Table of Contents


```
function [LaminateResults] = classicalLaminatev4( properties, angleVec,
layerThickness, midSurfStrains, forceResultants, deltaT, deltaM )

    % Computing the z-coordinates for the layer interfaces
    N = length(angleVec); % Extracting number of layers
    H = N*layerThickness; % computing total laminate thickness, assuming
constant layer thickness
    zCoord = linspace(-H/2, H/2, N+1); % computing z-coordinates

    % Initializing Qbar, alpha, and beta for each layer
    QbarLayer = zeros(3,3,N); % 3x3 matrix for each ply
    alphaLayerXYZ = zeros(3,1,N); % alpha_x, alpha_y, alpha_xy for each ply
    betaLayerXYZ = zeros(3,1,N); % beta_x, beta_y, beta_xy for each ply

    % Initializing ABD matrix
    Amat = zeros(3,3); % Initializing A matrix
    Bmat = zeros(3,3); % Initializing B matrix
    Dmat = zeros(3,3); % Initializing D matrix

    % Initialzing Unit Thermal Stress Resultants
    Nxhat_T = 0;
    Nyhat_T = 0;
    Nxyhat_T = 0;
    Mxhat_T = 0;
    Myhat_T = 0;
    Mxyhat_T = 0;

    % Initializing Unit Moisture Stress Resultants
    Nxhat_M = 0;
    Nyhat_M = 0;
    Nxyhat_M = 0;
    Mxhat_M = 0;
    Myhat_M = 0;
    Mxyhat_M = 0;

    zk = zCoord(2:end); % from second to last coordinates
    zkMinus1 = zCoord(1:end-1); % from first to second-to-last coordinates
    dz = zk - zkMinus1; % term used to calculate A matrices
    dzSquared = zk.^2 - zkMinus1.^2; % term used to calculate B matrices
    dzCubed = zk.^3 - zkMinus1.^3; % term used to calculate D matrices

    for plyIdx = 1:N
```

```
        theta = angleVec(plyIdx); % extracting the orientation angle
        QbarLayer(:,:,plyIdx) = TransformedRSM(properties, theta); % Using
Qbar function
        alphaLayerXYZ(:,1,plyIdx) = ThermalDeformCoeff(properties, theta); %
Using Alpha function
        betaLayerXYZ(:,1,plyIdx) = MoistureDeformCoeff(properties, theta); %
Using Beta function

        Amat = Amat + QbarLayer(:,:,plyIdx)*dz(plyIdx); % A matrix -
computed for each layer and then summed for all layers.
        Bmat = Bmat + (1/2)*QbarLayer(:,:,plyIdx)*dzSquared(plyIdx); % B
matrix - computed for each layer and then summed for all layers.
        Dmat = Dmat + (1/3)*QbarLayer(:,:,plyIdx)*dzCubed(plyIdx); % D
matrix - computed for each layer and then summed for all layers.

        % Computing Unit Thermal Stress Resultants for each layer, then
summed for all layers.
