

实用有限元分析期末大作业

平面应力问题的MATLAB实现

姓 名 李炎朴

学 科 专 业 机械工程

学 号 S230200205

研 究 方 向 超快激光微纳制造

学 院 机械与运载工程学院

上 课 教 师 王琥

目录

[1、 平面应力问题有限元分析的基本理论 3](#_Toc122708532)

[1.1 单元的几何和节点描述 3](#_Toc122708533)

[1.2 单元位移场的表达 3](#_Toc122708534)

[1.3 单元应变场的表达 4](#_Toc122708535)

[1.4 单元应力场的表达 4](#_Toc122708536)

[1.5 单元刚度方程表达 5](#_Toc122708537)

[2、平面应力问题有限元分析的过程 5](#_Toc122708538)

[2.1 有限元分析软件简介 5](#_Toc122708539)

[2.2 前处理 6](#_Toc122708540)

[2.3 提交计算 8](#_Toc122708541)

[2.4 后处理 13](#_Toc122708542)

[2.5 结果分析 15](#_Toc122708543)

[3、有限元在模具制造行业内的应用分析 16](#_Toc122708544)

[参考文献 18](#_Toc122708545)

# 平面应力问题有限元分析的基本理论

本项目采用三节点三角形单元进行平面应力问题有限元分析。平面问题三节点三角形单元的几何特征简单，是平面问题有限元分析中最基础的单元。

## 单元的几何和节点描述

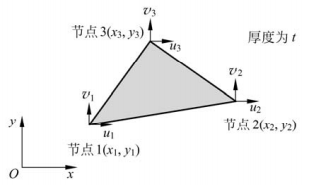


图1三角形三节点单元

如图所示，节点编号分别为1,2,3，用xy表示其在全局坐标系下的位置，uv表示各个节点沿xy方向的位移。每个三节点三角形单元拥有6个节点位移自由度，我们可以做如下定义，将节点上的位移组成列阵，将节点上的力组成列阵，则





## 单元位移场的表达

考虑到简单性，完备性，连续性以及待定系数的唯一确定性原则，取位移模式如下所示。



将节点坐标带入位移模式，求解并重写位移函数，以节点位移的形式进行表示，有





写成矩阵形式，有



其中N为形函数矩阵







上式中下标按1,2,3顺序轮换。

## 单元应变场的表达

由弹性力学平面问题几何方程可得



其中，几何方程的算子矩阵如下。



将式（1.6）带入式（1.10）得



. 其中几何矩阵为



## 单元应力场的表达



其中，平面应力问题的弹性系数矩阵为



## 单元刚度方程表达

单元刚度方程如下



其中，单元刚度矩阵由势能原理推出，即



# 2、平面应力问题有限元分析的过程

## 2.1 有限元分析软件简介

本项目前处理方面采用ANSYS软件进行划分网格，添加约束和载荷，计算过程则分别利用MATLAB自己编写的代码和ANSYS软件进行计算并对比。

ANSYS有限元软件包是一个多用途的有限元法计算机设计程序，可以用来求解结构、流体、电力、电磁场及碰撞等问题。因此它可应用于以下工业领域： 航空航天、汽车工业、生物医学、桥梁、建筑、电子产品、重型机械、微机电系统、运动器械等。软件主要包括三个部分：前处理模块，分析计算模块和后处理模块。前处理模块提供了一个强大的实体建模及网格划分工具，用户可以方便地构造有限元模型；分析计算模块包括结构分析（可进行线性分析、非线性分析和高度非线性分析）、流体动力学分析、电磁场分析、声场分析、压电分析以及多物理场的耦合分析，可模拟多种物理介质的相互作用，具有灵敏度分析及优化分析能力；后处理模块可将计算结果以彩色等值线显示、梯度显示、矢量显示、粒子流迹显示、立体切片显示、透明及半透明显示（可看到结构内部）等图形方式显示出来，也可将计算结果以图表、曲线形式显示或输出。

MATLAB主要面对科学计算、可视化以及交互式程序设计的高科技计算环境。它将数值分析、矩阵计算、科学数据可视化以及非线性动态系统的建模和仿真等诸多强大功能集成在一个易于使用的视窗环境中，为科学研究、工程设计以及必须进行有效数值计算的众多科学领域提供了一种全面的解决方案，并在很大程度上摆脱了传统非交互式程序设计语言（如C、Fortran）的编辑模式。MATLAB的基本数据单位是矩阵，它的指令表达式与数学、工程中常用的形式十分相似，故用MATLAB来解算问题要比用C，FORTRAN等语言完成相同的事情简捷得多，并且MATLAB也吸收了像Maple等软件的优点，使MATLAB成为一个强大的数学软件。在新的版本中也加入了对C，FORTRAN，C++，JAVA的支持。

## 2.2 前处理

考虑到问题属于平面应力问题，因此首先在SolidWorks软件中建立对应的截面片体，导入到ANSYS Workbench中，利用DesignModeler创建一个点载荷以备使用。分析类型切换成2D。

