六面体等参单元有限元分析的 matlab 实现

一、问题概述

本问题对下图所示 3D 等强度悬臂梁模型进行有限元分析，采用八节点六面体等参元进行求解域离散，通过 matlab 语言编程求解得出该模型受力后的位移场、应变场和应力场，并使用商业软件 solidworks 建模、有限元分析，最后通过比较两种方法的计算差距以此来验证所编写代码的正确性以及分析误差产生的原因。

。

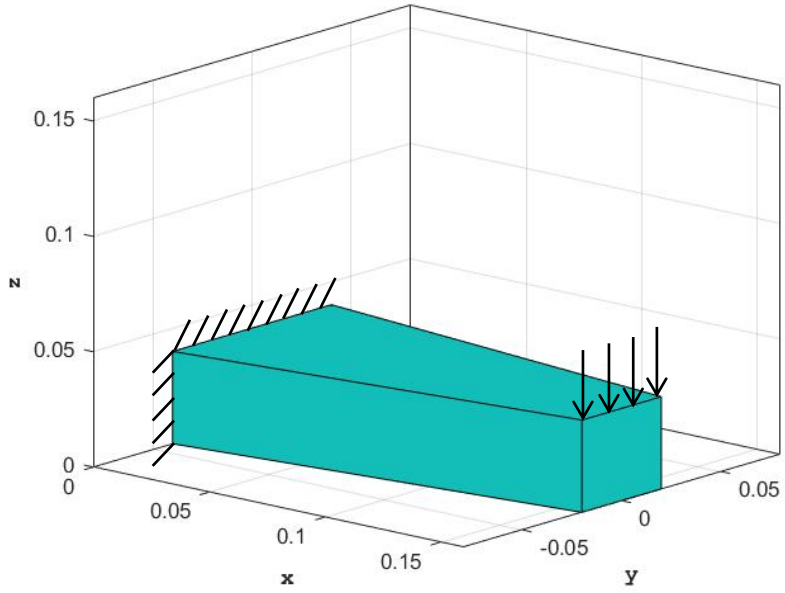


图 1 算例几何

考虑一个左端固定的悬臂梁如图 1 所示，其几何参数为：底面梯形高为 160mm，下底为 80mm，上底为 40mm，悬臂梁厚度为 40mm；材料参数为：杨氏模量 E=2.1e11Mpa，泊松比 μ=0.3；载荷情况为右端有一个分布载荷 f = 2F/ly N/m (F = −1000N)，求解得出该梁受力变形后的位移、应变和应力。

二、程序实现

2.1 网格划分

如图 2 所示，等分梯形 x 方向、y 方向长度，按图 2 所示的排列方式从左到右依次给各节点编号；图 3 表示第一个单元的节点信息，由第一层节点 1、2、13、12 和第二层节点 452、453、464、463 组成。

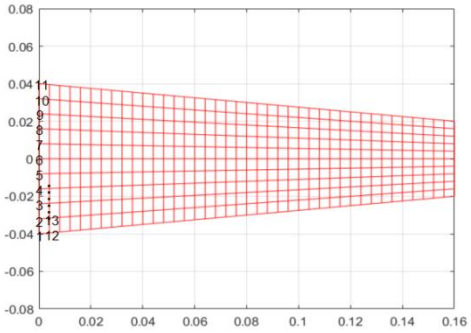


图 2 节点坐标图

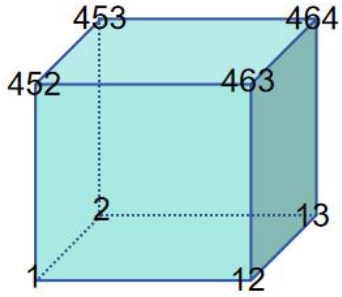


图 3 单元节点图

```
%等分梯形x方向、y方向长度以及悬臂梁厚度z方向长度
lengthx=0.16; %底面梯形高
lengthy=0.08; %梯形下底
lengthz=0.04; %悬臂梁厚度
lx=40; % x方向的单元数量
ly=10; % y方向的单元数量
lz=10; % z方向的单元数量
Element_number=lx*ly*lz; % 单元总数
No_nel=8; % 每个单元的节点数量
No_dof=3; % 每个节点的自由度
Node_number=(lx+1)*(ly+1)*(lz+1); % 节点总数
Kz=zeros(No_dof*Node_number,No_dof*Node_number);
ff=zeros(No_dof*Node_number,1);
uu=zeros(No_dof*Node_number,1);
E=2.1e11; % 弹性模量
NU=0.3; % 泊松比
%--------------------------------------------------------------
% 节点坐标信息，表示节点x、y、z方向的坐标
%--------------------------------------------------------------
g=1;
```

```
for hh=1:lz+1
    for i=1:lx+1
        e=1;h=2;
        for j=1:ly+1
            if e<=ly/2+1
                gcoord(g,1)=(i-1)*lengthx/lx;
                gcoord(g,2)=-0.5*lengthy/2*(1+(lx+1-i)/lx)*(1-(j-1)/(ly/2));
                gcoord(g,3)=(hh-1)*lengthz/lz;
                g=g+1;e=e+1;
            else
                gcoord(g,1)=(i-1)*lengthx/lx;
                gcoord(g,2)=-gcoord((i-1)*(ly+1)+e-h,2);
                gcoord(g,3)=(hh-1)*lengthz/lz;
                g=g+1;e=e+1;h=h+2;
            end
        end
    end
end
%-----------------------------------------------------------
% 单元节点信息，表示每个单元上的节点号码
%-----------------------------------------------------------
No_element=0;
for loopk=1:lz
    for loopi=1:lx
        for loopj=1:ly
            No_element=No_element+1;
            nodes(No_element,1)=(loopi-1)*(ly+1)+loopj+(loopk-1)*(lx+1)*(ly+1);
            nodes(No_element,2)=(loopi-1)*(ly+1)+loopj+1+(loopk-1)*(lx+1)*(ly+1);
            nodes(No_element,3)=(loopi-1)*(ly+1)+loopj+ly+2+(loopk-1)*(lx+1)*(ly+1);
            nodes(No_element,4)=(loopi-1)*(ly+1)+loopj+ly+1+(loopk-1)*(lx+1)*(ly+1);
            nodes(No_element,5)=(loopi-1)*(ly+1)+loopj+loopk*(lx+1)*(ly+1);
            nodes(No_element,6)=(loopi-1)*(ly+1)+loopj+1+loopk*(lx+1)*(ly+1);
            nodes(No_element,7)=(loopi-1)*(ly+1)+loopj+ly+2+loopk*(lx+1)*(ly+1);
            nodes(No_element,8)=(loopi-1)*(ly+1)+loopj+ly+1+loopk*(lx+1)*(ly+1);
        end
    end
end
```

