一、 弹塑性有限元方法简介

弹塑性有限元方法是将有限元方法应用于弹塑性材料的力学计算方法。有限元方法是一种求解微分方程的数值计算方法。基本思想是将求解域分解成有限个简单单元，在每个单元内部用节点处的函数值近似内插来表示未知场函数，并基于变分法，用节点处的未知函数值为未知量，建立方程组，使误差场函数达到最小值，并取得稳定解，整体上是一个“分解—近似—集总”的过程。弹塑性问题是材料非线性问题的一种，材料的变形立即发生，不随时间再次发生变化。弹塑性材料是是指材料的应力和应变关系不是线性的，弹塑性材料与弹性材料的不同之处在于材料本构关系的非线性。非线性问题可以划分为材料非线性、几何非线性和边界非线性问题。材料非线性问题可以依据是否受时间影响分为：弹塑性问题和粘弹性问题。而弹塑性问题是非线性问题的基础问题。求解弹塑性问题的方法在于将本构关系线性化，这样便可以用线性问题的表达格式进行分析求解。

二、 弹塑性有限元实验理论分析

1. 变分原理和里兹法

力学当中的变分原理是指将求位移函数的问题转化成求泛函极值的问题。基于等效积分原理、格林公式、变分运算法则和线性自伴随算子的性质可以实现这一转变。也可以说加权余量法是变分原理的基础。

里兹法（Rayleigh-Ritz）是基于变分原理的微分方程近似求解方法。弹性力学当中的变分原理有：最小位能原理、最小余能原理、Hellinger-Reissner 变分原理和“胡海昌—鹫津久一郎（Hu—Washizu）”变分原理等。

系统的总势能：

$$\Pi(\mathrm{u})=\int_{\Omega}[\frac{1}{2}\varepsilon^{\mathrm{T}}\sigma\text{-}\mathrm{u}^{\mathrm{T}}\bar{f}]d\Omega-\int_{\Gamma}\mathrm{u}^{\mathrm{T}}td\Gamma$$

真实的位移使总位能取最小值，也即：$\delta\Pi(\mathrm{u})=0$

又由几何关系和物理关系，可以将应变和应力都用位移来表示。

$$\varepsilon(\mathrm{u})=L\,\mathrm{u}\qquad\qquad\sigma(\mathrm{u})=D\varepsilon(\mathrm{u})=DL\mathrm{u}$$

未知函数即为位移场函数。取位移的近似解：$\mathrm{u}=\tilde{\mathrm{u}}=\sum_{i=1}^{n}N_i a_i$ ，其中 a_i 是未知量，N_i 是近似插值函数，也称为形函数。

则能量泛函可以表示为：

$$\Pi(\mathrm{u})=\frac{1}{2}a^{\mathrm{T}}\int_{V}(LN)^{\mathrm{T}}DLNdVa-a^{\mathrm{T}}\int_{V}N^{\mathrm{T}}fdV-a^{\mathrm{T}}\int_{S_\sigma}N^{\mathrm{T}}\bar{T}dS$$

其变分为：

$$\delta\Pi=\frac{\partial\Pi}{\partial \mathrm{a}_1}\delta a_1+\frac{\partial\Pi}{\partial \mathrm{a}_2}\delta a_2+\ldots\ldots+\frac{\partial\Pi}{\partial \mathrm{a}_n}\delta a_n=0$$

因为δa_n是任意的，所以：

$$\frac{\partial\Pi}{\partial \mathrm{a}_1}=0\qquad\frac{\partial\Pi}{\partial \mathrm{a}_2}=0\qquad\ldots\ldots\qquad\frac{\partial\Pi}{\partial \mathrm{a}_n}=0$$

也即得到关于节点函数值a_n 的方程组。简记该方程组为：$Ka=F$ 其中K称为刚度矩阵。

其中$K=\int_{V}(LN)^{\mathrm{T}}DLNdV$ $a=\begin{bmatrix}u_1 & u_2 & \ldots & u_n\end{bmatrix}^{\mathrm{T}}$ $F=\int_{V}N^{\mathrm{T}}fdV+\int_{S_\sigma}N^{\mathrm{T}}\bar{T}dS$

以上方法即为里兹法。实际上，有限元方法是在微分方程求解域离散化以后，在每一个微小单元上使用里兹法然后再集成。单元上的刚度矩阵称为单刚，整体刚度矩阵称为总刚。

2. 弹塑性单元分析

1) 单元位移模式和形函数

单元位移模式采用插值方法得到，既用结点的位移值通过某种插值方式得到位移场的分布。插值函数即为形函数，这种近似方法可以表示为：

$$\mathrm{u}(x)=\sum_{i=1}^{n}N_iu_i=Na^{\mathrm{e}}$$

其中形函数N_i 需要满足 0-1 特性，一个单元内各节点形函数的和为 1. 本次实验采用的形函数为一次矩形 4 节点单元，其形函数分别为：

$$N_1=\frac{1}{4}(1\text{-}\xi)(1\text{-}\eta)\qquad N_2=\frac{1}{4}(1\text{+}\xi)(1\text{-}\eta)$$
$$N_3=\frac{1}{4}(1\text{+}\xi)(1\text{+}\eta)\qquad N_4=\frac{1}{4}(1\text{-}\xi)(1\text{+}\eta)$$

2) 等参变换

利用结点坐标插值建立几何变换：

$$x=\sum_{i=1}^{4}N_i\,x_i \quad y=\sum_{i=1}^{4}N_i\,y_i$$

用矩阵形式表示为：

$$x=\begin{bmatrix}N_1 & N_2 & N_3 & N_4\end{bmatrix}\begin{bmatrix}x_1\\x_2\\x_3\\x_4\end{bmatrix} \quad y=\begin{bmatrix}N_1 & N_2 & N_3 & N_4\end{bmatrix}\begin{bmatrix}y_1\\y_2\\y_3\\y_4\end{bmatrix}$$

应变矩阵如下：

$$\varepsilon=\begin{pmatrix}\varepsilon_x\\\varepsilon_y\\\gamma_{xy}\end{pmatrix}=Lu=LNa^e=Ba^e=\begin{pmatrix}\frac{\partial N_1}{\partial x} & 0 & \frac{\partial N_2}{\partial x} & 0 & \frac{\partial N_3}{\partial x} & 0 & \frac{\partial N_4}{\partial x} & 0\\0 & \frac{\partial N_1}{\partial y} & 0 & \frac{\partial N_2}{\partial y} & 0 & \frac{\partial N_3}{\partial y} & 0 & \frac{\partial N_4}{\partial y}\\\frac{\partial N_1}{\partial y} & \frac{\partial N_1}{\partial x} & \frac{\partial N_2}{\partial y} & \frac{\partial N_2}{\partial x} & \frac{\partial N_3}{\partial y} & \frac{\partial N_3}{\partial x} & \frac{\partial N_4}{\partial y} & \frac{\partial N_4}{\partial x}\end{pmatrix}\begin{pmatrix}u_1\\v_1\\u_2\\v_2\\u_3\\v_3\\u_4\\v_4\end{pmatrix}$$

其中：

$$\begin{pmatrix}\frac{\partial N_i}{\partial x}\\\frac{\partial N_i}{\partial y}\end{pmatrix}=J^{-1}\begin{pmatrix}\frac{\partial N_i}{\partial \xi}\\\frac{\partial N_i}{\partial \eta}\end{pmatrix}$$

