Supporting Calculations

Matboard Bridge Project

*CIV102*

Contents

[2 Design 0 Calculations 2](#_Toc89131289)

[2.1 Hand Calculations and Code Output 2](#_Toc89131290)

[2.1.1 Reaction Forces, SFD and BMD 2](#_Toc89131291)

[2.1.2 Sectional Properties 3](#_Toc89131292)

[2.1.3 Flexural Failure: 4](#_Toc89131293)

[2.1.4 Matboard Shear Failure 5](#_Toc89131294)

[2.1.5 Glue Shear Failure 6](#_Toc89131295)

[2.1.6 Plate Buckling 7](#_Toc89131296)

[2.1.7 Shear Buckling 8](#_Toc89131297)

[2.1.8 Failure Load (Pfail) 9](#_Toc89131298)

[2.1.9 Midspan Deflection 10](#_Toc89131299)

[2.2 SFD and BMD Data Plots 10](#_Toc89131300)

[2.2.1 Pfail SFD and BMD 10](#_Toc89131301)

[2.2.2 SFD vs Material Shear and Shear Buckling Failures 10](#_Toc89131302)

[2.2.3 BMD vs Material Moment and Buckling Failures 11](#_Toc89131303)

[3 Appendices 12](#_Toc89131304)

[3.1 Appendix I: MATLAB Code Parameter Names 12](#_Toc89131305)

[3.2 Appendix II: Complete MATLAB Code: 13](#_Toc89131306)

# Design 0 Calculations

## Hand Calculations and Code Output

### Reaction Forces, SFD and BMD

**Diagram

Description automatically generated**

**Code Sample Call of Relevant Function(s):**

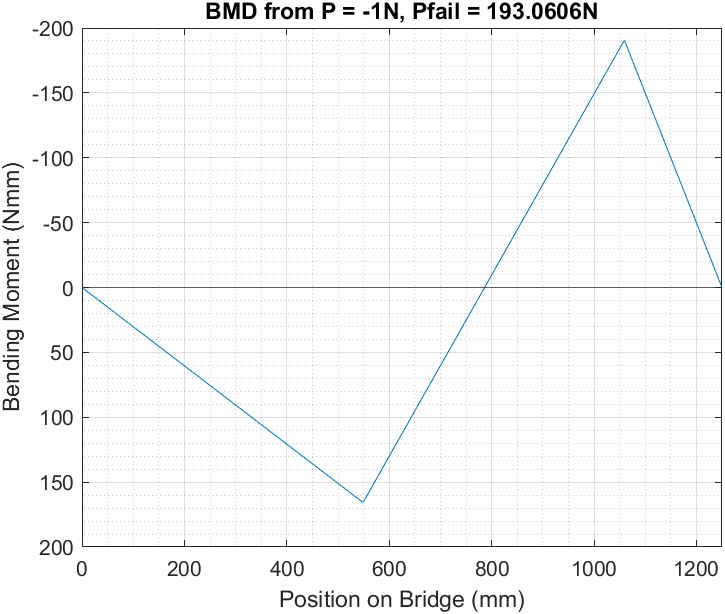
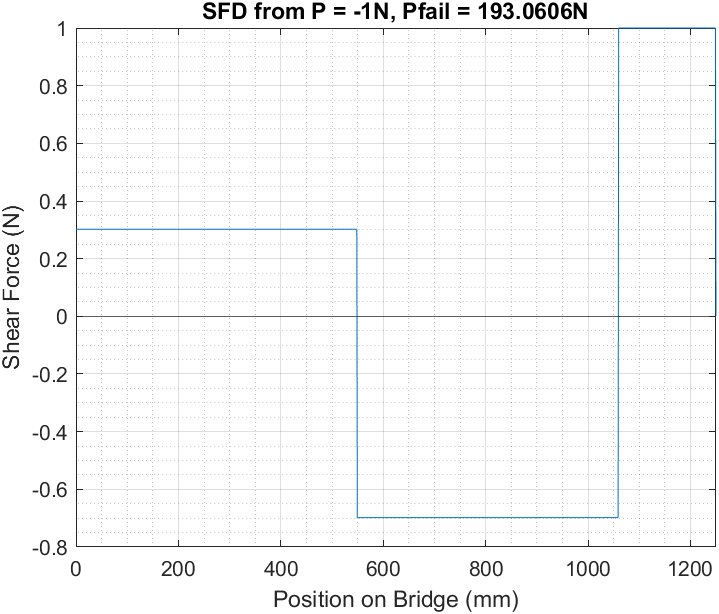
load = -1;

[SFD\_PL, BMD\_PL, P] = ApplyPL(550, load, x, P, n); % Construct SFD, BMD

[SFD\_PL, BMD\_PL, P] = ApplyPL(L, load, x, P, n); % Construct SFD, BMD

SFD\_PL and BMD\_PL are array containing the value of the SFD and BMD at each x coordinate. The ApplyPL function itself does not plot the SFD/BMD, however, for the sake of clarity the SFD and BMD were plotted using the VisualizePL function using the arrays generated using the function above.

**Code Output:**



### Sectional Properties

Diagram

Description automatically generated

Text, letter

Description automatically generated  


**Code Sample Call of Relevant Function(s):**

[Y\_bridge, I\_bridge, Q\_bridge, b\_bridge, section\_heights] = SectionProperties(xc, tfw, tft, hw, tw, ws, bfw, bft, rtw, rtt, rbw, rbt, n);

**Code Output:**

I\_bridge = 415675.1027

Y\_bridge = 41.7016

Q\_bridge = 6248.3312

b\_bridge = 2.54

section\_heights = 75

The output of the function above are all arrays of length n (the length of the bridge) except for b\_bridge and Q\_bridge which are n by y matrices where y is the height of the cross section. The code output section only displayed one value from the array. However, for design 0 all the values within these arrays are constant since the cross section is consistent throughout the length of the bridge. As for Q\_bridge and b\_bridge, their values at the centroid were be displayed as they will be used in further calculations (see section 2.1.4) For further details on the function parameters above and their definitions see Appendix II.

### Flexural Failure:

Diagram

Description automatically generated

**Code Sample Call of Relevant Function(s):**

[ m\_mat\_tension ] = MfailMatT( I\_bridge, Y\_bridge, section\_heights, SigT, BMD\_PL);

[ m\_mat\_compression ] = MfailMatC( I\_bridge, Y\_bridge, section\_heights, SigC, BMD\_PL );

**Code Output:**

m\_mat\_tension = 299043

m\_mat\_compression = -59808.6

The above functions output the matboard tension and compression failure moment values for each point on the bridge as an array of length n, where n is the length of the bridge. These functions automatically determine what side of the bridge experiences tension and compression based on where the BMD is negative or positive. For this reason, the arrays contain both the failure moment values for both the top and bottom flange according to the aforementioned condition. For the sake of comparison to the manual calculation seen above, the code output above only shows the m\_mat\_tension value for when the bottom flange is under tension. Similarly, to m\_mat\_tension, the value of m\_mat\_compression for when the bottom flange is under compression was shown.

