Table of Contents


```
clc; clear; close all;
```

ME 456 CLT Project Code

```
Author:     Ryan Melander
Started:    1/6/2022
Last Edit:  2/6/2022
```

Number of Plies

n = # plies vv = variable volume fraction 'yes' or 'no'

```
[n,vv] = numplies;
```

Engineering Parameters

E1 = modulus 1 for each layer (Pa) E2 = modulus 2 for each layer G12 = shear modulus for each layer v12 = major poissons ratio for each layer t = thickness of each layer (m) f = volume fraction of each layer theta = orientation in degrees of each layer (deg)

```
[E1,E2,G12,v12,t,f,theta] = eparam(n,vv);
```

Stiffness Tensors

Q = stiffness tensor for each layer Qbar = stiffness tensor in global frame for each layer

```
[Q,Qbar,S] = Qcalc(n,E1,E2,G12,v12,theta);
```

Macro Stiffness Constants

A = laminate extensional stiffnesses (Pa m) B = laminate coupling stiffnesses (Pa m^2) A = laminate bending stiffnesses (Pa m^3) z =

```
[A,B,D,z] = macrostiffness(Qbar,t,n);
ABBD = [A B; B D];
```

Applied Forces

[M,N] = appforces;

Display Values

```
disp('IMLS matrix, HMS fiber, 0.75 vf')
disp('Q (GPa)')
disp(Q(:,:,1))
disp(['Qbar (GPa), rotation ',num2str(theta),' degrees'])
disp(Qbar(:,:,1))
```

```
IMLS matrix, HMS fiber, 0.75 vf
Q (GPa)
  285.4521    1.3448         0
    1.3448    5.3260         0
         0         0    4.0342

Qbar (GPa), rotation 45 degrees
   77.4011   69.3327   70.0315
   69.3327   77.4011   70.0315
   70.0315   70.0315   72.0221
```

Function numplies

```
function [n,vv] = numplies
% Prompts user to input number of plies and then select yes or no for
% variable volume fraction
    prompt='Input number of plies:';
    name='Number of Plies';
    numlines=1;
    definput= {'1'};
    options.Resize= 'on';
    n = str2double(inputdlg(prompt,name,numlines,definput,options));
    vvf = questdlg({'Variable volume fraction?',...
        '(Yes allows user to input volume fraction)'},...
        'Input','Yes','No','Yes');
    switch vvf
        case 'No'
            vv = 0;
        case 'Yes'
            vv = 1;
    end
end
```