坐标变换的雅各比矩阵为：

$$J=\begin{pmatrix}\frac{\partial N_1}{\partial \xi} & \frac{\partial N_2}{\partial \xi} & \frac{\partial N_3}{\partial \xi} & \frac{\partial N_4}{\partial \xi}\\\frac{\partial N_1}{\partial \eta} & \frac{\partial N_2}{\partial \eta} & \frac{\partial N_3}{\partial \eta} & \frac{\partial N_4}{\partial \eta}\end{pmatrix}\begin{pmatrix}x_1 & y_1\\x_2 & y_2\\x_3 & y_3\\x_4 & y_4\end{pmatrix}$$

$$=\begin{pmatrix}-\frac{1}{4}(1-\eta) & \frac{1}{4}(1-\eta) & \frac{1}{4}(1+\eta) & -\frac{1}{4}(1+\eta)\\-\frac{1}{4}(1-\xi) & -\frac{1}{4}(1+\xi) & \frac{1}{4}(1+\xi) & \frac{1}{4}(1-\xi)\end{pmatrix}\begin{pmatrix}x_1 & y_1\\x_2 & y_2\\x_3 & y_3\\x_4 & y_4\end{pmatrix}$$

3) 弹塑性问题的增量方程

采用增量理论的弹塑性矩阵。由流动理论、塑性位势理论、广义胡克定律以及屈服准则，

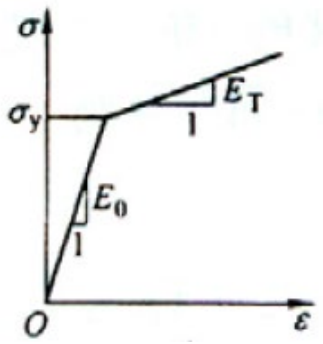
可以推导出弹塑性增量型本构关系：

$$d\sigma=\mathrm{D}_{ep}d\varepsilon$$

其中：

$$\mathrm{D}_{ep}=D-D_p=D-\frac{D\dfrac{\partial Q}{\partial\sigma}\left(\dfrac{\partial F}{\partial\sigma}\right)^{\mathrm{T}}D}{\left(\dfrac{\partial F}{\partial\sigma}\right)^{\mathrm{T}}D\left(\dfrac{\partial Q}{\partial\sigma}\right)}\text{-A}$$

对于线性强化弹塑性：



$$A=\frac{E_0E_T}{E_0-E_T}$$

式中，E_0 为初始弹性模量，E_T 为屈服后的模量。本次实验中是平面应力问题。应力和应变可表示为如下：

$$\sigma=\begin{pmatrix}\sigma_x\\ \sigma_y\\ \tau_{xy}\end{pmatrix}\qquad \varepsilon=\begin{pmatrix}\varepsilon_x\\ \varepsilon_y\\ \gamma_{xy}\end{pmatrix}$$

Von-mises 屈服条件：

$$F=f-k=\frac{1}{2}(S_x^2+S_y^2+2\tau_{xy}^2)-\frac{1}{3}\sigma_s^2$$

其中：

$$\begin{pmatrix}S_x & \tau_{xy}\\ \tau_{xy} & S_y\end{pmatrix}=\begin{pmatrix}\sigma_x-\sigma_m & \tau_{yx}\\ \tau_{yx} & \sigma_y-\sigma_m\end{pmatrix}\qquad \sigma_m=\frac{1}{3}(\sigma_x+\sigma_y)$$

由此可以得到：

$$\frac{\partial F}{\partial\sigma}=\begin{pmatrix}S_x\\ S_y\\ 2\tau_{xy}\end{pmatrix}$$

实验中认为材料满足 Drucker 公设，即：$F=Q$

D 是弹性矩阵：

$$D=\frac{E}{1-\nu^2}\begin{bmatrix}1 & \nu & 0\\ \nu & 1 & 0\\ 0 & 0 & \dfrac{1-\nu}{2}\end{bmatrix}$$

以上，代入到公式中，可以求得 D_P，如下：

$$D_P=\frac{E}{B(1-\nu^2)}\begin{bmatrix}(S_x+\nu S_y)^2 & (S_x+\nu S_y)(S_y+\nu S_x) & (1-\nu)(S_x+\nu S_y)\tau_{xy}\\ (S_x+\nu S_y)(S_y+\nu S_x) & (S_y+\nu S_x)^2 & (1-\nu)(S_y+\nu S_x)\tau_{xy}\\ (1-\nu)(S_x+\nu S_y)\tau_{xy} & (1-\nu)(S_y+\nu S_x)\tau_{xy} & (1-\nu)^2{\tau_{xy}}^2\end{bmatrix}$$

$$B=S_x^2+S_y^2+2\nu S_xS_y+2(1-\nu)\tau_{xy}^2+\frac{2(1-\nu)A\sigma_s^2}{9G}\qquad G=\frac{E}{2(1+\nu)}$$

由以上可以推出弹塑性矩阵 D_{ep}。

4) 数值积分

采用 2×2 Gauss 积分方案，采用精确积分方案。弹性刚度矩阵可以表示为：

$$K=\int\int B^TDBdS=\sum_{i=1}^{4}\left[B(\xi_i\frac{1}{\sqrt{3}},\eta_i\frac{1}{\sqrt{3}})^TDB(\xi_i\frac{1}{\sqrt{3}},\eta_i\frac{1}{\sqrt{3}})\left|J\right|\right]$$

塑性单元刚度矩阵可以表示为：

$$K^{ep}=\int\int B^TD_{ep}BdS=\sum_{i=1}^{4}\left[B(\xi_i\frac{1}{\sqrt{3}},\eta_i\frac{1}{\sqrt{3}})^TD_{ep}B(\xi_i\frac{1}{\sqrt{3}},\eta_i\frac{1}{\sqrt{3}})\left|J\right|\right]$$

5) 等效荷载

外部等效荷载的计算直接使用变分法中的公式：

$$F=\int_V N^{\mathrm{T}}fdV+\int_{S_\sigma}N^{\mathrm{T}}\bar{T}dS$$

其中的积分单元用等参元来计算，其中体元、面元以及线元需要利用雅各比行列式进行变换。本实验中的外部荷载为集中荷载，直接作用在节点上，不需要进行积分运算即可得到荷载向量。

3. 整体分析

由单元刚度矩阵以及等效荷载，可以集成整体刚度矩阵和整体荷载向量，即得到方程组：

$$Ka = P$$

其中，K 为整体刚度矩阵，a 为节点位移，是待求量，P 是整体荷载矩阵。

由此容易求得：

$$a = K^{-1}P$$

整体分析中，按照节点编号将单元矩阵集成为整体刚度矩阵是关键。Matlab 程序通过矩阵，比较容易实现这一过程。

4. 结果后处理

以上整体分析中得到的各节点位移回代到单元方程中，可以得到单元内部的任意位移的应变和应力：

$$\sigma = DB\mathrm{a} \quad \varepsilon = Ba \quad （其中 B = LN \ \text{为应变矩阵}）$$

由于应变矩阵 B 中含有求导操作，得到的应力与应变会损失一阶精度，而且得到的应力和应变值在整体上并不满足连续性。因此需要对各单元的应力和应变值进行磨平处理，单元磨平是比较利于程序运算的。实验中使用的单元为正方形四节点单元。应力单元磨平运算如下：

$$\begin{pmatrix} \sigma_x^1 \\ \sigma_y^1 \\ \tau_{xy}^1 \\ \sigma_x^2 \\ \sigma_y^2 \\ \tau_{xy}^2 \\ \sigma_x^3 \\ \sigma_y^3 \\ \tau_{xy}^3 \\ \sigma_x^4 \\ \sigma_y^4 \\ \tau_{xy}^4 \end{pmatrix} = \begin{pmatrix} a & \cdots & \cdots & b & & & c & & & b & & \\ \vdots & a & \ddots & \ddots & b & & & c & & & b & \\ & & a & & & b & & & c & & & b \\ & & & a & & & b & & & c & & \\ & & & & a & & & b & & & c & \\ & & & & & a & & & b & & & c \\ & & & & & & a & & & b & & \\ & & \text{对} & & \text{称} & & & a & & & b & \\ & & & & & & & & a & & & b \\ & & & & & & & & & a & & \\ & & & & & & & & & & a & \\ & & & & & & & & & & & a \end{pmatrix} \begin{pmatrix} \sigma_x^{\mathrm{I}} \\ \sigma_y^{\mathrm{I}} \\ \tau_{xy}^{\mathrm{I}} \\ \sigma_x^{\mathrm{II}} \\ \sigma_y^{\mathrm{II}} \\ \tau_{xy}^{\mathrm{II}} \\ \sigma_x^{\mathrm{III}} \\ \sigma_y^{\mathrm{III}} \\ \tau_{xy}^{\mathrm{III}} \\ \sigma_x^{\mathrm{IV}} \\ \sigma_y^{\mathrm{IV}} \\ \tau_{xy}^{\mathrm{IV}} \end{pmatrix}$$

