

**研究生期末大作业**

**课程名称：** 有限元方法及应用

**研究生姓名：** 夏艳红

**学号：** S230200215

**任课教师：** 王琥

# 一．线性三角形单元

在不规则形状物体受到外力作用时，可能会在某些部位造成应力集中，导致该部位易产生损坏，若能提前分析出应力大的部位，便能及时更改部件形状，以免造成不必要的损失。

下图为一受单一力的薄平板结构，平板长宽均为2，右下角为1\*1的缺口，板厚为0.2。杨氏模量为210Gpa，泊松比为0.3。

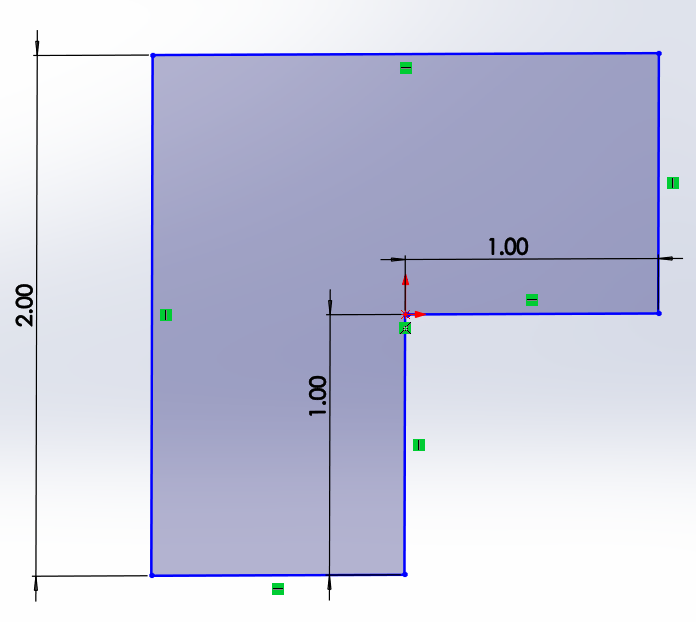


图1.平板结构

在划分不同数量表格及不同的受力下可得到下图结果

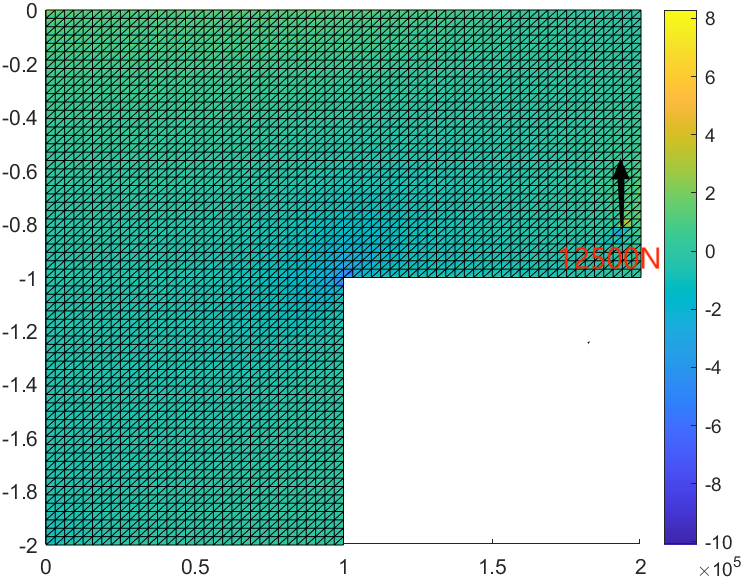
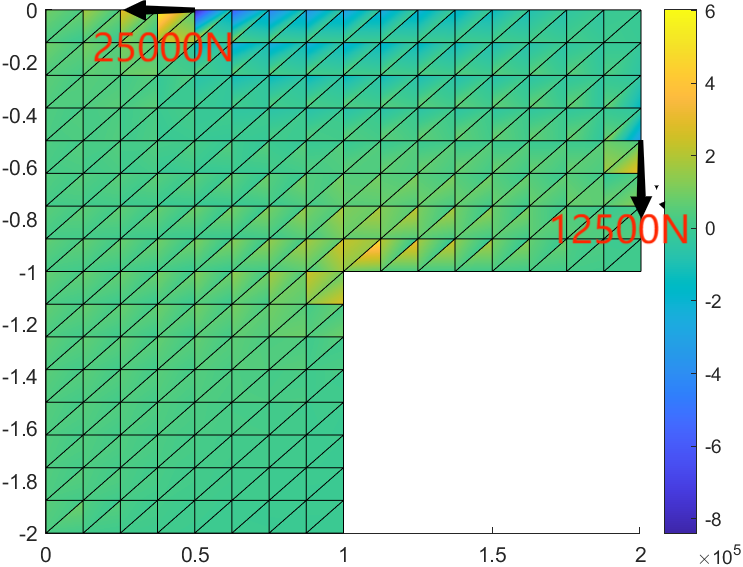


