湖南大學

有限元方法及应用
期末作业

作　业　题　目	三维八节点六面体单元求梁应变
学　生　姓　名	杜元杰
学　生　学　号	S230200190
专　业　班　级	2304
学　院　名　称	机械与运载工程学院
指　导　老　师	王琥

2024 年 1 月 30 日

摘　要

有限元分析（FEA）是一种广泛应用于工程和科学领域的数值分析方法。它通过将复杂的物理系统分解为较小的、易于处理的单元来模拟和分析系统的行为。这种方法在许多领域都有应用，包括航空航天、汽车、建筑、生物医学和许多其他行业。有限元求解中重要的一步是单元类型的选择，包括三角形、四边形、四面体、六面体等单元。

本文对八节点六面体单元做了如下工作：

（1）在平面三角形单元和四节点四边形单元理论的基础上，讨论了三维八节点六面体单元理论，并参考课堂中四边形单元程序使用三维八节点六面体单元非缩减积分编写了 MATLAB 有限元程序，计算了悬臂梁的应变与应力。

（2）使用 Abaqus 软件中 C3D8（八节点线性六面体单元）和非缩减积分求解同样条件下悬臂梁的应变与应力，并与 MATLAB 程序结果对比，其中 MATLAB 程序计算的应变与 Abaqus 结果非常接近，最大误差不超过 3%。

目录

一． 三维八节点六面体单位

对于三维八节点六面体单元，采用如下的节点编号规则：

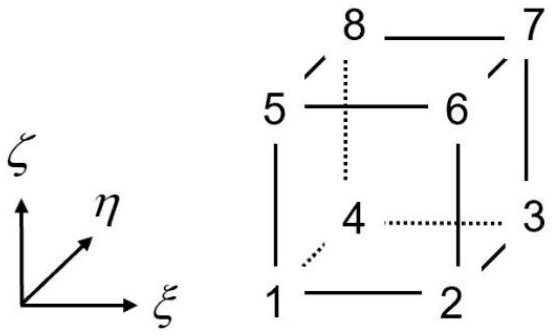


图 1 八节点六面体单元

形函数与等参坐标关系：

$$\begin{cases} \boldsymbol{N}_1=\dfrac{1}{8}(1-\xi)(1-\eta)(1-\zeta) \\ \boldsymbol{N}_2=\dfrac{1}{8}(1+\xi)(1-\eta)(1-\zeta) \\ \boldsymbol{N}_3=\dfrac{1}{8}(1+\xi)(1+\eta)(1-\zeta) \\ \boldsymbol{N}_4=\dfrac{1}{8}(1-\xi)(1+\eta)(1-\zeta) \\ \boldsymbol{N}_5=\dfrac{1}{8}(1-\xi)(1-\eta)(1+\zeta) \\ \boldsymbol{N}_6=\dfrac{1}{8}(1+\xi)(1-\eta)(1+\zeta) \\ \boldsymbol{N}_7=\dfrac{1}{8}(1+\xi)(1+\eta)(1+\zeta) \\ \boldsymbol{N}_8=\dfrac{1}{8}(1-\xi)(1+\eta)(1+\zeta) \end{cases} \tag{1}$$

单元刚度矩阵 ***k_element***，可以通过对 8 个等参点进行高斯积分求出：

$$\boldsymbol{k}\_element=\sum_{i=1}^{8} w_i \boldsymbol{B}_i^T \boldsymbol{D} \boldsymbol{B}_i \left|\boldsymbol{J}_i\right| \tag{2}$$

其中 w_i 是 8 个等参点的高斯积分权重，均为 1。$\boldsymbol{B}_i$ 为对应每个积分点的应变矩阵，$\boldsymbol{D}$ 为三维材料矩阵，$\boldsymbol{J}_i$ 为对应每个积分点的雅各比矩阵。

三维材料矩阵 $\boldsymbol{D}$ 与材料泊松比 ν 和杨氏模量 E 有关：

$$\boldsymbol{D}=\frac{E}{(1+\nu)(1-2\nu)}\begin{bmatrix}1&-\nu&\nu&\nu&0&0&0\\0&1&-\nu&\nu&0&0&0\\0&0&1&-\nu&0&0&0\\0&0&0&0.5&-\nu&0&0\\0&0&0&0&0.5&-\nu&0\\0&0&0&0&0&0.5&-\nu\end{bmatrix} \tag{3}$$

雅各比矩阵 $\boldsymbol{J}$ 行列式的绝对值$|\boldsymbol{J}|$表示物理坐标变换为自然坐标后体积的缩放系数。对于八节点六面体单元来说，8 个等参点中每个点都有各自对应的雅各比矩阵 $\boldsymbol{J}$，每个单元需要计算 8 个雅各比矩阵 $\boldsymbol{J}$：

$$\boldsymbol{J}=\begin{bmatrix}\frac{\partial \boldsymbol{N}_1}{\partial\xi}&\frac{\partial \boldsymbol{N}_2}{\partial\xi}&\frac{\partial N_3}{\partial\xi}&\frac{\partial \boldsymbol{N}_4}{\partial\xi}&\frac{\partial \boldsymbol{N}_5}{\partial\xi}&\frac{\partial \boldsymbol{N}_6}{\partial\xi}&\frac{\partial \boldsymbol{N}_7}{\partial\xi}&\frac{\partial \boldsymbol{N}_8}{\partial\xi}\\\frac{\partial \boldsymbol{N}_1}{\partial\eta}&\frac{\partial \boldsymbol{N}_2}{\partial\eta}&\frac{\partial \boldsymbol{N}_3}{\partial\eta}&\frac{\partial \boldsymbol{N}_4}{\partial\eta}&\frac{\partial \boldsymbol{N}_5}{\partial\eta}&\frac{\partial \boldsymbol{N}_6}{\partial\eta}&\frac{\partial \boldsymbol{N}_7}{\partial\eta}&\frac{\partial \boldsymbol{N}_8}{\partial\eta}\\\frac{\partial \boldsymbol{N}_1}{\partial\zeta}&\frac{\partial \boldsymbol{N}_2}{\partial\zeta}&\frac{\partial \boldsymbol{N}_3}{\partial\zeta}&\frac{\partial \boldsymbol{N}_4}{\partial\zeta}&\frac{\partial \boldsymbol{N}_5}{\partial\zeta}&\frac{\partial \boldsymbol{N}_6}{\partial\zeta}&\frac{\partial \boldsymbol{N}_7}{\partial\zeta}&\frac{\partial \boldsymbol{N}_8}{\partial\zeta}\end{bmatrix}\begin{bmatrix}x_1&y_1&z_1\\x_2&y_2&z_2\\x_3&y_3&z_3\\x_4&y_4&z_4\\x_5&y_5&z_5\\x_6&y_6&z_6\\x_7&y_7&z_7\\x_8&y_8&z_8\end{bmatrix} \tag{4}$$

求出各个点对应的雅各比矩阵 $\boldsymbol{J}$ 后，通过逆变换得到该等参点处形函数对物理坐标的导数：

$$\begin{Bmatrix}\frac{\partial \boldsymbol{N}_i}{\partial x}\\\frac{\partial \boldsymbol{N}_i}{\partial y}\\\frac{\partial \boldsymbol{N}_i}{\partial z}\end{Bmatrix}=\boldsymbol{J}^{-1}\begin{Bmatrix}\frac{\partial \boldsymbol{N}_i}{\partial\xi}\\\frac{\partial \boldsymbol{N}_i}{\partial\eta}\\\frac{\partial \boldsymbol{N}_i}{\partial\zeta}\end{Bmatrix} \tag{5}$$

其中，i = 1 - 8。

为了得到某点对应的应变矩阵 $\boldsymbol{B}$，需要将该点形函数对物理坐标导数的分量组装成为 $\boldsymbol{B}$ 的子块，再由 $\boldsymbol{B}$ 的子块组合为应变矩阵 $\boldsymbol{B}$。对于八节点六面体单元来说，8 个等参点中每个点都有各自对应的应变矩阵 $\boldsymbol{B}$，每个单元需要计算 8 个应变矩阵 $\boldsymbol{B}$ 和 64 个 $\boldsymbol{B}$ 的子块。

应变矩阵 $\boldsymbol{B}$ 和 $\boldsymbol{B}$ 的子块组合的规则是：

$$\boldsymbol{B}=[\boldsymbol{B}_1\quad\boldsymbol{B}_2\quad\boldsymbol{B}_3\quad\boldsymbol{B}_4\quad\boldsymbol{B}_5\quad\boldsymbol{B}_6\quad\boldsymbol{B}_7\quad\boldsymbol{B}_8] \tag{6}$$

$\boldsymbol{B}$ 的子块：

$$\boldsymbol{B}_i=\begin{bmatrix}\dfrac{\partial \boldsymbol{N}_i}{\partial x} & 0 & 0\\ 0 & \dfrac{\partial \boldsymbol{N}_i}{\partial y} & 0\\ 0 & 0 & \dfrac{\partial \boldsymbol{N}_i}{\partial z}\\ 0 & \dfrac{\partial \boldsymbol{N}_i}{\partial z} & \dfrac{\partial \boldsymbol{N}_i}{\partial y}\\ \dfrac{\partial \boldsymbol{N}_i}{\partial z} & 0 & \dfrac{\partial \boldsymbol{N}_i}{\partial x}\\ \dfrac{\partial \boldsymbol{N}_i}{\partial y} & \dfrac{\partial \boldsymbol{N}_i}{\partial x} & 0\end{bmatrix} \tag{7}$$

其中，i = 1 - 8。

8 个等参高斯积分点坐标：

$$\begin{cases}\left(-\dfrac{1}{\sqrt{3}},\ -\dfrac{1}{\sqrt{3}},\ -\dfrac{1}{\sqrt{3}}\right)\\ \left(\dfrac{1}{\sqrt{3}},\ -\dfrac{1}{\sqrt{3}},\ -\dfrac{1}{\sqrt{3}}\right)\\ \left(\dfrac{1}{\sqrt{3}},\ \dfrac{1}{\sqrt{3}},\ -\dfrac{1}{\sqrt{3}}\right)\\ \left(-\dfrac{1}{\sqrt{3}},\ \dfrac{1}{\sqrt{3}},\ -\dfrac{1}{\sqrt{3}}\right)\\ \left(-\dfrac{1}{\sqrt{3}},\ -\dfrac{1}{\sqrt{3}},\ \dfrac{1}{\sqrt{3}}\right)\\ \left(\dfrac{1}{\sqrt{3}},\ -\dfrac{1}{\sqrt{3}},\ \dfrac{1}{\sqrt{3}}\right)\\ \left(\dfrac{1}{\sqrt{3}},\ \dfrac{1}{\sqrt{3}},\ \dfrac{1}{\sqrt{3}}\right)\\ \left(-\dfrac{1}{\sqrt{3}},\ \dfrac{1}{\sqrt{3}},\ \dfrac{1}{\sqrt{3}}\right)\end{cases} \tag{8}$$

