Clement
Brousse

3/22/2022
clbrous@colostate.edu

Assignment #3

Contents

The scripts used are available on Github at https://github.com/ClementCSU/MECH535.

Clement Brousse
3/22/2022
clbrous@colostate.edu

Functions used

These are the functions used.

Rotation of the strains

Syntax

```
[E2] = Strain12x(E1,Theta)
```

Descriptions

This function rotates the strain **E1** with an angle **theta** expressed in degrees. The transformation is made with the double cosine and sine functions as presented in table 3.5 (Lessard). The output strain **E2** have the same size as the input strains **E1**. **E1** is a 3×1 array.

$$\boldsymbol{E1} = [\epsilon_{xx} \quad \epsilon_{yy} \quad \epsilon_{ss}]$$

Code

```
function [E2] = Strain12x(E1,Theta)
Theta = deg2rad(Theta);
%% Rotation of the strain (E1) from an angle (Theta)
% Compute intermediary values
p = (E1(1,1) + E1(2,1))/2; % [ ]
q = (E1(1,1) - E1(2,1))/2; % [ ]
r = E1(3,1)/2; % [ ]
M = [p, q, r; ...
    p, -q, -r; ...
    0, 2*r, -2*q];  % Define the transformation matrix (Table 3.5)
E2 = M * [1; cos(2*Theta); sin(2*Theta)]; % [ ] Strain in the second coordinate system
end
```

Known errors

The input strains **E1** should be a $3x1$ vectors. Any others sizes will return an error.

Rotation of the stress

Syntax

```
[S2] = Stresses12x(S1,Theta)
```

Description

This function rotates the stress **S1** with an angle **theta** expressed in degrees. The transformation is made with the double cosine and sine functions as presented in table 3.2 (Lessard). The output stress **E2** have the same size as the input strains **E1**. **S1** is a 3×1 array.

$$\boldsymbol{S1} = [\sigma_{xx} \quad \sigma_{yy} \quad \sigma_{ss}]$$

Code

```
function [S2] = Stresses12x(S1,Theta)
Theta = deg2rad(Theta);
%% Rotation of the stress (S1) from an angle (Theta)
% S1 is Sx, Sy, Ss
% Compute intermediary values
p = (S1(1,1) + S1(2,1))/2; % [Pa]
q = (S1(1,1) - S1(2,1))/2; % [Pa]
r = S1(3,1); % [Pa]
```

```
M = [p, q, r; ...
    p, -q, -r; ...
    0, r, -q]; % Define the transformation matrix (table 3.2)
S2 = M * [1; cos(2*Theta); sin(2*Theta)]; % [Pa] Stress in the second coordinate system
End
```

Known errors

The input strains **S1** should be a 3×1 vectors. Any other sizes will return an error.

Rotation of the Q matrix

Syntax

```
[Q2, q2] =Q12x(Q1, Theta)
```

Description

This function rotates the modulus matrix **Q1** matrix with an angle "theta". The transformation is based on the Table 3.9 (Lessard). **Q1** is a 4×1 array.

$$\mathbf{Q1} = [Q_{xx} \quad Q_{yy} \quad Q_{xy} \quad Q_{ss}]$$

The values of U are obtained with the following matrixial expression:

$$U = \frac{1}{8}\begin{bmatrix} 3 & 3 & 2 & 4 \\ 4 & -4 & 0 & 0 \\ 1 & 1 & -2 & 4 \\ 1 & 1 & -2 & 4 \end{bmatrix}[Q_{xx} \quad Q_{yy} \quad Q_{xy} \quad Q_{ss}]$$

The value of **Q2** is the modulus matrix expressed as a 6×1 array. **q2** is the same modulus matrix as **Q2** expressed as a 3×3 matrix shape.

$$\mathbf{Q2} = [Q_{11} \quad Q_{22} \quad Q_{12} \quad Q_{66} \quad Q_{16} \quad Q_{26}]$$

$$\mathbf{q2} = \begin{bmatrix} Q_{11} & Q_{12} & Q_{16} \\ Q_{12} & Q_{22} & Q_{26} \\ Q_{16} & Q_{26} & Q_{66} \end{bmatrix}$$

