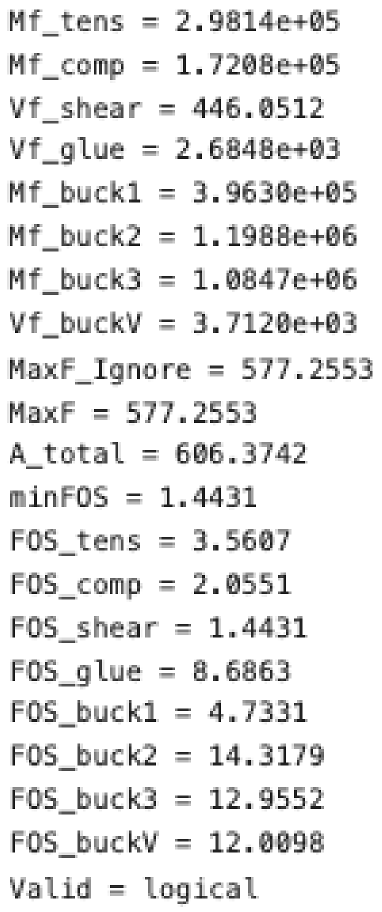
Code block that outputs the 8 factors of safety and also max moments and shear forces for your cross section design. It also checks if the design will fit on a matboard (uses a function that is defined at the end of the matlab script to do all this). Set to Test Case 1 Base 1

```
%%% All lengths are in mm, all forces are in N, all moments are in Nmm

h = 120; % height of cross section (mm)
mat_t = 1.27; % Thickness of matboard (mm)
glue_t = 11.27; % Glue tab width (mm)
bfw = 80; % Bottom flange width (mm)
tft = 2*1.27;    % top flange thickness
tfw = 110;        % top flange width
diaphram_spacing = 1250/10;    % length divided by 10 (10 = number of
diaphrams)
bft = 0;    % bottom flange thickness

[ybar, I, Q_glue, Q_cent, Mf_tens, Mf_comp, Vf_shear, Vf_glue, Mf_buck1,
Mf_buck2, Mf_buck3, Vf_buckV, MaxF_Ignore, MaxF, A_total, minFOS, FOS_tens,
FOS_comp, FOS_shear, FOS_glue, FOS_buck1, FOS_buck2, FOS_buck3, FOS_buckV]
= test_fn(h, mat_t, glue_t, bfw, bft, tft, tfw, diaphram_spacing)

Valid = A_total > 813 - (12 * hw * 77 * (10 + h - y)) / 1250      % boolean
variable is true if the cross section area can fit on the matboard with
enough area left over for 10 diaphrams
```

An example of a possible output is shown below.

- **Valid = logical means that the bridge can fit on the matboard. Mf and Vf correspond to maximum moment and maximum shear force (respectivley).**
- **Buck 1, 2, 3 are the three cases for thin plate buckling. Case 1 = top flange between webs. Case 2 = part of top flange protruding , to the right of the right web and left of left web. Case 3 = webs.**
- **Buck V is shear buckling**
- **Buck glue is shear stress failure at the glue locatoon**
- **All units are in N or Nmm**



```
Mf_tens = 2.9814e+05
Mf_comp = 1.7208e+05
Vf_shear = 446.0512
Vf_glue = 2.6848e+03
Mf_buck1 = 3.9630e+05
Mf_buck2 = 1.1988e+06
Mf_buck3 = 1.0847e+06
Vf_buckV = 3.7120e+03
MaxF_Ignore = 577.2553
MaxF = 577.2553
A_total = 606.3742
minFOS = 1.4431
FOS_tens = 3.5607
FOS_comp = 2.0551
FOS_shear = 1.4431
FOS_glue = 8.6863
FOS_buck1 = 4.7331
FOS_buck2 = 14.3179
FOS_buck3 = 12.9552
FOS_buckV = 12.0098
Valid = logical
```

SFD and BMD builder (currenty loaded with Test Case 1 Base Case 1). Based on the the framework from Page 17 of CIV102 Matboard Bridge Design Project [1]

[1] Bentz, Evan. *CIV102 Matboard Bridge Design Project*. University of Toronto, https://q.utoronto.ca/courses/363496/assignments/1403151.

