有限元理论与方法大作业

C3D8 变截面悬臂梁有限元分析程序开发

姓　　名　郭瑞春

学 科 专 业　机械工程

学　　号　B240200175

研 究 方 向　晶圆减薄机主动减振

学　　院　机械与运载工程学院

上 课 教 师　王　琥

目录

1. C3D8 变截面悬臂梁受力分析的基本理论 1

1.1. 变截面悬臂梁受力分析 1

1.2. 变截面悬臂梁有限元分析基本原理 2

1.3. 八结点六面体单元刚度矩阵 2

2. C3D8 变截面悬臂梁有限元受力分析过程 6

2.1. Abaqus 有限元分析软件简介 6

2.2. Abaqus 前处理 6

2.3. Abaqus 求解计算 7

2.4. Abaqus 后处理 7

3. C3D8 变截面悬臂梁有限元分析程序开发过程 9

3.1. 程序需求分析 9

3.2. 程序架构设计 9

3.3. 程序模块详解 10

3.4. 数值算例 17

3.5. 计算结果对标分析 18

4. 有限元在汽车行业内的应用分析 20

4.1. 有限元在汽车行业内的发展现状 21

4.2. 具体应用分析 21

参考文献 22

1. C3D8 变截面悬臂梁受力分析的基本理论

1.1. 变截面悬臂梁受力分析

悬臂梁不管是在工程设计还是在机械设计中都有着广泛的应用，其有着结构简单、经济实用等优点。

悬臂梁是指梁的一端为不产生轴向、垂直位移和转动的固定支座，另一端为自由端（可以产生平行于轴向和垂直于轴向的力）。在实际工程分析中，大部分实际工程受力部件都可以简化为悬臂梁。但是悬臂梁的缺点在于它的受力性能不好，即使只是在悬臂梁末端施加一个较小的载荷，通过较长力臂的放大作用，也会对底部连接处产生一个很大的弯矩。因此，对悬臂梁强度校核前的受力分析和对其进行优化设计对工程和机械领域的发展都有着极大的意义。

本次分析按照悬臂梁的力学特点对变截面悬臂梁进行载荷施加和约束，在右端面建立三个自由度的约束，最左端上边线施加节点载荷，如下图所示。

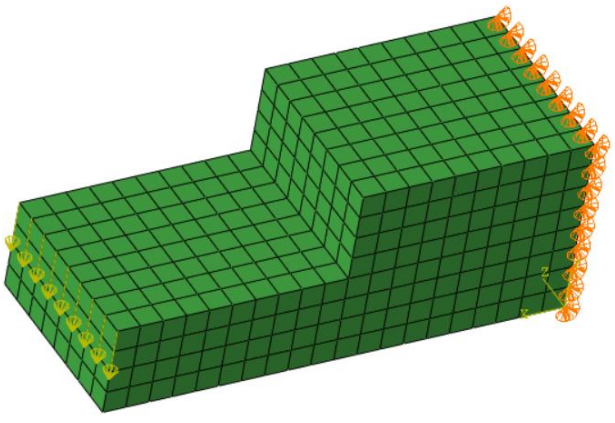


图 1-1 变截面悬臂梁受力示意图

1.2. 变截面悬臂梁有限元分析基本原理

本次变截面悬臂梁的受力分析采用线弹性静力学分析，线性静力学问题是简单且常见的有限元分析类型，不涉及任何非线性（材料非线性、几何非线性、接触等)，也不考虑惯性及时间相关的材料属性。考虑计算的精度及效率，采用 C3D8（三维 8 结点 6 面体网格）进行划分。有限元分析可大致分为前处理、计算、后处理三部分内容。本文利用 Matlab 及 Abaqus 分别对变截面悬臂梁进行有限元分析对标，分析其变形及受力。

1.3. 八结点六面体单元刚度矩阵

等参元节点坐标编号顺序如下：

Node	ξ_i	η_i	ζ_i
1	-1	-1	-1
2	+1	-1	-1
3	+1	+1	-1
4	-1	+1	-1
5	-1	-1	+1
6	+1	-1	+1
7	+1	+1	+1
8	-1	+1	+1

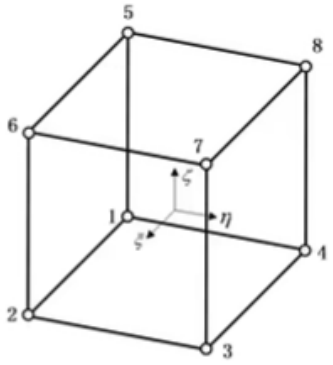


图 1-2 等参元结点坐标编号顺序示意图

对六面体网格的处理可建立形函数，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。其中插值函数为：

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

ξ, η, ζ 积分点坐标　　ξ_i, η_i, ζ_i 节点坐标

单元内的任意一点的坐标可以表示为节点上的坐标值。

$$\begin{cases} x = \sum_{i=1}^{N} N_i x_i \\ y = \sum_{i=1}^{N} N_i y_i, \\ z = \sum_{i=1}^{N} N_i z_i \end{cases}$$

有了坐标的表达式，由等参的定义，位移的表示和坐标完全一致，那么就可以得到单元内任意一点位移的表示：

$$\begin{cases} u = \sum_{i=1}^{N} N_i u_i \\ v = \sum_{i=1}^{N} N_i v_i \\ w = \sum_{i=1}^{N} N_i w_i \end{cases}$$

位移对坐标的微分就得到应变：

$$\{\varepsilon\} = \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \gamma_{xy} \\ \gamma_{yz} \\ \gamma_{zx} \end{Bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x} & 0 & 0 \\ 0 & \frac{\partial}{\partial y} & 0 \\ 0 & 0 & \frac{\partial}{\partial z} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial z} & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} & 0 & \frac{\partial}{\partial x} \end{bmatrix} \begin{Bmatrix} u \\ v \\ w \end{Bmatrix} = \sum_{i=1}^{N} \begin{bmatrix} \frac{\partial N_i}{\partial x} & 0 & 0 \\ 0 & \frac{\partial N_i}{\partial y} & 0 \\ 0 & 0 & \frac{\partial N_i}{\partial z} \\ \frac{\partial N_i}{\partial y} & \frac{\partial N_i}{\partial x} & 0 \\ 0 & \frac{\partial N_i}{\partial z} & \frac{\partial N_i}{\partial y} \\ \frac{\partial N_i}{\partial z} & 0 & \frac{\partial N_i}{\partial x} \end{bmatrix} \begin{Bmatrix} u_i \\ v_i \\ w_i \end{Bmatrix} = [B] \begin{Bmatrix} u_1 \\ v_1 \\ w_1 \\ \vdots \\ u_N \\ v_N \\ w_N \end{Bmatrix}$$

由输入的材料得到应力与应变的关系，即本构关系：

$$\{\sigma\} = \begin{Bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \tau_{xy} \\ \tau_{yz} \\ \tau_{zx} \end{Bmatrix} = [D]\{\varepsilon\}$$

$$= \frac{E(1-\nu)}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1 & \frac{\nu}{1-\nu} & \frac{\nu}{1-\nu} & 0 & 0 & 0 \\ \frac{\nu}{1-\nu} & 1 & \frac{\nu}{1-\nu} & 0 & 0 & 0 \\ \frac{\nu}{1-\nu} & \frac{\nu}{1-\nu} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1-2\nu}{2(1-\nu)} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1-2\nu}{2(1-\nu)} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1-2\nu}{2(1-\nu)} \end{bmatrix} \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \gamma_{xy} \\ \gamma_{yz} \\ \gamma_{zx} \end{Bmatrix}$$