图2.线性三角形单元所得应力分布

# 线性四面体元

若平板长宽高相近，便不可忽略某一方向的尺寸，若采用线性三角形单元计算便会出现较大误差，得到的结果也会不对，例如下图

该三维结构长宽高皆为2，右下角为1\*1\*2的缺口，其中，杨氏模量为210Gpa，泊松比为0.3。

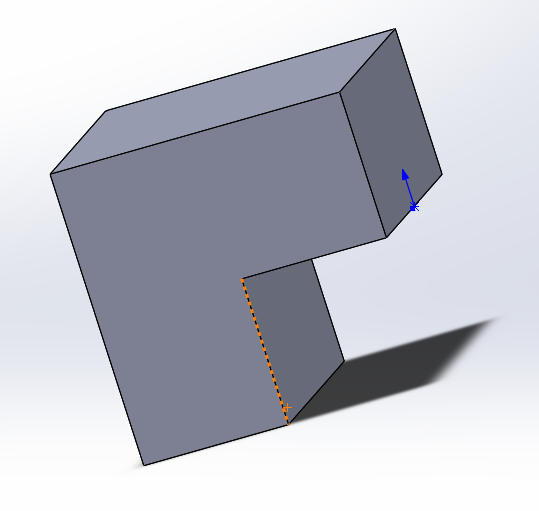


图3. 三维结构

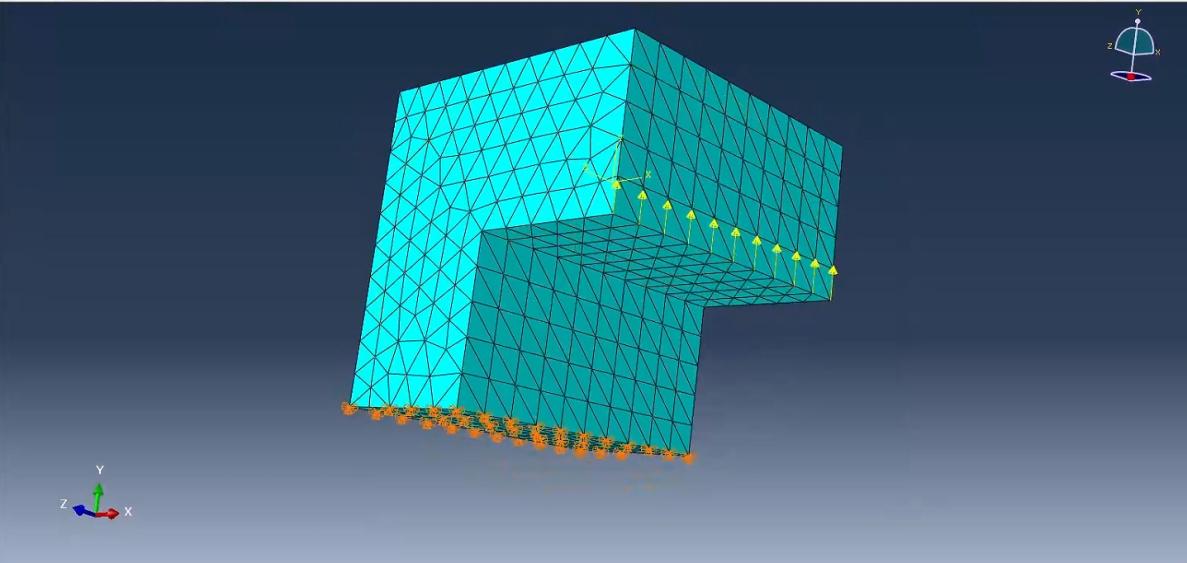
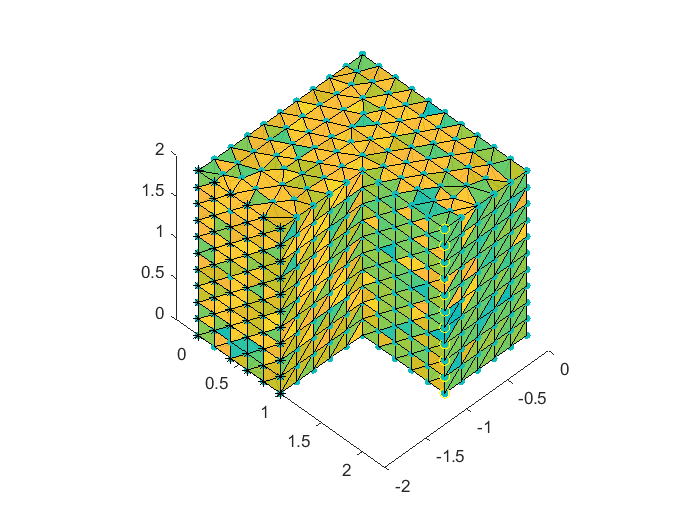


图4.abaqus网格划分及施加载荷、边界条件结果

此时运用线性四面体元可得到如下结果：



文件夹中文本文件xd、yd、zd分别为节点在x、y、z方向的位移分量

# 四．附录代码

1.线性三角形单元

**Main 1**

first\_time=cputime;

format long

%--------------------------------------------

%input data for control parameters

%--------------------------------------------

lengthx=2;              %length of x-axis side of problem

lengthy=2;              %length of y-axis side of proble

h=0.2;                  %thickness of problem

emodule=2.1\*10^11;            %elastic modulus

poisson=0.3;            %Poisson's ratio

fload=-12500;             % the total load

lx=16;                  % number of element in x-axis

ly=16;                   % number of element in y-axis

nel=2\*0.75\*lx\*ly;            % number of element

nnel=3;                 %number of nodes per element

ndof=2;                 %number of dofs per node

nnode=(lx/2+1)\*(ly+1)+(lx/2)\*(ly/2+1);    %total number of nodes in system

sdof=nnode\*ndof;        %total system dofs

edof=nnel\*ndof;         %degrees of freedom per element

x0=[];

for i=1:lx+1

    if i<=(lx/2+1)

         for j=1:ly+1

             x0=[x0; (i-1)\*lengthx/lx      -(j-1)\*lengthy/ly];

         end

    else

        for j=1:(ly/2+1)

            x0=[x0; (i-1)\*lengthx/lx      -(j-1)\*lengthy/ly];

        end

    end

end

nodes=[];

for i=1:lx

    if i<lx/2+1

        for j=1:ly

            nodes=[nodes; (ly+1)\*(i-1)+j (ly+1)\*i+j (ly+1)\*(i-1)+j+1;];

            nodes=[nodes; (ly+1)\*i+j (ly+1)\*i+j+1 (ly+1)\*(i-1)+j+1;];

        end

    elseif i==lx/2+1

            for j=1:ly/2

                nodes=[nodes; (ly+1)\*(i-1)+j (ly/2+1)\*(i-lx/2-1)+j+153  (ly+1)\*(i-1)+j+1;];

                nodes=[nodes; (ly/2+1)\*(i-lx/2-1)+j+153 (ly/2+1)\*(i-lx/2-1)+j+154 (ly+1)\*(i-1)+j+1;];

