湖南大學
HUNAN UNIVERSITY

实用有限元分析期末大作业

采用静力分析求解位移及应力问题

姓　　名	游婉婷
学 科 专 业	机械工程
学　　号	S230200182
研 究 方 向	磁脉冲管件焊接集磁器优化
学　　院	机械与运载工程学院
上 课 教 师	王琥

目录

一、问题描述

由于我的研究方向为管件磁脉冲焊接集磁器结构优化，在之前的研究中初步猜想将集磁器中间部分挖空，探究其成型效果，所以在该报告中研究对象为中间挖空一部分的铜制集磁器。

在电磁脉冲焊接工艺过程中，如图 1.1 所示，利用线圈装置的瞬间放电，线圈内的金属管感应产生巨大的洛伦兹力，使外管以高速撞击内管达到高强度的连接效果。同时，外管也会给集磁器一个反作用力，利用 MATLAB 程序模拟计算集磁器受到洛伦兹力的变形、应力及应变结果。集磁器结构示意图如图 1.2 所示。

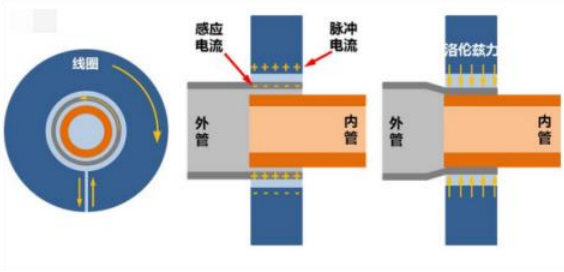


图 1.1 电磁脉冲焊接原理示意图

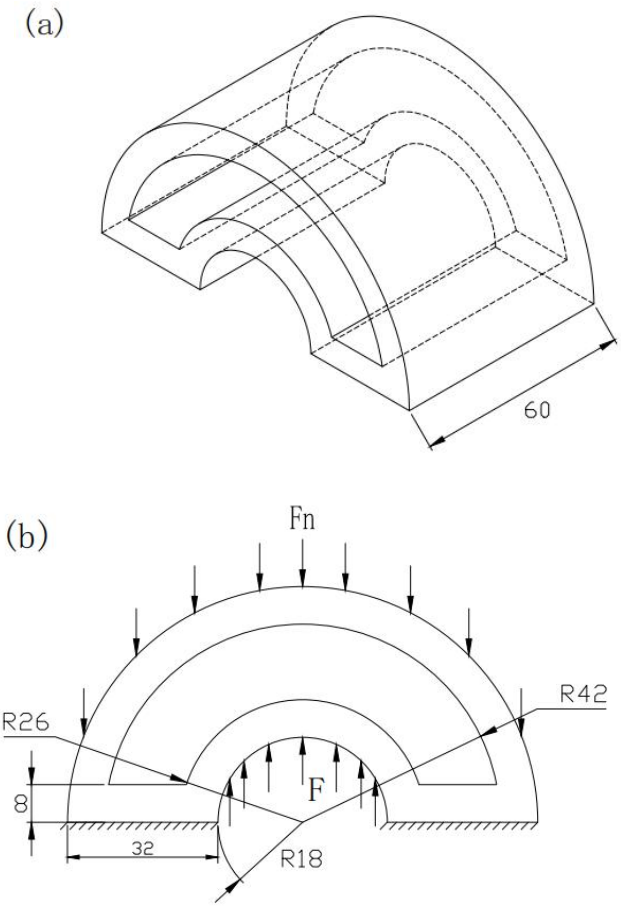


图 1.2 集磁器结构示意图

（a）结构等轴测视图 （b）横截面受力图

问题描述如下：集磁器结构尺寸大小与受力情况均标注在图（b）中，集磁器顶部受洛伦兹力均布力 Fn 大小为 500N，方向竖直向下；底部受支撑均布力 F 大小为 700N，方向竖直向上，底面为固定面。材料弹性模量 E=128GPa、泊松比 v=0.3。忽略重力影响，编写程序计算集磁器的位移和应力大小及其分布。

二、作业文件夹介绍

2.1 code 文件夹

包含主函数及其余 19 个自定义函数，这 19 个自定义函数中存在嵌套调用，函数的具体内容和详细注释见附件。

2.2 input 文件夹

1）mesh.k 文件：由 Hypermesh 导出，包含模型*NODE 及*ELEMENT_SOLID 关键字，包括网格编号及对应的 8 个节点坐标和所有节点编号及 x、y、z 坐标；

2）集磁器尺寸及受力示意图：表示集磁器的尺寸数值以及受力

2.3 result 文件夹

1）result_d.plt 文件：表示模型各节点 x、y、z 方向位移求解结果；

2）result_Sx.plt 文件：表示模型各节点 x 轴法向应力求解结果；

3）result_Sy.plt 文件：表示模型各节点 y 轴法向应力求解结果；

4）result_Sz.plt 文件：表示模型各节点 z 轴法向应力求解结果；

5）result_Sxy.plt 文件：表示模型各节点 xy 方向切应力求解结果；

6）result_Syz.plt 文件：表示模型各节点 yz 方向切应力求解结果；

7）result_Sxz.plt 文件：表示模型各节点 xz 方向切应力求解结果；

8）横截面变形示意图：展示模型横截面网格受载前后位置；

2.4 model 文件夹

包含集磁器的立体结构，有用于软件仿真的.x_t 格式和.SLDPRT 格式

三、八节点六面体单元基本理论

3.1 单元理论

六面体单元有根据不同方法有两种不同处理方式：

（1）一种是建立形函数，使用空间六面体的等参元：正方体单元做理论积分建立单元平衡方程。

$$N_i = \frac{(1+\xi_i\xi)(1+\eta_i\eta)(1+\zeta_i\zeta)}{8} \; ; (i=0, 1, 2, \cdots, 7)$$

其中$(\xi,\eta,\zeta)\in[-1,1]$，是局部坐标。

（2）另一种是不通过形函数转换，直接在全局坐标下，用高数值斯积分建立平衡方程。

3.1.1 节点描述

如图 3.1 所示的 8 节点六面体单元，单元的节点位移有 24 个自由度。单元的节点位移$\mathbf{q}^e$和节点力矩阵$\mathbf{P}^e$为：

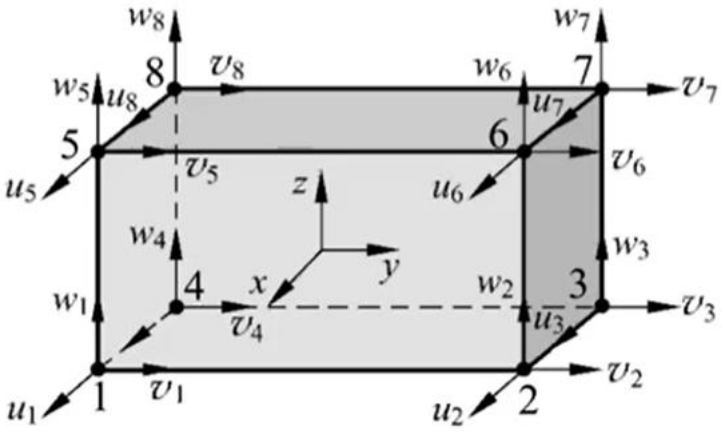


图 3.1 8 节点六面体单元

$$\mathbf{q}^e = [u_1 \;\; w_1 \;\; u_2 \;\; v_2 \;\; w_2 \cdots u_8 \;\; v_8 \;\; w_8]^T$$

$$\mathbf{P}^e = \left[P_{x1} \;\; P_{y1} \;\; P_{z1} \;\; P_{x2} \;\; P_{y2} \;\; P_{z2} \cdots P_{x8} \;\; P_{y8} \;\; P_{z8}\right]^T$$

3.1.2 单元上的场描述

该单元有 8 个节点，因此每个方向的位移场可以设定 8 个待定系数，即构造的插值函数有 8 个未知量。根据确定位移模式的基本原则，选取该单元的位移模式为：

$$u(\mathrm{x},\mathrm{y},\mathrm{z}) = a_0 + a_1x + a_2y + a_3z + a_4xy + a_5yz + a_6zx + a_7xyz$$
$$v(\mathrm{x},\mathrm{y},\mathrm{z}) = b_0 + b_1x + b_2y + b_3z + b_4xy + b_5yz + b_6zx + b_7xyz$$
$$w(\mathrm{x},\mathrm{y},\mathrm{z}) = c_0 + c_1x + c_2y + c_3z + c_4xy + c_5yz + c_6zx + c_7xyz$$