则体的刚度矩阵是：

$$[K]^e = \iiint_{V^e} [B]^T[D][B]dV = \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} [B]^T[D][B]|J|dgdhdr$$

$$= \sum_{i=1}^{Pi}\sum_{j=1}^{Pj}\sum_{m=1}^{Pm}\left\{W_iW_jW_m \cdot ([B]^T[D][B]|J|)|_{g_i,h_j,r_m}\right\}$$

其中 B*如下：

$$\mathbf{B}^* = \begin{bmatrix} \frac{\partial N_1^*}{\partial x} & 0 & 0 & \cdots & \frac{\partial N_8^*}{\partial x} & 0 & 0 \\ 0 & \frac{\partial N_1^*}{\partial y} & 0 & \cdots & 0 & \frac{\partial N_8^*}{\partial y} & 0 \\ 0 & 0 & \frac{\partial N_1^*}{\partial z} & \cdots & 0 & 0 & \frac{\partial N_8^*}{\partial z} \\ \frac{\partial N_1^*}{\partial y} & \frac{\partial N_1^*}{\partial x} & 0 & \cdots & \frac{\partial N_8^*}{\partial y} & \frac{\partial N_8^*}{\partial x} & 0 \\ 0 & \frac{\partial N_1^*}{\partial z} & \frac{\partial N_1^*}{\partial y} & \cdots & 0 & \frac{\partial N_8^*}{\partial z} & \frac{\partial N_8^*}{\partial y} \\ \frac{\partial N_1^*}{\partial z} & 0 & \frac{\partial N_1^*}{\partial x} & \cdots & \frac{\partial N_8^*}{\partial z} & 0 & \frac{\partial N_8^*}{\partial x} \end{bmatrix}_{6\times 24}$$

为求解 B*，导数换元得：

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

中间部分的矩阵为雅可比矩阵 J：

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

利用几何场插值：

$$x=\sum_{i=1}^{8}N_i^*x_i\,,\ y=\sum_{i=1}^{8}N_i^*y_i\,,\ z=\sum_{i=1}^{8}N_i^*z_i$$

可得雅可比矩阵插值形式：

$$\mathbf{J}=\begin{bmatrix}\sum_{i=1}^{8}\frac{\partial N_i^*}{\partial\xi}x_i & \sum_{i=1}^{8}\frac{\partial N_i^*}{\partial\xi}y_i & \sum_{i=1}^{8}\frac{\partial N_i^*}{\partial\xi}z_i\\ \sum_{i=1}^{8}\frac{\partial N_i^*}{\partial\eta}x_i & \sum_{i=1}^{8}\frac{\partial N_i^*}{\partial\eta}y_i & \sum_{i=1}^{8}\frac{\partial N_i^*}{\partial\eta}z_i\\ \sum_{i=1}^{8}\frac{\partial N_i^*}{\partial\zeta}x_i & \sum_{i=1}^{8}\frac{\partial N_i^*}{\partial\zeta}y_i & \sum_{i=1}^{8}\frac{\partial N_i^*}{\partial\zeta}z_i\end{bmatrix}$$

$$=\begin{bmatrix}\frac{\partial N_1^*}{\partial\xi} & \frac{\partial N_2^*}{\partial\xi} & \cdots & \frac{\partial N_8^*}{\partial\xi}\\ \frac{\partial N_1^*}{\partial\eta} & \frac{\partial N_2^*}{\partial\eta} & \cdots & \frac{\partial N_8^*}{\partial\eta}\\ \frac{\partial N_1^*}{\partial\zeta} & \frac{\partial N_2^*}{\partial\zeta} & \cdots & \frac{\partial N_8^*}{\partial\zeta}\end{bmatrix}\begin{bmatrix}x_1 & y_1 & z_1\\ x_2 & y_2 & z_2\\ \vdots & \vdots & \vdots\\ x_8 & y_8 & z_8\end{bmatrix}$$

对其求逆，带入到换元等式中：

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

可得出需要求解的 B*：

$$\mathbf{B}^*=\begin{bmatrix}\frac{\partial N_1^*}{\partial x} & 0 & 0 & \cdots & \frac{\partial N_8^*}{\partial x} & 0 & 0\\ 0 & \frac{\partial N_1^*}{\partial y} & 0 & \cdots & 0 & \frac{\partial N_8^*}{\partial y} & 0\\ 0 & 0 & \frac{\partial N_1^*}{\partial z} & \cdots & 0 & 0 & \frac{\partial N_8^*}{\partial z}\\ \frac{\partial N_1^*}{\partial y} & \frac{\partial N_1^*}{\partial x} & 0 & \cdots & \frac{\partial N_8^*}{\partial y} & \frac{\partial N_8^*}{\partial x} & 0\\ 0 & \frac{\partial N_1^*}{\partial z} & \frac{\partial N_1^*}{\partial y} & \cdots & 0 & \frac{\partial N_8^*}{\partial z} & \frac{\partial N_8^*}{\partial y}\\ \frac{\partial N_1^*}{\partial z} & 0 & \frac{\partial N_1^*}{\partial x} & \cdots & \frac{\partial N_8}{\partial z} & 0 & \frac{\partial N_8}{\partial x}\end{bmatrix}_{6\times24}$$

2. C3D8 变截面悬臂梁有限元受力分析过程

2.1. Abaqus 有限元分析软件简介

Abaqus 是一套功能强大的工程模拟的有限元软件，其解决问题的范围从相对简单的线性分析到许多复杂的非线性问题。ABAQU 作为通用的模拟工具，可以解决大量结构（应力 / 位移）问题。

Abaqus 目前有两个主求解器模块：Abaqus/Standard 和 Abaqus/Explicit。可以分析复杂的固体力学结构力学系统，特别是能够驾驭非常庞大复杂的问题和模拟高度非线性问题。

2.2. Abaqus 前处理

首先建立三维实体模型，大截面尺寸为 40*40*50mm，小截面尺寸为 40*20*50mm，在 Abaqus 中进行网格划分，网格尺寸为 5mm,其中结点总数：1341，单元总数：960，单元类型：C3D8。其具体的形状尺寸如下图所示。

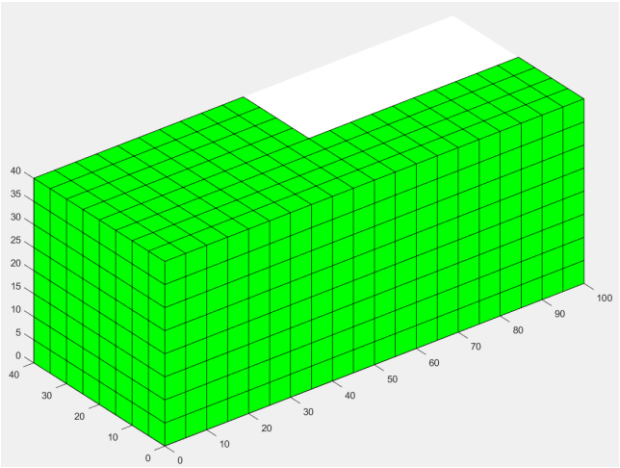


图 2-1 变截面悬臂梁尺寸示意图

在大截面端面所有结点处进行三个自由度的约束，在小截面上边线所有结点处施加 Y 方向的 500N 的载荷（结点 ID 查找 inp 文件，为 2, 3, 24, 25, 26, 27, 28, 29, 30），同时对其进行材料属性指派，使用最常用的钢材（steel）作为悬臂梁的材料，其弹性模量为 E=210000MPa，泊松比为 0.3。