Code

```
function [Q2, q2] =Q12x(Q1, Theta)
Theta = deg2rad(Theta);
%% Conversion of the stiffness matrix (Q) from in-axis to off-axis from an angle (Theta)
% Q2 is a column vector (1x6) and q2 is a 3x3 matrix shape.
% Compute intermediary values
M1 = [3, 3, 2, 4; ...
    4, -4, 0, 0; ...
    1, 1, -2, -4; ...
    1, 1, 6, -4; ...
    1, 1, -2, 4]; % Define first transformation matrix for U
U = (1/8) * M1 * Q1;
M2 = [U(1), cos(2*Theta), cos(4*Theta); ...
    U(1), -cos(2*Theta), cos(4*Theta); ...
    U(4), 0, -cos(4*Theta); ...
    U(5), 0, -cos(4*Theta); ...
    0, (1/2)*sin(2*Theta), sin(4*Theta); ...
    0, (1/2)*sin(2*Theta), -sin(4*Theta)]; % Define the second transformation matrix for Q
Q2 = M2 * [1; U(2); U(3)]; %[Pa] Stiffness off-axis (Q11, Q22, Q12, Q66, Q16, Q26)
```

```
q2 = [Q2(1), Q2(3), Q2(5); Q2(3), Q2(2), Q2(6); Q2(5), Q2(6), Q2(4)]; %[Pa] Stiffness off-axis in a 3x3 matrix shape.
end
```

Known errors

The input modulus matrix **Q1**should be a 4×1 array. Any other sizes will return an error.

Rotation of the S matrix

Syntax

```
[S2, s2] =S12x(S1, Theta)
```

Description

This function rotates the stiffness matrix **S1** matrix with an angle "theta". The transformation is based on the Table 3.12 (Lessard). **S1** is a 4×1 array.

$$\mathbf{S1} = [Q_{xx} \quad Q_{yy} \quad Q_{xy} \quad Q_{ss}]$$

The values of U are obtained with the following matrixial expression:

$$U = \frac{1}{8}\begin{bmatrix} 3 & 3 & 2 & 1 \\ 4 & -4 & 0 & 0 \\ 1 & 1 & -2 & -1 \\ 4 & 4 & -8 & 4 \end{bmatrix}[S_{xx} \quad S_{yy} \quad S_{xy} \quad S_{ss}]$$

The value of **S2** is the stiffness matrix expressed as a 6×1 array. **s2** is the same stiffness matrix as **s2** expressed as a 3×3 matrix shape.

$$\mathbf{S2} = [S_{11} \quad S_{22} \quad S_{12} \quad S_{66} \quad S_{16} \quad S_{26}]$$

$$\mathbf{s2} = \begin{bmatrix} S_{11} & S_{12} & S_{16} \\ S_{12} & S_{22} & S_{26} \\ S_{16} & S_{26} & S_{66} \end{bmatrix}$$

Code

```
function [S2, s2] =S12x(S1, Theta)
Theta = deg2rad(Theta);
%% Conversion of the compliance matrix (S1) from in-axis to off-axis from an angle (Theta)
% Compute intermediary values
M1 = [3, 3, 2, 1; ...
    4, -4, 0, 0; ...
    1, 1, -2, -1; ...
    1, 1, 6, -1; ...
    4, 4, -8, 4]; % Define first transformation matrix for U
U = (1/8) * M1 * S1;
M2 = [U(1), cos(2*Theta), cos(4*Theta); ...
    U(1), -cos(2*Theta), cos(4*Theta); ...
    U(4), 0, -cos(4*Theta); ...
    U(5), 0, -4*cos(4*Theta); ...
    0, sin(2*Theta), 2*sin(4*Theta); ...
    0, sin(2*Theta), -2*sin(4*Theta)]; % Define the second transformation matrix for S
S2 = M2 * [1; U(2); U(3)]; %[Pa^-1] Compliance off-axis (S11, S22, S12, S66, S16, S26)
s2 = [S2(1), S2(3), S2(5); S2(3), S2(2), S2(6); S2(5), S2(6), S2(4)]; %[Pa^-1] Full compliance off-axis in a 3x3 matrix shape.
end
```

Clement
Brousse
3/22/2022
clbrous@colostate.edu

Known errors

The input stiffness matrix `S1` should be a 4×1 array. Any other sizes will return an error.

