* Note FOS for all case 1, case 2, and case 4 is done by hand

The intermediate steps for design 0 load case 1 is too much for Google Docs(approximately 2 millions of them) and will cause errors in editing and downloading. If such data is needed, you can try generating them by the code or ask me to do so.

Output for all FOS for design 0 in load case 1:

The smallest FOS for flexural tension stress: 4.3741921899386735
The smallest FOS for flexural compression stress: 1.0596893302153323
The smallest FOS for shear stress: 2.5963630060579663
The smallest FOS for glue shear stress: 9.137781404872918
The smallest FOS for case 1 buckling stress: 0.610349
The smallest FOS for case 2 buckling stress: 4.15037
The smallest FOS for case 3 buckling: 4.825743694727476
The smallest FOS for case 4 buckling: 3.3023432343

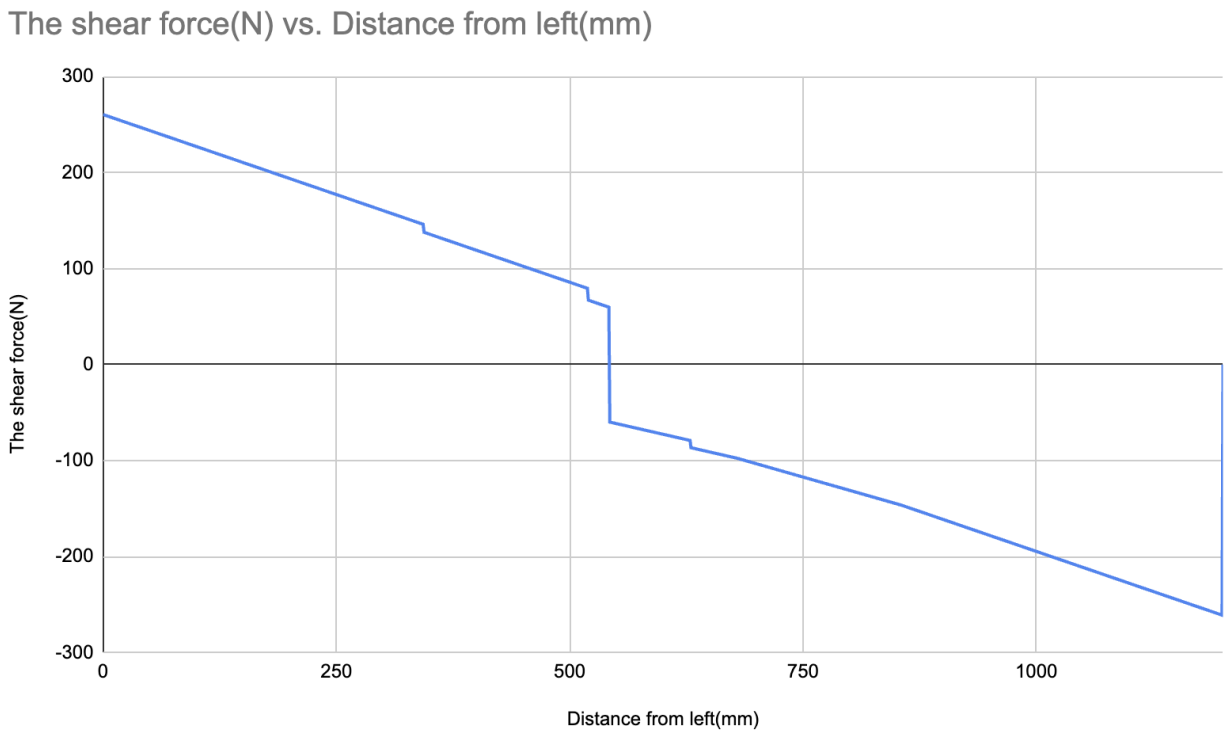

Figure 1: shear force envelope for design 0 load case 1

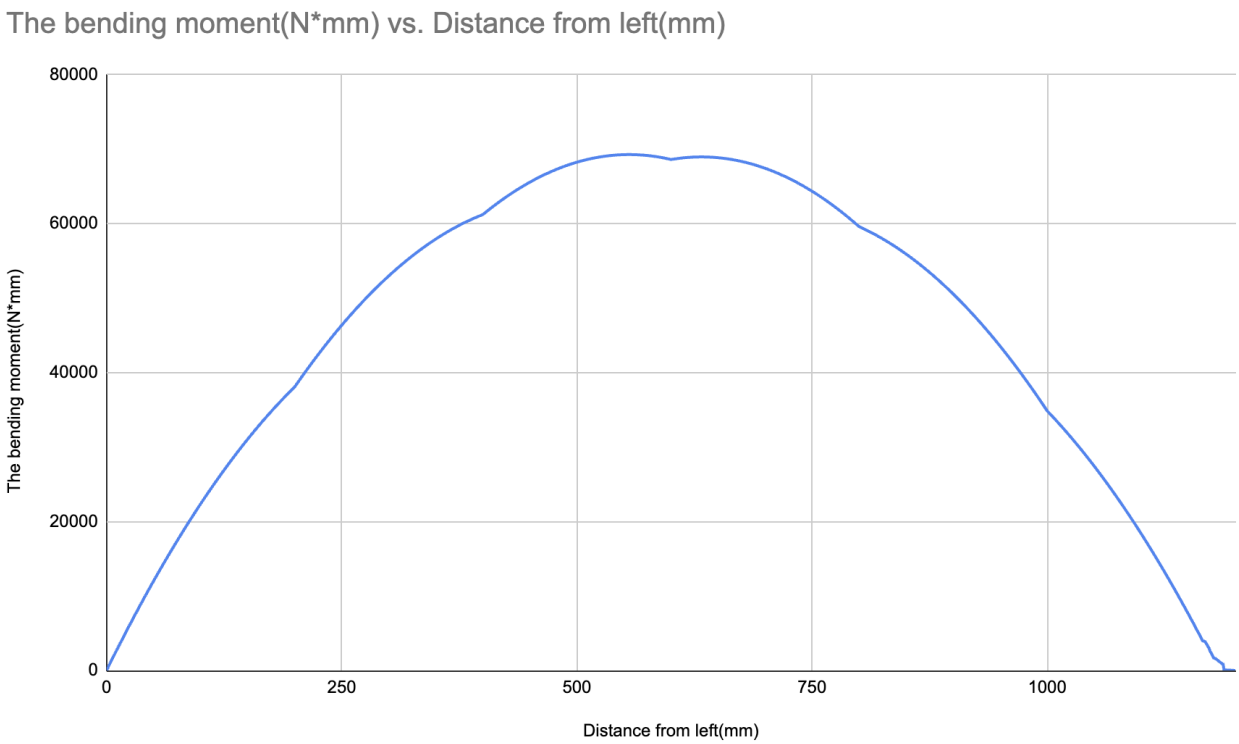

Figure 2: bending moment envelope for design 0 load case 1

Output for all FOS for final design in base case of loading case 2:

The smallest FOS for tension: 4.7262005160255995
The smallest FOS for compression: 2.305146455622289
The smallest FOS for shear: 3.443381340802213
The smallest FOS for glue shear: 17.23820531009729
The smallest FOS for case 1 buckling: 4.194664
The smallest FOS for case 2 buckling: 144.4013104
The smallest FOS for case 3 buckling: 7.41524428289616
The smallest FOS for case 4 buckling: 2.2719