网格划分情况如图 4 所示：

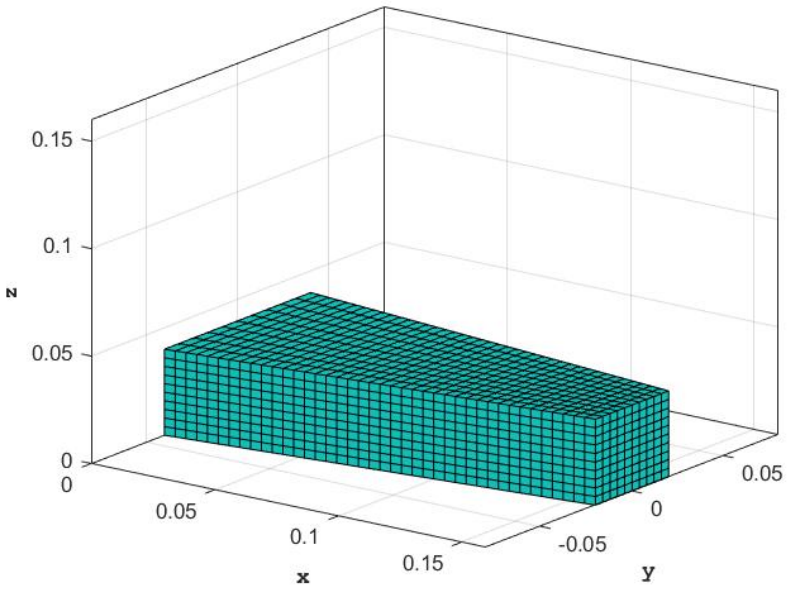


图 4 网格划分情况

2.2 施加边界条件和载荷

本算例进行网格划分后，受约束的节点为每层最左端，令这些节点的 x、y 和 z 方向自由度的位移值为 0，并在右上端节点处施加 z 方向的分布载荷 $f = 2F/ly\ ^{N}/_{m}$ ($F = -1000N$)，如图 5 所示。

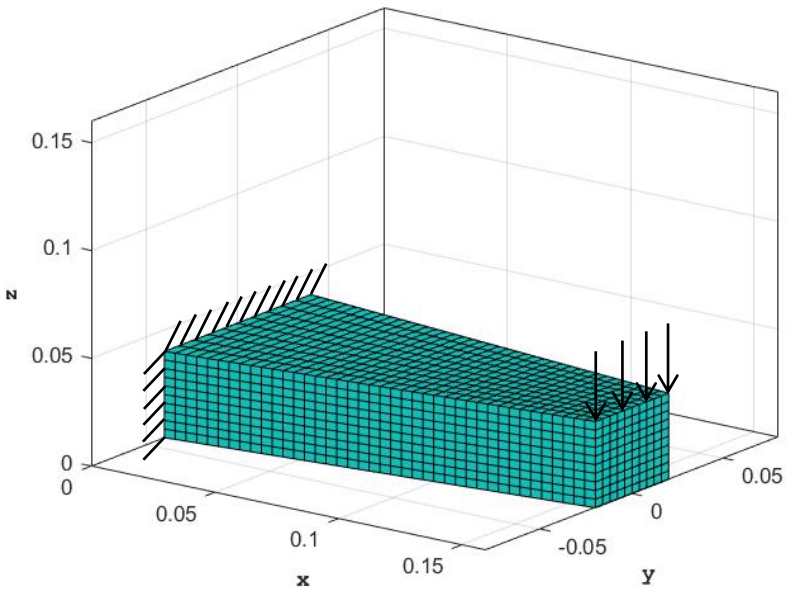


图 5 约束及载荷情况

```
% 施加边界条件
for hh=1:lz+1
    uu(1+3*(lx+1)*(ly+1)*(hh-1):3*(ly+1)+3*(lx+1)*(ly+1)*(hh-1),1)=0;
end
% 施加载荷
ff=zeros(3*(lz+1)*(lx+1)*(ly+1),1);
hhh=1;
for i=1:ly+1
    ff(3+3*(hhh-1)+3*lx*(ly+1)+3*lz*(lx+1)*(ly+1),1)=-1000/(ly+1);
    hhh=hhh+1;
end
```