            end

    else

        for j=1:ly/2

            nodes=[nodes; (ly/2+1)\*(i-lx/2-2)+j+153 (ly/2+1)\*(i-lx/2-1)+j+153 (ly/2+1)\*(i-lx/2-2)+j+154;];

            nodes=[nodes; (ly/2+1)\*(i-lx/2-1)+j+153 (ly/2+1)\*(i-lx/2-1)+j+154 (ly/2+1)\*(i-lx/2-2)+j+154;];

        end

    end

end

bcdof=[];

bcval=[];

for i=1:ly+1

        bcdof=[bcdof 1+2\*(i-1) 2+2\*(i-1)];

        bcval=[bcval  0   0];

end

ff=sparse(sdof,1);          %system force vector

k=sparse(edof,edof);        %initialization of element matrix

kk=sparse(sdof,sdof);       %system matrix

disp=sparse(sdof,1);        %system displacement vector

eldisp=sparse(edof,1);      %element displacement vector

stress=zeros(nel,3);        %matrix containing stress components

strain=zeros(nel,3);        %matrix containing strain components

index=sparse(edof,1);       %index vector

kinmtx=sparse(3,edof);      %kinematic matrix

matmtx=sparse(3,3);     %constitutive matrix

matmtx=fematiso(1,emodule,poisson);     %constitutive matrice

for iel=1:nel       %loop for the total number of element

    nd(1)=nodes(iel,1);         %1st connected node for (iel)-th element

    nd(2)=nodes(iel,2);         %2nd connected node for (iel)-th element

    nd(3)=nodes(iel,3);         %3rd connected node for (iel)-th element

    x1=x0(nd(1),1); y1=x0(nd(1),2);     %coord values of 1st node

    x2=x0(nd(2),1); y2=x0(nd(2),2);     %coord values of 2nd node

    x3=x0(nd(3),1); y3=x0(nd(3),2);     %coord values of 3rd node

    index=feeldof(nd,nnel,ndof);        %extract system dofs for the element

    area=abs(0.5\*(x1\*y2+x2\*y3+x3\*y1-x1\*y3-x2\*y1-x3\*y2)); %area of triangula

    area2=area\*2;

     dhdx=(1/area2)\*[(y2-y3) (y3-y1) (y1-y2)];       %derivatives w.r.t x

     dhdy=(1/area2)\*[(x3-x2) (x1-x3) (x2-x1)];       %derivatives w.r.t y

    kinmtx2=fekine2d(nnel,dhdx,dhdy);               %kinematic matrice

    k=kinmtx2'\*matmtx\*kinmtx2\*area\*h;              %element stiffness matrice

    kk=feasmb\_2(kk,k,index);                               %assemble element matrics

end

kk1=kk;

%--------------------------------------------------------------------------

%   force vector

%--------------------------------------------------------------------------

ff(442,1)=fload;

ff(137,1)=2\*fload;

%------------------------

%  apply boundary condition

%------------------------

[kk,ff]=feaplyc2(kk,ff,bcdof,bcval);

%-------------------------

%  solve the matrix equation

%-------------------------

%disp=kk\ff;

[LL UU]=lu(kk);

utemp=LL\ff;

disp=UU\utemp;

EU=0.5\*disp'\*kk1\*disp;

%---------------------------------------

%  element stress computation

%---------------------------------------

energy=0;

Patch\_xy = zeros(6,nel);

for ielp=1:nel           % loop for the total number of elements

     nd(1)=nodes(ielp,1); % 1st connected node for (iel)-th element

     nd(2)=nodes(ielp,2); % 2nd connected node for (iel)-th element

     nd(3)=nodes(ielp,3); % 3rd connected node for (iel)-th element

    x1=x0(nd(1),1); y1=x0(nd(1),2);% coord values of 1st node

    x2=x0(nd(2),1); y2=x0(nd(2),2);% coord values of 2nd node

    x3=x0(nd(3),1); y3=x0(nd(3),2);% coord values of 3rd node

    Patch\_xy(:,ielp)=[x1;x2;x3;y1;y2;y3];

    xcentre=(x1+x2+x3)/3; ycentre=(y1+y2+y3)/3;

    index=feeldof(nd,nnel,ndof);% extract system dofs associated with element

    %-------------------------------------------------------

    %  extract element displacement vector

    %-------------------------------------------------------

   for i=1:edof

        eldisp(i)=disp(index(i));

    end

    area=abs(0.5\*(x1\*y2+x2\*y3+x3\*y1-x1\*y3-x2\*y1-x3\*y2));  % area of triangule

    area2=area\*2;

    dhdx=(1/area2)\*[(y2-y3) (y3-y1) (y1-y2)];  % derivatives w.r.t. x-axis

    dhdy=(1/area2)\*[(x3-x2) (x1-x3) (x2-x1)];  % derivatives w.r.t. y-axis

    kinmtx2=fekine2d(nnel,dhdx,dhdy);          % compute kinematic matrix

    estrain=kinmtx2\*eldisp;             % compute strains

    estress=matmtx\*estrain;             % compute stresses

    for i=1:3

        strain(ielp,i)=estrain(i);          % store for each element

        stress(ielp,i)=estress(i);          % store for each element

    end

    energy=energy+0.5\*estrain'\*matmtx\*estrain\*area;

end

neigh\_node = cell(nnode,1);

indneigh=zeros(1,nnode);

for i=1:nel

    for j=1:3

        indneigh(nodes(i,j))=indneigh(nodes(i,j))+1;

        neigh\_node{nodes(i,j)}(indneigh(nodes(i,j)))=i;

    end

end

stress\_node=zeros(3,nnode);

for inode=1:nnode

      numel= indneigh(inode);

      for i=1:numel

            ind\_nel= neigh\_node{inode}(i);

           for j=1:3

                 stress\_node(j,inode)=stress\_node(j,inode)+stress(ind\_nel,j);

           end

    end

    stress\_node(:,inode)=stress\_node(:,inode)/numel;

