GUI development for laminate analysis

Sumit Sunil Kumar

July 1, 2019

Abstract

This is the work during 8 weeks internship with IIT Bombay.

1 Introduction

So i have prepared a set of codes along with a graphical user interface of the following results. I have programmed it in GNU Octave but mostly through octave-cli (command line interface) used in the terminal window in linux. My report deals with computing the various details of a layered material like the calculation of the 3 stiffness matrices A, B and D, the stress vector along with the stress vectors of the various layers and the reduced stiffness matrix Q along with a global reduced stiffness matrix $\overline{Q}$ which depends on the fiber orientation. I am also calculating stress concentration factor (Kt) of a finite plate with given dimensions of diameter of hole and thickness and also for infinite plate. Also I wrote a program to calculate if the material has failed or not failed by using tsai-hill failure theory.

Please note that you have to have GNU Octave installed. So you can just type "octave-cli" for running octave source codes and "octave - -no-gui" for running the GUI codes in a terminal (example, xterm).

2 Calculations and Formulae used

The reduced stiffness of an orthotropic material goes as

$$Q = \begin{bmatrix} Q_{11} & Q_{12} & 0 \\ Q_{21} & Q_{22} & 0 \\ 0 & 0 & Q_{66} \end{bmatrix} \tag{1}$$

where transverse moduli, poisson ratios and shear modulus are taken into consideration

$$Q_{11} = \frac{E_1}{1 - \gamma_{12} \times \gamma_{21}}$$

$$Q_{22} = \frac{E_2}{1 - \gamma_{12} \times \gamma_{21}}$$

$$Q_{12} = \frac{E_1 \times \gamma_{21}}{1 - \gamma_{12} \times \gamma_{21}}$$

$$Q_{21} = \frac{E_2 \times \gamma_{12}}{1 - \gamma_{12} \times \gamma_{21}}$$

$$Q_{66} = \frac{1}{G_{12}}$$

also note that

$$\gamma_{12} \times E_2 = \gamma_{21} \times E_1$$

But it is not necessary that the fiber is always oriented parallel to the material, hence comes the factor of fiber orientation which is basically the angle that the fiber direction makes i.e. θ

therefore for writing the stress and strain relation in global coordinates, we require the transformation matrices T_1 and T_2 respectively

and define the reduced stiffness matrix in global coordinates as

$$\overline{Q} = (T_1)^{-1} * Q * T_2$$

$$T_1 = \begin{bmatrix} m^2 & n^2 & 2mn \\ n^2 & m^2 & -2mn \\ -mn & mn & m^2 - n^2 \end{bmatrix} \tag{2}$$

$$T_2 = \begin{bmatrix} m^2 & n^2 & mn \\ n^2 & m^2 & -mn \\ -2mn & 2mn & m^2 - n^2 \end{bmatrix} \tag{3}$$

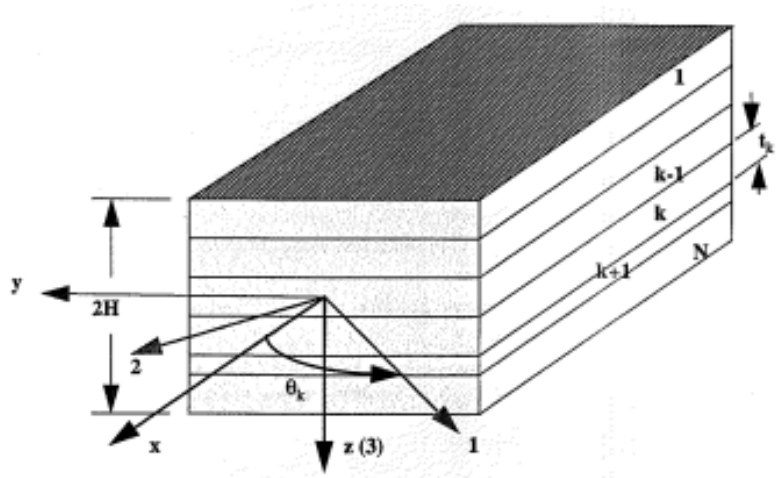


FIGURE 5.1 Composite Laminate

Figure 1: Composite Laminate (from [**?**])

where

$$m = cos(\theta)$$

$$n = sin(\theta)$$

after obtaining the following results, we can move on to cases where the lamina has more than 1 layer. So each layer will have a different fiber orientation and hence will have a different value of $\overline{Q}$

So for a material with k layers, we define the values of reduced stiffness matrix of the k^{th} layer as $\overline{Q}^k$ and the corresponding to z-location. z^k is the distance of k^{th} layer from the origin.