### Matboard Shear Failure

Diagram

Description automatically generated with medium confidence

**Code Sample Call of Relevant Function(s):**

[v\_fail] = Vfail(I\_bridge, b\_bridge, Y\_bridge, TauU, Q\_bridge);

**Code Output:**

v\_fail = 675.9181813

Since the function above outputs v\_fail, an array of length n, where n is the length of the bridge, only one value of v\_fail was shown in the sample output above.

### Glue Shear Failure

Text, letter

Description automatically generated



**Code Sample Call of Relevant Function(s):**

[ V\_GlueTF V\_GlueBF V\_GlueTW V\_GlueBW] = VglueFail(I\_bridge, Q\_bridge, b\_bridge, TauG, tft, bft, section\_heights, xc);

**Code Output:**

V\_GlueTF = NaN 🡨 Top flange layer joint

V\_GlueBF = NaN 🡨 Bottom flange layer joint

V\_GlueTW = 4714.769465 🡨 Top flange/web joint

V\_GlueBW = 557.7619034 🡨 Bottom flange/web joint

**Note:** The reason for the difference between the code and the and the hand calculations above is due to the lack of millimeter precision in the Q value used in calculating the above values in the function due to the limitations caused by storing Q in a matrix. For this reason, a significant factor of safety was used when considering the various glue failure loads in our final bridge design due to these inaccuracies.

### Plate Buckling

Diagram

Description automatically generated  
Diagram

Description automatically generated

**Code Sample Call of Relevant Function(s):**

[ M\_Buck1, M\_Buck2, M\_Buck3 ] = MfailBuck( xc, bfw, bft, tfw, tft, ws, tw, section\_heights, Y\_bridge,

**Code Output:**

M\_Buck1 (BMD > 0) = 46000.3 🡨 Plate #1

M\_Buck1 (BMD < 0) = -36730.9 🡨 Plate #6

M\_Buck2 (BMD > 0) = 293254 🡨 Plates #2 and #3

M\_Buck3 (BMD > 0) = 403585 🡨 Plates #4 and #5

M\_Buck3 (BMD < 0) = -202226 🡨 Plates #7 and #8

**Note:** All outputs of MfailBuck are arrays of length n. Only 2 values for each are shown above; one when the BMD is positive and the other when the BMD is negative. Note the lack of a value for M\_Buck2 when the BMD is negative. This is due to the lack of a ‘overhang’ on the bottom flange meaning there is no buckling of this kind when the BMD is negative since the bottom is in compression.

### Shear Buckling

Text

Description automatically generated

**Code Sample Call of Relevant Function(s):**

[ v\_buck ] = VfailBuck(xc, tw, a, hw, E, mu, n, I\_bridge, b\_bridge, Y\_bridge, Q\_bridge);

**Code Output:**

v\_buck(0 < x < 550) = 904.889 🡨 Section 1

v\_buck(550 < x < 1060) = 907.406 🡨 Section 2

v\_buck(1060 < x < 1250) = 1018.81 🡨 Section 3

**Note:** The output of VfailBuck is an array of length n. The values shown above represent the values of v\_buck within the specified intervals.

### Failure Load (Pfail)

A picture containing text, receipt

Description automatically generated

Text, letter

Description automatically generated

**Code Sample Call of Relevant Function(s):**

[ Pf, failure\_mode ] = FailLoad( load, SFD\_PL, BMD\_PL, v\_fail, v\_buck, m\_mat\_tension, m\_mat\_compression, M\_Buck1, M\_Buck2, M\_Buck3, V\_GlueTF, V\_GlueBF, V\_GlueTW, V\_GlueBW, true, "Design0");

**Code Output:**

Pf = 193.0605836

Failure\_mode = Center Flange Buckling

The difference in Pf and Pfail in hand calculations can likely be attributed to a rounding error. Additionally, for further details on function parameters, see Appendix II code comments.

### Midspan Deflection

ADD HERE

## SFD and BMD Data Plots

The following section contains the plots produced by plotting the arrays generated by the functions discussed in section 3.1 using the VisualizePL function. For details on this function see comments in Appendix II.

### Pfail SFD and BMD

Chart

Description automatically generatedChart, line chart

Description automatically generated

### SFD vs Material Shear and Shear Buckling Failures

Chart

Description automatically generatedChart

Description automatically generated

### BMD vs Material Moment and Buckling Failures

Chart

Description automatically generatedChart

Description automatically generated

# Appendices

## Appendix I: MATLAB Code Parameters

|  |  |  |
| --- | --- | --- |
| **Name** | **Purpose** | **Type** |
| n | Determines the number of locations to evaluate bridge failure | Integer |
| load | Stores value of point load to be applied | Integer |
| x | Defines x coordinates | Array |
| P | Stores point loads | Array |
| xc | Stores x coordinates of where cross section changes | Array |
| tfw | Stores top flange width | Array |
| tft | Stores top flange thickness | Array |
| hw | Stores web height | Array |
| tw | Stores web thickness | Array |
| bfw | Stores bottom flange width | Array |
| bft | Stores bottom flange thickness | Array |
| a | Stores x coordinates of diaphragms | Array |
| ws | Stores space between webs | Array |
| rtw | Stores width of top flange/web glue tab | Array |
| rtt | Stores thickness of top flange/web glue tab | Array |
| rbw | Stores width of bottom flange/web glue tab | Array |
| rbt | Stores thickness of bottom flange/web glue tab | Array |
| SigT | Stores matboard tensile strength | Integer |
| SigC | Stores matboard compressive strength | Integer |
| E | Stores young’s modulus of matboard | Integer |
| TauU | Stores shear stress of matboard | Integer |
| TauG | Stores shear stress of glue | Integer |
| mu | Stores poisson’s ratio | Integer |
| Y\_bridge | Stores centroid for each x coordinate of bridge | Array |
| I\_bridge | Stores second moment of area for each x coordinate of bridge | Array |
| Q\_bridge | Stores first moment of area for each x and y coordinate of bridge | Matrix |
| b\_bridge | Stores thickness for each x and y coordinate of bridge | Matrix |
| section\_heights | Stores height of bridge for each x coordinate | Array |
| v\_fail | Stores shear failure value for each x coordinate | Array |
| v\_buck | Stores shear buckling failure value for each x coordinate | Array |
| m\_mat\_tension | Stores moment tension failure value for each x coordinate | Array |
| m\_mat\_compression | Stores moment compression failure value for each x coordinate | Array |
| m\_Buck1 | Stores moment plate buckling failure values for center of top or bottom flange (based on BMD sign) | Array |
| m\_Buck2 | Stores moment plate buckling failure values for overhang of top or bottom flange (based on BMD sign) | Array |
| m\_Buck3 | Stores moment plate buckling failure values for webs | Array |
| V\_GlueTF | Stores shear failure value for top flange glue | Array |
| V\_GlueBF | Stores shear failure value for bottom flange glue | Array |
| V\_GlueTW | Stores shear failure value for top flange/web joint glue | Array |
| V\_GlueBW | Stores shear failure value for bottom flange/web joint glue | Array |

