```
function zn = find_natural_coords(a, xx, eltype)
%Find natural coordinates (zn) of any xyz point in an element
%Elt type does nothing right now, but might be used later

%a is the xyz point of interest, x is a 3x8 matrix of nodal
 coordinates


%Test Case
% a=[0.5 0.25 0.5]';
% x=[0 0 0 1 0 0 0 1 0 1 1 0 0 0 1 1 0 1 0 1 1 1 1 1]';
% find_natural_coords(a, x, 'hex')

% Initial guess of local coordinates
chi = 0.1; eta = 0.1; iota = 0.1;
zn = [chi; eta; iota];

%Hex Shape Functions
%  N1 = -((chi - 1)*(eta - 1)*(iota - 1))/ 8;
%  N2 = ((chi + 1)*(eta - 1)*(iota - 1))/ 8;
%  N3 = ((chi - 1)*(eta + 1)*(iota - 1))/ 8;
%  N4 = -((chi + 1)*(eta + 1)*(iota - 1))/ 8;
%  N5 = ((chi - 1)*(eta - 1)*(iota + 1))/ 8;
%  N6 = -((chi + 1)*(eta - 1)*(iota + 1))/ 8;
%  N8 = -((chi - 1)*(eta + 1)*(iota + 1))/8;
%  N7 = ((chi + 1)*(eta + 1)*(iota + 1))/8;
%  NN =[N1, 0, 0, N2, 0, 0, N3, 0, 0, N4, 0, 0, N5, 0, 0, N6, 0, 0,
 N7, ...
%      0, 0, N8, 0, 0;,  0, N1, 0, 0, N2, 0, 0, N3, 0, 0, N4, 0, 0,
 N5, ...
%     0, 0, N6, 0, 0, N7, 0, 0, N8, 0;,  0, 0, N1, 0, 0, N2, 0, 0,
 N3, ...
%     0, 0, N4, 0, 0, N5, 0, 0, N6, 0, 0, N7, 0, 0, N8];
% %
%  Nodal Coordinates
%  x = Transpose[x1, y1, z1, x2, y2, z2, x3, y3, z3, x4, y4, z4, x5,
%      y5, z5, x6, y6, z6, x7, y7, z7, x8, y8, z8];

%Reorder a 3x8 matrix to be a mess of a column matrix
x = [xx(:,1)' xx(:,2)' xx(:,3)' xx(:,4)' xx(:,5)' xx(:,6)' xx(:,7)'
 xx(:,8)']';

    x1=x(1); y1=x(2); z1=x(3); x2=x(4); y2=x(5); z2=x(6); x3=x(7);
 y3=x(8);
    z3=x(9); x4=x(10); y4=x(11); z4=x(12); x5=x(13); y5=x(14);
 z5=x(15);
    x6=x(16); y6=x(17); z6=x(18); x7=x(19); y7=x(20); z7=x(21);
 x8=x(22);
    y8=x(23); z8=x(24);

% Real coordinates of point we want natural coords for
a1=a(1); a2=a(2); a3=a(3);
```

```
a=[a1 a2 a3]';

%Solve NN*x-a with Newton Rapson
err = [1 1 1]';
tol=10^-6;

while abs(err(1))>tol || abs(err(2))>tol || abs(err(3))>tol

    %Calculate function F=NN*x-a=0 to be solved based on current
 values of
    %chi eta and iota
    F = [ 1/8 *(-8* a1 - (-1 + chi) * (-1 + eta) * (-1 + iota)* x1 +
 (1 + chi)* (-1 +  ...
     eta)* (-1 + iota)* x2 + (-1 + chi)* (1 + eta)* (-1 +  ...
     iota)* x3 - (1 + chi) *(1 + eta) *(-1 + iota) *x4 + (-1 + ...
     chi)* (-1 + eta)* (1 + iota)* x5 - (1 + chi)* (-1 + eta)* (1
 + ...
     iota)* x6 + (1 + chi)* (1 + eta)* (1 + iota)* x7 - (-1 + chi)* (1
 + ...
      eta)* (1 + iota)* x8);  1/8 ...
