```
function out=beam3kustom(mode,b,c,d,e)

% BEAM3 does as listed below. It is an Euler-Bernoulli
% beam/rod/torsion model.
% Beam properties (bprops) are in the order
% bprops=[E G rho A1 A2 A3 J1 J2 J3 Ixx1 Ixx2 Ixx3 Iyy1 Iyy2 Iyy3]
% Third node is in the middle.
% Fourth "node" defines the beam y plane and is actually from the
% points array.
%%
% Defining beam element properties in wfem input file:
% element properties
%   E G rho A1 A2 A3 J1 J2 J3 Izz1 Izz2 Izz3 Iyy1 Iyy2 Iyy3
% Torsional rigidity, $J$, must be less than or equal
% to $Iyy+Izz$ at any given cross section.
%
% Defining beam3 element in wfem input file:
%   node1 node2 node3 pointnumber materialnumber

%
% See wfem.m for more explanation.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Variables (global):
% -------------------
% K       :    Global stiffness matrix
% Ks      :    Global stiffness buckling matrix
% M       :    Global mass matrix
% nodes   :    [x y z] nodal locations
global ismatnewer
global K
global Ks
global M
global nodes % Node locations
global elprops
global element
global points
global Fepsn % Initial strain "forces".
global lines
global restart
global reload
global curlineno
global DoverL
global surfs
%
% Variables (local):
% ------------------
% bnodes  :    node/point numbers for actual beam nodes 1-2-3 and point
% k       :    stiffness matrix in local coordiates
% kg      :    stiffness matrix rotated into global coordinates
% m       :    mass matrix in local coordiates
% mg      :    mass matrix rotated into global coordinates
%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
% Copyright Joseph C. Slater, 7/26/2002.
% joseph.slater@wright.edu
out=0;
if strcmp(mode,'numofnodes')
    % This allows a code to find out how many nodes this element has
    out=2;
end
if strcmp(mode,'generate')
  elnum=c;%When this mode is called, the element number is the 3rd
          %argument.

          %The second argument (b) is the element
          %definition. For this element b is
          %node1 node2 node3 point(for rotation) and material#

          %There have to be 5 elements for this element's
          %definition (above)
  if length(b)==4
      element(elnum).nodes=b(1:2);
      element(elnum).properties=b(4);
      element(elnum).point=b(3);
  else
       b
      %There have to be five numbers on a line defining the
      %element.
      warndlg(['Element ' num2str(elnum) ' on line ' ...
               num2str(element(elnum).lineno) ' entered incorrectly.'],
...
              ['Malformed Element'],'modal')
      return
  end

end

% Here we figure out what the beam properties mean. If you need
% them in a mode, that mode should be in the if on the next line.
if strcmp(mode,'make')||strcmp(mode,'istrainforces')
  elnum=b;% When this mode is called, the element number is given
          % as the second input.
  bnodes=[element(elnum).nodes element(elnum).point];% The point is
                                                     % referred to
                                                     % as node 4
                                                     % below,
                                                     % although it
                                                     % actually
                                                     % calls the
                                                     % array points
                                                     % to get its
                                                     % location. Its
                                                     % not really a
                                                     % node, but
                                                     % just a point
```

```
                                                           % that helps
                                                           % define
                                                           % orientation. Your
                                                           % element may
                                                           % not need
                                                           % such a
                                                           % reference point.
  bprops=elprops(element(elnum).properties).a;% element(elnum).properties
                                              % stores the
                                              % properties number
                                              % of the current
                                              % elnum. elprops
                                              % contains this
                                              % data. This is
                                              % precisely the
                                              % material properties
                                              % line in an
                                              % array. You can pull
                                              % out any value you
                                              % need for your use.

%
  if length(bprops)==11
      E=bprops(1);
      G=bprops(2);
      rho=bprops(3);
      A1=bprops(4);
      A2=bprops(5);
      J1=bprops(6);
      J2=bprops(7);
      Izz1=bprops(8);
      Izz2=bprops(9);
      Iyy1=bprops(10);
      Iyy2=bprops(11);
  else
      warndlg(['The number of material properties set for ' ...
               'this element (' num2str(length(bprops)) ') isn''t ' ...
               'appropriate for a beam3 element. '      ...
               'Please refer to the manual.'],...