We also have

$$\sigma_x = \overline{Q}^k \times \epsilon_x$$

and

$$\epsilon_x = \epsilon^o{}_x + z \times \kappa_x$$

with this given data, we also can balance the forces per unit length and moments per unit length to get a simplified form as

$$N = [A]\epsilon^o + [B]\kappa$$

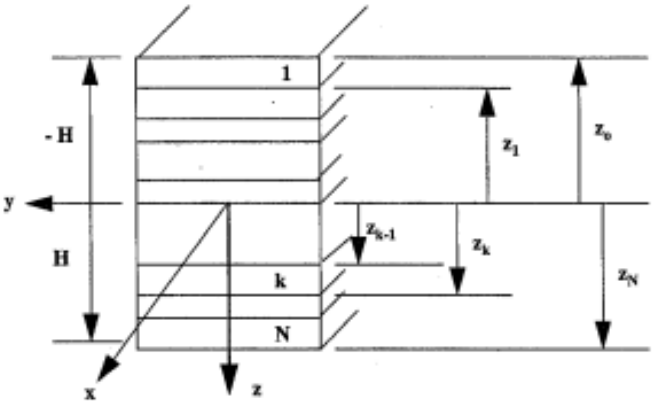


Figure 2: Laminate Coordinates

$$M = [B]\epsilon^o + [D]\kappa$$

here [A] and [B] and [D] are defined as a sum of all the layers of the material.

A represents in-plane stiffness and B represents stretching and bending. D is a bending stiffness matrix The formulae for A, B, D are

$$A = \sum_{k=1}^{N} \overline{Q}^k (z^k - z^{k-1})$$

$$B = \frac{1}{2} \sum_{k=1}^{N} \overline{Q}^k ((z^k)^2 - (z^{k-1})^2)$$

$$D = \frac{1}{3} \sum_{k=1}^{N} \overline{Q}^k ((z^k)^3 - (z^{k-1})^3)$$

where

$$[NM] = \begin{bmatrix} A & B \\ B & D \end{bmatrix} [\epsilon^o \kappa] \tag{4}$$

where

$$N_x = \int_{-H}^{H} \sigma_x dz$$

$$N_y = \int_{-H}^{H} \sigma_y dz$$

$$N_{xy} = \int_{-H}^{H} \tau_{xy} dz$$

$$M_x = \int_{-H}^{H} \sigma_x z dz$$

$$M_y = \int_{-H}^{H} \sigma_y z dz$$

$$M_{xy} = \int_{-H}^{H} \tau_{xy} z dz$$

$$N = \int_{-H}^{H} \sigma dz$$

$$N = \int_{-H}^{H} \overline{Q}^k \epsilon dz = \int_{-H}^{H} \overline{Q}^k \epsilon^o dz + \int_{-H}^{H} \overline{Q}^k \kappa z dz$$

therefore

$$N = \sum_{k=1}^{N} (\int_{z^{k-1}}^{z^k} \overline{Q}^k dz) \times \epsilon^o + \sum_{k=1}^{N} (\int_{z^{k-1}}^{z^k} \overline{Q}^k z dz) \times \kappa$$

so it as

$$N = [A]\epsilon^o + [B]\kappa$$

and

$$M = [B]\epsilon^o + [D]\kappa$$

as mentioned above we get this 6×6 matrix that relates the forces and moments to the strain relations. So we invert this matrix to get the strain vector in terms of forces and moments per unit length.

The strain matrix output is like

$$\begin{bmatrix} {\epsilon^o}_x \\ {\epsilon^o}_y \\ {\gamma^o}_{xy} \\ {\kappa^o}_x \\ {\kappa^o}_y \\ {\kappa^o}_{xy} \end{bmatrix} \tag{5}$$

Now we calculate the stress matrix through this essentially it will be a 3×1 matrix for each layer due to the formula we have that relates stress to $\overline{Q}$ of the k^{th} layer along with the distance of it from the mid-plane axis.

In the graphical user interface the values of the reduced stiffness matrix along with A, B, D and the inverse of the 6×6 matrix for getting strain as well as the final strain vector.

The stress matrix is in the source code written in octave-cli.