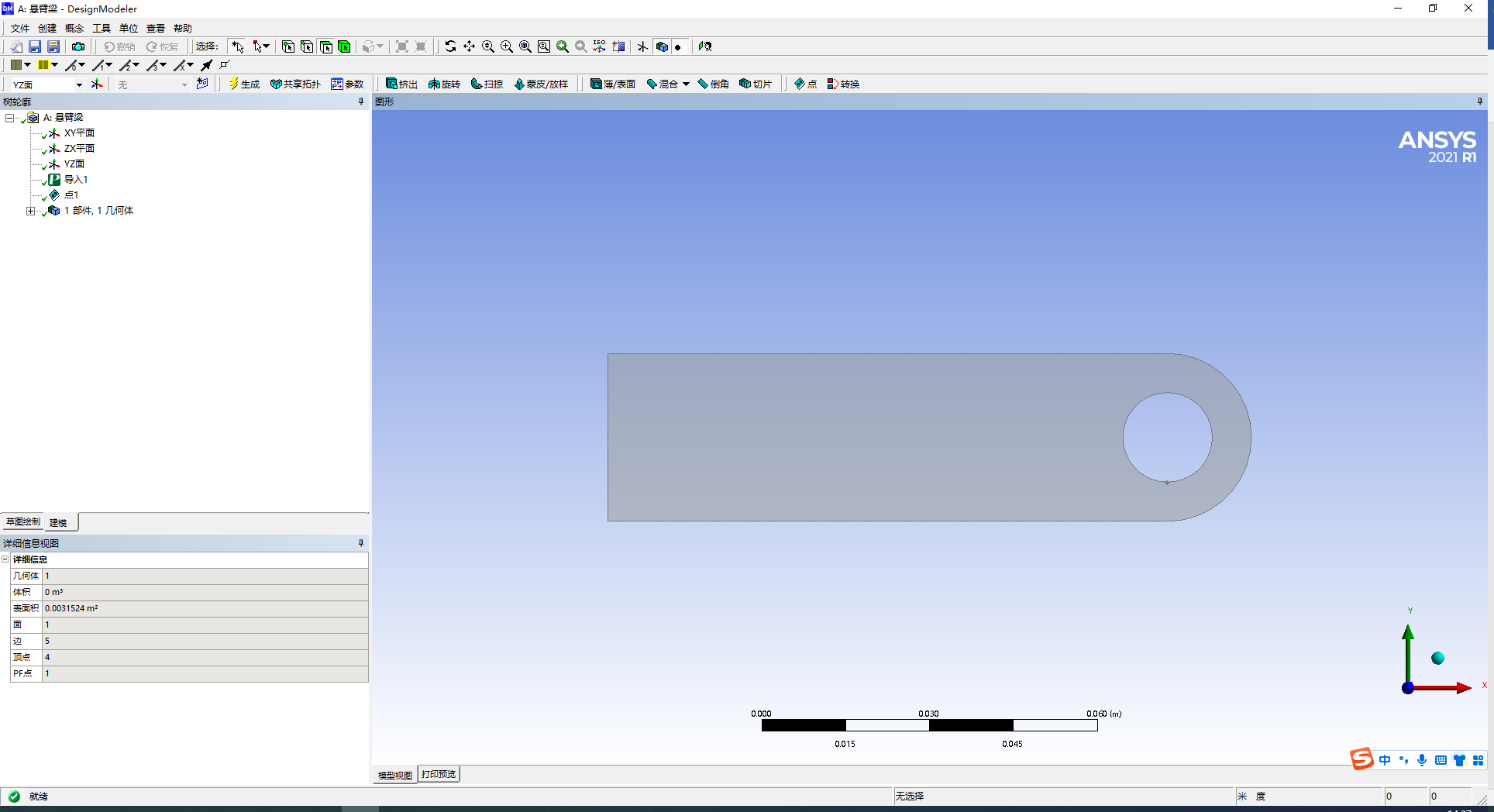


图2设置点载荷

打开模型，为其赋予模型厚度为1mm，2D行为为平面应力。赋予材料为结构钢，网格设置单元的阶为线型的，单元尺寸为4mm，在网格中插入一个方法，选择三角形，单元的阶为线型的。其目的是与接下来项目中MATLAB程序的单元形状和阶次进行统一。全部设定好之后划分网格。