        Nxhat_T = Nxhat_T
+ ( QbarLayer(1,1,plyIdx)*alphaLayerXYZ(1,1,plyIdx)
+ QbarLayer(1,2,plyIdx)*alphaLayerXYZ(2,1,plyIdx) +
QbarLayer(1,3,plyIdx)*alphaLayerXYZ(3,1,plyIdx) )*dz(plyIdx);
        Nyhat_T = Nyhat_T
+ ( QbarLayer(2,1,plyIdx)*alphaLayerXYZ(1,1,plyIdx)
+ QbarLayer(2,2,plyIdx)*alphaLayerXYZ(2,1,plyIdx) +
QbarLayer(2,3,plyIdx)*alphaLayerXYZ(3,1,plyIdx) )*dz(plyIdx);
        Nxyhat_T = Nxyhat_T
+ ( QbarLayer(3,1,plyIdx)*alphaLayerXYZ(1,1,plyIdx)
+ QbarLayer(3,2,plyIdx)*alphaLayerXYZ(2,1,plyIdx) +
QbarLayer(3,3,plyIdx)*alphaLayerXYZ(3,1,plyIdx) )*dz(plyIdx);

        Mxhat_T = Mxhat_T
+ 0.5*( QbarLayer(1,1,plyIdx)*alphaLayerXYZ(1,1,plyIdx)
+ QbarLayer(1,2,plyIdx)*alphaLayerXYZ(2,1,plyIdx) +
QbarLayer(1,3,plyIdx)*alphaLayerXYZ(3,1,plyIdx) )*dzSquared(plyIdx);
        Myhat_T = Myhat_T
+ 0.5*( QbarLayer(2,1,plyIdx)*alphaLayerXYZ(1,1,plyIdx)
+ QbarLayer(2,2,plyIdx)*alphaLayerXYZ(2,1,plyIdx) +
QbarLayer(2,3,plyIdx)*alphaLayerXYZ(3,1,plyIdx) )*dzSquared(plyIdx);
        Mxyhat_T = Mxyhat_T
+ 0.5*( QbarLayer(3,1,plyIdx)*alphaLayerXYZ(1,1,plyIdx)
+ QbarLayer(3,2,plyIdx)*alphaLayerXYZ(2,1,plyIdx) +
QbarLayer(3,3,plyIdx)*alphaLayerXYZ(3,1,plyIdx) )*dzSquared(plyIdx);

        % Computing Unit Moisture Stress Resultants for each layer, then
summed for all layers
        Nxhat_M = Nxhat_M + ( QbarLayer(1,1,plyIdx)*betaLayerXYZ(1,1,plyIdx)
+ QbarLayer(1,2,plyIdx)*betaLayerXYZ(2,1,plyIdx) +
QbarLayer(1,3,plyIdx)*betaLayerXYZ(3,1,plyIdx) )*dz(plyIdx);
        Nyhat_M = Nyhat_M + ( QbarLayer(2,1,plyIdx)*betaLayerXYZ(1,1,plyIdx)
+ QbarLayer(2,2,plyIdx)*betaLayerXYZ(2,1,plyIdx) +
QbarLayer(2,3,plyIdx)*betaLayerXYZ(3,1,plyIdx) )*dz(plyIdx);
        Nxyhat_M = Nxyhat_M
+ ( QbarLayer(3,1,plyIdx)*betaLayerXYZ(1,1,plyIdx)
+ QbarLayer(3,2,plyIdx)*betaLayerXYZ(2,1,plyIdx) +
```

```
QbarLayer(3,3,plyIdx)*betaLayerXYZ(3,1,plyIdx) )*dz(plyIdx);

        Mxhat_M = Mxhat_M
+ 0.5*( QbarLayer(1,1,plyIdx)*betaLayerXYZ(1,1,plyIdx)
+ QbarLayer(1,2,plyIdx)*betaLayerXYZ(2,1,plyIdx) +
QbarLayer(1,3,plyIdx)*betaLayerXYZ(3,1,plyIdx) )*dzSquared(plyIdx);
        Myhat_M = Myhat_M
+ 0.5*( QbarLayer(2,1,plyIdx)*betaLayerXYZ(1,1,plyIdx)
+ QbarLayer(2,2,plyIdx)*betaLayerXYZ(2,1,plyIdx) +
QbarLayer(2,3,plyIdx)*betaLayerXYZ(3,1,plyIdx) )*dzSquared(plyIdx);
        Mxyhat_M = Mxyhat_M
+ 0.5*( QbarLayer(3,1,plyIdx)*betaLayerXYZ(1,1,plyIdx)
+ QbarLayer(3,2,plyIdx)*betaLayerXYZ(2,1,plyIdx) +
QbarLayer(3,3,plyIdx)*betaLayerXYZ(3,1,plyIdx) )*dzSquared(plyIdx);

        % Displaying Per-Ply Properties
        format short g
        fprintf('--- Ply %d ---\n', plyIdx);
        fprintf('Orientation: %d°\n', theta);
        fprintf('[Qbar] = \n'); disp(QbarLayer(:,:,plyIdx));
        fprintf('α_x: %.3e\n',alphaLayerXYZ(1,1,plyIdx));
        fprintf('α_y: %.3e\n', alphaLayerXYZ(2,1,plyIdx));
        fprintf('α_xy: %.3e\n', alphaLayerXYZ(3,1,plyIdx));
        fprintf('β_x: %.3e\n',betaLayerXYZ(1,1,plyIdx));
        fprintf('β_y: %.3e\n',betaLayerXYZ(2,1,plyIdx));
        fprintf('β_xy: %.3e\n',betaLayerXYZ(3,1,plyIdx));
        fprintf('\n')
        format short
    end