2.3. Abaqus 求解计算

采用静态通用（Static，Genera）分析步进行分析。Abaqus/CAE 会自动创建一个初始分析步（Initial step），可以在其中施加边界条件，另外必须自己创建后续分析步（analysis step），用来施加载荷。由于模型相对简单，变形不大，因此可关闭几何非线性，最大增量步数为 1000，初始增量步为 0.01，最小为 1E-05，最大为 0.01。根据分析要求创建历程输出，输出变量 E、S（MISES、S11、S12、S13 等）、U（Umag、U1、U2、U3）。最后提交作业进行计算分析。

2.4. Abaqus 后处理

成功求解计算后，根据 odb 文件，进入后处理可视化模块，对需要的输出变量进行观察分析。可通过观察云图、输出具体数据、观察动画、切面视图等一系列后处理操作对模型进行研究分析，具体的位移、应力、应变云图如下图所示。

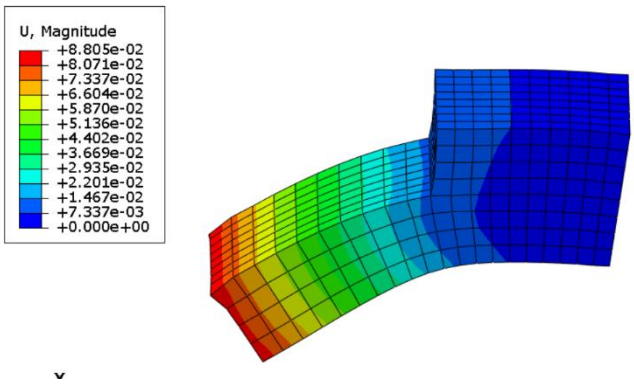


图 2-2Abaqus 位移 Umag 云图

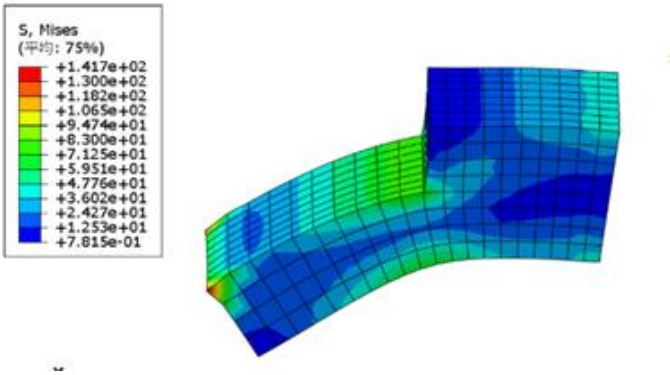


图 2-3 Abaqus　Mises 云图

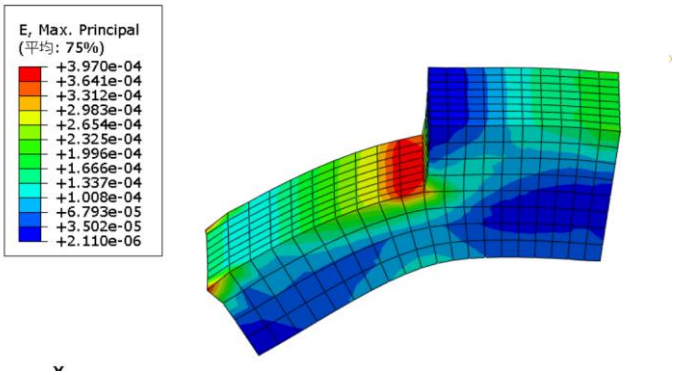


图 2-4 Abaqus　应变 Emax 云图

这里只简单展示利用 Abaqus 进行仿真计算所得的个别云图，具体数据及结果分析将在后面小节做详细介绍。另外，为了放大分析效果，将施加载荷设置为 500N，且调大了变形系数使得云图的变形看起来更加直观。

3. C3D8 变截面悬臂梁有限元分析程序开发过程

3.1. 程序需求分析

悬臂梁作为工程应用中常见的承力构件之一，对整个工程机械的静、动态性能方面起重要作用，应保证其具有变形量小、抗振性好、质量轻等优点。为了减轻悬臂梁的重量，提高悬臂梁的抗振性，国内外学者在机械结构方面做了大量研究。传统的设计方法虽然能够设计出符合要求的悬臂梁，但很难做到高刚度、轻量化。

随着互联网的蓬勃发展，利用现代设计手段可以在计算机上实现悬臂梁模型的建立、仿真、优化等操作。基于变截面悬臂梁有限元分析的基本原理，利用 Matlab 编写结构受力变形的求解程序，可快速高效的得到变截面悬臂梁的受力及位移云图及其具体数据。变截面悬臂梁有限元分析程序开发，可针对悬臂梁不同形状、不同材料本构、不同受力情况做快速调整，并进行高效分析，给悬臂梁的结构研究带来极大便利。