其中 u, v, w 分别表示节点 x, y, z 方向上的位移场。将 8 个点的坐标和位移分别代入插值函数，可以解出插值函数的待定系数（$a_{0-7}, b_{0-7}, c_{0-7}$）

$$\begin{cases} a_0 + a_1x + a_2y + a_3z + a_4xy + a_5yz + a_6zx + a_7xyz = u_i \\ b_0 + b_1x + b_2y + b_3z + b_4xy + b_5yz + b_6zx + b_7xyz = v_i \\ c_0 + c_1x + c_2y + c_3z + c_4xy + c_5yz + c_6zx + c_7xyz = w_i \end{cases} \quad (i=1, 2, \cdots, 8)$$

再整理可得该单元的**位移场**函数矩阵，即：

$$\boldsymbol{u} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} N_1 & 0 & 0 & N_2 & 0 & 0 & \cdots N_8 & 0 & 0 \\ 0 & N_1 & 0 & 0 & N_2 & 0 & \cdots 0 & N_8 & 0 \\ 0 & 0 & N_1 & 0 & 0 & N_2 & \cdots 0 & 0 & N_8 \end{bmatrix} \cdot \mathbf{q}^e = \boldsymbol{N} \cdot \mathbf{q}^e$$

由几何方程得单元**应变场**函数：

$$\boldsymbol{\varepsilon} = [\partial]\boldsymbol{u} = [\partial]\boldsymbol{N}\mathbf{q}^{\boldsymbol{e}} = \boldsymbol{B}\boldsymbol{q}^e$$

由物理方程得单元**应力场**函数：

$$\sigma = \boldsymbol{D}\boldsymbol{\varepsilon} = \boldsymbol{D}\boldsymbol{B}\boldsymbol{q}^e = \boldsymbol{S}\boldsymbol{q}^e$$

3.2 等参单元

坐标等参转换：

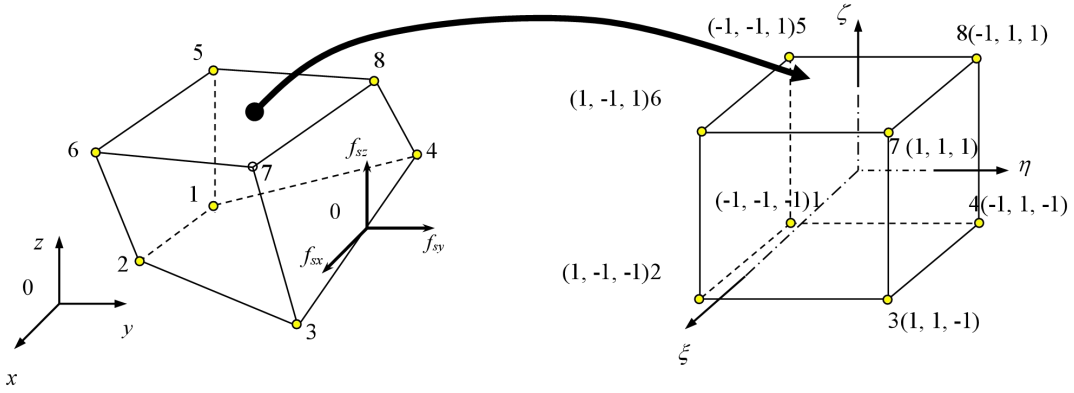


图 3.2 六面体单元等参转换

x, y, z 表达为：

$$x = \sum_{i=1}^{8} N_i(\xi,\eta,\zeta)x_i$$
$$y = \sum_{i=1}^{8} N_i(\xi,\eta,\zeta)y_i$$
$$z = \sum_{i=1}^{8} N_i(\xi,\eta,\zeta)z_i$$

形函数为：

$$N_i = \frac{1}{8}(1+\xi\xi_i)(1+\eta\eta_i)(1+\zeta\zeta_i)$$

B 矩阵为：

$$\mathbf{B}_i = \mathbf{L}\mathbf{N}_i = \begin{bmatrix} \partial N_i/\partial x & 0 & 0 \\ 0 & \partial N_i/\partial y & 0 \\ 0 & 0 & \partial N_i/\partial z \\ 0 & \partial N_i/\partial z & \partial N_i/\partial y \\ \partial N_i/\partial z & 0 & \partial N_i/\partial x \\ \partial N_i/\partial y & \partial N_i/\partial x & 0 \end{bmatrix}$$

链式法则：

$$\frac{\partial N_i}{\partial \xi} = \frac{\partial N_i}{\partial x}\frac{\partial x}{\partial \xi} + \frac{\partial N_i}{\partial y}\frac{\partial y}{\partial \xi} + \frac{\partial N_i}{\partial z}\frac{\partial z}{\partial \xi}$$
$$\frac{\partial N_i}{\partial \eta} = \frac{\partial N_i}{\partial x}\frac{\partial x}{\partial \eta} + \frac{\partial N_i}{\partial y}\frac{\partial y}{\partial \eta} + \frac{\partial N_i}{\partial z}\frac{\partial z}{\partial \eta}$$
$$\frac{\partial N_i}{\partial \zeta} = \frac{\partial N_i}{\partial x}\frac{\partial x}{\partial \zeta} + \frac{\partial N_i}{\partial y}\frac{\partial y}{\partial \zeta} + \frac{\partial N_i}{\partial z}\frac{\partial z}{\partial \zeta}$$

雅可比矩阵：

$$\mathbf{J} = \begin{bmatrix} \dfrac{\partial x}{\partial \xi} & \dfrac{\partial y}{\partial \xi} & \dfrac{\partial z}{\partial \xi} \\ \dfrac{\partial x}{\partial \eta} & \dfrac{\partial y}{\partial \eta} & \dfrac{\partial z}{\partial \eta} \\ \dfrac{\partial x}{\partial \zeta} & \dfrac{\partial y}{\partial \zeta} & \dfrac{\partial z}{\partial \zeta} \end{bmatrix}$$

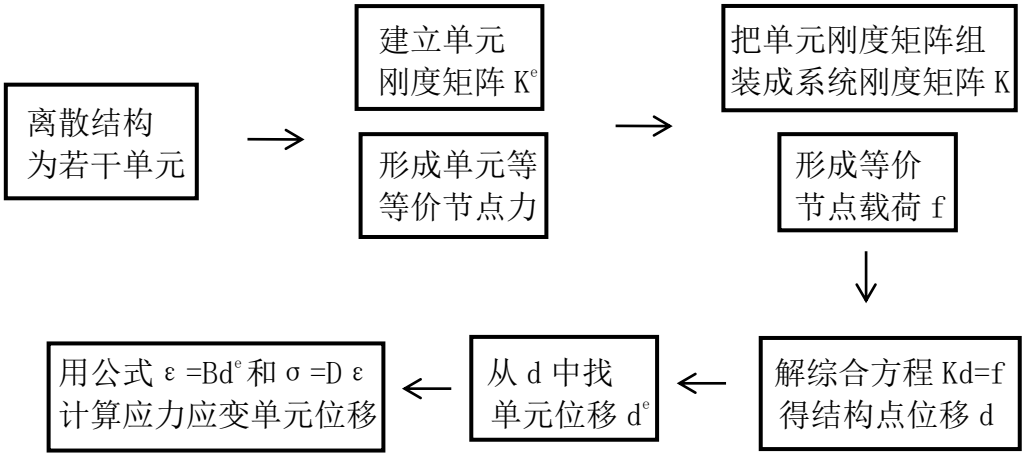
可写为以下形式：

$$\begin{Bmatrix} \dfrac{\partial N_i}{\partial \xi} \\ \dfrac{\partial N_i}{\partial \eta} \\ \dfrac{\partial N_i}{\partial \zeta} \end{Bmatrix} = \mathbf{J} \begin{Bmatrix} \dfrac{\partial N_i}{\partial x} \\ \dfrac{\partial N_i}{\partial y} \\ \dfrac{\partial N_i}{\partial z} \end{Bmatrix}$$

刚度矩阵为：

$$\mathbf{k}_e = \int_{V_e} \mathbf{B}^{\mathrm{T}} \mathbf{c} \mathbf{B} \mathrm{d}A = \int_{-1}^{+1} \int_{-1}^{+1} \int_{-1}^{+1} \mathbf{B}^{\mathrm{T}} \mathbf{c} \mathbf{B} \det[\mathbf{J}] \mathrm{d}\xi \mathrm{d}\eta \mathrm{d}\zeta$$

3.3 有限元计算流程



四、MATLAB 分析程序开发过程

4.1 程序需求分析

本次大作业，采用静力分析求解位移、应力问题，网格为八节点六面体网格，求解在不同位置受到两个方向相反的均布力的情况下，集磁器的变形和应力分布情况，先求解各节点位移再求解各节点所受应力。由于集磁器形状不规则，所以采用六面体网格划分实体时不用程序而是先在 Soliworks 中画出结构，再导入 HyperMesh 划分网格，导出 k 文件，再在 Matlab 界面读取 k 文件信息，得到节点编号、单元编号及对应的节点编号。

