Table of Contents


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

ME 456 CLT Project Code

```
Author:     Ryan Melander
Started:    1/6/2022
Last Edit:  3/24/2022
```

Number of Plies

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

Engineering Parameters

```
[E1,E2,G12,v12,t,f,theta,NameC,com,NameF,fib,NameM,mat] = eparam(n,vv);
```

Layer Stifnesses

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

Macro Stiffness Constants

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

Applied Forces

```
[NM] = appforces;
```

Stresses

```
[eps0,k,sigmabarT,epsbarT,sigmabarB,epsbarB] = stresses(NM,ABBD,Qbar,n,z);
```

Strength Parameters

```
[SLP,SLM,STP,STM,SLT] = sparam(vv,com,fib,mat,n,f,E1,E2,v12,G12);
```

Failure Check

```
[maxstress,maxstrain,tsai_hill] =
 failurecheck(theta,n,S,SLP,SLM,STP,STM,SLT,sigmabarT,sigmabarB,E1,E2,G12);
```

Ouput Results

```
output(n,NameM,mat,NameF,fib,E1,E2,G12,f,t,theta,maxstrain,maxstress,tsai_hill,ABBD,sigmab
```

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,NameC,com,NameF,fib,NameM,mat] =
 eparam(n,vv)
% Calculates layer properties for variable and nonvariable volume fraction

% Fiber material properties. E values in GPa
NameF = {'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];
fib = zeros(1,n);

% Matrix material properties. E values in GPa
NameM = {'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));
mat = zeros(1,n);

% Composite material properties. E values in GPa
NameC = {'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];
com = zeros(1,n);

% 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);
            com(i) = 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);
            fib(i) = fiber;
            mat(i) = matrix;
            % 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,ABBD] = 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) = ((1/2)*Qbar(r,c,i)*(z(i+1)^2-z(i)^2)) + B(r,c);
                % D (laminate bending stiffnesses) (GPa mm^3)
                D(r,c) = ((1/3)*Qbar(r,c,i)*(z(i+1)^3-z(i)^3)) + D(r,c);
            end
        end
    end
    ABBD = [A B; B D];
end
```

The file content above is properly syntax highlighted

Function appforces

```
function [NM] = appforces
% Allows user to input forces in units of MPa-mm and MPa-mm^2
    prompt = {'Nx (GPa mm):','Ny: (GPa mm)','Nxy: (GPa mm)',...
        'Mx: (GPa mm^2)','My: (GPa mm^2)','Mxy: (GPa mm^2)'};
    dlgtitle = 'Applied Forces';
    numlines = 1;
    definput = {'0','0','0','0','0','0'};
    options.Resize= 'on';
    NM = str2double(inputdlg(prompt,dlgtitle,numlines,definput,options));
end
```

The file content above is properly syntax highlighted

Function stresses

```
function [eps0,k,sigmabarT,epsbarT,sigmabarB,epsbarB] =
 stresses(NM,ABBD,Qbar,n,z)
% Calculates midplane strains and curvatures, and stresses and strains on
% the top and bottom of each layer. For example:
% sigmabarT(:,1) = stress on top of layer 1
% sigmabarB(:,1) = stress on bottom of layer 1
    % Midplane strains and curvatures
    ek = ABBD\NM;
    eps0 = ek(1:3);
    k = ek(4:6);
    % Lamina stresses and strains
    strain = zeros(3,n+1);
    % Strain array between all layers
    for i = 1:n+1
        strain(:,i) = [eps0(1)+z(i)*k(1); eps0(2)+z(i)*k(2);
 eps0(3)+z(i)*k(3)];
    end
    % Strains top and bottom of each layer
    epsbarT = strain(:,1:n);
    epsbarB = strain(:,2:n+1);
    % Stresses top and bottom of each layer eq. 7.34, 7.61 (GPa)
    sigmabarT = zeros(3,n);
    sigmabarB = zeros(3,n);
    for i = 1:n
       sigmabarT(:,i) = Qbar(:,:,i)*epsbarT(:,i);
       sigmabarB(:,i) = Qbar(:,:,i)*epsbarB(:,i);
    end
end
```

The file content above is properly syntax highlighted

Function sparameters

