湖南大學
HUNAN UNIVERSITY

四面体单元悬臂梁 MATLAB 有限元分析

学　　号：S230200222

姓　　名：张晓童

学　　院：机械与运载工程学院

专　　业：机械工程

班　　级：2312

授课教师：王 琥

提交时间　　2024 年　2 月

摘　要

本文主要是针对四面体单元进行 MATLAB 有限元编程，首先推导了局部坐标系下形函数对全局坐标系下的导数方程，运用弹性力学的相关知识进行单元刚度矩阵的运算、雅克比矩阵的推导。紧接着，求出节点位移之后，对位移结果进行后处理，运用几何方程求得节点应力应变。最后，利用了 patch 函数实现了四面体单元悬臂梁变形位移、应力云图的绘制。

关键词：四面体单元；形函数；单元刚度矩阵；

目　录

第 1 章　四面体单元

有限元法是求解工程问题的一种近似数值方法，近年来在工程领域中得到了广泛的应用。这种技术的应用依赖于对区域的离散，即对连续体进行离散化，这是数值解得到应用的关键，这种离散提供了几何简化单元集合作为整个复杂区域的近似，为使离散出的网格能更精确地逼近连续区域和有限元的结果在预定的误差范围内，应保证离散化后得到质量较高的网格，为此，提出了许多方法来生成有限元网格。

有限元法的基本步骤为：首先进行集合域的离散过程获得这个标准化的单元；然后通过能量原理，如去虚功原理或者是最小势能原理，获得单元的刚度方程；第三步为单元刚度集成装配，就是说把单元刚度集成成为整体刚度矩阵的一个过程；第四步为处理位移边界条件；第五步为计算位移场；第六步根据计算得到的位移场计算单元的其他物理量，比如应力应变。

这些步骤中核心的内容就是关于单元的研究，具体包括单元节点的描述，如在不同坐标系下，单元的节点坐标变化。这里的不同坐标系其实就是设计等参单元的过程，所涉及局部坐标系叫自然坐标系，然后它要和这个全局坐标系如整体直角坐标系进行节点坐标的变换。场的描述包括位移场、应力应变场的描述，可以用形函数描述应变场、应力场。

1.1 问题描述

如图 1 所示，有一悬臂梁，其左端完全约束，右端上侧节点受到向下的压力 100N。材料的弹性模量为 210000MPa，泊松比为 0.3。

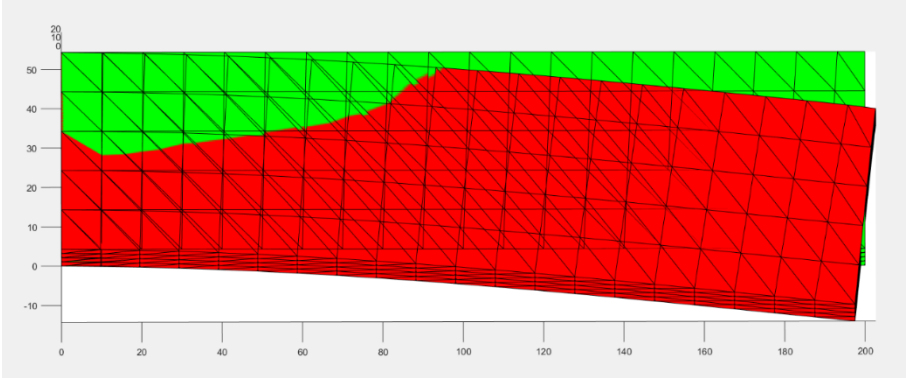

图 1 悬臂梁

1.2 空间弹性力学基本原理

弹性力学主要研究弹性体在外力作用或温度变化等外界因素下所产生的应力、应变和位移，从而解决结构或机械设计中所提出的强度和刚度问题。在连续性、完全弹性、均匀性、各向同性和小变形假定下，弹性力学问题化为线性问题。根据空间弹性问题可以推导出平衡方程、几何方程以及物理方程。在运用平衡方程、几何方程、物理方程的基础上即可以得出四面体等参单元的基本应力应变矩阵。

平衡方程为：

$$\begin{cases} \dfrac{\partial \sigma_{xx}}{\partial x}+\dfrac{\partial \tau_{xy}}{\partial y}+\dfrac{\partial \tau_{xz}}{\partial z}+\overline{b_x}=0 \\ \dfrac{\partial \sigma_{xy}}{\partial x}+\dfrac{\partial \tau_{yy}}{\partial y}+\dfrac{\partial \tau_{yz}}{\partial z}+\overline{b_y}=0 \\ \dfrac{\partial \sigma_{xz}}{\partial x}+\dfrac{\partial \tau_{yz}}{\partial y}+\dfrac{\partial \tau_{zz}}{\partial z}+\overline{b_z}=0 \end{cases} \tag{1-1}$$

几何方程为：

$$\begin{cases} \varepsilon_{xx}=\dfrac{\partial u}{\partial x}, \quad \varepsilon_{yy}=\dfrac{\partial v}{\partial y}, \quad \varepsilon_{zz}=\dfrac{\partial w}{\partial z} \\ \gamma_{xy}=\dfrac{\partial v}{\partial x}+\dfrac{\partial u}{\partial y}, \quad \gamma_{yz}=\dfrac{\partial w}{\partial y}+\dfrac{\partial v}{\partial z}, \quad \gamma_{zx}=\dfrac{\partial w}{\partial x}+\dfrac{\partial u}{\partial z} \end{cases} \tag{1-2}$$