    % Storing Layer Results
    LaminateResults.Qbar = QbarLayer;
    LaminateResults.alpha = alphaLayerXYZ;
    LaminateResults.beta = betaLayerXYZ;

    % Assembling ABD Matrix
    ABDmat = [Amat Bmat;
              Bmat Dmat];

    % Calculating abd matrix
    abdmat = (ABDmat) \ eye(6); % this computes the inverse of the ABD matrix
    amat = abdmat(1:3,1:3); % extracting the a matrix
    bmat = abdmat(1:3,4:6); % extracting the b matrix
    dmat = abdmat(4:6,4:6); % extracting the d

    % Storing ABD and abd Results
    LaminateResults.ABD = ABDmat;
    LaminateResults.abd = abdmat;
    LaminateResults.A = Amat;
    LaminateResults.B = Bmat;
    LaminateResults.D = Dmat;
    LaminateResults.a = amat;
    LaminateResults.b = bmat;
    LaminateResults.d = dmat;
```

```
% Storing Unit Thermal Stress Resultants
LaminateResults.Nxhat_T = Nxhat_T;
LaminateResults.Nyhat_T = Nyhat_T;
LaminateResults.Nxyhat_T = Nxyhat_T;
LaminateResults.Mxhat_T = Mxhat_T;
LaminateResults.Myhat_T = Myhat_T;
LaminateResults.Mxyhat_T = Mxyhat_T;

% Storing Unit Moisture Stress Resultants
LaminateResults.Nxhat_M = Nxhat_M;
LaminateResults.Nyhat_M = Nyhat_M;
LaminateResults.Nxyhat_M = Nxyhat_M;
LaminateResults.Mxhat_M = Mxhat_M;
LaminateResults.Myhat_M = Myhat_M;
LaminateResults.Mxyhat_M = Mxyhat_M;

% Calculating Effective Engineering Properties
Ebarx = (Amat(1,1).*Amat(2,2) - Amat(1,2).^2)/(Amat(2,2).*H) ;
Ebary = (Amat(1,1).*Amat(2,2) - Amat(1,2).^2)/(Amat(1,1).*H) ;
Gbarxy = Amat(3,3)./H ;
vbarxy = Amat(1,2)./Amat(2,2) ;
vbaryx = Amat(1,2)./Amat(1,1) ;

% Storing Effective Engineering Properties
LaminateResults.Ebarx = Ebarx;
LaminateResults.Ebary = Ebary;
LaminateResults.Gbarxy = Gbarxy;
LaminateResults.vbarxy = vbarxy;
LaminateResults.vbaryx = vbaryx;

% Printing the Unit Thermal and Moisture Resultants
format short g
fprintf('------ Unit Thermal Stress Resultants ------\n')
fprintf('Nxhat_T = %.3f\n', Nxhat_T)
fprintf('Nyhat_T = %.3f\n', Nyhat_T)
fprintf('Nxyhat_T = %.3f\n', Nxyhat_T)
fprintf('Mxhat_T = %.3f\n', Mxhat_T)
fprintf('Myhat_T = %.3f\n', Myhat_T)
fprintf('Mxyhat_T = %.3f\n', Mxyhat_T)
fprintf('\n')

fprintf('------ Unit Moisture Stress Resultants ------\n')
fprintf('Nxhat_M = %.3f\n', Nxhat_M)
fprintf('Nyhat_M = %.3f\n', Nyhat_M)
fprintf('Nxyhat_M = %.3f\n', Nxyhat_M)
fprintf('Mxhat_M = %.3f\n', Mxhat_M)
fprintf('Myhat_M = %.3f\n', Myhat_M)
fprintf('Mxyhat_M = %.3f\n', Mxyhat_M)
fprintf('\n')

% Printing ABD and abd matrices
format short g
fprintf('----------------------- [ABD] -----------------------\n')
```

```
    fprintf('[A] = \n'); disp(Amat);
    fprintf('[B] = \n'); disp(Bmat);
    fprintf('[D] = \n'); disp(Dmat);
    fprintf('[ABD] = \n'); disp(ABDmat);
    fprintf('\n')
    fprintf('----------------------- [abd] -----------------------\n')
    fprintf('[a] = \n'); disp(amat);
    fprintf('[b] = \n'); disp(bmat);
    fprintf('[d] = \n'); disp(dmat);
    fprintf('[abd] = \n'); disp(abdmat);
    fprintf('\n')
    format short

    % Printing Effective Engineering Properties
    format short g