Computation Strains and Stress

Syntax

```
[Results] = ComputeStrainStress(Results, Input, TempS, TempQ, Stress, Strain)
```

Description

This function computes either the off-axis strain or the off-axis stress knowing the other off-axis properties. The correct input should be given. The input `Stress` and `Strain` can't be both given at the same time. `Results` contains the detail of each laminate. `Input` contains the detail of the laminate. The laminate is assumed to be symmetric and only the symmetric part should be insert. It is a $n \times 3$ array following this format:

$$\text{Input} = \begin{bmatrix} a_1 & \theta_1 & h_1 \\ a_2 & \theta_2 & h_2 \\ \dots & \dots & \dots \\ a_n & \theta_n & h_n \end{bmatrix}$$

Where n is the total number of plies. a_i is an integer used to select the material (see **Error! Reference source not found.**). The values of θ_i are the laminate orientation and $2z_{c_i}$ is the total layer thickness. `TempS` and `TempQ` are a $4 \times m$ matrices with m the total number of material presented in the database. `Results`, `TempS` and `TempQ` are all obtained with the `ReadDataFileV3` function.

The method to compute the Strain is presented in Figure 3.7 (Lessard) and the method for the stresses in the Figure 3.6 (Lessard).

The output `Results` is a structure. The architecture of the structure is presented in detail later (see Architecture of the Results output). This script add to the existing structure the "Strain" and "Stress" field for the "OnAxis" (on-axis) and the "OffAxis" (off-axis) ass presented in Figure 1. The computation assumed all the input are given in the SI unit.

The computation is done by looking for the on-axis values. The result is checked by using the off-axis S or Q matrices. If the difference is below 10^{-15}, the results are considered correct. Otherwise, a warning displays "`Error for the off axis values`".

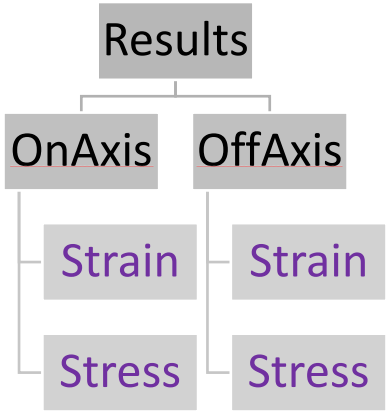

Figure 1: Detail of the `Results` structure as affected by the `ComputeStrainStress` script.

Code

```
function [Results] = ComputeStrainStress(Results, Input, TempS, TempQ, Stress, Strain)
%% This code compute from either the off-axis strain or the off-axis stress the other off-axis
properties.
if exist('Stress','var') % The off-axis stress is given
    for ii = 1:size(Input, 1)
        % First method by the on-axis properties
        Results(ii).OffAxis.Stress = Stress;

        Results(ii).OnAxis.Stress = Stresses12x(Results(ii).OffAxis.Stress, Input(ii,2));

        Results(ii).OnAxis.Strain = Results(ii).OnAxis.S * Results(ii).OnAxis.Stress;

        Results(ii).OffAxis.Strain = Strain12x(Results(ii).OnAxis.Strain, -Input(ii,2));

        % Cross check the results by the off-axis stiffness
        [~, Results(ii).OffAxis.S] = S12x(TempS(:, Input(ii,1)), Input(ii,2)); % Get the off-axis
stiffness
        tempStrain = Results(ii).OffAxis.S * Stress;

        % Compare both method and assume a 10^-15 error is acceptable.
        if abs(tempStrain)-abs(Results(ii).OffAxis.Strain)<10^-15
            disp("Computation correct for the stress")
        else
            warning("Error for the off axis values, ")
            disp(num2str(abs(tempStrain)-abs(Results(ii).OffAxis.Strain)))
        end
    end
elseif  exist('Strain','var') % The off-axis stress is given
    for ii = 1:size(Input, 1)
        Results(ii).OffAxis.Strain = Strain;

        Results(ii).OnAxis.Strain = Strain12x(Results(ii).OffAxis.Strain, Input(ii,2));

        Results(ii).OnAxis.Stress = Results(ii).OnAxis.Q * Results(ii).OnAxis.Strain;

        Results(ii).OffAxis.Stress = Stresses12x(Results(ii).OnAxis.Stress, -Input(ii,2));
        C = Results(ii).OffAxis.Stress;

        % Cross check the results by the off-axis Q
        [~, Results(ii).OffAxis.Q] = Q12x(TempQ(:, Input(ii,1)), -Input(ii,2));
        TempStress = Results(ii).OffAxis.Q * Results(ii).OffAxis.Strain;

        if abs(TempStress)-abs(Results(ii).OffAxis.Q * Results(ii).OffAxis.Strain)<10^-15
            disp("Computation correct for the strain")
        else
            warning("Error for the off axis values")
        end
    end
end
end
```