$a = 1 + \frac{\sqrt{3}}{2}, b = -\frac{1}{2}, c = 1 - \frac{\sqrt{3}}{2}$；$\left(\sigma_{\mathrm{x}}^{\mathrm{I}} \cdots\cdots \tau_{\mathrm{xy}}^{\mathrm{IV}}\right)^{\mathrm{T}}$ 为*Gauss*点处的应力值，$\left(\sigma_{\mathrm{x}}^{1} \cdots\cdots \tau_{\mathrm{xy}}^{4}\right)^{\mathrm{T}}$ 为单元结点的应力值。

应变的磨平同理。单元内部的应力值可以用节点处的应力值内插得到。

三、 切线刚度法计算弹塑性问题

等效应力达到屈服条件以后，应力与应变由以下关系所确定：$d\sigma = D_{ep}d\varepsilon$。其中弹塑性矩阵$\mathrm{D}_{ep}$中含有应力，它是加载过程的函数。采用增量形式近似代替微分形式，以使求解成为可能。计算中D_{ep}在$\Delta\sigma$范围内变化不大，因此可假设在每一步加载中是一个常数，并且该加载步前的应力状态近似计算出D_{ep} 即：$\Delta\sigma = D_{ep}\Delta\varepsilon$ 。单元刚度矩阵K 在一个加载步中也同样作常数，即：

$$K = \iint B^T D_{ep} B dS \text{。}$$

在一个变形体中，不仅各点的应力状态是不同的，而且随着加载而变化着，通常受变形体外力作用时，从一个区域到另一个区域，等效应力是逐渐地达到屈服极限，即进入塑性状态。为了简化，这里的塑性状态即为弹塑性状态。变形体的单元分为三类：弹性单元、塑性单元和过渡单元。各单元有不同的本构关系和单元刚度矩阵。

对于整体来说，总体刚度矩阵可以表示为：

$$K = \sum_{i=1}^{n_1} k_i^e + \sum_{j=1}^{n_2} k_j^{ep} + \sum_{m=1}^{n_3} k_m^g$$

式中k_i^e是弹性单元刚度矩阵，k_j^{ep}为塑性单元刚度矩阵，k_m^g为过渡单元刚度矩阵。在加载过程中，各单元的状态是变化的，为此K也是变化的。在计算中，每增加一个载荷增量，就是重新计算一次整体刚度矩阵K，这就是变刚度法的由来。

整体刚度矩阵求得以后，就可以根据下列载荷和位移的线性方程组求解出未知的节点位移增量。$K\Delta u = \Delta p$ ，进而就能求得各单元的应变和应力增量。如下：

$$\Delta\varepsilon = B\Delta u \qquad \Delta\sigma = D\Delta\varepsilon = DB\Delta u = S\Delta u$$

同理可以得到弹性单元和过渡单元刚度矩阵。其中在计算过渡单元刚度矩阵的过程中，需要把其中的弹塑性矩阵用弹性矩阵和塑性矩阵来表示：

$$\overline{D_{ep}} = mD + (1-m)D_{ep}$$

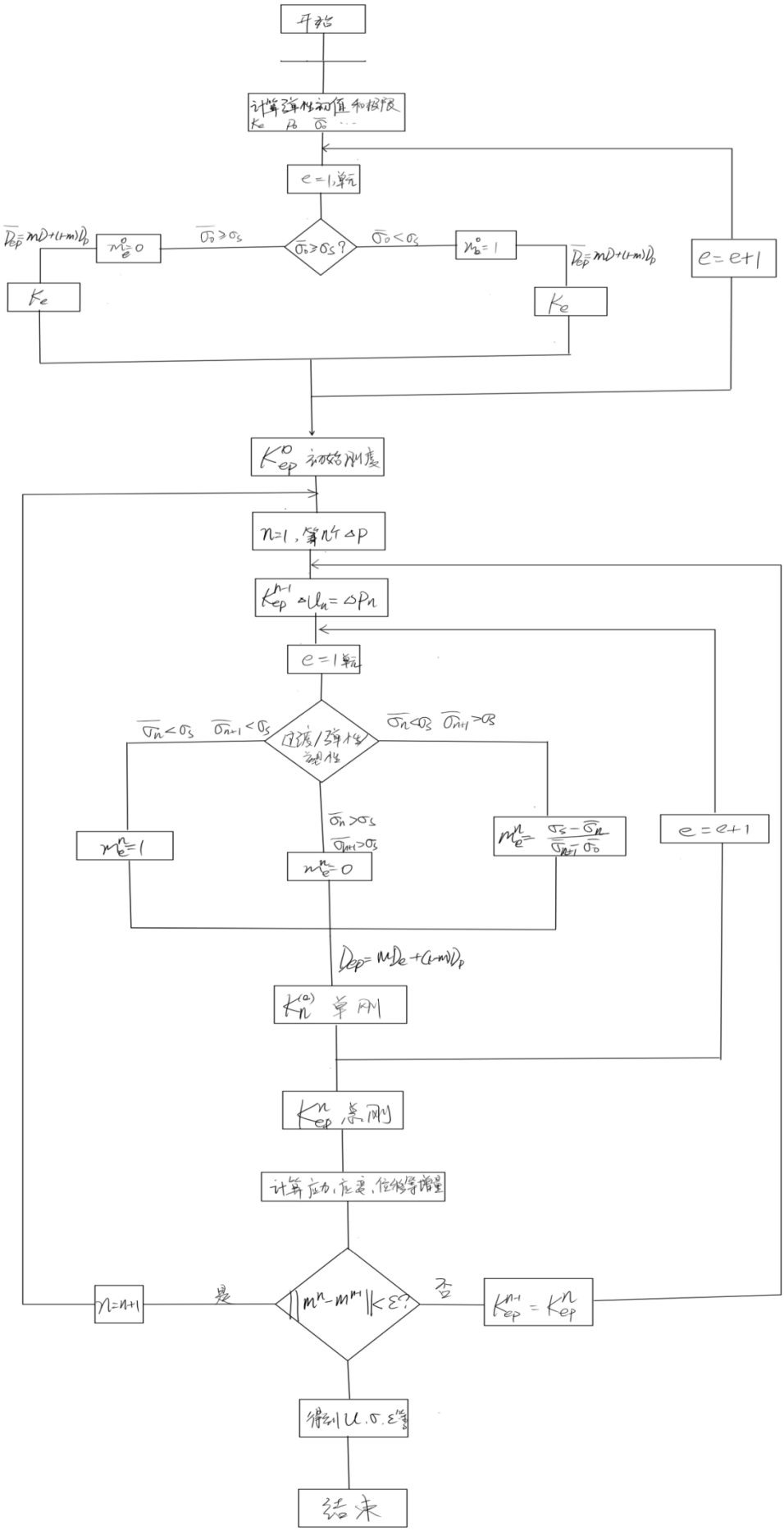
其中：

$$m = \frac{\sigma_s - \bar{\sigma}_n}{\bar{\sigma}_{n+1} - \bar{\sigma}_n}$$

过渡单元的系数值，通过迭代计算得到。

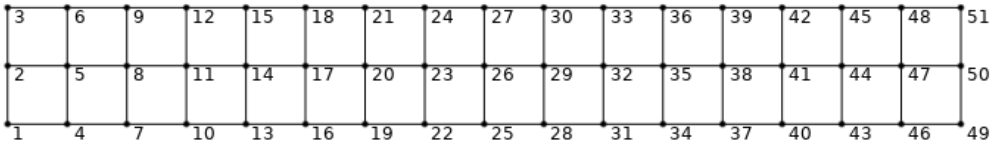
四、　算法与程序设计

1. 程序框图：



2. 单元网格划分

Matlab 网格划分为 0.5m*0.5m 的正方形网格，如下图所示：程序见附录。本实验中采用了三种网格疏密进行对比。单元数分别为：16*2、80*10、160*20. 其编号依旧如图所示规律进行编号。

