```
function [ D , Ksis] = Solver_v2( Cor,Pos,R,E,dofConstraints,P,nel,ndof)
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here


% Here we define the position of the Gauss points in intrinsic co-ordinates
%for the 2x2x2 Gauss integration that must be performed to calculate each
%elements stiffness matrix

l=1;
gpoint(1)=-0.577350269189626;
gpoint(2)=0.577350269189626;
for k=1:2
    for j=1:2
        for i=1:2
W(l,:)=[gpoint(i),gpoint(j),gpoint(k)];
l=l+1;
        end
    end
end
%_________________________________________________|

clear l i j k;



tic
% Preallocate Variables

pxn=zeros(8); % Will hold global x coordinateof each of the nodes in el.
pyn=zeros(8);
pzn=zeros(8);
A=zeros(6,24,nel);% Array holding strain displacement matrices for each el.
K(24,24,nel)=0; % Array holding all element stiffness matrices


%___Element Stiffnes Matrix Calculation___________________________________
for s=1:nel;



% Assign the pxn,pyn,pzn. pxn(local node #)=position of node
for u=1:8
pxn(u) =Cor(Pos(s,u),1);  %x(u)
pyn(u) =Cor(Pos(s,u),2);  %y(u)
pzn(u) =Cor(Pos(s,u),3);  %z(u)
```

```
end
clear u

% find e n j which are intrinsic co-ordinates of gauss points e(i) is e
% co-ordiate of ith Gauss point
for i=1:8;


e=W(i,1);
n=W(i,2);
J=W(i,3);


% Call to fundction that returns the Jacobian for transformation between
% intrinic and global co-ordinates and the derivatives of the shape
% functions with respect to global co-ordinates Hx,Hy,Hz at teh ith Gauss
% point
[Jacobi,Hx,Hy,Hz]=AconnectH8(pxn,pyn,pzn,e,n,J);


% Generate the strain displacement matrix for the element
for f=1:8;
A(:,3*(f-1)+1,s)= [Hx(f);   0   ;   0  ;  Hy(f) ;   0   ;  Hz(f)];
A(:,3*(f-1)+2,s)= [ 0   ; Hy(f) ;   0  ;  Hx(f) ;  Hz(f);   0   ];
A(:,3*(f-1)+3,s)= [ 0   ;   0   ; Hz(f);   0    ;  Hy(f);  Hx(f)];
end

% Gauss integration is performed upon looping over all i
              K(:,:,s)=K(:,:,s)+det(Jacobi)*A(:,:,s)'*E*A(:,:,s);



end

end
clear i f
%_________________________________________________________________________




%______Assemble System Stiffness Matrix___________________________________
% This code assembles the vectors a b c which hold the indices and values
% of the non-zero elements of the system stiffness matrix.

l=0;
```

```
for n=1:nel;
        for i=1:24; % n.b 24 degrees of freedom in each element
            for j=1:24;

                if K(i,j,n) ~=0
                   l=l+1;

                   a(l)=R(n,j);
                   b(l)=R(n,i);
                   c(l)=K(i,j,n);

                end



            end%
        end%
end%_______________________________________________|
%Ksis:Global system stiffness matrix
clear n i j

%Use vectors a b c to generate a sparse matrix
Ksis=sparse(a,b,c);




% Apply Essential Boundary Conditions

bcwt=trace(Ksis)/24; % bcwt is used to ensure matrix is properly scaled

P =P'- Ksis(:,dofConstraints(:,1))*dofConstraints(:,2);
Ksis(:,dofConstraints(:,1)) = 0;
Ksis(dofConstraints(:,1),:) = 0;  % Here setting rows and columns correspo
% nding to constrained dofs to zero
Ksis(dofConstraints(:,1),dofConstraints(:,1)) = ...
    bcwt*speye(length(dofConstraints(:,1)));
P(dofConstraints(:,1)) = bcwt*dofConstraints(:,2); % These two lines make
% sure we haven't changed the system of linear equations in doing so.

%n.b both rows and columns corresponding to constrained dofs are set to zer
%o. Not stricly neccessary but maintains symmetry -> faster solution.
```

```
%Solve for displacements D using Cholesky Decomposition. The \ operator
%recognises sparse symmetric matrices and optimises accordingly.

    D = Ksis\P;




end
```

```
Error using Solver_v2 (line 33)
Not enough input arguments.
```