二． MATLAB 程序设计

MATLAB 程序框架图 1 所示：

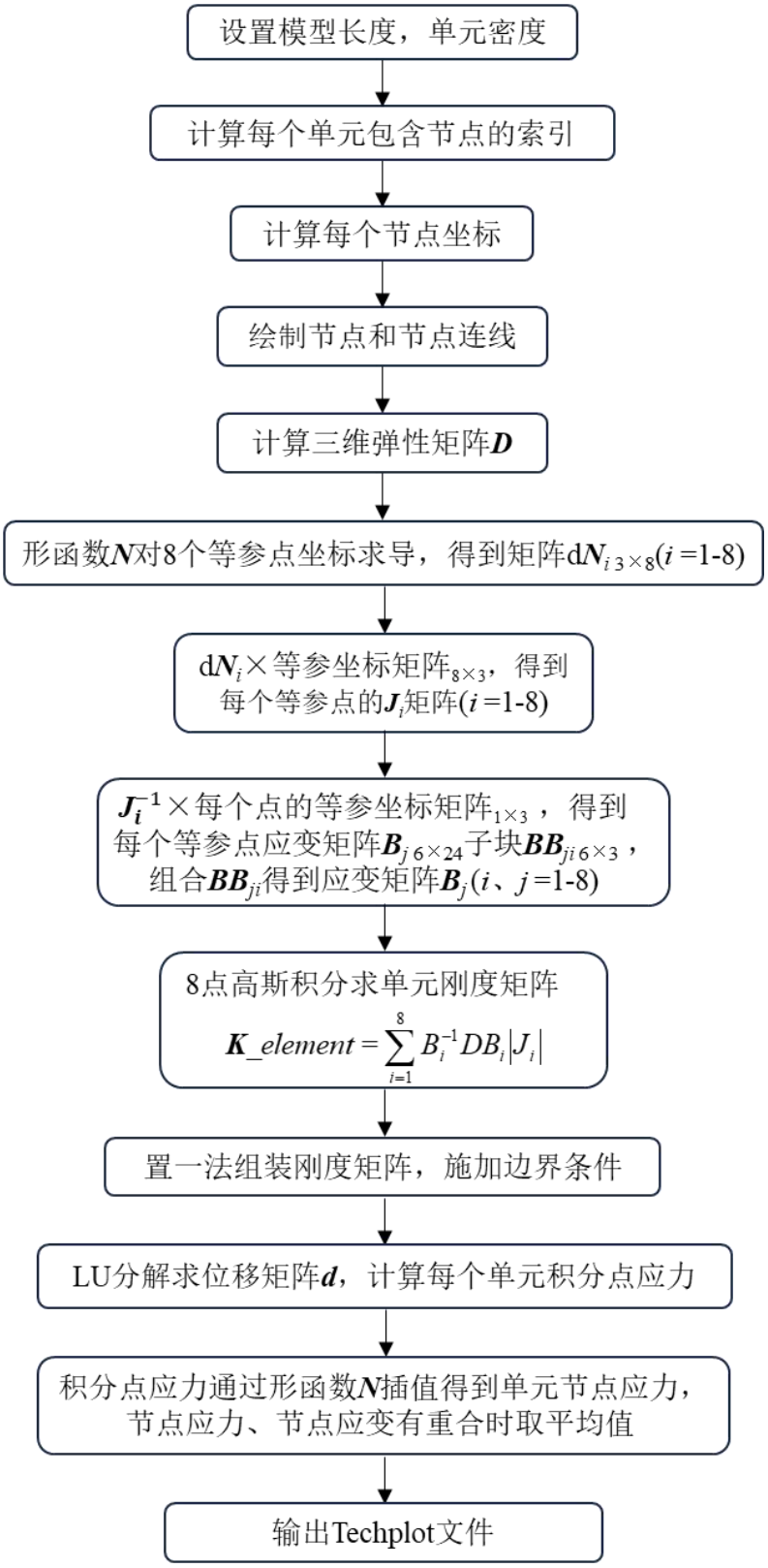


图 2 MATLAB 程序流程

三．Abaqus 求解

1.建模

在部件中新建模型，设置长度为 1m，宽度和高度为 0.2m。

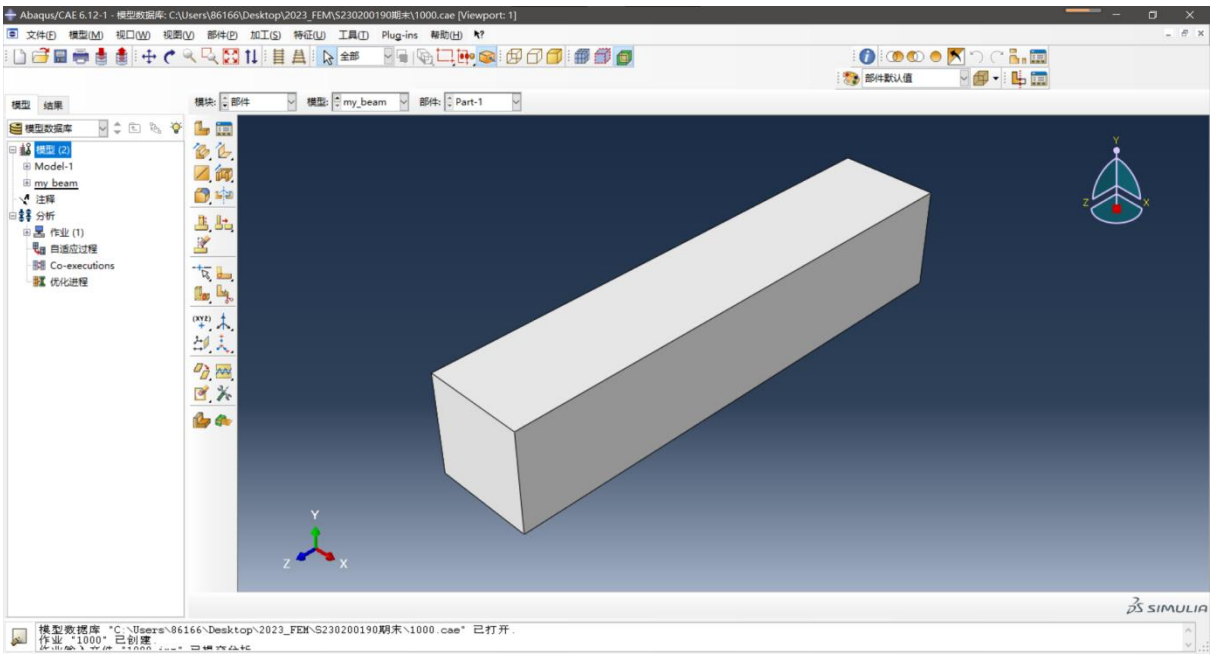


图 3 悬臂梁模型

2.添加属性

在属性中设置杨氏模量 E=210MPa，泊松比 P=0.3。

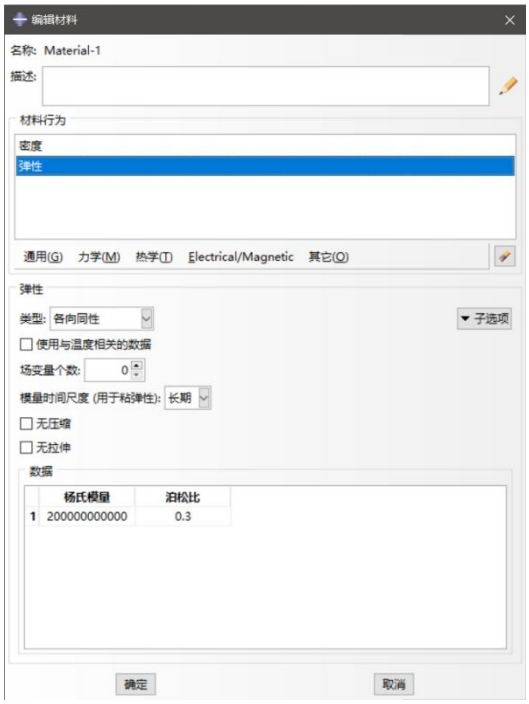


图 4 材料属性

3.创建网格

设置近似全局尺寸为 0.03333。保证宽度和高度方向有 6 个网格，长度方向 30 个网格，共 1080 个网格。

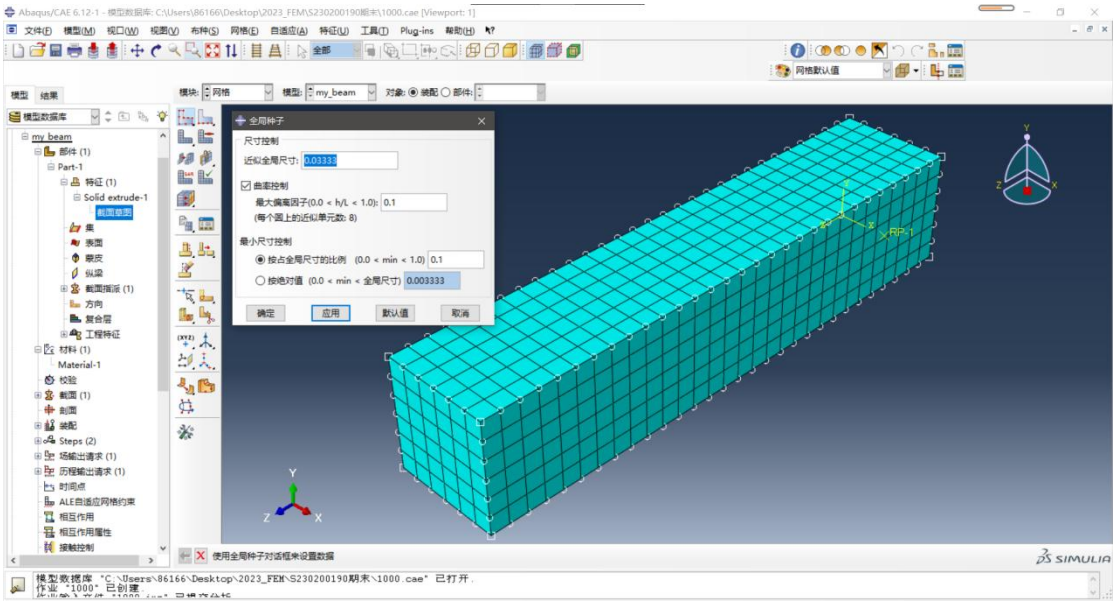


图 5 网格划分

单元类型选择 C3D8，非缩减积分。