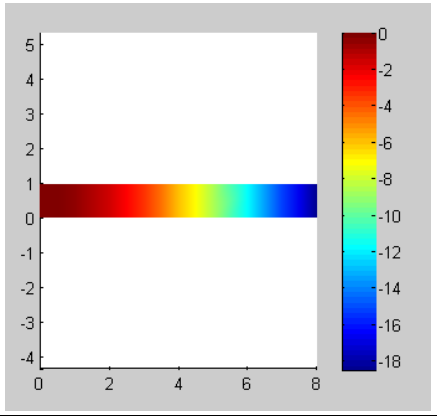
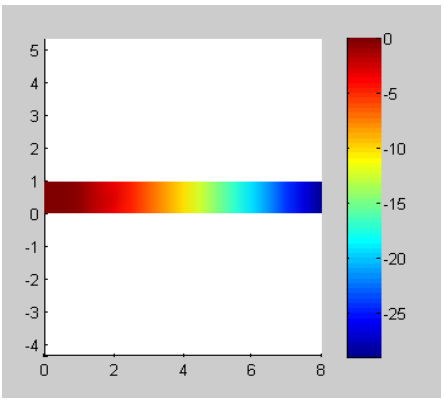


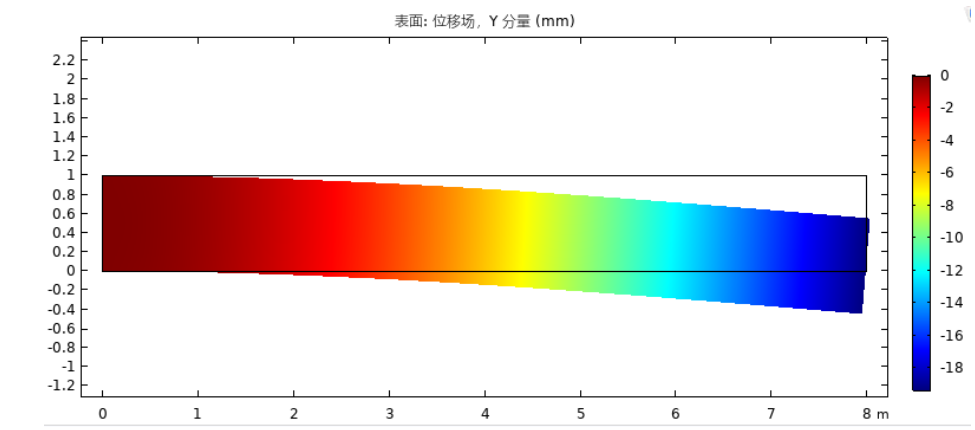
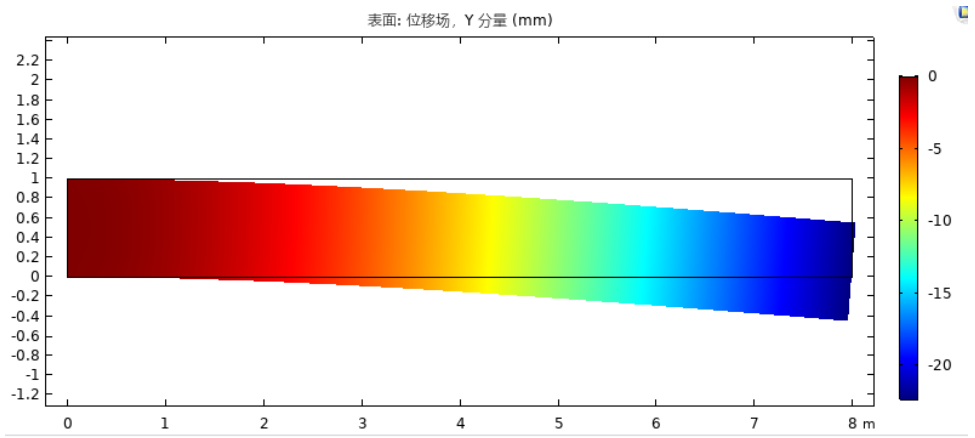
五、 Matlab 计算结果与 Comsol 计算结果与分析

16*2 单元时：纵向位移值（向下） mm

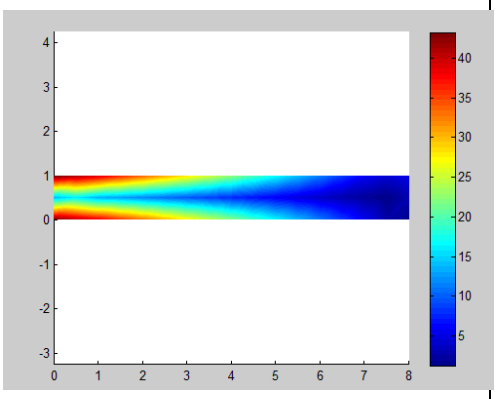
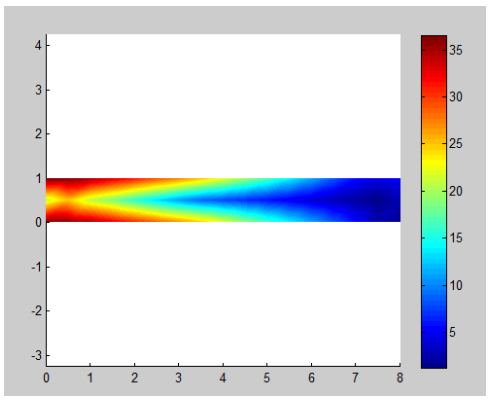
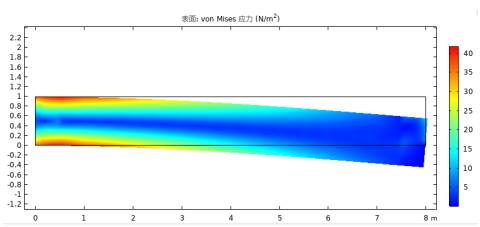
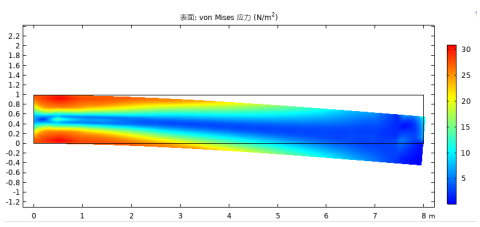
项目	坐标（m）	（1.6，0.5）	（3.2，0.5）	（4.8，0.5）	（6.4，0.5）	（8.0，0.5）
弹性解	弹性参考值	1.180	4.334	8.963	14.575	20.672
	Matlab 弹性解	1.057	3.876	8.008	13.019	18.469
	Comsol 弹性解	1.096	4.046	8.379	13.633	19.339
弹塑性解	弹塑性参考值	1.946	6.728	13.077	20.409	28.236
	Matlab 弹塑性解	2.106	7.164	13.710	21.136	29.00
	Comsol 弹塑性解	1.3992	4.9866	9.9854	15.906	22.277