3 The Octave codes and Output

3.1 stress.m

```
function[Q, t1, t2, Q_g, A, B, D, E, strain, glob_strn, stress]
= stress(N, e1, e2, g12, mu12, mu21)

if(nargin != 6)
printf("Usage inputs are stress(N, e1, e2, g12, mu12, mu21)\n");
return;
endif


N = input("Select number of layers: ");
theta = zeros(1, N);
z = zeros(1, N);

for i = 1:N
printf('Enter the angle of the %d layer', i);
theta(i) = input(" (in theta): ");
printf('Enter the distance from the base for %d layer', i);
z(i) = input(" (in units): ");
endfor


 printf('enter the value of Nx');
 Nx = input(" (in units) ");
```

```
 printf('enter the value of Ny');
 Ny = input(" (in units) ");

 printf('enter the value of Nxy');
 Nxy = input(" (in units) ");

 printf('enter the value of Mx');
 Mx = input(" (in units) ");

 printf('enter the value of My');
 My = input(" (in units) ");

 printf('enter the value of Mxy');
 Mxy = input(" (in units) ");

 printf('enter the value of x');

 x = input(" (the distance from the mid-plane) ");

 norm_moment = [Nx; Ny; Nxy; Mx; My; Mxy];

Q_g = zeros(3, 3, N);
for i = 1:N
 [Q, t1, t2, Q_g(:,:,i)] = global_stress(theta(i), e1, e2, g12, mu12, mu21);
endfor

A = zeros(3,3);
B = zeros(3,3);
D = zeros(3,3);
sumation1 = zeros(3,3);
sumation2 = zeros(3,3);
sumation3 = zeros(3,3);
Z = zeros(6,6);
E = zeros(6,6);


for i = 2:N
 num1 = Q_g(:,:,i)*(z(i) - z(i-1));
```

```
 sumation1 = sumation1 + num1;
endfor

A = sumation1;

for i = 2:N
 num2 = Q_g(:,:,i)*((z(i))^2 - (z(i-1))^2)*(1/2);
 sumation2 = sumation2 + num2;
endfor

B = sumation2;

for i = 2:N
 num3 = Q_g(:,:,i)*((z(i))^3 - (z(i-1))^3)*(1/3);
 sumation3 = sumation3 + num3;
endfor

D = sumation3;

strain = zeros(6,1);

glob_strn = zeros(3,1);
stress = zeros(3,1,N);
Z = [A, B; B, D];
E = inv(Z);

for i = 1:6
 strain(i,1) = E(i,1)*Nx + E(i,2)*Ny + E(i,3)*Nxy +
               E(i,4)*Mx + E(i,5)*My + E(i,6)*Mxy;
endfor

for i = 1:3
 glob_strn(i,1) = strain(i,1) + x*strain(i+3,1);
endfor

for j = 1:N
  stress(:,1,j) = Q_g(:,:,j)*glob_strn();
end
```

3.2 global_stress.m

```
function [Q, t1, t2, Q_global] = global_stress(theta, e1, e2, g12, mu12, mu21)

if(nargin != 6)
printf("Usage global_stress(theta, e1, e2, g12, mu12, mu21)\n");
return
endif

% Now we have the desired values from user
% We define the output
Q = zeros(3,3);
t1 = zeros(3,3);
t2 = zeros(3,3);

%define a constant ..
k = 1 - mu12*mu21;

m = cos(theta);
n = sin(theta);

% Based on book ... page ...
Q(1,1) = e1/k;
Q(1,2) = (mu12*e2)/k;
Q(2,1) = (mu21*e1)/k;
Q(2,2) = e2/k;
Q(3,3) = g12;

t1(1,1) = m^2;
t1(1,2) = n^2;
t1(1,3) = 2*m*n;
t1(2,1) = t1(1,2);
t1(2,2) = t1(1,1);
t1(2,3) = -1*t1(1,3);
t1(3,1) = -m*n;
t1(3,2) = -1*t1(3,1);
t1(3,3) = t1(1,1) - t1(1,2);
```

```
t2 = t1;

t2(1,3) = t1(3,2);
t2(2,3) = -1*t2(3,2);
t2(3,1) = -1*t1(1,3);
t2(3,2) = -1*t2(3,1);

Q_global = inv(t1)*Q*t2;

return
```