       * (-8 *a2 - (-1 + chi) *(-1 + eta) *(-1 + iota)* y1 + (1 +
 chi)* (-1 +  ...
     eta)* (-1 + iota)* y2 + (-1 + chi)* (1 + eta)* (-1 +  ...
     iota)* y3 - (1 + chi)* (1 + eta)* (-1 + iota)* y4 + (-1 +  ...
     chi)* (-1 + eta)* (1 + iota)* y5 - (1 + chi)* (-1 + eta)* (1 +
  ...
     iota)* y6 + (1 + chi)* (1 + eta)* (1 + iota)* y7 - (-1 + chi)* (1
 + ...
      eta)* (1 + iota)* y8);  1/8 ...
       * (-8* a3 - (-1 + chi) *(-1 + eta)* (-1 + iota) *z1 + (1 + chi)
 *(-1 +  ...
     eta)* (-1 + iota)* z2 + (-1 + chi)* (1 + eta)* (-1 +  ...
     iota)* z3 - (1 + chi)* (1 + eta)* (-1 + iota)* z4 + (-1 +  ...
     chi)* (-1 + eta)* (1 + iota)* z5 - (1 + chi)* (-1 + eta)* (1 +
  ...
     iota)* z6 + (1 + chi)* (1 + eta)* (1 + iota)* z7 - (-1 + chi)* (1
 + ...
      eta)* (1 + iota)* z8)];

    %Jacobian of F
    df1dz1 = -(1/8)*(-1 + eta)*(-1 + iota)*x1 + 1/8*(-1 + eta)*(-1 +
 iota)*x2 + ...
    1/8*(1 + eta)*(-1 + iota)*x3 - 1/8*(1 + eta)*(-1 + iota)*x4 + ...
    1/8*(-1 + eta)*(1 + iota)*x5 - 1/8*(-1 + eta)*(1 + iota)*x6 + ...
    1/8*(1 + eta)*(1 + iota)*x7 - 1/8*(1 + eta)*(1 + iota)*x8;

    df2dz1 =  -(1/8)*(-1 + eta)*(-1 + iota)*y1 + 1/8*(-1 + eta)*(-1 +
 iota)*y2 + ...
    1/8*(1 + eta)*(-1 + iota)*y3 - 1/8*(1 + eta)*(-1 + iota)*y4 + ...
    1/8*(-1 + eta)*(1 + iota)*y5 - 1/8*(-1 + eta)*(1 + iota)*y6 + ...
    1/8*(1 + eta)*(1 + iota)*y7 - 1/8*(1 + eta)*(1 + iota)*y8;

    df3dz1 =  -(1/8)*(-1 + eta)*(-1 + iota)*z1 + 1/8*(-1 + eta)*(-1 +
 iota)*z2 + ...
```

```
   1/8*(1 + eta)*(-1 + iota)*z3 - 1/8*(1 + eta)*(-1 + iota)*z4 + ...
   1/8*(-1 + eta)*(1 + iota)*z5 - 1/8*(-1 + eta)*(1 + iota)*z6 + ...
   1/8*(1 + eta)*(1 + iota)*z7 - 1/8*(1 + eta)*(1 + iota)*z8;

   df1dz2 =  -(1/8)*(-1 + chi)*(-1 + iota)*x1 + 1/8*(1 + chi)*(-1 +
iota)*x2 + ...
   1/8*(-1 + chi)*(-1 + iota)*x3 - 1/8*(1 + chi)*(-1 + iota)*x4 + ...
   1/8*(-1 + chi)*(1 + iota)*x5 - 1/8*(1 + chi)*(1 + iota)*x6 + ...
   1/8*(1 + chi)*(1 + iota)*x7 - 1/8*(-1 + chi)*(1 + iota)*x8;

   df2dz2 =  -(1/8)*(-1 + chi)*(-1 + iota)*y1 + 1/8*(1 + chi)*(-1 +
iota)*y2 + ...
   1/8*(-1 + chi)*(-1 + iota)*y3 - 1/8*(1 + chi)*(-1 + iota)*y4 + ...
   1/8*(-1 + chi)*(1 + iota)*y5 - 1/8*(1 + chi)*(1 + iota)*y6 + ...
   1/8*(1 + chi)*(1 + iota)*y7 - 1/8*(-1 + chi)*(1 + iota)*y8;

   df3dz2 =  -(1/8)*(-1 + chi)*(-1 + iota)*z1 + 1/8*(1 + chi)*(-1 +
iota)*z2 + ...
   1/8*(-1 + chi)*(-1 + iota)*z3 - 1/8*(1 + chi)*(-1 + iota)*z4 + ...
   1/8*(-1 + chi)*(1 + iota)*z5 - 1/8*(1 + chi)*(1 + iota)*z6 + ...
