

Isoparametric Constant Strain Triangle, CST

```

function Plane_Stress_T3_RS
%.....
% Plane stress displacements for constant strain triangle, T3
%
% ISOPARAMETRIC VERSION
%.....
%           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

```

Isoparametric 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

% Tabulate quadrature data for unit triangle
n_q = 1 ; % exact for linear polynomials
r_q (1) = 1./3 ; s_q (1) = 1./3 ; w_q (1)= 1./2 ; % tabulated

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

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

% Define nodal coordinates
xy_e (1:n_n, 1) = x(e_nodes) ; % x coord at el nodes
xy_e (1:n_n, 2) = y(e_nodes) ; % y coord at el nodes

% Numerical Intergation of S_e, C_e
S_e = zeros (n_i, n_i) ; C_e = zeros (n_i, 1) ; % initialize values
E_q = zeros (n_i, n_i) ; B_q = zeros (3, n_i) ; % initialize values

for q = 1:n_q ; % begin loop --->
    r = r_q (q) ; s = s_q (q) ; w = w_q (q) ; % recover data
    H_q = [(1-r-s), r, s] ; % interpolations
    b_q = [-1, 1, 0 ; ... % dH/dr
           -1, 0, 1 ] ; % dH/ds

% Global position and Jacobian
xy_q = H_q * xy_e ; % 1 x 2 % global (x, y)
Jacobian = b_q * xy_e ; % 2 x 2 % d(x,y)/d(r,s)
J_det = abs (det (Jacobian)) ; % 1 x 1 % determinant
J_Inv = inv (Jacobian) ; % 2 x 2 % inverse

% global derivatives of H
d_q = J_Inv * b_q ; % 2 x 3 % dH/dx & dH/dy
E_q = E_e ; % 3 x 3 % properties
t_q = thick ; % 1 x 1 % thickness

```

Isoparametric Constant Strain Triangle, CST

```

%      define 3 by 6 strain-displacement matrix, B_q
B_q(1,1:6)=[d_q(1,1), 0.0      , d_q(1,2), 0.0      , d_q(1,3), 0.0      ];
B_q(2,1:6)=[0.0      , d_q(2,1), 0.0      , d_q(2,2), 0.0,      d_q(2,3)];
B_q(3,1:6)=[d_q(2,1), d_q(1,1), d_q(2,2), d_q(1,2), d_q(2,3), d_q(1,3)];

%      ELEMENT MATRICES UPDATES
S_e = S_e + (B_q' * E_q * B_q) * t_q * J_det * w_q (q) ; % stiff

%      body force resultants
point_x = H_q * body_force(1)*t_q * J_det * w_q(q); % x resultant
point_y = H_q * body_force(2)*t_q * J_det * w_q(q); % y resultant
C_e(1:2:5) = C_e(1:2:5) + point_x'      ; % odd rows x
C_e(2:2:6) = C_e(2:2:6) + point_y'      ; % even rows y
end % for q-th quadrature point (el matrices complete) <-----

%      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      ; % DOF loop      =====>
    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

```

Isoparametric Constant Strain Triangle, CST

```

% 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
        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 ; % element loop ==>>
    thick = t * el_type (j) ; % integer multiple
    e_nodes = msh_typ_nodes (j, (pre_e+2):(pre_e+4)); % connectivity

% Define nodal coordinates
xy_e (1:n_n, 1) = x(e_nodes) ; % x coord at el nodes
xy_e (1:n_n, 2) = y(e_nodes) ; % y coord at el nodes

for q = 1:n_q ; % begin loop --->
    r = r_q (q) ; s = s_q (q) ; w = w_q (q) ; % recover data
    H_q = [(1-r-s), r, s] ; % interpolations
    b_q = [-1, 1, 0 ; ... % dH/dr
           -1, 0, 1 ] ; % dH/ds

% Global position and Jacobian
xy_q = H_q * xy_e ; % 1 x 2 % global (x, y)
Jacobian = b_q * xy_e ; % 2 x 2 % d(x,y)/d(r,s)
J_Inv = inv (Jacobian) ; % 2 x 2 % inverse

% global derivatives of H
d_q = J_Inv * b_q ; % 2 x 3 % dH/dx & dH/dy
E_q = E_e ; % 3 x 3 % properties

% define 3 by 6 strain-displacement matrix, B_q
B_q(1,1:6)=[d_q(1,1), 0.0 , d_q(1,2), 0.0 , d_q(1,3), 0.0 ] ;
B_q(2,1:6)=[0.0 , d_q(2,1), 0.0 , d_q(2,2), 0.0, d_q(2,3)];
B_q(3,1:6)=[d_q(2,1), d_q(1,1), d_q(2,2), d_q(1,2), d_q(2,3), d_q(1,3)];

% 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) ; % 6 x 1 % gather element DOF

```

Isoparametric Constant Strain Triangle, CST

```
%      COMPUTE STRESSES, SAVE LOCATION AND VALUES
Strain = B_q * T_e      ; % 3 x 1      % matrix product
Stress = E_q * Strain   ; % 3 x 1      % matrix product
fprintf (fid, '%g %g %g %g %g \n', xy_q(1:2), Stress(1:3)) ;
fprintf ('%g %g %g %g %g %g \n', j, xy_q(1:2), Stress(1:3)) ;
end % numerical integration loop over q points <----
end % loop 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 Plane_Stress_T3_RS
```