```
function [SLP,SLM,STP,STM,SLT] = sparam(vv,com,fib,mat,n,f,E1,E2,v12,G12)
% Calculates the longitudinal tensile/compressive strengths, transverse
% tensile/compressive strengths, and in-plane shear strength of each lamina
SLP = zeros(1,n);
SLM = zeros(1,n);
STP = zeros(1,n);
STM = zeros(1,n);
SLT = zeros(1,n);

% Fiber strength properties (GPa)
NameF = {'Boron','HMS','AS','T300','KEV','S-G','E-G'};
SLpf = [4.140 1.720 2.410 2.410 2.760 4.140 2.760];
SLmf = [4.830 1.380 1.790 2.070 0.517 0 0];
ef1 = [0.008 0.007 0.018 0.014 0.024 0.057 0.048];
Ef2 = [400 6.21 13.8 13.8 4.14 85.5 73.1];
Gf12 = [167 7.58 13.8 8.96 2.9 35.6 30.1];
fd = [1.4224e-5 7.62e-06 7.62e-06 7.62e-06 1.17e-05 9.14e-06 9.14e-06];
```

```
% Matrix strength properties (GPa)
NameM = {'LM','IMLS','IMHS','HM','Polyimide','PMR'};
Sp = [0.055 0.048 0.1034 0.138 0.103 0.379];
Sm = [0.103 0.145 0.2413 0.345 0.207 0.110];
Sltm = [0.055 0.048 0.090 0.103 0.090 0.055];
eTp = [0.081 0.014 0.02 0.02 0.02 0.02];
Em = [2.21 3.45 3.45 5.17 3.45 3.24];
vm = [0.43 0.41 0.35 0.35 0.35 0.36];
Gm = Em./(2.*(1+vm));

% Composite strength properties (GPa)
NameC = {'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'};
SLPc = [1.586 1.448 2.578 2.06 1.373 1.379 1.103 0.584];
SLMc = [2.482 1.172 1.62 1.08 1.573 0.276 0.621 0.803];
STPc = [0.0627 0.0483 0.0758 0.078 0.118 0.0276 0.0276 0.043];
STMc = [0.241 0.248 0.248 0.196 0.157 0.0648 0.138 0.187];
SLTc = [0.0827 0.0621 0.0621 0.157 0.128 0.06 0.0827 0.065];

    if vv == 0  % no variable volume fraction
        for i = 1:n
            % properties in table 4.1
            SLP(i) = SLPc(com(i));
            SLM(i) = SLMc(com(i));
            STP(i) = STPc(com(i));
            STM(i) = STMc(com(i));
            SLT(i) = SLTc(com(i));
        end

    elseif vv == 1   % yes variable volume fraction
        fs = zeros(1,n);
        F = zeros(1,n);
        Fs = zeros(1,n);
        for i = 1:n
            % longitudinal tensile strength
            SLP(i) = SLpf(fib(i))*f(i) + Sp(mat(i))*(1-f(i));
            % longitudinal compressive strength
            SLM(i) = (E1(i)*eTp(mat(i)))/v12(i);
            % transverse tensile strength
            fs(i) = fd(fib(i))/sqrt(4*f(i)/pi);
            F(i) = 1/(((fd(fib(i))/fs(i)))*((Em(mat(i))/Ef2(fib(i)))-1)+1);
            STP(i) = (E2(i)*Sm(mat(i)))/(Em(mat(i))*F(i));
            % transverse compressive strength
            STM(i) = Sm(mat(i));
            % in-plane shear strength
            Fs(i) = 1/(((fd(fib(i))/fs(i)))*((Gm(mat(i))/Gf12(fib(i)))-1)+1);
            SLT(i) = (G12(i)*Sltm(mat(i)))/(Gm(mat(i))*Fs(i));
        end
    end
 end
```

The file content above is properly syntax highlighted

Function failurecheck