物理方程为：

$$\begin{Bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \tau_{xy} \\ \tau_{yz} \\ \tau_{xz} \end{Bmatrix} = \begin{bmatrix} 1-v & v & v & 0 & 0 & 0 \\ v & 1-v & v & 0 & 0 & 0 \\ v & v & 1-v & 0 & 0 & 0 \\ 0 & 0 & 0 & \dfrac{1-2v}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \dfrac{1-2v}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \dfrac{1-2v}{2} \end{bmatrix} \begin{Bmatrix} \varepsilon_{xxx} \\ \varepsilon_{yyy} \\ \varepsilon_{zz} \\ \varepsilon_{xy} \\ \varepsilon_{yz} \\ \varepsilon_{xz} \end{Bmatrix} \cdot \frac{E}{(1+v)(1-2V)} \tag{1-3}$$

将物理方程简化为：

$$\{\sigma\}=[D]\{\varepsilon\} \tag{1-4}$$

其中，σ代表应力矩阵，ε代表应变矩阵，D代表系数矩阵，E代表弹性模量。

由几何方程可知，同一点的应力状态情况一样，可以证明，在物体内任意一点，若已知ε_{xx}，ε_{yy}，ε_{zz}，γ_{xy}，γ_{yz}，γ_{zx}即可求得经过该点的任意截面上（方向

余弦已知）的正应变和切应变。故这六个应变分量完全确定了该点的形变状态。因此，当弹性体的位移分量完全确定时，应变分量是完全确定的。反过来，当应变分量完全确定时，位移分量却不完全确定；这是因为，具有确定形状的物体，可能发生不同的刚体位移。

1.3 四面体单元体积坐标

在有限元方法中，若要将这个离散的边界，可以离散为这个曲线或者曲面的求解域，需要将这种形状规则的比如立方体、正三角形等这些形状规则的这个单元变换为边界为曲线或者是曲面的单元。

这是一个变化的过程，不能用非常标准的这种立方体单元来离散边界。四面体单元的参数坐标其实就是它的体积坐标，我们就是要把他的参数坐标和这个直角坐标进行相互映射，其实就是全局坐标转换。

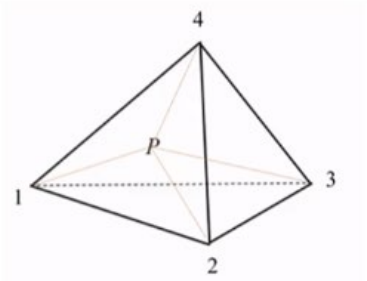

图 2 四面体节点编号

有限元分析中的四面体单元如图 2 所示，将一个完整的结构看成是由有限个四面体单元组成的实体，对单元及其节点位移的求解就是有限元分析的最终目的。在分析四面体有限元时，我们第一步需要做的是对四面体单元进行节点编号，如图 2 所示。编号后的节点及其坐标如下式所示：

$$L_1 = \frac{vol(P234)}{vol(1234)} \tag{1-5}$$

$$L_2 = \frac{vol(P134)}{vol(1234)} \tag{1-6}$$

$$L_3 = \frac{vol(P124)}{vol(1234)} \tag{1-7}$$

$$L_4 = \frac{vol(P123)}{vol(1234)} \tag{1-8}$$

$$L_1 + L_2 + L_3 + L_4 = 1 \tag{1-9}$$

其参数坐标下的形函数即为体积坐标，可表示为：

$$N_i^* = L_i \ (i = 1, 2, 3, 4) \tag{1-10}$$

上面的公式中，L_i分别表示各部分体积与单元体积的比值，其中，$i=1,2,3,4$。

接下来推导四面体单元的单元刚度矩阵，单元刚度矩阵涉及到局部坐标系和全局坐标系的变换，全局坐标系下的单元刚度矩阵如下所示：

$$K_e=\iiint_{\Omega_e} B(x,y,z)^T\,DB(x,y,z)dx\,dy\,dz \tag{1-11}$$

K_e代表四面体单元的刚度矩阵，矩阵$B(x,y,z)$代表位移与应变的传递矩阵，矩阵D代表应力与应变的传递矩阵。进一步地，根据全局坐标系下的单元刚度矩阵进行坐标变换后可以得到局部坐标系下的单元刚度矩阵：

$$\begin{aligned}K_e&=\int_0^1\int_0^{1-L_4}\int_0^{1-L_3-L_4} B^{*T}DB^*|J|dL_2\,dL_3\,dL_4\\&=\int_0^1\int_0^{1-L_4}\int_0^{1-L_3-L_4} B^{*T}(L_1,L_2,L_3,L_4)DB^*(L_1,L_2,L_3,L_4)|J|dL_2\,dL_3\,dL_4\\&=\frac{1}{6}B^{*T}\left(\frac{1}{4},\frac{1}{4},\frac{1}{4},\frac{1}{4}\right)DB^*\left(\frac{1}{4},\frac{1}{4},\frac{1}{4},\frac{1}{4}\right)|J|\end{aligned} \tag{1-12}$$