Known errors

If both **Stress** and **Strain** are given, the code will assume neglect **Strain** and compute the strain. No errors and warning are returned.

Read datafile

Syntax

```
[Results, Name, TempE, TempNu, TempQ, TempS, TempMaxS]=ReadDataFileV3(filepath, Input)
```

Description

This function read a text file with all the material mechanical properties. **filepath** is the path to the text file (see Database) expressed as a **string**. The should be written to accommodate the directory separator for the operating system used. It is usually "\" for Windows systems and "/" for Linux-based systems as MacOS.

Input contains the detail of the laminate. The laminate is assumed to be symmetric and only the symmetric part should be insert. It is a $n \times 3$ array following this format:

$$\text{Input} = \begin{bmatrix} a_1 & \theta_1 & h_1 \\ a_2 & \theta_2 & h_2 \\ \dots & \dots & \dots \\ a_n & \theta_n & h_n \end{bmatrix}$$

Where n is the total number of plies. a_i is an integer used to select the material (see **Error! Reference source not found.**).

Name is a $1 \times m$ cell with the name off all the material presented in the database. **TempE** is a $3 \times m$ double with the values of the in-plane modulus (E_{xx}, E_{yy}, E_{xy}) of each material read. It is expressed in Pa.

$$\mathbf{TempE} = \begin{bmatrix} E_{xx1} & E_{xx2} & \dots & E_{xxm} \\ E_{yy1} & E_{yy2} & \dots & E_{yym} \\ E_{xy1} & E_{xy2} & \dots & E_{xym} \end{bmatrix}$$

TempNu is the list of all the Poisson's ratio (ν) for the material presents in the database. It is a $1 \times m$ double with m the number of materials read. It is unitless.

$$\mathbf{TempNu} = [\nu_1 \quad \nu_2 \quad \dots \quad \nu_m]$$

TempQ, and **TempS** are the list of all the modulus and stiffness component on-axis. They are a $4 \times m$ double with m the number of material read. These are express in Pa for the modulus and Pa^{-1} for the stiffness.

$$\mathbf{TempQ} = \begin{bmatrix} Q_{xx1} & Q_{xx2} & \dots & Q_{xxm} \\ Q_{yy1} & Q_{yy2} & \dots & Q_{yym} \\ Q_{xy1} & Q_{xy2} & \dots & Q_{xym} \\ Q_{ss1} & Q_{ss2} & \dots & Q_{ssm} \end{bmatrix} \quad \mathbf{TempS} = \begin{bmatrix} S_{xx1} & S_{xx2} & \dots & S_{xxm} \\ S_{yy1} & S_{yy2} & \dots & S_{yym} \\ S_{xy1} & S_{xy2} & \dots & S_{xym} \\ S_{ss1} & S_{ss2} & \dots & S_{ssm} \end{bmatrix}$$

TempMaxS is a 5× m double expressed in Pa.

$$\mathbf{TempMaxS} = \begin{bmatrix} X_1 & X_2 & \dots & X_m \\ X_1^{'} & X_2^{'} & \dots & X_m^{'} \\ Y_1 & Y_2 & \dots & Y_m \\ Y_1^{'} & Y_2^{'} & \dots & Y_m^{'} \\ S_1 & S_2 & \dots & S_m \end{bmatrix}$$

The value of the in-plane modulus are computed with Table 4.4 (Lessard). The values computed ($[A^*]$, $[a^*]$ and $[V^*]$) are the normalized by the total height of the laminate. With v_i the normalized height of the i^{th} laminate, h_i the height of the i^{th} laminate and h the total height of the lamina.