3.2. 程序架构设计

基于上述提出的各项需求分析，本次程序开发过程基于技术路线进行模块划分。根据有限元分析的特点及求解流程（前处理、求解计算、后处理），对本次程序进行设计。

总体求解思路为读取 inp 相关数据，计算一阶六面体单元的刚度矩阵，组装整体刚度矩阵，施加边界条件、施加载荷，计算节点位移，计算单元应力。

主要包括主程序、一阶六面体单元刚度矩阵、一阶六面体单元形

函数、应力应变矩阵、求解主程序、应力应变计算程序、云图绘制程序、输出结果程序、Inp 文件读取程序几个模块。

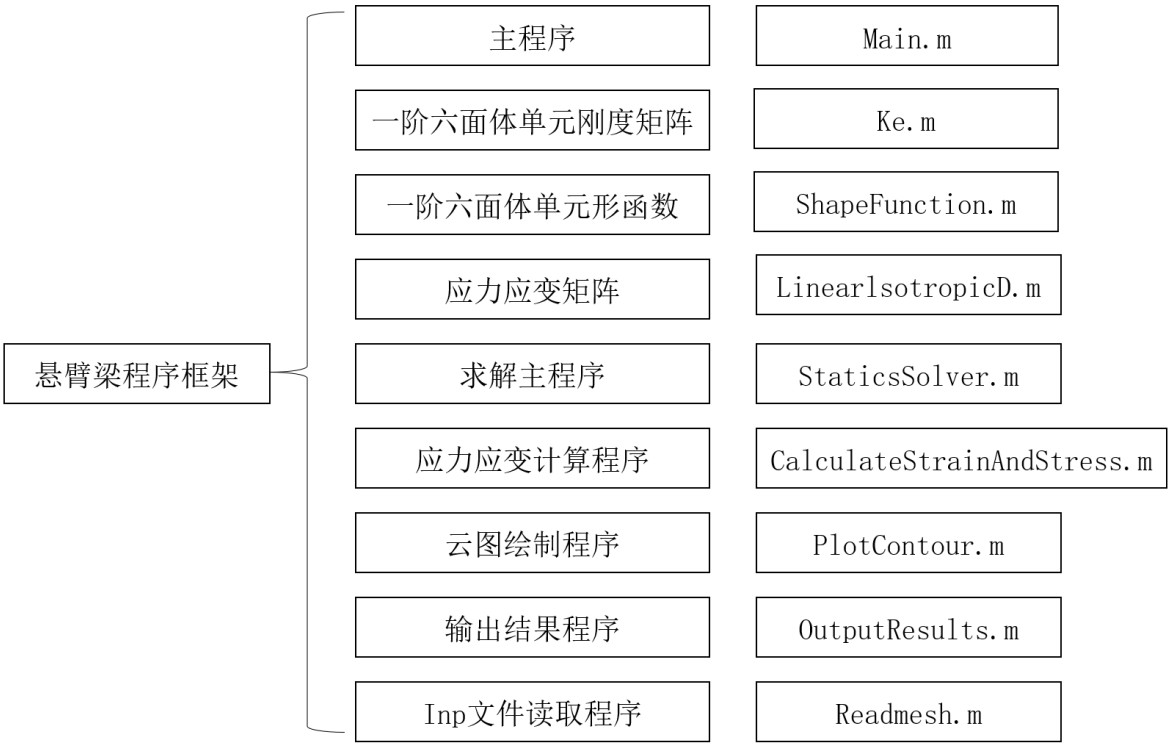


图 3-1 Matlab 程序框架图

3.3. 程序模块详解

本次悬臂梁的程序开发是借用网络上已有的脚本程序进行开发，在已有的基础上针对自己的需求进行修改完善，对于建立的悬臂梁模型进行有限元分析。

下面将对主程序、一阶六面体单元刚度矩阵、一阶六面体单元形函数、应力应变矩阵、求解主程序、应力应变计算程序、云图绘制程序、输出结果程序、Inp 文件读取程序几个模块做详细介绍。

3.3.1. 主程序

主程序大致包含三个部分。

第一部分：使用 Readmesh 函数读取 inp 文件，得到结点坐标信

息，单元网格信息，在 inp 文件中获取外力及约束结点编号，设置与 Abaqus 前处理设置相同的外力矩阵及约束矩阵，在九个结点处设置 Y 方向上 500N 的均质载荷，并约束一端面的所有结点的三个自由度，并对此悬臂梁赋予材料属性。

第二部分调用应力应变求解矩阵及位移。

第三部分：设置输出内容，输出具体参数的 txt 文件及位移、应力云图。云图绘制采用 PlotContour 函数，本程序仅绘制出网格图、位移云图及应力云图，因为研究对象是线弹性体，位移是第一求解未知量，最为准确，应力云图通过应力矩阵变换而得。

```
function Main()
%读取inp文件获得节点坐标信息Nodes及单元信息Elements
[Nodes, Elements] = Readmesh( 'DataFile.inp' );
% 外力矩阵 Forces=[受力节点  受力方向(1,2,3分别代表x,y,z)  外力大小]  外力节点的编号在inp文件里面找
Forces=[2 2 -500;3 2 -500;24 2 -500;25 2 -500;26 2 -500;27 2 -500;28 2 -500;29 2 -500;30 2 -500;];
%约束节点的编号在inp文件里面找
ConNumber=[9,12,  13,  14, 116, 117, 118, 119, 120, 121, 122, 141, 142, 143, 144, 145....
,146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161....
 ,516, 517, 518, 519, 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531....
 ,532, 533, 534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 547....
 ,548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 560, 561, 562, 563....
 ,564,];
%约束矩阵 Constraints=[强制位移节点  强制位移方向(1,2,3分别代表x,y,z)  强制位移大小]
Constraints=zeros(size(ConNumber,2)*3,3);
for i=1:size(ConNumber,2)
Constraints(3*i-2:3*i,:)=[ConNumber(i) 1 0;ConNumber(i) 2 0;ConNumber(i) 3 0;];
end
E=210000; %弹性模量
u=0.3;    %泊松比
%调用应变应力矩阵D
D=LinearIsotropicD(E,u);
U=StaticsSolver(E,u,Forces,Constraints,Nodes,Elements);
% 输出结果
OutputTXT = fopen('Results.txt','w'); %打开一个可写文件，用于写入计算结果
OutputResults(OutputTXT,Nodes,Elements,D,U)%调用输出结果文件
fclose(OutputTXT);
edit('Results.txt')
end
```

图 3-2 Matlab 主程序

3.3.2. 应力应变矩阵

$$\mathbf{D}=\frac{E}{(1+\mu)(1-2\mu)}\times\begin{bmatrix}1-\mu & \mu & \mu & 0 & 0 & 0\\ & 1-\mu & \mu & 0 & 0 & 0\\ & & 1-\mu & 0 & 0 & 0\\ & 对 & & (1-2\mu)/2 & 0 & 0\\ & 称 & & & (1-2\mu)/2 & 0\\ & & & & & (1-2\mu)/2\end{bmatrix}_{6\times6}$$

图 3-3 应力应变矩阵

```
%  D线弹性材料应力-应变矩阵
%  E弹性模量
%  u泊松比
function [D]=LinearIsotropicD(E,u)
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
end
```

图 3-4 Matlab 应力应变矩阵

根据应力应变矩阵转换成 Matlab 中代码形式，输入弹性模量及泊松比即可输出相应的 D。

3.3.3. 形函数矩阵

```
%  N局部坐标下的形函数矩阵
%  NDerivative形函数矩阵对全局坐标的导数
%  JacobiDET雅可比行列式
%  GaussPoint高斯点坐标
%  ElementNodeCoordinate单元节点坐标（8*3，每一行代表一个节点的坐标）
function [N,NDerivative,JacobiDET] = ShapeFunction(GaussPoint,ElementNodeCoordinate)
 %等参元坐标  每一列代表一个点的坐标
ParentNodes=[-1  1  1 -1 -1  1  1 -1;
    -1 -1  1  1 -1 -1  1  1;
    -1 -1 -1 -1  1  1  1  1];
N=zeros(8,1); %初始化形函数矩阵8*1
ParentNDerivative=zeros(3,8);%初始化形函数对局部坐标的导数矩阵3*8
%计算形函数及形函数对局部坐标导数
for I=1:8
    XPoint = ParentNodes(1,I);
    YPoint = ParentNodes(2,I);
    ZPoint = ParentNodes(3,I);
    ShapePart = [1+GaussPoint(1)*XPoint 1+GaussPoint(2)*YPoint 1+GaussPoint(3)*ZPoint]
    N(I) = 0.125*ShapePart(1)*ShapePart(2)*ShapePart(3);
    ParentNDerivative(1,I) = 0.125*XPoint*ShapePart(2)*ShapePart(3);
    ParentNDerivative(2,I) = 0.125*YPoint*ShapePart(1)*ShapePart(3);
    ParentNDerivative(3,I) = 0.125*ZPoint*ShapePart(1)*ShapePart(2);
end
Jacobi = ParentNDerivative*ElementNodeCoordinate;%计算雅可比矩阵
JacobiDET = det(Jacobi);%计算雅可比行列式
JacobiINV=inv(Jacobi);%对雅可比行列式求逆
NDerivative=JacobiINV*ParentNDerivative;%利用雅可比行列式的逆计算形函数对结构坐标的导数
end
```

图 3-5 Matlab 形函数矩阵

N 用于构造插值函数求解结点应力应变，NDerivative 形函数矩阵、JacobiDET 雅可比行列式用于求解单刚。

```
for I=1:8
    XPoint = ParentNodes(1,I);
    YPoint = ParentNodes(2,I);
    ZPoint = ParentNodes(3,I);
    ShapePart = [1+GaussPoint(1)*XPoint 1+GaussPoint(2)*YPoint 1+GaussPoint(3)*ZPoint];
    N(I) = 0.125*ShapePart(1)*ShapePart(2)*ShapePart(3);
    ParentNDerivative(1,I) = 0.125*XPoint*ShapePart(2)*ShapePart(3);
    ParentNDerivative(2,I) = 0.125*YPoint*ShapePart(1)*ShapePart(3);
    ParentNDerivative(3,I) = 0.125*ZPoint*ShapePart(1)*ShapePart(2);
```