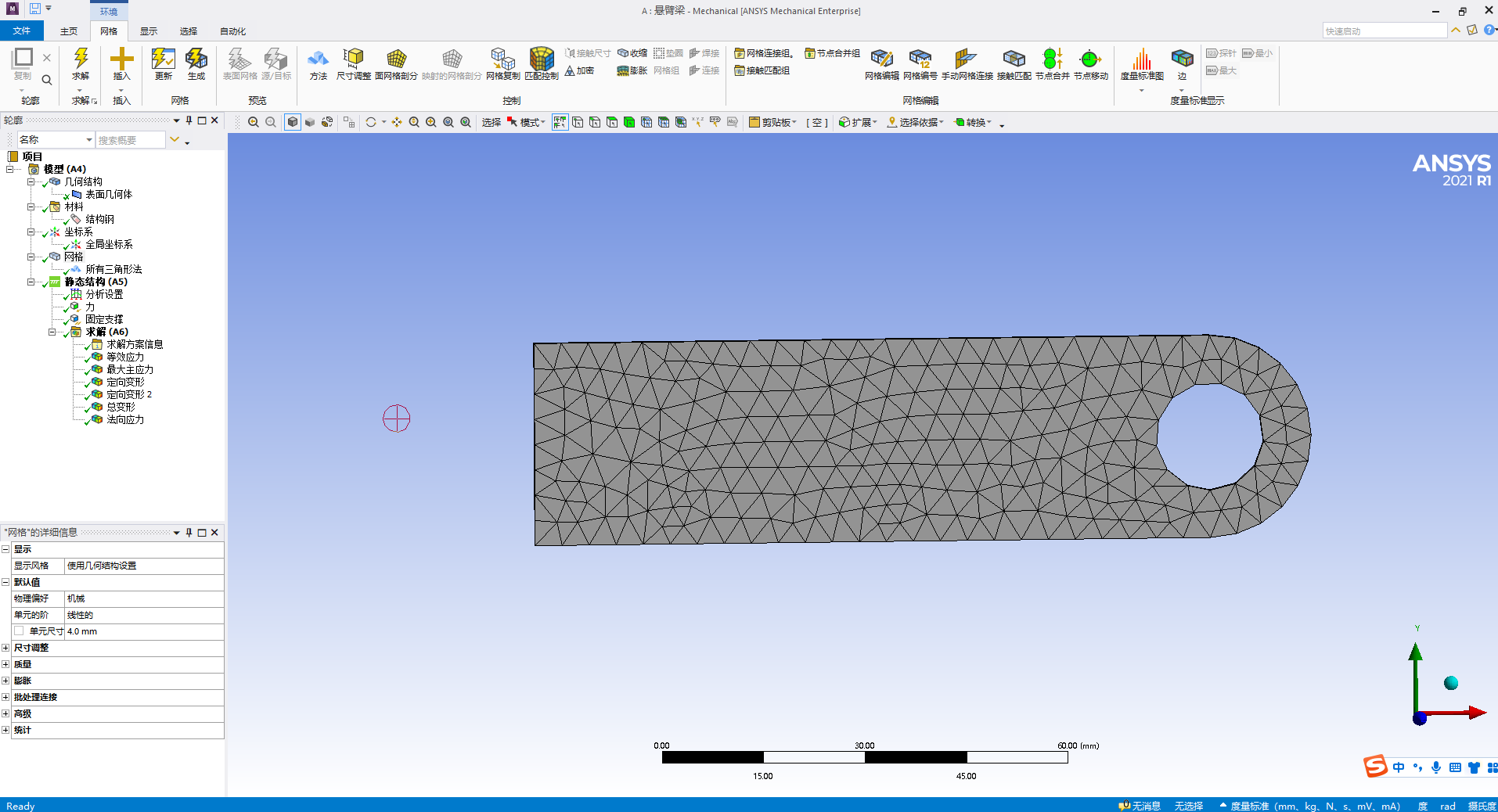


图3 划分网格

在静态结构选项里加入力，选择几何结构为之前绘制的点载荷，大小为100N，方向为y轴负方向。

在静态结构中加入固定支撑，选择几何结构为左边的竖直边，将其固定。

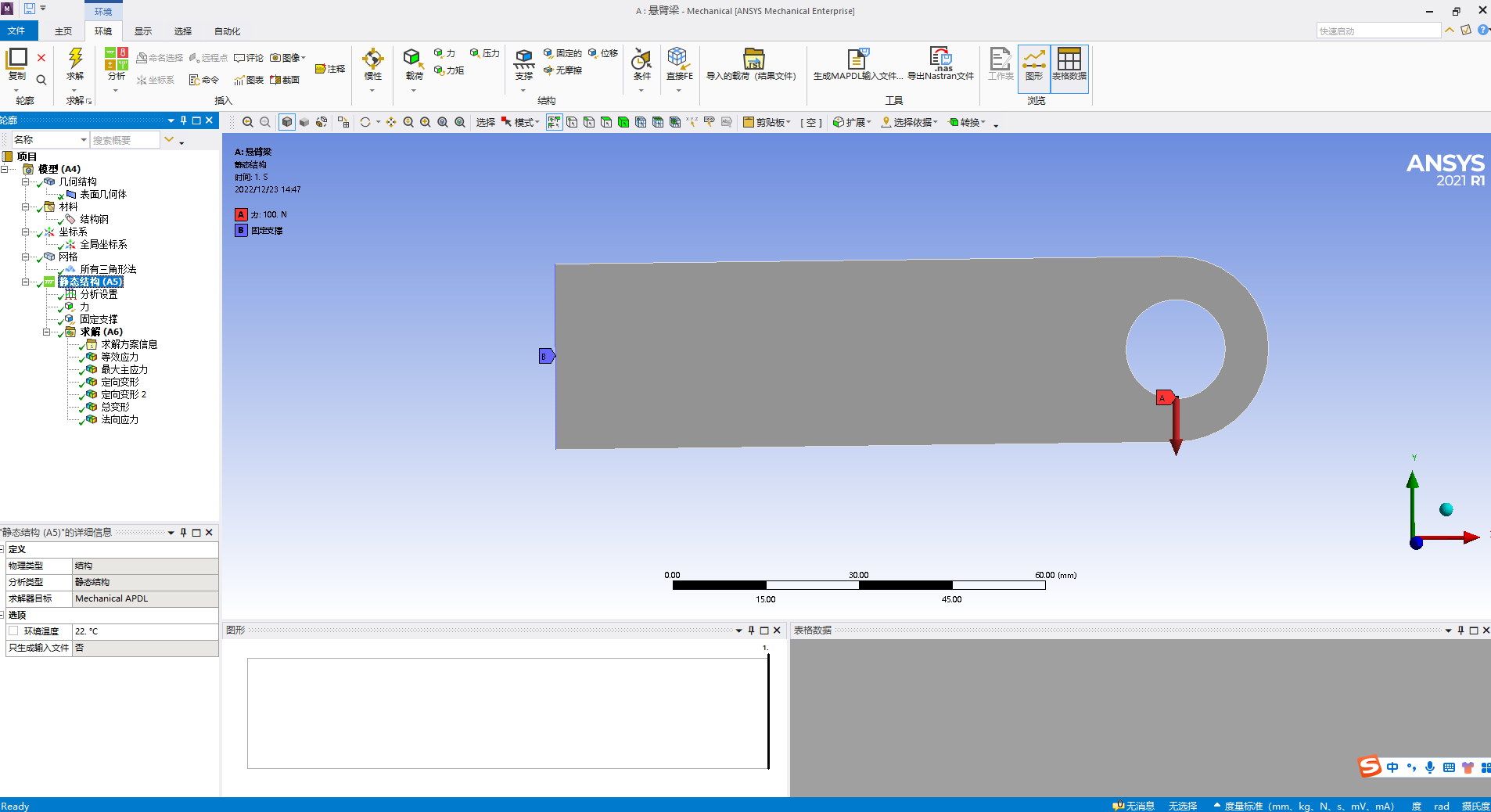


图4 添加载荷和约束

提交计算后，导出未平均的等效应力，因为等效应力是以单元来计算的，因此未平均的等效应力信息中包含了单元的编号，节点的位置以及单元和节点的关系。这部分数据将会用在MATLAB程序中。