以密度为 4mm 的六面体网格划分模型，得到共计 1860 个网格单元，2976 个节点，8928 个自由度。对底面 288 个节点施加约束，对顶面 592 个节点施加竖直向下的集中力，对底部 304 个节点施加竖直向上的集中力。图 4.1 为以 4mm 的密度划分网格示意图，图 4.2 为地面约束节点，图 4.3 为上下两面受载荷的节点。

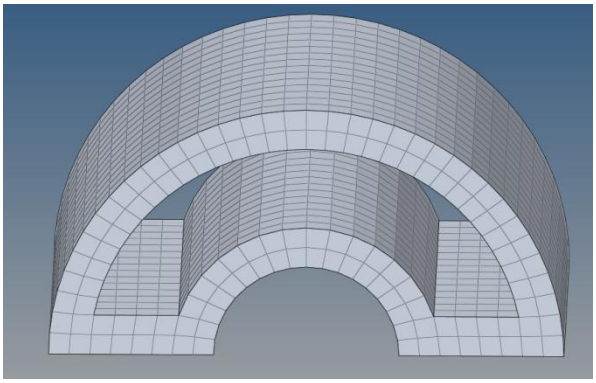


图 4.1 以 4mm 划分模型

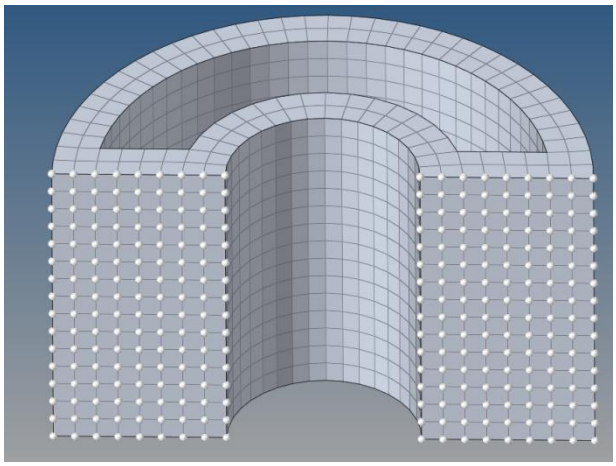


图 4.2 底面约束

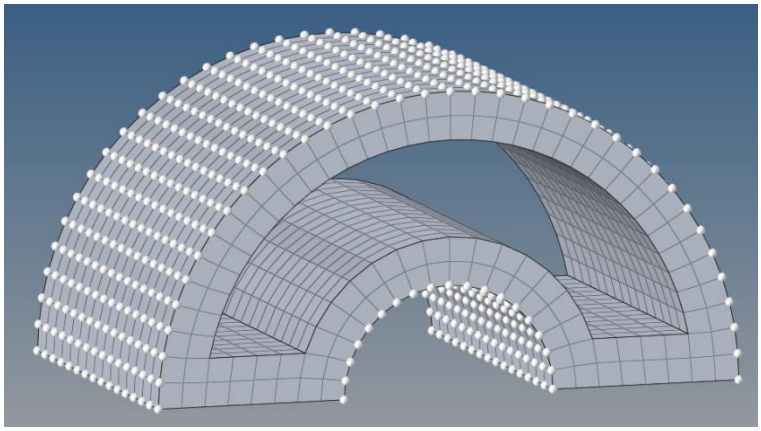


图 4.3 受载荷节点

4.2 程序架构设计

本次课程设计使用了 Solidworks2022、HyperMesh2019、Workbench2022、Matlab2020、Tecplot2022。基于 Solidworks 进行三维模型建模，具体为中空集磁器模型，以 STEP 格式导出文件，将文件导入到商业软件 Hypermesh 中进行网格划分，再导出为 k 文件。在二次开发软件 Matlab 中读入 k 文件，获取网格和节点信息，计算材料本构矩阵、B 矩阵以及刚度矩阵，最后添加约束和载荷条件并求解位移及应力，具体的求解程序在所提交的 Matlab 文件中。运行程序所生成的 plt 文件可在 Tecplot2022 中查看。而在 Solidworks 中，将结构模型以 x_t 格式导出文件，将文件导入软件 Workbench 中进行加载和边界设置得到仿真结果。整体流程图，如图 4.4 所示。

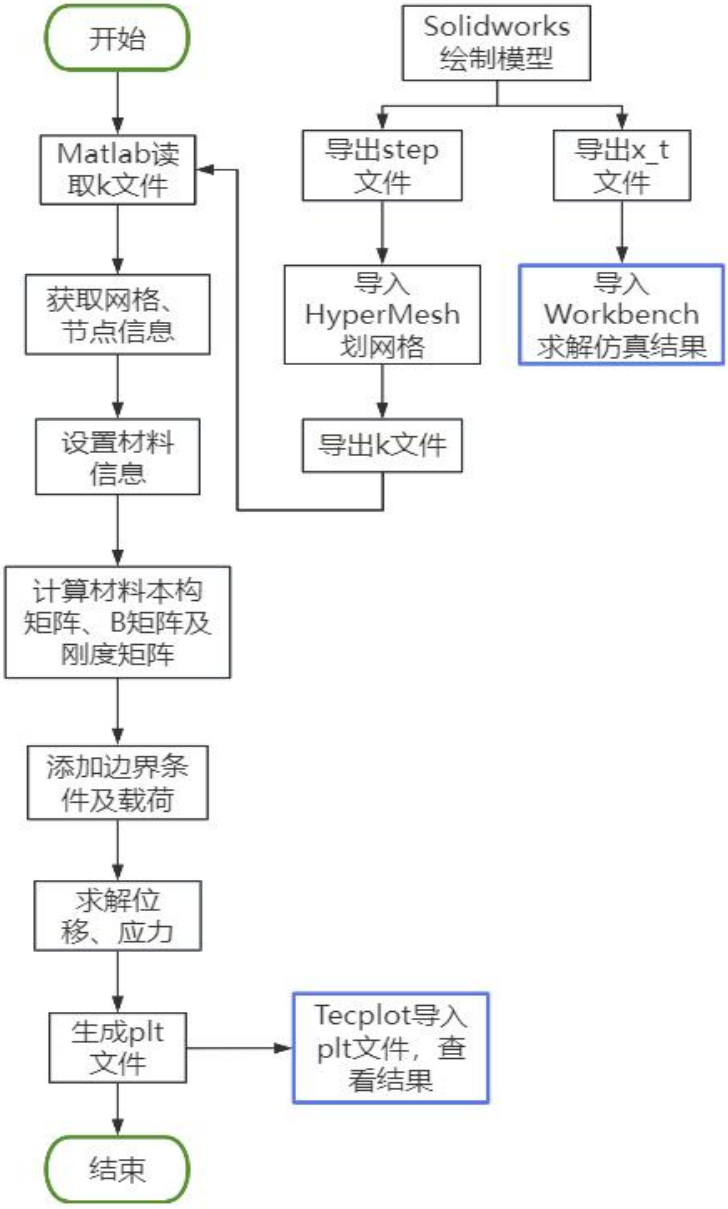


图 4.4 整体流程图

4.3 程序模块介绍

程序总共分为 13 个模块，下面对其功能进行介绍：

main 主程序:读取节点、单元信息，求解位移、应力值，包含以下函数：

1）Solve_element：函数求解网格编号及对应 8 个节点编号；

2）Solve_node：求解节点 x,y,z 坐标；

3）Solve_K 函数：求解整体刚度矩阵；

4）Jacob 函数：求形函数偏导-求解 B 矩阵；

5）Feglqd 函数：调取权重、高斯积分点坐标；

6）Element_Dof 函数：提取单元自由度序号；

7）Hexahedral3D8Node_Load 函数：在受载节点处施加载荷；

8）Hexahedral3D8Node_Feaplyc2 函数：去除整体刚度矩阵的奇异性；

9）Hexahedral3D8Node_Stress2 函数：计算网格单元中心点的应力值；

10）Plot_d 函数：绘制横截面变形示意图；

11）Createplt_d 函数：生成位移 plt 文件；

12）Createplt_Sx 函数：生成 x 轴法向应力 plt 文件；

13）Createplt_Sy 函数：生成 y 轴法向应力 plt 文件；

14）Createplt_Sz 函数：生成 z 轴法向应力 plt 文件；

15）Createplt_Sxy 函数：生成 xy 方向切应力 plt 文件；

16）Createplt_Syz 函数：生成 yz 方向切应力 plt 文件；

17）Createplt_Sxz 函数：生成 xz 方向切应力 plt 文件；

4.4 计算结果对标分析

4.4.1 横截面变形示意图

首先为了分析集磁器横截面受到载荷后的变形趋势，将变形前的横截面网格分布采用蓝色虚线绘制，变形后的横截面网格采用红色实现绘制，经过 patch 函数连接并进行对比，研究变形方向是否与载荷方向一致。由于求解的位移结果数量级很小而若单独画出显示不出区别，所以在前面乘以位移放大系数，使得位移变化明显。从图 4.5 可以看出集磁器受到 Fn 的压力后顶部向下产生压缩变形，