The file content above is properly syntax highlighted

Function eparam

```
function [E1,E2,G12,v12,t,f,theta] = eparam(n,vv)
% Calculates layer properties for variable and nonvariable volume fraction

% Fiber material properties. E values in GPa
listfib = {'Boron','HMS','AS','T300','KEV','S-G','E-G'};
Ef1a = [400 379 213.7 221 152 85.5 73.1];
Ef2a = [400 6.21 13.8 13.8 4.14 85.5 73.1];
vf12a = [0.2 0.2 0.2 0.2 0.35 0.2 0.22];
Gf12a = [167 7.58 13.8 8.96 2.9 35.6 30.1];

% Matrix material properties. E values in GPa
listmat = {'LM','IMLS','IMHS','HM','Polyimide','PMR'};
Ema = [2.21 3.45 3.45 5.17 3.45 3.24];
vma = [0.43 0.41 0.35 0.35 0.35 0.36];
Gma = Ema./(2.*(1+vma));

% Composite material properties. E values in GPa
listcom = {'Boron/5505 boron/epoxy','AS/3501 carbon/epoxy',...
        'IM7/8551-7 carbon/epoxy','AS4/APC2 carbon/PEEK','B4/6061 boron/
aluminum',...
        'Kevlar 49/934 aramid/epoxy','Scotchply 1002 E-glass/epoxy',...
        'E-glass/470-36/vinyl ester'};
Ec1 = [204 139 162 131 235 75.8 38.6 24.4];
Ec2 = [18.5 9 8.34 8.7 137 5.5 8.27 6.87];
Gc12 = [5.59 6.9 2.07 5 47 2.3 4.14 2.89];
vc12 = [0.23 0.3 0.34 0.28 0.3 0.34 0.26 0.32];
fc = [0.5 0.65 0.6 0.58 0.5 0.65 0.45 0.3];

% Preallocating arrays
f = zeros(1,n);
E1 = zeros(1,n);
v12 = zeros(1,n);
E2 = zeros(1,n);
G12 = zeros(1,n);

    if vv == 0  % no variable vf
        % Choose composite for all layers
        [composite] = choosecomposite;
        for i = 1:n
            E1(i) = Ec1(composite);
            v12(i) = vc12(composite);
            E2(i) = Ec2(composite);
            G12(i) = Gc12(composite);
            f(i) = fc(composite);
        end

    elseif vv == 1 % yes variable vf
        % Choose matrix for all layers
        [matrix] = choosematrix;
        for i = 1:n
```

```
            % Choose fiber material for each layer
            [fiber] = choosefiber(i);
            % Fiber/matrix properties for chosen materials
            Ef1 = Ef1a(fiber);
            Ef2 = Ef2a(fiber);
            vf12 = vf12a(fiber);
            Gf12 = Gf12a(fiber);
            Em = Ema(matrix);
            vm = vma(matrix);
            Gm = Gma(matrix);
            % volume fraction input for each layer
            definput= {'0.5'};
            options.Resize= 'on';
            f(i) = str2double(inputdlg(['Input fiber volume fraction between 0
 and 1 for layer ',num2str(i)]...
            ,'Volume Fraction',1,definput,options));
            if f(i) < 0 || f(i) > 1
                f(i) = 0.5;
            end
            % Properties for each layer calculated
            E1(i) = Ef1*f(i) + Em*(1-f(i));
            v12(i) = vf12*f(i) + vm*(1-f(i));
            E2(i) = Em*((1-sqrt(f(i)))+(sqrt(f(i))/(1-(sqrt(f(i))*(1-(Em/
Ef2))))));
            G12(i) = Gm*((1-sqrt(f(i)))+(sqrt(f(i))/(1-(sqrt(f(i))*(1-(Gm/
Gf12))))));
        end
    end

    % Thickness of each layer (mm)
    t = zeros(1,n);
    for i = 1:n
        definput= {'0.25'};
        options.Resize= 'on';
        t(i) = str2double(inputdlg(['Input thickness (mm) for layer
 ',num2str(i)]...
        ,'Thickness',1,definput,options));
        if t(i) <= 0
            t(i) = 0.0025;
        end
    end

    % Orientation angle for each layer
    theta = zeros(1,n);
    for i = 1:n
        definput= {'0'};
        options.Resize= 'on';
        theta(i) = str2double(inputdlg(['Input orientation angle (-90 to 90)
 for layer ',num2str(i)]...
        ,'Orientation',1,definput,options));
        if theta(i) < -90 || theta(i) > 90
            theta(i) = 0;
        elseif theta(i) == -90
            theta(i) = 90;
```

```
        end
    end
end
```

The file content above is properly syntax highlighted

Function choosecomposite

```
function [composite] = choosecomposite
% Allows user to select from premade composite materials for all layers
    list = {'Boron/5505 boron/epoxy','AS/3501 carbon/epoxy',...
        'IM7/8551-7 carbon/epoxy','AS4/APC2 carbon/PEEK','B4/6061 boron/
aluminum',...
        'Kevlar 49/934 aramid/epoxy','Scotchply 1002 E-glass/epoxy',...
        'E-glass/470-36/vinyl ester'};
    [indx,tf] = listdlg('ListString',list,'PromptString',...
        'Select a composite material for all layers','SelectionMode',...
        'single','Name','Composite Material','ListSize',[250 115]);
    if tf == 0
        composite = 1;
    else
        composite = indx;
    end
end
```