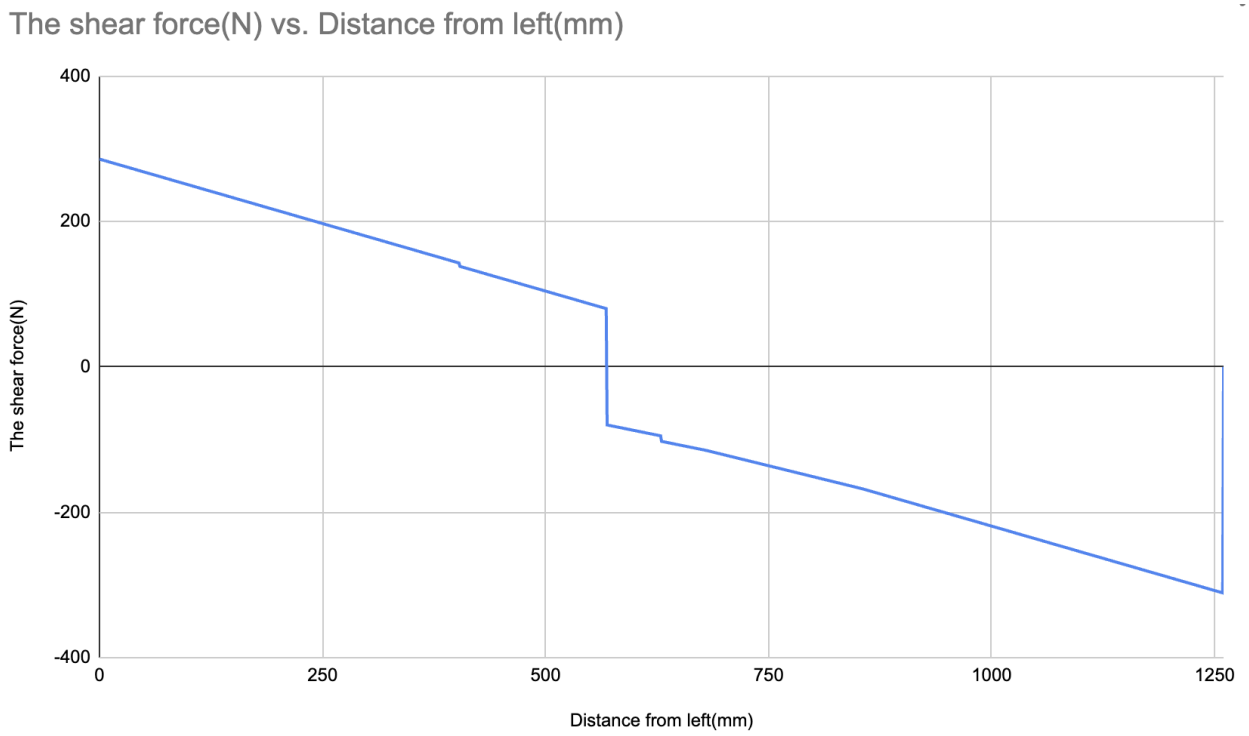

Figure 3: shear force envelope for final design with base case for load case 2

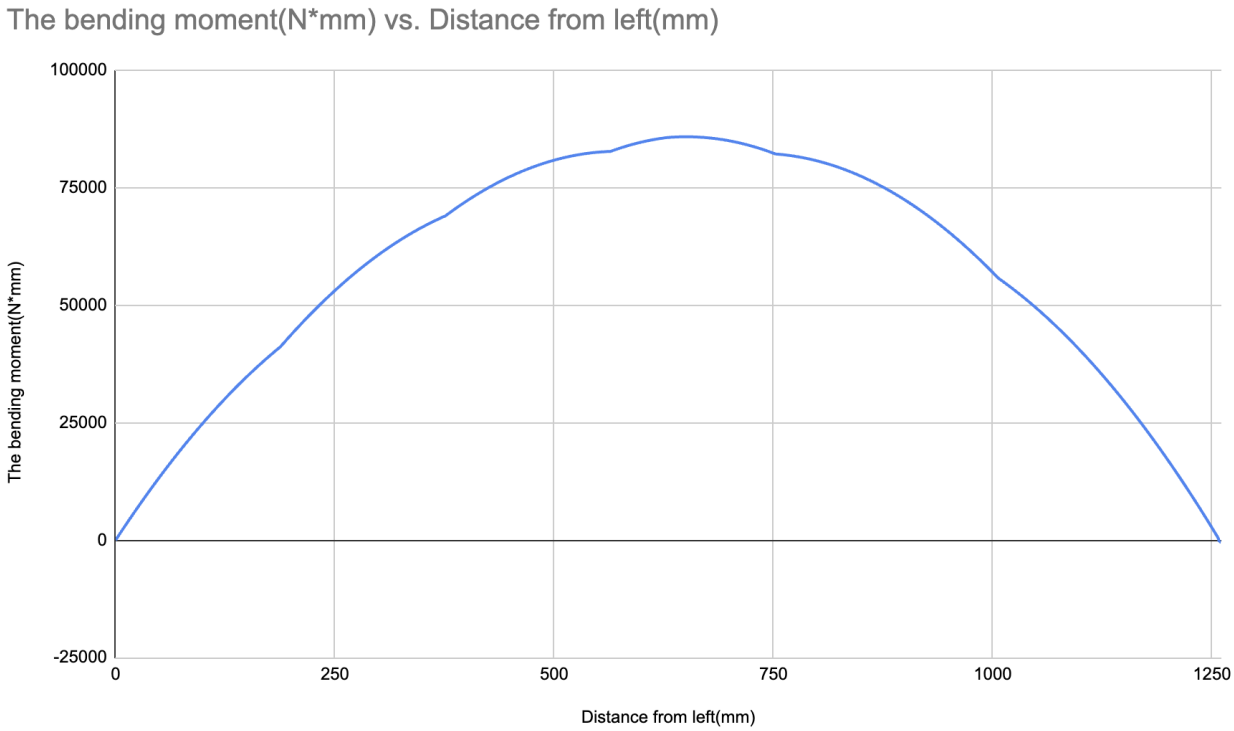

Figure 4: bending moment envelope for final design with base case for load case 2