```
% bridge
L = 1200; % bridge length (mm)
n = 100; % number of segments (to split x axis into)
x = linspace(0, L, n+1); % x axis

% train
P = 400; % train weight (N)
x_train_initial = [0, -176, -340, -516, -680, -856]; % initial position of
train wheels (assume the train begins moving -- left to right -- when the
front wheel is on the leftmost support)
P_train = [66.6,66.6, 66.6,66.6,66.6,66.6]; % weight of each wheel
positions = 1000; % number of train positions
```

```
step = L / (positions - 1); % step  (mm)


SFDi = zeros(positions, n+1); % stores shear force diagram for every
position
BMDi = zeros(positions, n+1); % same but for bending moment diagram

% iterate over all positions
for position = 1:positions
    x_train = x_train_initial + step * (position - 1);   % move train

    valid_indices = (x_train >= 0) & (x_train <= L);   % indicies for train
locatio that are on bridge
    x_train = x_train(valid_indices);
    P_train_valid = P_train(valid_indices);

    % for python users, (x_train >= 0) & (x_train <= L) basically is all
    % the indicies in x_train where the conditions (x_train >= 0) and
(x_train <= L)
    % are both true. And we select the weights and positions at those
    % indicies.

    % reactions forces at the support for this position

    R_A = sum(P_train_valid .* (L - x_train)) / L; % Reaction at A. We set
the net moment at B equal to zero.
    R_B = sum(P_train_valid) - R_A; % Reaction at B. Found by setting net
force in y direction to zero

    % SFD and BMD for this position

    % SFD

    for j = 1:n+1   % iterate through every point on the bridge
        SF = R_A;        % we start with the reaction force at A and
subtract shear force due to train wheels at every point before or at the
point on the bridge to get the shear force at that point
        for k = 1:length(x_train)
            if x(j) >= x_train(k)
                SF = SF - P_train_valid(k); % subtract load contributions
if the location of the point on the bridge is greater than the location of
the load
            end
        end
        SFDi(position, j) = SF; % store shear force diagram for this positon


    % BMD
```

```
        BM = R_A * x(j);     % same logic as the SFD calculation, but we
instead begin at the moment at the point on the bridge due to the reaction
force at A and subtract the moment caused by the train loads
        for k = 1:length(x_train)
            if x(j) >= x_train(k)
                BM = BM - P_train_valid(k) * (x(j) - x_train(k)); %
subtract moment caused at the point by train loads before the point on the
bridge
            end
        end
        BMDi(position, j) = BM; % store bending moment diagram for this
position
    end
end

% SFD/BMD envelope
SFD = max(abs(SFDi), [], 1);     % absolute value ...
BMD = max(BMDi, [], 1);
plot(x, BMD)
plot(x, SFD)

Max_Shear = max(SFD)
Max_Moment = max(BMD)
```

Function used in first block to compute the 8 factors of safety, maximum train weight the bridge is able to resist, and the maximum bending moments and shear forces: [Mf_tens, Mf_comp, Vf_shear, Vf_glue, Mf_buck1, Mf_buck2, Mf_buck3, Vf_buckV, MaxF_Ignore, MaxF, A_total, minFOS, FOS_tens, FOS_comp, FOS_shear, FOS_glue, FOS_buck1, FOS_buck2, FOS_buck3, FOS_buckV]

In other words, it yields outputs like

```
Mf_tens = 2.9814e+05
Mf_comp = 1.7208e+05
Vf_shear = 446.0512
Vf_glue = 2.6848e+03
Mf_buck1 = 3.9630e+05
Mf_buck2 = 1.1988e+06
Mf_buck3 = 1.0847e+06
Vf_buckV = 3.7120e+03
MaxF_Ignore = 577.2553
MaxF = 577.2553
A_total = 606.3742
minFOS = 1.4431
FOS_tens = 3.5607
FOS_comp = 2.0551
FOS_shear = 1.4431
FOS_glue = 8.6863
FOS_buck1 = 4.7331
FOS_buck2 = 14.3179
FOS_buck3 = 12.9552