end

%-------------------------------------------------------------

% Data output

%-------------------------------------------------------------

fid\_out=fopen('result\_beam02.plt','w');

fprintf(fid\_out,'TITLE="test case governed by poisson equation"\n');

fprintf(fid\_out,'VARIABLES="x" "y" "u" "v" "sigax"  "sigmay" "sigmaxy"\n');

fprintf(fid\_out,'ZONE T="flow-field", N= %8d,E=%8d,ET=TRIANGLE, F=FEPOINT\n',nnode,nel);

disp=full(disp);

for i=1:nnode

       fprintf(fid\_out,'%16.6e%16.6e%16.6e%16.6e%16.6e%16.6e%16.6e\n',x0(i,1),x0(i,2),disp(2\*i-1),disp(2\*i),stress\_node(1,i),stress\_node(2,i),stress\_node(3,i));

end

for i=1:nel

      fprintf(fid\_out,'%8d%8d%8d\n',nodes(i,1),nodes(i,2),nodes(i,3));

end

scatter(x0(:,1), x0(:,2),50,disp(1:2:end),"filled");

colorbar

clf

hold on

patch(Patch\_xy(1:3,:), Patch\_xy(4:6,:),stress.');

colorbar

**Main 2**

first\_time=cputime;

format long

%--------------------------------------------

%input data for control parameters

%--------------------------------------------

lengthx=2;              %length of x-axis side of problem

lengthy=2;              %length of y-axis side of proble

h=0.2;                  %thickness of problem

emodule=2.1\*10^11;            %elastic modulus

poisson=0.3;            %Poisson's ratio

fload=-12500;             % the total load

lx=64;                  % number of element in x-axis

ly=64;                   % number of element in y-axis

nel=2\*0.75\*lx\*ly;            % number of element

nnel=3;                 %number of nodes per element

ndof=2;                 %number of dofs per node

nnode=(lx/2+1)\*(ly+1)+(lx/2)\*(ly/2+1);    %total number of nodes in system

sdof=nnode\*ndof;        %total system dofs

edof=nnel\*ndof;         %degrees of freedom per element

x0=[];

for i=1:lx+1

    if i<=(lx/2+1)

         for j=1:ly+1

             x0=[x0; (i-1)\*lengthx/lx      -(j-1)\*lengthy/ly];

         end

    else

        for j=1:(ly/2+1)

            x0=[x0; (i-1)\*lengthx/lx      -(j-1)\*lengthy/ly];

        end

    end

end

nodes=[];

for i=1:lx

    if i<lx/2+1

        for j=1:ly

            nodes=[nodes; (ly+1)\*(i-1)+j (ly+1)\*i+j (ly+1)\*(i-1)+j+1;];

            nodes=[nodes; (ly+1)\*i+j (ly+1)\*i+j+1 (ly+1)\*(i-1)+j+1;];

        end

    elseif i==lx/2+1

            for j=1:ly/2

                nodes=[nodes; (ly+1)\*(i-1)+j (ly/2+1)\*(i-lx/2-1)+j+2145  (ly+1)\*(i-1)+j+1;];

                nodes=[nodes; (ly/2+1)\*(i-lx/2-1)+j+2145 (ly/2+1)\*(i-lx/2-1)+j+2146 (ly+1)\*(i-1)+j+1;];

            end

    else

        for j=1:ly/2

            nodes=[nodes; (ly/2+1)\*(i-lx/2-2)+j+2145 (ly/2+1)\*(i-lx/2-1)+j+2145 (ly/2+1)\*(i-lx/2-2)+j+2146;];

            nodes=[nodes; (ly/2+1)\*(i-lx/2-1)+j+2145 (ly/2+1)\*(i-lx/2-1)+j+2146 (ly/2+1)\*(i-lx/2-2)+j+2146;];

        end

    end

end

bcdof=[];

bcval=[];

for i=1:ly+1

        bcdof=[bcdof 1+2\*(i-1) 2+2\*(i-1)];

        bcval=[bcval  0   0];

end

ff=sparse(sdof,1);          %system force vector

k=sparse(edof,edof);        %initialization of element matrix

kk=sparse(sdof,sdof);       %system matrix

disp=sparse(sdof,1);        %system displacement vector

eldisp=sparse(edof,1);      %element displacement vector

stress=zeros(nel,3);        %matrix containing stress components

strain=zeros(nel,3);        %matrix containing strain components

index=sparse(edof,1);       %index vector

kinmtx=sparse(3,edof);      %kinematic matrix

matmtx=sparse(3,3);     %constitutive matrix

matmtx=fematiso(1,emodule,poisson);     %constitutive matrice

for iel=1:nel       %loop for the total number of element

    nd(1)=nodes(iel,1);         %1st connected node for (iel)-th element

    nd(2)=nodes(iel,2);         %2nd connected node for (iel)-th element

    nd(3)=nodes(iel,3);         %3rd connected node for (iel)-th element

    x1=x0(nd(1),1); y1=x0(nd(1),2);     %coord values of 1st node

    x2=x0(nd(2),1); y2=x0(nd(2),2);     %coord values of 2nd node

    x3=x0(nd(3),1); y3=x0(nd(3),2);     %coord values of 3rd node

    index=feeldof(nd,nnel,ndof);        %extract system dofs for the element

    area=abs(0.5\*(x1\*y2+x2\*y3+x3\*y1-x1\*y3-x2\*y1-x3\*y2)); %area of triangula

    area2=area\*2;

     dhdx=(1/area2)\*[(y2-y3) (y3-y1) (y1-y2)];       %derivatives w.r.t x

     dhdy=(1/area2)\*[(x3-x2) (x1-x3) (x2-x1)];       %derivatives w.r.t y

    kinmtx2=fekine2d(nnel,dhdx,dhdy);               %kinematic matrice

    k=kinmtx2'\*matmtx\*kinmtx2\*area\*h;              %element stiffness matrice

    kk=feasmb\_2(kk,k,index);                               %assemble element matrics

end

kk1=kk;

