

Constant Strain Triangle, CST

```

function Plane_Stress_T3_XY
%.....
% Plane stress displacements for constant strain triangle, T3
%           XY COORDINATES CLOSED FORM INTEGRALS
%.....
%           Set given constants
n_g = 2           ; % number of DOF per node
n_n = 3           ; % number of nodes per element
n_i = n_n*n_g    ; % number of DOF per element

% MODEL input file controls (for various data generators)
pre_e = 0 ; % Dummy items before el_type & connectivity
pre_p = 0 ; % Dummy items before BC_flag % coordinates

%           READ MESH AND EBC INPUT DATA
% specific problem data from MODEL data files (sequential)
load msh_bc_xyz.tmp           ; % bc_flag, x_coord, y_coord
n_m = size (msh_bc_xyz,1) ; % number of nodal points in mesh
n_d = n_g*n_m                 ; % system degrees of freedom (DOF)
if ( n_m == 0 )               ; % data missing !
    error ('Error missing file msh_bc_xyz.tmp')
end % if error
fprintf ('Read %g mesh coordinate pairs \n', n_m)
P = msh_bc_xyz (1:n_m, (pre_p+1)) ; % extract integer Packed BC flag
x = msh_bc_xyz (1:n_m, (pre_p+2)) ; % extract x column
y = msh_bc_xyz (1:n_m, (pre_p+3)) ; % extract y column

load msh_typ_nodes.tmp       ; % el_type, connectivity list (3)
n_e = size (msh_typ_nodes,1) ; % number of elements
if ( n_e == 0 )              ; % data file missing
    error ('Error missing file msh_typ_nodes.tmp')
end % if error
nod_per_el = size (msh_typ_nodes,2) - pre_e - 1 ; % data checking
fprintf ('Read %g elements with %g nodes each \n', n_e, nod_per_el)
if ( nod_per_el ~= n_n )
    error ('Mesh requires 3 nodes per element')
end % if
el_type = msh_typ_nodes(:, pre_e+1) ; % element type number >= 1
n_t      = max(el_type)                ; % number of element types

%           BOOKKEEPING FOR BC FLAGS AND VALUES
% Allocate for BC flags and values
EBC_flag = zeros(n_m, n_g) ; NBC_flag = zeros(n_m, n_g) ;
EBC_value = zeros(n_m, n_g) ; NBC_value = zeros(n_m, n_g) ;

% Extract EBC flags from packed integer flag P (for n_g=2 only)
for j = 1:n_m           % loop over each node, unpack 2 digit flag
    Boolean = floor(P(j)/10) ; % extract first digit
    EBC_flag (j, 1) = Boolean ; % insert first flag
    EBC_flag (j, 2) = P (j) - Boolean * 10 ; % extract 2-nd digit
end % for node j in the mesh

% Reorder and count BC flags and values
EBC_flag = reshape (EBC_flag', n_d, 1) ; % change to vector
NBC_flag = reshape (NBC_flag', n_d, 1) ; % change to vector
EBC_value = reshape (EBC_value', n_d, 1) ; % change to vector
NBC_value = reshape (NBC_value', n_d, 1) ; % change to vector