Below is java code:

```
import java.io.IOException;
import java.lang.Math;
import java.io.FileWriter;
import java.util.Arrays;


public class civ102_calculation{
   // Run the train through the bridge first, so do 1200 loops
   // Try to find which section of the bridge will change the sign of the
moment, this will be where we may seprate the bridge section
   // so the geometric can withstand most moment
   // Bascially, find the maximum shear force and the moment of the 1200
points accorss the bridge, and plot those points on a graph, result in
   // shear force and moment envolope
   // create a list, fill with 6 points, those 6 points are the point of the
train wheel is at
   // every time, add 1 to each point. If the point is positive, add it to the
using list.

   // Below is all the arrays needed
   private static double[][][] sfd = new double[1261][1261][2];
   private static double[][][] bmd = new double[1261][1261][2];
   private static double x = 0;
   private static double[][] trainLoad = {{x,90},{x- 176, 90},
{x-340,66.66},{x-516, 66.66}, {x-680,66.66}, {x - 856,66.66}};
   private static double aY = 0;
   private static double bY = 0;
   private static double[] sfdMax = new double[1261];
   private static double[] bmdMax = new double[1261];

   private static double b1 = 100;
   //
   private static double a = 1.27;
   private static double h = 140;
   private static double[] tension = new double[1261];
   private static double[] compression = new double[1261];
   private static double[] maxShearStress = new double[1261];
   private static double[] maxGlueShearStress = new double[1261];
   private static double[] tensionFOS = new double[1261];
   private static double[] compressionFOS = new double[1261];
   private static double[] maxShearStressFOS = new double[1261];
   private static double[] maxGlueShearStressFOS = new double[1261];
   private static double[] case3Buck = new double[5];
   private static double minTFOS = 0;
   private static double minCFOS = 0;
   private static double minSFOS = 0;
```

```
   private static double minGFOS = 0;



   public static void main(String[] args) {
       //The following will go through all the load situation on the bridge,
with all 1260 points
       for(int i = 0; i < 1261; i++){

           // Below will change the location of the train's tire
           trainLoad[0][0] = x ;
           trainLoad[1][0] = x - 176;
           trainLoad[2][0] = x - 340;
           trainLoad[3][0] = x - 516;
           trainLoad[4][0] = x - 680;
           trainLoad[5][0] = x - 856;

           //Below will find the global equllbrium
           double totalM= 4635.225;
           double totalF = 7.3575;
           for(int j = 0; j < 6; j++){
               // This will determine if the train's tire is on the bridge
               if (trainLoad[j][0] >= 0) {
                   totalM += trainLoad[j][1] * trainLoad[j][0];
                   totalF += trainLoad[j][1];
               }
           }

           // Find the two support forces
           bY = totalM/1260;
           aY = totalF - bY;




           // the following will create 1261 shear force diagram
           // The following will also create 1261 bending moment diagram
           for(int k = 0; k < 1261; k++){
               // This will add in the shear force at a specific location on
the diagram
               // i means the location of the train tires
               // k means the location of the specific point under the loading
condition specificed by i
               // The 0 will add the location of the shear force and bending
moment
               sfd[i][k][0] = k;
               bmd[i][k][0] = k;
```

```
                //Codes below is use to find the shear force and bening moment
diagram

                if(k == 0){
                    sfd[i][k][1] += aY;
                }

                if(k > 0){
                    sfd[i][k][1] = sfd[i][k-1][1];
                }


                if(k == 630){
                    sfd[i][k][1] -= 7.3575;
                }



                if(k == 1260){
                    sfd[i][k][1] += bY;

                }

                // If the train has load at this location, subtract it from the
shear force
                for(int load = 0; load < 6; load++){
                    if(trainLoad[load][0] == k){
                        sfd[i][k][1] -= trainLoad[load][1];
                        break;
                    }
                }