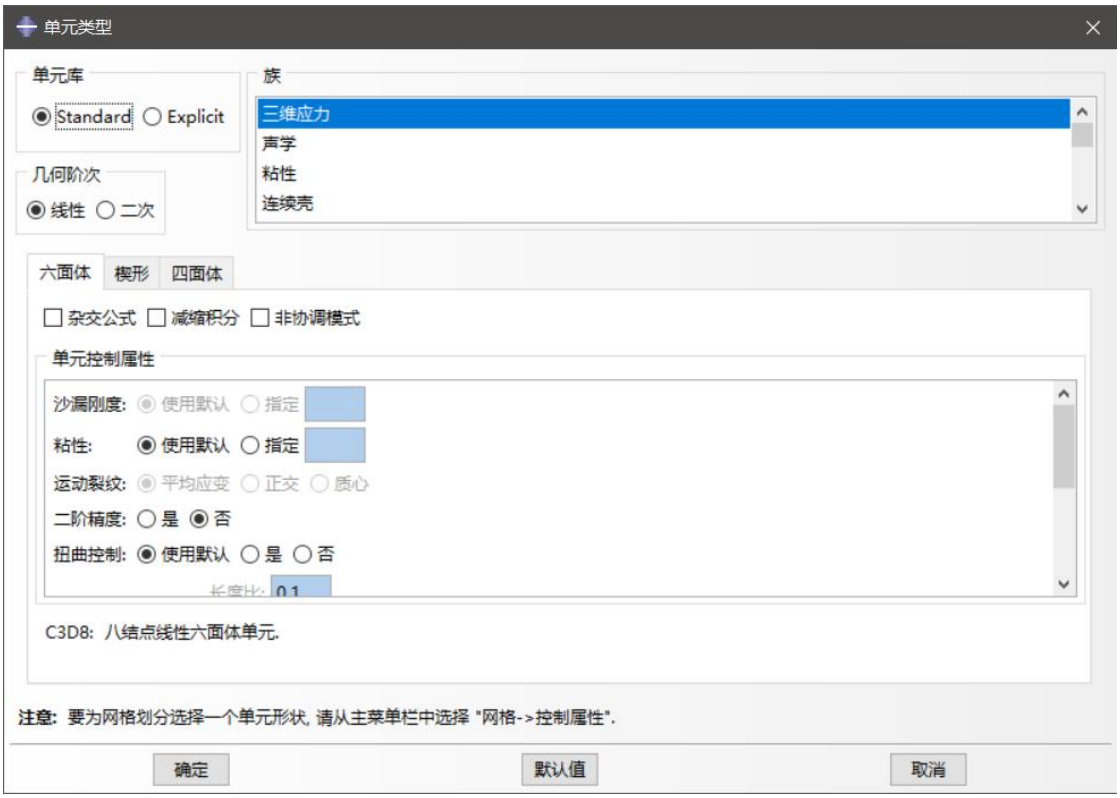


图 6 单元类型

4.载荷和边界条件

在图中的节点上添加集中载荷 CF1=2000N，CF2=1000N，CF3=1000N。

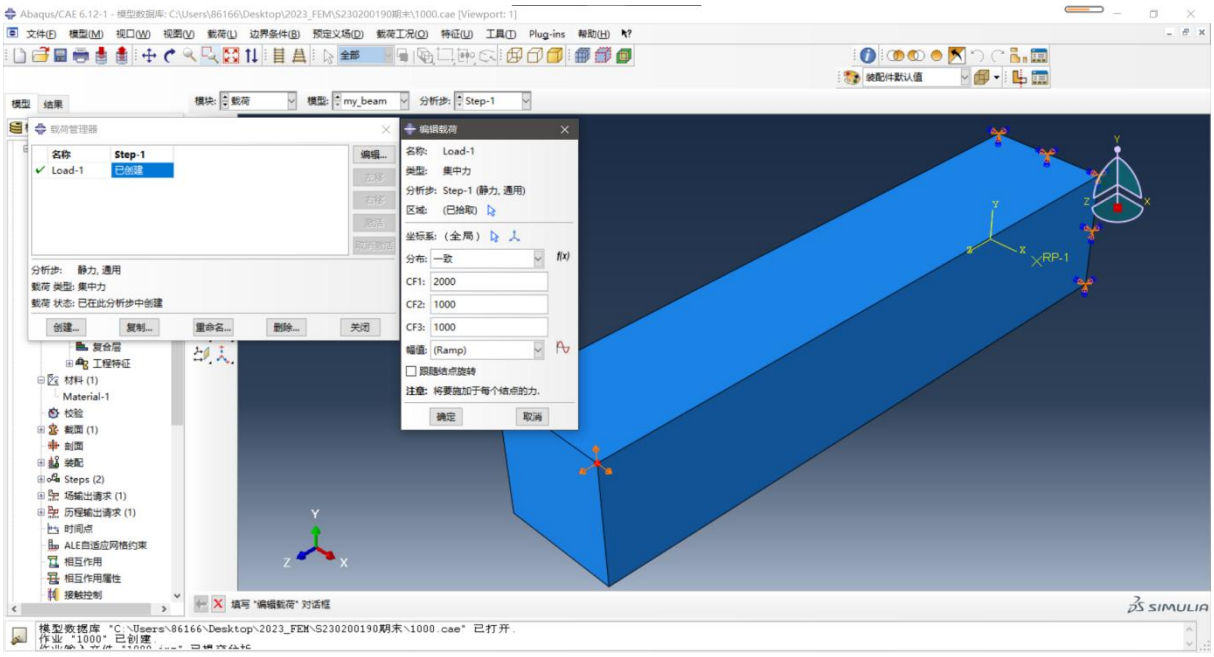


图 7 载荷

在图 8 中底面上施加位移边界条件，选择完全固定。

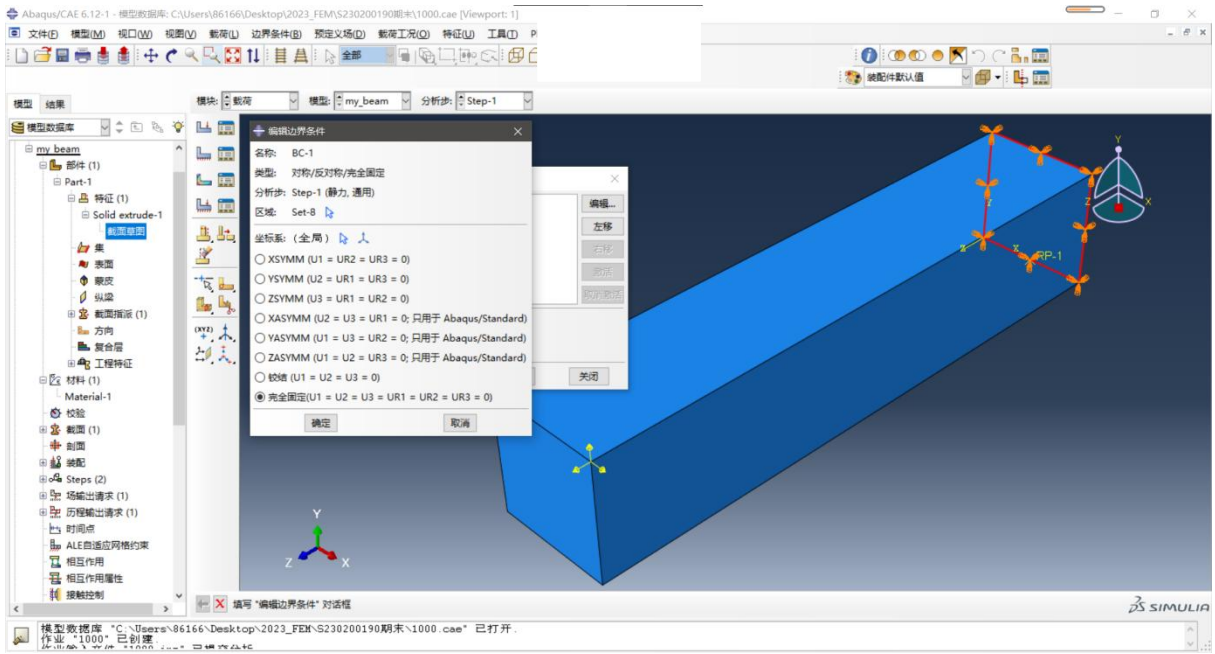


图 8 位移边界条件

5.求解

选择之前的模型创建作业然后进行提交求解。

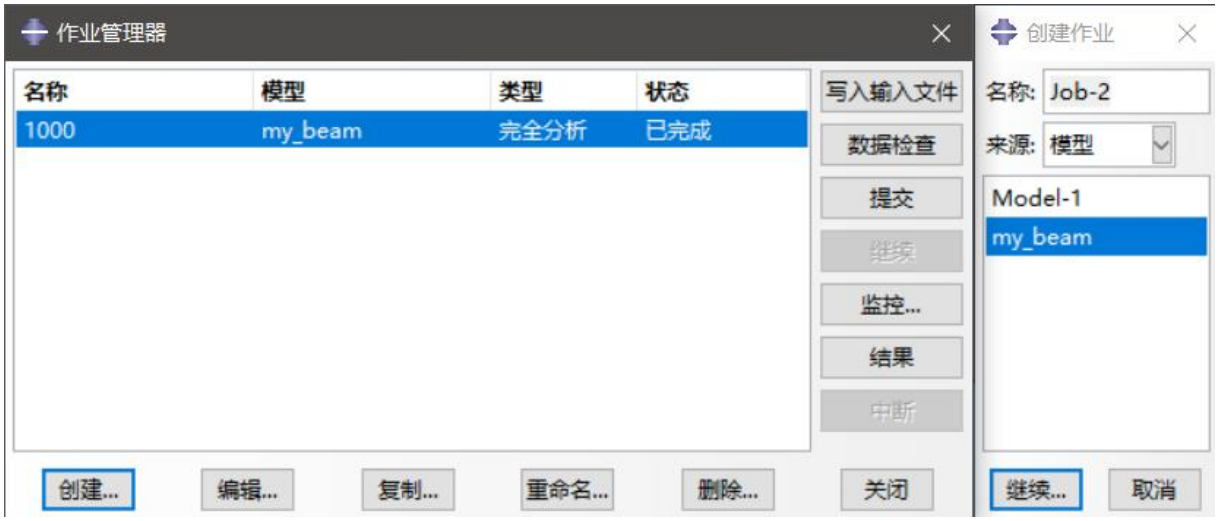


图 9 求解流程

6.结果

选择可视化模块，可以查看位移，应力云图。

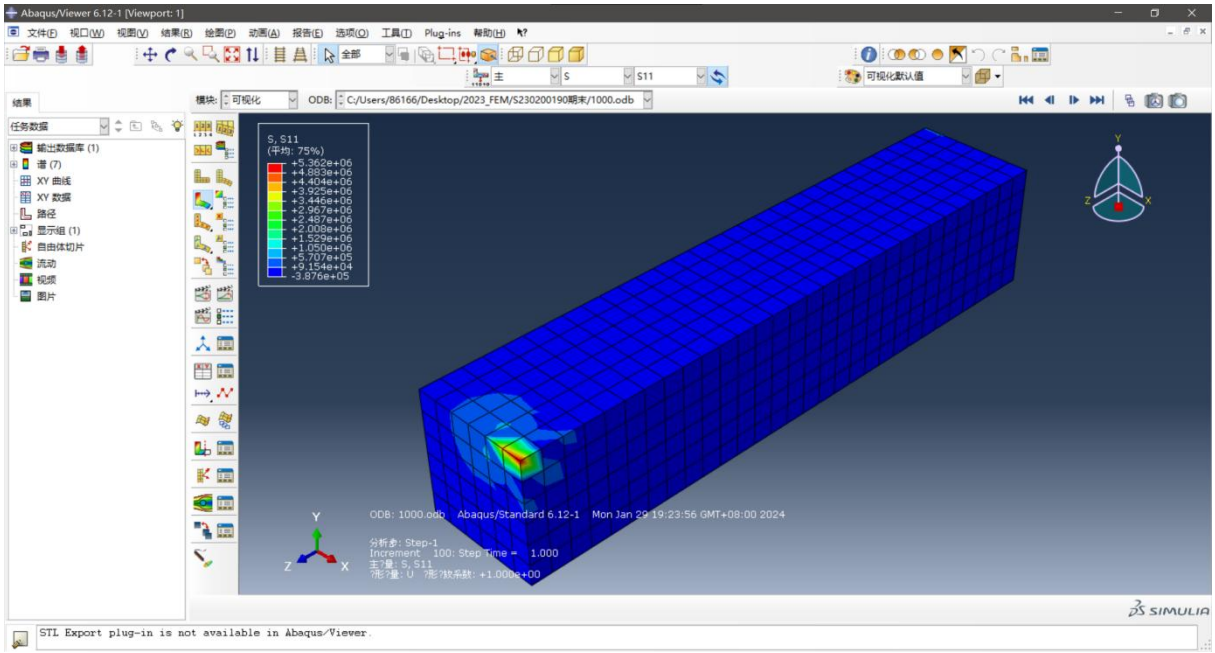
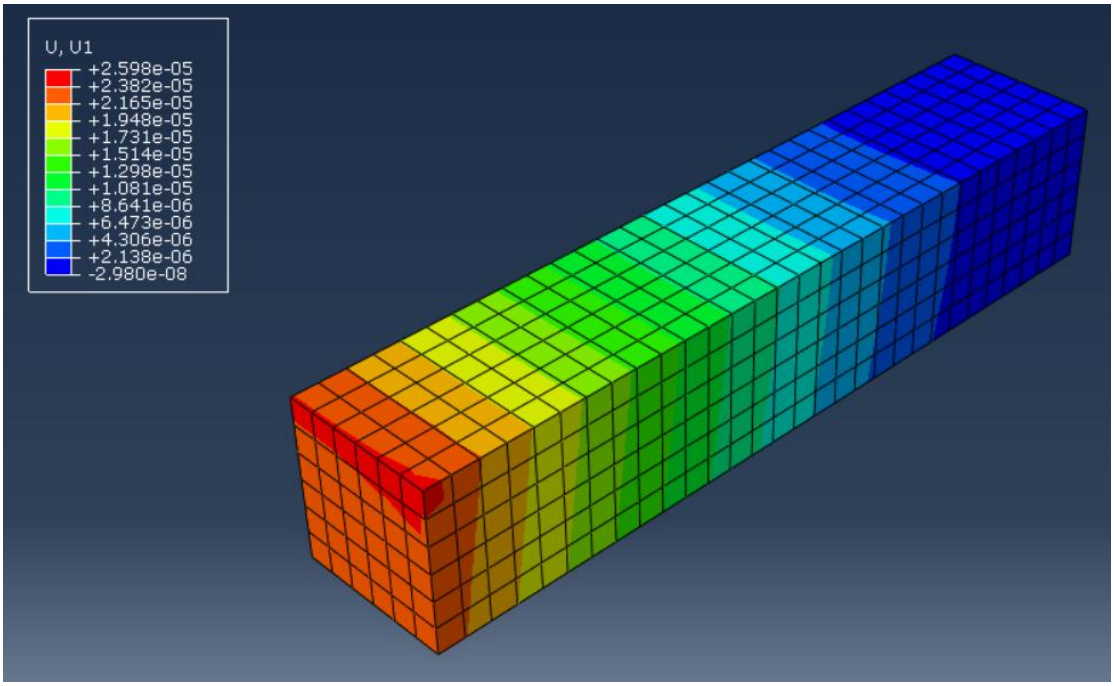