$$v_i = \frac{h_i}{h}$$

The values of V^* are computed with the following equations:

$$V_1^* = \sum_{i=1}^{m} cos(2 \times \theta_i)\, v_i$$
$$V_2^* = \sum_{i=1}^{m} cos(4 \times \theta_i)\, v_i$$
$$V_3^* = \sum_{i=1}^{m} sin(2 \times \theta_i)\, v_i$$
$$V_4^* = \sum_{i=1}^{m} sin(2 \times \theta_i)\, v_i$$

The values of $[A_{ii}^*]$ are computed with the Table 4.4 (Lessard).

The data in **Results** are presented in the annex (see Detail of the "Results" structure).

Code

```
function [Results, Name, TempE, TempNu, TempQ, TempS, TempMaxS]=ReadDataFileV3(filepath, Input)
%% Read the data file with all the materials
% Save all the layers properties in the "Results" structure.
xlRange = 'C1:ZZ23';
[TempNum,TempString] = xlsread(filepath,1,xlRange);
% Convert the data tot the correct format
TempE = [TempNum([1:2,4],:)].*1e9;
TempNu = TempNum(3,:);
TempQ = [TempNum(8:11,:)].*1e9;
TempS = [TempNum(12:15,:)].*1e-12;
TempMaxS = [TempNum(16:20,:)].*1e6;
% Create the list of materials
for ii = 1:size(TempString,2)
    Name{1,ii} = strcat(TempString(1,ii), " / ", TempString(2,ii));
end

% Compute each layers properties
for ii = 1:size(Input, 1)
    % On-axis properties
    Results(ii).OnAxis.E = TempE(:, Input(ii,1));
    Results(ii).OnAxis.Nu = TempNu(:, Input(ii,1));
    tempQ = TempQ(:, Input(ii,1)); % Convert Q to the full matrix shape (3x3)
    Results(ii).OnAxis.Q = [tempQ(1), TempQ(3), 0; tempQ(3), TempQ(2), 0; 0, 0, TempQ(4)];
    tempS = TempS(:, Input(ii,1)); % Convert S to the full matrix shape (3x3)
    Results(ii).OnAxis.S = [tempS(1), TempS(3), 0; tempS(3), TempS(2), 0; 0, 0, TempS(4)];
```

```
    Results(ii).OnAxis.S_max = TempMaxS(:, Input(ii,1));

    Results(ii).OffAxis.Vi = Input(ii,3)/sum(Input(:,3)); % Relatif volume of the layer.
    TempVi(ii) = Results(ii).OffAxis.Vi;

    % Off axis properties
    [Results(ii).OffAxis.Q, Results(ii).OffAxis.V1, Results(ii).OffAxis.U] = Q12x(TempQ(:,
Input(ii,1)), -Input(ii,2));
    [Results(ii).OffAxis.S, Results(ii).OffAxis.V2] = S12x(TempS(:, Input(ii,1)), -Input(ii,2));
end

% Computation of the inplane modulus
V = zeros(1,4);
% Compute the Vs* (with the relative heights)
V(1) = sum(cos(2*Input(:,1)) .* [TempVi(:)]);
V(2) = sum(cos(4*Input(:,1)) .* [TempVi(:)]);
V(3) = sum(sin(2*Input(:,1)) .* [TempVi(:)]);
V(4) = sum(sin(2*Input(:,1)) .* [TempVi(:)]);

for ii = 1:size(Input, 1) % Work on eaches laminates
    U = zeros(1:5);
    % Access to the Us
    U = [Results(ii).OffAxis.U];

    % Get the A* matrix (table 4.4)
    Temp1 = [U(1), V(1), V(2); ...
        U(1), -V(1), V(2); ...
        U(4), 0, -V(2); ...
        U(5), 0, -V(2); ...
        0, 0.5*V(3), V(4); ...
        0, 0.5*V(3), -V(4)];
    Temp2 = [1; U(2); U(3)];
    A = Temp1 * Temp2;
    a = [A(1), A(3), A(5); ...
        A(3), A(2), A(6),; ...
        A(5), A(6), A(4)];
    Results(ii).OffAxis.A = A; % Normalised inplane modulus (A11, A22, A12, A66, A16, A26)
    Results(ii).OffAxis.A1 = a; % Normalised inplane modulus with a 3x3 matrix shape
    Results(ii).OffAxis.a = inv(Results(ii).OffAxis.A1); % Normalized inplane compliance with a
3x3 matrix shape
    Results(ii).OffAxis.a1 = [Results(ii).OffAxis.a(1,1); Results(ii).OffAxis.a(2,2);
Results(ii).OffAxis.a(1,2); ...
    Results(ii).OffAxis.a(3,3); Results(ii).OffAxis.a(1,3); Results(ii).OffAxis.a(2,3)]; %
Normalized inplane compliance (a11, a22, a12, a66, a16, a26)
end
end
```