## Appendix II: Complete MATLAB Code:

%% 0. Initialize Parameters

n = 1250; % Number of locations to evaluate bridge failure

L = 1250; % Length of bridge

x = linspace(0, L, n); % Define x coordinate

P = zeros(1,n); % Initializes Loads

%% 1. Point Loading Analysis (SFD, BMD)

load = -1062;

[SFD\_PL, BMD\_PL, P] = ApplyPL(550, load, x, P, n); % Construct SFD, BMD

[SFD\_PL, BMD\_PL, P] = ApplyPL(L, load, x, P, n); % Construct SFD, BMD

xc = [0 550 1060 L]; % Location, x, of cross-section change

tfw = [100 100 100 100]; % Top Flange Width

tft = [1.27 1.27 1.27 1.27]; % Top Flange Thickness

hw = [72.46 72.46 72.46 72.46]; % Web Height

tw = [1.27 1.27 1.27 1.27]; % Web Thickness (Assuming 2 separate webs)

bfw = [80 80 80 80]; % Bottom Flange Width

bft = [1.27 1.27 1.27 1.27]; % Bottom Flange Thickness

a = [0 550 1060 L]; % Diaphragm x coords

ws = [77.46 77.46 77.46 77.46]; % Web Spacing

rtw = [10 10 10 10]; % little rectangles at top of webs for glue area width

rtt = [1.27 1.27 1.27 1.27]; % little rectangles at top of webs for glue area thickness

rbw = [0 0 0 0]; % little rectangles at bottom of webs for glue area width

rbt = [0 0 0 0]; % little rectangles at bottom of webs for glue area thickness

% Optional but you need to ensure that your geometric inputs are correctly implemented

VisualizeBridge(xc, tfw, tft, hw, tw, ws, bfw, bft, rtw, rtt, rbw, rbt);

%% 3. Define Material Properties

SigT = 30;

SigC = 6;

E = 4000;

TauU = 4;

TauG = 2;

mu = 0.2;

% Calls Function to calculate geometric properties (output is in the form of arrays)

[Y\_bridge, I\_bridge, Q\_bridge, b\_bridge, section\_heights] = SectionProperties(xc, tfw, tft, hw, tw, ws, bfw, bft, rtw, rtt, rbw, rbt, n );

% Outputs Calculations from above function for each cross section

for c = 1:length(xc)

    if xc(c) ~=0

        "Y bar cross section " + n + " = " + Y\_bridge(xc(c))

        "I for cross section " + n + " = " + I\_bridge(xc(c))

        "Q for cross section at centriod " + n + " = " + Q\_bridge(xc(c), round(Y\_bridge(xc(c)))) % Note this value is rounded due to mm precision in Q matrix

        "Height of cross section " + n + " = " + section\_heights(xc(c))

    end

end

[v\_fail] = Vfail(I\_bridge, b\_bridge, Y\_bridge, TauU, Q\_bridge);

[ v\_buck ] = VfailBuck(xc, tw, a, hw, E, mu, n, I\_bridge, b\_bridge, Y\_bridge, Q\_bridge);

[ m\_mat\_tension ] = MfailMatT( I\_bridge, Y\_bridge, section\_heights, SigT, BMD\_PL);

[ m\_mat\_compression ] = MfailMatC( I\_bridge, Y\_bridge, section\_heights, SigC, BMD\_PL );

[ M\_Buck1, M\_Buck2, M\_Buck3 ] = MfailBuck( xc, bfw, bft, tfw, tft, ws, tw, section\_heights, Y\_bridge, I\_bridge, E, mu, BMD\_PL);

[ V\_GlueTF V\_GlueBF V\_GlueTW V\_GlueBW] = VglueFail(I\_bridge, Q\_bridge, b\_bridge, TauG, tft, bft, section\_heights, xc);

% FAILURE TYPES FOR FailLoad FUNCTION:

% matt = moment matboard tension failure

% matc = moment matboard compression failure

% buck1 = case 1 buckling of top or bottom flange (center part between webs)

% buck2 = case 2 buckling of top or bottom flange (part that sticks out past the webs)

% buck3 = case 3 buckling of left and right webs

% vmat = matboard shear failure

% vbuck = matboard shear buckling failure

% vgluetf = shear glue failure of top flange

% vgluebf = shear glue failure of bottom flange

% vgluetw = shear glue failure of top flange/web connection

% vgluebw = shear glue failure of bottom flange/web connection

% Calls function to calculate failure load and failure mode

[ Pf, failure\_mode ] = FailLoad( load, SFD\_PL, BMD\_PL, v\_fail, v\_buck, m\_mat\_tension, m\_mat\_compression, M\_Buck1, M\_Buck2, M\_Buck3, V\_GlueTF, V\_GlueBF, V\_GlueTW, V\_GlueBW, true, "Design0");

VisulizePL(x, load, Pf, SFD\_PL, BMD\_PL, v\_fail, v\_buck, m\_mat\_tension, m\_mat\_compression, M\_Buck1, M\_Buck2, M\_Buck3, V\_GlueTF, V\_GlueBF, V\_GlueTW, V\_GlueBW);

% Outputs Failure load and fail type

"Pfail = " + Pf

"Failure mode = " + failure\_mode

% Calls function to check if design overuses material

[ material\_ok ] = MaterialCheck(xc, tfw, tft, hw, tw, ws, bfw, bft, rtw, rtt, rbw, rbt);

% Outputs Material Check results

"Material is: " +  material\_ok

%%

function [ SFD, BMD, Loads ] = ApplyPL( xP, P, x, Loads, n )

% Constructs SFD and BMD from application of 1 Point Load. Assumes fixed location of supports

%   Input: location and magnitude of point load. The previous Loads can be entered as input to

%       construct SFD of multiple point loads

%   Output: SFD, BMD, Loads, all 1-D arrays of length n

    Loads(xP) = P;

    moment = 0;

    force = 0;

    % Calculates sum of forces and moments for use in reaction force

    % calculations

    for i = 1:length(Loads)

        if and(i ~= 1060, i ~= 1)

            moment = Loads(i) \* i + moment;

            force = Loads(i) + force;

        end

    end

    % calculates reaction forces

    b = -moment/1060;

    a = -b - force;

    % Adds reaction forces to Loads array

    Loads(1) = a;

    Loads(1060) = b;

    % Initializes new SFD

    SFD = zeros(1, n);

    % Calculates shear force at every point of bridge

    shear = 0;

    for i = 1:length(Loads)

        if Loads(i) ~= 0

            shear = shear + Loads(i);

        end

        SFD(i) = shear;

    end

    % Integrates SFD

    BMD = cumtrapz(x, SFD);

end

%%

function [ ] = VisualizeBridge( csc, tfw, tft, wh, wt, ws, bfw, bft, rw, rh, rbw, rbh)

    % Provides a graphical interpretation of user geometric inputs

    %    Input: cross section changes x cordinates, top flange width, top flange thinckness, web height, web thickness, web spacing,

    %        bottom flange width, bottom flange thickness, etc...