图 10 可视化

四．实验结果与讨论：

1.Abqus 与 MATLAB 位移对比：



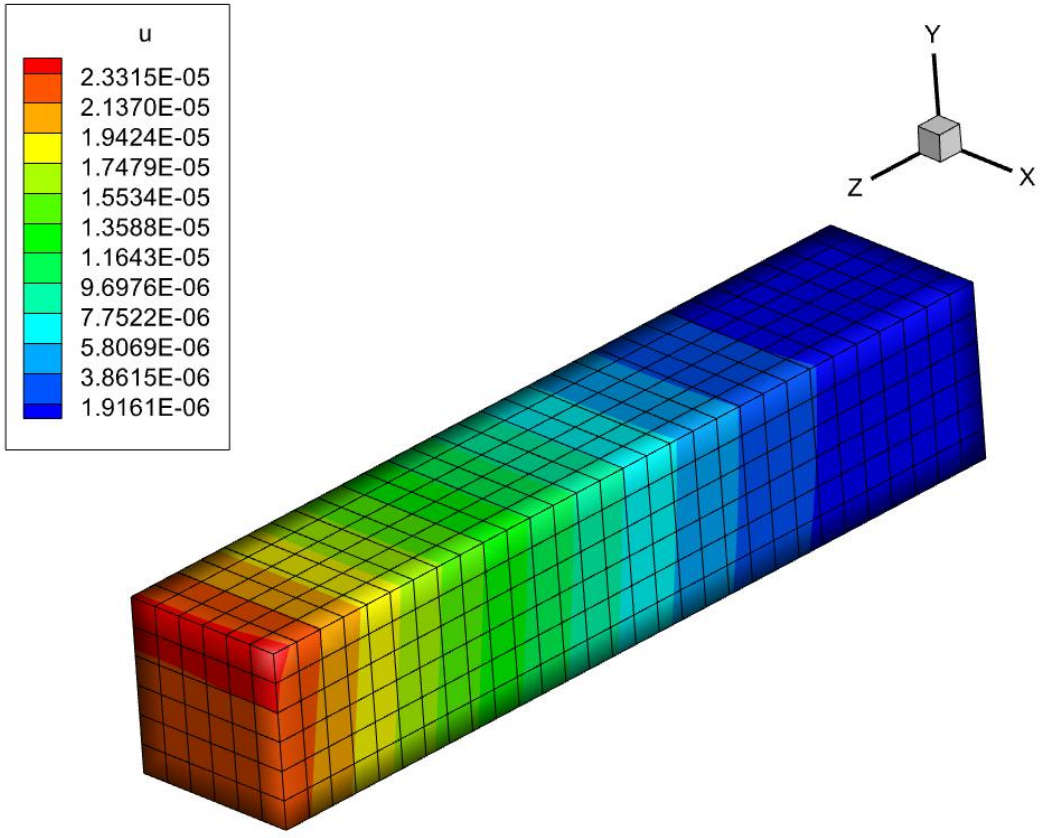


图 11 Abaqus（上）与 MATLAB（下）*x* 方向位移比较

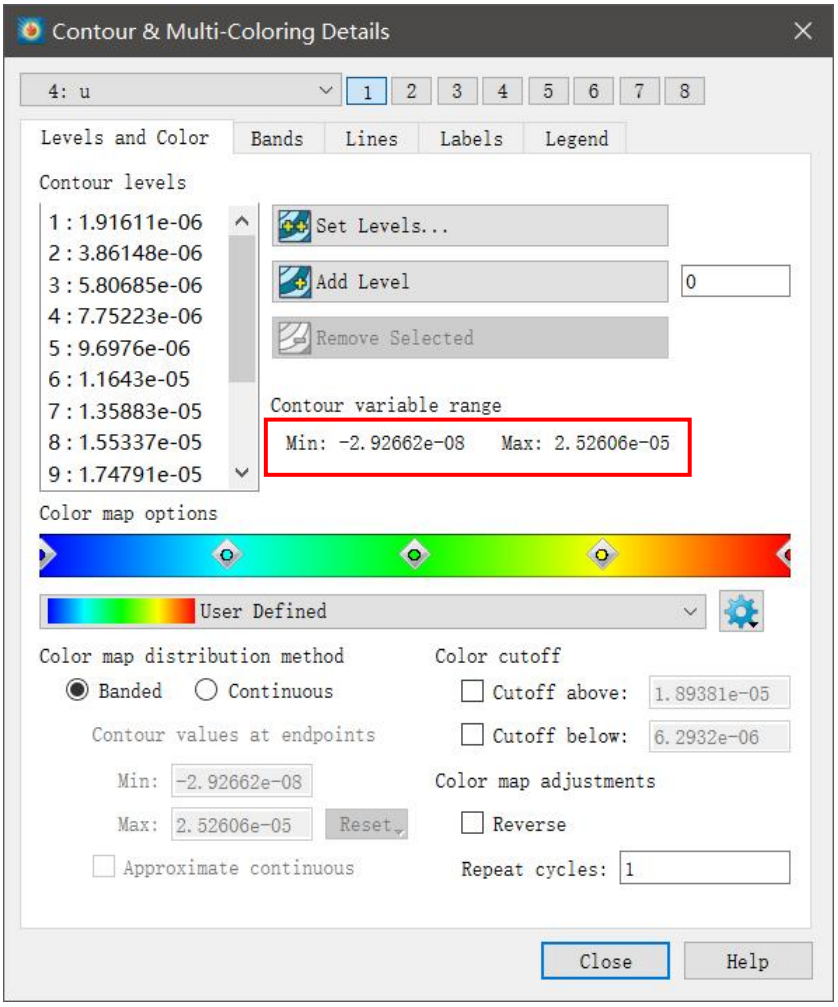


图 12 MATLAB x 方向位移最大值与最小值

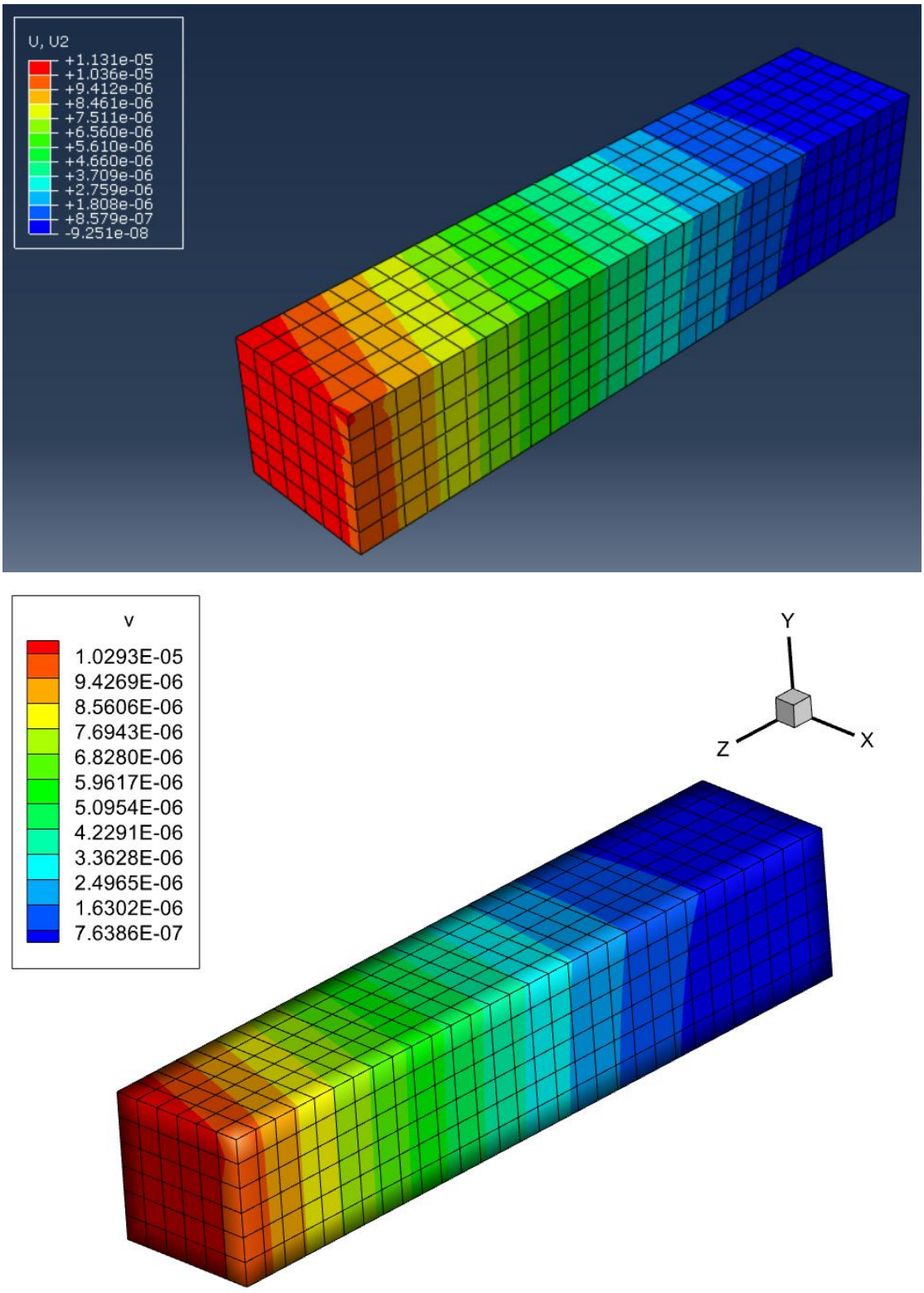


图 13 Abaqus（上）与 MATLAB（下）*y* 方向位移比较

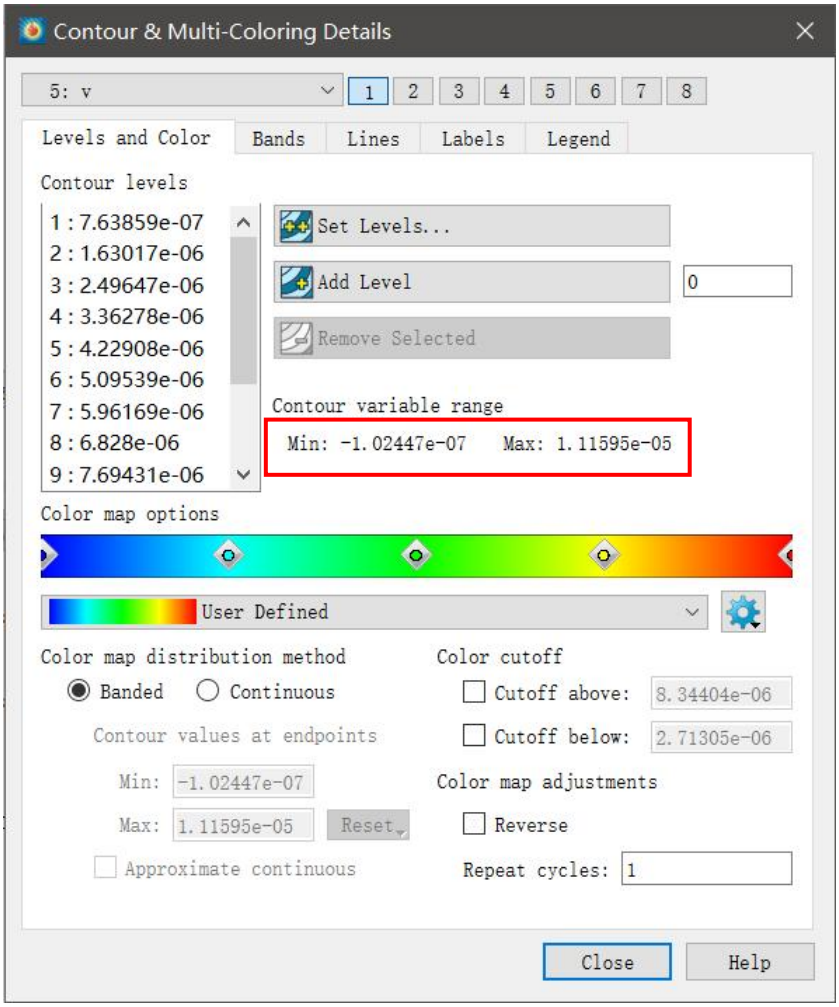


图 14 MATLAB y 方向位移最大值与最小值

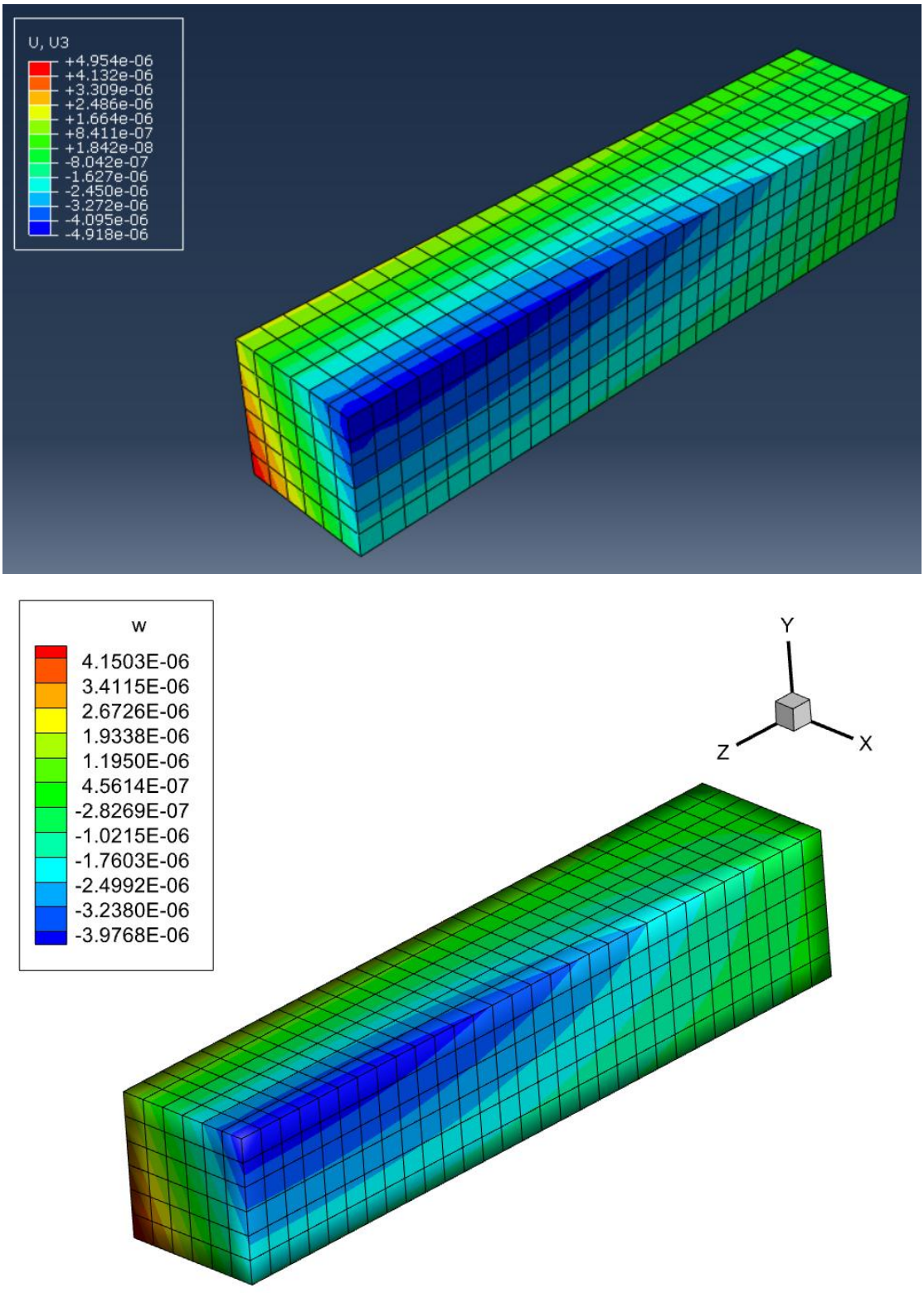