```

Constant Strain Triangle, CST

```

% Optional storage allocation for reaction recovery
EBC_count = sum(EBC_flag) ; % number of EBC's
NBC_count = sum(NBC_flag) ; % number of NBC's
EBC_row = zeros(EBC_count, n_d) ; EBC_col = zeros(EBC_count, 1) ;
EBC_react = zeros(EBC_count, 1) ; % reaction allocation

% SET ELEMENT PROPERTIES and body force
% Manually set constant element properties
t = 5e-3 ; % default thickness
E = 15e9 ; % Elastic modulus
nu = 0.25 ; % Poisson's ratio
E_v = E/(1 - nu^2) ; % constant
E_e = zeros (3, 3) ; % constitutive matrix
E_e (1, 1) = E_v ; E_e (1, 2) = E_v * nu ; % non-zero term
E_e (2, 1) = E_v * nu ; E_e (2, 2) = E_v ; % non-zero term
E_e (3, 3) = E_v * (1 - nu) / 2 ; % non-zero term
has_body_force = 1 ; % body forces on
body_force (1:2) = [ 5e5, 0. ] ; % components

% GENERATE ELEMENT MATRICES AND ASSYMBLE INTO SYSTEM
% Assemble n_d by n_d square matrix terms
S_e = zeros (n_i, n_i) ; C_e = zeros (n_i, 1) ; % initialize values
S = zeros (n_d, n_d) ; C = zeros (n_d, 1) ; % initialize sums

for j = 1:n_e ; % loop over elements
    thick = t * el_type (j) ; % integer multiple
    e_nodes = msh_typ_nodes (j, (pre_e+2):(pre_e+4)); % connectivity

% define nodal coordinates, ccw: i, j, k
x_e = x(e_nodes) ; y_e = y(e_nodes) ; % coord at el nodes
x_i = x_e(1) ; x_j = x_e(2) ; x_k = x_e(3) ; % change notation
y_i = y_e(1) ; y_j = y_e(2) ; y_k = y_e(3) ; % change notation

% define centroid coordinates (quadrature point)
x_cg = (x_i + x_j + x_k)/3 ; y_cg = (y_i + y_j + y_k)/3 ;

% geometric parameters: H_i (x,y) = (a_i + b_i*x + c_i*y)/two_a
a_i = x_j * y_k - x_k * y_j ; b_i = y_j - y_k ; c_i = x_k - x_j ;
a_j = x_k * y_i - x_i * y_k ; b_j = y_k - y_i ; c_j = x_i - x_k ;
a_k = x_i * y_j - x_j * y_i ; b_k = y_i - y_j ; c_k = x_j - x_i ;

% calculate twice element area
two_a = abs(a_i + a_j + a_k) ; % = b_j*c_k - b_k*c_j also

% define 3 by 6 strain-displacement matrix, B_e
B_e (1, 1:6) = [ b_i, 0.0, b_j, 0.0, b_k, 0.0 ] / two_a ;
B_e (2, 1:6) = [ 0.0, c_i, 0.0, c_j, 0.0, c_k ] / two_a ;
B_e (3, 1:6) = [ c_i, b_i, c_j, b_j, c_k, b_k ] / two_a ;

% ELEMENT MATRICES
% stiffness matrix, with constant jacobian
S_e = ( B_e' * E_e * B_e ) * thick * two_a / 2 ; % stiffness

% body force per unit volume
if ( has_body_force ) % then check keyword values in array
    X_x = body_force (1) * thick * two_a / 6.0 ; % x resultant
    X_y = body_force (2) * thick * two_a / 6.0 ; % y resultant
    C_e (1:6) = [ X_x, X_y, X_x, X_y, X_x, X_y ] ; % vector result
end % if or set up properties for body force

```

Constant Strain Triangle, CST

```

%          SCATTER TO SYSTEM ARRAYS
% Insert completed element matrices into system matrices
rows (1:2:5) = n_g*(e_nodes(1:3) - 1) + 1 ; % element DOF numbers
rows (2:2:6) = n_g*(e_nodes(1:3) - 1) + 2 ; % element DOF numbers
S (rows, rows) = S (rows, rows) + S_e      ; % add to system sq
C (rows) = C (rows) + C_e                  ; % add to sys column
end % for each j element in mesh

%          INSERT EBC AND NBC (if any)
kount = 0                                ; % initialize counter
for j = 1:n_d                             % check all DOF for displacement BC or pt force
    if ( NBC_flag(j) )                    ; % then NBC here
        C (j) = C (j) + NBC_value (j)    ; % add force value
    end % if NBC for this DOF

%          ENFORCE ESSENTIAL BOUNDARY CONDITIONS
if ( EBC_flag(j) )                        % then EBC here
    if ( EBC_flag(j) == NBC_flag(j) )    % warning both here
        fprintf ('Usually an error: EBC and NBC at same DOF \n')
    end % if fatal BC data

%          Save reaction data to be destroyed by EBC solver trick
kount = kount + 1                        ; % copy counter
EBC_row(kount, 1:n_d) = S (j, 1:n_d)    ; % copy reaction data
EBC_col(kount)       = C (j)            ; % copy reaction data

%          Insert EBC identity to avoid matrix partition accounting
EBC = EBC_value(j)                       ; % recover EBC value
C (:) = C (:) - EBC * S (:, j)           ; % carry known column to RHS
S (:, j) = 0 ; S (j, :) = 0              ; % clear, restore symmetry
S (j, j) = 1 ; C (j) = EBC               ; % insert identity into row
end % if EBC for this DOF
end % for over all j-th DOF

%          DISPLACEMENT SOLUTION & SAVE
T = S \ C                                ; % Compute unknown displacements
fprintf ('\n') ;
fprintf('X_disp   Y_disp   at %g nodes \n', n_m)
disp (reshape(T, n_g, n_m)')
% save results (displacements) to MODEL file: node_results.tmp
fid = fopen('node_results.tmp', 'w')      ; % open for writing
for j = 1:n_g:n_d                          ; % save displacements
    fprintf ( fid, '%g %g \n', T (j), T (j+1) ) ; % n_g=2 only !
end % for j DOF

%          REACTION RECOVERY & SAVE
EBC_react = EBC_row * T - EBC_col        ; % Reaction from matrix (+-)
% save reactions (forces) to MODEL file: node_reaction.tmp
fprintf ('Node, Component, Value, Eq. for %g Reactions \n', EBC_count) ;
fid = fopen('node_reaction.tmp', 'w')    ; % open for writing
kount = 0                                ; % initialize counter
for j = 1:n_d                             ; % extract all EBC reactions
    if ( EBC_flag(j) )                    % then EBC here
%          Output node_number, component_number, value, equation_number
        kount = kount + 1                  ; % copy counter
        node = ceil(j/n_g)                 ; % node at DOF j
        j_g = j - (node - 1)*n_g          ; % 1 <= j_g <= n_g
        React = EBC_react (kount, 1)      ; % reaction value
    end
end