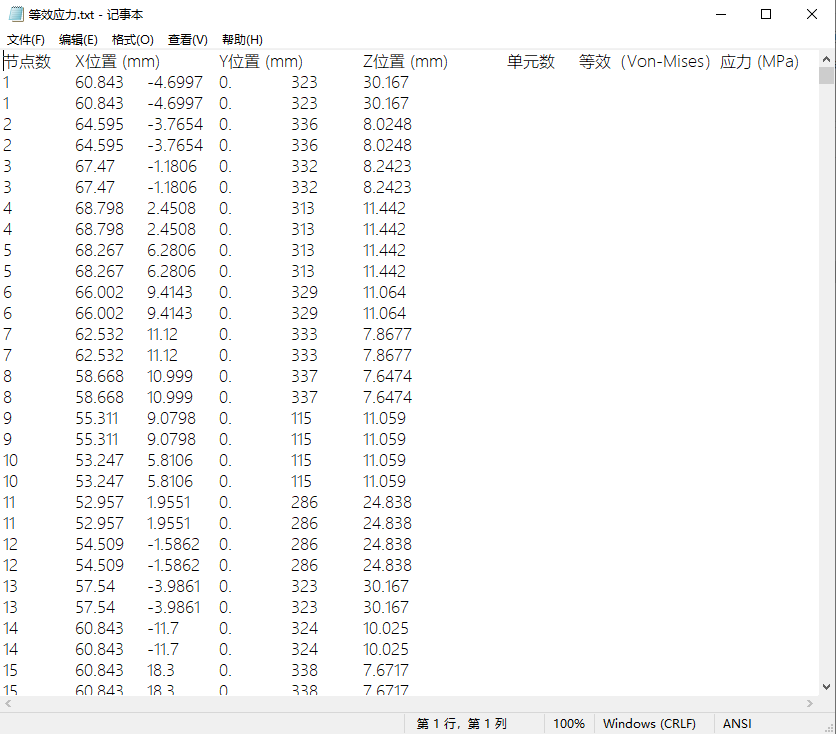


图5 提取节点和单元数据

利用wps表格进行数据处理，将所需要的节点数据和单元数据处理成以下形式。

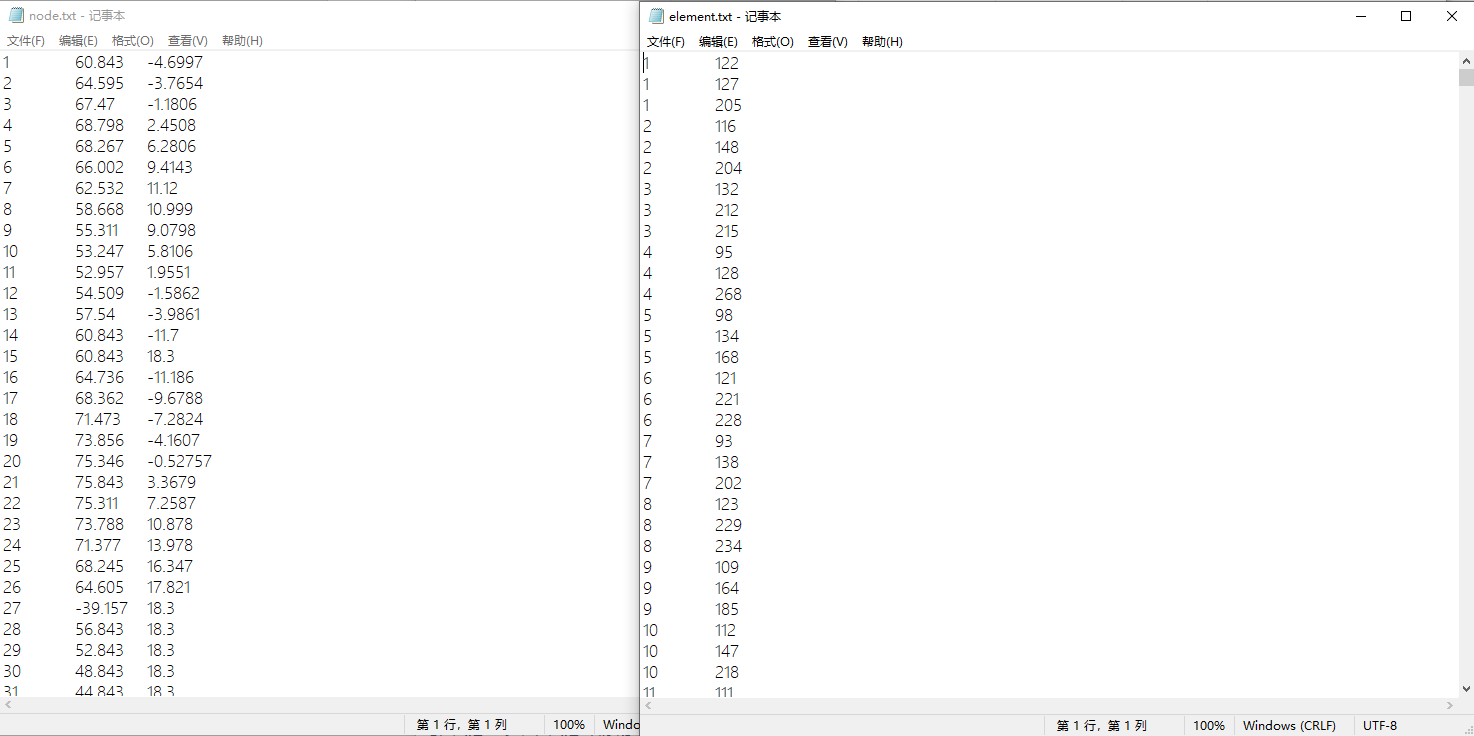


图6 编辑节点和单元文件

其中，node.txt文件储存了所有节点的编号以及对应的坐标信息，element.txt信息储存了每一个单元对应的节点编号，但这里节点编号的顺序并不是按照逆时针排列的，因此后期还需要经过一步处理，否则会缠上计算错误。

载荷和约束需要手动创建txt文件，根据ANSYS上对应的节点编号来编写文件，最终文件如下所示。

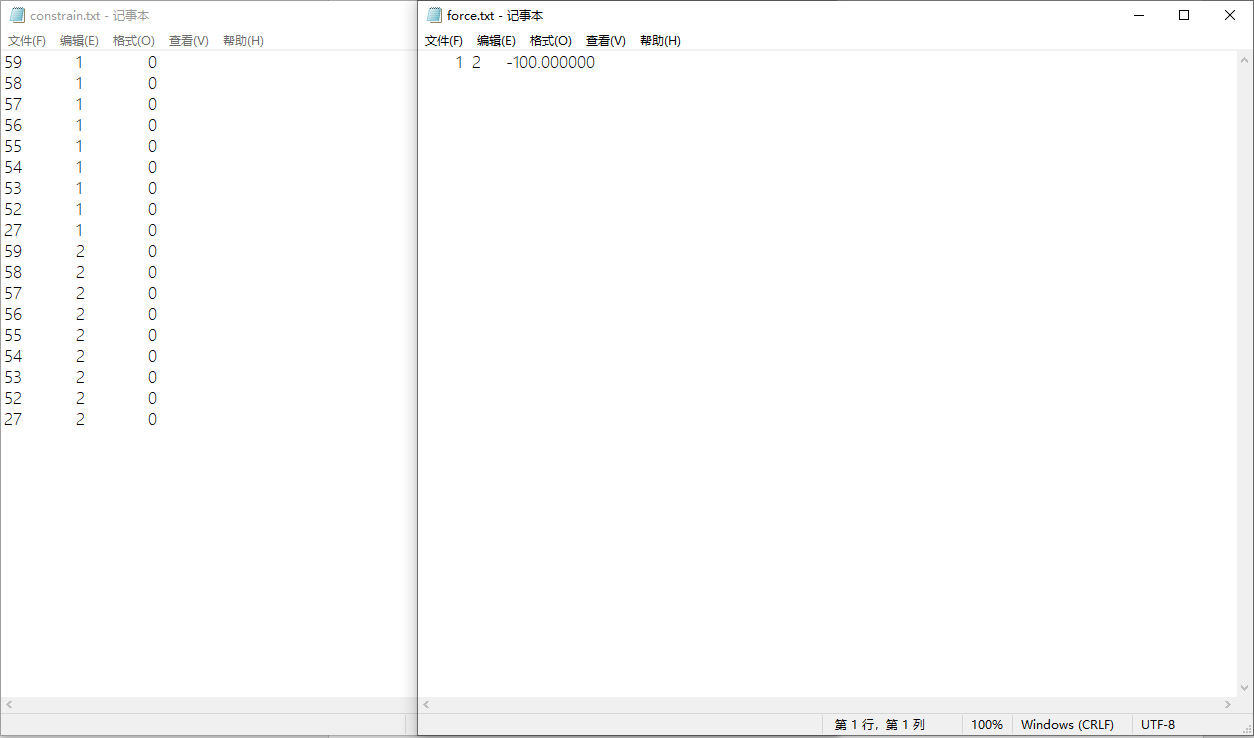


图7 编辑载荷和约束文件

Constrain.txt文件为约束文件，三列依次为节点编号，约束方向，约束位移，例如第一行就是59号节点在x方向位移为0mm。force.txt文件为载荷文件，三列依次为节点编号，力的方向，力的大小，本项目中为1号节点在y方向施加-100N的力。

做完上述处理即可提交计算。

## 2.3 提交计算

ANSYS在做完上述处理后即可直接提交计算得出结果。

MATLAB需要编写计算函数与相应代码。

本项目的MATLAB程序共有4个函数，分别是判断是否需要更换单元节点排序的nodesOrder函数，计算单元刚度矩阵的Stiffness函数，组装总体刚度矩阵的Assembly函数和计算单元应力分量的Stress函数。相关代码如下。

function needChange = nodesOrder(xi,yi,xj,yj,xm,ym)