    %    Output: plots polygons of crossections

    for i = 1:length(csc)

        % Calculates basic coordinates of features

        y\_top\_flang\_top\_cord = bft(i) + wh(i) + tft(i);

        y\_top\_flang\_bot\_cord = y\_top\_flang\_top\_cord - tft(i);

        x\_right\_web\_cord = tfw(i) - ((tfw(i) - ws(i)) / 2) + wt(i);

        x\_left\_web\_cord = ((tfw(i) - ws(i)) / 2) - wt(i);

        x\_left\_bot\_flang\_cord = ((tfw(i) - bfw(i)) / 2);

        x\_right\_bot\_flang\_cord = tfw(i) - ((tfw(i) - bfw(i)) / 2);

        outside\_x\_cords = [x\_left\_bot\_flang\_cord x\_left\_bot\_flang\_cord x\_left\_web\_cord x\_left\_web\_cord 0 0 tfw(i) tfw(i) x\_right\_web\_cord x\_right\_web\_cord x\_right\_bot\_flang\_cord x\_right\_bot\_flang\_cord ];

        outside\_y\_cords = [0 bft(i) bft(i) y\_top\_flang\_bot\_cord y\_top\_flang\_bot\_cord y\_top\_flang\_top\_cord y\_top\_flang\_top\_cord y\_top\_flang\_bot\_cord y\_top\_flang\_bot\_cord bft(i) bft(i) 0];

        x\_left\_inside\_web = x\_left\_web\_cord + wt(i);

        x\_right\_inside\_web = x\_right\_web\_cord - wt(i);

        x\_left\_top\_rectangle = x\_left\_inside\_web + rw(i);

        x\_right\_top\_rectangle = x\_right\_inside\_web - rw(i);

        x\_left\_bot\_rectangle = x\_left\_inside\_web + rbw(i);

        x\_right\_bot\_rectangle = x\_right\_inside\_web - rbw(i);

        y\_rectangle\_top = y\_top\_flang\_bot\_cord - rh(i);

        y\_rectangle\_bot = bft(i) + rbh(i);

        inside\_x\_cords = [x\_left\_bot\_rectangle x\_left\_bot\_rectangle x\_left\_inside\_web x\_left\_inside\_web x\_left\_top\_rectangle x\_left\_top\_rectangle x\_right\_top\_rectangle x\_right\_top\_rectangle x\_right\_inside\_web x\_right\_inside\_web x\_right\_bot\_rectangle x\_right\_bot\_rectangle];

        inside\_y\_cords = [bft(i) y\_rectangle\_bot y\_rectangle\_bot y\_rectangle\_top y\_rectangle\_top y\_top\_flang\_bot\_cord y\_top\_flang\_bot\_cord y\_rectangle\_top y\_rectangle\_top y\_rectangle\_bot y\_rectangle\_bot bft(i)];

        % Draws rectangles for features

        cross\_section\_shape = polyshape(outside\_x\_cords, outside\_y\_cords);

        cross\_section\_shape = addboundary(cross\_section\_shape, inside\_x\_cords, inside\_y\_cords);

        % Plots polygons

        subplot(ceil(length(csc)/ 2), ceil(length(csc)/ 2), i)

        plot(cross\_section\_shape)

        % Labels Plots

        if i ~= 1

            title("Cross Section from x = " + csc(i-1) + " to " + csc(i))

        else

            title("Cross Section at x = 0")

        end

        axis equal

    end

end

function out = getVarName(var)

    out = inputname(1);

end

%%

function [ Y\_bar, I, Q, b, heights ] = SectionProperties( csc, tfw, tft, wh, wt, ws, bfw, bft, rtw, rtt, rbw, rbt, n )

    % Calculates important sectional properties. Including but not limited to ybar, I, Q, etc.

    %    Input: cross section changes x cordinates, top flange width, top flange thinckness, web height, web thickness, web spacing,

    %        bottom flange width, bottom flange thickness, etc

    % Output: Sectional Properties at every value of x. Each property is a 1-D array of length n with the exception of b and Q which are n \* max\_height matrices.

    Y\_bar = zeros(1,n);

    I = zeros(1, n);

    for i = 1:length(csc)-1

         heights(csc(i)+1:csc(i+1)) = (tft(i) + wh(i) + bft(i));

    end

    max\_height = ceil(max(heights));

    Q = zeros(n, max\_height);

    b = zeros(n, max\_height);

    for i = 1:length(csc)

        % Areas for features

        A\_top = tfw(i) \* tft(i);

        A\_webs = 2 \* wt(i) \* wh(i);

        A\_bot = bft(i) \* bfw(i);

        A\_r\_top = 2 \* (rtw(i) \* rtt(i));

        A\_r\_bot = 2 \* (rbw(i) \* rbt(i));

        % Local centriods for features

        Y\_top = (tft(i)/2 + wh(i) + bft(i));

        Y\_webs = (wh(i)/2 + bft(i));

        Y\_bot = (bft(i)/2);

        Y\_r\_top = (wh(i) + bft(i) - rtt(i)/2);

        Y\_r\_bot = (bft(i) + rbt(i)/2);

        % I for features

        I\_top = (tfw(i) \* (tft(i) ^ 3)) /12;

        I\_webs = 2 \* (wt(i) \* (wh(i) ^ 3)) /12;

        I\_bot = (bfw(i) \* (bft(i) ^ 3)) /12;

        I\_r\_top = 2 \* (rtw(i) \* (rtt(i) ^ 3)) /12;

        I\_r\_bot = 2 \* (rbw(i) \* (rbt(i) ^ 3)) /12;

        % Calculates I and Y for cross-section

        Y\_section = ((A\_top \* Y\_top) + (A\_webs \* Y\_webs) + (A\_bot \* Y\_bot ) + (A\_r\_top \* Y\_r\_top) + (A\_r\_bot \* Y\_r\_bot)) / (A\_bot + A\_top + A\_webs + A\_r\_bot + A\_r\_top);

        I\_section = (I\_top + (A\_top \* ((Y\_section - Y\_top) ^ 2))) + (I\_webs + (A\_webs \* ((Y\_section - Y\_webs) ^ 2))) + (I\_bot + (A\_bot \* ((Y\_section - Y\_bot) ^ 2))) + (I\_r\_top + (A\_r\_top \* ((Y\_section - Y\_r\_top) ^ 2))) + (I\_r\_bot + (A\_r\_bot \* ((Y\_section - Y\_r\_bot) ^ 2)));

        % Calculates Q, Adds I, Q and Y\_bar for every x value of cross-section

        if i ~= length(csc)

            for j = csc(i):(csc(i+1)-1)

                Y\_bar(j+1) = Y\_section;

                I(j+1) = I\_section;

                for y = 1:((tft(i) + wh(i) + bft(i)));

                    if y <= bft(i)