2.3 刚度矩阵建立

```
No_element=0;
for loopk=1:lz
    for loopi=1:lx
        for loopj=1:ly
            No_element=No_element+1;
            x1=gcoord(nodes(No_element,1),1);y1=gcoord(nodes(No_element,1),2);
            z1=gcoord(nodes(No_element,1),3);
            x2=gcoord(nodes(No_element,2),1);y2=gcoord(nodes(No_element,2),2);
            z2=gcoord(nodes(No_element,2),3);
            x3=gcoord(nodes(No_element,3),1);y3=gcoord(nodes(No_element,3),2);
            z3=gcoord(nodes(No_element,3),3);
            x4=gcoord(nodes(No_element,4),1);y4=gcoord(nodes(No_element,4),2);
            z4=gcoord(nodes(No_element,4),3);
            x5=gcoord(nodes(No_element,5),1);y5=gcoord(nodes(No_element,5),2);
            z5=gcoord(nodes(No_element,5),3);
            x6=gcoord(nodes(No_element,6),1);y6=gcoord(nodes(No_element,6),2);
            z6=gcoord(nodes(No_element,6),3);
            x7=gcoord(nodes(No_element,7),1);y7=gcoord(nodes(No_element,7),2);
            z7=gcoord(nodes(No_element,7),3);
            x8=gcoord(nodes(No_element,8),1);y8=gcoord(nodes(No_element,8),2);
            z8=gcoord(nodes(No_element,8),3);
            ke=Hexahedra1_3D8Node_Stiffness(E,NU,x1,y1,z1,x2,y2,z2,x3,y3,z3,x4,y4,z4
                                            ,x5,y5,z5,x6,y6,z6,x7,y7,z7,x8,y8,z8);
            j1=nodes(No_element,1);j2=nodes(No_element,2);j3=nodes(No_element,3);
            j4=nodes(No_element,4);j5=nodes(No_element,5);j6=nodes(No_element,6);
            j7=nodes(No_element,7);j8=nodes(No_element,8);
```

```
            Kz=Hexahedra1_3D8Node_Assembly(Kz,ke, j1, j2, j3, j4, j5, j6, j7, j8);
        end
    end
end
```

刚度矩阵的建立由上述程序完成，主要完成形函数及其导数的计算、雅可比矩阵及其行列式的计算、应变矩阵 B 的计算、高斯积分、单刚的建立和总刚的组装等，其中各子程序的功能及代码在注释中有说明，鉴于篇幅的原因，子程序的具体代码放在附录中。

2.4 边界条件处理及方程求解

```
% 零位移边界条件处理：对角线元素改1法
for hh=1:lz+1
    uu(1+3*(lx+1)*(ly+1)*(hh-1):3*(ly+1)+3*(lx+1)*(ly+1)*(hh-1),1)=0;
    Kz(1+3*(lx+1)*(ly+1)*(hh-1):3*(ly+1)+3*(lx+1)*(ly+1)*(hh-1),:)=0;
    Kz(:,1+3*(lx+1)*(ly+1)*(hh-1):3*(ly+1)+3*(lx+1)*(ly+1)*(hh-1))=0;
    for i=1:3*(ly+1)
        Kz(i+3*(lx+1)*(ly+1)*(hh-1),i+3*(lx+1)*(ly+1)*(hh-1))=1;
    end
end
ff=zeros(3*(lz+1)*(lx+1)*(ly+1),1);
hhh=1;
for i=1:ly+1
    ff(3+3*(hhh-1)+3*lx*(ly+1)+3*lz*(lx+1)*(ly+1),1)=-1000/(ly+1);
    hhh=hhh+1;
end
% 方程求解
uu=Kz\ff;                                   % 得到节点位移
```

通过对角线元素该 1 法修改刚度矩阵和整体载荷向量，将约束节点的载荷设为 0，将约束自由度所在刚度矩阵的主元设为 1，同行其余元素设为 0，使总体刚度矩阵非奇异方程可解。总体刚度矩阵 kz 是呈三对角线带状分布的，uu=Kz\ff，最终求解得出节点位移 uu。

2.5 应变、应力求解

```
% 单元应变、应力
No_element=0;
for loopk=1:lz
```

```
    for loopi=1:lx
        for loopj=1:ly
            No_element=No_element+1;
            u=zeros(24,1);
            x1=gcoord(nodes(No_element,1),1);y1=gcoord(nodes(No_element,1),2);
            z1=gcoord(nodes(No_element,1),3);
            x2=gcoord(nodes(No_element,2),1);y2=gcoord(nodes(No_element,2),2);
            z2=gcoord(nodes(No_element,2),3);
            x3=gcoord(nodes(No_element,3),1);y3=gcoord(nodes(No_element,3),2);
            z3=gcoord(nodes(No_element,3),3);
            x4=gcoord(nodes(No_element,4),1);y4=gcoord(nodes(No_element,4),2);
            z4=gcoord(nodes(No_element,4),3);
            x5=gcoord(nodes(No_element,5),1);y5=gcoord(nodes(No_element,5),2);
            z5=gcoord(nodes(No_element,5),3);
            x6=gcoord(nodes(No_element,6),1);y6=gcoord(nodes(No_element,6),2);
            z6=gcoord(nodes(No_element,6),3);
            x7=gcoord(nodes(No_element,7),1);y7=gcoord(nodes(No_element,7),2);
            z7=gcoord(nodes(No_element,7),3);
            x8=gcoord(nodes(No_element,8),1);y8=gcoord(nodes(No_element,8),2);
            z8=gcoord(nodes(No_element,8),3);
            xx(:,No_element)=[x1;x2;x3;x4;x5;x6;x7;x8];
            yy(:,No_element)=[y1;y2;y3;y4;y5;y6;y7;y8];
            zz(:,No_element)=[z1;z2;z3;z4;z5;z6;z7;z8];
            m=nodes(No_element,1);l=nodes(No_element,2);j=nodes(No_element,3);
            i=nodes(No_element,4);q=nodes(No_element,5);p=nodes(No_element,6);
            o=nodes(No_element,7);n=nodes(No_element,8);
            u(1)=uu((m-1)*3+1);u(2)=uu((m-1)*3+2);u(3)=uu((m-1)*3+3);
            u(4)=uu((l-1)*3+1);u(5)=uu((l-1)*3+2);u(6)=uu((l-1)*3+3);
            u(7)=uu((j-1)*3+1);u(8)=uu((j-1)*3+2);u(9)=uu((j-1)*3+3);
            u(10)=uu((i-1)*3+1);u(11)=uu((i-1)*3+2);u(12)=uu((i-1)*3+3);
            u(13)=uu((q-1)*3+1);u(14)=uu((q-1)*3+2);u(15)=uu((q-1)*3+3);
            u(16)=uu((p-1)*3+1);u(17)=uu((p-1)*3+2);u(18)=uu((p-1)*3+3);
            u(19)=uu((o-1)*3+1);u(20)=uu((o-1)*3+2);u(21)=uu((o-1)*3+3);
            u(22)=uu((n-1)*3+1);u(23)=uu((n-1)*3+2);u(24)=uu((n-1)*3+3);
            [strain,stress]=Hexahedral_3D8Node_Stress(E,NU,x1,y1,z1,x2,y2,z2,x3,y3,
                            z3,x4,y4,z4,x5,y5,z5,x6,y6,z6,x7,y7,z7,x8,y8,z8,u);
            P_(No_element,:)=Principal(strain(1),strain(2),strain(3),...
                                       strain(4),strain(5),strain(6));
            P(No_element,:)=Principal(stress(1),stress(2),stress(3),...
                                      stress(4),stress(5),stress(6));
            Stress(:,No_element)=stress;
            Strain(:,No_element)=strain;
        end
    end
```

```
end
```