%该函数进行单元节点顺序的判定

%输入单元节点坐标

%计算单元面积，如果为负则需要调换顺序

A = (xi\*(yj-ym) + xj\*(ym-yi) + xm\*(yi-yj))/2;

if A<0

needChange = 1;

else

needChange = 0;

end

end

function k=Stiffness(E,NU,t,xi,yi,xj,yj,xm,ym)

%该函数计算单元的刚度矩阵

%输入弹性模量E，泊松比NU，厚度t

%输入三个节点i、j、m的坐标xi,yi,xj,yj,xm,ym

%输出单元刚度矩阵k(6X6)

A = (xi\*(yj-ym) + xj\*(ym-yi) + xm\*(yi-yj))/2;

betai = yj-ym;

betaj = ym-yi;

betam = yi-yj;

gammai = xm-xj;

gammaj = xi-xm;

gammam = xj-xi;

B = [betai 0 betaj 0 betam 0 ;

0 gammai 0 gammaj 0 gammam ;

gammai betai gammaj betaj gammam betam]/(2\*A);

D = (E/(1-NU\*NU))\*[1 NU 0 ; NU 1 0 ; 0 0 (1-NU)/2];

k= t\*A\*B'\*D\*B;

function z = Assembly(KK,k,i,j,m)

%该函数进行单元刚度矩阵的组装

%输入单元刚度矩阵k

%输入单元的节点编号I、j、m

%输出整体刚度矩阵KK

DOF(1)=2\*i-1;

DOF(2)=2\*i;

DOF(3)=2\*j-1;

DOF(4)=2\*j;

DOF(5)=2\*m-1;

DOF(6)=2\*m;

for n1=1:6

for n2=1:6

KK(DOF(n1),DOF(n2))= KK(DOF(n1),DOF(n2))+k(n1,n2);

end

end

z=KK;

function stress=Stress(E,NU,xi,yi,xj,yj,xm,ym,u)

%该函数计算单元的应力

%输入弹性模量E，泊松比NU，厚度t

%输入三个节点i、j、m的坐标xi,yi,xj,yj,xm,ym

%输出单元的应力stress(3X1)，由于它为常应力单元，则单元的应力分量为Sx,Sy,Sz

A = (xi\*(yj-ym) + xj\*(ym-yi) + xm\*(yi-yj))/2;

betai = yj-ym;

betaj = ym-yi;

betam = yi-yj;

gammai = xm-xj;

gammaj = xi-xm;

gammam = xj-xi;

B = [betai 0 betaj 0 betam 0 ;

0 gammai 0 gammaj 0 gammam ;

gammai betai gammaj betaj gammam betam]/(2\*A);

D = (E/(1-NU\*NU))\*[1 NU 0 ; NU 1 0 ; 0 0 (1-NU)/2];

stress = D\*B\*u;

主程序则对相关数据进行处理和调用。相关代码如下。

clear;clc

tic

%读入相应信息文件，给出指定工作目录

load .\node.txt

load .\element.txt

load .\constrain.txt

load .\force.txt

element1 = zeros(size(element,1)/3,3);

%确定节点、单元个数

%编号从1开始

for i=0:size(element,1)/3-1

element1(i+1,1) = element(i\*3+1,2);

element1(i+1,2) = element(i\*3+2,2);

element1(i+1,3) = element(i\*3+3,2);

end

elem = element1;

[nnode,nn]=size(node);%提取节点数存放在nnode里

[nelem,ee]=size(elem);%提取单元数存放在nelem里

[nforce,ff]=size(force);%提取力个数存放在nforce里

[nconstrain,cc]=size(constrain);%提取约束个数存放在nconstrain里面

for i=1:nelem

if nodesOrder(node(elem(i,1),2),node(elem(i,1),3) ,node(elem(i,2),2),node(elem(i,2),3) ,node(elem(i,3),2),node(elem(i,3),3))==1;%element负责提取对应i号单元的节点编号，node负责提取该编号节点的对应坐标

elem(i,1)=element1(i,2);

elem(i,2)=element1(i,1);

end

end

%预先设定总体刚度矩阵、节点力向量、节点约束向量

%针对二维问题

KK=zeros(2\*nnode);%总刚矩阵规模为节点数×2

FF=zeros(2\*nnode,1);%节点力向量

UU=zeros(2\*nnode,1);%节点位移向量

StrainElem=zeros(nelem,3);

StressElem=zeros(nelem,3);%单元应力向量，两个正应力和一个切应力

StressNode=zeros(nnode,3);%节点应力向量

k=zeros(6,6); % 单元刚度矩阵

%给出相应材料及计算参数

E=2e05;%弹性模量

NU=0.3;%泊松比

t=1;%厚度

%单元循环形成总体刚度矩阵

for i=1:nelem

k=Stiffness(E,NU,t,node(elem(i,1),2),node(elem(i,1),3) ,node(elem(i,2),2),node(elem(i,2),3) ,node(elem(i,3),2),node(elem(i,3),3));%element负责提取对应i号单元的节点编号，node负责提取该编号节点的对应坐标

KK =Assembly(KK,k,elem(i,1),elem(i,2),elem(i,3));%element控制输入的第i个节点编号，k为刚刚求出的单元刚度矩阵

end

%载荷的处理

for i=1:nforce%多少个力循环多少次

m=force(i,1);%读力所在的节点

n=force(i,2);%读力的方向

FF(2\*(m-1)+n)= force(i,3);%读力的大小，存入节点力向量中

end

%边界条件的处理

for i=1:nconstrain

m=constrain(i,1);%读取边界条件所在节点

n=constrain(i,2);%读取边界条件的方向（x还是y方向固定）

UU(2\*(m-1)+n)=constrain(i,3);

KK(2\*(m-1)+n,:)=0;%边界条件所在行列，除其本身以外，全部置零

KK(:,2\*(m-1)+n)=0;

KK(2\*(m-1)+n,2\*(m-1)+n)=1;

FF(2\*(m-1)+n)=0;%边界条件所在位置的节点力向量置零

end

% 求解节点位移

UU=KK\FF;

%求解单元应力

for i=1:nelem

% 求该单元的节点位移

l=elem(i,1);%第i个单元的第一个节点

m=elem(i,2);

n=elem(i,3);

u=[UU(2\*l-1),UU(2\*l),UU(2\*m-1),UU(2\*m),UU(2\*n-1),UU(2\*n)]';%提取出第i个单元各个节点的位移

% 求单元应力

stresstemp=Stress(E,NU,node(elem(i,1),2),node(elem(i,1),3),node(elem(i,2),2),node(elem(i,2),3),node(elem(i,3),2),node(elem(i,3),3),u);