受到 F 的压力后底部向上产生拉伸变形，证明变形趋势符合受载情况。

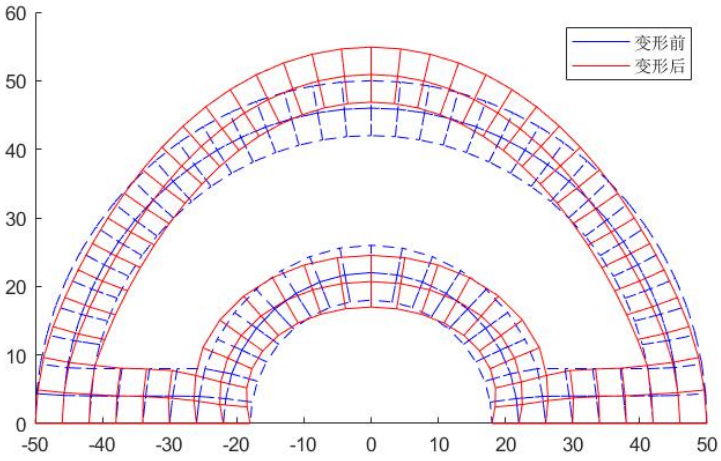


图 4.5 横截面变形示意图

4.4.2 x 方向位移云图对比

由于集磁器模型节点太多，x 方向变形情况复杂，所以选择比较变形云图等高线以及变形趋势是否一致，并且求采用 Matlab 求解结果相较于软件仿真的误差，用最大位移作为比较依据。

用 Tecplot 软件导入 result_d.plt 文件并显示 x 方向变形结果，如图 4.6 所示。我们可以观察发现顶面变形有四个集中点，在该位置 x 方向位移出现极值，这是 x 方向位移的大致分布情况。

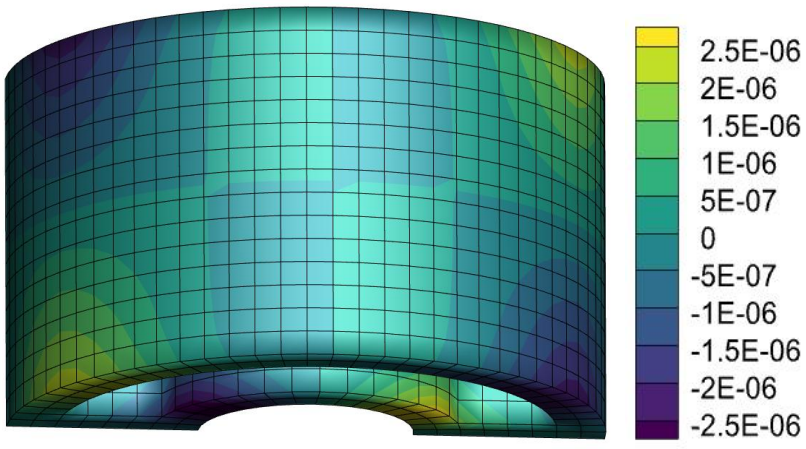


图 4.6 Tecplot 显示 x 方向变形结果

下面采用 Workbench 软件对集磁器结构经过工程数据设置、导入几何结构（.x_t 格式）、固定支撑、设置载荷条件这四个步骤后，得到有限元仿真结果，如图 4.7 所示。首先，我们对比集磁器顶部以及底部的 x 方向位移分布情况，可

明显得到顶面变形有四个集中点，在该位置 x 方向位移出现极值。经过对比可以得出用 Matlab 求解 x 方向位移结果趋势正确，下面需要对两种方式的位移数值进行精度分析，得出程序求解的误差。

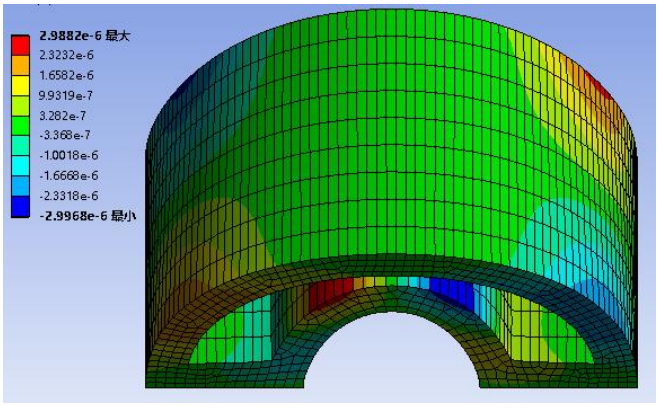


图 4.7 Workbench 仿真 x 方向变形结果

图 4.8 为 Tecplot 显示程序运行产生的位移结果，其 x 方向位移最大值为 2.94073e-6 米，而由上图可知由 Workbench 仿真得出的 x 方向位移结果最大值为 2.9882e-6 米，**误差为 2.5%**，表明该程序计算出的 x 方向位移结果精度较高。

```
Contour variable range
  Min: -2.94073e-06     Max: 2.94073e-06
```

图 4.8 x 方向位移程序运行结果

4.4.3 y 方向位移云图对比

用 Tecplot 软件导入 result_d.plt 文件并显示 y 方向变形结果，如图 4.9 所示。我们可以观察发现顶面变形从中间向两侧扩散并且数值逐渐变大，这是 y 方向位移的大致分布情况。

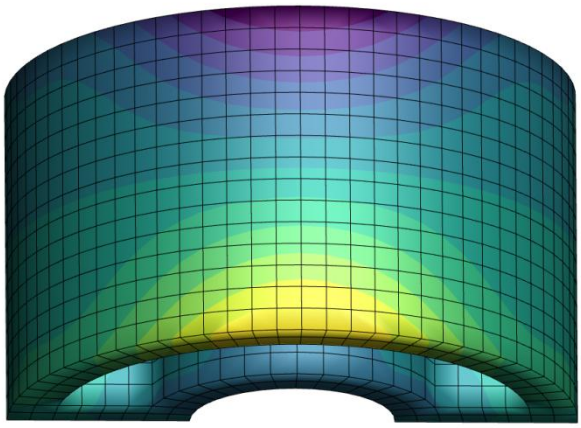


图 4.9 Tecplot 显示 y 方向变形结果

下面采用 Workbench 软件对集磁器结构经过工程数据设置、导入几何结构（.x_t 格式）、固定支撑、设置载荷条件这四个步骤后，得到有限元仿真结果，如图 4.10 所示。首先，我们对比集磁器顶部以及底部的 y 方向位移分布情况，可明显看出顶面变形从中间向两侧扩散并且数值逐渐变大。经过对比可以得出用 Matlab 求解位移结果趋势正确，下面需要对两种方式的位移数值进行精度分析，得出程序求解的误差。

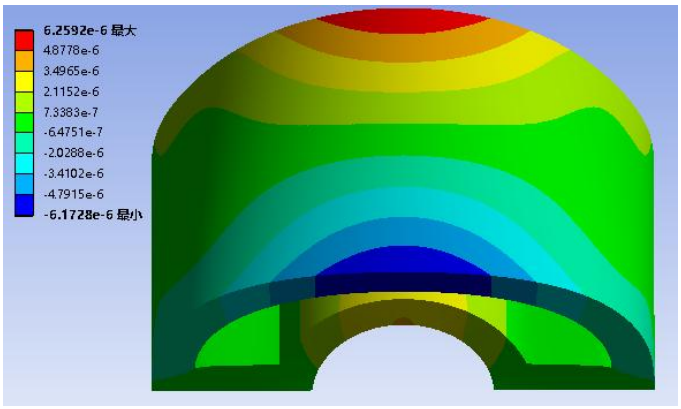


图 4.10 Workbench 仿真 y 方向变形结果

图 4.11 为 Tecplot 显示程序运行产生的 y 方向位移结果，其位移最大值为 6.1243-5 米，而由上图可知由 Workbench 仿真得出的 y 方向位移结果最大值为 6.2592e-5 米，**误差为 2.8%**，表明该程序计算出的 y 方向位移结果精度较高。