    fprintf('------ Effective Engineering Properties ------\n')
    fprintf('Ebarx = %.3e\n', Ebarx)
    fprintf('Ebary = %.3e\n', Ebary)
    fprintf('Gbarxy = %.3e\n', Gbarxy)
    fprintf('vbarxy = %.3e\n', vbarxy)
    fprintf('vbaryx = %.3e\n', vbaryx)
    fprintf('\n')
    fprintf('WARNING: THESE EFFECTIVE PROPERTIES ARE ONLY VALID FOR
BALANCED, SYMMETRIC LAMINATES! \n')
    fprintf('\n')

Not enough input arguments.

Error in classicalLaminatev4 (line 4)
    N = length(angleVec); % Extracting number of layers
               ^^^^^^^^
```

Computing Midsurface Strains/Stresses

```
    % Computing Thermal and Moisture Effects
    % Use Equation 11.69 from Hyer
    Nx_T = Nxhat_T*deltaT;
    Ny_T = Nyhat_T*deltaT;
    Nxy_T = Nxyhat_T*deltaT;
    Mx_T = Mxhat_T*deltaT;
    My_T = Myhat_T*deltaT;
    Mxy_T = Mxyhat_T*deltaT;

    Nx_M = Nxhat_M*deltaM;
    Ny_M = Nyhat_M*deltaM;
    Nxy_M = Nxyhat_M*deltaM;
    Mx_M = Mxhat_M*deltaM;
    My_M = Myhat_M*deltaM;
    Mxy_M = Mxyhat_M*deltaM;

    % Branch 1: Known Midsurface Strains, Unknown Force and Moment
    % Resultants
    if all(isnan(struct2array(forceResultants))) % If the force resultants
```

```
are not given/unkown
        % Extracting specified midsurface strains from input structure
        eps_x0 = midSurfStrains.eps_x0;
        eps_y0 = midSurfStrains.eps_y0;
        gamma_xy0 = midSurfStrains.gamma_xy0;
        K_x0 = midSurfStrains.K_x0;
        K_y0 = midSurfStrains.K_y0;
        K_xy0 = midSurfStrains.K_xy0;

        eps0vecXYZ = [eps_x0; eps_y0; gamma_xy0]; % building vector of
midsurface strains
        KvecXYZ = [K_x0; K_y0; K_xy0]; % building vector of midsurface
curvatures
        epsilonKappa0 = [eps0vecXYZ; KvecXYZ];

        % Compute the force and moment resultants
        forceMoment = ABDmat*epsilonKappa0;
        forcevec = forceMoment(1:3);
        momentvec = forceMoment(4:6);

        % Display Applied Mid-Surface Strains and Curvatures
        fprintf('----------------------- Applied Mid-Surface Strains and
Curvatures -----------------------\n')
        fprintf(' εx0 = %g\n', eps_x0);
        fprintf(' εy0 = %g\n', eps_y0);
        fprintf(' γxy0 = %g\n', gamma_xy0);
        fprintf(' Kx0 = %g\n', K_x0);
        fprintf(' Ky0 = %g\n', K_y0);
        fprintf(' Kxy0 = %g\n', K_xy0);
        fprintf('\n')

        % Display Resultant Forces and Moments
        fprintf('----------------------- Resultant Forces and Moments
-----------------------\n')
        fprintf(' Nx = %d\n', forcevec(1));
        fprintf(' Ny = %d\n', forcevec(2));
        fprintf(' Nxy = %d\n', forcevec(3));
        fprintf(' Mx = %d\n', momentvec(1));
        fprintf(' My = %d\n', momentvec(2));
        fprintf(' Mxy = %d\n', momentvec(3));

        % Display Thermal Forces and Moments
        fprintf('----------------------- Thermal Forces and Moments
-----------------------\n')
        fprintf(' NxT = %d\n', Nx_T);
        fprintf(' NyT = %d\n', Ny_T);
        fprintf(' NxyT = %d\n', Nxy_T);
        fprintf(' MxT = %d\n', Mx_T);
        fprintf(' MyT = %d\n', My_T);
        fprintf(' MxyT = %d\n', Mxy_T);

