```
function [DEFL,REACT,ELE_FOR,AFLAG] = ud_3d1el(...

nnodes,coord,concen,fixity,nele,ends,A,Izz,Iyy,J,Cw,IsSym,Ysc,Zsc,Betay,Betaz,
Betaw,Zzz,Zyy,Ayy,Azz,...
    E,v,Fy,YldSurf,Wt,webdir,beta_ang,w,thermal,truss,anatype);

% Code developed by Mrunmayi Mungekar and Devasmit Dutta
%
% UD_3D1EL performs a user defined three-dimensional
% first-order elastic analysis of a structural system.
%

%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
%
%  Functions Called
%               < to be defined by the student >
%
%  Dictionary of Variables
%     Input Information:
%       nnodes         ==  total number of nodes
%       coord(i,1:3)   ==  node i's coordinates
%                            coord(i,1) = X coordinate
%                            coord(i,2) = Y coordinate
%                            coord(i,3) = Z coordinate
%       concen(i,1:6)  ==  concentrated loads for node i's 6 d.o.f.
%                            concen(i,1) = force in global X direction
%                            concen(i,2) = force in global Y direction
%                            concen(i,3) = force in global Z direction
%                            concen(i,4) = moment about global X axis
%                            concen(i,5) = moment about global Y axis
%                            concen(i,6) = moment about global Z axis
%       fixity(i,1:6)  ==  prescribed displacements for node i's 6 d.o.f.
%                          Note: A free d.o.f. will have a value of NaN
%                          and hence, you will find the Matlab function
%                          isnan very useful.
%                          Examples: If fixity(15,3) is set to NaN, then node
15's
%                                      Z-disp component is free;
%                                    If fixity(2,6) is set to 0.0, then node
2's
%                                      Z-rotation component is supported;
%                                    If fixity(5,2) is set to -2.1, then node
5's
%                                      Y-disp component is supported and
defined
%                                      with a settlement of -2.1 units.
%                            fixity(i,1) = prescribed disp. in global X
direction
%                            fixity(i,2) = prescribed disp. in global Y
direction
```

```
%                             fixity(i,3) = prescribed disp. in global Z
direction
%                             fixity(i,4) = prescribed rotation about global X
axis
%                             fixity(i,5) = prescribed rotation about global Y
axis
%                             fixity(i,6) = prescribed rotation about global Z
axis
%       nele            ==  total number of elements
%       ends(i,1:14)    ==  element i's nodal information
%                             ends(i,1) = start node #
%                             ends(i,2) = finish node #
%                             ends(i,3) = flag to indicate whether or not
flexural
%                             moments are released at start node.  ends(i,3)=0
both not
%                             released (rigid connection); ends(i,3)=1 both
flexural
%                             moments are released (pinned connection);
ends(i,3)=2
%                             at least one of the flexural moments are
partially or fully
%                             released (see below for connection stiffness
attributes)
%                             ends(i,4) = flag to indicate whether or not
flexural
%                             moments are released at finish node.
ends(i,4)=0 both not
%                             released (rigid connection); ends(i,4)=1 both
flexural
%                             moments are released (pinned connection);
ends(i,4)=2
%                             at least one of the flexural moments are
partially or fully
%                             released (see below for connection stiffness
attributes)
%                             ends(i,5) = flag to indicate the degree of
warping
%                             restraint at start node.  ends(i,5)=0 warping
free;
%                             ends(i,5)=1 warping fixed; ends(i,5)=2 warping
continuous
%                             ends(i,6) = flag to indicate the degree of
warping
%                             restraint at finish node.  ends(i,6)=0 warping
free;
%                             ends(i,6)=1 warping fixed; ends(i,6)=2 warping
continuous
%                             ends(i,7) = rotational spring stiffness at the
start
%                             node and about element i's local z-z axis.
%                             ends(i,8) = rotational spring stiffness at the
start
%                             node and about element i's local y-y axis.
```

```
%                             ends(i,9) = rotational spring stiffness at the
finish
%                             node and about element i's local z-z axis.
%                             ends(i,10) = rotational spring stiffness at the
finish
%                             node and about element i's local y-y axis.
%                             ends(i,11) = connection moment capacity Mpz at
the start
%                             node and about element i's local z-z axis.
%                             ends(i,12) = connection moment capacity Mpy at
the start
%                             node and about element i's local y-y axis.
%                             ends(i,13) = connection moment capacity Mpz at
the finish
%                             node and about element i's local z-z axis.