图 15 Abaqus（上）与 MATLAB（下）z 方向位移比较

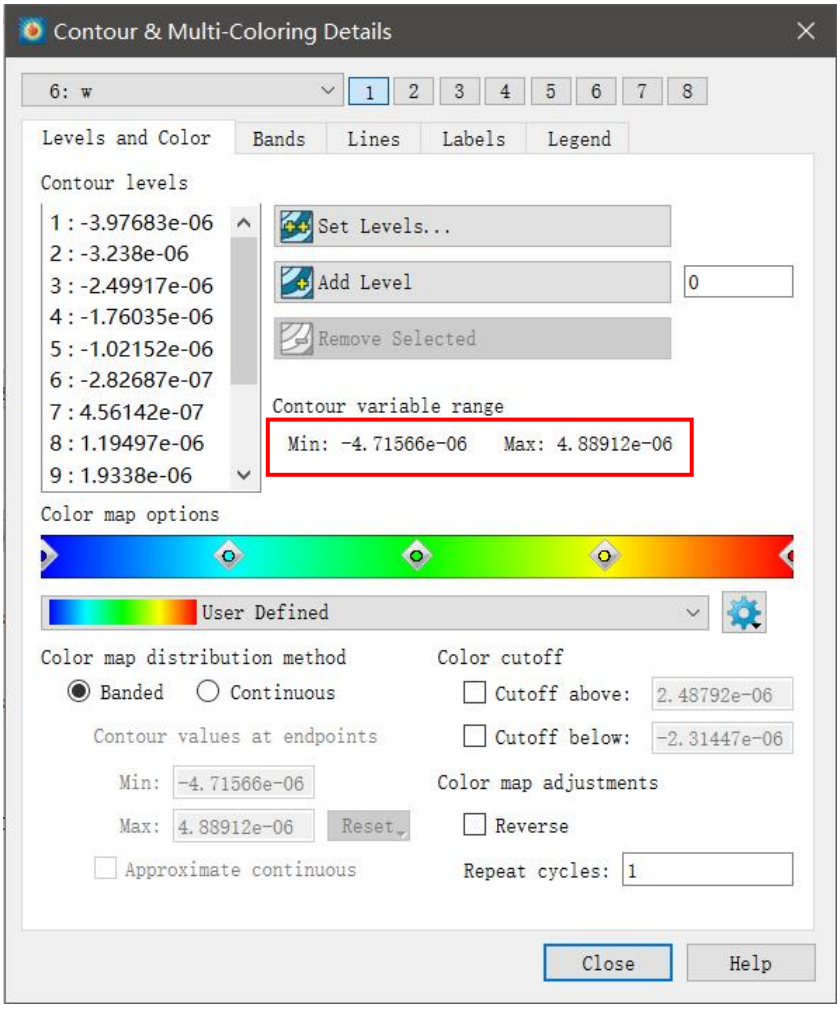


图 16 MATLAB z 方向位移最大值与最小值

表 1 Abaqus、MATLAB 最大位移 单位：m

位移	Abaqus	**MATLAB**	误差
$u_{\max}/10^{-5}$	2.526	**2.598**	2.9%
$v_{\max}/10^{-5}$	1.131	**1.116**	-1.3%
$w_{\max}/10^{-6}$	4.954	**4.889**	-1.3%

从图 11 - 图 16、表 1 可以看出，MATLAB 计算的节点位移最大值和 Abaqus 结果非常接近，并且位移的分布也和 Abaqus 结果非常接近，这是因为都使用了 C3D8 单元非缩减积分。