                //Below will calculate the bending momement
                if(k == 0){
                    bmd[i][k][1] = 0;
                }else{
                    bmd[i][k][1] = sfd[i][k][1]  + bmd[i][k-1][1];
                }

            }
            x++;
        }

        // The following meant to find the biggest shear force at every
location, it also finds the biggest bending moment everywhere
        // This will create shear force and bending moment envelope
        for(int i = 0; i < 1261; i++){
            // It will begin at location i with the first sencario, which is
when the train's tire is at x = 0
```

```
            double maxSFD[] = {0, Math.abs(sfd[0][i][1])};
            double maxBMD[] = {0, Math.abs(bmd[0][i][1])};
            // The j will change the location of the train tire
            for(int j = 0; j < 1261; j++){
                double tmpSFD = Math.abs(sfd[j][i][1]);
                double tmpBMD = Math.abs(bmd[j][i][1]);
                if(tmpSFD > maxSFD[1]){
                    maxSFD[1] = tmpSFD;
                    maxSFD[0] = j;
                }
                if(tmpBMD > maxBMD[1]){
                    maxBMD[1] = tmpBMD;
                    maxBMD[0] = j;
                }
            }
            sfdMax[i] = sfd[(int)maxSFD[0]][i][1];
            bmdMax[i] = bmd[(int)maxBMD[0]][i][1];

        }

        // At the right end, SFD must be zero
        sfdMax[1260] = 0;
        // At the left and right end, BMD must be zero
        bmdMax[1260] = 0;

        output(sfdMax, "sfdMax");
        output(bmdMax, "bmdMax");


        // Below will calculate the different FOSs in different height



        // The following code uses the function below to find the y_bar, second
moment of area,
        // and the various stresses
        double y_bar = y_bar();
        double I = I(y_bar);
        System.out.println("y_bar is: " + y_bar);
        System.out.println("I is: " + I);
        tension = tensionStress(y_bar, I);
        compression = compressionStress(y_bar, I);
        maxShearStress = maxShearStress(sfdMax, y_bar, I);
        maxGlueShearStress = maxGlueSStress(sfdMax, y_bar, I);
        case3Buck[0] = case3Buckle(Math.abs(h + 2.54 - y_bar));

        //The following code is used to calculate teh FOS
        // The reason some FOS will be icnreased to 100000000000 is because
sometimes the moment and shear force is 0,
```

```
        // lead to infinity problem in java
        for (int i = 0; i < 1261; i++) {
            if(tension[i] == 0.0){
                tensionFOS[i] = 1000000000000000.0;
            }else{
                tensionFOS[i] = 30 / tension[i];
            }
            if(compression[i] == 0.0){
                compressionFOS[i] = 1000000000000000.0;
            }else{
                compressionFOS[i] = 6 / compression[i];
            }
            if(maxShearStress[i] == 0.0){
                maxShearStressFOS[i] = 1000000000000000.0;
            }else{
                maxShearStressFOS[i] = 4 / maxShearStress[i];
            }
            if(maxGlueShearStress[i] == 0){
                maxGlueShearStressFOS[i] = 1000000000000000.0;
            }else{
                maxGlueShearStressFOS[i] = 2 / maxGlueShearStress[i];
            }

        }


        // The folllowing code is used to find the smallest FOS
        // it uses a for loop to go through all the FOS arrays and find the
smallest value in them
        minTFOS = tensionFOS[0];
        minCFOS = compressionFOS[0];
        minSFOS = maxShearStressFOS[0];
        minGFOS = maxGlueShearStressFOS[0];
        for (int i = 0; i < 1261; i++) {
            double tmpTFOS = tensionFOS[i];
            double tmpCFOS = compressionFOS[i];
            double tmpSFOS = maxShearStressFOS[i];
            double tmpGFOS = maxGlueShearStressFOS[i];

            if (tmpTFOS < minTFOS) {
                minTFOS = tmpTFOS;
            }
            if (tmpCFOS < minCFOS) {
                minCFOS = tmpCFOS;
            }
            if (tmpSFOS < minSFOS) {
                minSFOS = tmpSFOS;
            }
            if (tmpGFOS < minGFOS) {
```

```
                minGFOS = tmpGFOS;
            }
        }

        // Below is used to give instant feeback on the FOSs
        System.out.println("Height: " + h + " Width: " + b1);
        System.out.println("The smallest FOS for tension: " +  minTFOS);
        System.out.println("The smallest FOS for compression: " + minCFOS);