                        b(j+1, y) = bft(i);

                        sub\_area = y \* bfw(i);

                        sub\_centriod = (bft(i) - y)/2;

                        d = abs(Y\_bar(csc(i) + 1) - sub\_centriod);

                        Q(j+1, y) = sub\_area \* d;

                    elseif bft(i) < y && y <= (bft(i) + rbt(i))

                        b(j+1, y) = 2 \* (rbw(i) + wt(i));

                        web\_height = y - bft(i);

                        sub\_area = A\_bot + (2 \* web\_height  \* (wt(i) + rbw(i)));

                        sub\_centriod = ((A\_bot \* Y\_bot) + ((2 \* web\_height \* wt(i)) \* ((web\_height / 2) + bft(i)))) / sub\_area;

                        d = abs(Y\_bar(csc(i) + 1) - sub\_centriod);

                        Q(j+1, y) = sub\_area \* d;

                    elseif (bft(i) + rbt(i)) < y && y <= (bft(i) + wh(i) - rtt(i))

                        b(j+1, y) = 2 \* wt(i);

                        web\_height = y - bft(i) - rbt(i);

                        sub\_area = A\_bot + (2 \* web\_height  \* wt(i)) + A\_r\_bot;

                        sub\_centriod = ((A\_bot \* Y\_bot) + ((2 \* web\_height \* wt(i)) \* ((web\_height / 2) + bft(i) + rbt(i))) + (A\_r\_bot \* Y\_r\_bot)) / sub\_area;

                        d = abs(Y\_bar(csc(i) + 1) - sub\_centriod);

                        Q(j+1, y) = sub\_area \* d;

                    elseif  (bft(i) + wh(i) - rtt(i)) < y && y <= (bft(i) + wh(i))

                        b(j+1, y) = 2 \* (rtw(i) + wt(i));

                        web\_height = y - wh(i) + rtt(i) - bft(i);

                        main\_web\_area =  (2 \* (wh(i) - rtt(i)) \* wt(i));

                        sub\_area = A\_bot + (2 \* web\_height  \* (wt(i) + rtw(i))) + A\_r\_bot + main\_web\_area;

                        sub\_centriod = ((A\_bot \* Y\_bot) + (2 \* web\_height \* (wt(i) + rtw(i)) \* ((web\_height / 2) + bft(i) + (wh(i) - rtt(i)))) + (A\_r\_bot \* Y\_r\_bot) + (main\_web\_area \* (((wh(i) - rtt(i))/2) + bft(i)))) / sub\_area;

                        d = abs(Y\_bar(csc(i) + 1) - sub\_centriod);

                        Q(j+1, y) = sub\_area \* d;

                    else

                        b(j+1, y) = tfw(i);

                        top\_height = y - wh(i) - bft(i);

                        sub\_area = A\_bot + A\_webs + (top\_height \* tfw(i)) + A\_r\_bot + A\_r\_top;

                        sub\_centriod = ((A\_bot \* Y\_bot) + (A\_webs \* Y\_webs) + (A\_r\_top \* Y\_r\_top) + (A\_r\_bot \* Y\_r\_bot) + (top\_height \* tfw(i) \* ((top\_height / 2) + wh(i) + bft(i)))) / sub\_area;

                        d = abs(Y\_bar(csc(i) + 1) - sub\_centriod);

                        Q(j+1, y) = sub\_area \* d;

                    end

                end

            end

        else

            for j = csc(i):(n)

                Y\_bar(j) = Y\_section;

                I(j) = I\_section;

                for y = 1:((tft(i) + wh(i) + bft(i)))

                    if y <= bft(i)

                        b(j, y) = bft(i);

                        sub\_area = y \* bfw(i);

                        sub\_centriod = (bft(i) - y)/2;

                        d = abs(Y\_bar(csc(i)) - sub\_centriod);

                        Q(j, y) = sub\_area \* d;

                    elseif bft(i) < y && y <= (bft(i) + rbt(i))

                        b(j, y) = 2 \* (rbw(i) + wt(i));

                        web\_height = y - bft(i);

                        sub\_area = A\_bot + (2 \* web\_height  \* (wt(i) + rbw(i)));

                        sub\_centriod = ((A\_bot \* Y\_bot) + ((2 \* web\_height \* wt(i)) \* ((web\_height / 2) + bft(i)))) / sub\_area;

                        d = abs(Y\_bar(csc(i)) - sub\_centriod);

                        Q(j, y) = sub\_area \* d;

                    elseif (bft(i) + rbt(i)) < y && y <= (bft(i) + wh(i) - rtt(i))

                        b(j, y) = 2 \* wt(i);

                        web\_height = y - bft(i) - rbt(i);

                        sub\_area = A\_bot + (2 \* web\_height  \* wt(i)) + A\_r\_bot;

                        sub\_centriod = ((A\_bot \* Y\_bot) + ((2 \* web\_height \* wt(i)) \* ((web\_height / 2) + bft(i) + rbt(i))) + (A\_r\_bot \* Y\_r\_bot)) / sub\_area;

                        d = abs(Y\_bar(csc(i)) - sub\_centriod);

                        Q(j, y) = sub\_area \* d;

                    elseif  (bft(i) + wh(i) - rtt(i)) < y && y <= (bft(i) + wh(i))

                        b(j, y) = 2 \* (rtw(i) + wt(i));

                        web\_height = y - wh(i) + rtt(i) - bft(i);

                        main\_web\_area =  (2 \* (wh(i) - rtt(i)) \* wt(i));

                        sub\_area = A\_bot + (2 \* web\_height  \* (wt(i) + rtw(i))) + A\_r\_bot + main\_web\_area;

                        sub\_centriod = ((A\_bot \* Y\_bot) + (2 \* web\_height \* (wt(i) + rtw(i)) \* ((web\_height / 2) + bft(i) + (wh(i) - rtt(i)))) + (A\_r\_bot \* Y\_r\_bot) + (main\_web\_area \* (((wh(i) - rtt(i))/2) + bft(i)))) / sub\_area;

                        d = abs(Y\_bar(csc(i)) - sub\_centriod);

                        Q(j, y) = sub\_area \* d;

                    else

                        b(j, y) = tfw(i);

                        top\_height = y - wh(i) - bft(i);

                        sub\_area = A\_bot + A\_webs + (top\_height \* tfw(i)) + A\_r\_bot + A\_r\_top;

                        sub\_centriod = ((A\_bot \* Y\_bot) + (A\_webs \* Y\_webs) + (A\_r\_top \* Y\_r\_top) + (A\_r\_bot \* Y\_r\_bot) + (top\_height \* tfw(i) \* ((top\_height / 2) + wh(i) + bft(i)))) / sub\_area;

                        d = abs(Y\_bar(csc(i)) - sub\_centriod);

                        Q(j, y) = sub\_area \* d;

                    end

                end

            end

        end

    end

end

function [ V\_fail ] = Vfail( I, b, Y\_bar, TauU, Q)

    % Calculates shear forces at every value of x that would cause a matboard shear failure

    %   Input: Sectional Properties (list of 1-D arrays), TauU (scalar material property)

    %   Output: V\_fail a 1-D array of length n

    Qcent = zeros(1,length(Y\_bar));

    bcent = zeros(1,length(Y\_bar));

    for i = 1:length(Y\_bar)

        Qcent(i) = Q(i, round(Y\_bar(i)));

        bcent(i) = b(i, round(Y\_bar(i)));

    end

    V\_fail = TauU .\* I .\* bcent ./ Qcent;

end

function [ V\_Buck ] = VfailBuck(csc, wt, ds, wh, E, mu, n, I, b, Y\_bar, Q)

    % Calculates shear forces at every value of x that would cause a shear buckling failure in the web

    %   Input: Sectional Properties (list of 1-D arrays), E, mu (material property), etc...