3.3 GUI Screenshots

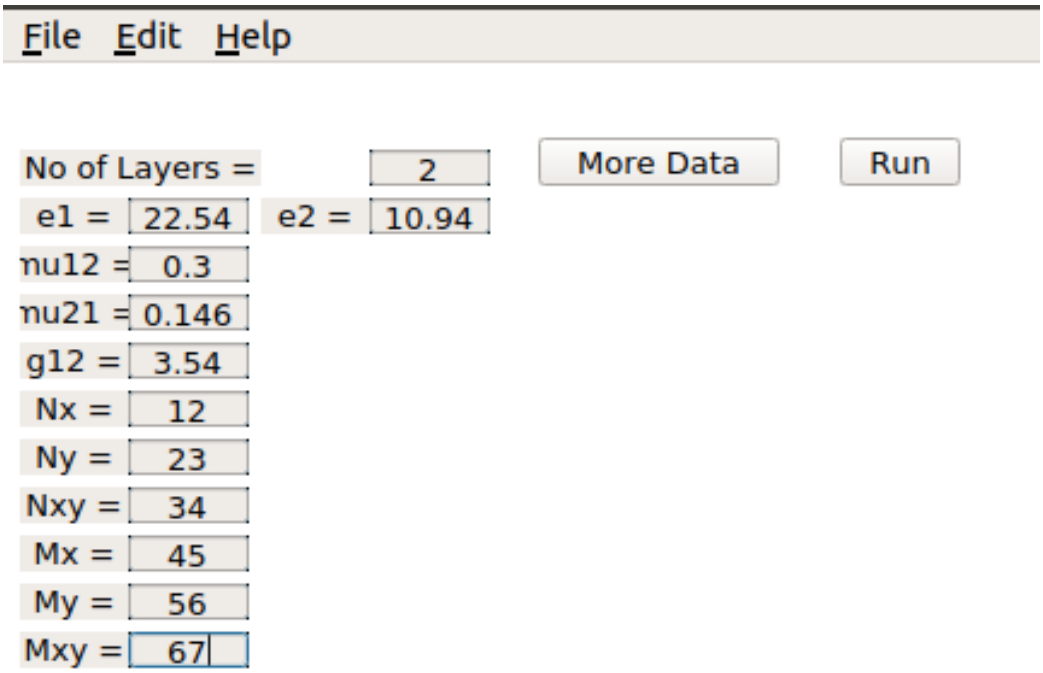


Figure 3: Input to the GUI

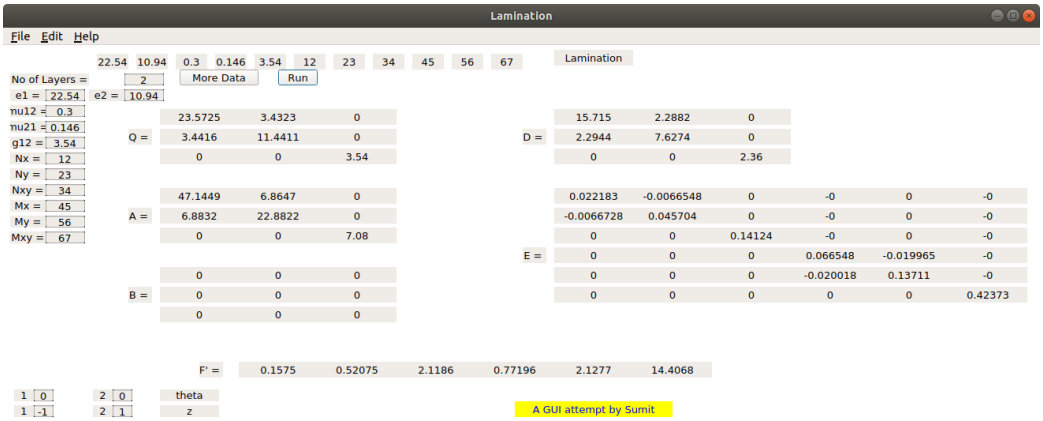


Figure 4: Output after run.

3.4 Stress Concentration

```
function[Q, t1, t2, Q_g, A, KTinfi, KT] =
finiteplate(N, e1, e2, g12, mu12, mu21, d, w, theta, z)


Q_g = zeros(3, 3, N);
for i = 1:N
 [Q, t1, t2, Q_g(:,:,i)] = global_stress(theta(i), e1, e2, g12, mu12, mu21);
endfor


b = d/w;
K = 3*(1-b)/(2 + (1-b)^3);
bm2 = (1/2)*(sqrt(1 - 8*(K - 1)) - 1)

A = zeros(3,3);
sumation = zeros(3,3);


for i = 2:N
 num(:,:,i) = Q_g(:,:,i)*(z(i) - z(i-1));
 sumation = sumation + num(:,:,i);
```

```
endfor

A = sumation

KTinfi = 1 + sqrt((2/A(2,2))*(((A(1,1)*A(2,2) -
         A(1,2)^2)/(2*A(3,3))) - A(1,2) + sqrt(A(1,1)*A(2,2))));

KT = KTinfi/(K + (1/2)*((bm2)^3)*(KTinfi - 3)*(1 - bm2));
```

