```
% Linear Finite Element Program for 2-D Elasticity
%
% Copyright (C) Arif Masud and Tim Truster
%
% This program computes a numerical solution to a finite element model
% using input on the geometry and physical properties of a mesh, and on the
% boundary conditions and applied loads. The routine assembles element
% quantities into the stiffness matrix and force vector to create a system
% of equations which is then solved for the nodal values of the
% displacement field. Boundary conditions are applied to constrain the
% stiffness matrix and to augment the force vector. The output is a list of
% the displacements printed on screen, contour plots of the displacement
% fields, and a plot of the deformed configuration of the mesh.
%
% Mesh input should be uploaded by running an input .m file before
% executing this program.
%
% Format of required input:
%
%   numnp:           = number of nodes in the mesh (length(NodeTable))
%
%   numel:           = number of elements in the mesh
%
%   nen:             = maximum number of nodes per element (4)
%
%   PSPS:            = flag for plane stress ('s') or plane strain ('n')
%
%   NodeTable:       = table of mesh nodal coordinates defining the
%                      geometry of the mesh; format of the table is as
%                      follows:
%                          Nodes  |             x-coord  y-coord
%                          n1     |  NodeTable = [x1     y1
%                          n2     |               x2     y2
%                          ...    |               ..     ..
%                          nnumnp |               xnumnp ynumnp];
%
%   ix:              = table of mesh connectivity information, specifying
%                      how nodes are attached to elements and how materials
%                      are assigned to elements; entries in the first nen
%                      columns correspond to the rows of NodeTable
%                      representing the nodes attached to element e;
%                      entries in the last nen+1 column are rows from MateT
%                      signifying the material properties assigned to
%                      element e; format of the table is as follows:
%                          Elements  |         n1    n2    n3    n4   mat
%                          e1        |  ix = [e1n1  e1n2  e1n3  e1n4 e1mat
%                          e2        |        e2n1  e2n2  e2n3  e2n4 e2mat
%                          ...       |         ..    ..    ..    ..   ..
%                          enumel    |        values for element numel   ];
%
%   MateT:           = table of mesh material properties for each distinct
%                      set of material properties; these sets are
```

```
%                       referenced by element e by setting the value of
%                       ix(e,nen+1) to the row number of the desired
%                       material set; format of the table is as follows:
%                           Materials  |           E   v   t
%                           mat1       |  MateT = [E1  v1  t1
%                           mat2       |           E2  v2  t2
%                           ...        |           ..  ..  ..];
%
%   BCLIndex:         = list of the number of boundary conditions and loads
%                       applied to the mesh; first entry is the number of
%                       prescribed displacements at nodes; second entry is
%                       the number of nodal forces
%
%   NodeBC:           = table of prescribed nodal displacement boundary
%                       conditions; it contains lists of nodes, the
%                       direction of the displacement prescribed (x=1, y=2),
%                       and the value of the displacement (set 0 for fixed
%                       boundary); the length of the table must match the
%                       entry in BCLIndex(1), otherwise an error will result
%                       if too few conditions are given or extra BCs will be
%                       ignored in the model input module;  format of the
%                       table is as follows:
%                           BCs  |            nodes direction value
%                           bc1  |  NodeBC = [bc1n   bc1dir   bc1u
%                           bc2  |            bc2n   bc2dir   bc2u
%                           ...  |             ..     ..       .. ];
%