%--------------------------------------------------------------------------

%   force vector

%--------------------------------------------------------------------------

ff(6258,1)=-fload;

%ff(5000,1)=2\*fload;

%------------------------

%  apply boundary condition

%------------------------

[kk,ff]=feaplyc2(kk,ff,bcdof,bcval);

%-------------------------

%  solve the matrix equation

%-------------------------

%disp=kk\ff;

[LL UU]=lu(kk);

utemp=LL\ff;

disp=UU\utemp;

EU=0.5\*disp'\*kk1\*disp;

%---------------------------------------

%  element stress computation

%---------------------------------------

energy=0;

Patch\_xy = zeros(6,nel);

for ielp=1:nel           % loop for the total number of elements

     nd(1)=nodes(ielp,1); % 1st connected node for (iel)-th element

     nd(2)=nodes(ielp,2); % 2nd connected node for (iel)-th element

     nd(3)=nodes(ielp,3); % 3rd connected node for (iel)-th element

    x1=x0(nd(1),1); y1=x0(nd(1),2);% coord values of 1st node

    x2=x0(nd(2),1); y2=x0(nd(2),2);% coord values of 2nd node

    x3=x0(nd(3),1); y3=x0(nd(3),2);% coord values of 3rd node

    Patch\_xy(:,ielp)=[x1;x2;x3;y1;y2;y3];

    xcentre=(x1+x2+x3)/3; ycentre=(y1+y2+y3)/3;

    index=feeldof(nd,nnel,ndof);% extract system dofs associated with element

    %-------------------------------------------------------

    %  extract element displacement vector

    %-------------------------------------------------------

   for i=1:edof

        eldisp(i)=disp(index(i));

    end

    area=abs(0.5\*(x1\*y2+x2\*y3+x3\*y1-x1\*y3-x2\*y1-x3\*y2));  % area of triangule

    area2=area\*2;

    dhdx=(1/area2)\*[(y2-y3) (y3-y1) (y1-y2)];  % derivatives w.r.t. x-axis

    dhdy=(1/area2)\*[(x3-x2) (x1-x3) (x2-x1)];  % derivatives w.r.t. y-axis

    kinmtx2=fekine2d(nnel,dhdx,dhdy);          % compute kinematic matrix

    estrain=kinmtx2\*eldisp;             % compute strains

    estress=matmtx\*estrain;             % compute stresses

    for i=1:3

        strain(ielp,i)=estrain(i);          % store for each element

        stress(ielp,i)=estress(i);          % store for each element

    end

    energy=energy+0.5\*estrain'\*matmtx\*estrain\*area;

end

neigh\_node = cell(nnode,1);

indneigh=zeros(1,nnode);

for i=1:nel

    for j=1:3

        indneigh(nodes(i,j))=indneigh(nodes(i,j))+1;

        neigh\_node{nodes(i,j)}(indneigh(nodes(i,j)))=i;

    end

end

stress\_node=zeros(3,nnode);

for inode=1:nnode

      numel= indneigh(inode);

      for i=1:numel

            ind\_nel= neigh\_node{inode}(i);

           for j=1:3

                 stress\_node(j,inode)=stress\_node(j,inode)+stress(ind\_nel,j);

           end

    end

    stress\_node(:,inode)=stress\_node(:,inode)/numel;

end

%-------------------------------------------------------------

% Data output

%-------------------------------------------------------------

fid\_out=fopen('result\_beam01.plt','w');

fprintf(fid\_out,'TITLE="test case governed by poisson equation"\n');

fprintf(fid\_out,'VARIABLES="x" "y" "u" "v" "sigax"  "sigmay" "sigmaxy"\n');

fprintf(fid\_out,'ZONE T="flow-field", N= %8d,E=%8d,ET=TRIANGLE, F=FEPOINT\n',nnode,nel);

disp=full(disp);

for i=1:nnode

       fprintf(fid\_out,'%16.6e%16.6e%16.6e%16.6e%16.6e%16.6e%16.6e\n',x0(i,1),x0(i,2),disp(2\*i-1),disp(2\*i),stress\_node(1,i),stress\_node(2,i),stress\_node(3,i));

end

for i=1:nel

      fprintf(fid\_out,'%8d%8d%8d\n',nodes(i,1),nodes(i,2),nodes(i,3));

end

scatter(x0(:,1), x0(:,2),50,disp(1:2:end),"filled");

colorbar

clf

hold on

patch(Patch\_xy(1:3,:), Patch\_xy(4:6,:),stress.');

colorbar

function [kk,ff]=feaplyc2(kk,ff,bcdof,bcval)

 n=length(bcdof);

 sdof=size(kk);

 for i=1:n

      c=bcdof(i);

      for  j=1:sdof

             kk(c,j)=0;

      end

 kk(c,c)=1;

    ff(c)=bcval(i);

 end

function [kk]=feasmb\_2(kk,k,index)

edof = length(index);

for i=1:edof

    ii=index(i);

    for j=1:edof

        jj=index(j);

        kk(ii,jj)=kk(ii,jj)+k(i,j);

    end

end

function [index]=feeldof(nd,nnel,ndof)

 k=0;

   for i=1:nnel

        start = (nd(i)-1)\*ndof;

        for j=1:ndof

              k=k+1;

              index(k)=start+j;

       end

   end

function [kinmtx2]=fekine2d(nnel,dhdx,dhdy)

 for i=1:nnel

       i1=(i-1)\*2+1;

       i2=i1+1;

       kinmtx2(1,i1)=dhdx(i);

       kinmtx2(2,i2)=dhdy(i);

       kinmtx2(3,i1)=dhdy(i);

       kinmtx2(3,i2)=dhdx(i);

 end

function [matmtrx]=fematiso(iopt,elastic,poisson)

if  iopt==1        % plane stress

 matmtrx= elastic/(1-poisson\*poisson)\* ...

    [1  poisson 0; ...

     poisson  1  0; ...

     0  0  (1-poisson)/2];

elseif   iopt==2        % plane strain

 matmtrx= elastic/((1+poisson)\*(1-2\*poisson))\* ...