3.5 GUI Screenshots

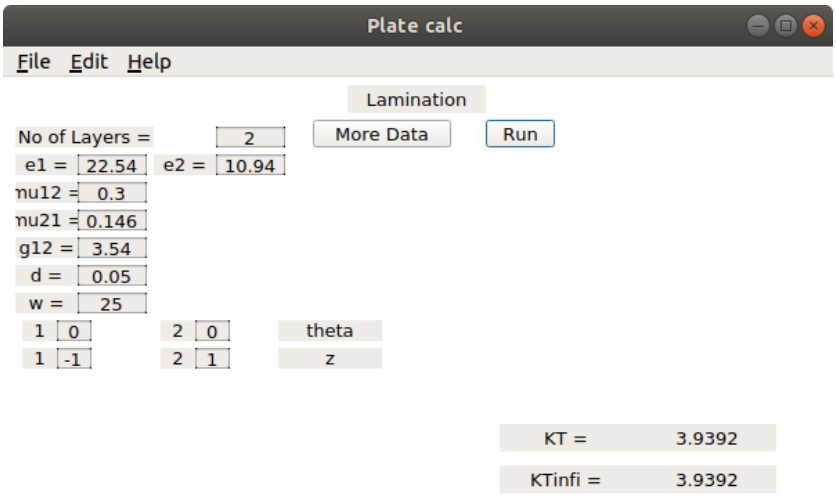


Figure 5: Output of stress concentration.

4 Failure criteria

We use tsai-hill theory for calculating whether a material has failed. It is generally used in anisotropic materials i.e. materials which show different properties when in different directions. Essentially the original quadratic equation is

$$
\begin{aligned}
(G+H)\sigma_1^2 + (F+H)\sigma_2^2 + (G+F)\sigma_3^2 \quad & - \\
2H\sigma_1\sigma_2 - 2G\sigma_1\sigma_3 - 2F\sigma_2\sigma_3 + 2L\tau_{23}^2 + 2M\tau_{13}^2 + 2N\tau_{12}^2 \quad & = \quad 1 \qquad (6)
\end{aligned}
$$

here F, G, H, L, M, N are material strength parameters. If this function is less than or equal to 1 then the material is safe whereas if it is greater than 1, it corresponds to failure. In case of plane stress, $\sigma_3 = \tau_{13} = \tau_{23} = 0$ Hence with theta involved, we get,

$$
\frac{\cos^4\theta}{X^2} + \left(\frac{1}{S^2} - \frac{1}{X^2}\right)(\sin^2\theta\cos^2\theta) + \frac{\sin^4\theta}{Y^2} = \frac{1}{\sigma_x^2}
$$