FOS_buckV = 12.0098
Valid = logical
```

```
function [ybar, I, Qglue, Qcent, Mf_tens, Mf_comp, Vf_shear, Vf_glue,
Mf_buck1, Mf_buck2, Mf_buck3, Vf_buckV, MaxF_Ignore, MaxF, A_total, minFOS,
FOS_tens, FOS_comp, FOS_shear, FOS_glue, FOS_buck1, FOS_buck2, FOS_buck3,
FOS_buckV] = test_fn(h, mat_t, glue_t, bfw, bft, tft, tfw, diaphram_spacing)
% bridge
L = 1250; % bridge length (mm)
n = 1250; % number of segments (to split x axis into)
x = linspace(0, L, n+1); % x axis
a = diaphram_spacing;

% train
P = 452; % train weight (N)
x_train_initial = [0, -176, -340, -516, -680, -856]; % initial position of
train wheels (assume the train begins moving -- left to right -- when the
front wheel is on the leftmost support)
P_train = [67.5,67.5,67.5, 67.5,91, 91]; % weight of each wheel
positions = 100; % number of train positions
step = L / (positions - 1); % step  (mm)
```

```
SFDi = zeros(positions, n+1); % stores shear force diagram for every
position
BMDi = zeros(positions, n+1); % same but for bending moment diagram

% iterate over all positions
for position = 1:positions
    x_train = x_train_initial + step * (position - 1);   % move train

    x_train = x_train((x_train >= 0) & (x_train <= L));     % choose train
positions that are on the bridge
    P_train_valid = P_train((x_train >= 0) & (x_train <= L));    % choose
train weights that are on the bridge

    % for python users, (x_train >= 0) & (x_train <= L) basically is all
    % the indicies in x_train where the conditions (x_train >= 0) and
(x_train <= L)
    % are both true. And we select the weights and positions at those
    % indicies.

    % reactions forces at the support for this position

    R_A = sum(P_train_valid .* (L - x_train)) / L; % Reaction at A. We set
the net moment at B equal to zero.
    R_B = sum(P_train_valid) - R_A; % Reaction at B. Found by setting net
force in y direction to zero

    % SFD and BMD for this position

    % SFD

    for j = 1:n+1   % iterate through every point on the bridge
        SF = R_A;        % we start with the reaction force at A and
subtract shear force due to train wheels at every point before or at the
point on the bridge to get the shear force at that point
        for k = 1:length(x_train)
            if x(j) >= x_train(k)
                SF = SF - P_train_valid(k); % subtract load contributions
if the location of the point on the bridge is greater than the location of
the load
            end
        end
        SFDi(position, j) = SF; % store shear force diagram for this positon



    % BMD

        BM = R_A * x(j);     % same logic as the SFD calculation, but we
instead begin at the moment at the point on the bridge due to the reaction
force at A and subtract the moment caused by the train loads
        for k = 1:length(x_train)
```

```
            if x(j) >= x_train(k)
                BM = BM - P_train_valid(k) * (x(j) - x_train(k)); %
subtract moment caused at the point by train loads before the point on the
bridge
            end
        end
        BMDi(position, j) = BM; % store bending moment diagram for this
position
    end
end

% SFD/BMD envelope
SFD = max(abs(SFDi), [], 1);     % absolute value ...
BMD = max(BMDi, [], 1);
plot(x, BMD)
plot(x, SFD)

%%%%% Cross Section %%%%%%

hw = h - tft - bft; % web height (mm)

% Centroid
A_total = tfw * tft + bfw * bft + 2 * (glue_t * 1.27) + 2 * (mat_t * hw);

ybar = ( ...
    tfw * tft * (h - tft / 2) + ... % top flange
    bfw * bft * (bft / 2) + ... % bottom flange
    2 * (glue_t * 1.27) * (h - tft - 1.27 / 2) + ... % glue tabs
    2 * (mat_t * hw) * (hw / 2 + bft) ... % Webs
    ) / A_total;

% Second Moment of Area I
I = ( ...