%   NodeLoad:         = table of prescribed nodal forces; it contains lists
%                       of nodes, the direction of the force prescribed
%                       (x=1, y=2), and the value of the force; the length
%                       of the table must match the entry in BCLIndex(2),
%                       otherwise an error will result if too few conditions
%                       are given or extra loads will be ignored in the
%                       model input module; format of the table is as
%                       follows:
%                           Loads  |              nodes direction value
%                           P1     |  NodeLoad = [ P1n    P1dir    P1P
%                           P2     |               P2n    P2dir    P2P
%                           ...    |               ..     ..       .. ];
%
% The following numbering convention is used for 4-node quadrilateral
% elements:
%
%           4 -------------- 3       2
%           |                |       | \
%           |                |       |  \
%           |                |       |   \
%           |                |       |    \
%           |                |       |     \
%           |                |       |      \
%           1 -------------- 2       3-------1
%

format compact
```

```
% Updating the NodeTable to match initial displacement :
NodeTable = [0        0
             L        0
             L+dis(1,storej) H+dis(2,storej)
             dis(3,storej) H+dis(4,storej)];

% Calling FormFE to make one-pass through the Algorithm :
FormFE
%

% Storing the Computed Acceleration iterates :
acc(:,storej)  = -(1+alpha)*Mdd\F_bar_int;
an(:,1) = acc(:,1);

% Optional Parameter to include any tractions :
Fext = 0*linspace(0,1,tSteps);

% dn3x(:,1) = 0;
% dn3y(:,1) = do;
% dn4x(:,1) = 0;
% dn4y(:,1) = do;


%%%%%%%%%%%%%%%%%%                                          %%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%    Major Modification in the Code        %%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%                                          %%%%%%%%%%%%%%%%%%%%%

for n=1:tSteps-1

    if n==1
        FEXT(:,n+1) = Fd;
    else
        FEXT(:,n+1) = Fdtilda;
    end

    IntF_store(:,n) = F_bar_int;
    res = 1;Res_0 = res;
    iter = 1;

    % Initializing the Predictors for the Newton Step (iteration in Multi-Correcto
    dis(:,storej) = dn(:,n) + delt*vn(:,n) + delt^2/2*(1-2*beta)*an(:,n);
    vel(:,storej) = vn(:,n) + delt*(1-gamma)*an(:,n);
    acc(:,storej) = an(:,n);

    while (res>tol*Res_0) && iter < maxiter

        % Form and Solve Matrix System for current iterate :
        FormFE

        % Calling SolveFE for the solution to the linearized system :
        SolveFE

        % Store the Initial Residual (Res_0) :
```

```
        if iter==1
            Res_0=res;
        end

        % Update the Node Table to include the displacement iterate :
        NodeTable = [0        0
                     L        0
                     L+dis(1,storej+1) H+dis(2,storej+1)
                     dis(3,storej+1) H+dis(4,storej+1)];

        residual(n,iter) = res;
        iter=iter+1;                          % Update the iteration counter
        storej=storej+1;
    end
    n=n+1;
%     if norm(Mstar) > 1e7
%         break
%     end
    %     Updating the last converged iterate :
    dn(:,n) = dis(:,storej);    vn(:,n) = vel(:,storej);    an(:,n) = acc(:,storej

    % Store the Displacement from the last converged iterate (plotting) :
    %     Disp(:,n) = dn;
    %     Vel(:,n) = vn;
    %     Acc(:,n) = an;

    Stress(n,:)=stress(4,:);    % We define the Integration Point counterclockwise
    Strain(n,:)=strain(4,:);
end

% Output Results

Node_U_V = zeros(numnp,2*ndf);

for node = 1:numnp
    for dir = 1:ndf
        gDOF = NDOFT(node, dir);
        if gDOF <= neq
            Node_U_V(node, dir) = dn(gDOF,n);
        else
            Node_U_V(node, dir) = ModelDc(gDOF - neq);
        end
    end
end

Node_U_V;
maxuvw = zeros(ndm,1);
maxxyz = zeros(ndm,1);

for i = 1:ndm
    maxuvw(i) = max(abs(Node_U_V(:,i)));
    maxxyz(i) = max(NodeTable(:,i));
end
```

```
len = sqrt(maxxyz'*maxxyz);
perc = 5/100;
factor = len/max(maxuvw)*perc;
NodeTable2 = NodeTable;

for i = 1:ndm
    NodeTable2(:,i) = NodeTable2(:,i) + Node_U_V(:,i)*factor;
end

dn=dn';vn = vn';an=an';

% Producing the Output for Plotting :
Output
```

Published with MATLAB® R2013a