StressElem(i,1)=stresstemp(1,1);%将函数计算完成后的单元位移放在整个矩阵里面，x正应力

StressElem(i,2)=stresstemp(2,1);%y正应力

StressElem(i,3)=stresstemp(3,1);%xy切应力

StressElem(i,4)=sqrt((stresstemp(1,1)+stresstemp(2,1))^2-3\*(stresstemp(1,1)\*stresstemp(2,1)-stresstemp(3,1)^2));%mises等效应力

end

% 求节点应力－计算节点周边所有单元应力的平均值

StressNode=zeros(nnode,3);

for i=1:nnode

numElem=0;

for j=1:nelem

if (elem(j,1)==i)%第j个单元的第一个节点是否是i节点，如果是，则将该单元的两个正应力一个切应力加入到节点单元的应力中去

StressNode(i,1)=StressNode(i,1)+StressElem(j,1);

StressNode(i,2)=StressNode(i,2)+StressElem(j,2);

StressNode(i,3)=StressNode(i,3)+StressElem(j,3);

numElem=numElem+1;

elseif (elem(j,2)==i)

StressNode(i,1)=StressNode(i,1)+StressElem(j,1);

StressNode(i,2)=StressNode(i,2)+StressElem(j,2);

StressNode(i,3)=StressNode(i,3)+StressElem(j,3);

numElem=numElem+1;

elseif (elem(j,3)==i)

StressNode(i,1)=StressNode(i,1)+StressElem(j,1);

StressNode(i,2)=StressNode(i,2)+StressElem(j,2);

StressNode(i,3)=StressNode(i,3)+StressElem(j,3);

numElem=numElem+1;

end

end

StressNode(i,1)=StressNode(i,1)/numElem;%平均

StressNode(i,2)=StressNode(i,2)/numElem;

StressNode(i,3)=StressNode(i,3)/numElem;