使用单元中心处应变、应力值代替单元应变、应力，对于同一单元，各节点应变、应力值相同。

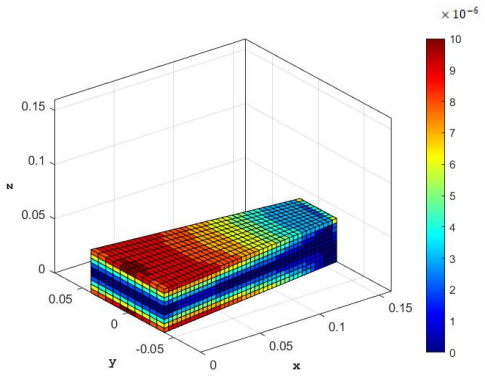
2.5 后处理

```
%合位移云图
for ii=1:Element_number
    m=nodes(ii,1);l=nodes(ii,2);j=nodes(ii,3);i=nodes(ii,4);
    q=nodes(ii,5);p=nodes(ii,6);o=nodes(ii,7);n=nodes(ii,8);
    hsv(1)=uu((m-1)*3+1);hsv(2)=uu((m-1)*3+2);hsv(3)=uu((m-1)*3+3);
    hsv(4)=uu((l-1)*3+1);hsv(5)=uu((l-1)*3+2);hsv(6)=uu((l-1)*3+3);
    hsv(7)=uu((j-1)*3+1);hsv(8)=uu((j-1)*3+2);hsv(9)=uu((j-1)*3+3);
    hsv(10)=uu((i-1)*3+1);hsv(11)=uu((i-1)*3+2);hsv(12)=uu((i-1)*3+3);
    hsv(13)=uu((q-1)*3+1);hsv(14)=uu((q-1)*3+2);hsv(15)=uu((q-1)*3+3);
    hsv(16)=uu((p-1)*3+1);hsv(17)=uu((p-1)*3+2);hsv(18)=uu((p-1)*3+3);
    hsv(19)=uu((o-1)*3+1);hsv(20)=uu((o-1)*3+2);hsv(21)=uu((o-1)*3+3);
    hsv(22)=uu((n-1)*3+1);hsv(23)=uu((n-1)*3+2);hsv(24)=uu((n-1)*3+3);
    vertices_matrix = [xx(:,ii),yy(:,ii),zz(:,ii)];
    cc=1;
    hsv=[sqrt(hsv(cc)^2+hsv(cc+1)^2+hsv(cc+2)^2);
    sqrt(hsv(cc+3)^2+hsv(cc+3+1)^2+hsv(cc+3+2)^2);
    sqrt(hsv(cc+6)^2+hsv(cc+6+1)^2+hsv(cc+6+2)^2);
    sqrt(hsv(cc+9)^2+hsv(cc+9+1)^2+hsv(cc+9+2)^2);
    sqrt(hsv(cc+12)^2+hsv(cc+12+1)^2+hsv(cc+12+2)^2);
    sqrt(hsv(cc+15)^2+hsv(cc+15+1)^2+hsv(cc+15+2)^2);
    sqrt(hsv(cc+18)^2+hsv(cc+18+1)^2+hsv(cc+18+2)^2);
    sqrt(hsv(cc+21)^2+hsv(cc+21+1)^2+hsv(cc+21+2)^2)];
    max_x(ii)=max(hsv);min_n(ii)=min(hsv);
    faces_matrix=[1 2 6 5; 2 3 7 6;3 4 8 7;4 1 5 8;1 2 3 4;5 6 7 8];
    h = patch('Vertices',vertices_matrix,'Faces',faces_matrix,...
              'FaceVertexCData',hsv, 'FaceColor','interp');
    view(3)
    xlabel('x', 'FontName', 'courier new', 'FontWeight', 'bold');
    ylabel('y', 'FontName', 'courier new', 'FontWeight', 'bold');
    zlabel('z', 'FontName', 'courier new', 'FontWeight', 'bold');
    grid on;
    box on;
    colormap jet
    hold on;
```