图 3-6 Matlab 形函数矩阵 2

其中 N（i）即为上文提到的形函数：

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

ParentNDerivative 为形函数对高斯点坐标的导数：

$$N^*_{i,\xi} = \frac{1}{8}\xi_i(1+\eta_i\eta)(1+\zeta_i\zeta)$$
$$N^*_{i,\eta} = \frac{1}{8}\eta_i(1+\xi_i\xi)(1+\zeta_i\zeta)$$
$$N^*_{i,\zeta} = \frac{1}{8}\zeta_i(1+\xi_i\xi)(1+\eta_i\eta)$$

最后计算形函数对全局坐标的导数：

$$\begin{bmatrix} \frac{\partial N_1^*}{\partial x} & \frac{\partial N_2^*}{\partial x} & \cdots & \frac{\partial N_8^*}{\partial x} \\ \frac{\partial N_1^*}{\partial y} & \frac{\partial N_2^*}{\partial y} & \cdots & \frac{\partial N_8^*}{\partial y} \\ \frac{\partial N_1^*}{\partial z} & \frac{\partial N_2^*}{\partial z} & \cdots & \frac{\partial N_8^*}{\partial z} \end{bmatrix}_{3\times 8} = \mathbf{J}^{-1} \begin{bmatrix} \frac{\partial N_1^*}{\partial \xi} & \frac{\partial N_1^*}{\partial \xi} & \cdots & \frac{\partial N_8^*}{\partial \xi} \\ \frac{\partial N_1^*}{\partial \eta} & \frac{\partial N_1^*}{\partial \eta} & \cdots & \frac{\partial N_8^*}{\partial \eta} \\ \frac{\partial N_1^*}{\partial \zeta} & \frac{\partial N_1^*}{\partial \zeta} & \cdots & \frac{\partial N_8^*}{\partial \zeta} \end{bmatrix}_3$$

3.3.4. 一阶六面体单元刚度矩阵

根据六面体单元的特性利用 For 循环 8 个高斯点（2*2*2），高斯点坐标为（GP1,GP2,GP3），最终得出体单元的刚度矩阵：

$$[K]^e = \iiint_{V^e} [B]^T[D][B]dV = \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} [B]^T[D][B]|J|dgdhdr$$
$$= \sum_{i=1}^{Pi}\sum_{j=1}^{Pj}\sum_{m=1}^{Pm} \left\{W_iW_jW_m \cdot ([B]^T[D][B]|J|)|_{g_i,h_j,r_m}\right\}$$

具体的 Matlab 代码如下：

```
%%%%%%%%%%%  一阶六面体单元单位刚度矩阵  %%%%%%%%%%%
% Ke单元刚度矩阵
% D计算各向同性线弹性材料应力-应变矩阵
% ElementNodeCoordinate单元节点坐标（8*3，每一行代表一个节点的坐标）
function [Ke]=Ke(D, ElementNodeCoordinate)
% 高斯积分点坐标
GaussCoordinate=[-0.57735026918963D0, 0.57735026918963D0];
%高斯积分点权重
GaussWeight=[1.00000000000000D0, 1.00000000000000D0];
%初始化单元刚度阵
Ke=zeros(24,24);
%循环高斯点
for X=1:2
    for Y=1:2
        for Z=1:2
            GP1=GaussCoordinate(X); GP2=GaussCoordinate(Y); GP3=GaussCoordinate(Z); %高斯点坐标
            % 计算形函数对总体坐标的导数（NDerivative）及雅可比矩阵行列式（JacobiDET）
            [~,NDerivative, JacobiDET] = ShapeFunction([GP1 GP2 GP3], ElementNodeCoordinate);
            Coefficient=GaussWeight(X)*GaussWeight(Y)*GaussWeight(Z)*JacobiDET;
            %计算B矩阵  利用形函数对总体坐标的导数（NDerivative）对B进行计算
            B=zeros(6,24);
            for I=1:8
                COL=(I-1)*3+1:(I-1)*3+3;
                B(:,COL)=[NDerivative(1,I) 0         0;
                    0         NDerivative(2,I) 0;
                    0         0         NDerivative(3,I);
                    NDerivative(2,I) NDerivative(1,I) 0;
                    0         NDerivative(3,I) NDerivative(2,I);
                    NDerivative(3,I) 0         NDerivative(1,I)];
            end
            Ke=Ke+Coefficient*B'*D*B;  %叠加刚度阵
        end
    end
```

图 3-7 Matlab 单元刚度矩阵

3.3.5. 求解主程序

求解出程序实际上是求解位移向量，对上部分求解的单元刚度矩阵进行总和，求解总体刚度矩阵，if 语句施加外载荷，乘大数法施加约束，最终求解位移 u=k/force。

```
function [U]=StaticsSolver(E,u,Forces,Constraints,Nodes,Elements)
Dof=3;
NodeCount = size(Nodes,1);  % 节点个数
ElementCount= size(Elements,1);  %单元个数
Dofs = Dof*NodeCount; %总自由度数
U=sparse(Dofs,1);   % 初始化结构位移
K = sparse(Dofs,Dofs);  %初始化总体刚度阵
Force = sparse(Dofs,1);  %初始化外力向量
%计算应力-应变矩阵D
D=LinearIsotropicD(E,u);
for I=1:ElementCount
    % 单元节点坐标
    ElementNodeCoordinate=Nodes(Elements(I,:),:);
    % 计算单刚
    ElementStiffnessMatrix=Ke(D,ElementNodeCoordinate);
    % 计算单元节点自由度编号
    ElementNodeDOF=zeros(1,24);
    for J=1:8
        II=(J-1)*Dof+1;
        ElementNodeDOF(II:II+2)=(Elements(I,J)-1)*Dof+1:(Elements(I,J)-1)*Dof+3;
    end
    K(ElementNodeDOF,ElementNodeDOF)=K(ElementNodeDOF,ElementNodeDOF)+ElementStiffnessMatrix;
end
% 施加外力
if size(Forces,1)>0
    ForceDOF = Dof*(Forces(:,1)-1)+Forces(:,2);   %计算外力自由度编号
    Force(ForceDOF) = Force(ForceDOF) + Forces(:,3);
end
% 乘大数法施加位移约束
BigNumber=1e8;
ConstraintsNumber=size(Constraints,1);
if ConstraintsNumber~=0
    FixedDof=Dof*(Constraints(:,1)-1)+Constraints(:,2);  %被约束的自由度编号(列向量)
    for i=1:ConstraintsNumber
        K(FixedDof(i),FixedDof(i))=K(FixedDof(i),FixedDof(i))*BigNumber;
        Force(FixedDof(i))=Constraints(i,3)*K(FixedDof(i),FixedDof(i));
    end
end
%计算位移
U = K\Force;
```