              'Bad element property definition.','modal');
  end
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Beam properties (bprops) are in the order
% bprops=[E G rho A1 A2 A3 J1 J2 J3 Izz1 Izz2 Izz3 Iyy1 Iyy2 Iyy3]
% For a linear beam they are
% bprops=[E G rho A1 A2 J1 J2 Izz1 Izz2 Iyy1 Iyy2]

if strcmp(mode,'make')

  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  %
```

```
% Define beam node locations for easy later referencing
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

x1=nodes(bnodes(1),1);
y1=nodes(bnodes(1),2);
z1=nodes(bnodes(1),3);
x2=nodes(bnodes(2),1);
y2=nodes(bnodes(2),2);
z2=nodes(bnodes(2),3);
x3=points(bnodes(3),1);
y3=points(bnodes(3),2);
z3=points(bnodes(3),3);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Shape functions for higher order beam.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Shape functions in matrix polynomial form (polyval style) for bending
bn1 =    [0.25    0    -0.75    0.5];
bn1d =   [0.75    0    -0.75];
bn1dd =  [1.5     0];
bn2 =    [0.25   -0.25 -0.25    0.25];
bn2d =   [0.75   -0.5  -0.25];
bn2dd =  [1.5    -0.5];
bn3 =    [-0.25   0     0.75    0.5];
bn3d =   [-0.75   0     0.75];
bn3dd =  [-1.5    0];
bn4 =    [0.25    0.25 -0.25   -0.25];
bn4d =   [0.75    0.5  -0.25];
bn4dd =  [1.5     0.5];


% Shape functions in matrix polynomial form (polyval style) for
% torsion/rod
rn1=  [-0.5 0.5];
rn1d= [-0.5];
rn2=  [.5 .5];
rn2d= [0.5];
numbeamgauss=5; % Number of Gauss points for integration of beam
element
[bgpts,bgpw]=gauss(numbeamgauss);
kb1=zeros(4,4);% For this beam, 2 nodes, 2DOF each, is a 4 by 4
               % matrix.
kb2=kb1; %Stiffness matrix for the x-z plane beam element.
l=norm([x2 y2 z2]-[x1 y1 z1]);
propertynum=num2str(element(elnum).properties);
% Allowable aspect ratio. I recommend D/l=.1
if isempty(DoverL)==1
  DoverL=.1;
end
%Euler bernoulli beams must be slender. Warn if not.
```

```
  if sqrt(A1*4/pi)/l>DoverL|sqrt(A2*4/pi)/l>DoverL
    warndlg({['Dimensions of element ' num2str(elnum) ' using properties
'...
          propertynum ' are more suitable for a Timoshenko beam.'];...
         'radius divided by length is too large'},...
        'Improper application of element.','replace')
  end
  % This took some work, but provide bounds on other values.
  if (Izz1+Iyy1)<(1/2.1*A1^2/pi)|(Izz2+Iyy2)<(1/2.1*A2^2/pi)
    %2.0 would be exact for a circle
    warndlg({['Iyy+Izz for properties number' propertynum ' can''t be as
'...
          'low as have been given.'];...
         'Nonphysical properties.'},['Impossible cross sectional' ...
             ' properties'],'replace')
  end
  slenderness=min([sqrt((Izz1+Iyy1)/A1) sqrt((Izz2+Iyy2)/A2)])/l;
  % Check if this is a beam or something so thin that its really a
  % string.
  if slenderness<.002
    disp([num2str(elnum) ['is a rediculously thin element. Please' ...
             ' check numbers.']])
  end

  Jac=l/2;% Beam Jacobian. valid only if node three is in the
        % middle of the beam. Luck for us, it always is (or the
        % code yells at you)
        % Local Bending in x-y plane
  for i=1:numbeamgauss
    beamsfs=[polyval(bn1dd,bgpts(i))/Jac^2;%evaluating second
                                           %derivatives of shape
                                           %functions to use in
                                           %generating stiffness
                                           %matrix. (at gauss point)
           polyval(bn2dd,bgpts(i))/Jac;
           polyval(bn3dd,bgpts(i))/Jac^2;
           polyval(bn4dd,bgpts(i))/Jac];
    Izz=polyval(rn1*Izz1+rn2*Izz2,bgpts(i));%Find Izz at
                                                     %Gauss point
    kb1=kb1+bgpw(i)*beamsfs*beamsfs'*Izz*E*Jac;%This is the Gauss
                                               %integration part.