位移云图：（单元数：16*2）

Matlab：弹性解，竖向位移：	Matlab：弹塑性解，竖向位移：
	

Comsol 弹性解云图	
Comsol 弹塑性解云图	

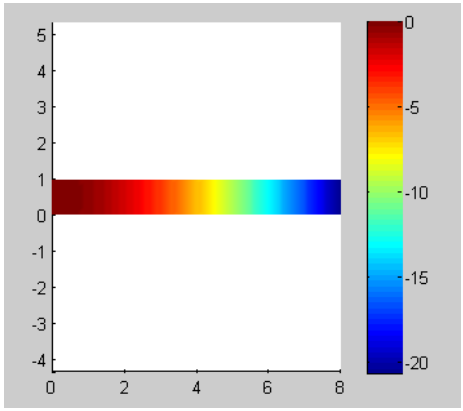
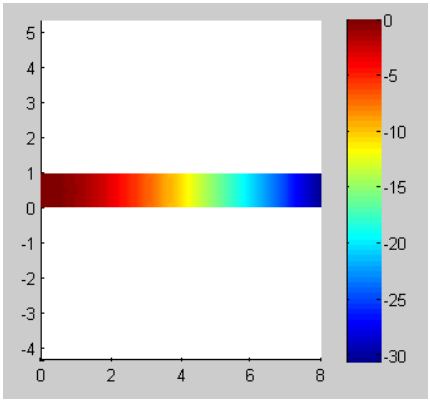
Von-mises 应力云图：（单元数：16*2）

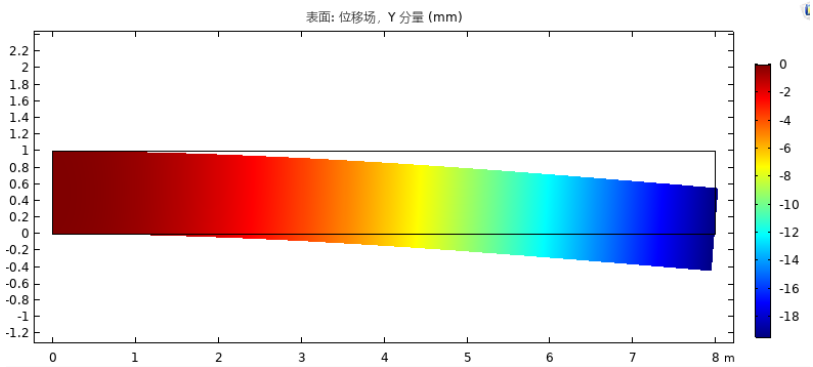
	弹性	弹塑性
Matlab 计算结果		
Comsol 模型		

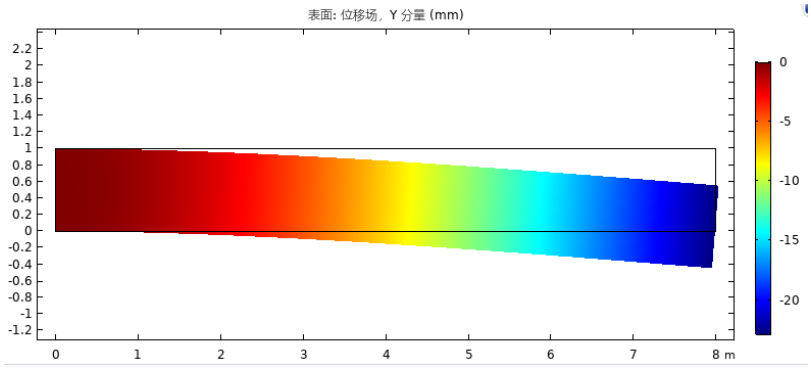
80*10 单元时：纵向位移值（向下） mm

项目	坐标（m）	（1.6，0.5）	（3.2，0.5）	（4.8，0.5）	（6.4，0.5）	（8.0，0.5）
弹性解	弹性参考值	1.180	4.334	8.963	14.575	20.672
	Matlab 弹性解	1.174	4.314	8.921	14.507	20.578
	Comsol 弹性解	1.104	4.061	8.400	13.662	19.380
弹塑性解	弹塑性参考值	1.946	6.728	13.077	20.409	28.236
	Matlab 弹塑性解	2.279	7.543	14.391	22.217	30.529
	Comsol 弹塑性解	1.488	5.1797	10.278	16.297	22.773

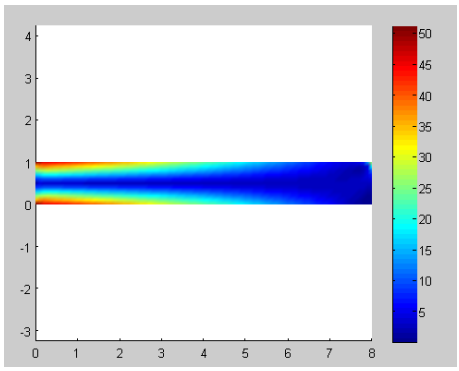
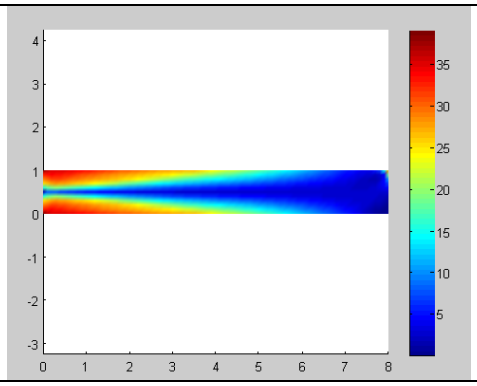
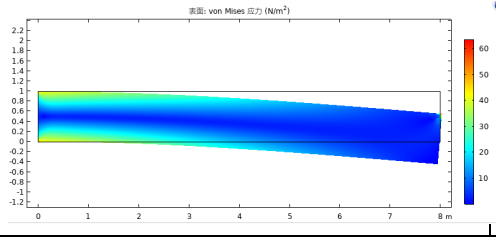
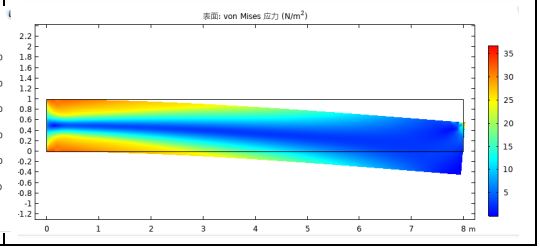
纵向位移云图：（单元数：80*10）

Matlab：弹性解，竖向位移：	Matlab：弹塑性解，竖向位移：
	

Comsol 弹性解 云图	

Comsol 弹塑性解云图	

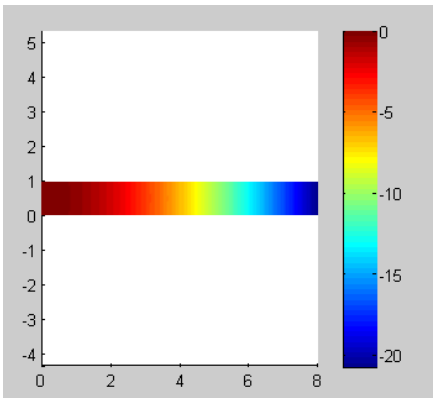
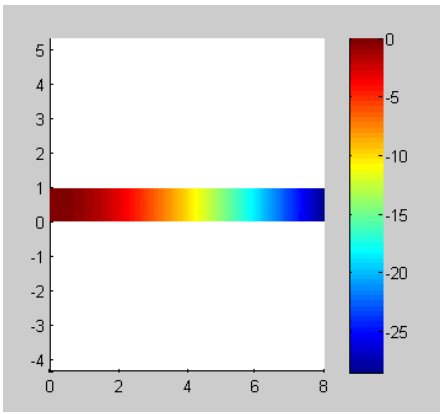
Von-mises 应力云图（单元数：80*10）：