1.4 局部坐标系下形函数对全局坐标的导数

在 1.2 中推导了四面体单元的刚度矩阵求解框架，接下来我们需要分别知道全局坐标系下的位移应变传递矩阵B、等参坐标系下的位移应变传递矩阵B^*和雅可比矩阵J。

其中B^*矩阵可以表示为：

$$B^*=\begin{bmatrix}\frac{\partial N_1^*}{\partial x} & 0 & 0 & \cdots & \frac{\partial N_4^*}{\partial x} & 0 & 0\\ 0 & \frac{\partial N_1^*}{\partial y} & 0 & \cdots & 0 & \frac{\partial N_4^*}{\partial y} & 0\\ 0 & 0 & \frac{\partial N_1^*}{\partial z} & \cdots & 0 & 0 & \frac{\partial N_4^*}{\partial z}\\ \frac{\partial N_1^*}{\partial y} & \frac{\partial N_1^*}{\partial x} & 0 & \cdots & \frac{\partial N_4^*}{\partial y} & \frac{\partial N_4^*}{\partial x} & 0\\ 0 & \frac{\partial N_1^*}{\partial z} & \frac{\partial N_1^*}{\partial y} & \cdots & 0 & \frac{\partial N_4^*}{\partial z} & \frac{\partial N_4^*}{\partial y}\\ \frac{\partial N_1^*}{\partial z} & 0 & \frac{\partial N_1^*}{\partial x} & \cdots & \frac{\partial N_4^*}{\partial z} & 0 & \frac{\partial N_4^*}{\partial x}\end{bmatrix}_{6\times 12} \tag{1-13}$$

在公式（1-13）中，其中N^*为在局部坐标系下的形函数，但是却需要对全局坐标系(x,y,z)进行求偏导，因此需要进行一系列的坐标转换。首先第一步需要构造一个两个坐标系的链式求导法则，该链式求导法则表达式如下所示：

$$\begin{aligned}\frac{\partial N_i^*}{\partial \xi}&=\frac{\partial N_i^*}{\partial x}\frac{\partial x}{\partial \xi}+\frac{\partial N_i^*}{\partial y}\frac{\partial y}{\partial \xi}+\frac{\partial N_i^*}{\partial z}\frac{\partial z}{\partial \xi}\\ \frac{\partial N_i^*}{\partial \eta}&=\frac{\partial N_i^*}{\partial x}\frac{\partial x}{\partial \eta}+\frac{\partial N_i^*}{\partial y}\frac{\partial y}{\partial \eta}+\frac{\partial N_i^*}{\partial z}\frac{\partial z}{\partial \eta}\\ \frac{\partial N_i^*}{\partial \zeta}&=\frac{\partial N_i^*}{\partial x}\frac{\partial x}{\partial \zeta}+\frac{\partial N_i^*}{\partial y}\frac{\partial y}{\partial \zeta}+\frac{\partial N_i^*}{\partial z}\frac{\partial z}{\partial \zeta}\end{aligned} \tag{1-14}$$

将公式（1-14）转换为矩阵形式可以得到：

$$\begin{bmatrix}\frac{\partial N_i^*}{\partial \xi}\\ \frac{\partial N_i^*}{\partial \eta}\\ \frac{\partial N_i^*}{\partial \zeta}\end{bmatrix}=\begin{bmatrix}\frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} & \frac{\partial z}{\partial \xi}\\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} & \frac{\partial z}{\partial \eta}\\ \frac{\partial x}{\partial \zeta} & \frac{\partial y}{\partial \zeta} & \frac{\partial z}{\partial \zeta}\end{bmatrix}\begin{bmatrix}\frac{\partial N_i^*}{\partial x}\\ \frac{\partial N_i^*}{\partial y}\\ \frac{\partial N_i^*}{\partial z}\end{bmatrix} \tag{1-15}$$

链式求导表达式确定后，求解目标变为已知公式（1-15）中最左侧的矩阵，需要我们求解公式（1-15）中最右侧的矩阵。一个完整的有限元程序基本组成部分包括前处理，分析主程序，还有后处理。在前处理模块中实现的是这个节点的坐标、单元的节点编号、网格划分以及边界条件这些输入，然后在分析的主程序模块的是求解整体刚度方程，也就是$ku=p$的刚度方程，然后在后处理模块中就是实现结果的展示。因此求解局部坐标系下的插值函数对全局坐标系下的偏导数成为了关键的一环。要想通过链式法则求出该偏导数，首先应该求解出公式（1-15）中间的矩阵，这个矩阵即为之前提到的雅可比矩阵J，因此，雅可比矩阵可以表示为：

$$J=\begin{bmatrix}\frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} & \frac{\partial z}{\partial \xi}\\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} & \frac{\partial z}{\partial \eta}\\ \frac{\partial x}{\partial \zeta} & \frac{\partial y}{\partial \zeta} & \frac{\partial z}{\partial \zeta}\end{bmatrix} \tag{1-16}$$

现在开始求解雅可比矩阵J，首先利用几何场插值法。几何场插值法的表达式为：

$$\begin{cases}x=\sum_{i=1}^{4}N_i^*x_i\\ y=\sum_{i=1}^{4}N_i^*y_i\\ z=\sum_{i=1}^{4}N_i^*z_i\end{cases} \tag{1-17}$$