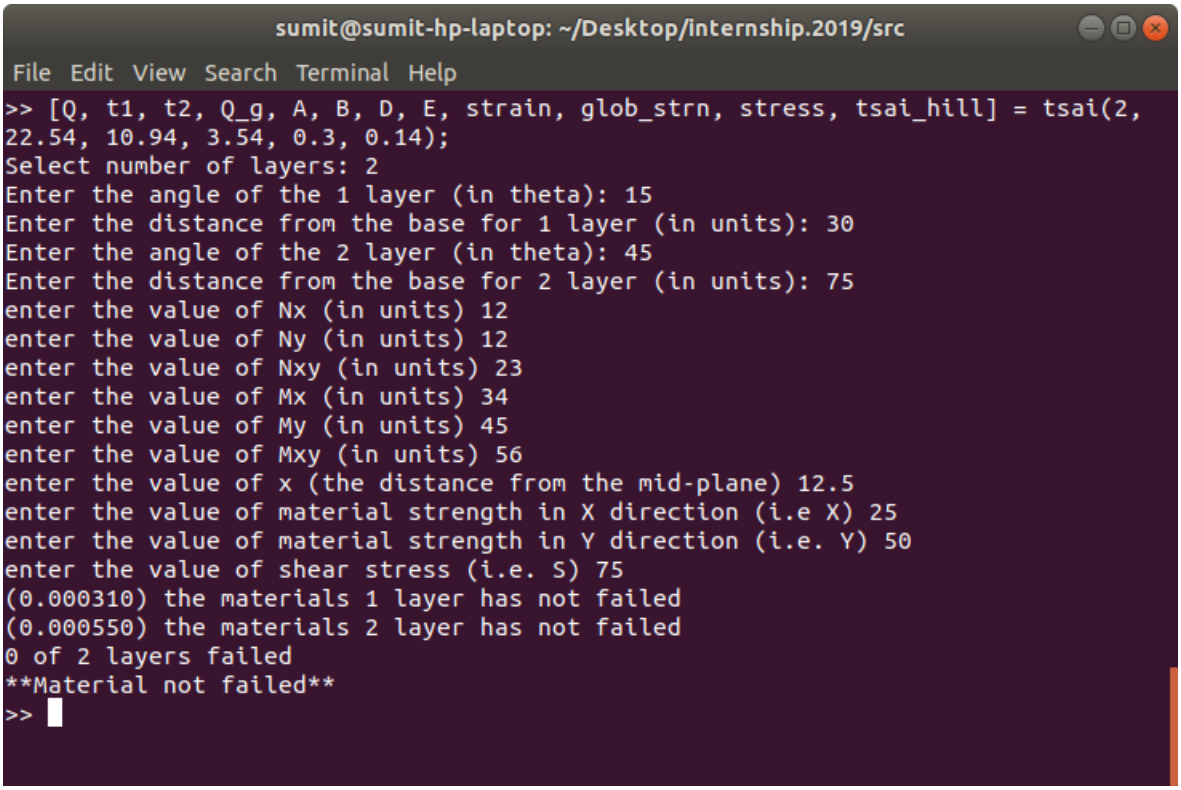
I have written a code that shows how many layers have failed in a layered material and whether it has failed or not.

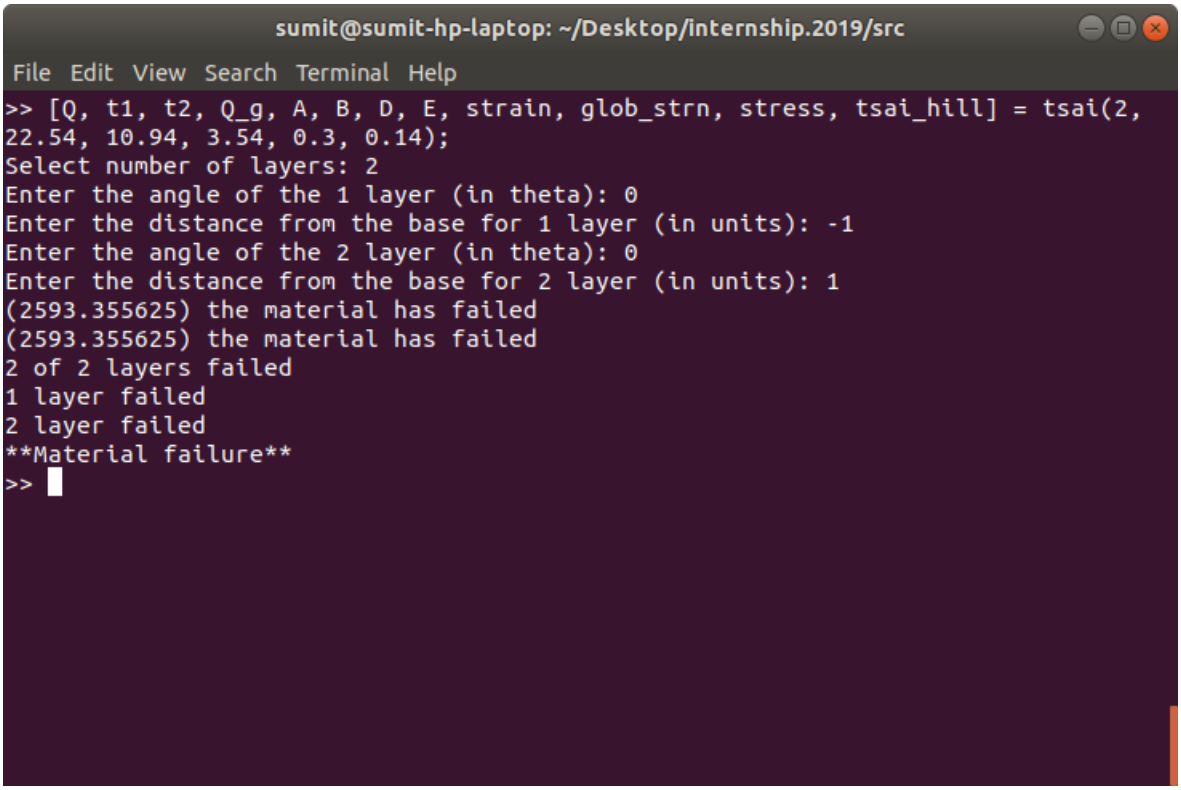
4.1 The code

```
%tsai-hill criteria
tsai_hill = zeros(1,1,N);
failure_flag = zeros(1,N);
global_failure = 0;

for i = 1:N
ct = cos(theta(i));
st = sin(theta(i));
tsai_hill(1,1,i) = (stress(1,1,i)^2)*(ct^4/X^2 +
(1/S^2 - 1/X^2)*(st*ct)^2 + st^4/Y^2);
endfor

for i = 1:N
if(tsai_hill(1,1,i) > 0 & tsai_hill(1,1,i) < 1)
printf("(%f) the materials %d layer has not failed\n", tsai_hill(1,1,i),i);
failure_flag(1,i) = 1;

else
printf("(%f) the material has failed\n", tsai_hill(1,1,i));
global_failure = global_failure+1;
endif
```

```
endfor

printf("%d of %d layers failed\n",global_failure,N);
if(global_failure > 0)
for i=1:N
 if(failure_flag(1,i) == 0)
   printf("%d layer failed\n",i);
 endif
endfor
printf("**Material failure**\n");

else
printf("**Material not failed**\n");
endif
```