   [(1-poisson)  poisson 0;

   poisson  (1-poisson)  0;

   0  0  (1-2\*poisson)/2];

 elseif  iopt==3     % axisymmetry

 matmtrx= elastic/((1+poisson)\*(1-2\*poisson))\* ...

   [(1-poisson)  poisson  poisson  0;

   poisson  (1-poisson)   poisson  0;

   poisson  poisson  (1-poisson)   0;

   0    0    0   (1-2\*poisson)/2];

elseif  iopt==4        % three-dimension

 matmtrx= elastic/((1+poisson)\*(1-2\*poisson))\* ...

   [(1-poisson)  poisson  poisson   0   0    0;

   poisson  (1-poisson)   poisson   0   0    0;

   poisson  poisson  (1-poisson)    0   0    0;

   0    0    0    (1-2\*poisson)/2   0    0;

   0    0    0    0    (1-2\*poisson)/2   0;

   0    0    0    0    0   (1-2\*poisson)/2];

end

return

first\_time=cputime;

format long

%--------------------------------------------

%input data for control parameters

%--------------------------------------------

lengthx=4;              %length of x-axis side of problem

lengthy=2;              %length of y-axis side of proble

emodule=1.0;            %elastic modulus

poisson=0.0;            %Poisson's ratio

fload=-1;             % the total load

lx=16;                  % number of element in x-axis

ly=8;                   % number of element in y-axis

nel=2\*lx\*ly;            % number of element

nnel=3;                 %number of nodes per element

ndof=2;                 %number of dofs per node

nnode=(lx+1)\*(ly+1);    %total number of nodes in system

sdof=nnode\*ndof;        %total system dofs

edof=nnel\*ndof;         %degrees of freedom per element

x0=[];

for i=1:lx+1

    for j=1:ly+1

          x0=[x0; (i-1)\*lengthx/lx      -0.5\*lengthy\*(1+(lx+1-i)/lx)\*(1-(j-1)/ly)];

    end

end

nodes=[];

for i=1:lx

    for j=1:ly

        nodes=[nodes; (ly+1)\*(i-1)+j (ly+1)\*i+j (ly+1)\*(i-1)+j+1;];

        nodes=[nodes; (ly+1)\*i+j (ly+1)\*i+j+1 (ly+1)\*(i-1)+j+1;];

    end

end

bcdof=[];

bcval=[];

for i=1:ly+1

        bcdof=[bcdof 1+2\*(i-1) 2+2\*(i-1)];

        bcval=[bcval  0   0];

end

ff=sparse(sdof,1);          %system force vector

k=sparse(edof,edof);        %initialization of element matrix

kk=sparse(sdof,sdof);       %system matrix

disp=sparse(sdof,1);        %system displacement vector

eldisp=sparse(edof,1);      %element displacement vector

stress=zeros(nel,3);        %matrix containing stress components

strain=zeros(nel,3);        %matrix containing strain components

index=sparse(edof,1);       %index vector

kinmtx=sparse(3,edof);      %kinematic matrix

matmtx=sparse(3,3);     %constitutive matrix

matmtx=fematiso(1,emodule,poisson);     %constitutive matrice

for iel=1:nel       %loop for the total number of element

    nd(1)=nodes(iel,1);         %1st connected node for (iel)-th element

    nd(2)=nodes(iel,2);         %2nd connected node for (iel)-th element

    nd(3)=nodes(iel,3);         %3rd connected node for (iel)-th element

    x1=x0(nd(1),1); y1=x0(nd(1),2);     %coord values of 1st node

    x2=x0(nd(2),1); y2=x0(nd(2),2);     %coord values of 2nd node

    x3=x0(nd(3),1); y3=x0(nd(3),2);     %coord values of 3rd node

    index=feeldof(nd,nnel,ndof);        %extract system dofs for the element

area=0.5\*(x1\*y2+x2\*y3+x3\*y1-x1\*y3-x2\*y1-x3\*y2); %area of triangula

    area2=area\*2;

     dhdx=(1/area2)\*[(y2-y3) (y3-y1) (y1-y2)];       %derivatives w.r.t x

     dhdy=(1/area2)\*[(x3-x2) (x1-x3) (x2-x1)];       %derivatives w.r.t y

    kinmtx2=fekine2d(nnel,dhdx,dhdy);               %kinematic matrice

    k=kinmtx2'\*matmtx\*kinmtx2\*area;              %element stiffness matrice

    kk=feasmb\_2(kk,k,index);                               %assemble element matrics

end

kk1=kk;

%--------------------------------------------------------------------------

%   force vector

%--------------------------------------------------------------------------

ff(sdof,1)=fload;

%------------------------

%  apply boundary condition

%------------------------

[kk,ff]=feaplyc2(kk,ff,bcdof,bcval);

%-------------------------

%  solve the matrix equation

%-------------------------

%disp=kk\ff;

[LL UU]=lu(kk);

utemp=LL\ff;

disp=UU\utemp;

EU=0.5\*disp'\*kk1\*disp;

%---------------------------------------

%  element stress computation

%---------------------------------------

energy=0;

for ielp=1:nel           % loop for the total number of elements

     nd(1)=nodes(ielp,1); % 1st connected node for (iel)-th element

     nd(2)=nodes(ielp,2); % 2nd connected node for (iel)-th element

     nd(3)=nodes(ielp,3); % 3rd connected node for (iel)-th element

    x1=x0(nd(1),1); y1=x0(nd(1),2);% coord values of 1st node

    x2=x0(nd(2),1); y2=x0(nd(2),2);% coord values of 2nd node

    x3=x0(nd(3),1); y3=x0(nd(3),2);% coord values of 3rd node

    xcentre=(x1+x2+x3)/3; ycentre=(y1+y2+y3)/3;

    index=feeldof(nd,nnel,ndof);% extract system dofs associated with element

    %-------------------------------------------------------

    %  extract element displacement vector

    %-------------------------------------------------------

   for i=1:edof

        eldisp(i)=disp(index(i));

