BendingBucklingFailure3) min(BendingBucklingFailure4) min(BendingBucklingFailure5) min(BendingBucklingFailure6) min(BendingBucklingFailure7) min(BendingBucklingFailure8) min(ShearFailureMat) min(ShearFailureGlue) min(ShearBucklingFailure)];
Pf = min(Pforces);

end

function [] = VisualizePL(x, SFD_PL, BMD_PL, V_Mat, V_Glue, V_Buck, M_MatT, M_MatC, M_Buck1, M_Buck2, M_Buck3, M_Buck4, M_Buck5, M_Buck6, M_Buck7, M_Buck8)

    axis = zeros(1,length(x));
    %Plots all outputs of design process
    figure;

    hold on;
    %shear force diagram
    subplot(2, 3, 1)
    plot(x, SFD_PL, "k")
    hold on
    plot(x,axis,"k")
    hold off
    title1 = strcat("SFD from P = ", int2str(floor(Pf)), "N Pfail = ", num2str(Pf), " N");
    title(title1)

    %shear force vs mat shear fail
    subplot(2, 3, 2)
    plot(x, SFD_PL, "k")
    hold on
    plot(x, V_Mat, "r")
    hold on
    plot(x, V_Glue, "g")
    hold on
    plot(x, -V_Mat, "r")
    hold on
    plot(x, -V_Glue, "g")
    hold on
    plot(x,axis,"k")
    legend("Shear Force", "Matboard Shear Failure", "Glue Shear Failure",'Location','none')

```

```

title("SFD vs Material Shear Failure")
hold off

subplot(2, 3, 3)
plot(x, SFD_PL, "k")
hold on
plot(x, V_Buck, "r")
hold on
plot(x, -V_Buck, "r")
hold on
hold on
plot(x,axis,"k")
legend("Shear Force", "Buckling Failure", 'Location','northwest','NumColumns',2)
title("SFD vs Shear Buckling Failure")
hold off

%BMD Diagrams
subplot(2, 3, 4)
plot(x, BMD_PL, "k")
hold on
hold on
plot(x,axis,"k")
title2 = strcat("BMD from P = ", int2str(floor(Pf)), "N Pfail = ", int2str(Pf), " N")
title(title2)
set(gca, 'YDir', 'reverse')
set(gca, 'YDir', 'reverse')
hold off

subplot(2, 3, 5)
plot(x, BMD_PL, "k")
hold on
plot(x, M_MatT, "r")
hold on
plot(x, M_MatC, "b")
hold on
plot(x,axis,"k")
hold on
legend("Bending Moment", "Matboard Tension Failure", "Matboard Compression Failure",
title("BFD vs Material Moment Failures")
set(gca, 'YDir', 'reverse')
hold off

subplot(2, 3, 6)
plot(x, BMD_PL, "-k")
hold on
plot(x, M_Buck1, "-r")
hold on
plot(x, M_Buck2, 'Color', [0.74, 0.36, 0.04])
hold on
plot(x, M_Buck3, 'Color', [0.4, 0.75, 0.13])
hold on
plot(x, M_Buck4, "-r")
hold on
plot(x, M_Buck5, 'Color', [0.4, 0.75, 0.13])

```

```

hold on
plot(x, M_Buck6, 'Color', [0.4, 0.75, 0.13])
hold on
plot(x, M_Buck7, 'Color', [0.74, 0.36, 0.04])
hold on
plot(x, M_Buck8, 'Color', [0.4, 0.75, 0.13])
hold on
plot(x,axis,"k")
hold on
%legend("Bending Moment", "Mid Flange Buckling", "Side Flange Buckling", "Web Compr
title("BMD vs Moment Buckling Failures")
legend("Bending Moment", "Side Flange", "Mid Flange", "Web Flange", 'Location','nort
set(gca, 'YDir', 'reverse')
hold off

```

```

end

```