	弹性	弹塑性
Matlab 计算结果		
Comsol 模型		

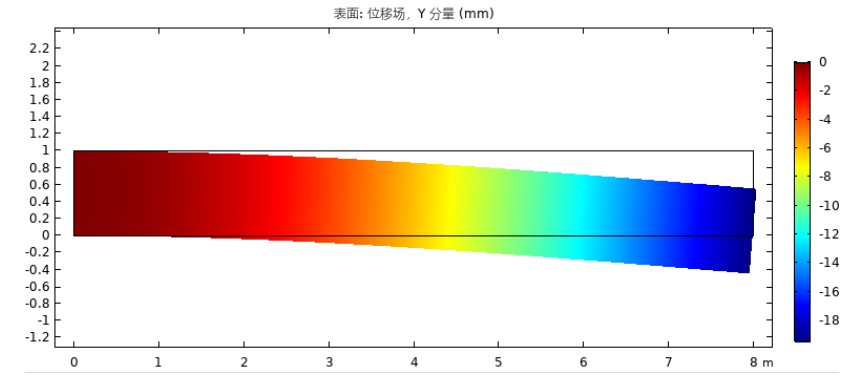
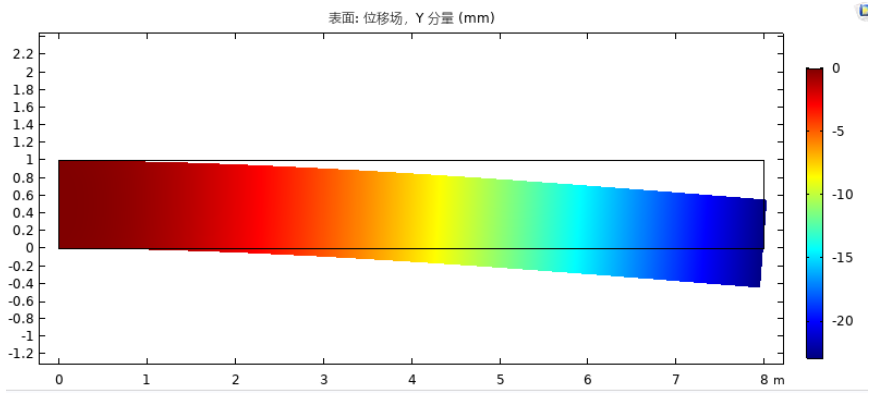
160*20 单元时：纵向位移值（向下） mm

项目	坐标（m）	（1.6，0.5）	（3.2，0.5）	（4.8，0.5）	（6.4，0.5）	（8.0，0.5）
弹性解	弹性参考值	1.180	4.334	8.963	14.575	20.672
	Matlab 弹性解	1.180	4.331	8.956	14.562	20.655
	Comsol 弹性解	1.105	4.062	8.402	13.664	19.382
弹塑性解	弹塑性参考值	1.946	6.728	13.077	20.409	28.236
	Matlab 弹塑性解	2.072	6.880	13.233	20.569	28.388
	Comsol 弹塑性解	1.492	5.188	10.291	16.316	22.797

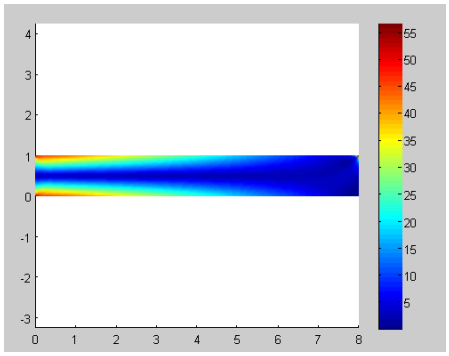
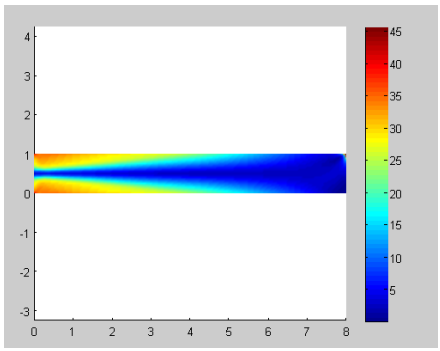
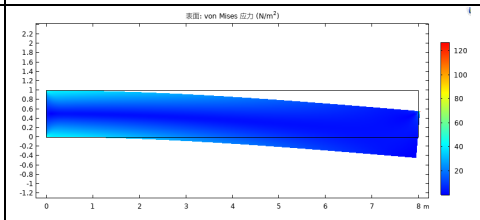
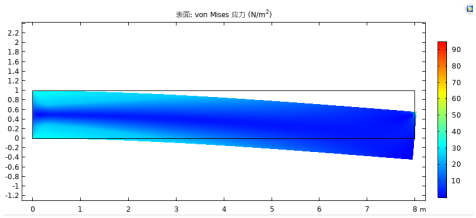
纵向位移云图：（单元数：160*20）

Matlab：弹性解，竖向位移：	Matlab：弹塑性解，竖向位移：
	

Comsol 软件结果：

Comsol 弹性解云图	
Comsol 弹塑性解云图	

Von-mises 应力云图（单元数：160*20）：

	弹性	弹塑性
Matlab 计算结果		
Comsol 模型		

六、 实验总结

从以上得到三种位移结果来看，使用切线刚度法通过 matlab 程序得到解随着单元划分越加细致而得到更加准确的解。与给定的参考值相差很小。Comsol 软件随着网格划分的细化，精度提高不明显，这可能是由于 comsol 软件计算的过程使用方程的积分弱形式。同时，四边形均匀分布的网格，在 comsol 计算中并不是最优的网格形式。

对于应力的计算结果，在单元划分为 16*2 和 80*10 时，采用切线刚度法得到应力解和 comsol 得到的应力解，没有显著的差异。但是当网格划分为 160*20 时，comsol 计算结果中出现部分奇异点，计算结果比用切线刚度法 matlab 程序得到的结果要差。

另外，在程序设计过程中，关于过渡单元系数的计算，对计算结果影响很小，这种影响也可以通过细化网格划分弥补。实际上，从 matlab 计算的结果来看，在 16*2 的单元划分时，得到的弹性和塑性结果误差较大。随着网格细化到 80*10 时，弹性解的精度增加，而弹塑性解的误差依然较大，进一步细化网格成 160*20 时，弹性和弹塑性解，都达到了较好的精度，与参考值差异较小。

虽然经过迭代能够得到稳定的 m 系数值，但是极大增加了程序的复杂程度，建立的多

维数组，浪费内存，影响计算速度。当采用 160*20 个单元时，需要接近两分钟的计算时间，较 comsol 软件计算的时间要长约 0.5 分钟。Matlab 程序虽然得到了较好的计算结果，但是程序设计复杂，效率不高。 Comsol 软件以多物理场耦合为主要的计算任务，对于各种不同的微分方程有较好的适应性，操作简单。在不要求较高精度的情况下，是一种高效的有限元分析工具。