将公式（1-17）代入雅可比矩阵内可以得到：

$$J=\begin{bmatrix}\sum_{i=1}^{4}\frac{\partial N_i^*}{\partial\xi}x_i & \sum_{i=1}^{4}\frac{\partial N_i^*}{\partial\xi}y_i & \sum_{i=1}^{4}\frac{\partial N_i^*}{\partial\xi}z_i\\ \sum_{i=1}^{4}\frac{\partial N_i^*}{\partial\eta}x_i & \sum_{i=1}^{4}\frac{\partial N_i^*}{\partial\eta}y_i & \sum_{i=1}^{4}\frac{\partial N_i^*}{\partial\eta}z_i\\ \sum_{i=1}^{4}\frac{\partial N_i^*}{\partial\zeta}x_i & \sum_{i=1}^{4}\frac{\partial N_i^*}{\partial\zeta}y_i & \sum_{i=1}^{4}\frac{\partial N_i^*}{\partial\zeta}z_i\end{bmatrix} \tag{1-18}$$
$$=\begin{bmatrix}\frac{\partial N_1^*}{\partial\xi} & \frac{\partial N_2^*}{\partial\xi} & \frac{\partial N_3^*}{\partial\xi} & \frac{\partial N_4^*}{\partial\xi}\\ \frac{\partial N_1^*}{\partial\eta} & \frac{\partial N_2^*}{\partial\eta} & \frac{\partial N_3^*}{\partial\eta} & \frac{\partial N_4^*}{\partial\eta}\\ \frac{\partial N_1^*}{\partial\zeta} & \frac{\partial N_2^*}{\partial\zeta} & \frac{\partial N_3^*}{\partial\zeta} & \frac{\partial N_4^*}{\partial\zeta}\end{bmatrix}\begin{bmatrix}x_1 & y_1 & z_1\\ x_2 & y_2 & z_2\\ x_3 & y_3 & z_3\\ x_4 & y_4 & z_4\end{bmatrix}$$

通过上述方法求得了雅可比矩阵J，因此同样也求解出了雅可比矩阵J的逆矩阵。接下来可以进行求偏导数的换元：

$$\begin{Bmatrix}\frac{\partial N_i^*}{\partial x}\\ \frac{\partial N_i^*}{\partial y}\\ \frac{\partial N_i^*}{\partial z}\end{Bmatrix}=J^{-1}\begin{Bmatrix}\frac{\partial N_i^*}{\partial\xi}\\ \frac{\partial N_i^*}{\partial\eta}\\ \frac{\partial N_i^*}{\partial\zeta}\end{Bmatrix} \tag{1-19}$$

通过上面的分析，我们就求出了B^*中的插值函数对全局坐标系的偏导数，写为矩阵形式后表达式如下：

$$B^*=\begin{bmatrix}\frac{\partial N_1^*}{\partial x} & 0 & 0 & \cdots & \frac{\partial N_4^*}{\partial x} & 0 & 0\\ 0 & \frac{\partial N_1^*}{\partial y} & 0 & \cdots & 0 & \frac{\partial N_4^*}{\partial y} & 0\\ 0 & 0 & \frac{\partial N_1^*}{\partial z} & \cdots & 0 & 0 & \frac{\partial N_4^*}{\partial z}\\ \frac{\partial N_1^*}{\partial y} & \frac{\partial N_1^*}{\partial x} & 0 & \cdots & \frac{\partial N_4^*}{\partial y} & \frac{\partial N_4^*}{\partial x} & 0\\ 0 & \frac{\partial N_1^*}{\partial z} & \frac{\partial N_1^*}{\partial y} & \cdots & 0 & \frac{\partial N_4^*}{\partial z} & \frac{\partial N_4^*}{\partial y}\\ \frac{\partial N_1^*}{\partial z} & 0 & \frac{\partial N_1^*}{\partial x} & \cdots & \frac{\partial N_4^*}{\partial z} & 0 & \frac{\partial N_4^*}{\partial x}\end{bmatrix}_{6\times12} \tag{1-20}$$

1.5 应力应变矩阵的求解

现在我们已经求解出了B^*矩阵与雅可比矩阵J，接下来要求解出应力应变矩阵D。矩阵D的表达式如下所示：

$$D=\frac{E}{(1+\mu)(1-2\mu)}\times \begin{bmatrix} 1-\mu & \mu & \mu & 0 & 0 & 0 \\ \mu & 1-\mu & \mu & 0 & 0 & 0 \\ \mu & \mu & 1-\mu & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1-2\mu}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1-2\mu}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1-2\mu}{2} \end{bmatrix}_{6\times 6} \tag{1-21}$$