Contour variable range

Min: -6.1243e-06 Max: 6.1243e-06

图 4.11 y 方向位移程序运行结果

4.4.4 z 方向位移云图对比

用 Tecplot 软件导入 result_d.plt 文件并显示 z 方向变形结果，如图 4.12 所示。我们可以观察发现顶面变形均小于底面变形且小了一个数量级，在底部两端的变形达到最大值，这是 z 方向位移的大致分布情况。

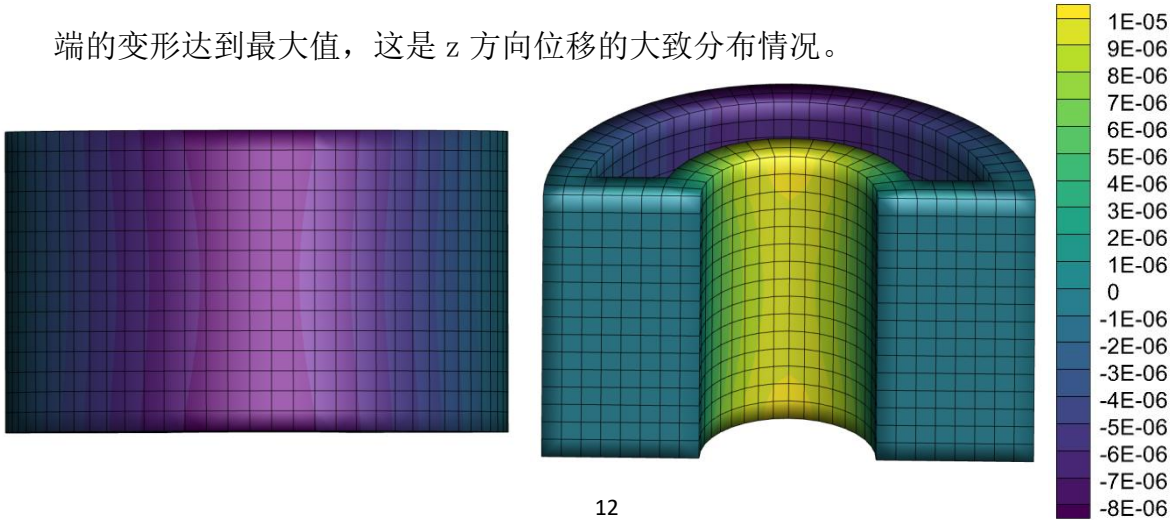


图 4.12 Tecplot 显示 z 方向变形结果

下面采用 Workbench 软件对集磁器结构经过工程数据设置、导入几何结构（.x_t 格式）、固定支撑、设置载荷条件这四个步骤后，得到有限元仿真结果，如图 4.13 所示。首先，我们对比集磁器顶部以及底部的位移分布情况，可明显得到顶面变形均小于底面变形，且在底部两端的变形达到最大值。经过对比可以得出用 Matlab 求解位移结果趋势正确，下面需要对两种方式的位移数值进行精度分析，得出程序求解的误差。

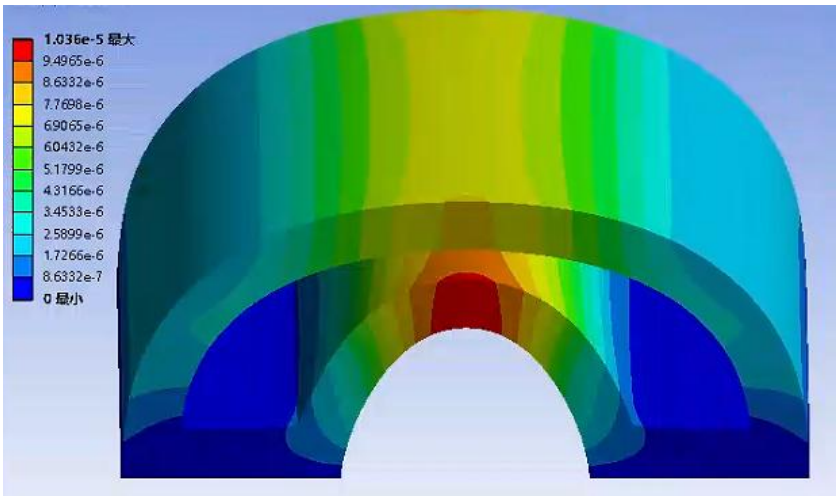


图 4.13 Workbench 仿真 z 方向变形结果

图 4.14 为 Tecplot 显示程序运行产生的 z 方向位移结果，其位移最大值为 1.03386e-5 米，而由上图可知由 Workbench 仿真得出的 z 方向位移结果最大值为 1.036e-5 米，**误差为 2.1%**，表明该程序计算出的 z 方向位移结果精度较高。

```
Contour variable range
  Min: -8.5864e-06    Max: 1.03386e-05
```

图 4.14 z 方向位移程序运行结果

4.4.5 x 轴法向应力云图对比

由于集磁器模型节点太多，应力分布情况复杂，所以选择比较 x 轴应力云图等高线以及趋势是否一致，并且求采用 Matlab 求解结果相较于软件仿真的误差，用最大应力作为比较依据。用 Tecplot 软件导入 result_Sx.plt 文件显示变形结果，如图 4.15 所示。我们可以观察发现在底部上端的应力达到最大值，在顶部上端应力达到最大值，从上往下呈现水平等高线。

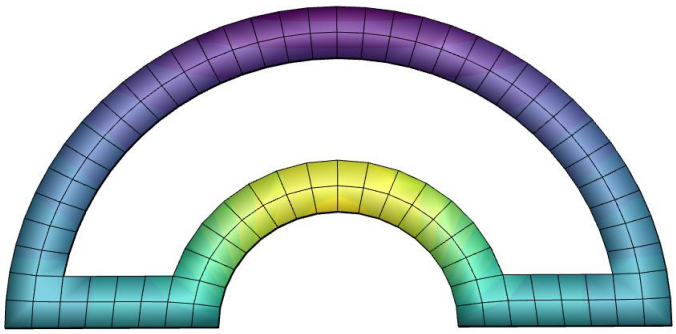


图 4.15 Tecplot 显示 x 轴法向应力结果

下面采用 Workbench 软件对集磁器结构进行有限元仿真，得结果如图 4.16 所示。首先，我们对比发现 Matlab 求解 x 轴法向应力结果在底部上端的应力达到最大值，在顶部上端应力达到最大值，从上往下呈现水平等高线。所以可得应力分布大致正确，下面需要对两种方式的应力数值进行精度分析，得出程序求解的误差。

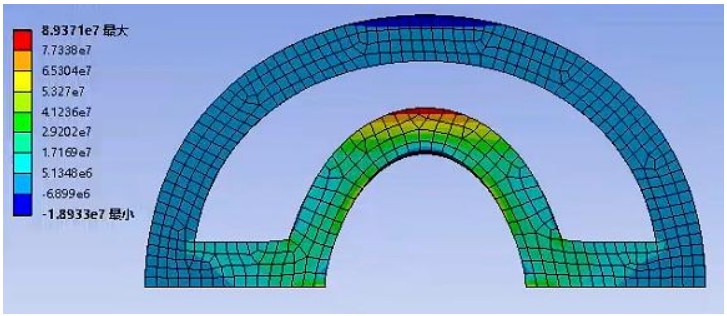


图 4.16 Workbench 仿真 x 轴法向应力结果

图 4.17 为 Tecplot 显示程序运行产生的 x 轴法向应力结果，其应力最大值 8.726e+7 Pa，而由上图可知由 Workbench 仿真得出的 x 轴法向应力结果最大值为 8.9371e+7 Pa，**误差为 2.4%**，表明该程序计算出的应力结果精度较高。

```
Contour variable range
  Min: -1.886e+07    Max: 8.726e+07
```

图 4.17 x 轴法向应力程序运行结果

4.4.6 xz 方向剪切应力云图对比

用 Tecplot 软件导入 result_Szx.plt 文件显示变形结果，如图 4.18 所示。我们可以观察发现在底部上端两侧的应力出现极值，等高线向外扩散且应力值逐渐递减。