```
function
 [maxstress,maxstrain,tsai_hill]=failurecheck(theta,n,S,SLP,SLM,STP,STM,SLT,sigmabarT,sigm
% -------------------------
    % Rotate stresses into principal frame
    sigmaT = zeros(3,n);
    sigmaB = zeros(3,n);
    epsT = zeros(3,n);
    epsB = zeros(3,n);
    for i = 1:n
        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];
        sigmaT(:,i) = T*sigmabarT(:,i);
        sigmaB(:,i) = T*sigmabarB(:,i);
        epsT(:,i) = S(3,3,i)*sigmaT(:,i);
        epsB(:,i) = S(3,3,i)*sigmaT(:,i);
    end
    % Max Stress
    maxstress = zeros(3,n);
    for i = 1:n
        if -SLM(i) > sigmaT(1,i) || sigmaT(1,i) > SLP(i) || -SLM(i) >
 sigmaB(1,i) || sigmaB(1,i) > SLP(i)
            maxstress(1,i) = 1;
        elseif -STM(i) > sigmaT(2,i) || sigmaT(2,i) > STP(i) || -STM(i) >
 sigmaB(2,i) || sigmaB(2,i) > STP(i)
            maxstress(2,i) = 1;
        elseif abs(sigmaT(3,i)) > SLT(i) || abs(sigmaB(3,i)) > SLT(i)
            maxstress(3,i) = 1;
        end
    end
    % Max Strain
    maxstrain = zeros(3,n);
    for i = 1:n
        eLP = SLP(i)/E1(i);
        eTP = STP(i)/E2(i);
        eLM = SLM(i)/E1(i);
        eTM = STM(i)/E2(i);
        eLT = SLT(i)/G12(i);
        if -eLM > epsT(1,i) || eLP < epsT(1,i) || -eLM > epsB(1,i) || eLP <
 epsB(1,i)
            maxstrain(1,i) = 1;
        elseif -eTM > epsT(2,i) || eTP < epsT(2,i) || -eTM > epsB(2,i) || eTP
 < epsB(2,i)
            maxstrain(2,i) = 1;
        elseif abs(epsT(3,i)) > eLT || abs(epsB(3,i)) > eLT
            maxstrain(3,i) = 1;
        end
    end
```

```
    % Tsai-Hill
    tsai_hill = zeros(1,n);
    SL = 1; ST = 1; % For cases of zeros stress
    for i = 1:n
        % set SL and ST for tension of compression
        if sign(sigmaT(1,i)) > 1
            SL = SLP(i);
        elseif sign(sigmaT(1,i)) < 1
            SL = SLM(i);
        elseif sign(sigmaT(2,i)) > 1
            ST = STP(i);
        elseif sign(sigmaT(2,i)) < 1
            ST = STM(i);
        end
        top = (sigmaT(1,i)^2)/(SL^2) - (sigmaT(1,i)*sigmaT(2,i))/(SL^2) +
 (sigmaT(2,i)^2)/(ST^2) + (sigmaT(3,i)^2)/(SLT(i)^2);
        % set SL and ST for tension of compression
        if sign(sigmaB(1,i)) > 1
            SL = SLP(i);
        elseif sign(sigmaB(1,i)) < 1
            SL = SLM(i);
        elseif sign(sigmaB(2,i)) > 1
            ST = STP(i);
        elseif sign(sigmaB(2,i)) < 1
            ST = STM(i);
        end
        bottom = (sigmaB(1,i)^2)/(SL^2) - (sigmaB(1,i)*sigmaB(2,i))/(SL^2) +
 (sigmaB(2,i)^2)/(ST^2) + (sigmaB(3,i)^2)/(SLT(i)^2);
        % check for failure
        if top >= 1 || bottom >= 1
            tsai_hill(i) = 1;
        end
    end
end
```

The file content above is properly syntax highlighted

Function output