  end
  % Local Bending in x-z plane
  for i=1:numbeamgauss
    beamsfs=[polyval(bn1dd,bgpts(i))/Jac^2;
           -polyval(bn2dd,bgpts(i))/Jac;
           polyval(bn3dd,bgpts(i))/Jac^2;
           -polyval(bn4dd,bgpts(i))/Jac];
    Iyy=polyval(rn1*Iyy1+rn2*Iyy2,bgpts(i));
    kb2=kb2+bgpw(i)*beamsfs*beamsfs'*Iyy*E*Jac;
  end

  % Local Extension in x, torsion about x
  numrodgauss=3;% Number of points to use for gauss point integration
```

```
  [rgpts,rgpw]=gauss(numrodgauss);
  krod=zeros(2,2);
  ktor=zeros(2,2);
  for i=1:numrodgauss
    rodsfs=[polyval(rn1d,rgpts(i))/Jac;
            polyval(rn2d,rgpts(i))/Jac];
    if (J1>(Iyy1+Izz1))|(J2>(Iyy2+Izz2))
      if (J1>(Iyy1+Izz1))
      disp('WARNING: J1 must be <= Iyy1+Izz1')%More checks for reality
      end
      if (J2>(Iyy2+Izz2))
      disp('WARNING: J2 must be <= Iyy2+Izz2')%More checks for reality
      end
      disp(['Error in element properties number '...
          num2str(element(elnum).properties) ...
          'used by element ' num2str(elnum) ' on line'...
          num2str(element(elnum).lineno) '.'])
    end
    J=polyval(rn1*J1+rn2*J2,bgpts(i));% J at gauss point.
    A=polyval(rn1*A1+rn2*A2,bgpts(i));% A at gauss point
    krod=krod+rgpw(i)*rodsfs*rodsfs'*A*E*Jac;%Since the shape
                                            %functions and Gauss
                                            %points are the same,
                                            %we are doing the rod
                                            %and torsion rod
                                            %together.
    ktor=ktor+rgpw(i)*rodsfs*rodsfs'*J*G*Jac;
  end

  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  %
  % Derivation of Mass matrices
  %
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  numbeamgauss=numbeamgauss+3; %Need more gauss points for the mass
                               %matrix.
  [bgpts,bgpw]=gauss(numbeamgauss);
  mb1=zeros(4,4); %initialize empty mass matrix
  % Local Bending in x-y plane
  for i=1:numbeamgauss
    beamsfs=[polyval(bn1,bgpts(i));
             polyval(bn2,bgpts(i))*Jac;
             polyval(bn3,bgpts(i));
             polyval(bn4,bgpts(i))*Jac];
    A=polyval(rn1*A1+rn2*A2,bgpts(i));
    mb1=mb1+bgpw(i)*beamsfs*beamsfs'*rho*A*Jac;%pause, and reflect
                                              %(OK, this was for
debugging)
  end

  % Local Bending in x-z plane
  mb2=zeros(4,4);
  for i=1:numbeamgauss
    beamsfs=[polyval(bn1,bgpts(i));
```

```
            -polyval(bn2,bgpts(i))*Jac;
            polyval(bn3,bgpts(i));
            -polyval(bn4,bgpts(i))*Jac];
  A=polyval(rn1*A1+rn2*A2,bgpts(i));
  mb2=mb2+bgpw(i)*beamsfs*beamsfs'*rho*A*Jac;
end

% Local Extension in x, torsion about x
numrodgauss=numrodgauss+1; %Need more gauss points for the mass
                           %matrix.