%                             ends(i,14) = connection moment capacity Mpy at
the finish
%                             node and about element i's local y-y axis.
%       A(i)           ==  element i's cross sectional area
%       Izz(i)         ==  element i's moment of inertia about its local z-z
axis
%       Iyy(i)         ==  element i's moment of inertia about its local y-y
axis
%       J(i)           ==  element i's torsional constant
%       Cw(i)          ==  element i's warping constant
%       Zzz(i)         ==  element i's plastic section modulus about its
local z-z axis
%       Zyy(i)         ==  element i's plastic section modulus about its
local y-y axis
%       Ayy(i)         ==  element i's effective shear area along its local y-
y axis
%       Azz(i)         ==  element i's effective shear area along its local z-
z axis
%       E(i)           ==  element i's material elastic modulus, Young's
Modulus
%       v(i)           ==  element i's material Poisson's ratio
%       Fy(i)          ==  element i's material yield strength
%       YldSurf(i)     ==  element i's yield surface maximum values
%                              YldSurf(i,1) = maximum P/Py value
%                              YldSurf(i,2) = maximum Mz/Mpz value
%                              YldSurf(i,3) = maximum My/Mpy value
%       Wt(i)          ==  element i's material weight density
%                          (Assume that gravity is directed in the negative
global Y dir)
%       webdir(i,1:3)  ==  element i's unit web vector.  This is a unit vector
%                          that defines the element's local y-y axis with
respect
%                          to the global coordinate system.  It is based on
the
%                          structure's undeformed geometry.
%                              webdir(i,1) = x component of element's unit
web vector
%                              webdir(i,2) = y component of element's unit
web vector
```

```
%                               webdir(i,3) = z component of element's unit
web vector
%                           NOTE: An element's 3x3 rotation matrix, [g], is
constructed
%                           as follows: First, calculate a unit vector,
x_vect, that
%                           describes the element's local x-axis. Second, take
the
%                           cross product of x_vect and webdir(i,:) to obtain
z_vect,
%                           i.e. z_vect = cross(x_vect,webdir(i,:)). Third,
set z_vect
%                           to a unit vector, i.e. z_vect = z_vect/
norm(z_vect).
%                           Finally, the first row of [g] is x_vect, its
second row is
%                           webdir(i,:), and its third row is z_vect.
%       beta_ang(i)    ==  element i's web rotation angle.  These values are
%                           provided for those students who are required to
calculate
%                           their own unit web vectors (see above).  It is
based
%                           on the structure's undeformed geometry.
%                           Note:  MASTAN2 uses the following convention for
%                                  defining a member's default web orientation:
%                                  A vector defing the element's local y-axis
%                                  with respect to the global coordinate system
%                                  will have a positive component in the global
%                                  Y direction.  If the element's local x-axis,
%                                  its length axis, is aligned with the global
Y
%                                  axis, then element's local y-axis is aligned
%                                  with global negative X axis.  After this
initial
%                                  orientation, element i may be rotated about
%                                  its local x-axis by the amount defined by
%                                  its web rotation angle, beta_ang(i).  The
%                                  angle is in radians and assumes a right-hand
%                                  convention about the local x-axis which
runs from
%                                  the element's start node to its finish node.
%       w(i,1:3)         ==  element i's uniform load which references its
%                             local coordinate system
%                               w(i,1) = x component of uniform load
%                               w(i,2) = y component of uniform load
%                               w(i,3) = z component of uniform load
%       thermal(i,1:4)   ==  element i's thermal strain effects which
reference its
%                             local coordinate system
%                               thermal(i,1) = coefficient of thermal expansion
%                               thermal(i,2) = change in temperature at
centroid
%                               thermal(i,3) = linear temperature gradient in
local y-dir
```

```
%                                          = (T_up_y - T_btm_y) / depth_y
%                             thermal(i,4) = linear temperature gradient in
local z-dir
%                                          = (T_up_z - T_btm_z) / width_z
%       truss            ==  flag to indicate if structure is a truss or not
%                             truss = 0   System is not a truss
%                             truss = 1   System is a truss
%       anatype          ==  flag to indicate which type of analysis is
requested
%                             anatype = 1  First-Order Elastic
%                             anatype = 2  Second-Order Elastic
%                             anatype = 3  First-Order Inelastic
%                             anatype = 4  Second-Order Inelastic
%                             anatype = 5  Elastic Buckling (Eigenvalue)
%                             anatype = 6  Inelastic Buckling (Eigenvalue)
%
%     Local Information:
%              < to be defined by the student >
%
%     Output Information:
%       DEFL(i,1:6)      ==  node i's calculated 6 d.o.f. deflections
%                             DEFL(i,1) = displacement in X direction
%                             DEFL(i,2) = displacement in Y direction
%                             DEFL(i,3) = displacement in Z direction
%                             DEFL(i,4) = rotation about X direction
%                             DEFL(i,5) = rotation about Y direction
%                             DEFL(i,6) = rotation about Z direction
%       REACT(i,1:6)     ==  reactions for supported node i's 6 d.o.f.