在公式（1-21）中E代表材料的弹性模量，μ代表材料的泊松比。得到应力应变矩阵后的有限元分析思路如下的流程图所示：

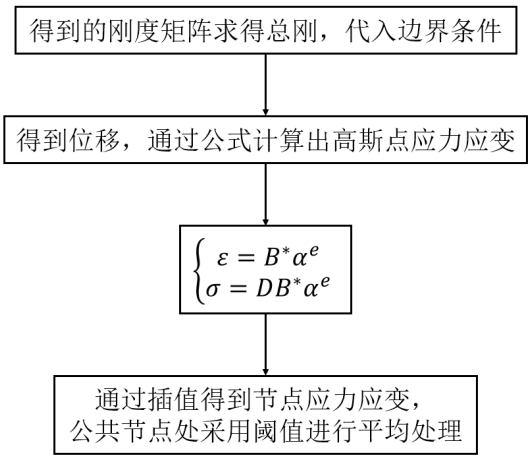

图 3 有限元分析思路流程图

所谓的平均处理是为了便于后续画图应力应变的云图，在公共节点处，由于其处于在两个及以上的单元内，这时可能存在不同的单元内计算出的这一节点的应变应力不同的情况，此时需要采取平均处理，即取这多个单元内的计算出的应力与应变，计算出其平均之后作为最终的应力与应变。

第 2 章 MATLAB 程序及结果展示

2.1 程序框架解读

该四面体单元有限元分析 MATLAB 代码框架如图 4 所示。

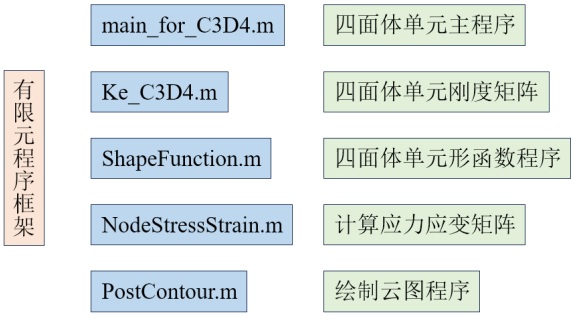

图 4 程序框架

主函数为 main_for_C3D4.m，主程序内定义了有限元分析的最基本参数，如弹性模量，泊松比等关键信息。Ke_C3D4.m 程序则定义了求解四面体单元的刚度矩阵，随后在 ShapeFunction.m 程序内定义了相关的形函数，最后求解应力应变矩阵。该程序求解出应力应变后即可进行绘制云图，程序详细说明及结果在下文中有详细说明。

2.2 MATLAB 程序说明

主程序除了定义相关基本参数外，还读取 nodes_and_elements.mat 文件，获得节点坐标信息，单元信息，外力矩阵约束节点编号以及约束矩阵。本次有限元分析的四面体取杨氏模量 210000MPa，泊松比为 0.3。

```
clear all;
clc;
close all;
E=210000; %弹性模量
u=0.3;    %泊松比
D=E/((1+u)*(1-2*u))*[1-u u u 0 0 0;......
    u 1-u u 0 0 0;......
    u u 1-u 0 0 0;......
    0 0 0 (1-2*u)/2 0 0;......
    0 0 0 0 (1-2*u)/2 0;......
    0 0 0 0 0 (1-2*u)/2];%elastic modules matrix
load nodes_and_elements.mat
Forces=[7 2 -100;10 2 -100;12 2 -100;13 2 -100;14 2 -100;]; %[nodeID direction value]
ConID=[2,   3,   5,   8,  36,  37,  38,  61,  62,  63,  64,  87,  88,  89,  90,  91.....
  92,  93, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260];
Constraints=zeros(size(ConID,2)*3,3);
```

图 5 读取 mat 文件

ShapeFunction.m 文件定义了一阶四面体单元形函数。JacobiDET 用于求解单刚。如图 6 所示。

```
function [NDxyz,JacobiDET] = ShapeFunction(ElementNodeCoordinate)
%计算形函数及形函数对局部坐标ksi eta zeta的导数
NDL = [-1 1 0 0;-1 0 1 0;-1 0 0 1];%3*4  [N1Dksi N2Dksi N3Dksi N4Dksi;N1Deta N2Deta N3Deta N4Deta……]
Jacobi = NDL*ElementNodeCoordinate;%计算雅可比矩阵3*4 4*3
JacobiDET = det(Jacobi);%计算雅可比行列式3*3 [DxDksi DyDksi DzDksi;DxDeta……
JacobiINV=inv(Jacobi);%对雅可比行列式求逆3*3
NDxyz=JacobiINV*NDL;%利用雅可比行列式的逆计算形函数对结构坐标的导数[DN1Dx DN2Dx DN3Dx;DN1Dy DN2Dy DN3Dy;……]
end
```

图 6 四面体单元形函数

在主程序中还进行了整体刚度矩阵组装，施加外力载荷，利用乘大数法施加约束以及位移求解。其对应的程序分别如图 7、图 8、图 9 所示。

```
for I=1:ElementNum
    % 单元节点坐标
    ElementNodeCoordinate=Nodes(Elements(I,:),:);
    % 计算单元刚度矩阵
    ElementStiffnessMatrix=Ke_C3D4(D,ElementNodeCoordinate);
    % 计算单元节点自由度编号
    ElementNodeDOF=zeros(1,12);
    for J=1:4
        II=(J-1)*Dof+1;
        ElementNodeDOF(II:II+2)=(Elements(I,J)-1)*Dof+1:(Elements(I,J)-1)*Dof+3;
    end
    %整体刚度矩阵组装，根据单元节点自由度组装总刚
    K(ElementNodeDOF,ElementNodeDOF)=K(ElementNodeDOF,ElementNodeDOF)+ElementStiffnessMatrix;
end
```