[rgpts,rgpw]=gauss(numrodgauss);
mrod=zeros(2,2); %initialize empty mass matrix
mtor=zeros(2,2);
for i=1:numrodgauss
  rodsfs=[polyval(rn1,rgpts(i));
          polyval(rn2,rgpts(i))];
  J=polyval(rn1*(Iyy1+Izz1)+rn2*(Iyy2+Izz2),bgpts(i));
  A=polyval(rn1*A1+rn2*A2,bgpts(i));
  mrod=mrod+rgpw(i)*rodsfs*rodsfs'*A*rho*Jac;
  mtor=mtor+rgpw(i)*rodsfs*rodsfs'*J*rho*Jac;
end

% Assembling each stiffness matrix into the complete elemental
% stiffness matrix. We're just telling the sub-elements to be put
% into the correct spots for the total element.
k=zeros(12,12);
k([2 6 8 12],[2 6 8 12])=kb1;
k([3 5 9 11],[3 5 9 11])=kb2;
k([1 7],[1 7])=krod;
k([4 10],[4 10])=ktor;

% Assembling each mass matrix into the complete elemental
% mass matrix
m=zeros(12,12);
m([2 6 8 12],[2 6 8 12])=mb1;
m([3 5 9 11],[3 5 9 11])=mb2;
m([1 7],[1 7])=mrod;
m([4 10],[4 10])=mtor;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Coordinate rotations
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

R1=([x2 y2 z2]-[x1 y1 z1]);% Vector along element
lam1=R1/norm(R1);% Unit direction
R2=([x3 y3 z3]-[x1 y1 z1]);% Unit direction to point
R2perp=R2-dot(R2,lam1)*lam1;% Part of R2 perpendicular to lam1
udirec=0;
while norm(R2perp)<10*eps% If R2perp is too small, (point in line
                         % with element, we need to cover the
                         % users a$$ and generate a point that
                         % isn't. We should put out a warning,
```

```
                              % but I commented it out.
    udirec=udirec+1;
    %disp('oops'); %This was my warning.
    %pause
    [minval,minloc]=min(lam1);
    R2perp=zeros(1,3);
    R2perp(udirec)=1;
    R2perp=R2perp-dot(R2perp,lam1)*lam1;
  end
  %Make the unit direction vectors for rotating and put them in the
  %rotation matrix.
  lam2=R2perp/norm(R2perp);
  lam3=cross(lam1,lam2);
  lamloc=[lam1;lam2;lam3];
  lam=sparse(12,12);
  lam(1:3,1:3)=lamloc;
  lam(4:6,4:6)=lamloc;
  lam(7:9,7:9)=lamloc;
  lam(10:12,10:12)=lamloc;
%   lam(13:15,13:15)=lamloc;
%   lam(16:18,16:18)=lamloc;

% $$$     lam=[lamloc z z z z z;
% $$$          z lamloc z z z z;
% $$$          z z lamloc z z z;
% $$$          z z z lamloc z z;
% $$$          z z z z lamloc z;
% $$$          z z z z z lamloc];
  element(elnum).lambda=lam;
  element(elnum).m=m;
  element(elnum).k=k;

  kg=lam'*k*lam;
  mg=lam'*m*lam;

  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  %
  % Assembling matrices into global matrices
  %
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  bn1=bnodes(1);bn2=bnodes(2);
  indices=[bn1*6+(-5:0) bn2*6+(-5:0)] ;


  K(indices,indices)=K(indices,indices)+kg;
  M(indices,indices)=M(indices,indices)+mg;

  % At this point we also know how to draw the element (what lines
  % and surfaces exist). For the beam3 element, 2 lines are
  % appropriate. Just add the pair of node numbers to the lines
  % array and that line will always be drawn.
  numlines=size(lines,1);
  lines(numlines+1,:)=[bn1 bn2];
```

```
  %If I have 4 nodes that I want to use to represent a surface, I
  %do the following.
  panelcolor=[1 0 1];% This picks a color. You can change the
                     % numbes between 0 and 1.
  %Don't like this color? Use colorui to pick another one. Another
  %option is that if we can't see the elements separately we can
  %chunk up x*y*z, divide by x*y*x of element, see if we get
  %integer powers or not to define colors that vary by panel.


  % You need to uncomment this line and assign values to node1,
  % node2, node3, and node4 in order to draw A SINGLE SURFACE. For
  % a brick, you need 6 lines like this.
  %surfs=[surfs;node1 node2 node3 node4 panelcolor];

  %Each surface can have a different color if you like. Just change
  %the last three numbers on the row corresponding to that
  %surface.

%diag(M)
elseif strcmp(mode,'istrainforces')
  % You don't need this
  % We need to have the stiffness matrix and the coordinate roation
matrix.



elseif strcmp(mode,'draw')
elseif strcmp(mode,'buckle')
end
```