        % Display Moisture Forces and Moments
        fprintf('----------------------- Moisture Forces and Moments
-----------------------\n')
```

```
        fprintf(' NxM = %d\n', Nx_M);
        fprintf(' NyM = %d\n', Ny_M);
        fprintf(' NxyM = %d\n', Nxy_M);
        fprintf(' MxM = %d\n', Mx_M);
        fprintf(' MyM = %d\n', My_M);
        fprintf(' MxyM = %d\n', Mxy_M);

    % Branch 2: Known Stress Resultants, Unknown Midsurface Strains
    elseif all(isnan(struct2array(midSurfStrains))) % if the midsurface
strains/curvatures are not given/unknown
        % Extracting specified force resultants from input structure
        N_x = forceResultants.N_x;
        N_y = forceResultants.N_y;
        N_xy = forceResultants.N_xy;
        M_x = forceResultants.M_x;
        M_y = forceResultants.M_y;
        M_xy = forceResultants.M_xy;

        % Using Equation 11.66 from Hyer to include thermal and moisture
effects
        forceMoment = [N_x + Nx_T + Nx_M;
                       N_y + Ny_T + Ny_M;
                       N_xy + Nxy_T + Nxy_M;
                       M_x + Mx_T + Mx_M;
                       M_y + My_T + My_M;
                       M_xy + Mxy_T + Mxy_M;];

        epsilonKappa0 = abdmat*forceMoment;
        eps0vecXYZ = epsilonKappa0(1:3); % use this for later calculations
        KvecXYZ = epsilonKappa0(4:6); % use this for later calculations

        % Display Applied Forces and Moments
        fprintf('----------------------- Applied Forces and Moments
-----------------------\n')
        fprintf(' Nx = %d\n', N_x);
        fprintf(' Ny = %d\n', N_y);
        fprintf(' Nxy = %d\n', N_xy);
        fprintf(' Mx = %d\n', M_x);
        fprintf(' My = %d\n', M_y);
        fprintf(' Mxy = %d\n', M_xy);
        fprintf('\n')

        % Display Resultant Mid-Surface Strains and Curvatures
        fprintf('----------------------- Resultant Mid-Surface Strains and
Curvatures -----------------------\n')
        fprintf(' εx0 = %g\n', eps0vecXYZ(1));
        fprintf(' εy0 = %g\n', eps0vecXYZ(2));
        fprintf(' γxy0 = %g\n', eps0vecXYZ(3));
        fprintf(' Kx0 = %g\n', KvecXYZ(1));
        fprintf(' Ky0 = %g\n', KvecXYZ(2));
        fprintf(' Kxy0 = %g\n', KvecXYZ(3));
        fprintf('\n')

        % Display Thermal Forces and Moments
```

```
        fprintf('----------------------- Thermal Forces and Moments 
-----------------------\n')
        fprintf(' NxT = %d\n', Nx_T);
        fprintf(' NyT = %d\n', Ny_T);
        fprintf(' NxyT = %d\n', Nxy_T);
        fprintf(' MxT = %d\n', Mx_T);
        fprintf(' MyT = %d\n', My_T);
        fprintf(' MxyT = %d\n', Mxy_T);

        % Display Moisture Forces and Moments
        fprintf('----------------------- Moisture Forces and Moments 
-----------------------\n')
        fprintf(' NxM = %d\n', Nx_M);
        fprintf(' NyM = %d\n', Ny_M);
        fprintf(' NxyM = %d\n', Nxy_M);
        fprintf(' MxM = %d\n', Mx_M);
        fprintf(' MyM = %d\n', My_M);
        fprintf(' MxyM = %d\n', Mxy_M);

    % Branch 3: If Neither ForceResultants or midSurfStrains are specified.