```
function
 output(n,NameM,mat,NameF,fib,E1,E2,G12,f,t,theta,maxstrain,maxstress,tsai_hill,ABBD,sigma
% -----------------------------
filename = 'output_CLT_milestone2.txt';
fid = fopen(filename,'w');

% Title
fprintf(fid,'%s',filename);
fprintf(fid,'\t\tNumber of plies: %s',num2str(n));
fprintf(fid,'\t\tGenerated %s \r\n',date);
fprintf(fid,'Units: (GPa,mm) \r\n');
```

```
% Laminate Info
% add for vv = 1 or 0
fprintf(fid,'\n\nLayup Info:');
fprintf(fid,'\n\tMatrix\t\tFiber\tVf\t\tThick\tTheta\tE1\t\t\tE2\t\tG12');
for i = 1:n
    fprintf(fid,'\n%s',num2str(i));
    fprintf(fid,'\t%s',string(NameM(mat(i))));
    fprintf(fid,'\t%s',string(NameF(fib(i))));
    fprintf(fid,'\t%4.2f',f(i));
    fprintf(fid,'\t%4.2f',t(i));
    fprintf(fid,'\t%4.0f',theta(i));
    fprintf(fid,'\t%4.2f',E1(i));
    fprintf(fid,'\t\t%4.2f',E2(i));
    fprintf(fid,'\t%4.2f',G12(i));
end

% Applied Forces
fprintf(fid,'\n\nApplied Forces');
fprintf(fid,'\n%4.3f',NM);

% Failure
fprintf(fid,'\n\n\nFailure Criteria (1 signifies failure of layer for given
 criteria');
fprintf(fid,'\n\nMax Strain');
fprintf(fid,'\n\t\tLong\tTran\tShear');
for i = 1:n
      fprintf(fid,'\n%s',num2str(i));
      fprintf(fid,'\t\t%s',num2str(maxstrain(1,i)));
      fprintf(fid,'\t\t%s',num2str(maxstrain(2,i)));
      fprintf(fid,'\t\t%s',num2str(maxstrain(3,i)));
end
fprintf(fid,'\n\nMax Stress');
fprintf(fid,'\n\t\tLong\tTran\tShear');
for i = 1:n
      fprintf(fid,'\n%s',num2str(i));
      fprintf(fid,'\t\t%s',num2str(maxstress(1,i)));
      fprintf(fid,'\t\t%s',num2str(maxstress(2,i)));
      fprintf(fid,'\t\t%s',num2str(maxstress(3,i)));
end
fprintf(fid,'\n\nTsai-Hill');
for i = 1:n
      fprintf(fid,'\n%s',num2str(i));
      fprintf(fid,'\t\t%s',num2str(tsai_hill(i)));
end

% Stiffness, Stress, Strain
fprintf(fid,'\n\n\nStiffness, Stress, and Strain');
fprintf(fid,'\n\nABBD Stiffness Matrix:');
fprintf(fid,'\n%4.2f\t\t%4.2f\t\t%4.2f\t\t%4.2f\t\t%4.2f\t\t%4.2f',ABBD);
fprintf(fid,'\n\nStresses (rows = plies, columns = x y tau');
fprintf(fid,'\n\nTop');
fprintf(fid,'\n%4.3f\t\t%4.3f\t\t%4.3f\t\t%4.3f\t\t%4.3f\t\t%4.3f',sigmabarT);
fprintf(fid,'\n\nBottom');
fprintf(fid,'\n%4.3f\t\t%4.3f\t\t%4.3f\t\t%4.3f\t\t%4.3f\t\t%4.3f',sigmabarB);
```

The file content above is properly syntax highlighted

Published with MATLAB® R2022a