2.Abqus 与 MATLAB 应力对比：

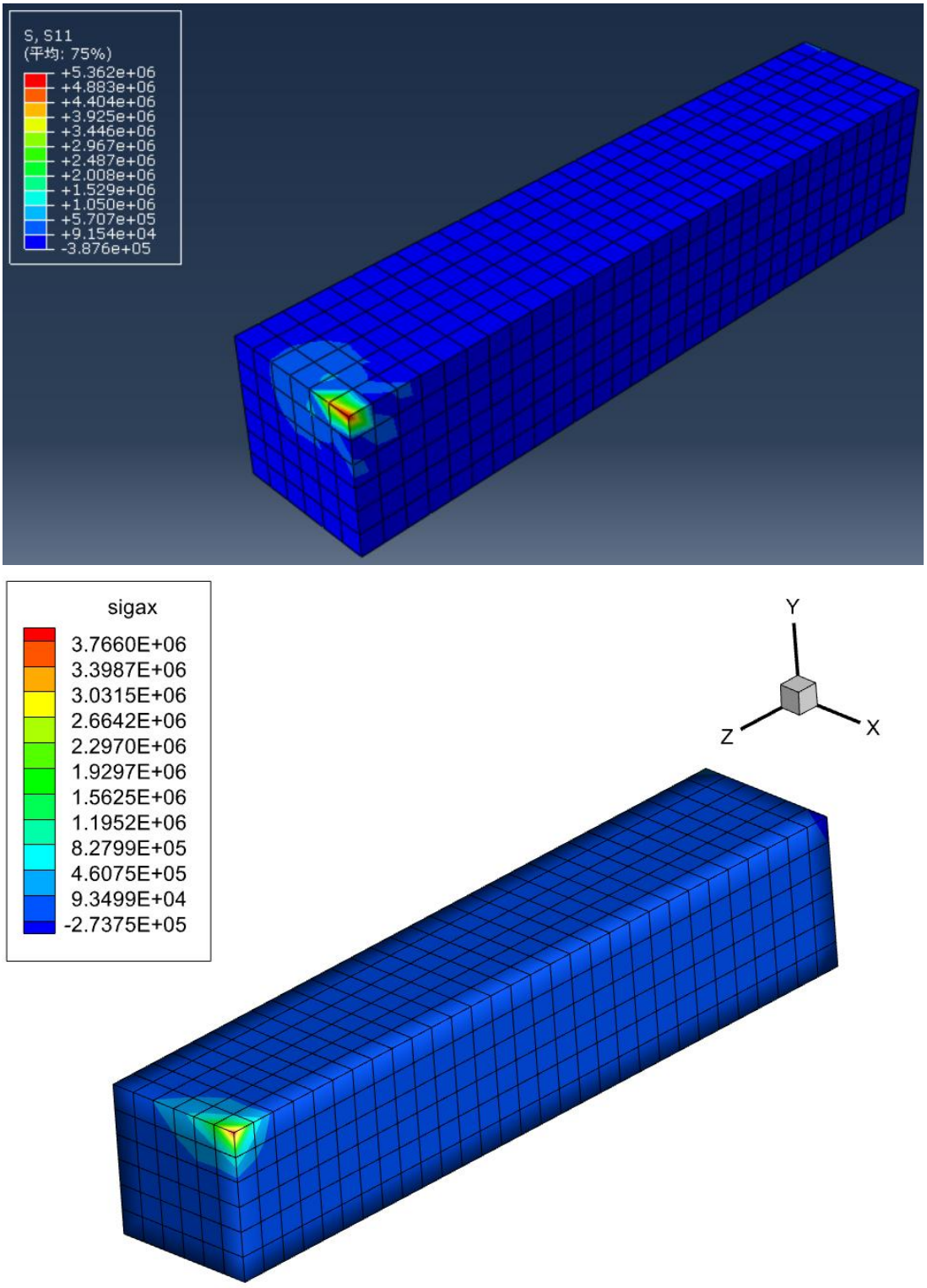


图 17 Abaqus（上）与 MATLAB（下）σ_x 比较

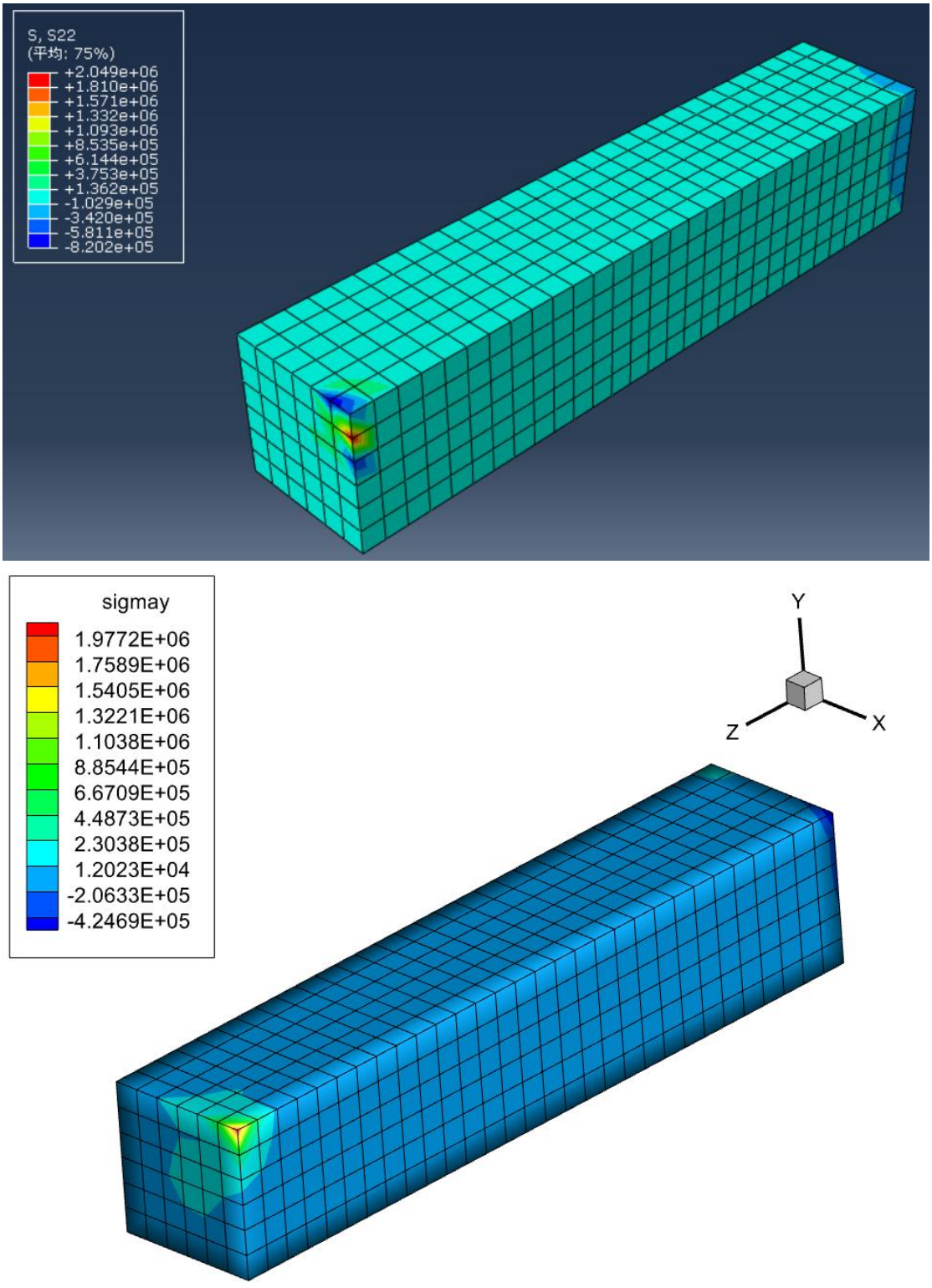


图 18 Abaqus（上）与 MATLAB（下）σ_y 比较

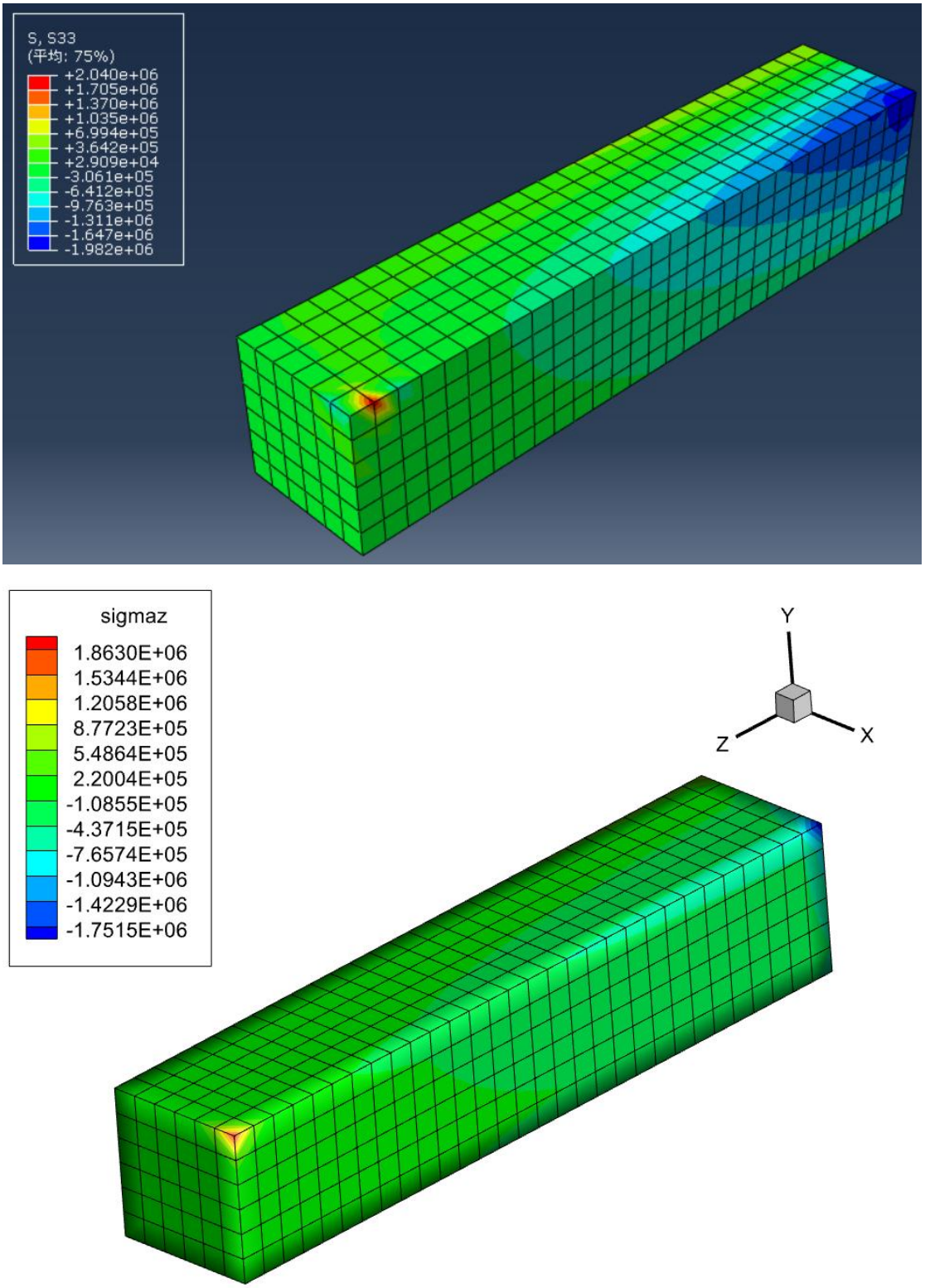


图 19 Abaqus（上）与 MATLAB（下）σ_z 比较

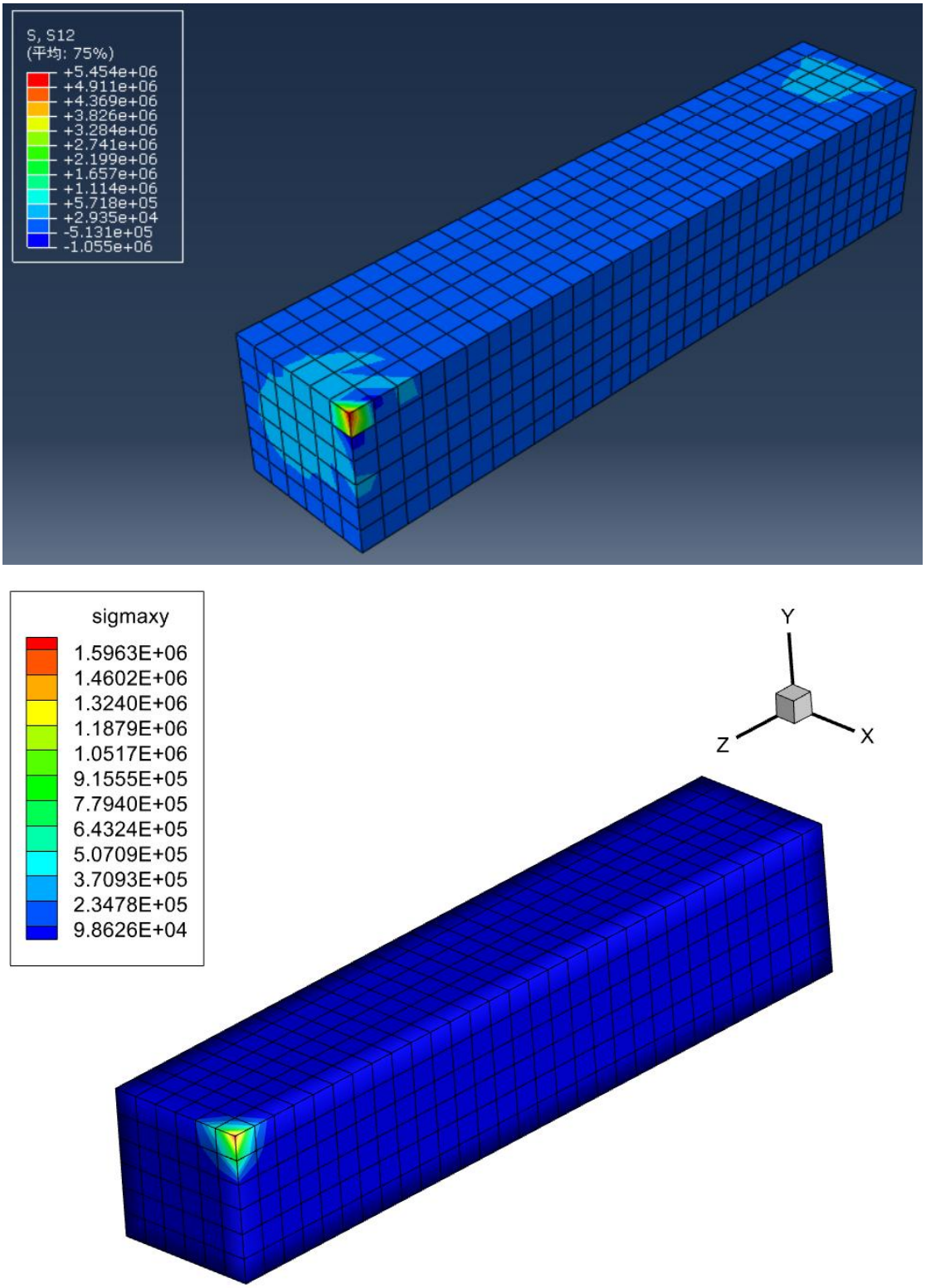


图 20 Abaqus（上）与 MATLAB（下）σ_{xy} 比较

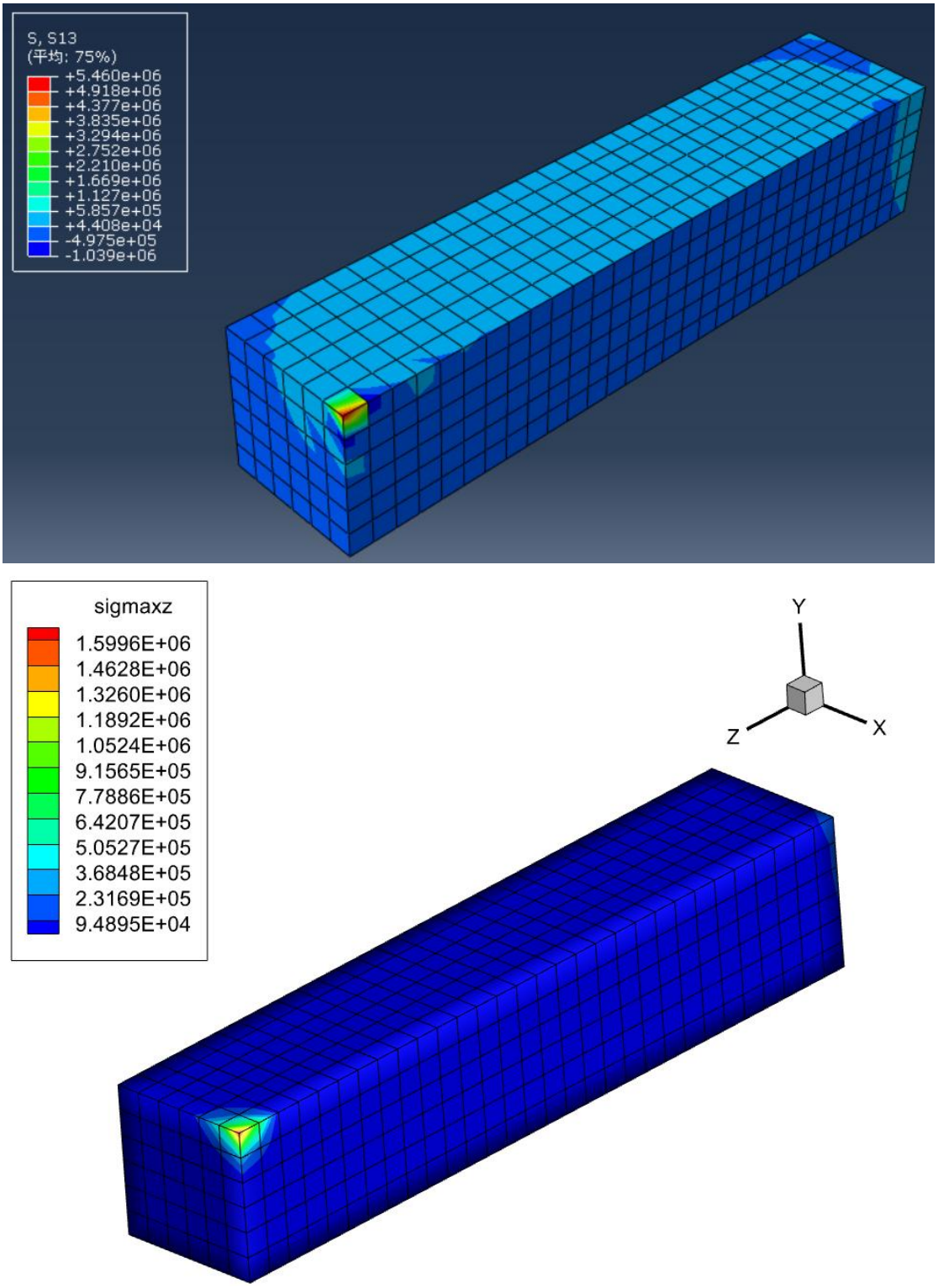


图 21 Abaqus（上）与 MATLAB（下）σ_{xz} 比较

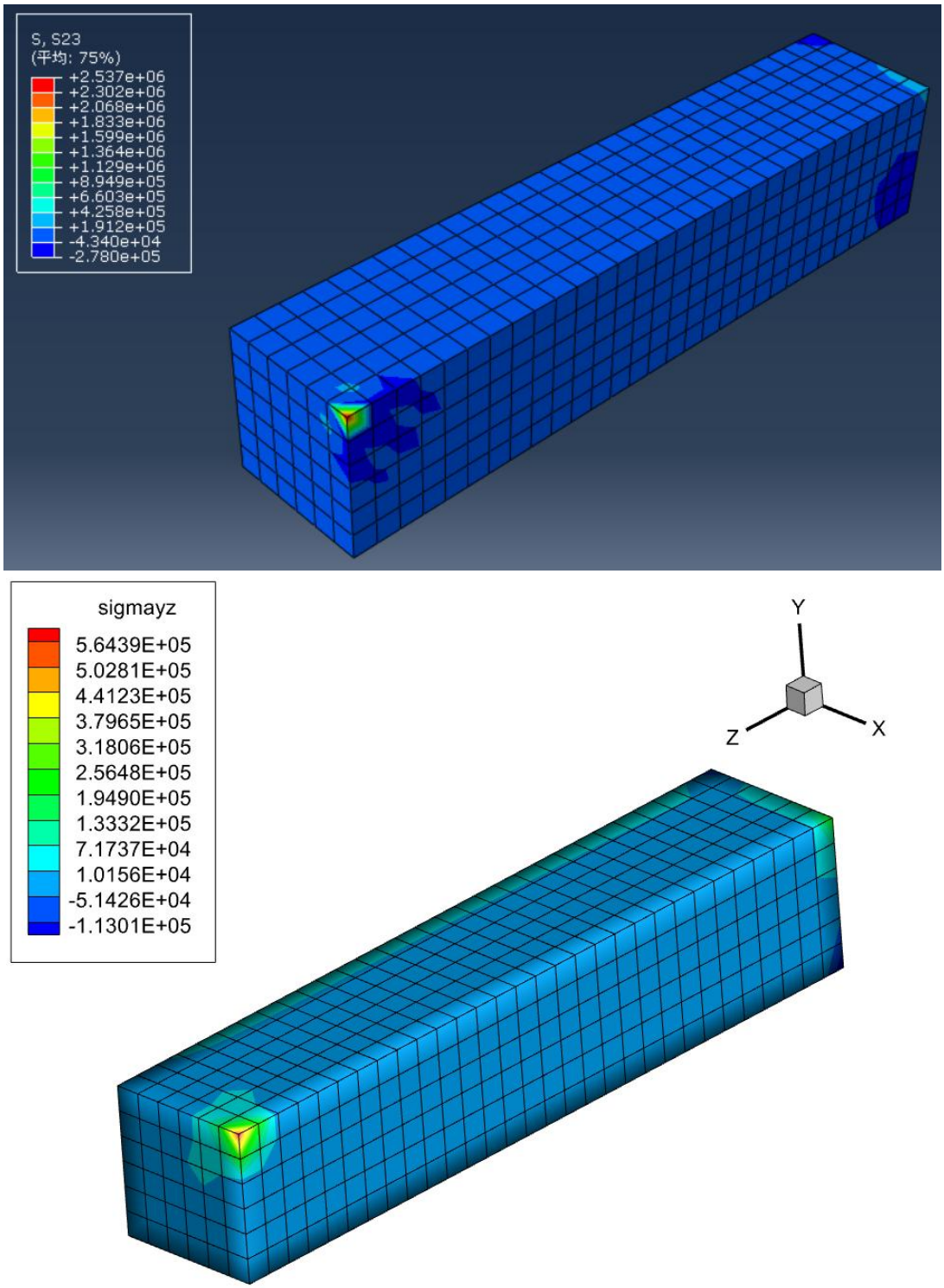


图 22 Abaqus（上）与 MATLAB（下）σ_{yz} 比较

从图 17 - 图 22 来看，MATLAB 计算的应力分布总体和 Abaqus 接近，但是数值上差距较大。原因在于，MATLAB 程序对积分点采用形函数外插得到节点应力，高斯积分应力虽然是单元内最佳应力点，但在节点上并非最佳。Abaqus 可能采了用其它方法从积分点获得节点上的应力。

参考文献

[1] 蒙坤江,喻勇.基于 MATLAB 的三维有限元程序设计——以八节点六面体等参单元为例[C]//四川省力学学会 2008 年学术大会论文集.2008.
[2] 浮涛.基于 MATLAB 的有限元结构分析[J].四川水泥,2022,(07):36-38.
[3] 吴小希,周小波,李彬.MATLAB 在结构力学分析中的应用[J].城市建设理论研究:电子版,2011(29).
[4] 德 卡坦 P.I,德 Kattan P.I.MATLAB 有限元分析与应用[M].清华大学出版社,2004.