图 3-8 Matlab 求解主程序

3.3.6. 计算应力应变程序

计算应力应变矩阵通过位移矩阵求解高斯点应力应变，再采用应力磨平，构造插值函数，得到节点处的应力应变。

```
function [NodeStrain,NodeStress,GaussStrain,GaussStress]=CalculateStrainAndStress(U,D,Nodes,Elements)
ElementCount= size(Elements,1);  %单元个数
GaussCoordinate=[-0.577350269189863D0, 0.577350269189863D0];  %高斯积分点坐标
GaussWeight=[1.0000000000000000D0, 1.0000000000000000D0];  %高斯积分点权重
GaussPointNumber=0;%高斯积分点编号
INODE=0;% 节点编号
Dof=3;
%定义矩阵维度 加快运行速度
GaussStrain=zeros(6,ElementCount*8);
GaussStress=zeros(6,ElementCount*8);
NodeStrain=zeros(6,ElementCount*8);
NodeStress=zeros(6,ElementCount*8);
%循环组装总刚
for I=1:ElementCount
    % 单元节点坐标
    ElementNodeCoordinate=Nodes(Elements(I,:),:);
    % 计算单元节点自由度编号
    ElementNodeDOF=zeros(1,24);
    for J=1:8
        II=(J-1)*Dof+1;
        ElementNodeDOF(II:II+2)=(Elements(I,J)-1)*Dof+1:(Elements(I,J)-1)*Dof+3;
    end
    K=1;
    InterpolationMatrix=zeros(8,8);%求解节点应力应变的插值矩阵
    %循环高斯点
    for X=1:2
        for Y=1:2
            for Z=1:2
                E1=GaussCoordinate(X); E2=GaussCoordinate(Y); E3=GaussCoordinate(Z);
                GaussPointNumber = GaussPointNumber + 1;
                % 计算局部坐标下的形函数及形函数导数
                [N,NDerivative, ~] = ShapeFunction([E1 E2 E3], ElementNodeCoordinate);
                ElementNodeDisplacement=U(ElementNodeDOF);
                ElementNodeDisplacement=reshape(ElementNodeDisplacement,Dof,8);
                % 计算高斯点应变 GausspointStrain3_3 3*3的应变矩阵
```

图 3-9 Matlab 计算应力应变程序

3.3.7. 云图输出程序

对于 C3D8 三维八节点体单元，按顺序给定结点的坐标及相应的值，以及每个面上点的连接顺序，定义元胞数组，将数据信息放入元胞数组，对不同单元的同一结点处进行磨平处理，用 patch 函数绘制三维图像。

```
%%%%%%%%%%  一阶六面体单元绘制云图程序  %%%%%%%%%%
%  Nodes节点坐标信息
%  Elements单元信息
%  U位移矩阵
%  Component云图上节点的值，可以是位移、应力、应变等
function PlotContour(Nodes,Elements,U,Component)
NodeCount = size(Nodes,1) ;  %  节点个数
ElementCount = size(Elements,1) ;    %单元个数
ElementNodeCount=8;  %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(ElementNodeCount,ElementCount) ;
Y = zeros(ElementNodeCount,ElementCount) ;
Z = zeros(ElementNodeCount,ElementCount) ;
value = zeros(ElementNodeCount,ElementCount) ;
%判断矩阵类型（位移，应力，应变）
if size(Component,1)>1
    for i=1:ElementCount
        nd=Elements(i,:);
        value(:,i) = Component(nd) ;
    end
else
    %先进行磨平，再把Component行向量转化成矩阵形式
    Difference=max(Component)-min(Component);%全域上的最大值-最小值
    AVG=0.75;  % 默认阈值75%
```

图 3-10 Matlab 输出云图程序

3.3.8. 输出结果程序

输出结果程序用于输出结点位移信息、高斯点的应力应变信息及所需云图的输出。

```
%求得Umag位移矩阵
Umag=zeros(NodeCount,1);
for i=1:NodeCount
    Umag(i)=sqrt(U(3*i-2)^2+U(3*i-1)^2+U(3*i)^2);
end
  %将文节点位移、高斯点应力应变写入TXT中
  fprintf(OutputTXT,'\r\n Node          U1          U2          U3');
  for I=1:NodeCount
    II=Dof*(I-1);
    fprintf(OutputTXT,'\r\n%5d %11.3e %11.3e %11.3e',I,U(II+1:II+3)+0);
  end
  fprintf(OutputTXT,'\r\n\r\nElement GaussStrain\r\n');
  fprintf(OutputTXT,'\r\n         E11         E22         E33         E12         E23         E13');
  for I=1:ElementCount
    fprintf(OutputTXT,'\r\nElement %5d',I);
    II=(I-1)*8;
    fprintf(OutputTXT,'\r\n%11.3e %11.3e %11.3e %11.3e %11.3e %11.3e',GaussStrain(1:6,II+1:II+8));
  end
  fprintf(OutputTXT,'\r\n\r\nElement GaussStress\r\n');
  fprintf(OutputTXT,'\r\n         S11         S22         S33         S12         S23         S13');
  for I=1:ElementCount
    fprintf(OutputTXT,'\r\nElement %5d',I);
    II=(I-1)*8;
    fprintf(OutputTXT,'\r\n%11.3e %11.3e %11.3e %11.3e %11.3e %11.3e',GaussStress(1:6,II+1:II+8));
  end
  fprintf(OutputTXT,'\r\n\r\n');
  fprintf(1,'\t\t *** Successful end of program ***\n');
%绘制变形前网格
  for i=1:1:size(Elements,1)
    points=Nodes(Elements(i,:),:);
    mesh=1:1:8;%网格信息
    %六面体单元节点坐标
    vertices_matrix = [points(mesh(1,:),1),points(mesh(1,:),2),points(mesh(1,:),3)];
    %六面体单元节点顺序
    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];%给出每个面的节点序号
    patch('vertices', vertices_matrix,'faces',faces_matrix,'facecolor','g');
    view(3);hold on%绘图
end
axis equal
alpha(1);
```

图 3-11 Matlab 结果输出程序

3.3.9. 读取 inp 程序

```
function [Nodes, Elements] = Readmesh( fname )
fid = fopen(fname,'rt');  %fname文件名   r读取   t以txt格式打开
S = textscan(fid,'%s','Delimiter','\n'); %已经打开的文件   字符向量形式设
S = S{1};
%找到Node关键字所在的位置
idxS = strfind(S, 'Node');  %返回元胞数组 若数组中没有相应元素，则返回空
idx1 = find(not(cellfun(@isempty, idxS))); %cellfun(fun,A) 对元胞数组A
%找到Element关键字所在的位置
idxS = strfind(S, 'Element');
idx2 = find(not(cellfun(@isempty, idxS)));
%找到Nset关键字所在位置
idxS = strfind(S, 'Nset');
idx3 = find(not(cellfun(@isempty, idxS)));
% 取出节点信息(元胞数组)
Nodes = S(idx1(1)+1:idx2(1)-1);  %以元胞数组形式取出
%将元胞数组转换为矩阵
Nodes = cell2mat(cellfun(@str2num,Nodes,'UniformOutput',false));  %'Un
% 取出单元(元胞数组)
elements = S(idx2+1:idx3(1)-1) ;
% 将元胞数组转换为矩阵
Elements = cell2mat(cellfun(@str2num,elements,'UniformOutput',false));
Nodes=Nodes(:,2:end);
Elements=Elements(:,2:end);
end
```

图 3-12 Matlab 读取 inp 文件程序

除了读取六面体单元，该程序可读取多种类型单元(三角形、四边形、三棱柱、六面体)的节点及单元信息。

3.4. 数值算例

此次变截面悬臂梁设计为左右两个矩形端面，在大截面端面所有结点处施加三个自由度的约束（为简便查找结点，在 Abaqus 中提前设置好约束，然后用记事本打开 inp 文件，*Boundary 下即为约束信息，可获取结点 ID），在小截面上边线所有结点处施加 Y 方向的 500N 的载荷（结点 ID 查找 inp 文件，*Cload 下侧为载荷信息，可获取结点 ID 为 2，3，24，25，26，27，28，29，30），同时对其进行材料属性指派，使用最常用的钢材（steel）作为悬臂梁的材料，其弹性模量为 E=210000MPa，泊松比为 0.3。