4.2 The corresponding output


sumit@sumit-hp-laptop: ~/Desktop/internship.2019/src
File Edit View Search Terminal Help
>> [Q, t1, t2, Q_g, A, B, D, E, strain, glob_strn, stress, tsai_hill] = tsai(2,
22.54, 10.94, 3.54, 0.3, 0.14);
Select number of layers: 2
Enter the angle of the 1 layer (in theta): 15
Enter the distance from the base for 1 layer (in units): 30
Enter the angle of the 2 layer (in theta): 45
Enter the distance from the base for 2 layer (in units): 75
enter the value of Nx (in units) 12
enter the value of Ny (in units) 12
enter the value of Nxy (in units) 23
enter the value of Mx (in units) 34
enter the value of My (in units) 45
enter the value of Mxy (in units) 56
enter the value of x (the distance from the mid-plane) 12.5
enter the value of material strength in X direction (i.e X) 25
enter the value of material strength in Y direction (i.e. Y) 50
enter the value of shear stress (i.e. S) 75
(0.000310) the materials 1 layer has not failed
(0.000550) the materials 2 layer has not failed
0 of 2 layers failed
Material not failed
>>


sumit@sumit-hp-laptop: ~/Desktop/internship.2019/src
File Edit View Search Terminal Help
>> [Q, t1, t2, Q_g, A, B, D, E, strain, glob_strn, stress, tsai_hill] = tsai(2,
22.54, 10.94, 3.54, 0.3, 0.14);
Select number of layers: 2
Enter the angle of the 1 layer (in theta): 0
Enter the distance from the base for 1 layer (in units): -1
Enter the angle of the 2 layer (in theta): 0
Enter the distance from the base for 2 layer (in units): 1
(2593.355625) the material has failed
(2593.355625) the material has failed
2 of 2 layers failed
1 layer failed
2 layer failed
Material failure
>>