附 录

MATLAB 程序：

```
% 学号：S230200190，姓名：杜元杰
% 8 节点 6 面体等参单元 求解悬臂梁的应力应变
% 宽度、高度 0.2m，长度 1m，网格数量 1080 个
% 一端固支，最后一个节点 x 方向受力为 2000，y、z 方向受力为 1000N，均使用国际单位制
% 位移计算结果与 Abqus 使用 C3D8 单元相差很小
format short
clear all;
x_length = 0.2;
y_length = 0.2;
z_length = 1;
lx = 6;%x 方向单元个数
ly = lx;%y 方向单元个数
lz = 5 * ly;%z 方向单元个数
elements_num = lx * ly * lz;%单元个数
nodes_num = (lx + 1) * (ly + 1) * (lz + 1);
elements = zeros(elements_num, 8);%每行为单元包含节点的索引

%计算每个单元包含节点的索引
count = 0;
for i=1:lx
    for j=1:ly
        for k=1:lz
            count = count + 1;
            elements(count,:) =[
             (lx + 1) * (ly + 1) * (k - 1) + (lx +1) * (j - 1) + i,
```

```
            (lx + 1) * (ly + 1) * (k - 1) + (lx +1) * (j - 1) + i
+ 1,
            (lx + 1) * (ly + 1) * (k - 1) + (lx +1) * (j ) + i + 1,
            (lx + 1) * (ly + 1) * (k - 1) + (lx +1) * (j ) + i,
            (lx + 1) * (ly + 1) * (k ) + (lx +1) * (j - 1) + i,
            (lx + 1) * (ly + 1) * (k ) + (lx +1) * (j - 1) + i + 1,
            (lx + 1) * (ly + 1) * (k ) + (lx +1) * (j ) + i + 1,
            (lx + 1) * (ly + 1) * (k ) + (lx +1) * (j ) + i,];
        end
    end
end

%计算每个节点坐标
x_nodes = zeros(nodes_num,1);%节点 x 坐标
y_nodes = zeros(nodes_num,1);%节点 y 坐标
z_nodes = zeros(nodes_num,1);%节点 z 坐标
count = 0;
for k=1:lz+1
    for j=1:ly+1
        for i=1:lx+1
            count = count + 1;
            x_nodes(count) = (i - 1) / lx * x_length;
            y_nodes(count) = (j - 1) / ly * y_length;
            z_nodes(count) = (k - 1) / lz * z_length;
        end
    end
end

%绘图
```

```
figure(1)
hold on
axis equal
axis off
plot3(x_nodes,   z_nodes, y_nodes,'o');
xlabel('x');
ylabel('y');
zlabel('z');
view(-45, -30);
for i = 1 : nodes_num
    text(x_nodes(i),   z_nodes(i),
y_nodes(i),num2str(i),'FontSize',8);
end
for i = 1 : elements_num
    x = [];
    y = [];
    z = [];
    for j = 1 : 8
        x = [x; x_nodes(elements(i, j))];
        y = [y; y_nodes(elements(i, j))];
        z = [z; z_nodes(elements(i, j))];
    end
    for j = 1 : 4
        line([x(j),x(j + 4)], [z(j),z(j + 4)], [y(j),y(j + 4)] );
    end
    for j = 0 : 4 : 4

line([x(1+j),x(2+j),x(3+j),x(4+j),x(1+j)] ,[z(1+j),z(2+j),z(3+j)
,z(4+j),z(1+j)],[y(1+j),y(2+j),y(3+j),y(4+j),y(1+j)]);
```

```
    end
end

% 计算三维弹性矩阵 D
E = 200e9;% Q235，单位都是国际单位制
poisson = 0.3;% Q235
D = E /((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];

% 形函数
syms m n o;
N = [1/8 * (1 - m) * (1 - n) * (1 - o);
     1/8 * (1 + m) * (1 - n) * (1 - o);
     1/8 * (1 + m) * (1 + n) * (1 - o);
     1/8 * (1 - m) * (1 + n) * (1 - o);
     1/8 * (1 - m) * (1 - n) * (1 + o);
     1/8 * (1 + m) * (1 - n) * (1 + o);
     1/8 * (1 + m) * (1 + n) * (1 + o);
     1/8 * (1 - m) * (1 + n) * (1 + o)];
N = N.';

% 8 个积分点等参坐标
xyz = [-1 / sqrt(3), -1 / sqrt(3), -1 / sqrt(3);
       1 / sqrt(3), -1 / sqrt(3), -1 / sqrt(3);
```

```
        1 / sqrt(3), 1 / sqrt(3), -1 / sqrt(3);
        -1 / sqrt(3), 1 / sqrt(3), -1 / sqrt(3);
        -1 / sqrt(3), -1 / sqrt(3), 1 / sqrt(3);
        1 / sqrt(3), -1 / sqrt(3), 1 / sqrt(3);
        1 / sqrt(3), 1 / sqrt(3), 1 / sqrt(3);
        -1 / sqrt(3), 1 / sqrt(3), 1 / sqrt(3);];

% 计算单元刚度矩阵 k_element 并组装系统刚度矩阵 k_system
k_system = sparse(3 * nodes_num, 3 * nodes_num);
for i = 1 : elements_num
    x = [];
    y = [];
    z = [];
    index = [];
    for j = 1 : 8
        x = [x; x_nodes(elements(i, j))];
        y = [y; y_nodes(elements(i, j))];
        z = [z; z_nodes(elements(i, j))];
        index = [index;
                elements(i, j) * 3 - 2;
                elements(i, j) * 3 - 1;
                elements(i, j) * 3 ];
    end

    % 求 3×3 J 矩阵
    JJ = zeros(3,3,8);% 8 个点的 J 矩阵
    dN = zeros(3,8,8);% 8 个点对等参坐标求导的矩阵
    for j = 1 : 8
        xx = xyz(j, 1);
```

```
    yy = xyz(j, 2);
    zz = xyz(j, 3);
    dN(:,:,j) = [double(subs(diff(N,m),{n, o}, {yy, zz}))
    double(subs(diff(N,n),{m, o}, {xx, zz}))
    double(subs(diff(N,o),{m, n}, {xx, yy}))];
    JJ(:, :, j) = dN(:,:,j) * [x, y, z];
end

%求 6×24 B 矩阵
BB = zeros(6,24,8);% 8 个点的 B 矩阵
for k = 1:8
    B = [];
    for j = 1 : 8
        dNk = dN(:,:,k);% 第 k 个点的 N 矩阵
        d = JJ(:,:,k) \ dNk(:,j);
        Bi = [d(1) 0 0;% 求 B 的子块
              0 d(2) 0;
              0 0 d(3);
              0 d(3) d(2) ;
              d(3) 0 d(1);
              d(2) d(1) 0];
        B = [B,Bi];
    end
    BB(:,:,k) = B;
end

% 计算单元刚度矩阵 k_element
k_element = zeros(24, 24);
```

```
    % 8 个点高斯积分，权重均为 1
    for j = 1 : 8
        k_element = k_element + BB(:, :, j).' * D * BB(:, :, j) *
abs(det(JJ(:, :, j)));
    end

    % 组装系统刚度矩阵 k_system
    for k = 1 : 24
        for j =  1 : 24
            k_system(index(k), index(j)) = k_system(index(k),
index(j)) + k_element(k, j);
        end
    end
end

% 位移边界条件
bcnode = [];
bcdof = [];
bcval = [];
for i = 1 : (lx + 1) * (ly + 1)
    bcnode = [bcnode; i];%边界条件的节点序号
    bcdof = [bcdof; i * 3 - 2; i * 3 - 1; i * 3];%边界点自由度序号
    bcval = [bcval; 0; 0; 0];%边界点自由度的值
end

% 系统 f 向量，最后一个点 x、y、z 方向均受力 1000 N
f_system = sparse(nodes_num * 3, 1);
f_system(nodes_num * 3 - 2) = 2000;
f_system(nodes_num * 3 - 1) = 1000;
```

```
f_system(nodes_num * 3 ) = 1000;

% 施加边界条件，置一法
for i = 1 : length(bcdof)
    c = bcdof(i);
    for j = 1 : nodes_num * 3
        k_system(c, j) = 0;
    end
    k_system(c, c) = 1;
    f_system(c) = bcval(i);
end

% LU 分解 + 高斯消元 计算 位移矩阵，K * d = f
k_system = full(k_system);
[L, U]=lu(k_system);% K = L * U
utemp = L \ f_system;% utemp = L^-1 * f
d_system = U \ utemp;% d = U^-1 * utemp

% 计算单元应力应变
stress_element = zeros(8, 6, elements_num);% 每个单元内应力不是常数
strain_element = zeros(8, 6, elements_num);% 每个单元内应变不是常数
for i = 1 : elements_num
    x = [];
    y = [];
    z = [];
    index = [];
    for j = 1 : 8
        x = [x; x_nodes(elements(i, j))];
        y = [y; y_nodes(elements(i, j))];
```

```
    z = [z; z_nodes(elements(i, j))];
    index = [index;
            elements(i, j) * 3 - 2;
            elements(i, j) * 3 - 1;
            elements(i, j) * 3 ];
end

%求 J 矩阵
JJ = zeros(3,3,8);% 每个点的 J 矩阵
dN = zeros(3,8,8);% 每个点对等参坐标求导的矩阵
count = 0;
for j = 1 : 8
    xx = xyz(j, 1);
    yy = xyz(j, 2);
    zz = xyz(j, 3);
    dN(:,:,j) = [double(subs(diff(N,m),{n, o}, {yy, zz}))
    double(subs(diff(N,n),{m, o}, {xx, zz}))
    double(subs(diff(N,o),{m, n}, {xx, yy}))];
    JJ(:, :, j) = dN(:,:,j) * [x, y, z];
end

%求 B 矩阵
BB = zeros(6,24,8);%每个点的 B 矩阵
for k = 1:8
    B = [];
    for j = 1 : 8
        dNk = dN(:,:,k);
        d = JJ(:,:,k) \ dNk(:,j);
        Bi = [d(1) 0 0;%求 B 的子块
```

```
            0 d(2) 0;
            0 0 d(3);
            d(2) d(1) 0;
            0 d(3) d(2);
            d(3) 0 d(1)];
        B = [B,Bi];
    end
    BB(:,:,k) = B;
end


d_element = [];% 单元位移矩阵
%计算 8 个积分点的应力
for j = 1 : 24
    d_element = [d_element; d_system(index(j))];
end
stress1 = [];% 8 个积分点的应力
for j = 1 : 8
    stress1 = [stress1; (D * BB(:, :, j) * d_element).'];
end

% 积分点应力->节点应力
stress2 = zeros(8, 6);% 8 个节点的应力
strain2 = zeros(8, 6);% 8 个节点的应变
for j = 1 : 8;
    xx = xyz(j, 1);
    yy = xyz(j, 2);
    zz = xyz(j, 3);
    NN = double(subs(N, {m, n, o}, {xx, yy, zz}));
```

```
        for k = 1 : 8
            % 节点的应力等于 积分点处形函数 * 积分点应力，再求和
            stress2(j, :) = stress2(j, :) + NN(k) * stress1(k, :);
        end
    end

    % 节点应变->节点应变
    for j = 1 : 8
        strain2(j, :) = (D \ (stress2(j, :)).').';
    end

    % 保存单元应力和应变
    stress_element(:, :, i) = stress2;
    strain_element(:, :, i) = strain2;
end

% 重合的节点应力和应变取平均值
stress_system = zeros(nodes_num, 6);
strain_system = zeros(nodes_num, 6);
count = zeros(nodes_num,1);% 重复次数
for i = 1 : elements_num
    for j = 1 : 8
        node = elements(i, j);
        stress_system(node, :) = strain_system(node, :) +
stress_element(j, :, i);
        strain_system(node, :) = strain_system(node, :) +
strain_element(j, :, i);
        count(node, 1) = count(node, 1) + 1;
    end
```

```
end
for i = 1 : nodes_num
    stress_system(i,:) = stress_system(i,:) / count(i);
    strain_system(i,:) = strain_system(i,:) / count(i);
end

% 输出数据导 techplot
fid_out=fopen('1000.plt','w');
fprintf(fid_out,'TITLE="test case governed by poisson
equation"\n');
fprintf(fid_out,'VARIABLES="x" "y" "z" "u" "v" "w" "sigax"
"sigmay" "sigmaz" "sigmaxy" "sigmayz" "sigmaxz"\n');
% ET=BRICK，设置为八节点四边形单元
fprintf(fid_out,'ZONE T="flow-field", N= %8d,E=%8d,ET=BRICK,
F=FEPOINT\n',nodes_num,elements_num);
d_system = full(d_system);
% 每个节点坐标、位移、应力
for i=1:nodes_num

fprintf(fid_out,'%16.6e %16.6e %16.6e %16.6e %16.6e %16.6e %16.6e
 %16.6e %16.6e %16.6e %16.6e %16.6e \n',x_nodes(i),y_nodes(i),
z_nodes(i), d_system(3*i-2), d_system(3*i - 1), d_system(3*i),
stress_system(i,1), stress_system(i,2), stress_system(i,3),
stress_system(i,4), stress_system(i,5), stress_system(i,6));
end
% 输出每个单元节点的序号
for i=1:elements_num

fprintf(fid_out,'%8d%8d%8d%8d%8d%8d%8d%8d\n',elements(i,1),ele
```

```
ments(i,2),elements(i,3),elements(i,4),elements(i,5),elements(
i,6),elements(i,7),elements(i,8));
end
```