具体计算结果如下：

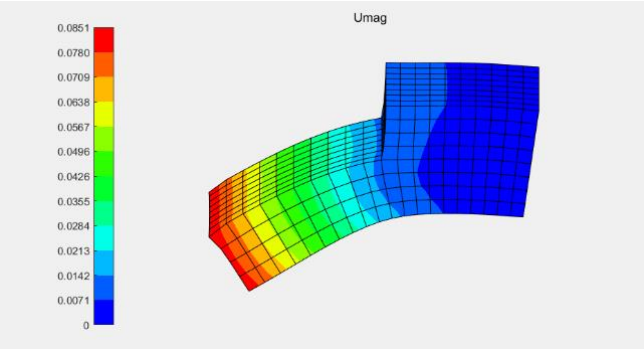


图 3-13 Matlab 位移 Umag 云图

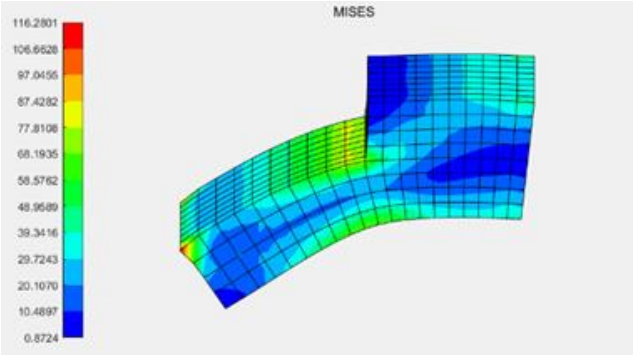


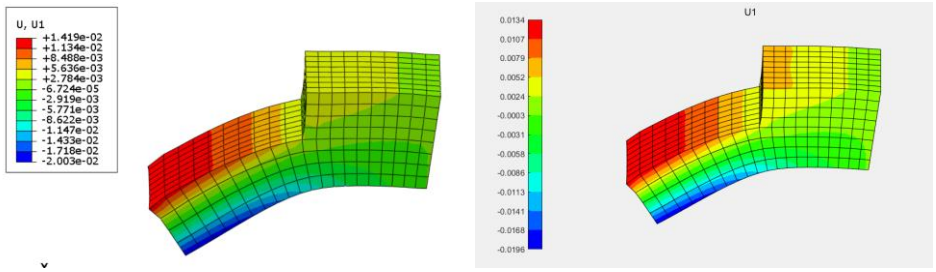
图 3-14 Matlab Mises 云图

这里只简单展示利用 Matlab 进行仿真计算所得的个别云图，具体数据及结果分析将在下一小节做详细介绍。

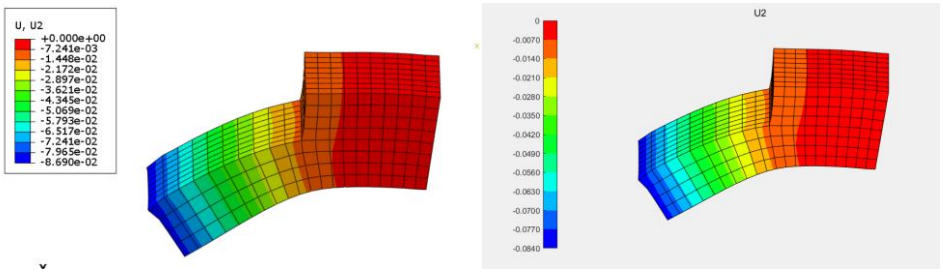
3.5. 计算结果对标分析

对比此次分析的各参数的最大值如下表所示，列出位移、应力云图对比情况，由于参数较多，这里只列举个别参数的对比情况。

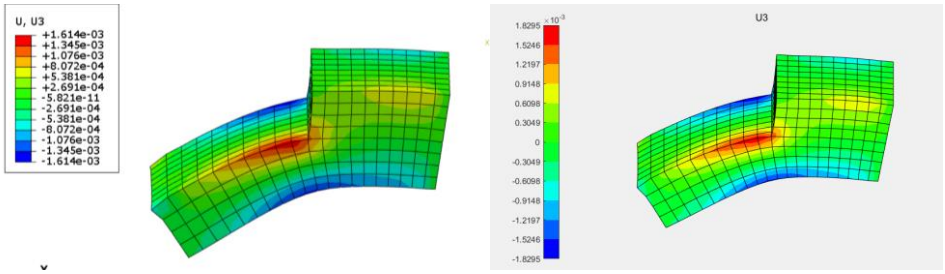
下面将对 U1、U2、U3、Umag 位移及 S11、S22、S33、MSES 应力云图进行对比。



