```
function [ElemK, ElemF,fint,ElemM] = Elast2d_Elem(xl,mateprop,nel,ndf,stress,Fbody
%
% Copyright (C) Arif Masud and Tim Truster
%
% Subroutine to compute stiffness matrix and force vector for linear
% 2-dimensional elasticity element. Element currently supports bilinear
% quadrilateral elements with the following node and shape function
% labelling scheme:
%
%  (-1, 1)  4 -------------- 3 ( 1, 1)
%           |       s        |
%           |       ^        |
%           |       |        |
%           |       .-> r    |
%           |                |
%           |                |
%  (-1,-1)  1 -------------- 2 ( 1,-1)
%
% Element local coordinates (r,s) are defined by a coordinate axis with the
% origin at the center of the element; the corners of the element have
% local coordinate values as shown in the figure.
%
% Definitions for input:
%
%   xl:             = local array containing (x,y) coordinates of nodes
%                     forming the element; format is as follows:
%                         Nodes    |        n1  n2  n3  n4
%                         x-coord  |  xl = [x1  x2  x3  x4
%                         y-coord  |        y1  y2  y3  y4];
%
%   mateprop:       = vector of material properties:
%                         mateprop = [E v t];
%                                  = [(Young's Modulus) (Poisson's Ratio)
%                                     (thickness)];
%
%   nel:            = number of nodes on current element (4)
%
%   ndf:            = max number of DOF per node (2)
%
%   ndm:            = space dimension of mesh (2)
%
%   PSPS:           = flag for plane stress ('s') or plane strain ('n')
%
% Definitions for output:
%
%   ElemK:          = element stiffness matrix containing stiffness
%                     entries in the following arrangement, where
%                     wij corresponds to weighting function (i), coordinate
%                     direction (j), and ukl corresponds to displacement
%                     function (k), coordinate direction (l):
%                                u1x  u1y  u2x  u2y  u3x  u3y  u4x  u4y
%                     w1x  ElemK[ .    .    .    .    .    .    .    .
```

```
%                       w1y         .    .    .    .    .    .    .    .
%                       w2x         .    .    .    .    .    .    .    .
%                       w2y         .    .    .    .    .    .    .    .
%                       w3x         .    .    .    .    .    .    .    .
%                       w3y         .    .    .    .    .    .    .    .
%                       w4x         .    .    .    .    .    .    .    .
%                       w4y         .    .    .    .    .    .    .    . ];
%
%   ElemF:            = element force vector containing force entries in the
%                       following arrangement:
%                       w1x  ElemF[ .
%                       w1y         .
%                       w2x         .
%                       w2y         .
%                       w3x         .
%                       w3y         .
%                       w4x         .
%                       w4y         . ];
%
% Definitions of local constants:
%
%   nst:              = size of element arrays (ndf*nel)
%
%


ul_elem = reshape(ul,ndf*nel,1);
ul_elem2 = [ul_elem(1:2:length(ul_elem)) ul_elem(2:2:length(ul_elem))];

% Set Material Properties for the nonlinear material
mu1= mateprop(1);
mu2 = mateprop(2);
kappa = mateprop(3);
thick = mateprop(4);

% Initialize Matrix and Vector

nst = nel*ndf;
ElemK = zeros(nst);
ElemF = zeros(nst,1);
fint = zeros(nst,1);
ElemM = zeros(nst);


% Body Force if required:
Fb(1,1) = Fbody(1);
Fb(2,1) = Fbody(2);
% Load Guass Integration Points

if nel == 3
    lint = 4;
else
    lint = 4;
end
```

```
% Loop over integration points

for l = 1:lint

    if nel == 3
        [Wgt,r,s] =  intpntt(l,lint,0);
    else
        [Wgt,r,s] =  intpntq(l,lint,0);
    end

    % Evaluate local basis functions at integration point
    shp = shpl_2d(r,s,nel);

    shpm = [shp(3,1) 0        shp(3,2) 0        shp(3,3) 0         shp(3,4) 0
        0        shp(3,1) 0        shp(3,2) 0         shp(3,3) 0        shp(3,4)];

    % Evaluate first derivatives of basis functions at int. point
    [Qxy, Jdet] = shpg_2d(shp,xl,nel);

    % Form B matrix
    if nel == 3
        Bmat = [Qxy(1,1) 0        Qxy(1,2) 0        Qxy(1,3) 0
            0        Qxy(2,1) 0        Qxy(2,2) 0        Qxy(2,3)
            Qxy(2,1) Qxy(1,1) Qxy(2,2) Qxy(1,2) Qxy(2,3) Qxy(1,3)];
    else
        Bmat = [Qxy(1,1) 0        Qxy(1,2) 0        Qxy(1,3) 0        Qxy(1,4) 0
            0        Qxy(2,1) 0        Qxy(2,2) 0        Qxy(2,3) 0        Qxy(2,4
            Qxy(2,1) Qxy(1,1) Qxy(2,2) Qxy(1,2) Qxy(2,3) Qxy(1,3) Qxy(2,4) Qxy(1,4
    end


    %%%%%%%%%%%%%%%% MODIFICATION %%%%%%%%%%%%%%%%%%%%%%%%

    Bmat2=[Qxy(1,1) Qxy(1,2) Qxy(1,3) Qxy(1,4);
        Qxy(2,1) Qxy(2,2) Qxy(2,3) Qxy(2,4)];

    gradU = Bmat2*ul_elem2;

    F=transpose(gradU)+eye(length(gradU));
    J=det(F);
    N=[0;1];

    FN=F*N;
    Identity=eye(2);

    for xi=1:length(F)
        for xj=1:length(F)
            for xk=1:length(F)
                for ll=1:length(F)
                    c(xi,xj,xk,ll)=(-mu1+(kappa+mu1)*(2*J-1))*Identity(xi,xj)*Iden
                        (mu1-(kappa+mu1)*(J-1))*(Identity(xi,ll)*Identity(xj,xk)+I
                end
            end
```

```
        end
    end

    rho = 1 + (shp(3,3)+shp(3,4))/50;


    Dmat=[c(1,1,1,1) c(1,1,2,2) c(1,1,1,2);
        c(2,2,1,1) c(2,2,2,2) c(2,2,1,2);
        c(1,2,1,1) c(1,2,2,2) c(1,2,1,2)];

    sigma=[stress(l,1) stress(l,3);
        stress(l,3) stress(l,2)];

    GP=Bmat2'*sigma*Bmat2        ;                                        %geome

    GP_mat=[eye(2)*GP(1,1) eye(2)*GP(1,2) eye(2)*GP(1,3) eye(2)*GP(1,4);    %terms
        eye(2)*GP(2,1) eye(2)*GP(2,2) eye(2)*GP(2,3) eye(2)*GP(2,4);
        eye(2)*GP(3,1) eye(2)*GP(3,2) eye(2)*GP(3,3) eye(2)*GP(3,4);
        eye(2)*GP(4,1) eye(2)*GP(4,2) eye(2)*GP(4,3) eye(2)*GP(4,4)];

    % Update integration weighting factor

    W = Wgt*Jdet*thick;

    % Initial Stress Term :
    ElemK = ElemK + W*Bmat'*Dmat*Bmat + W*GP_mat;

    ElemF = ElemF + W*shpm'*Fb;

    fint = fint+W*Bmat'*stress(l,:)' ; %Computing interal forces in the element


    ElemM = ElemM + rho*Wgt/4*(shpm')*(shpm);

end

end
```

Published with MATLAB® R2013a