图 7 整体刚度矩阵组装

```
% 施加外力
if size(Forces,1)>0
    %计算外力自由度编号,第一列为节点号，第二列为方向编号
    ForceDOF = Dof*(Forces(:,1)-1)+Forces(:,2);
    F(ForceDOF) = F(ForceDOF) + Forces(:,3);
end
```

图 8 施加外力载荷

```
% 乘大数法（罚函数法）施加位移约束
BigNumber=1e8;
ConsNum=size(Constraints,1);
if ConsNum~=0
    FixedDof=Dof*(Constraints(:,1)-1)+Constraints(:,2);  %被约束的自由度编号(列向量)
    for i=1:ConsNum
        K(FixedDof(i),FixedDof(i))=K(FixedDof(i),FixedDof(i))*BigNumber;
        F(FixedDof(i))=Constraints(i,3)*K(FixedDof(i),FixedDof(i));
    end
end
```

图 9 乘大数法施加位移约束

在接下来的 NodeStressStrain.m 程序中，定义了计算应力应变的函数。其中包括了节点的应变、节点的应力、高斯点的应变与高斯点的应力。计算应力应变

程序通过位移矩阵求解高斯点应力应变，再采用应力磨平，构造出节点插值函数，得到节点处的应力应变。

对于公共节点处的唯一采用的是阈值平均法，当不同单元内计算出的节点的应力应变差别不大时，应力应变可以采用取平均值的做法，但当不同单元内计算同一节点的应力应变差别过大，则是一种不合理的结果，这时需要考虑单元格划分的方式以及节点定义是否合理。计算应力应变函数如图 10 所示。

```
function [NodeStrain,NodeStress]=NodeStressStrain(U,D,Nodes,Elements)
ElementNum= size(Elements,1);  %单元个数
Dof=3;
NodeStrain=zeros(6,ElementNum*4);%11 22 33 12 13 23
NodeStress=zeros(6,ElementNum*4);
Strain = zeros(6,1);
Stress = zeros(6,1);
%循环组装总刚
for i=1:ElementNum
    % 单元节点坐标
    ElementNodeCoordinate=Nodes(Elements(i,:),:); %4*3
    % 计算单元节点自由度编号,即索引位置
    ElementNodeDOF=zeros(1,12);%3*4
    for j=1:4
        ii=(j-1)*Dof+1;
        %单元各节点在总刚度矩阵中的位置索引
        ElementNodeDOF(ii:ii+2)=(Elements(i,j)-1)*Dof+1:(Elements(i,j)-1)*Dof+3;
    end
     % 计算形函数导数
    [NDxyz, ~] = ShapeFunction(ElementNodeCoordinate);%[DN1Dx DN2Dx DN3Dx;DN1Dy DN2Dy DN3Dy;……]
    ElementNodeDisplacement=U(ElementNodeDOF);%12*1 节点位移列阵
    ElementNodeDisplacement=reshape(ElementNodeDisplacement,Dof,4);%3*4
    % 计算单元应变 Strain3_3 3*3的应变矩阵
    Strain3_3=ElementNodeDisplacement*NDxyz';%高斯积分点处应变 3*4  4*3

    %把单元应变矩阵改写成6*1
    Strain=[Strain3_3(1,1) Strain3_3(2,2) Strain3_3(3,3) ...
    Strain3_3(1,2)+Strain3_3(2,1)....
    Strain3_3(2,3)+Strain3_3(3,2) Strain3_3(1,3)+Strain3_3(3,1)]';
    Stress(1:6,1) = D*Strain;%高斯积分点处应变
    %求得节点应力应变
    for X = 1:4
        NodeStrain(1:6,((i-1)*4+X)) = Strain(1:6,1);%按照各单元4节点依次排列 1号单元4节点；2号单元4节点
        NodeStress(1:6,((i-1)*4+X)) = Stress(1:6,1);
    end
end
end
```

图 10 计算应力应变程序

在绘制云图程序中，对于 C3D8 三维单元，顺序给定点的坐标与相应的值，以及每个面上点的连接顺序，将数据信息放入元胞数组，并用 patch 函数绘制三维图像。

PostContour.m 程序为绘制云图程序。其中(X,Y,Z)分别表示在全局坐标系下节点(x,y,z)方向的坐标值，Value 值对应的值，这一值可以为位移、应变或者应力，将 Value 值赋予在 mat 文件中定义好的每一个单元节点。除此以外还需要对