    %   Output: V\_Buck a 1-D array of length n

    factor = (5 \* (pi ^ 2) \* E ) / (12 \* (1 - (mu ^ 2) ));

    a\_values = zeros(1, n);

    web\_thicknesses = zeros(1, n);

    web\_heights = zeros(1, n);

    % Finds web thicknesses, web heights and a values for cross sections

    for i = 1:(length(csc)-1)

        web\_thicknesses((csc(i)+1):(csc(i+1))) = wt(i);

        web\_heights((csc(i)+1):(csc(i+1))) = wh(i);

        a\_values((csc(i)+1):(csc(i+1))) = ds(i+1) - ds(i);

    end

    % Finds TauCrit

    TauCrit = factor .\* (((web\_thicknesses ./ web\_heights) .^ 2) + ((web\_thicknesses ./ a\_values) .^ 2));

    % Calls V\_fail function using tau crit value above

    [ V\_Buck ] = Vfail( I, b, Y\_bar, TauCrit, Q);

end

function [] = VisulizePL(x, P, Pfail, SFD, BMD, V\_fail, V\_buck, M\_MatT, M\_MatC, M\_Buck1, M\_Buck2, M\_Buck3, V\_GlueTF, V\_GlueBF, V\_GlueTW, V\_GlueBW)

    % Plots all graphs (i.e BMD, SFD, Failure loads etc...)

    %   Inputs: All failure condition loads as arrays of length n, SFD, BMD,

    %   Outputs: Plots of above values/arrays

    figure('WindowState', 'maximized')

    % Plots SFD for Pfail and lables

    subplot(2,3,1);

    hold on;

    plot(x, SFD);

    set(gca, 'XAxisLocation', 'bottom', 'YAxisLocation', 'origin');

    title(("SFD from P = " + P + "N, Pfail = " +  Pfail + "N"))

    xlim([0,x(end)])

    grid on;

    grid minor

    box on;

    yline(0)

    xlabel('Position on Bridge (mm)')

    ylabel('Shear Force (N)')

    % Plots BMD vs V\_buck and lables

    subplot(2,3,2);

    hold on;

    plot(x, SFD);

    plot(x, V\_fail(V\_fail~=0), 'r-');

    plot(x, -V\_fail(V\_fail~=0), 'r-');

    plot(x, V\_GlueTF(V\_GlueTF~=0), 'g-')

    plot(x, -V\_GlueTF(V\_GlueTF~=0), 'g-')

    plot(x, V\_GlueBF(V\_GlueBF~=0), 'b-')

    plot(x, -V\_GlueBF(V\_GlueBF~=0), 'b-')

    plot(x, V\_GlueTW(V\_GlueTW~=0), 'm-')

    plot(x, -V\_GlueTW(V\_GlueTW~=0), 'm-')

    plot(x, V\_GlueBW(V\_GlueBW~=0), 'c-')

    plot(x, -V\_GlueBW(V\_GlueBW~=0), 'c-')

    set(gca, 'XAxisLocation', 'bottom', 'YAxisLocation', 'origin');

    title('SFD vs Material Shear Failure')

    xlim([0,x(end)])

    grid on;

    grid minor

    box on;

    ylim([-max(V\_fail) - 100, max(V\_fail) + 100])

    yline(0)

    legend('', '', 'Matboard Shear Failure', '', 'Glue Shear (TF) Failure', '',  'Glue Shear (BF) Failure', '', 'Glue Shear (TW) Failure', '', 'Glue Shear (BW) Failure', 'FontSize', 7)

    xlabel('Position on Bridge (mm)')

    ylabel('Shear Force (N)')

    % Plots SFD vs V\_buck and lables

    subplot(2,3,3);

    hold on;

    plot(x, SFD);

    plot(x, V\_buck, 'r-');

    plot(x, -V\_buck, 'r-');

    set(gca, 'XAxisLocation', 'bottom', 'YAxisLocation', 'origin');

    title('SFD vs Shear Buckling Failure')

    xlim([0,x(end)]);

    ylim([-max(V\_buck) - 100, max(V\_buck) + 100]);

    yline(0);

    legend('', '', 'Web Shear Buckling Failure', '', 'FontSize', 7)

    xlabel('Position on Bridge (mm)')

    ylabel('Shear Force (N)')

    grid on;

    grid minor;

    box on;

    % Plots BMD for Pfail and lables

    subplot(2,3,4);

    hold on;

    plot(x, BMD);

    set(gca, 'XAxisLocation', 'bottom', 'YAxisLocation', 'origin', 'YDir', 'reverse');

    title("BMD from P = " + P + "N, Pfail = " +  Pfail + "N")

    xlim([0,x(end)])

    grid on;

    grid minor

    box on;

    yline(0)

    xlabel('Position on Bridge (mm)')

    ylabel('Bending Moment (Nmm)')

    % Plots BMD vs Material Moment Failures and lables

    subplot(2,3,5);

    hold on;

    plot(x, BMD);

    plot(x, M\_MatT)

    plot(x, M\_MatC)

    set(gca, 'XAxisLocation', 'bottom', 'YAxisLocation', 'origin', 'YDir', 'reverse');

    title('BMD vs Material Moment Failures ')

    xlim([0,x(end)])

    grid on;

    grid minor

    box on;

    yline(0)

    legend('', 'Matboard Tension Failure', 'Matboard Compression Failure', '', 'Location', 'Northwest', 'FontSize', 7)

    xlabel('Position on Bridge (mm)')

    ylabel('Bending Moment (Nmm)')

    % Plots Moment Buckling and lables

    subplot(2,3,6);

    hold on;

    plot(x, BMD);

    plot(x, M\_Buck1);

    plot(x, M\_Buck2);

    plot(x, M\_Buck3);

    set(gca, 'XAxisLocation', 'bottom', 'YAxisLocation', 'origin', 'YDir', 'reverse');

    title('BMD vs Material Buckling Failures ')

    xlim([0,x(end)])

    grid on;

    grid minor

    box on;

    yline(0)

    legend({'', 'Mid Flange Buckling', 'Side Flange Buclking', 'Web Compression Buckling', ''}, 'Location', 'Northwest', 'FontSize', 7)

    xlabel('Position on Bridge (mm)')

    ylabel('Bending Moment (Nmm)')

end

function [ M\_MatT ] = MfailMatT( I, Y\_bar, heights, SigT, BMD )

    % Calculates bending moments at every value of x that would cause a matboard tension failure

    %   Input: Sectional Properties (list of 1-D arrays), SigT (material property), BMD (1-D array), etc...