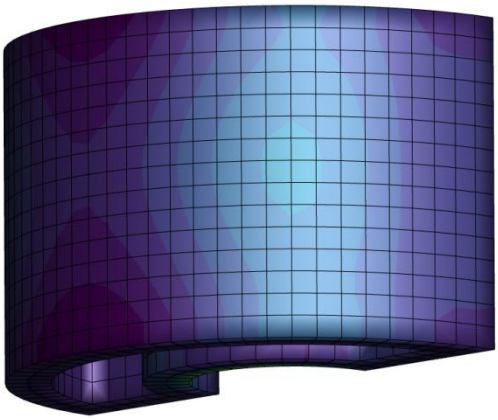


图 4.18 Tecplot 显示 xz 方向剪切应力结果

下面采用 Workbench 软件对集磁器结构进行有限元仿真，得结果如图 4.19 所示。首先，我们对比发现 Matlab 求解 xz 方向剪切应力结果在底部上端两侧的应力出现极值，等高线向外扩散且应力值逐渐递减。所以可得 xz 方向剪切应力结果分布大致正确，下面需要对两种方式的应力数值进行精度分析，得出程序求解的误差。

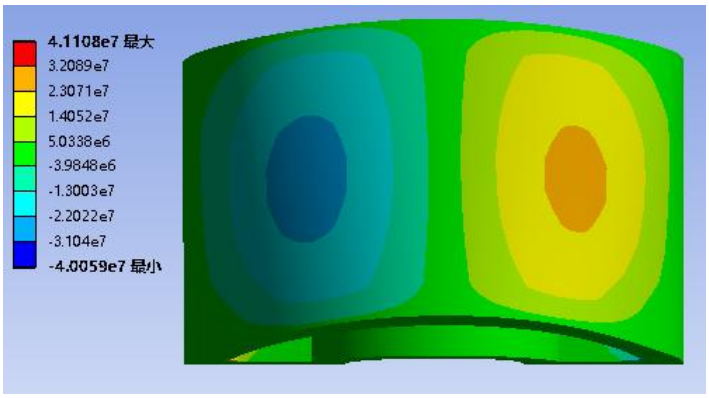


图 4.19 Workbench 仿真 xz 方向剪切应力结果

图 4.20 为 Tecplot 显示程序运行产生的 xz 方向剪切应力结果，其应力最大值 4.2658e+7 Pa，而由上图可知由 Workbench 仿真得出的 xz 方向剪切应力结果最大值为 4.1108e+7 Pa，**误差为 3.2%**，表明该程序计算出的应力结果精度较高。

```
Contour variable range
  Min: -4.1379e+07    Max: 4.2658e+07
```

图 4.20 xz 方向剪切应力程序运行结果

除此之外，在应力的求解步骤中，还采用程序计算了求解了 y、z 轴法向应力以及 xy、yz 方向的剪切应力分布，但在此只选择求解结果中一个法向应力及一个剪切应力与其对应的软件仿真结果做比较判断求解是否正确。

综上所述，通过以上对节点 x 方向、y 方向和 z 方向的位移结果以及选取某个法向应力和剪切应力结果进行对标分析。可以看出根据集磁器的受力情况，在 Matlab 中编写的程序能够较为精确地计算出集磁器的节点位移以及各方向应力，并且所有计算结果的误差基本在 2.5%左右，说明该程序具有较高的可靠性，能够为后续研究中间挖空部分的不同图形形成的集磁器结构提供强度校验。

附件

1.main 主程序

```
clear all;
% 定义参数
E=1.28e11;
poisson=0.3;
F=7.5e3;
load_k=8.0e4;
kFileStr='mesh.k';   % 读取 k 文件
overview=importdata(kFileStr,',');
element_solid=Solve_element(overview);   % 求解单元信息
node=Solve_node(overview);   % 求解节点坐标
[K]=Solve_K(poisson,element_solid,node,E);   % 求解刚度矩阵
uni_load=Hexahedral3D8Node_Load(node,load_k,F);   % 得到载荷矩阵
[KK,ff]=Hexahedral3D8Node_Feaplyc2(node,K,uni_load);   % 保证刚度矩阵非奇异
[LL, UU]=lu(KK);
utemp=LL\ff;
d=UU\utemp;   % 求解节点位移
% 求解应力
[sx,sy,sz,sxy,syz,szx]=Solve_stress(element_solid,node,E,poisson,d);
plot_d(node,element_solid,d)   % 绘制变形示意图
Createplt_d(node,element_solid,d)    % 生成位移 plt 文件
Createplt_Sx(node,element_solid,sx)    % 生成 x 方向应力 plt 文件
Createplt_Sy(node,element_solid,sy)    % 生成 y 方向应力 plt 文件
Createplt_Sz(node,element_solid,sz)    % 生成 z 方向应力 plt 文件
Createplt_Sxy(node,element_solid,sxy)    % 生成 xy 平面切应力 plt 文件
Createplt_Syz(node,element_solid,syz)    % 生成 yz 平面切应力 plt 文件
Createplt_Szx(node,element_solid,szx)    % 生成 zx 平面切应力 plt 文件
```

2.Solve_element 函数（求解网格编号及对应 8 个节点编号）

```
function  element_solid=Solve_element(overview)
for  i=2986:2:6704
Element_Solid(i-2985,:)=strsplit(overview{i,1});  % 将元胞中每个元素的内容分来
end
Element_Solid(:,1)=[];
for  i=2986:2:6704
    Element_solid(i-2985,1)=str2num(Element_Solid{i-2985,1});
    Element_solid(i-2985,2)=str2num(Element_Solid{i-2985,2});
end
for i=2987:2:6705
Element_Node(i-2986,:)=strsplit(overview{i,1});
```

```
end
Element_Node(:,1)=[];
for i=2987:2:6705
    Element_solid(i-2986,2)=str2num(Element_Node{i-2986,1});
    Element_solid(i-2986,3)=str2num(Element_Node{i-2986,2});
    Element_solid(i-2986,4)=str2num(Element_Node{i-2986,3});
    Element_solid(i-2986,5)=str2num(Element_Node{i-2986,4});
    Element_solid(i-2986,6)=str2num(Element_Node{i-2986,5});
    Element_solid(i-2986,7)=str2num(Element_Node{i-2986,6});
    Element_solid(i-2986,8)=str2num(Element_Node{i-2986,7});
    Element_solid(i-2986,9)=str2num(Element_Node{i-2986,8});
end
i=1;
for b=1:1860
    element_solid(b,1)=Element_solid(i,1);
    element_solid(b,2)=Element_solid(i,2);
    element_solid(b,3)=Element_solid(i,3);
    element_solid(b,4)=Element_solid(i,4);
    element_solid(b,5)=Element_solid(i,5);
    element_solid(b,6)=Element_solid(i,6);
    element_solid(b,7)=Element_solid(i,7);
    element_solid(b,8)=Element_solid(i,8);
    element_solid(b,9)=Element_solid(i,9);
    i=i+2;
end
for b=1:1800
     element_solid(b,1)=element_solid(b,1)-124;
end
```

3. Solve_node 函数（求解节点 x, y, z 坐标）

```
function node=Solve_node(overview)
for i=7:2982
Node(i-6,:)=strsplit(overview{i,1});  % 将元胞中每个元素的内容分来
end
Node(:,1)=[];
for i=7:2982
    node(i-6,1)=str2num(Node{i-6,1});
    node(i-6,2)=str2num(Node{i-6,2});
    node(i-6,3)=str2num(Node{i-6,3});
    node(i-6,4)=str2num(Node{i-6,4});
end
for i=1:2976  % 将坐标小于 0.1 的值置 0
     if abs(node(i,4))<0.1
       node(i,4)=0;
```

```
      end
end
for i=1:2976
      if abs(node(i,2))<0.1
            node(i,2)=0;
      end
end
for i=1:2976
     if abs(node(i,3))<0.1
       node(i,3)=0;
      end
end
```

4.Solve_K 函数（求解整体刚度矩阵）

```
function [K]=Solve_K(poisson,element_solid,node,E)
eles_num=size(element_solid,1);    % 单元数
nodes_num=size(node,1);    % 节点数
total_dof=nodes_num*3;    % 总自由度数
nel=8;node_dof=3;
temp_element_solid=element_solid;temp_element_solid(:,1)=[];
temp_node=node;temp_node(:,1)=[];
element_dof=nel*node_dof;    % 一个单元自由度数
matmtx=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];    % 材料矩阵
K=sparse(total_dof,total_dof);        % 初始化整体刚度矩阵
nx=2;
ny=2;
nz=2;
% 调取权重、高斯积分点坐标
[point,weight]=feglqd(nx,ny,nz);
for i=1:eles_num
    for j=1:nel
        nod(j)=temp_element_solid(i,j);    % 节点编号
       x(j)=temp_node(nod(j),1);    % 节点 x 坐标
       y(j)=temp_node(nod(j),2);    % 节点 y 坐标
       z(j)=temp_node(nod(j),3);    % 节点 z 坐标
    end
    Ke=sparse(element_dof,element_dof);      % 初始化单元刚度矩阵
    for ix=1:nx
```

```
        sx=point(ix,1);          % x 方向高斯积分点
        sw=weight(ix,1);         % x 方向权重
        for iy=1:ny
            mx=point(iy,2);        % y 方向高斯积分点
            mw=weight(iy,2);       % y 方向权重
            for iz=1:nz
                tx=point(iz,3);       % z 方向高斯积分点
                tw=weight(iz,3);      % z 方向高斯积分点
                [J,B]=Jacob(sx,mx,tx,nel,x,y,z);     % 求解雅可比矩阵及 B 矩阵
                det_J=det(J);
                Ke=Ke+B'*matmtx*B*sw*mw*tw*det_J;
            end
          end
      end
      index=Element_Dof(nod,nel,node_dof);     % 提取单元自由度
     K(index,index)=K(index,index) + Ke;      % 组装单元刚度矩阵
end
```