不同单元的同一节点进行磨平处理。绘制云图程序如图 11 所示。

```
function PostContour(Nodes,Elements,U,Component_val)
NodeNum = size(Nodes,1) ;  %  节点个数
ElementNum = size(Elements,1) ;    %单元个数
ElementNodeNum=4;  %每个单元节点数
% 矩阵初始化，X Y Z点的坐标；value点的值，对每个单元按照节点序号依次绘制云图
X = zeros(ElementNodeNum,ElementNum) ;%4*单元个数；各单元各节点的横坐标
Y = zeros(ElementNodeNum,ElementNum) ;
Z = zeros(ElementNodeNum,ElementNum) ;
value = zeros(ElementNodeNum,ElementNum) ;
```

```
%判断矩阵类型（位移，应力，应变）
if size(Component_val,1)>1%位移
    for i=1:ElementNum
        nd=Elements(i,:);
        value(:,i) = Component_val(nd) ;
    end
else %应力应变
    %先进行磨平（avg 积分点插值到节点处再取平均）
    Difference=max(Component_val)-min(Component_val);%全域上的最大值-最小值
    AVG=0.75;  % 默认阈值75%
    for i=1:1:NodeNum %遍历节点，进行应力磨平
        TElements=Elements';%转置Elements
        itemp=(TElements==i);%进行逻辑判断，itemp:元素为0 1的矩阵
        Cut=max(Component_val(1,itemp))-min(Component_val(1,itemp));%该节点的应力（应变）差
        if 0<Cut&&Cut<=AVG*Difference(1)%判断是否满足阈值条件
            Component_val(1,itemp)=mean(Component_val(1,itemp));%进行应力（应变）磨平
        end
    end
    value=reshape(Component_val,ElementNodeNum,ElementNum);%将Component的值赋给value
end
```

```
% 绘制变形后云图
newNodes=Nodes';
newNodes=newNodes(:);
DeformationCoefficient=5.0e2; %变形放大系数
newNodes=newNodes+DeformationCoefficient*U;
newNodes=reshape(newNodes,[3,size(Nodes,1)]);
newNodes=newNodes';
% 定义单元每个面的节点顺序(顺时针或逆时针)
if ElementNodeNum == 4  % C3D4单元
    fm = [1 2 4;2 3 4;3 1 4;1 3 2];
elseif ElementNodeNum==6 %C3D6单元
    fm=[1 2 3 1;4 5 6 4;1 2 5 4;1 3 6 4;2 3 6 5];
elseif ElementNodeNum == 8  % C3D8单元
    fm = [1 2 6 5; 2 3 7 6; 3 4 8 7; 4 1 5 8; 1 2 3 4; 5 6 7 8];
elseif ElementNodeNum == 20  % C3D20单元
    fm = [1,9,2,10,3,11,4,12;5,13,6,14,7,15,8,16;1,9,2,18,6,13,5,17;
        2,10,3,19,7,14,6,18;3,11,4,20,8,15,7,19;1,17,5,16,8,20,4,12];
end
xyz = cell(1,ElementNum) ;
profile = xyz ;
```

图 11(a) 绘制云图程序

```
%将节点位置赋予其对应值
for e=1:ElementNum %循环获取每个单元内节点的坐标
    nd=Elements(e,:);
    X = newNodes(nd,1) ;
    Y = newNodes(nd,2) ;
    Z = newNodes(nd,3) ;
    xyz{e} = [X  Y Z] ;
    profile{e} = value(:,e);
end
```

```
% 采用patch函数进行绘图
figure
cellfun(@patch,repmat({'Vertices'},1,ElementNum),xyz,.....%用多边形面片的方式显示各个面
    repmat({'Faces'},1,ElementNum),repmat({fm},1,ElementNum),.....
    repmat({'FaceVertexCdata'},1,ElementNum),profile,.....
    repmat({'FaceColor'},1,ElementNum),repmat({'interp'},1,ElementNum));
view(3);
rotate3d on;
axis off; %不显示坐标轴
colormap(jet);
caxis([min(Component_val),max(Component_val)]);
t1=caxis;
t1=linspace(t1(1),t1(2),13);
colorbar('ytick',t1,'Location','westoutside');
axis equal;
end
```

图 11(b) 绘制云图程序

在主程序中，完成了结果输出功能，输出了原始网格和变形网格叠加图以及相关应力应变图片与标题。结果输出程序如图 12 所示。

```
% 输出结果
% 计算高斯点、节点应力应变值
[NodeStrain,NodeStress]=NodeStressStrain(U,D,Nodes,Elements);
% 求得MISES应力矩阵
ElementNodeCount=4; % 每个单元节点数
MISES=zeros(1,ElementNum*ElementNodeCount);
for I=1:ElementNum*ElementNodeCount
    MISES(I)=sqrt(0.5)*sqrt((NodeStress(1,I)-NodeStress(2,I))^2+(NodeStress(1,I)-NodeStress(3,I))^2+....
        (NodeStress(2,I)-NodeStress(3,I))^2+6*(NodeStress(4,I)^2+NodeStress(5,I)^2+NodeStress(6,I)^2));
end

newNodes=Nodes';
newNodes=newNodes(:);
SF=5.0e2; % 变形放大系数
newNodes=newNodes+SF*U;
newNodes=reshape(newNodes,[3,size(Nodes,1)]);
newNodes=newNodes';
```

图 12 结果输出程序

```
% 绘制原始网格和变形网格叠加图
  for i=1:1:size(Elements,1)
    points=Nodes(Elements(i,:),:);
    deformed_points=newNodes(Elements(i,:),:);
    mesh=1:1:4;% 网格信息
    % 四面体单元节点坐标
    vertices_matrix = [points(mesh(1,:),1),points(mesh(1,:),2),points(mesh(1,:),3)];
    deformed_vertices_matrix=[deformed_points(mesh(1,:),1),deformed_points(mesh(1,:),2),deformed_points(mesh(1,:),3)]
    % 四面体单元节点顺序
    faces_matrix= [1 2 4;2 3 4;3 1 4;1 3 2];% 给出每个面的节点序号
    patch('vertices', vertices_matrix,'faces',faces_matrix,'facecolor','g','FaceAlpha',.5);
    view(3);hold on% 绘图
    patch('vertices', deformed_vertices_matrix,'faces',faces_matrix,'facecolor','r','FaceAlpha',.5);

  end
axis equal
view(3);
alpha(1);
```