    end

    area=0.5\*(x1\*y2+x2\*y3+x3\*y1-x1\*y3-x2\*y1-x3\*y2);  % area of triangule

    area2=area\*2;

    dhdx=(1/area2)\*[(y2-y3) (y3-y1) (y1-y2)];  % derivatives w.r.t. x-axis

    dhdy=(1/area2)\*[(x3-x2) (x1-x3) (x2-x1)];  % derivatives w.r.t. y-axis

    kinmtx2=fekine2d(nnel,dhdx,dhdy);          % compute kinematic matrix

    estrain=kinmtx2\*eldisp;             % compute strains

    estress=matmtx\*estrain;             % compute stresses

    for i=1:3

        strain(ielp,i)=estrain(i);          % store for each element

        stress(ielp,i)=estress(i);          % store for each element

    end

    energy=energy+0.5\*estrain'\*matmtx\*estrain\*area;

end

neigh\_node = cell(nnode,1);

indneigh=zeros(1,nnode);

for i=1:nel

    for j=1:3

        indneigh(nodes(i,j))=indneigh(nodes(i,j))+1;

        neigh\_node{nodes(i,j)}(indneigh(nodes(i,j)))=i;

    end

end

stress\_node=zeros(3,nnode);

for inode=1:nnode

      numel= indneigh(inode);

      for i=1:numel

            ind\_nel= neigh\_node{inode}(i);

           for j=1:3

                 stress\_node(j,inode)=stress\_node(j,inode)+stress(ind\_nel,j);

           end

    end

    stress\_node(:,inode)=stress\_node(:,inode)/numel;

end

%-------------------------------------------------------------

% Data output

%-------------------------------------------------------------

fid\_out=fopen('result\_beam01.plt','w');

fprintf(fid\_out,'TITLE="test case governed by poisson equation"\n');

fprintf(fid\_out,'VARIABLES="x" "y" "u" "v" "sigax"  "sigmay" "sigmaxy"\n');

fprintf(fid\_out,'ZONE T="flow-field", N= %8d,E=%8d,ET=TRIANGLE, F=FEPOINT\n',nnode,nel);

disp=full(disp);

for i=1:nnode

       fprintf(fid\_out,'%16.6e%16.6e%16.6e%16.6e%16.6e%16.6e%16.6e\n',x0(i,1),x0(i,2),disp(2\*i-1),disp(2\*i),stress\_node(1,i),stress\_node(2,i),stress\_node(3,i));

end

for i=1:nel

      fprintf(fid\_out,'%8d%8d%8d\n',nodes(i,1),nodes(i,2),nodes(i,3));

end

1. 线性四面体元

Main

clear

clc

format short

lengthx=2; %length of x-axis side of problem

lengthy=2; %length of y-axis side of proble

lengthz=2;

%h=0.2; %thickness of problem

elastic=2.1\*10^11; %elastic modulus

poisson=0.3; %Poisson's ratio

iopt=4;

Nodes = load("node.txt"); %total number of nodes in system

Load = load("load.txt")';

bdn = load("boundary.txt")';

elements = load("element.txt");

nnode=length(Nodes);%total number of nodes in system

nel=length(elements);% number of element

fload=[0,1250,0]; % the total load

%lx=16; % number of element in x-axis

%ly=16; % number of element in y-axis

%lz=16;

%nel=5\*0.75\*lx\*ly\*lz; % number of element

nnel=8; %number of nodes per element

ndof=3; %number of dofs per node

%nnode=(lx/2+1)\*(ly+1)\*(lz+1)+(lx/2)\*(ly+1)\*(lz/2+1); %total number of nodes in system

sdof=nnode\*ndof; %total system dofs

edof=nnel\*ndof; %degrees of freedom per element

B=zeros(6,12,nel);

K=zeros(sdof,sdof);

%Assemble tetrahedral units

for i=1:nel

element\_nodes\_index = elements(i,2:5);

four\_nodes\_matrix = Nodes(element\_nodes\_index,2:end);

x1=four\_nodes\_matrix(1,1);

y1=four\_nodes\_matrix(1,2);

z1=four\_nodes\_matrix(1,3);

x2=four\_nodes\_matrix(2,1);

y2=four\_nodes\_matrix(2,2);

z2=four\_nodes\_matrix(2,3);

x3=four\_nodes\_matrix(3,1);

y3=four\_nodes\_matrix(3,2);

z3=four\_nodes\_matrix(3,3);

x4=four\_nodes\_matrix(4,1);

y4=four\_nodes\_matrix(4,2);

z4=four\_nodes\_matrix(4,3);

[k, b,D]=TetrahedronElementStiffness(elastic,poisson,x1,y1,z1,x2,y2,z2,x3,y3,z3,x4,y4,z4);

B(:,:,i)= b;

for j = 1:4

row\_ = element\_nodes\_index(j);

for m = 1:4

col\_ = element\_nodes\_index(m);

K((3\*row\_-2):row\_\*3,(3\*col\_-2):col\_\*3) = ...

K((3\*row\_-2):row\_\*3,(3\*col\_-2):col\_\*3)...

+ k((3\*j-2):3\*j,(3\*m-2):3\*m);

end

end

end

F = zeros(sdof,1);

nLoad = length(Load);

for i = 1 :ndof

F((Load-1)\*ndof+i,1) = fload(i)/nLoad;

end

Constrain=constrain(bdn,ndof);

K\_constrain = K;

F\_constrain = F;

K\_constrain(Constrain,:) = [];

F\_constrain(Constrain,:) = [];

K\_constrain(:,Constrain) = [];

U\_ = K\_constrain\F\_constrain;

%Return node displacement

U=nodedisplacement(U\_,bdn);

%Post-processing calculation of cell node stress

snodes = zeros(nnode, 10);

xd = U(1:3:end);

yd = U(2:3:end);

zd = U(3:3:end);

for i = 1:nel

elements\_nodes\_index = elements(i,2:5);

u = zeros(12,1);

for index = 1:4

u((index-1)\*3+1:index\*3,:) = [xd(elements\_nodes\_index(index));...