5. Jacob 函数（求形函数偏导-求解 B 矩阵）

```
function  [J,B]=Jacob(sx,mx,tx,nel,x,y,z)
 dNdr(1)=-1/8*(1-mx)*(1-tx);  % 形函数偏导
 dNdr(2)=1/8*(1-mx)*(1-tx);
 dNdr(3)=1/8*(1+mx)*(1-tx);
 dNdr(4)=-1/8*(1+mx)*(1-tx);
 dNdr(5)=-1/8*(1-mx)*(1+tx);
 dNdr(6)=1/8*(1-mx)*(1+tx);
 dNdr(7)=1/8*(1+mx)*(1+tx);
 dNdr(8)=-1/8*(1+mx)*(1+tx);

 dNds(1)=-1/8*(1-sx)*(1-tx);
 dNds(2)=-1/8*(1+sx)*(1-tx);
 dNds(3)=1/8*(1+sx)*(1-tx);
 dNds(4)=1/8*(1-sx)*(1-tx);
 dNds(5)=-1/8*(1-sx)*(1+tx);
 dNds(6)=-1/8*(1+sx)*(1+tx);
 dNds(7)=1/8*(1+sx)*(1+tx);
 dNds(8)=1/8*(1-sx)*(1+tx);

 dNdt(1)=-1/8*(1-sx)*(1-mx);
 dNdt(2)=-1/8*(1+sx)*(1-mx);
 dNdt(3)=-1/8*(1+sx)*(1+mx);
 dNdt(4)=-1/8*(1-sx)*(1+mx);
 dNdt(5)=1/8*(1-sx)*(1-mx);
 dNdt(6)=1/8*(1+sx)*(1-mx);
```

```
dNdt(7)=1/8*(1+sx)*(1+mx);
dNdt(8)=1/8*(1-sx)*(1+mx);
J=zeros(3,3);
for  i=1:nel   % 求解雅可比矩阵
     J(1,1)=J(1,1)+dNdr(i)*x(i);
     J(1,2)=J(1,2)+dNdr(i)*y(i);
     J(1,3)=J(1,3)+dNdr(i)*z(i);
     J(2,1)=J(2,1)+dNds(i)*x(i);
     J(2,2)=J(2,2)+dNds(i)*y(i);
     J(2,3)=J(2,3)+dNds(i)*z(i);
     J(3,1)=J(3,1)+dNdt(i)*x(i);
     J(3,2)=J(3,2)+dNdt(i)*y(i);
     J(3,3)=J(3,3)+dNdt(i)*z(i);
end
inv_J=inv(J);  % 求逆
for  j=1:nel
     dNdx(j)=inv_J(1,1)*dNdr(j)+inv_J(1,2)*dNds(j)+inv_J(1,3)*dNdt(j);
     dNdy(j)=inv_J(2,1)*dNdr(j)+inv_J(2,2)*dNds(j)+inv_J(2,3)*dNdt(j);
     dNdz(j)=inv_J(3,1)*dNdr(j)+inv_J(3,2)*dNds(j)+inv_J(3,3)*dNdt(j);
end
for j=1:nel
     i1=(j-1)*3+1;
     i2=i1+1;
     i3=i1+2;
     B(1,i1)=dNdx(j);
     B(2,i2)=dNdy(j);
     B(3,i3)=dNdz(j);
     B(4,i1)=dNdy(j);
     B(4,i2)=dNdx(j);
     B(5,i2)=dNdz(j);
     B(5,i3)=dNdy(j);
     B(6,i1)=dNdz(j);
     B(6,i3)=dNdx(j);
  end
```

6. Feglqd 函数（调取权重、高斯积分点坐标）

```
function [point,weight]=Feglqd(nx,ny,nz)
    % 判断最后一个积分点，取最长的一个积分点
    if nx > ny
       ngl=nx;
    else
       ngl=ny;
    end
% 初始化权重、积分点
```

```
    point=zeros(2,3);
    weight=zeros(2,3);
% 找到对应的积分点和权重
[pointx,weightx]=feglqd1(nx);       % x 方向积分级数
[pointy,weighty]=feglqd1(ny);       % y 方向积分级数
[pointz,weightz]=feglqd1(nz);       % z 方向积分级数
% 二维正交
for intx=1:nx
    point(intx,1)=pointx(intx);
    weight(intx,1)=weightx(intx);
end
for inty=1:ny
    point(inty,2)=pointy(inty);
    weight(inty,2)=weighty(inty);
end
for intz=1:nz
    point(intz,3)=pointz(intz);
    weight(intz,3)=weightz(intz);
end
```

7.Element_Dof 函数（提取单元自由度序号）

```
function [index]=Element_Dof(nod,nel,node_dof)
   k=0;
   for i=1:nel
       temp=(nod(i)-1)*node_dof;
       for j=1:node_dof
           k=k+1;
           index(k)=temp+j;
       end
   end
```

8.Hexahedral3D8Node_Load 函数（施加节点载荷）

```
function uni_load=Hexahedral3D8Node_Load(node,load_k,F)
X=[-18,-17.727,-16.914,-15.588,-13.789,-11.570,-9.000,-6.156,-3.126,0,3.126,
6.156,9,11.57,13.789,15.588,16.914,17.727,18];  % 节点横坐标
Y=[0,3.126,6.156,9.000,11.570,13.789,15.588,16.914,17.727,18.000,17.727,16.
914,15.588,13.789,11.570,9.000,6.156,3.126,0];  % 节点纵坐标
Z=[60,56,52,48,44,40,36,32,28,24,20,16,12,8,4,0];  % 节点竖坐标
uni_load=zeros(size(node,1)*3,1);
for i=1:19
     for j=1:19
          for k=1:16
           temporary_x=X(i);
           temporary_y=Y(j);
```

```
            temporary_z=Z(k);
            row1_x=find(abs(node(:,2)-temporary_x)<0.001);
            row1_y=find(abs(node(:,3)-temporary_y)<0.001);
            row1_z=find(abs(node(:,4)-temporary_z)<0.001);
            num1=intersect(row1_x,row1_y);  % 通过节点坐标找到节点编号
            num=intersect(num1,row1_z);
            uni_load(3*num)=load_k;  % 施加洛伦兹力
          end
      end
end
X=[-50,-49.810,-49.240,-48.296,-46.985,-45.315,-43.301,-40.958,-38.302,-35.
355,-32.139,-28.679,-25.000,-21.131,-17.101,-12.941,-8.682,-4.358,0,4.358,8.
682,12.941,17.101,21.131,25.000,28.679,32.139,35.355,38.302,40.958,43.301,4
5.315,46.985,48.296,49.240,49.810,50.000];
Y=[0,4.358,8.682,12.941,17.101,21.131,25.000,28.679,32.139,35.355,38.302,40.
958,43.301,45.315,46.985,48.296,49.240,49.810,50,49.810,49.240,48.296,46.98
5,45.315,43.301,40.958,38.302,35.355,32.139,28.679,25.000,21.131,17.101,12.
941,8.682,4.358,0.000];
Z=[60,56,52,48,44,40,36,32,28,24,20,16,12,8,4,0];
for i=1:37
      for j=1:37
          for k=1:16
             temporary_x=X(i);
             temporary_y=Y(j);
             temporary_z=Z(k);
             row2_x=find(abs(node(:,2)-temporary_x)<0.001);
             row2_y=find(abs(node(:,3)-temporary_y)<0.001);
             row2_z=find(abs(node(:,4)-temporary_z)<0.001);
             num2=intersect(row2_x,row2_y);
             num=intersect(num2,row2_z);
             uni_load(3*num)=-F;  % 施加支持力
          end
      end
end
```