附录：Matlab 代码

```
%% 这是一个平面应力弹塑性有限元程序（主程序）
%% 四边形网格
clc;  % 清空命令行窗口
clear; %清除工作空间
close all; %关闭所有图像
%% 参数设置及网格划分
error_L2s = [];
E=1e5; %弹性模量
Et=0.2*E;
nu = 0.25; %泊松比
sigmas=25; %mises 屈服应力强度
Lx = 8; %定义单元右边界（左边界为 0）
Ly = 1;%定义单元上边界;
numelx = 16;%定义分割的 x 方向单元数目
numely = 2;%定义分割的 y 方向单元数目
hx = Lx/numelx;%x 方向上的单元长度
hy = Ly/numely;%y 方向上的单元长度
numel = numelx*numely;%单元的数目
u_b = zeros(2*(numely+1),1); %定义第一类边界条件, x=0, 处位移为 0
numnodx = numelx + 1; % x 方向节点个数比单元个数多 1
numnody = numely + 1; % y 方向节点个数比单元个数多 1
numnod = numnodx*numnody;%总的节点个数
nel = 4;%每个单元的节点数目,即每个单元上有几个形函数参与作用
coordx = linspace(0,Lx,numnodx)'; %等分节点的坐标（为了方便，我这里采用等分的方式，事实上单元长度可以不一致，非均匀网格）
coordy = linspace(0,Ly,numnody)'; %等分节点的坐标（为了方便，我这里采用等分的方式，事实上单元长度可以不一致）
```

```
[X, Y] = meshgrid(coordx,coordy);%张成网格，X和Y分别表示对应位置的横纵坐标
coord = [X(:) Y(:)];%把网格一列一列扯开，coord的每一行是对应节点的坐标，按顺序排列
[connect,A] = connect_mat(numnodx,numnody,nel);%连接矩阵，表示每个单元周围的节点编号，也就是涉及的形函数编号
nodeb = 1:numnody; % 强制性边界点的编号
nodeval = u_b; %假设边界值都为u_b
P=zeros(2*numnod,1);
P(end) = -1;  %节点荷载

%%  计算系数矩阵K和右端荷载矩阵，单刚组装总刚，以及边界处理
D=elasm(E,nu); %广义胡克定律系数矩阵
K= totalstiff(D,connect,coord,numel,nel,numnod );
bnoderow = zeros(2*length(nodeb),1);
for i =1:length(nodeb)
    bnoderow(2*i-1:2*i,1)=[2*nodeb(i)-1;2*nodeb(i)];
end
K(bnoderow,:)=[];
K(:,bnoderow)=[]; %删除掉位移为0的点对应，刚度矩阵的行和列
P(bnoderow)=[];  %删除固定边界对应的荷载分量
a = K\P;  %计算弹性解
tota = zeros(2*numnod,1);
rowke = setdiff(1:numnod,nodeb);
rowk=setdiff(1:2*numnod,bnoderow);
tota(rowk)=a;
u=zeros(numnod,1); %水平位移
v=zeros(numnod,1); %纵向位移
wholea=zeros(numnod,1); %总位移
for i = 1:numnod
    u(i) = tota(2*i-1);%取出水平位移
    v(i) = tota(2*i); %取出纵向位移
    wholea(i) = sqrt(u(i)^2+v(i)^2); %计算总位移，注意和tota的区别
end

%% 绘图
u = reshape(u,numnody,numnodx);   %重构横向位移矩阵
v = reshape(v,numnody,numnodx); %重构纵向位移矩阵
subplot(2,2,1)
surface(coordx,coordy,1000*v);colorbar;  %绘制位移表面
shading interp
ylim([0,1])    %控制轴坐标范围
xlim([0,8])
axis equal   %轴单位相同
subplot(2,2,3)
mesh(coordx,coordy,1000*v)
```

```
ylim([0,1])    %控制轴坐标范围
xlim([0,8])
% axis equal   %轴单位相同

%% 结果后处理
%应力计算
%应变计算
gaussxi=1/sqrt(3)*[-1,1,1,-1];  %高斯点 xi 坐标
gausseta=1/sqrt(3)*[-1,-1,1,1];
stress=zeros(numel,nel*3);
strain=zeros(numel,nel*3);
for e = 1:numel
    %计算 gauss 点处的应力和应变
    for i_gauss = 1:4 %逐个取高斯点
        [Bgauss,~] = B(e,coord,connect,gaussxi(i_gauss),gausseta(i_gauss));%计算各单元高斯点处的应变矩阵
        %1-3 列为第一个高斯点的值，依此类推

[stress(e,3*i_gauss-2:3*i_gauss),strain(e,3*i_gauss-2:3*i_gauss)]=stress_strain(e,tota,connect,D,Bgauss
);
    end
end

%% 磨平
%利用高斯点外推节点应力 sigma 每一行是每一个单元，高斯点处的应力值，分别为 sigmax，sigmay,tauxy
%注意到 sigma 行向量的特点，构建外推矩阵
apara=1+sqrt(3)/2;
bpara=-1/2;
cpara=1-sqrt(3)/2;
aeye = eye(3)*apara;
beye = eye(3)*bpara;
ceye = eye(3)*cpara;
eoutp=eye(4)*apara+diag([1,1,1]*bpara,1)+diag([1,1,1]*bpara,-1)+diag([1,1]*cpara,2)+diag([1,1]*cpara,-2
)+diag(bpara,3)+diag(bpara,-3);
outp=zeros(12,12);
for i =1:4  %利用 for 循环构造外推矩阵
    outp(3*i-2:3*i,3*i-2:3*i)=aeye;
    if i < 4
    outp(3*i-2:3*i,3*i+1:3*i+3)=beye;
    outp(3*i+1:3*i+3,3*i-2:3*i)=beye;
    if i < 3
    outp(3*i-2:3*i,3*i+4:3*i+6)=ceye;
    outp(3*i+4:3*i+6,3*i-2:3*i)=ceye;
    end
    else
```

```
        outp(1:3,10:12)=beye;
        outp(10:12,1:3)=beye;
    end
end
smnode=stress*outp; %得到节点处磨平后的应力值

%% 计算每一个单元的最大等效应力
%单元的最大等效应力在单元节点处，见 test.m 中的绘图结果。
sigma_max = zeros(32,1);
Ixi=[-1,1,1,-1];
Ieta=[-1,-1,1,1];
for e =1:numel
    sigma_i = zeros(4,1);
    for i =1:4
        [~,sigma_i(i)] = sigmainner(smnode, e,Ixi(i),Ieta(i));
    end
    sigma_max(e) =max(sigma_i);  %取得单元的最大等效应力，行数对应单元数
end

%% 变刚度切线法求弹塑性问题
%首先计算出最大弹性荷载 P0 对应的弹性解
totsigma = max(sigma_max); %判断 sigma_max 与 sigmas 之间的关系
L = totsigma/sigmas;
P0=1/L*P;
n = 10; %设定剩余荷载的增量数量
deltP=1/n*(1-1/L)*P;
[sigma0_max,sigma0,tota0,smnode0] =
solver( K,P0,D,numnod,numel,nel,bnoderow,outp ,Ixi,Ieta,coord,connect,gaussxi,gausseta);

%% 施加固定荷载增量 deltP
%这里的关键在于变单元刚度
% sigma1_max = sigma0_max; %初始化最大等效应力为最大弹性荷载时的的值
elsigma_max = sigma0_max;
gausstress=sigma0;
Kepll=sparse(2*numnod,2*numnod);
me=zeros(numel,1);
for e = 1:numel
    if (elsigma_max(e)-sigmas)*(abs(elsigma_max(e)-sigmas)>1e-10)<0   %设定精度为 1e-10.
        me(e)=1;
    else
        me(e)=0;
    end
    Ke = zeros(8,8);
    for i_gauss = 1:4 %采用高斯积分方案
```

```
        [Bgauss,J] = B(e,coord,connect,gaussxi(i_gauss),gausseta(i_gauss));%计算各单元高斯点处的应变矩阵
        Dep = e_plasm( sigmas,E,Et,nu,gausstress(e,(3*i_gauss-2:3*i_gauss)));
        Dhat = me(e)*D+(1-me(e))*Dep;
        Ke = Ke+ Bgauss'*Dhat*Bgauss*det(J); %单元刚度矩阵
    end
    totn = connect(e,:); %单元节点编号
    totn2 = zeros(2*length(totn),1);  %有水平和竖直位移, 矩阵扩大一倍
    for i =1:nel
        totn2(2*i-1:2*i,1) =  [2*totn(i)-1;2*totn(i)];  %扩大后单元节点对应的行号
    end
    Kepll(totn2,totn2) = Kepll(totn2,totn2)+Ke;  %组装整体刚度

end
Kepll(bnoderow,:) = [];
Kepll(:,bnoderow) = [];
[kx,ky]=size(Kepll);
Kep=zeros(kx,ky,n+1);
Kep(:,:,1) = full(K);
delu=zeros(n,length(P));
deltotu = zeros(2*numnod,n);
delsigma = zeros(numel,3*nel,n);
delsigma_max =zeros(e,n);
delsigma_max(:,1)=sigma0_max;
m=zeros(numel,n);
m(:,1)=me;
sigmatot(:,:,1)=sigma0;
smdelsigma(:,:,1)=smnode0;
totu=tota0;
j = 0;
for ip = 1:n;
    jwhile=1;
    kwhile=0;
%     while 1
    kwhile=kwhile+1;
    delu(ip,:)=Kep(:,:,ip)\deltP;  %deltaU_2
    deltotu(:,ip) = zeros(2*numnod,1);
    deltotu(rowk,ip)=delu(ip,:);
    for e = 1:numel
    %计算 gauss 点处的应力和应变
        for i_gauss = 1:4 %逐个取高斯点
            [Bgauss,~] = B(e,coord,connect,gaussxi(i_gauss),gausseta(i_gauss));%计算各单元高斯点处的应变
矩阵
            Dep = e_plasm( sigmas,E,Et,nu,sigmatot(e,(3*i_gauss-2:3*i_gauss),ip));
            Dhat = m(e,ip)*D+(1-m(e,ip))*Dep;
```

```
[delsigma(e,3*i_gauss-2:3*i_gauss,ip),~]=stress_strain(e,deltotu(:,ip),connect,Dhat,Bgauss);
        end
        s = zeros(ip,3*nel);
        sm = zeros(ip,3*nel);
        for is=1:ip
            s(is,:) =delsigma(e,:,is);
            sm(is,:)=delsigma(e,:,is)*outp;
        end
        sigmatot(e,:,ip+1)=sigmatot(e,:,1)+sum(s);
        smdelsigma(e,:,ip+1)=smdelsigma(e,:,1)+sum(sm); %每个单元应力磨平后的节点值
        delsigma_i = zeros(nel,1);
        for i =1:nel
            [~,delsigma_i(i)] = sigmainner(smdelsigma(:,:,ip+1), e,Ixi(i),Ieta(i));
        end
        delsigma_max(e,ip+1) =max(delsigma_i);  %取得单元的最大等效应力，行数对应单元数
        if (delsigma_max(e,ip)-sigmas)*(abs(delsigma_max(e)-sigmas)>1e-10)<0
            if (delsigma_max(e,ip+1)-sigmas)*(abs(delsigma_max(e,ip+1)-sigmas)>1e-10)<0
                m(e,ip+1) = 1;
            else
m(e,ip+1)=(sigmas-delsigma_max(e,ip))/(delsigma_max(e,ip+1)-delsigma_max(e,ip));
                jwhile=jwhile+1;
            end
        else
            m(e,ip+1)=0;
        end
        Ke = zeros(8,8);
        for i_gauss = 1:4 %采用高斯积分方案
            [Bgauss,J] = B(e,coord,connect,gaussxi(i_gauss),gausseta(i_gauss));%计算各单元高斯点处的应变
矩阵
            Dep = e_plasm( sigmas,E,Et,nu,sigmatot(e,(3*i_gauss-2:3*i_gauss),ip+1));
            Dhat = m(e,ip+1)*D+(1-m(e,ip+1))*Dep;
            Ke = Ke+ Bgauss'*Dhat*Bgauss*det(J); %单元刚度矩阵
        end
        totnin = connect(e,:); %单元节点编号
        [~,In] = setdiff(totnin,nodeb);
        totn = totnin(sort(In));
        [~,kkill,~] = intersect(totnin,nodeb);
        if isempty(kkill)==0
                kl = [2*kkill-1;2*kkill];
                Ke(kl,:) = [];
                Ke(:,kl) = [];
        end
        totn2 = zeros(2*length(totn),1);  %有水平和竖直位移，矩阵扩大一倍
```

```
        for i =1:length(totn)
            nr = find(rowke==totn(i));
          totn2(2*i-1:2*i,1) =  [2*nr-1;2*nr];  %扩大后单元节点对应的行号
        end
        Kep(totn2,totn2,ip+1) = Kep(totn2,totn2,ip+1)+Ke;  %组装整体刚度
    end
    epm(:,kwhile)=m(:,ip+1);
%     if jwhile >1
%         Kep(:,:,ip) = Kep(:,:,ip+1);
% %         if norm(epm(:,jwhile+1)-epm(:,ip))<1e-3
% %             j=j+1
% %             break
% %         end
%          continue
%     else
%         break
%     end
%     end
    totu = totu + deltotu(:,ip);
end
uep=zeros(numnod,1); %水平位移
vep=zeros(numnod,1); %纵向位移
wholea=zeros(numnod,1); %总位移
for i = 1:numnod
    uep(i) = totu(2*i-1);%取出水平位移
    vep(i) = totu(2*i); %取出纵向位移
end

%% 绘图
uep = reshape(uep,numnody,numnodx);   %重构横向位移矩阵
vep = reshape(vep,numnody,numnodx); %重构纵向位移矩阵
subplot(2,2,2)
surface(coordx,coordy,1000*vep);colorbar;  %绘制位移表面
shading interp
ylim([0,1])    %控制轴坐标范围
xlim([0,8])
axis equal   %轴单位相同
subplot(2,2,4)
mesh(coordx,coordy,1000*vep)
ylim([0,1])    %控制轴坐标范围
xlim([0,8])
% axis equal   %轴单位相同
```