Known errors

The data file should be written as an excel file with the data on the first spreadsheet between the column C and ZZ. The data are between the row 1 and 23. The format of the data should not change from this format.

Clement Brousse
3/22/2022
clbrous@colostate.edu

Verbalization

Syntax

```
[]=VerbalizedInputOutput(Results, Input, Name)
```

Description

This code displays the mechanical parameters used in the computation for each layer. When the parameters are the same for different layer, they are only display once. **Results**, **Input**, **Name** are the output of the **ReadDataFileV3** functions.

Code

```
function []=VerbalizedInputOutput(Results, Input, Name)
%% Display the details of the laminate inputed
C = unique(Input(:,1)); % Get the list of differents materials used
disp(['The given laminate contains ', num2str(size(Input, 1)), ' laminates.'])
k = find([Input(:,1)] == C);
fprintf('\n')
disp("Description of the materials used")
for ii = C % Describe the on-axis properties for each material used
    %     k = find([Input(:,1)] == C); % All the ply with the ii material
    disp(['Plies #', num2str(k'), ' :'])
    disp(['The laminate is ', char(Name{1,ii})])

    disp('The on-axis properties are:')

    disp(['E_x = ', num2str(Results(ii).OnAxis.E(1,1).*1e-9), ' (GPa)'])
    disp(['E_y = ', num2str(Results(ii).OnAxis.E(2,1).*1e-9), ' (GPa)'])
    disp(['E_s = ', num2str(Results(ii).OnAxis.E(3,1).*1e-9), ' (GPa)'])

    disp(['Nu = ', num2str(Results(ii).OnAxis.Nu), ' ( )'])

    disp(['[Q_x] = ', num2str(Results(ii).OnAxis.Q(1,1).*1e-9), ' (GPa)'])
    disp(['[Q_y] = ', num2str(Results(ii).OnAxis.Q(2,2).*1e-9), ' (GPa)'])
    disp(['[Q_xy] = ', num2str(Results(ii).OnAxis.Q(1,2).*1e-9), ' (GPa)'])
    disp(['[Q_s] = ', num2str(Results(ii).OnAxis.Q(3,3).*1e-9), ' (GPa)'])

    disp(['[S_x] = ', num2str(Results(ii).OnAxis.S(1,1).*1e+12), ' (1/TPa)'])
    disp(['[S_y] = ', num2str(Results(ii).OnAxis.S(2,2).*1e+12), ' (1/TPa)'])
    disp(['[S_xy] = ', num2str(Results(ii).OnAxis.S(1,2).*1e+12), ' (1/TPa)'])
    disp(['[S_s] = ', num2str(Results(ii).OnAxis.S(3,3).*1e+12), ' (1/TPa)'])

    disp(['[X] = ', num2str(Results(ii).OnAxis.S_max(1).*1e-6), ' (MPa)'])
    disp(['[X''] = ', num2str(Results(ii).OnAxis.S_max(2).*1e-6), ' (MPa)'])

    disp(['[Y] = ', num2str(Results(ii).OnAxis.S_max(3).*1e-6), ' (MPa)'])
    disp(['[Y''] = ', num2str(Results(ii).OnAxis.S_max(4).*1e-6), ' (MPa)'])

    disp(['S_c =', num2str(Results(ii).OnAxis.S_max(5).*1e-6), ' (MPa)'])

    fprintf('\n')
end

disp("Description of the plies used")
for ii = 1:size(Input, 1) % Describe the off-axis properties for each laminate
    disp(['Plies #', num2str(ii), ' :'])
    disp(['The laminate is ', char(Name{1,ii})])
```

```
    disp('The off-axis properties are:')

    disp(['[Q_11] = ', num2str(Results(ii).OffAxis.Q(1,1).*1e-9), ' (GPa)'])
    disp(['[Q_22] = ', num2str(Results(ii).OffAxis.Q(2,2).*1e-9), ' (GPa)'])
    disp(['[Q_12] = ', num2str(Results(ii).OffAxis.Q(1,2).*1e-9), ' (GPa)'])
    disp(['[Q_66] = ', num2str(Results(ii).OffAxis.Q(3,3).*1e-9), ' (GPa)'])

    disp(['[S_11] = ', num2str(Results(ii).OffAxis.S(1,1).*1e+12), ' (1/TPa)'])
    disp(['[S_22] = ', num2str(Results(ii).OffAxis.S(2,2).*1e+12), ' (1/TPa)'])
    disp(['[S_12] = ', num2str(Results(ii).OffAxis.S(1,2).*1e+12), ' (1/TPa)'])
    disp(['[S_66] = ', num2str(Results(ii).OffAxis.S(3,3).*1e+12), ' (1/TPa)'])

    fprintf('\n')
end
```