yd(elements\_nodes\_index(index));...

zd(elements\_nodes\_index(index));];

end

sigma = (D\*B(:,:,i)\*u)';

for index = 1:4

node\_index = elements\_nodes\_index(index);

snodes(node\_index,1:6) = snodes(node\_index,1:6) + sigma;

snodes(node\_index, 10) = snodes(node\_index, 10) +1;

end

end

snodes(:,1:6) = snodes(:,1:6)./snodes(:,10);

for i = 1:nnode

sigma = snodes(i,1:6);

snodes(i,7:9) =TetrahedronElementPStresses(sigma);

end

figure1 = figure();

hold on;

%figure2 = figure();

scatter3(Nodes(:,2),Nodes(:,3),Nodes(:,4),20,snodes(:,1),'fill');

%figure3 = figure();

scatter3(Nodes(:,2),Nodes(:,3),Nodes(:,4),20,snodes(:,2),'fill');

%figure4 = figure();

scatter3(Nodes(:,2),Nodes(:,3),Nodes(:,4),20,snodes(:,3),'fill');

tetramesh(elements(:,2:5), Nodes(:,2:4));

hold on

%figure5 = figure();

scatter3(Nodes(:,2),Nodes(:,3),Nodes(:,4),"g.");

axis equal;

%绘制边界节点

Boundary\_nodes\_coor = Nodes(bdn,2:4);

scatter3(Boundary\_nodes\_coor(:,1),Boundary\_nodes\_coor(:,2),Boundary\_nodes\_coor(:,3),"black\*");

%绘制加载节点

Load\_nodes\_coor = Nodes(Load,2:4);

scatter3(Load\_nodes\_coor(:,1),Load\_nodes\_coor(:,2),Load\_nodes\_coor(:,3),"yo");

view([45,45])

function[K,B,D]=TetrahedronElementStiffness(E,NU,x1,y1,z1,x2,y2,z2,x3,y3,z3,x4,y4,z4)

xyz=[1 x1 y1 z1;1 x2 y2 z2;1 x3 y3 z3;1 x4 y4 z4];

V=det(xyz)/6;

mbeta1=[1 y2 z2;1 y3 z3;1 y4 z4];

mbeta2=[1 y1 z1;1 y3 z3;1 y4 z4];

mbeta3=[1 y1 z1;1 y2 z2;1 y4 z4];

mbeta4=[1 y1 z1;1 y2 z2;1 y3 z3];

mgamma1=[1 x2 z2;1 x3 z3;1 x4 z4];

mgamma2=[1 x1 z1;1 x3 z3;1 x4 z4];

mgamma3=[1 x1 z1;1 x2 z2;1 x4 z4];

mgamma4=[1 x1 z1;1 x2 z2;1 x3 z3];

mdelta1=[1 x2 y2;1 x3 y3;1 x4 y4];

mdelta2=[1 x1 y1;1 x3 y3;1 x4 y4];

mdelta3=[1 x1 y1;1 x2 y2;1 x4 y4];

mdelta4=[1 x1 y1;1 x2 y2;1 x3 y3];

beta1=-1\*det(mbeta1);

beta2=det(mbeta2);

beta3=-1\*det(mbeta3);

beta4=det(mbeta4);

gamma1=det(mgamma1);

gamma2=-1\*det(mgamma2);

gamma3=det(mgamma3);

gamma4=-1\*det(mgamma4);

delta1=-1\*det(mdelta1);

delta2=det(mdelta2);

delta3=-1\*det(mdelta3);

delta4=det(mdelta4);

B1=[beta1 0 0;0 gamma1 0;0 0 delta1;gamma1 beta1 0;0 delta1 gamma1;delta1 0 beta1];

B2=[beta2 0 0;0 gamma2 0;0 0 delta2;gamma2 beta2 0;0 delta2 gamma2;delta2 0 beta2];

B3=[beta3 0 0;0 gamma3 0;0 0 delta3;gamma3 beta3 0;0 delta3 gamma3;delta3 0 beta3];

B4=[beta4 0 0;0 gamma4 0;0 0 delta4;gamma4 beta4 0;0 delta4 gamma4;delta4 0 beta4];

B=[B1 B2 B3 B4]/(6\*V);

D=(E/((1+NU)\*(1-2\*NU)))\*[1-NU NU NU 0 0 0;NU 1-NU NU 0 0 0;NU NU 1-NU 0 0 0;0 0 0 (1-2\*NU)/2 0 0;0 0 0 0 (1-2\*NU)/2 0;0 0 0 0 0 (1-2\*NU)/2];

K=V\*B'\*D\*B;

function y=TetrahedronElementPStresses(sigma)

s1=sigma(1)+sigma(2)+sigma(5);

s2=sigma(1)\*sigma(2)+sigma(1)\*sigma(3)+sigma(2)\*sigma(3)-sigma(4)\*sigma(4)-sigma(5)\*sigma(5)-sigma(6)\*sigma(6);

ms3=[sigma(1) sigma(4) sigma(6);sigma(4) sigma(2) sigma(5);sigma(6) sigma(5) sigma(3)];

s3=det(ms3);

y=[s1;s2;s3];

function U=nodedisplacement(U\_,Boundary\_nodes)

U = U\_;

for i = 1:length(Boundary\_nodes)

    index = Boundary\_nodes(i);

    forward\_ = U(1:(index-1)\*3,:);

    backward\_ = U((index-1)\*3+1:end,:);

    U = [forward\_;0;0;0;backward\_];

end

function cons=constrain(Boundary\_nodes,ndof)

con\_dofs = zeros(length(Boundary\_nodes),3);

for i = 1:ndof

    con\_dofs(:,i) = (Boundary\_nodes-1)\*ndof+i;

end

switch ndof

    case 1

        cons = con\_dofs(:,1);

    case 2

        cons = [con\_dofs(:,1);con\_dofs(:,2)];

    case 3

        cons = [con\_dofs(:,1);con\_dofs(:,2);con\_dofs(:,3)];

end