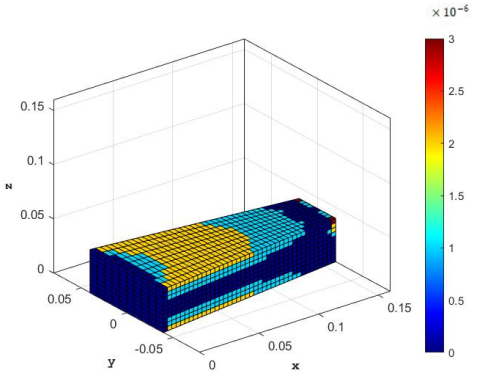
```
    colorbar('ylim',[min(min_n) max(max_x)],'ytick',[min(min_n):0.15e-5:max(max_x)])
    title('合位移云图');
end
axis([0 0.16 -0.08 0.08 0 0.16])
% Vonmisers应力图
for i=1:Element_number
    vertices_matrix = [xx(:,i),yy(:,i),zz(:,i)];
    hsv = ones(8,1)*
          (((P(i,1)-P(i,2))^2+(P(i,2)-P(i,3))^2+(P(i,3)-P(i,1))^2)/2)^0.5;
    faces_matrix = [1 2 6 5; 2 3 7 6;3 4 8 7;4 1 5 8;1 2 3 4;5 6 7 8];
    h = patch('Vertices',vertices_matrix,'Faces',faces_matrix,...
              'FaceVertexCData',hsv, 'FaceColor','interp');
    view(3)
    xlabel('x', 'FontName', 'courier new', 'FontWeight', 'bold');
    ylabel('y', 'FontName', 'courier new', 'FontWeight', 'bold');
    zlabel('z', 'FontName', 'courier new', 'FontWeight', 'bold');
    grid on;
    box on;
    hold on;
    colorbar
    colormap jet
    title('Vonmisers应力图');
end
axis([0 0.16 -0.08 0.08 0 0.16])
%应变云图
for j=1:6
    for i=1:Element_number
        vertices_matrix = [xx(:,i),yy(:,i),zz(:,i)];
        hsv = ones(8,1)*(floor(abs(Strain(j,i))));
        faces_matrix = [1 2 6 5; 2 3 7 6;3 4 8 7;4 1 5 8;1 2 3 4;5 6 7 8];
        h = patch('Vertices',vertices_matrix,'Faces',faces_matrix,...
                  'FaceVertexCData',hsv, 'FaceColor','interp');
        view(3)
        xlabel('x', 'FontName', 'courier new', 'FontWeight', 'bold');
        ylabel('y', 'FontName', 'courier new', 'FontWeight', 'bold');
        zlabel('z', 'FontName', 'courier new', 'FontWeight', 'bold');
        grid on;
        box on;
        hold on;
        colorbar
        colormap jet
        title('应变云图');
    end
    axis([0 0.16 -0.08 0.08 0 0.16])
```

```
end
%应力云图
for j=1:6
    for i=1:Element_number
        vertices_matrix = [xx(:,i),yy(:,i),zz(:,i)];
        hsv = ones(8,1)*(floor(abs(Stress(j,i))));
        faces_matrix = [1 2 6 5; 2 3 7 6;3 4 8 7;4 1 5 8;1 2 3 4;5 6 7 8];
        h = patch('Vertices',vertices_matrix,'Faces',faces_matrix,
                  'FaceVertexCData',hsv, 'FaceColor','interp');
        view(3)
        xlabel('x', 'FontName', 'courier new', 'FontWeight', 'bold');
        ylabel('y', 'FontName', 'courier new', 'FontWeight', 'bold');
        zlabel('z', 'FontName', 'courier new', 'FontWeight', 'bold');
        grid on;
        box on;
        hold on;
        colorbar
        colormap jet
        title('应力云图');
    end
    axis([0 0.16 -0.08 0.08 0 0.16])
end
```

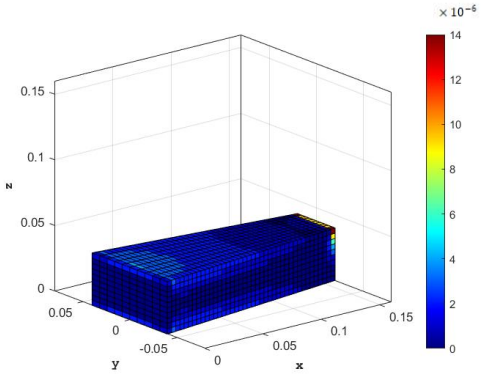
以上程序实现计算数据的可视化绘图，子函数具体代码及注释见附件，程序所有的计算结果均输出到了程序文件夹下的 result.mat 文件中，以下是六面体等参单元各个方向的应变、应力云图。