图 13 绘制原始网格和变形网格叠加图

```
% 绘制Umag云图
Umag=zeros(NodeNum,1);
for i=1:NodeNum
    Umag(i)=sqrt(U(3*i-2)^2+U(3*i-1)^2+U(3*i)^2);
end
PostContour(Nodes,Elements,U,Umag)
title('位移')
% 绘制MISES云图
PostContour(Nodes,Elements,U,MISES)
title('应力')
```

图 14 绘制应力应变云图

2.3 结果呈现

原始网格和变形网格叠加图如图 15 所示。位移云图与应力云图如图 16 和图 17 所示。

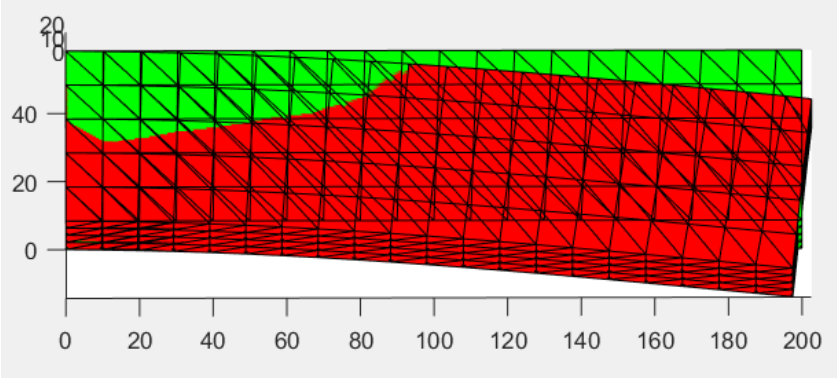

图 15 原始网格和变形网格叠加图

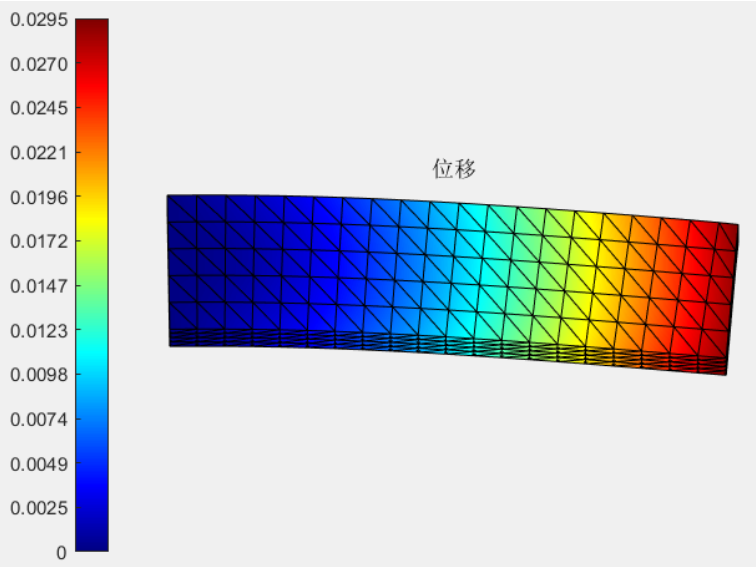

图 16 位移云图

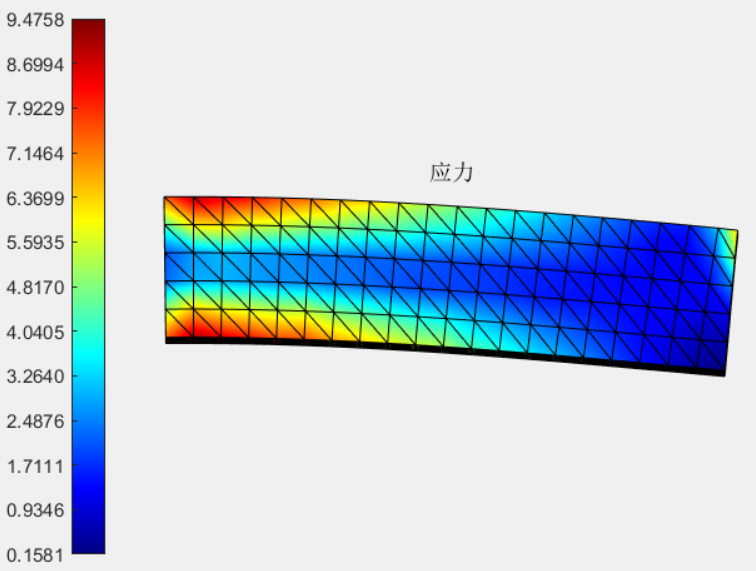

图 17 应力云图

2.4 总结

本文主要是针对四面体单元进行 MATLAB 有限元编程，首先推导了局部坐标系下形函数对全局坐标系下的导数方程，运用弹性力学的相关知识进行单元刚度矩阵的运算、雅克比矩阵的推导。紧接着，求出节点位移之后，对位移结果进行后处理，运用几何方程求得节点应力应变。最后，利用了 patch 函数实现了悬臂梁原始网格和变形网格叠加图、位移、应力云图的绘制。