    %   Output: M\_MatT a 1-D array of length n

    for i = 1 : length(BMD)

        if BMD(i) > 0 % If the moment is positive, the tension failure will be at the bottom

            M\_MatT(i) = SigT \* I(i) / (Y\_bar(i));

        elseif BMD(i) < 0 % If the moment is negative, the tension failure will be at the top

            M\_MatT(i) = -SigT \* I(i) / (heights(i) - Y\_bar(i));

        end

    end

end

function [ M\_MatC ] = MfailMatC( I, Y\_bar, heights, SigC, BMD ) % Similar to MfailMatT

    % Calculates bending moments at every value of x that would cause a matboard Compression failure

    %   Input: Sectional Properties (list of 1-D arrays), SigC (material property), BMD (1-D array), etc...

    %   Output: M\_MatC a 1-D array of length n

    for i = 1 : length(BMD)

        if BMD(i) > 0 % If the moment is positive, the compression failure will be at the top

            M\_MatC(i) = SigC \* I(i) / (heights(i) - Y\_bar(i));

        elseif BMD(i) < 0 % If the moment is negative, the compression failure will be at the bottom

            M\_MatC(i) = -SigC \* I(i) / Y\_bar(i);

        end

    end

end

function [ M\_Buck1 M\_Buck2 M\_Buck3 ] = MfailBuck( csc, bfw, bft, tfw, tft, ws, wt, heights, Y\_bar, I, E, mu, BMD)

    % Calculates bending moments at every value of x that would cause a buckling failure

    % Input: Sectional Properties (list of 1-D arrays), E, mu (material property), BMD (1-D array)

    % Output: M\_Buck1, M\_Buck2, M\_Buck3 each 1-D array of length n

    for c = 1:3

        % Case 1 buckling

        if c == 1

            factor = (4 \* (pi ^ 2) \* E) / (12 \* (1 - (mu ^ 2)));

            for i = 1:length(BMD);

                z = find(csc <= i, 1, 'last');

                if BMD(i) < 0 % if moment negative, compression on bottom

                    t = bft(z);

                    y = - Y\_bar(i);

                    b =  ws(z);

                    M\_Buck1(i) = (factor \* ((t / b) ^ 2)) \* I(i) / y;

                elseif BMD(i) > 0 % if moment positive, compression on top

                    t = tft(z);

                    y = heights(i) - Y\_bar(i);

                    b = ws(z);

                    M\_Buck1(i) = (factor \* ((t / b) ^ 2)) \* I(i) / y;

                end

            end

        % Case 2 buckling

        elseif c == 2

            factor = (0.425 \* (pi ^ 2) \* E) / (12 \* (1 - (mu ^ 2)));

            for i = 1:length(BMD);

                z = find(csc <= i, 1, 'last');

                if BMD(i) < 0 % if moment negative, compression on bottom

                    t = bft(z);

                    y = - Y\_bar(i);

                    b = (bfw(z) - (2 \* wt(z)) - ws(z)) / 2;

                    if b ~= 0 && bfw(z) ~= 0

                        M\_Buck2(i) = (factor \* ((t / b) ^ 2)) \* I(i) / y;

                    else

                        M\_Buck2(i) = 0;

                    end

                elseif BMD(i) > 0 % if moment positive, compression on top

                    t = tft(z);

                    y = heights(i) - Y\_bar(i);

                    b = (tfw(z) - ws(z) - (2 \* wt(z))) / 2;

                    if b ~= 0 && tfw(z) ~= 0

                        M\_Buck2(i) = (factor \* ((t / b) ^ 2)) \* I(i) / y;

                    else

                        M\_Buck2(i) = 0;

                    end

                end

            end

        % Case 3 Buckling

        else

            factor = (6 \* (pi ^ 2) \* E) / (12 \* (1 - (mu ^ 2)));

            for i = 1:length(BMD);

                z = find(csc <= i, 1,'last');

                if BMD(i) < 0 % if moment negative, compression on bottom

                    t = wt(z);

                    y = - Y\_bar(i);

                    b = abs(y) - bft(z);

                    if b ~= 0

                        M\_Buck3(i) = (factor \* ((t / b) ^ 2)) \* I(i) / y;

                    end

                elseif BMD(i) > 0 % if moment positive, compression on top

                    t = wt(z);

                    y = heights(i) - Y\_bar(i);

                    b = abs(y) - tft(z);

                    if b ~= 0

                        M\_Buck3(i) = (factor \* ((t / b) ^ 2)) \* I(i) / y;

                    end

                end

            end

        end

    end

end

function [ Pf, failure\_mode ] = FailLoad( P, SFD, BMD, V\_Mat, V\_Buck, M\_MatT, M\_MatC, M\_Buck1, M\_Buck2, M\_Buck3, V\_GlueTF, V\_GlueBF, V\_GlueTW, V\_GlueBW, write\_to\_xls, design\_name )

    % Calculates the magnitude of the load P that will cause one of the failure mechanisms to occur as well as the failure mode, and outputs all failure load value to excel sheet

    %   Input: SFD, BMD under the currently applied points loads (P) (each 1-D array of length n)

    %       {V\_Mat, V\_Glue, … M\_MatT, M\_MatC, … } (each 1-D array of length n), current load P, write\_to\_xls (a boolean value) and design\_name (a string)

    %   Output: Failure Load value Pf, and mode of failure (failure\_mode) and writes to an excel sheet (optional) if write\_to\_xls is true

    % Calculates failure loads

    SFD\_new = SFD ./ abs(P);

    BMD\_new = BMD ./ abs(P);

    fail\_v\_mat = V\_Mat ./ SFD\_new;

    fail\_v\_buck = V\_Buck ./ SFD\_new;

    fail\_m\_matt = M\_MatT ./ BMD\_new;

    fail\_m\_matc = M\_MatC ./ BMD\_new;

    fail\_m\_buck1 = M\_Buck1 ./ BMD\_new;

    fail\_m\_buck2 = M\_Buck2 ./ BMD\_new;

    fail\_m\_buck3 = M\_Buck3 ./ BMD\_new;

    fail\_v\_gluetf = V\_GlueTF ./ SFD\_new;

    fail\_v\_gluebf = V\_GlueBF ./ SFD\_new;

    fail\_v\_gluetw = V\_GlueTW ./ SFD\_new;

    fail\_v\_gluebw = V\_GlueBW ./ SFD\_new;