end

## 2.4 后处理

ANSYS有非常直观的图形界面来查看后处理的结果，以下是ANSYS得出的单元等效应力和节点总变形的可视化结果。

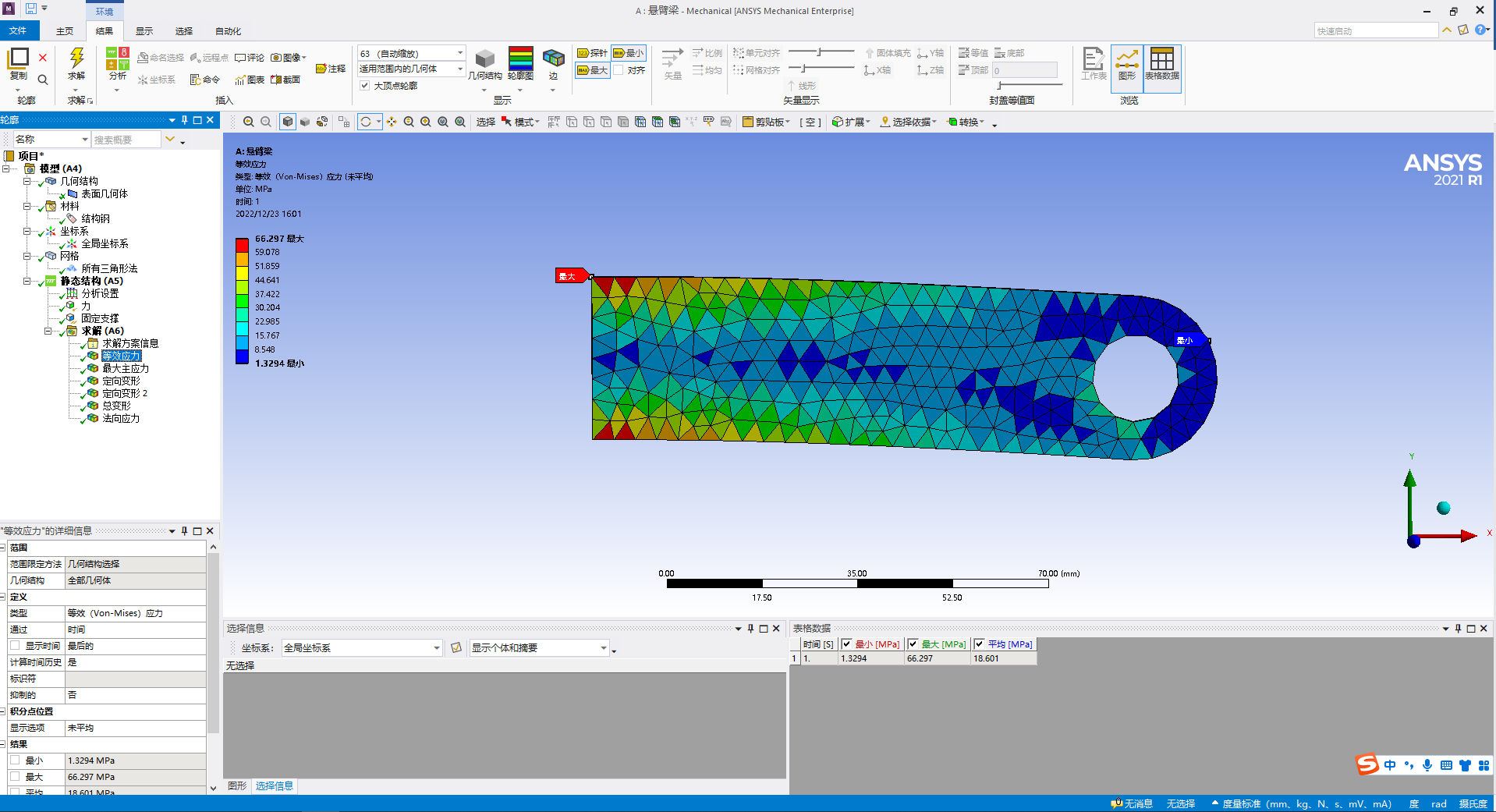


图8 平均应力云图

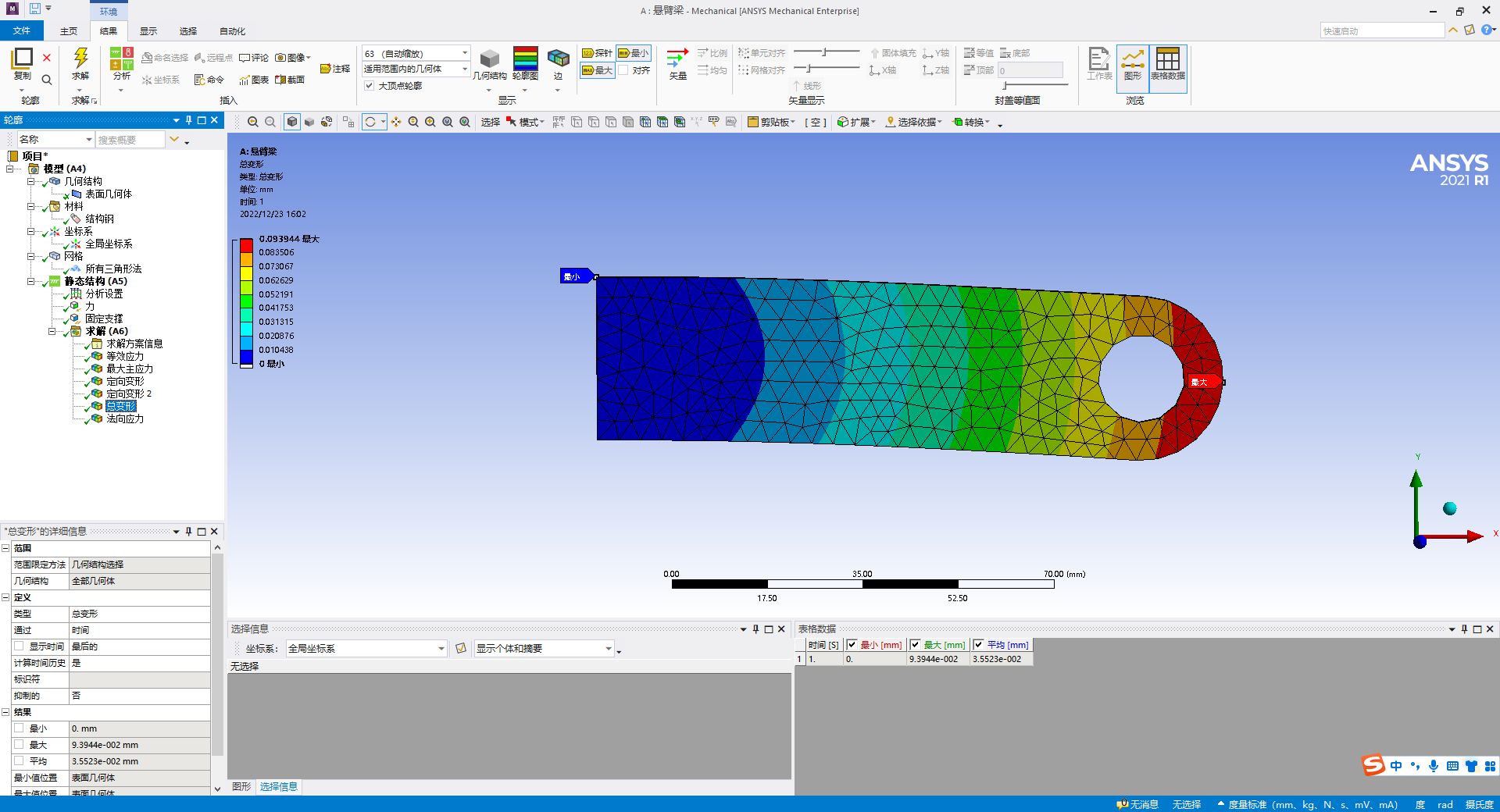


图9 节点位移云图

MATLAB数据都存储在程序保存的矩阵中，为了方便查看，将其导出为txt格式，本项目共导出了三个常用的结果，分别是节点位移，节点应力和单元应力，相关代码如下。

% 输出节点位移值

fid=fopen('.\node\_displace.dat','w');

fprintf(fid,'\n%s\n','-------------------- NODE DISPLACEMENT ---------------------------');

fprintf(fid,'\n%s\n',' Node X-coordinate Y-coordinate X-disp Y-disp disp ');

for i=1:nnode

fprintf(fid,'%10d%18.8f%18.8f%18.8f%18.8f%18.8f\n',node(i,1),node(i,2),node(i,3),UU(2\*(i-1)+1),UU(2\*(i-1)+2),(UU(2\*(i-1)+1)^2+UU(2\*(i-1)+2)^2)^0.5);

end

% 关闭文件

fclose(fid);

% 输出单元应力值

fid=fopen('.\elem\_stress.dat','w');

fprintf(fid,'\n%s\n','-------------------- ELEMENT STRESS ----------------------------');

fprintf(fid,'\n%s\n',' Node X-stress Y-stress Shear stress von-Mises ');

for i=1:nelem

fprintf(fid,'%10d%18.8f%18.8f%18.8f%18.8f\n',i,StressElem(i,1),StressElem(i,2),StressElem(i,3),StressElem(i,4));

end

% 关闭文件

fclose(fid);

% 输出节点应力值

fid=fopen('.\node\_stress.dat','w');

fprintf(fid,'\n%s\n','-------------------- NODE STRESS --------------------------');

fprintf(fid,'\n%s\n',' Node X-stress Y-stress Shear stress ');

for i=1:nnode

fprintf(fid,'%10d%18.8f%18.8f%18.8f\n',node(i,1),StressNode(i,1),StressNode(i,2),StressNode(i,3));

end

% 关闭文件

fclose(fid);

toc

## 2.5 结果分析

从计算速度而言，4mm单元在ANSYS上计算耗费了2s，而在MATLAB程序上只花费了0.38秒，但增大网格密度后进行计算，2mm单元在ANSYS上花费了6秒，而MATLAB程序花费了14秒，1mm单元在ANSYS上花费了12秒，但在MATLAB程序中足足运行了729秒，究其原因可能是，ANSYS Workbench在每次求解时都需要重新提交数学模型，虽然本项目并没有分析十分复杂的数据，但是每次提交都会占用一定量的时间，而MATLAB程序在前处理过程中就已经将数据基本处理好了，读取起来十分方便，因此在计算量不大的情况下，MATLAB程序会略胜一筹。而当网格加密后，总刚矩阵的规模变得很大，ANSYS自身经过多年的迭代，包括其求解器和分布式运算等技术发展的十分成熟，因此仍可以较快得到结果，而自己编写的MATLAB程序由于编程技术原因，反复使用for循环，矩阵运算等等，且没有采用并行计算技术，因此计算速度大幅下降。甚至在0.5mm单元出现了爆内存，计算不出来的情况。况且本项目中MATLAB的前处理网格还是利用ANSYS的网格文件导出的，因此二者的运行时间实质上并没有做将前处理的时间计算入内，否则MATLAB的运行时间智慧更长。