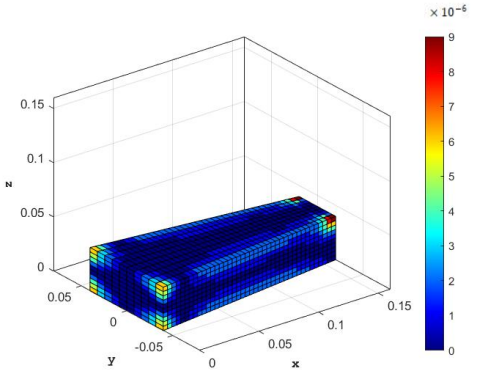
ε_x



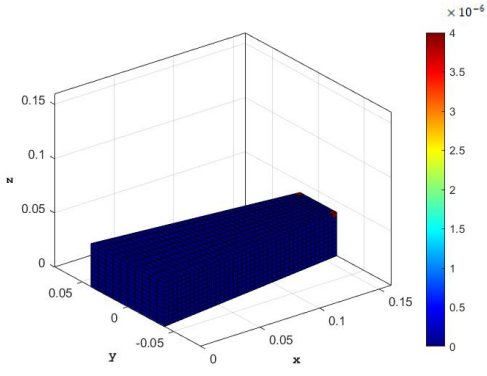
ε_y



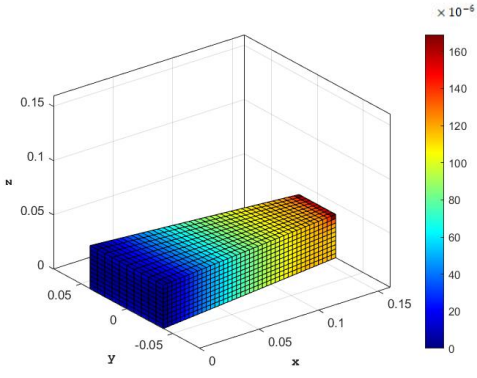
ε_z



ε_{xy}

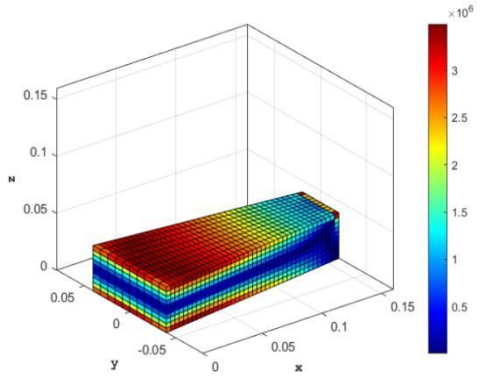


ε_{yz}

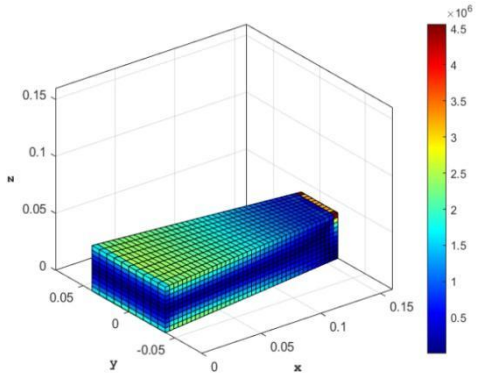


ε_{xz}

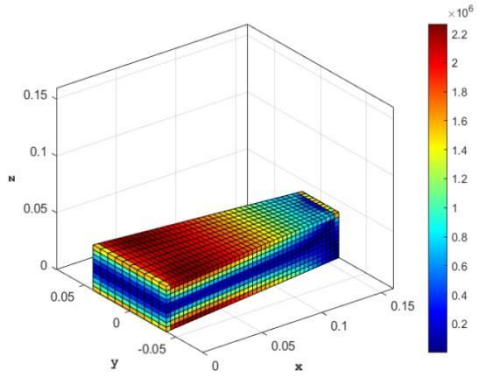
图 6 应变云图



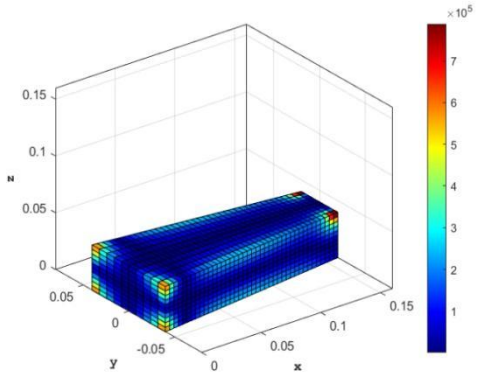
σ_x



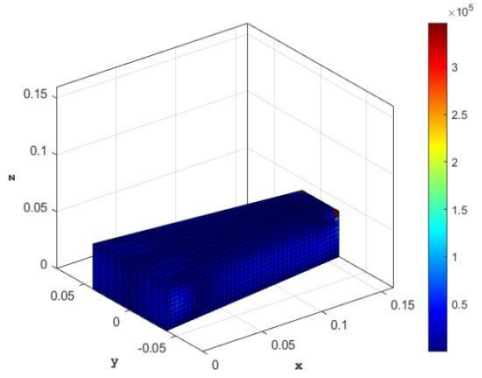
σ_y



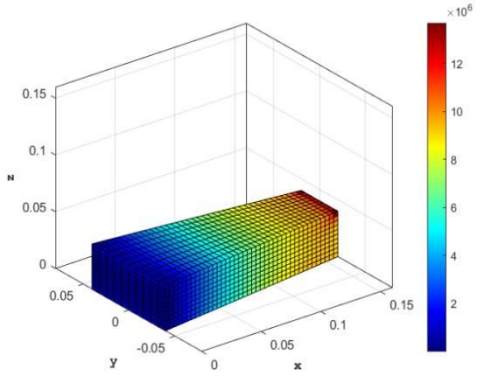
σ_z



σ_{xy}



σ_{yz}



σ_{xz}

图 7 应力云图

三、求解结果分析

3.1 计算结果对比

将程序求解结果与软件计算结果进行对比，下述左图为 matlab 程序求解结果，右图为 solidworks 软件求解结果。

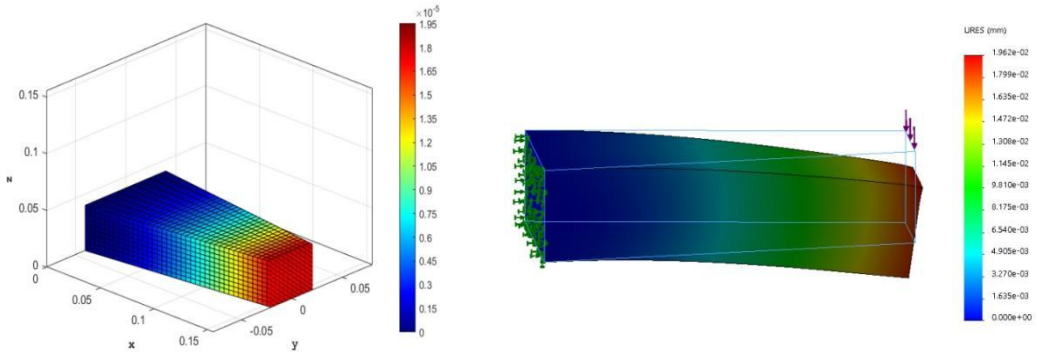


图 8 总位移结果对比

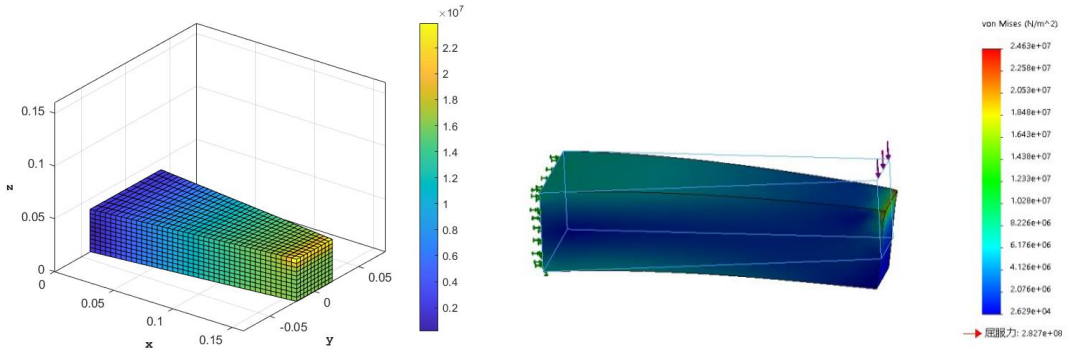


图 9 Vonmisers 应力结果对比

由云图可知，两种方法求得的云图形状相近。通过 matlab 计算的最大总位移为$1.95 \times 10^{-5}\mathrm{m}$，Vonmisers 最大应力为$2.3901 \times 10^{7}\mathrm{pa}$，而通过 solidworks 计算的最大总位移为$1.962 \times 10^{-5}\mathrm{m}$，Vonmisers 最大应力为$2.463 \times 10^{7}\mathrm{pa}$，最大位移相对计算误差为 0.61%；最大 Vonmisers 应力的相对计算误差为 2.96%。位移的求解精度较高，应力求解误差比较大，但仍在误差之内，从而证明所编写代码的正确性。

3.2 误差原文分析

经过查阅资料，产生误差的原因可能主要受因为六面体 8 节点等参元缺少边中点，不能很好的适应等强度悬臂梁的弯曲变形，加之单元的划分比较少，网格解存在一定的误差。此外，在应力求解的过程中，使用单元中心处应力值代替单元应力，从而导致应力误差相对较大。由于六面体等参元在高斯点处的应力值精度较高，下一步将尝试通过应力插值外推，先求出高斯点处的应力再乘以高斯点处型函数矩阵的逆从而得到改进的节点应力值。