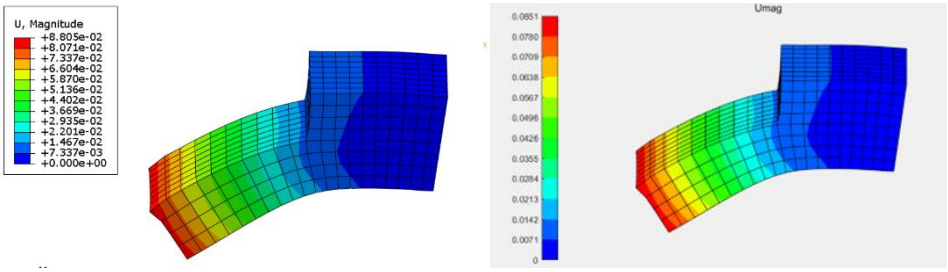
(a)U1 对比图 左 Abaqus 右 Matlab



(b)U2 对比图 左 Abaqus 右 Matlab

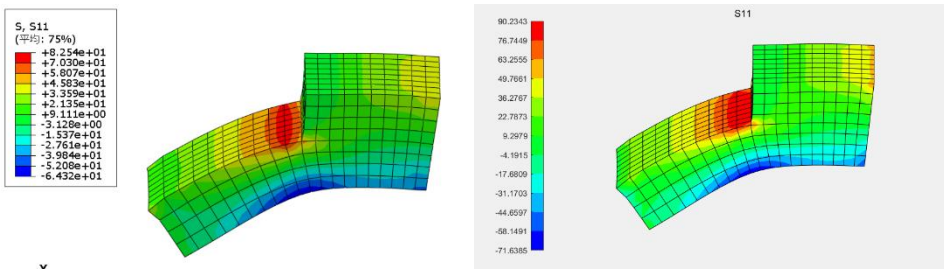


(c)U3 对比图 左 Abaqus 右 Matlab

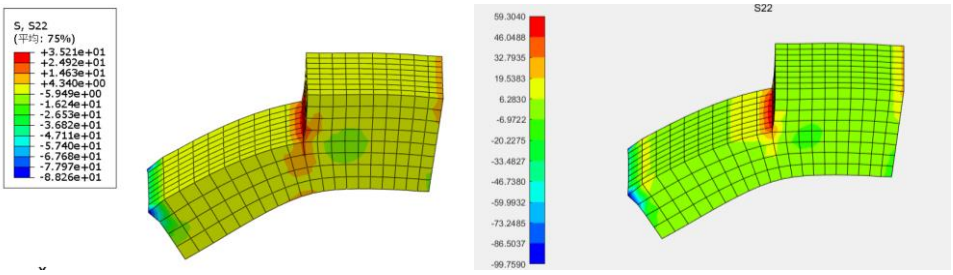


(d)Umag 对比图 左 Abaqus 右 Matlab

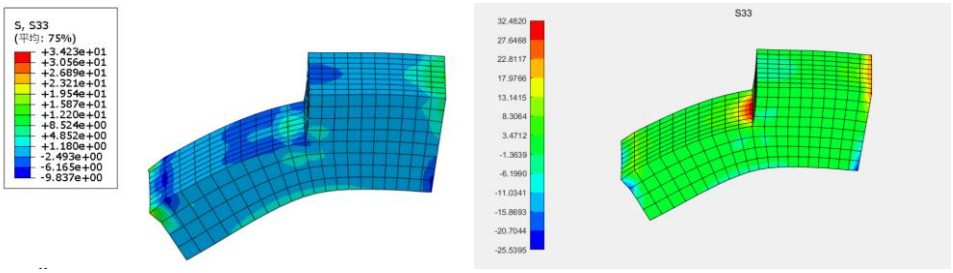
图 3-15 位移云图对比



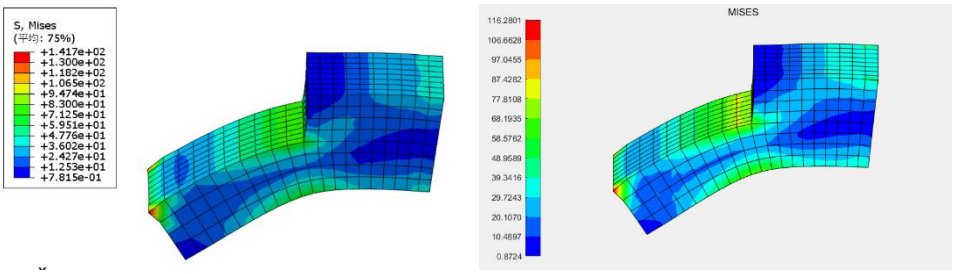
(a)S11 对比图 左 Abaqus 右 Matlab



(b)S22 对比图 左 Abaqus 右 Matlab



(c)S33 对比图 左 Abaqus 右 Matlab



(d)MISES 对比图 左 Abaqus 右 Matlab

图 3-16 应力云图对比

由云图对比可大致看出，对于位移云图利用 Abaqus 仿真分析与 Matlab 有限元分析得到的结果一致性较好，而对于应力云图对标稍有误差，这是正常现象。由上文提到的此次分析用到的位移及应力应变计算的基本原理来看，因为研究对象是线弹性体，位移是第一求解未知量，最为准确。应力应变是通过应力应变矩阵变换而得，而 Abaqus 有自己的应力应变换算方法，因此存在一定差距。

下表为具体的参数对比，并分析它们之间存在的误差大小。取 Abaqus 仿真分析与 Matlab 有限元分析得到的 U1、U2、U3、Umag 位移及 S11、S22、S33、MSES 应力最大值进行对比，然后取一结点 27，也就是施加结点的中间结点及位移和应力最大点，做受力分析后的数据对比。

表 3-1 位移、应力最大值对比

参数	Matlab	Abaqus	差值	参数	Matlab	Abaqus	差值
U1	0.0134	0.01419	5.57%	S11	90.23	82.54	8.52%
U2	-0.0840	-0.0869	3.34%	S22	59.30	35.21	40.62%
U3	0.001829	0.001614	11.75%	S33	32.48	34.23	5.11%
Umag	0.0851	0.08805	3.35%	MISES	116.28	141.7	17.94%

表 3-2 结点 27 位移、应力对比

参数	Matlab	Abaqus	差值
U1	0.01323	0.0138469	4.46%
U2	-0.08253	-0.0849893	2.89%
U3	-1.043E-15	0	/

经过参数对比，与云图结果一直，位得到的结果一致性较好，而对于应力对标稍有误差，而单一结点的对标结果比最大值好，保持在 5%以内。

4. 有限元在汽车行业内的应用分析

基于有限元方法的数值计算已被广泛应用于汽车行业的设计和开发。一方面由于用户的体验需求持续增长，另一方面，前处理、计算和后处理的有限元分析功能已经达到了一定的完善状态，允许交互式处理和结果的图形呈现。

4.1. 有限元在汽车行业内的发展现状

如何利用高效合理的设计方法降低制造成本和重量，同时又用合适的材料保证车身结构的刚度、强度和稳定性，是汽车制造商现在应该解决的最大问题。汽车的零部件可以达到万件以上，而且大多是非标件，因此传统的结构设计方法不仅工作量大，而且由于计算误差较大而导致结构余量设计不合理。有限元法可以在传统的设计方法中引入虚拟仿真，大大降低了试验成本，缩短了制造周期。

有限元法已广泛应用于汽车零部件的机械设计，如汽车车架的轻量化设计、车架和车身振动特性分析等，可以有效地解决复杂零部件整体变形和应力分布问题。此外，有限元法还为变速器壳体、悬架系统、制动系统、车轮等汽车零部件的强度和刚度分析设计提供了强有力的支持。采用有限元法对不同工况下的应力情况进行分析，有效地提高了设计效率，减小了计算误差。

4.2. 具体应用分析

有限元法作为一种有效的数值分析方法，已广泛应用于机械设计等领域。由于有限元法具有在试验阶段模拟车辆部件结构和分析冲击环境的优势，可以在设计阶段提供合理的优化方案，可以在保持结构稳定性的同时研究轻量化设计以及未来汽车开发使用的替代材料（例如塑料、铝等）。因此，有限元法在制造设计中占据主导地位。目前比较流行的有限元软件有 NASTRAN、ADINA、ANSYS、ABAQUS、MARC、COSMOS 等。

除了整体车身及轻量化，有限元方法还可以进行重要零部件的研究，如汽车车轮、钢板弹簧、车身构件等零部件。

参考文献

[1]祁先平.基于 MATLAB 的平面桁架有限元法[J].兰州工业学院学报,2016,23(03):31-35.

[2]赵经文，王宏钰．结构有限元分析 [M]．哈尔滨:哈尔滨工业大学出版社，1998：1-2．

[3]徐斌，高跃飞，余龙．MATLAB 有限元结构动力分析与工程应用 [M]．北京:清华大学出版社，2009．

[4]徐荣桥．结构分析的有限元法与 MATLA 程序设计 [M]．北京:人民交通出版社，2006．

[5]陈怀琛．MATLAB 及其在理工课程中的应用指南 [M]．西安:西安电子科技大学出版社，2000．

[6]高俊斌．MATLAB 语言与程序设计 [M]．武汉:华中理工大学出版社，1998．

[7]刘晶波，杜修力．结构动力学[M]．北京:机械工业出版社，2005．

[8]马志贵，刘世忠．基于 MATLAB 的结构动力特性分析 [J]．兰州工业学院学报，2014(1):18-22．

[9]鸿庆，任侠．结构有限元分析[M]．北京:中国铁道出版社，2000．

[10]刘卫国． MATLAB 程序设计教程 [M]．北京： 中国水利水电出版社，2005.

[11]PRIES H, Wille H C. SOME EXAMPLES OF MODERN VEHICLE DESIGN ANALYSIS BY THE FINITE ELEMENT METHOD[J]. International Journal of Vehicle Design, 1984, 5(HS-036 655).

[12]Gu S. Application of finite element method in mechanical design of automotive parts[C].IOP Conference Series. Materials Science and Engineering. IOP Publishing, 2017, 231(1).

[13]Melosh R J. Finite element analysis of automobile structures[J]. SAE Transactions, 1974: 1341-1355.