分析计算结果，导出ANSYS的单元应力结果，节点应力结果和节点位移结果，与MATLAB导出的文件放入同一表格中作对比。部分结果截图如下。

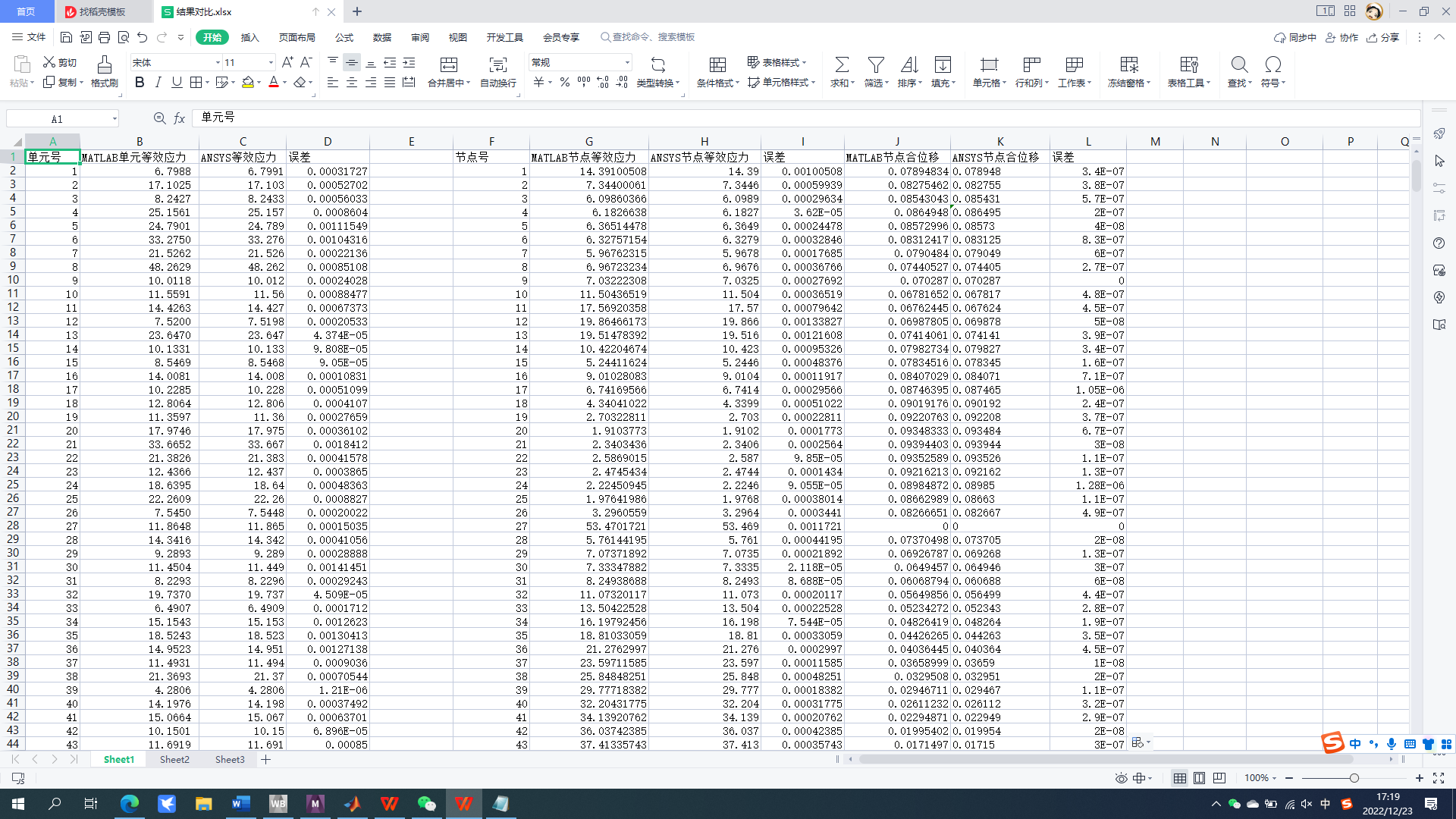


图10 结果对比

其误差大多在0.0001量级上，且基本在舍入后结果相同。进一步比对，各个结果的最大最小值及其出现的位置都能互相对应上。因此，MATLAB计算结果的精度是可以接受的。

# 3、有限元在模具制造行业内的应用分析

冲压成形是现代工业中非常常见的加工方法，冲压制品在汽车，航空，家用电器，日常用品等领域有着非常广泛的应用。但冲压模具的成本高，加工周期长，而且需要反复试模才能投入生产使用。此外，模具的工作环境一般比较恶劣，因此想要增加模具的寿命，传统方法是简单加厚，或者使用组合模具，但一来会带来成本的增加，二来有些时候反而会造成寿命进一步减少，因此，越来越多的企业开始采用有限元方法来对模具进行分析，通过有限元分析，可以及早发现模具问题，节约材料和成本，提高精度和可靠性，避免多次修模试模的过程，缩短模具调试周期，延长模具使用寿命。

针对冲压模具本身的有限元分析是一方面，另一方面是针对冲压制品的有限元分析。这在汽车领域应用尤其广泛。比如对于复杂制件，例如汽车覆盖件，可以事先利用有限元仿真来模拟冲压和回弹过程，根据得到的结果来改进模具形状，对回弹进行补偿。对于一些拉深制件，同样可以采用有限元方法进行仿真，反复优化压边力，拉延筋等，起到优化起皱裂纹等缺陷的作用。

注塑制品的生产也越来越依赖于有限元分析。注塑成型应该是最广泛使用，但是最不完全了解的制造技术，因为这个行业涉及到聚合物的流体行为，传热学等等，因此极端依赖经验。而如今随着研究的深入，人们能建立更准确的数学模型来描述注塑过程，这也就为有限元分析的发展提供了可能。

如今的注塑生产已经和有限元分析高度结合。汽车的塑料内饰或是塑料零件在很多时候是较难完成填充的，因此必须要仔细设置塑料流体的填充速度和压力。反复在注塑机上进行试验显然是不合理的，因此通过有限元可以节约成本提高效率。如今的注塑零件也越来越复杂，同样涉及到填充和冷却的问题，利用有限元分析就可以反复优化模具结构，加快迭代速度，也为复杂模具的设计提供了技术论证和数据支持，尽量减少经验和试模带来的不确定性。

显然，有限元方法已经广泛应用于模具制造行业中，显著加快了模具设计制造的速度，节约了模具的生产成本，也让生产出的产品质量更好，结构更复杂。

# 参考文献

[1]崔济东.有限单元法—编程与软件应用[M]. 北京:中国建筑工业出版社, 2019.

[2]曾攀.有限元基础教程[M]. 北京:清华大学出版社, 2008. -.

[3]曾森.有限单元法基础及MATLAB编程[M]. 北京:高等教育出版社, 2016.

[4]王焕定.有限单元法基础及MATLAB编程[M]. 北京:高等教育出版社, 2012.

[5]黄荣学.有限元法在冲压模具分析中的应用及展望[J].锻压技术,:,2005.(2):69-72.

[6]方刚.基于有限元分析的汽车覆盖件模具设计[J].模具工业,2002,(6):6-10.

[7]尹红玲,数值模拟技术在新产品模具设计中的应用[J].模具技术,2003,(5):5-8.