附录

```
function Ke=Hexahedra1_3D8Node_Stiffness(E,NU,x1,y1,z1,x2,y2,z2,x3,y3,z3,x4,y4,z4,
                                         x5,y5,z5,x6,y6,z6,x7,y7,z7,x8,y8,z8)
Loc=[x4 y4 z4 ; x3 y3 z3;x2 y2 z2;x1 y1 z1;x8 y8 z8;x7 y7 z7;x6 y6 z6; x5 y5 z5];
gsx=[-0.7745966692 0 0.7745966692];%高斯求积坐标点与系数
gsw=[0.55555555556 0.888888888889 0.55555555556];
Ke=zeros(24,24);
for ii=1:3 %三维高斯求积
    sx=gsx(ii);
    sw=gsw(ii);
    for jj=1 :3
        nx=gsx(jj);
        nw=gsw(jj);
        for kk=1:3
            tx=gsx(kk);
            tw=gsw(kk);
            Ke=Ke+(sw*nw*tw*BDca1c(sx, nx,tx,Loc,E,NU));
        end
    end
end
end
```

```
function BD=BDca1c(s,n,t,Loc,E,NU)
N1=(1+s)*(1-t)*(1-n)/8; N2=(1+s)*(1+t)*(1-n)/8; N3=(1-s)*(1+t)*(1-n)/8;
N4=(1-s)*(1-t)*(1-n)/8; N5=(1+s)*(1-t)*(1+n)/8; N6=(1+s)*(1+t)*(1+n)/8;
N7=(1-s)*(1+t)*(1+n)/8; N8=(1-s)*(1-t)*(1+n)/8;
dNsnt=[(1-n)*(1-t)/8,-(1+s)*(1-n)/8,-(1+s)*(1-t)/8;
       (1-n)*(1+t)/8,(1+s)*(1-n)/8,-(1+s)*(1+t)/8;
       -(1-n)*(1+t)/8,(1-s)*(1-n)/8,-(1-s)*(1+t)/8;
       -(1-n)*(1-t)/8,-(1-s)*(1-n)/8,-(1-s)*(1-t)/8;
       (1+n)*(1-t)/8, -(1+s)*(1+n)/8, (1+s)*(1-t)/8;
       (1+n)*(1+t)/8,(1+s)*(1+n)/8,(1+s)*(1+t)/8;
       -(1+n)*(1+t)/8, (1-s)*(1+n)/8,(1-s)*(1+t)/8;
       -(1+n)*(1-t)/8, -(1-s)*(1+n)/8,(1-s)*(1-t)/8;];
dNsnt1=dNsnt';
J=dNsnt1*Loc;
detJ=det(J) ;
% 定义B矩阵的系数
a=J(2,2)*J(3,3)-J(2,3)*J(3,2); b=J(1,2)*J(3,3)-J(1,3)*J(3,2);
c=J(1,2)*J(2,3)-J(1,3)*J(2,2); d=J(2,1)*J(3,3)-J(2,3)*J(3,1);
e=J(1,1)*J(3,3)-J(1,3)*J(3,1); f=J(1,1)*J(2,3)-J(1,3)*J(2,1);
g=J(2,1)*J(3,2)-J(2,2)*J(3,1); h=J(1,1)*J(3,2)-J(1,2)*J(3,1);
l=J(1,1)*J(2,2)-J(1,2)*J(2,1);
% 通过循环计算各个矩阵
```

```
Bs=zeros(6,3,8);
for i=1:8
    Bs(:,:,i)=[a*dNsnt(i,1)-b*dNsnt(i,2)+c*dNsnt(i,3),0,0;
               0,-d*dNsnt(i,1)+e*dNsnt(i,2)-f*dNsnt(i,3),0;
               0,0,g*dNsnt(i,1)-h*dNsnt(i,2)+l*dNsnt(i,3);
               -d*dNsnt(i,1)+e*dNsnt(i,2)-f*dNsnt(i,3),...
               a*dNsnt(i,1)-b*dNsnt(i,2)+c*dNsnt(i,3),0;
               0,g*dNsnt(i,1)-h*dNsnt(i,2)+l*dNsnt(i,3),...
               -d*dNsnt(i,1)+e*dNsnt(i,2)-f*dNsnt(i,3);
               g*dNsnt(i,1)-h*dNsnt(i,2)+l*dNsnt(i,3),0,...
               a*dNsnt(i,1)-b*dNsnt(i,2)+c*dNsnt(i,3)]/detJ;
end
% 计算 B 矩阵
B=[Bs(:,:,1),Bs(:,:,2),Bs(:,:,3),Bs(:,:,4),...
  Bs(:,:,5),Bs(:,:,6),Bs(:,:,7),Bs(:,:,8)];
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;
                         0,0,0,0.5-NU,0,0;0,0,0,0,0.5-NU,0;0,0,0,0,0,0.5-NU];
BD=detJ*transpose(B)*D*B;
end
```

```
function Kz=Hexahedra1_3D8Node_Assemb1y(Kz,ke, j1, j2, j3, j4, j5, j6, j7, j8)