    elseif all(isnan(struct2array(forceResultants))) && 
all(isnan(struct2array(midSurfStrains)))
        % This is just for a case where I want the code to just compute the
        % ABD and abd matrices for a particular laminate
        eps0vecXYZ = zeros(3,1);
        KvecXYZ = zeros(3,1);
        disp(" THE CODE IS ONLY COMPUTING ABD AND abd MATRICES. IGNORE 
STRESS AND STRAIN RESULTS.")
    else
        error('ERROR: ONLY SPECIFY EITHER FORCES AND MOMENTS OR MIDSURFACE 
STRAINS AND CURVATURES.')
    end
```

Starting per Layer Computations (Strains and Stresses)

```
    epsLayerXYZ = zeros(3,2,N); % per layer, both top and bottom surface 
Total strains in XYZ
    sigmaLayerXYZ = zeros(3,2,N); % per layer, both top and bottom surface 
stresses in XYZ

    epsLayerXYZ_mech = zeros(3,2,N); % per layer, both top and bottom 
surface mechanical strains in XYZ

    epsLayer123 = zeros(3,2,N); % per layer, both top and bottom surface 
strains in 123
    sigmaLayer123 = zeros(3,2,N); % per layer, both top and bottom surface 
stresses in 123
```

Compute strains in XYZ coordinate system at each layer interface

```
    % Compute epsilon and sigma for each layer at its top and bottom
    for plyIdx = 1:N
        theta = angleVec(plyIdx); % extracting the orientation angle

        % Top and Bottom Layer Calculations
        plyIdxTop = plyIdx; % Index of ply top surface
        plyIdxBot = plyIdx + 1; % Index of ply bottom surface
        zTop = zCoord(plyIdxTop); % zCoord of top surface
        zBot = zCoord(plyIdxBot);  % zCoord of bottom surface

        % Total Strains in XYZ
        epsLayerXYZ(:,1,plyIdx) = eps0vecXYZ + zTop.*KvecXYZ; % Top Surface
        epsLayerXYZ(:,2,plyIdx) = eps0vecXYZ + zBot.*KvecXYZ; % Bottom
Surface

        % Mechanical Strains in XYZ
        epsLayerXYZ_mech(:,1,plyIdx) = eps0vecXYZ + zTop.*KvecXYZ -
alphaLayerXYZ(:,1,plyIdx)*deltaT - betaLayerXYZ(:,1,plyIdx)*deltaM ; % Top
Surface
        epsLayerXYZ_mech(:,2,plyIdx) = eps0vecXYZ + zBot.*KvecXYZ -
alphaLayerXYZ(:,1,plyIdx)*deltaT - betaLayerXYZ(:,1,plyIdx)*deltaM ; %
Bottom Surface

        % Stresses in XYZ
        sigmaLayerXYZ(:,1,plyIdx) =
QbarLayer(:,:,plyIdx)*epsLayerXYZ_mech(:,1,plyIdx); % Top Surface
        sigmaLayerXYZ(:,2,plyIdx) =
QbarLayer(:,:,plyIdx)*epsLayerXYZ_mech(:,2,plyIdx); % Bottom Surface

        % Transforming Strains in XYZ to 123
        m = cosd(theta);
        n = sind(theta);

        % General Form Transformation Matrix
        T = [m.^2   n.^2     2*m.*n;
             n.^2   m.^2    -2*m.*n;
             -m.*n  m.*n   m.^2 - n.^2];

        % Reuter Matrix (for strain transformation)
        R = [1 0 0 ;
             0 1 0 ;
             0 0 2];

        % Strains in 123
        epsLayer123(:,1,plyIdx) = R*T*R^-1*epsLayerXYZ(:,1,plyIdx); % Top
Surface