9.Hexahedral3D8Node_Feaplyc2 函数（去除刚度矩阵奇异性）

```
function [KK,ff]=Hexahedral3D8Node_Feaplyc2(node,K,uni_load)
num=find(node(:,3)-0<1e-9);  % 将固定面上的节点位移置 0
KK=K;
ff=uni_load;
for i=1:size(num,1)
    r=num(i);
    KK(3*r-2,:)=0;
    KK(:,3*r-2)=0;
```

```
    KK(3*r-2,3*r-2)=1;
    KK(3*r-1,:)=0;
    KK(:,3*r-1)=0;
    KK(3*r-1,3*r-1)=1;
    KK(3*r,:)=0;
    KK(:,3*r)=0;
    KK(3*r,3*r)=1;
    ff(3*r-2)=0;
    ff(3*r-1)=0;
    ff(3*r)=0;
end
```

10.Hexahedral3D8Node_Stress2 函数（计算单元中心点的应力）

```
function
stress=Hexahedral3D8Node_Stress2(E,poisson,b,d,element_solid,x1,y1,z1,x2,y2,
z2,x3,y3,z3,x4,y4,z4,x5,y5,z5,x6,y6,z6,x7,y7,z7,x8,y8,z8)
% 输入弹性模量 E，泊松比 poisson
% 输入 8 个节点的坐标
% 输入单元的位移列阵 d
% 输出单元中心的应力 stress(6X1)，应力分量为 Sx,Sy,Sz,Sxy,Syz,Szx
% 定义形状函数 N
node=element_solid(b,2:9);
d_local=[d(3*node(1)-2);d(3*node(1)-1);d(3*node(1));d(3*node(2)-2);d(3*node
(2)-1);d(3*node(2));d(3*node(3)-2);d(3*node(3)-1);d(3*node(3));d(3*node(4)-
2);d(3*node(4)-1);d(3*node(4));d(3*node(5)-2);d(3*node(5)-1);d(3*node(5));d
(3*node(6)-2);d(3*node(6)-1);d(3*node(6));d(3*node(7)-2);d(3*node(7)-1);d(3
*node(7));d(3*node(8)-2);d(3*node(8)-1);d(3*node(8))];
Loc=[x1 y1 z1;x2 y2 z2;x3 y3 z3;x4 y4 z4;x5 y5 z5;x6 y6 z6;x7 y7 z7;x8 y8 z8;];
nx=2;
ny=2;
nz=2;
point=Feglqd(nx,ny,nz);
for ix=1:nx
     sx=point(ix,1);
     for iy=1:ny
          mx=point(iy,2);
          for iz=1:nz
               tx=point(iz,3);
% N1-N8 对 mx,tx,sx 的导数矩阵
dNsnt=[-(1-mx)*(1+tx)/8,-(1-sx)*(1+tx)/8,(1-sx)*(1-mx)/8;
       (1-mx)*(1+tx)/8,-(1+sx)*(1+tx)/8,(1+sx)*(1-mx)/8;
       (1-mx)*(1-tx)/8,-(1+sx)*(1-tx)/8,-(1+sx)*(1-mx)/8;
       -(1-mx)*(1-tx)/8,-(1-sx)*(1-tx)/8,-(1-sx)*(1-mx)/8;
       -(1+mx)*(1-tx)/8,(1-sx)*(1+tx)/8,(1-sx)*(1+mx)/8;
```

```
        (1+mx)*(1+tx)/8,(1+sx)*(1+tx)/8,(1+sx)*(1+mx)/8;
        (1+mx)*(1-tx)/8,(1+sx)*(1-tx)/8,-(1+sx)*(1+mx)/8;
        -(1+mx)*(1-tx)/8,(1-sx)*(1-tx)/8,-(1-sx)*(1+mx)/8;];
dNsnt=dNsnt';
J=dNsnt*Loc;
dNxyz=pinv(J)*dNsnt;
B=zeros(6,24);
for ii=1:8  % 计算 B 矩阵
    Bii=[dNxyz(1,ii) 0 0;0 dNxyz(2,ii) 0;
         0 0 dNxyz(3,ii);dNxyz(2,ii) dNxyz(1,ii) 0;
         0 dNxyz(3,ii) dNxyz(2,ii);dNxyz(3,ii) 0 dNxyz(1,ii);];
     B(:,3*(ii-1)+1:3*ii)=Bii;
end
          end
      end
end
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];
w=D*B*d_local;
% 在单元的中心处计算应力
wcent=subs(w,{sx,tx,mx},{0,0,0});
stress=double(wcent);
```

11. Plot_d 函数（绘制横截面变形示意图）

```
function []=Plot_d(node,element_solid,d)
eles_num=size(element_solid,1);
nodes_num=size(node,1);
A=find(abs(node(:,4)-60)<1e-6);   % 定义横截面点
b=0;
for i=1:eles_num
    if
(ismember(element_solid(i,2),A)==1||ismember(element_solid(i,3),A)==1||isme
mber(element_solid(i,4),A)==1||ismember(element_solid(i,5),A)==1)
        b=b+1;
        I(b)=i;   % 找到横截面点对应的单元编号
    end
end
X=d(1:3:8928);Y=d(2:3:8928);   % 得到横截面上点的 x,y 方向位移
for i=1:nodes_num
```

```
    X(i)=node(i,2)+X(i)*8e5;    % 由于变形太小，将位移量乘以放大系数达到同一量级
end
for i=1:nodes_num
    Y(i)=node(i,3)+Y(i)*8e5;
end
figure('name','Fig','Position',[400,200,600,350])
for i=1:124
patch(node(element_solid(I(i),2:5),2),node(element_solid(I(i),2:5),3),node(
element_solid(I(i),2:5),4),'w','FaceColor','none','LineStyle','--','EdgeCol
or','b')   % 未变形网格用蓝色虚线表示
hold  on
patch(X(element_solid(I(i),2:5)),Y(element_solid(I(i),2:5)),node(element_so
lid(I(i),2:5),4),'w','FaceColor','none','EdgeColor','r')   % 变形后网格用红色
实线表示
end
```

12.Createplt_d 函数（生成位移 plt 文件）

```
function []=Createplt_d(node,element_solid,d)
eles_num=size(element_solid,1);
nodes_num=size(node,1);
fid=fopen('result_d.plt','w');   % 以只写模式打开 result_d.plt 文件
fprintf(fid,'TITLE="test case governed by poisson equation"\n');
fprintf(fid,'VARIABLES="x""y""z""U""V""W"\n');
fprintf(fid,'ZONE N=%8d,E=%8d,ET=BRICK,F=FEPOINT\n',nodes_num,eles_num);
for i = 1:nodes_num
fprintf(fid,'%16.6e%16.6e%16.6e%16.6e%16.6e%16.6e\n',node(i,2),node(i,3),no
de(i,4),d(3*i-2),d(3*i-1),d(3*i));   % 输出节点坐标及对应方向位移
end
for i=1:eles_num
fprintf(fid,'%8d%8d%8d%8d%8d%8d%8d%8d\n',element_solid(i,2),element_solid(i,
3),element_solid(i,4),element_solid(i,5),element_solid(i,6),element_solid(i,
7),element_solid(i,8),element_solid(i,9));   % 输出单元对应 8 个节点编号
end
fclose(fid);
```

13.Createplt_Sx 函数（生成 x 轴法向应力 plt 文件）

```
function []=Createplt_Sx(node,element_solid,sx)
eles_num=size(element_solid,1);
nodes_num=size(node,1);
for i=1:1860
    Sx(element_solid(i,2))=sx(i);
    Sx(element_solid(i,3))=sx(i);
    Sx(element_solid(i,4))=sx(i);
```

```
    Sx(element_solid(i,5))=sx(i);
    Sx(element_solid(i,6))=sx(i);
    Sx(element_solid(i,7))=sx(i);
    Sx(element_solid(i,8))=sx(i);
    Sx(element_solid(i,9))=sx(i);
end
fid=fopen('result_Sx.plt','w');
fprintf(fid,'TITLE="test case governed by poisson equation"\n');
fprintf(fid,'VARIABLES="x""y""z""sigax"\n');
fprintf(fid,'ZONE N=%8d,E=%8d,ET=BRICK,F=FEPOINT\n',nodes_num,eles_num);
for i=1:nodes_num
fprintf(fid,'%16.6e%16.6e%16.6e%16.6e\n',node(i,2),node(i,3),node(i,4),Sx(1,
i));
end
for i=1:eles_num
fprintf(fid,'%8d%8d%8d%8d%8d%8d%8d%8d\n',element_solid(i,2),element_solid(i,
3),element_solid(i,4),element_solid(i,5),element_solid(i,6),element_solid(i,
7),element_solid(i,8),element_solid(i,9));
end
fclose(fid);
```

主函数中生成 y、z 轴法向应力 plt 文件及 xy、yz、zx 方向切应力 plt 文件的函数与上面 Createplt_Sx 函数类似，所以不在此赘述。