    (tfw * tft^3) / 12 + tfw * tft * ((h - tft / 2) - ybar)^2 + ... % top
flange
    (bfw * bft^3) / 12 + bfw * bft * (ybar - bft / 2)^2 + ... % bottom
flange
    2 * ((glue_t * 1.27^3) / 12 + glue_t * 1.27 * ((h - tft - 1.27 / 2) -
ybar)^2) + ... % glue tabs
    2 * ((mat_t * hw^3) / 12 + mat_t * hw * ((hw / 2 + bft) - ybar)^2) ...
% webs
    );

% Q at glue
Qglue = tfw * tft * (h - tft / 2 - ybar);   % area of top flange times
distance from top flange centroid to centroid

% Q at centroid
if ybar > h - 1.27 - tft
```

```
    Qcent = 2 ∗ (glue_t+mat_t) ∗ (h − tft − ybar)^2 / 2 + tfw ∗ tft ∗ (h −
tft / 2 − ybar); % centroid is within glue tabs
else
    Qcent = 2 ∗ (glue_t)∗1.27 ∗ ((h − tft − 1.27/2) − ybar) + tfw ∗ tft ∗
(h − tft / 2 − ybar) + 2 ∗ mat_t∗(hw − ybar)^2 / 2;   % centroid is below
glue tabs
end

% compressive and tensile stresses
S_top = BMD ∗ (h − ybar) /I;         % max compressive stress / top
S_bot = BMD ∗ ybar / I;               % max tensile stress / bottom

% shear stress at centroid and glue
T_cent = SFD ∗Qcent / (I ∗ mat_t);  % centroid
T_glue = SFD ∗Qglue / (I ∗ 2 ∗ (glue_t + 1.27)); % glue

%% Buckling Capacities (Corrected and Consistent Units)
E = 4000;             % young modulus MPa
mu = 0.2;
S_tens = 30;          % tensile strength MPa
S_comp = 6;           % compressive strength MPa
T_max = 4;            % max shear stress Mpa
T_gmax = 2;           % max shear stress glue Mpa

S_buck1 = (4 ∗ pi^2 ∗ E) / (12 ∗ (1 − mu^2)) ∗ (tft/bfw)^2;
S_buck2 = (0.4254 ∗ pi^2 ∗ E) / (12 ∗ (1 − mu^2)) ∗ (tft/((tfw − bfw) /
2))^2;
S_buck3 = (6 ∗ pi^2 ∗ E) / (12 ∗ (1 − mu^2)) ∗ ( mat_t / (hw + 1.27 + bft −
ybar))^2;
Shear_Buck = (5 ∗ pi^2 ∗ E) / (12 ∗ (1 − mu^2)) ∗ ((mat_t / (hw + 1.27 +
bft − ybar))^2 + (mat_t / a)^2);  % a = diaphram spacing in mm

FOS_tens = S_tens / abs(max(S_bot));
FOS_comp = S_comp / abs(max(S_top));
FOS_shear = T_max / abs(max(T_cent));
FOS_glue = T_gmax / abs(max(T_glue));
FOS_buck1 = S_buck1 / abs(max(S_top));
FOS_buck2 = S_buck2 / abs(max(S_top));
FOS_buck3 = S_buck3 / abs(max(S_top));
FOS_buckV = Shear_Buck / abs(max(T_cent));


minFOS = min([min(FOS_tens), min(FOS_comp), min(FOS_shear), min(FOS_glue),
min(FOS_buck1), min(FOS_buck2), min(FOS_buck3), min(FOS_buckV)]);
%smallest FOS

MaxF = P ∗ minFOS;  % smallest FOS times P gets us failure load
MaxF_Ignore = P ∗ FOS_shear;   % since FOS_comp is usualy our lowest FOS
and FOS_shear is the second lowest, this is the failure load assuming we
ignore FOS_comp
```

```
% the maximum moments and shear forces based on each FOS

Mf_tens = FOS_tens*max(BMD);
Mf_comp = FOS_comp*max(BMD);
Vf_shear = FOS_shear*max(SFD);
Vf_glue = FOS_glue*max(SFD);
Mf_buck1 = FOS_buck1*max(BMD);
Mf_buck2 = FOS_buck2*max(BMD);
Mf_buck3 = FOS_buck3*max(BMD);
Vf_buckV = FOS_buckV*max(SFD);

end
```