    % Finds lowest failure load for each array

    matt = min(fail\_m\_matt(fail\_m\_matt>0));

    matc = min(fail\_m\_matc(fail\_m\_matc>0));

    buck1 = min(fail\_m\_buck1(fail\_m\_buck1>0));

    buck2 = min(fail\_m\_buck2(fail\_m\_buck2>0));

    buck3 = min(fail\_m\_buck3(fail\_m\_buck3>0));

    vmat = min(fail\_v\_mat(fail\_v\_mat>0));

    vbuck = min(fail\_v\_buck(fail\_v\_buck>0));

    vgluetf = min(fail\_v\_gluetf(fail\_v\_gluetf>0));

    vgluebf = min(fail\_v\_gluebf(fail\_v\_gluebf>0));

    vgluetw = min(fail\_v\_gluetw(fail\_v\_gluetw>0));

    vgluebw = min(fail\_v\_gluebw(fail\_v\_gluebw>0));

    % Optionally write to .xls file if write\_to\_xls is true

    if write\_to\_xls == true

        filename = design\_name + '.xls';

        writematrix(design\_name, filename);

        writematrix("Moment Matboard Tension Failure", filename, 'range', 'A2')

        writematrix(matt, filename, 'range', 'B2:C2');

        writematrix("Moment Matboard Compression Failure", filename, 'range', 'A3')

        writematrix(matc, filename, 'range', 'B3:C3');

        writematrix("Case 1 Buckling of Top or Bottom Flange (center part between webs)", filename, 'range', 'A4')

        writematrix(buck1, filename, 'range', 'B4:C4');

        writematrix("Case 2 Buckling of Top or Bottom Flange (part that sticks out past the webs)", filename, 'range', 'A5')

        writematrix(buck2, filename, 'range', 'B5:C5');

        writematrix("Case 3 Buckling of Left and Right Webs", filename, 'range', 'A6')

        writematrix(buck3, filename, 'range', 'B6:C6');

        writematrix("Matboard Shear Failure", filename, 'range', 'A7')

        writematrix(vmat, filename, 'range', 'B7:C7');

        writematrix("Matboard Shear Buckling Failure", filename, 'range', 'A8')

        writematrix(vbuck, filename, 'range', 'B8:C8');

        writematrix("Shear Glue Failure of Top Flange", filename, 'range', 'A9')

        writematrix(vgluetf, filename, 'range', 'B9:C9');

        writematrix("Shear Glue Failure of Bottom Flange", filename, 'range', 'A10')

        writematrix(vgluebf, filename, 'range', 'B10:C10');

        writematrix("Shear Glue Failure of Top Flange/Web Connection", filename, 'range', 'A11')

        writematrix(vgluetw, filename, 'range', 'B11:C11');

        writematrix("Shear Glue Failure of Bottom Flange/Web Connection", filename, 'range', 'A12')

        writematrix(vgluebw, filename, 'range', 'B12:C12');

    end

    % Outputs failure load and failure condition

    results = ["Tension Failure", "Compression Failure", "Center Flange Buckling", "Protruding Flange Buckling", "Web Buckling", "V material", "vbuck", "Glue Shear tf", "Glue Shear bottom", "glue shear top width", "glue shear bottom width"];

    [Pf, fail\_index] = min([abs(matt) abs(matc) abs(buck1) abs(buck2) abs(buck3) abs(vmat) abs(vbuck) abs(vgluetf) abs(vgluebf) abs(vgluetw) abs(vgluebw)]);

    % if fail\_index = 1: matt, 2: matc, 3: buck1, 4: buck2, 5: buck3, 6: vmat, 7: vbuck, 8: vgluetf, 9: vgluebf, 10: vgluetw, 11: vgluebw

    failure\_mode = results(fail\_index);

end

function [material\_ok] = MaterialCheck( xc, tfw, tft, wh, wt, ws, bfw, bft, rw, rh, rbw, rbh)

    % Checks if current design overuses material. Note does not take into account diaphragms and uses factor of safety of 1.1.

    %   Input: cross-sectional dimensions

    %   Output: matrial\_ok (true or false) based on above conditions.

    matboard\_thickness = 1.27;

    top = tfw .\* tft;

    top\_glues = 2 \* rh .\* rw;

    bottom\_glues = 2 \* rbh .\* rbw;

    webs = 2 \* wh .\* wt;

    bottom = bfw .\* bft;

    total\_cross\_section\_width = (top + top\_glues + bottom\_glues + webs + bottom)/matboard\_thickness;

    safety\_factor = .9;

    total\_area  = 0;

    for i = 2:length(xc)

        delta\_x = xc(i) - xc(i - 1);

        total\_area  = total\_area + delta\_x \* total\_cross\_section\_width(i);

    end

    total\_material\_area = 813 \* 1016;

    material\_ok = total\_area < total\_material\_area \* safety\_factor;

end

function [ V\_GlueTF V\_GlueBF V\_GlueTW V\_GlueBW ] = VglueFail(I, Q, b, TauG, tft, bft, heights, csc)

    % Calculates glue failures for top flange, bottom flange and top and bottom web/flange joints

    %   Input: Sectional properties (I, Q, b, sectional\_height, etc), flange dimensions, cross-section changes, etc...

    %   Output: V\_GlueTF (top flange glue failure), V\_GlueBF (bottom flange glue failure), V\_GlueTW (Top flange/web joint glue failure)

    %           V\_GlueBW (bottom flange/web joint glue failure) all arrays of length n.

    V\_GlueBF = NaN(1,length(heights));

    V\_GlueTF = NaN(1,length(heights));

    V\_GlueTW = NaN(1,length(heights));

    V\_GlueBW = NaN(1,length(heights));

    for i = 1 : length(heights)

        z = find(csc <= i, 1, 'last');

        % Top Flange Glue

        if (tft(z) / 1.27) > 1

            glue\_y = ceil(heights(i) - 1.27);

            V\_GlueTF(i) = (TauG \* b(i, glue\_y) \* I(i)) / ((Q(i, floor(heights(i) - 1.27)) + Q(i, ceil(heights(i) - 1.27))) / 2);

        end

        % Bottom Flange Glue

        if (bft(z) / 1.27) > 1

            glue\_y = floor(1.27);

            V\_GlueBF(i) = (TauG \* b(i, glue\_y) \* I(i)) / ((Q(i, floor(1.27)) + Q(i, ceil(1.27)))/2);

        end

        % Top web/flange glue (there will always be a top flange therefore no if statement needed)

        %   (note: rounds up heights(i) - tft(i) in order to get smaller b value and larger Q value in order to get lower V\_GlueTW for safety

        V\_GlueTW(i) = (TauG \* b(i, floor(heights(i) - tft(z))) \* I(i)) / ((Q(i, floor(heights(i) - tft(z))) + Q(i, ceil(heights(i) - tft(z)))) / 2);

        % Bottom web/flange glue (note: rounds up bft(i) in order to get smaller b value and larger Q value in order to get lower V\_GlueBW for safety

        if bft(z) > 0

            V\_GlueBW(i) = (TauG \* b(i, ceil(bft(z))) \* I(i)) / ((Q(i, floor(bft(z))) + Q(i, ceil(bft(z))))/2);

        end

    end

end