```

Constant Strain Triangle, CST

```

    fprintf ( fid, '%g %g %g %g \n', node, j_g, React, j) ;
    fprintf ('%g %g %g %g \n', node, j_g, React, j) ;
end % if EBC for this DOF
end % for over all j-th DOF

% STRESS RECOVERY & SAVE
fid = fopen('el_qp_xyz_fluxes.tmp', 'w') ; % open for writing
fprintf ('\n') ;
fprintf('Elem, X_qp, Y_qp, Sig_x, Sig_y, Sig_xy \n')
for j = 1:n_e ; % loop over elements
    thick = t * el_type (j) ; % integer multiple
    e_nodes = msh_typ_nodes (j, (pre_e+2):(pre_e+4)); % connectivity

% define nodal coordinates, ccw: i, j, k
x_e = x(e_nodes) ; y_e = y(e_nodes) ; % coord at el nodes
x_i = x_e(1) ; x_j = x_e(2) ; x_k = x_e(3) ; % change notation
y_i = y_e(1) ; y_j = y_e(2) ; y_k = y_e(3) ; % change notation

% geometric parameters: H_i (x,y) = (a_i + b_i*x + c_i*y)/two_a
a_i = x_j * y_k - x_k * y_j ; b_i = y_j - y_k ; c_i = x_k - x_j ;
a_j = x_k * y_i - x_i * y_k ; b_j = y_k - y_i ; c_j = x_i - x_k ;
a_k = x_i * y_j - x_j * y_i ; b_k = y_i - y_j ; c_k = x_j - x_i ;

% calculate twice element area
two_a = abs(a_i + a_j + a_k) ; % = b_j*c_k - b_k*c_j also

% RECOVER B MATRIX & ITS LOCATION & ELEMENT DOF
% define centroid coordinates (quadrature point)
x_cg = (x_i + x_j + x_k)/3 ; y_cg = (y_i + y_j + y_k)/3 ;

% define 3 by 6 strain-displacement matrix, B_e
B_e (1, 1:6) = [ b_i, 0.0, b_j, 0.0, b_k, 0.0 ] / two_a ;
B_e (2, 1:6) = [ 0.0, c_i, 0.0, c_j, 0.0, c_k ] / two_a ;
B_e (3, 1:6) = [ c_i, b_i, c_j, b_j, c_k, b_k ] / two_a ;

% get DOF numbers for this element, gather solution
rows (1:2:5) = n_g*(e_nodes(1:3) - 1) + 1 ; % element DOF numbers
rows (2:2:6) = n_g*(e_nodes(1:3) - 1) + 2 ; % element DOF numbers
T_e = T (rows) ; % gather element DOF

% COMPUTE STRESSES, SAVE LOCATION AND VALUES
Strain = B_e * T_e ; % matrix product
Stress = E_e * Strain ; % matrix product
fprintf (fid, '%g %g %g %g \n', x_cg, y_cg, Stress(1:3)) ;
fprintf ('%g %g %g %g %g \n', j, x_cg, y_cg, Stress(1:3)) ;
end % for each j element in mesh

% End finite element calculations.
% See /home/mech517/public_html/Matlab_Plots for graphic options
% http://www.owl.net.rice.edu/~mech517/help_plot.html for help

% end of Plane_Stress_T3_XY

```