The file content above is properly syntax highlighted

Function choosematrix

```
function [matrix] = choosematrix
% Allows user to select matrix material for all layers from 6 materials
    list = {'LM','IMLS','IMHS','HM','Polyimide','PMR'};
    [indx,tf] = listdlg('ListString',list,'PromptString',...
        'Select a matrix material for all layers','SelectionMode',...
        'single','Name','Matrix Material','ListSize',[250 100]);
    if tf == 0
        matrix = 1;
    else
        matrix = indx;
    end
end
```

The file content above is properly syntax highlighted

Function choosefiber

```
function [fiber] = choosefiber(i)
```

```
% Allows user to select fiber material from 7 different materials
    list = {'Boron','HMS','AS','T300','KEV','S-Glass','E-Glass'};
    [indx,tf] = listdlg('ListString',list,'PromptString',...
        ['Select a fiber material for layer ',num2str(i)],'SelectionMode',...
        'single','Name','Fiber Material','ListSize',[250 100]);
    if tf == 0
        fiber = 1;
    else
        fiber = indx;
    end
end
```

The file content above is properly syntax highlighted

Function Qcalc

```
function [Q,Qbar,S] = Qcalc(n,E1,E2,G12,v12,theta)
% Preallocate matrices
    Q = zeros(3,3,n);
    Qbar = zeros(3,3,n);
    S = zeros(3,3,n);
    for i = 1:n
        % Compliance value calculations
        S(1,1,i) = 1/E1(i);
        S(2,2,i) = 1/E2(i);
        S(1,2,i) = -v12(i)/E1(i);
        S(2,1,i) = S(1,2,i);
        S(3,3,i) = 1/G12(i);
        % Stiffness value calculations, fill Q matrix
        Q(1,1,i) = S(2,2,i)/(S(1,1,i)*S(2,2,i) - S(1,2,i)^2);
        Q(1,2,i) = -S(1,2,i)/(S(1,1,i)*S(2,2,i) - S(1,2,i)^2);
        Q(2,1,i) = Q(1,2,i);
        Q(2,2,i) = S(1,1,i)/(S(1,1,i)*S(2,2,i) - S(1,2,i)^2);
        Q(3,3,i) = 1/S(3,3,i);
        % Transformation matrix calculations
        s = sind(theta(i));
        c = cosd(theta(i));
        T = [c^2 s^2 2*c*s; s^2 c^2 -2*c*s; -c*s c*s c^2-s^2];
        T2 = [T(1,1) T(1,2) T(1,3)/2;...
            T(2,1) T(2,2) T(2,3)/2; 2*T(3,1) 2*T(3,2) T(3,3)];
        % Qbar calculations: inv(T)*Q*T2
        Qbar(:,:,i) = T\Q(:,:,i)*T2;
    end
end
```

The file content above is properly syntax highlighted

Function macrostiffness

```
function [A,B,D,z] = macrostiffness(Qbar,t,n)
% A,B,D macrostiffness matrix calculations (eqns: 7.41, 7.42, 7.43)
    % Distances z (mm)
    z = zeros(1,n+1);
    z(1) = -sum(t,'all')/2;
    for i = 1:n
        z(i+1) = z(i) + t(i);
    end
    % Initialize A,B,D matrices
    A = zeros(3);
    B = zeros(3);
    D = zeros(3);
    for i = 1:n
        for r = 1:3
            for c = 1:3
                % A (laminate extensional stiffness) (GPa mm)
                A(r,c) = Qbar(r,c,i)*(z(i+1)-z(i)) + A(r,c);
                % B (laminate coupling stiffnesses) (GPa mm^2)
                B(r,c) = Qbar(r,c,i)*(z(i+1)^2-z(i)^2) + B(r,c);
                % D (laminate bending stiffnesses) (GPa mm^3)
                D(r,c) = Qbar(r,c,i)*(z(i+1)^3-z(i)^3) + D(r,c);
            end
        end
    end
    % Constants after summation (eqns: 7.42, 7.43)
    B = B/2;
    D = D/3;
end
```

The file content above is properly syntax highlighted

Published with MATLAB® R2021b