        epsLayer123(:,2,plyIdx) = R*T*R^-1*epsLayerXYZ(:,2,plyIdx); % Bottom
Surface
```

```
        % Stresses in 123
        sigmaLayer123(:,1,plyIdx) = T*sigmaLayerXYZ(:,1,plyIdx); % Top
Surface
        sigmaLayer123(:,2,plyIdx) = T*sigmaLayerXYZ(:,2,plyIdx); % Bottom
Surface
    end

    % Store results for each layer
    LaminateResults.zCoord = zCoord;
    LaminateResults.epsLayerXYZ = epsLayerXYZ;
    LaminateResults.sigmaLayerXYZ = sigmaLayerXYZ;
    LaminateResults.epsLayer123 = epsLayer123;
    LaminateResults.sigmaLayer123 = sigmaLayer123;
```

Creating Tabular Output for Stresses and Strains in XYZ

```
    % Strain Table
    LayerNum = zeros(2*N,1);
    LayerZCoord = zeros(2*N,1);
    LayerEpsX = zeros(2*N,1);
    LayerEpsY = zeros(2*N,1);
    LayerGamXY = zeros(2*N,1);

    for plyIdx = 1:N
        LayerNum(2*plyIdx-1) = plyIdx; % Layer number for top interface
        LayerNum(2*plyIdx) = plyIdx;     % Layer number for bottom interface
        LayerZCoord(2*plyIdx-1) = zCoord(plyIdx); % Z-coordinate for top
interface
        LayerZCoord(2*plyIdx) = zCoord(plyIdx + 1); % Z-coordinate for
bottom interface
        LayerEpsX(2*plyIdx-1) = epsLayerXYZ(1,1,plyIdx); % Epsilon X for top
interface
        LayerEpsX(2*plyIdx) = epsLayerXYZ(1,2,plyIdx);     % Epsilon X for
bottom interface
        LayerEpsY(2*plyIdx-1) = epsLayerXYZ(2,1,plyIdx); % Epsilon Y for top
interface
        LayerEpsY(2*plyIdx) = epsLayerXYZ(2,2,plyIdx);     % Epsilon Y for
bottom interface
        LayerGamXY(2*plyIdx-1) = epsLayerXYZ(3,1,plyIdx); % Gamma XY for top
interface
        LayerGamXY(2*plyIdx) = epsLayerXYZ(3,2,plyIdx);     % Gamma XY for
bottom interface
    end

    strainValuesXYZ = table(LayerNum, LayerZCoord, LayerEpsX, LayerEpsY,
LayerGamXY,...
                         'VariableNames',{'Layer', 'z', 'εx', 'εy', 'γxy'});

    LaminateResults.StrainTableXYZ = strainValuesXYZ;
    fprintf('----------------------Strain Results (XYZ)----------------------
\n')
```

```
    fprintf('\n')
    disp(strainValuesXYZ)

    % Stress Table
    LayerSigmaX = zeros(2*N,1);
    LayerSigmaY = zeros(2*N,1);
    LayerTauXY = zeros(2*N,1);
    for plyIdx = 1:N
        LayerSigmaX(2*plyIdx-1) = sigmaLayerXYZ(1,1,plyIdx); % Sigma X for
top interface
        LayerSigmaX(2*plyIdx) = sigmaLayerXYZ(1,2,plyIdx);      % Sigma X for
bottom interface
        LayerSigmaY(2*plyIdx-1) = sigmaLayerXYZ(2,1,plyIdx); % Sigma Y for
top interface
        LayerSigmaY(2*plyIdx) = sigmaLayerXYZ(2,2,plyIdx);      % Sigma Y for
bottom interface
        LayerTauXY(2*plyIdx-1) = sigmaLayerXYZ(3,1,plyIdx); % Tau XY for top
interface
        LayerTauXY(2*plyIdx) = sigmaLayerXYZ(3,2,plyIdx);      % Tau XY for
bottom interface
    end

    stressValuesXYZ = table(LayerNum, LayerZCoord, LayerSigmaX, LayerSigmaY,
LayerTauXY,...