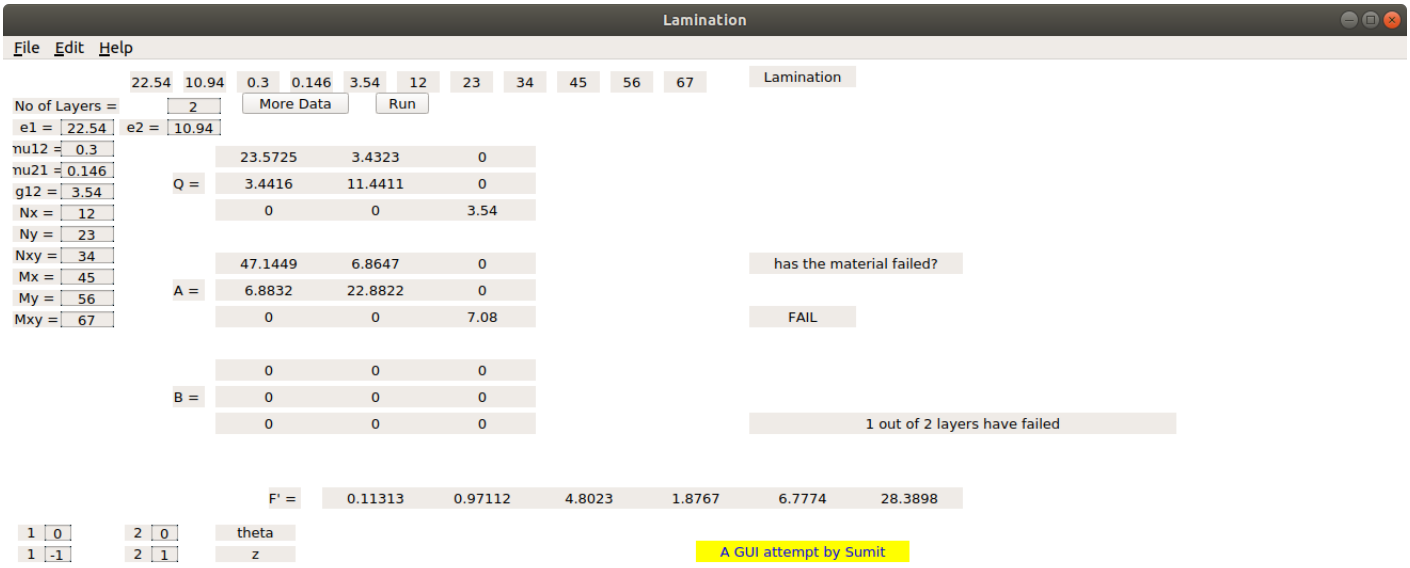


Figure 6: for the fail

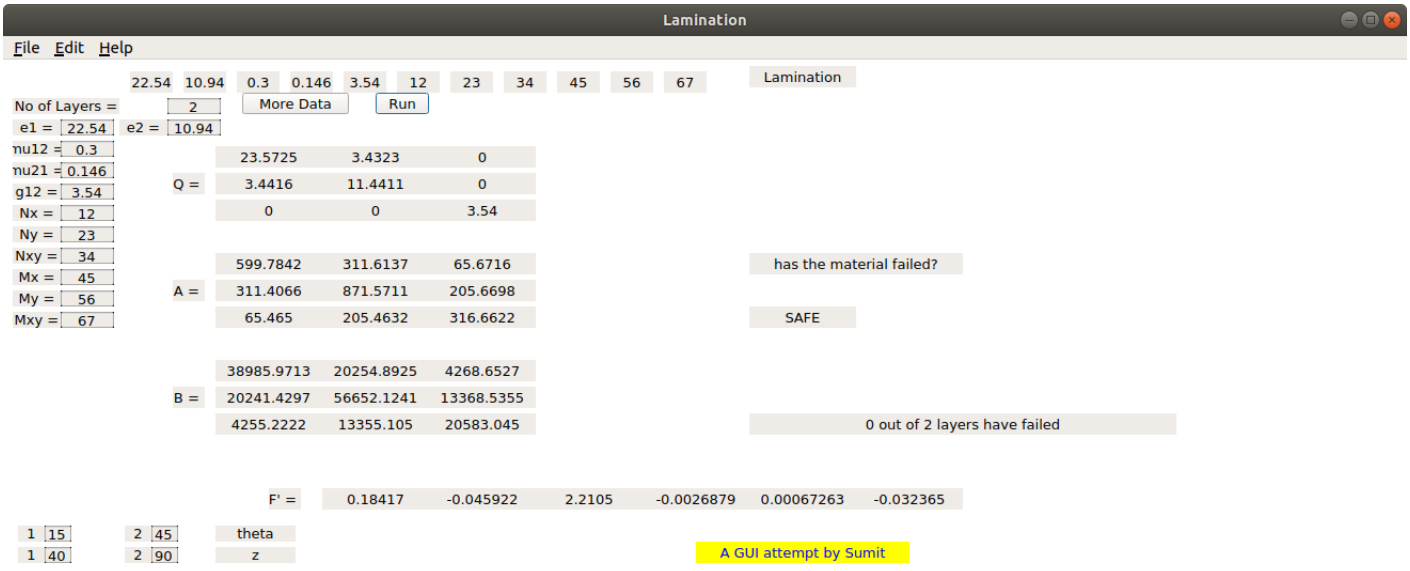


Figure 7: for a success