        System.out.println("The smallest FOS for shear: " + minSFOS);
        System.out.println("The smallest FOS for glue shear: " + minGFOS);
        System.out.println("The smallest FOS for case 3 buckling: " +
case3Buck[0] / Arrays.stream(compression).max().getAsDouble());
        System.out.println("The case 3 buckling stress is: " + case3Buck[0]);



        // Below is used to output the graphs of verious FOS and shear
force/beanding moment envelope
        output(tension, "tension");
        output(compression, "compression");
        output(maxShearStress, "maxShearStress");
        output(maxGlueShearStress, "maxGlueShearStress");
        output(tensionFOS, "tensionFOS");
        output(compressionFOS, "compressionFOS");
        output(maxShearStressFOS, "maxShearStressFOS");
        output(maxGlueShearStressFOS, "maxGlueShearStressFOS");
        output(tensionFOS, "tensionFOS");
    }

    // This functtion is used to find the centeriod axis
    public static double y_bar(){
        double area_top = 2*b1*a;
        double area_vertical = 2*h*1.27;
        double area_glue_joint = 17.46*1.27;

        double centre_glue_joint = h - 1.27/2;
        double centre_top = (a)+h;
        double centre_vertical = h/2;

        return ((area_top)*centre_top + area_glue_joint*centre_glue_joint +
area_vertical*centre_vertical)/(area_top + area_vertical + area_glue_joint);

    }

    // This function is used to find the second moment of area value
    public static double I(double y_bar){
        double I1 = b1*(a*a*a*2*2*2)/12;
        double I2 = 2*1.27*(h*h*h)/12;
        double I3 = 17.46*1.27*1.27*1.27/12;
```

```
        double area_glue_joint = 17.46*1.27;
        double area_top = 2*b1*a;
        double area_vertical = 2*h*1.27;

        double y_top = Math.abs((a)+h - y_bar);
        double y_vertical = Math.abs(h/2 - y_bar);
        double y_glue_joint = Math.abs(h - 1.27/2 - y_bar);

        return I1 + I2 + I3 + area_top*y_top*y_top +
area_vertical*y_vertical*y_vertical +
area_glue_joint*y_glue_joint*y_glue_joint;

    }

    // This function is used to find the tension stress of every point
    public static double[] tensionStress(double y_bar, double I){
        double[] tension = new double[1261];
        for(int i = 0; i < 1258; i++){
            tension[i] = bmdMax[i]*(Math.abs(y_bar))/I;
        }
        return tension;
    }

    // This function is used to find the compression stress of every point

    public static double[] compressionStress(double y_bar, double I){
        double[] compress = new double[1261];
        for(int i = 0; i < 1258; i++){
            compress[i] = bmdMax[i]*(Math.abs(h + 2*1.27-y_bar))/I;
        }
        return compress;
    }

    // This function is used to find the maximum shear stress of every point

    public static double[] maxShearStress(double[] sfdMax, double y_bar, double
I){
        double b = 1.27*2;
        double Q = 2*1.27*(y_bar)*(y_bar/2);
        double[] maxShearStress = new double[1261];
        for(int i = 0; i < 1260; i ++){
            maxShearStress[i] = Math.abs(sfdMax[i]*Q/(I*b));
        }
        return maxShearStress;
    }

    // This function is used to find the maximum shear stress of glue joint of
every point
```

```
   public static double[] maxGlueSStress(double[] sfdMax, double y_bar, double
I){
       double b = 10*2;
       double Q = 2*1.27*b1*(Math.abs(1.27+h-y_bar));
       double[] maxGlueShearStress = new double[1261];
       for(int i = 0; i < 1260; i ++){
           maxGlueShearStress[i] = Math.abs(sfdMax[i]*Q/(I*b));
       }
       return maxGlueShearStress;
   }

   // This function is used to find the critial stress for load case 3
   public static double case3Buckle(double b){
       return ((6*Math.PI*Math.PI*4000)/(12*(1-0.2*0.2)))*(1.27/b)*(1.27/b);
   }

   // This function will create output that combines the information in the
array
   public static void output(double[] tmp, String name){
       FileWriter csvWriter_sfd = null;
       try {
           csvWriter_sfd = new FileWriter(name +".csv");

           // Write each element of the array into the CSV file
           for (int i = 0; i < tmp.length; i++) {
               csvWriter_sfd.append(Double.toString(tmp[i]));
               csvWriter_sfd.append("\n");
           }

           csvWriter_sfd.flush();
           csvWriter_sfd.close();
       } catch (IOException e) {
           e.printStackTrace();
       }
   }




}
```