K1oc=[3*j4-2:3*j4,3*j3-2:3*j3, ...
      3*j2-2:3*j2,3*j1-2:3*j1,...
      3*j8-2:3*j8,3*j7-2:3*j7, ...
      3*j6-2:3*j6,3*j5-2:3*j5];
for ii=1:24 %把元素放入整体矩阵中的对应位置
    for jj=1:24
        Kz(K1oc (ii), K1oc (jj))=Kz(K1oc(ii), K1oc(jj))+ke(ii, jj);
    end
end
end
```

```
function [strain,stress]=Hexahedra1_3D8Node_Stress(E,NU,x1,y1,z1,x2,y2,z2,x3,y3,z3,
                         x4,y4,z4,x5,y5,z5,x6,y6,z6,x7,y7,z7,x8,y8,z8,u)
s=0;t=0;n=0;
Loc=[x4 y4 z4 ; x3 y3 z3;x2 y2 z2;x1 y1 z1;x8 y8 z8;x7 y7 z7;x6 y6 z6; x5 y5 z5];
dNsnt=[(1-n)*(1-t)/8,-(1+s)*(1-n)/8,-(1+s)*(1-t)/8;
       (1-n)*(1+t)/8,(1+s)*(1-n)/8,-(1+s)*(1+t)/8;
       -(1-n)*(1+t)/8,(1-s)*(1-n)/8,-(1-s)*(1+t)/8;
       -(1-n)*(1-t)/8,-(1-s)*(1-n)/8,-(1-s)*(1-t)/8;
       (1+n)*(1-t)/8, -(1+s)*(1+n)/8, (1+s)*(1-t)/8;
       (1+n)*(1+t)/8,(1+s)*(1+n)/8,(1+s)*(1+t)/8;
       -(1+n)*(1+t)/8, (1-s)*(1+n)/8,(1-s)*(1+t)/8;
       -(1+n)*(1-t)/8, -(1-s)*(1+n)/8,(1-s)*(1-t)/8;];
dNsnt1=dNsnt';
J=dNsnt1*Loc;
```

```
detJ=det(J) ;
% 定义B矩阵的系数
a=J(2,2)*J(3,3)-J(2,3)*J(3,2); b=J(1,2)*J(3,3)-J(1,3)*J(3,2);
c=J(1,2)*J(2,3)-J(1,3)*J(2,2); d=J(2,1)*J(3,3)-J(2,3)*J(3,1);
e=J(1,1)*J(3,3)-J(1,3)*J(3,1); f=J(1,1)*J(2,3)-J(1,3)*J(2,1);
g=J(2,1)*J(3,2)-J(2,2)*J(3,1); h=J(1,1)*J(3,2)-J(1,2)*J(3,1);
l=J(1,1)*J(2,2)-J(1,2)*J(2,1);
% 通过循环计算各个矩阵
Bs=zeros(6,3,8);
for i=1:8
    Bs(:,:,i)=[a*dNsnt(i,1)-b*dNsnt(i,2)+c*dNsnt(i,3),0,0;
               0,-d*dNsnt(i,1)+e*dNsnt(i,2)-f*dNsnt(i,3),0;
               0,0,g*dNsnt(i,1)-h*dNsnt(i,2)+l*dNsnt(i,3);
               -d*dNsnt(i,1)+e*dNsnt(i,2)-f*dNsnt(i,3),...
               a*dNsnt(i,1)-b*dNsnt(i,2)+c*dNsnt(i,3),0;
               0,g*dNsnt(i,1)-h*dNsnt(i,2)+l*dNsnt(i,3),...
               -d*dNsnt(i,1)+e*dNsnt(i,2)-f*dNsnt(i,3);
               g*dNsnt(i,1)-h*dNsnt(i,2)+l*dNsnt(i,3),0,...
               a*dNsnt(i,1)-b*dNsnt(i,2)+c*dNsnt(i,3)]/detJ;
end
% 计算 B 矩阵
B=[Bs(:,:,1),Bs(:,:,2),Bs(:,:,3),Bs(:,:,4),...
  Bs(:,:,5),Bs(:,:,6),Bs(:,:,7),Bs(:,:,8)];
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;
                         0,0,0,0.5-NU,0,0;0,0,0,0,0.5-NU,0;0,0,0,0,0,0.5-NU];
w=D*B*u;
strain=B*u/3;
stress=w/3;
end
```

```
function P=Principal(sigma_11,sigma_22,sigma_33,sigma_12,sigma_23,sigma_31)
%sigma_11,sigma_22,sigma_33三个正应力 sigma_12,sigma_23,sigma_31三个切应力
Sig=[sigma_11,sigma_22,sigma_33];
Tau=[sigma_12,sigma_23,sigma_31];
I_1=sum(Sig);
I_2=sum(Sig.*[Sig(2:3),Sig(1)])-sum(Tau.*Tau);
I_3=prod(Sig)+2*prod(Tau)-sum(Sig.*[Tau(2:3),Tau(1)].*[Tau(2:3),Tau(1)]);
p=[1 -I_1 I_2 -I_3];
P=sort(roots(p)','descend');%解三次方程并降序排列
end
```