Known errors

If the values in **`Results`** are not in SI units, the units expressed will not be corrected.

3/22/2022
clbrous@colostate.edu

Annex

Database

The database is created from data in the Appendix C (Lessard).

	Type	CFRP	GFRP	KFRP	CFRP	CFRTP
	Fiber/cloth	T300	e-glass	Kev 49	AS	AS4
	Matrix	N5208	epoxy	epoxy	H3501	PEEK
Ply constant data	Ex (Gpa)	181	38.6	76	138	134
	Ey (Gpa)	10.3	8.27	5.5	8.96	8.9
	nu/x	0.28	0.26	0.34	0.3	0.28
	Es (Gpa)	7.17	4.14	2.3	7.1	5.1
	v/f	0.7	0.45	0.6	0.66	0.66
	rho	1.6	1.8	1.46	1.6	1.6
	ho (mm)	0.125	0.125	0.125	0.125	0.125
Q @0deg (Gpa)	Qxx	181.8	39.2	76.6	138.8	134.7
	Qyy	10.35	8.39	5.55	9.01	8.95
	Qxy	2.9	2.18	1.89	2.7	2.51
	Qss	7.17	4.14	2.3	7.1	5.1
S @0deg (1/TPa)	Sxx	5.5	7.2	13.2	7.2	7.5
	Syy	97.1	111.6	181.8	111.6	112.4
	Sxy	-1.5	-2.2	-4.5	-2.2	-2.1
	Sss	139.5	140.8	434.8	140.8	196.1
Maximum stress (Mpa)	X	1500	1062	1400	1447	2130
	X'	1500	610	235	1447	1100
	Y	40	31	12	51.7	80
	Y'	246	118	53	206	200
	S	68	72	34	93	160

Table 1: Model of data based used.

Detail of the "Results" structure

The results are save as a structure between the functions. The structure contains the data from the data file in red, the strain and stress computed in purple. The various values computed in the functions are in green.

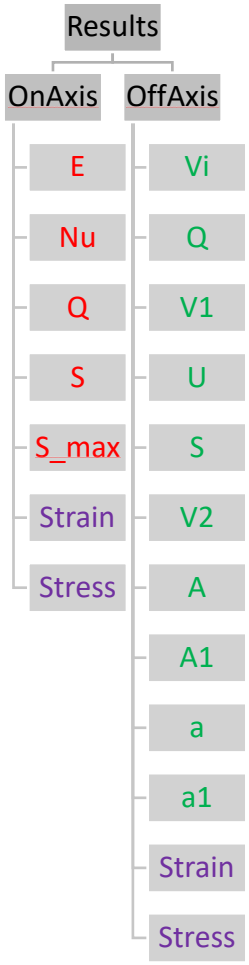

Figure 2: Architecture of the Results output

Bibliography

Lessard, L. (n.d.). *Mechanics of Composite Materials (MECH 535).* Departement of mechanical engineering, McGill University.