%                             REACT(i,1) = force in X direction
%                             REACT(i,2) = force in Y direction
%                             REACT(i,3) = force in Z direction
%                             REACT(i,4) = moment about X direction
%                             REACT(i,5) = moment about Y direction
%                             REACT(i,6) = moment about Z direction
%       ELE_FOR(i,1:1?)  ==  element i's internal forces and moments
%                            Note: All values reference the element's local
%                                  coordinate system.
%                             ELE_FOR(i,1)  = x-force at start node
%                             ELE_FOR(i,2)  = y-force at start node
%                             ELE_FOR(i,3)  = z-force at start node
%                             ELE_FOR(i,4)  = x-moment at start node
%                             ELE_FOR(i,5)  = y-moment at start node
%                             ELE_FOR(i,6)  = z-moment at start node
%                             ELE_FOR(i,7)  = x-force at end node
%                             ELE_FOR(i,8)  = y-force at end node
%                             ELE_FOR(i,9)  = z-force at end node
%                             ELE_FOR(i,10) = x-moment at end node
%                             ELE_FOR(i,11) = y-moment at end node
%                             ELE_FOR(i,12) = z-moment at end node
%                            If you are not programming warping torsion, the
ELE_FOR
%                            array needs to contain only 12 columns, i.e.
ELE_FOR(i,1:12)
%                            For those programming warping torsion, the
```

```
bimoments and
%                             rates of twist should be stored as follows.
%                               ELE_FOR(i,13) = bimoment at start node
%                               ELE_FOR(i,14) = bimoment at end node
%                               ELE_FOR(i,15) = rate of twist at start node
%                               ELE_FOR(i,16) = rate of twist at end node
%       AFLAG             ==  logical flag to indicate if a successful
%                             analysis has been completed
%                               AFLAG = 1     Successful
%                               AFLAG = 0     Unstable Structure
%                               AFLAG = inf   No analysis code available
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%
%
%  Start by defining all output arrays to be empty
%
DEFL=[]; REACT=[]; ELE_FOR=[];
%
% Display each input of the function on a new line
disp('Number of nodes');
disp(nnodes);

disp('Coordinates of nodes');
disp(coord);

disp('Concentrated loads');
disp(concen);

disp('Prescribed displacements');
disp(fixity);

disp('Number of elements');
disp(nele);

disp('Element nodal information');
disp(ends);

disp('Cross sectional area');
disp(A);

disp('Moment of inertia about local z axis');
disp(Izz);

disp('Moment of inertia about local y axis');
disp(Iyy);

disp('Torsional constant');
disp(J);

disp('Warping constant');
disp(Cw);

disp('Plastic section modulus about local z axis');
```

```
disp(Zzz);

disp('Plastic section modulus about local y axis');
disp(Zyy);

disp('Effective shear area along local y axis');
disp(Ayy);

disp('Effective shear area along local z axis');
disp(Azz);

disp('Elastic modulus');
disp(E);

disp('Poisson''s ratio');
disp(v);

disp('Yield strength');
disp(Fy);

disp('Yield surface maximum values');
disp(YldSurf);

disp('Weight density');
disp(Wt);

disp('Unit web vector');
disp(webdir);

disp('Web rotation angle');
disp(beta_ang);

disp('Uniform loads');
disp(w);

disp('Thermal strain effects');
disp(thermal);

disp('Truss flag');
disp(truss);

disp('Analysis type');
disp(anatype);

AFLAG = inf;
%
%  STUDENT NOTE:
%     In order for this routine to become fully active AFLAG
%     must be changed.
%
%
%  Student's code starts here...
%
%
```

```
%
%  Good luck CE Student!!!
%
```

Number of nodes

```
Not enough input arguments.

Error in ud_3d1el (line 217)
disp(nnodes);
```

Published with MATLAB® R2023b