                            'VariableNames',{'Layer', 'z', 'σx', 'σy',
'τxy'});
    LaminateResults.StressTableXYZ = stressValuesXYZ;
    fprintf('-----------------------Stress Results
(XYZ)-----------------------\n')
    fprintf('\n')
    disp(stressValuesXYZ)
```

Creating Tabular Output for Stresses and Strains in 123

```
    % Strain Table
    LayerEps1 = zeros(2*N,1);
    LayerEps2 = zeros(2*N,1);
    LayerGam12 = zeros(2*N,1);

    for plyIdx = 1:N
        LayerNum(2*plyIdx-1) = plyIdx; % Layer number for top interface
        LayerNum(2*plyIdx) = plyIdx;      % Layer number for bottom interface
        LayerZCoord(2*plyIdx-1) = zCoord(plyIdx); % Z-coordinate for top
interface
        LayerZCoord(2*plyIdx) = zCoord(plyIdx + 1); % Z-coordinate for
bottom interface
        LayerEps1(2*plyIdx-1) = epsLayer123(1,1,plyIdx); % Epsilon X for top
interface
        LayerEps1(2*plyIdx) = epsLayer123(1,2,plyIdx);      % Epsilon X for
bottom interface
        LayerEps2(2*plyIdx-1) = epsLayer123(2,1,plyIdx); % Epsilon Y for top
```

```
interface
        LayerEps2(2*plyIdx) = epsLayer123(2,2,plyIdx);     % Epsilon Y for
bottom interface
        LayerGam12(2*plyIdx-1) = epsLayer123(3,1,plyIdx); % Gamma XY for top
interface
        LayerGam12(2*plyIdx) = epsLayer123(3,2,plyIdx);     % Gamma XY for
bottom interface
    end

    strainValues123 = table(LayerNum, LayerZCoord, LayerEps1, LayerEps2,
LayerGam12,...
                         'VariableNames',{'Layer', 'z', 'ε1', 'ε2', 'γ12'});

    LaminateResults.StrainTable123 = strainValues123;
    fprintf('----------------------Strain Results (123)----------------------
\n')
    fprintf('\n')
    disp(strainValues123)

    % Stress Table
    LayerSigma1 = zeros(2*N,1);
    LayerSigma2 = zeros(2*N,1);
    LayerTau12 = zeros(2*N,1);
    for plyIdx = 1:N
        LayerSigma1(2*plyIdx-1) = sigmaLayer123(1,1,plyIdx); % Sigma X for
top interface
        LayerSigma1(2*plyIdx) = sigmaLayer123(1,2,plyIdx);     % Sigma X for
bottom interface
        LayerSigma2(2*plyIdx-1) = sigmaLayer123(2,1,plyIdx); % Sigma Y for
top interface
        LayerSigma2(2*plyIdx) = sigmaLayer123(2,2,plyIdx);     % Sigma Y for
bottom interface
        LayerTau12(2*plyIdx-1) = sigmaLayer123(3,1,plyIdx); % Tau XY for top
interface
        LayerTau12(2*plyIdx) = sigmaLayer123(3,2,plyIdx);     % Tau XY for
bottom interface
    end

    stressValues123 = table(LayerNum, LayerZCoord, LayerSigma1, LayerSigma2,
LayerTau12,...
                            'VariableNames',{'Layer', 'z', 'σ1', 'σ2',
'τ12'});
    LaminateResults.StressTable123 = stressValues123;
    fprintf('-----------------------Stress Results
(123)-----------------------\n')
    fprintf('\n')
    disp(stressValues123)

end
```

Published with MATLAB® R2025a