   1/8*(1 + chi)*(1 + iota)*z7 - 1/8*(-1 + chi)*(1 + iota)*z8;

   df1dz3 =  -(1/8)*(-1 + chi)*(-1 + eta)*x1 + 1/8*(1 + chi)*(-1 +
eta)*x2 + ...
   1/8*(-1 + chi)*(1 + eta)*x3 - 1/8*(1 + chi)*(1 + eta)*x4 + ...
   1/8*(-1 + chi)*(-1 + eta)*x5 - 1/8*(1 + chi)*(-1 + eta)*x6 + ...
   1/8*(1 + chi)*(1 + eta)*x7 - 1/8*(-1 + chi)*(1 + eta)*x8;

   df2dz3 =  -(1/8)*(-1 + chi)*(-1 + eta)*y1 + 1/8*(1 + chi)*(-1 +
eta)*y2 + ...
   1/8*(-1 + chi)*(1 + eta)*y3 - 1/8*(1 + chi)*(1 + eta)*y4 + ...
   1/8*(-1 + chi)*(-1 + eta)*y5 - 1/8*(1 + chi)*(-1 + eta)*y6 + ...
   1/8*(1 + chi)*(1 + eta)*y7 - 1/8*(-1 + chi)*(1 + eta)*y8;

   df3dz3 =  -(1/8)*(-1 + chi)*(-1 + eta)*z1 + 1/8*(1 + chi)*(-1 +
eta)*z2 + ...
   1/8*(-1 + chi)*(1 + eta)*z3 - 1/8*(1 + chi)*(1 + eta)*z4 + ...
   1/8*(-1 + chi)*(-1 + eta)*z5 - 1/8*(1 + chi)*(-1 + eta)*z6 + ...
   1/8*(1 + chi)*(1 + eta)*z7 - 1/8*(-1 + chi)*(1 + eta)*z8;

   J=[df1dz1, df1dz2, df1dz3; df2dz1, df2dz2, df2dz3; df3dz1, df3dz2,
df3dz3];

   %Solve J(zn)*delz=f(zn) for delz
   delz=-J\F;

   det(J);
   %Calculate the new guess for the natural coordinates zn+1 (znew)
   % delz = zn+1 + zn
   znew = delz + zn;
   zn = znew;
   chi = zn(1); eta = zn(2); iota = zn(3);
%     if chi>1
%         chi=chi/abs(chi);
%     end

    %Check if that is the solution
    %Compute the shape functions with the new guess of chi eta and
 iota
     N1 = -((chi - 1)*(eta - 1)*(iota - 1))/ 8;
     N2 = ((chi + 1)*(eta - 1)*(iota - 1))/ 8;
     N3 = ((chi - 1)*(eta + 1)*(iota - 1))/ 8;
     N4 = -((chi + 1)*(eta + 1)*(iota - 1))/ 8;
     N5 = ((chi - 1)*(eta - 1)*(iota + 1))/ 8;
     N6 = -((chi + 1)*(eta - 1)*(iota + 1))/ 8;
     N8 = -((chi - 1)*(eta + 1)*(iota + 1))/8;
     N7 = ((chi + 1)*(eta + 1)*(iota + 1))/8;
     NN = [N1, 0, 0, N2, 0, 0, N3, 0, 0, N4, 0, 0, N5, 0, 0, N6, 0, 0,
 N7, ...
         0, 0, N8, 0, 0;  0, N1, 0, 0, N2, 0, 0, N3, 0, 0, N4, 0, 0,
 N5, ...
        0, 0, N6, 0, 0, N7, 0, 0, N8, 0;  0, 0, N1, 0, 0, N2, 0, 0,
 N3, ...
        0, 0, N4, 0, 0, N5, 0, 0, N6, 0, 0, N7, 0, 0, N8];

   %Check the error between the known xyz coordinate (a) and the
 calculated
   % xyz based on the natural coordinates
    err = (a - NN*x);
end

%Round machine zero results to actual zero for simplicity
for j=1:length(zn)
    if abs(zn(j)) < 1E-12
        zn(j)=0;
    end
end


end
```

```
Not enough input arguments.

Error in find_natural_coords (line 36)
x = [xx(:,1)' xx(:,2)' xx(:,3)' xx(:,4)' xx(:,5)' xx(:,6)' xx(:,7)'
 xx(:,8)']';
```

*Published with MATLAB® R2019b*