调用的函数：

```
function [connect_mat,A] = connect_mat( numnodx,numnody,nel)
```

```
%输入横纵坐标的节点数目，和单元自由度
%输出连接矩阵，每个单元涉及的节点的编号
xn = 1:(numnodx*numnody);%拉成一条编号
A = reshape(xn,numnody,numnodx);%同形状编号
B = zeros(numnody,numnodx);
for i = 1:numnody
B(i,:)=A(numnody+1-i,:);%和坐标轴一直，向右和向上为正。
end
A=B;
for i = 1:(numnodx-1)*(numnody-1)
    yg = rem(i,numnody-1);%xg 表示单元为下边界数起第几个
    if yg == 0
        yg = numnody-1;
    end
    xg = ceil(i/(numnody-1));%左边界其数第几个
    a = A(numnody-yg:numnody-yg+1,xg:xg+1);%这个小矩阵，拉直了就是连接矩阵
    a_veg = a(:);
    connect_mat(i,1:nel) =a_veg([2 4 3 1]);   %逆时针编码，与标准四节点单元编码一致，左下为 1
end
end
```

```
function [ sigmainner,sigmahat ] = sigmainner( smnode, e,xi,eta )
%单元内部得到任意位置的应力值和该点的等效应力值
%计算单元内部的应力，用节点处磨平后的应力值内插   smnode 是磨平以后的单元节点值；e 是单元数，xi 和 eta 是单元等参坐标
% sigmainner 是一个一行三列的矩阵。分别对应：sigmainnerx sigmainnery tauxy
%sigmahat 是等效应力值 判断是否进入塑性，及判断 sigmahat 与 sigmas 之间的关系
N1 = 1/4*(1-xi)*(1-eta);
N2 = 1/4*(1+xi)*(1-eta);
N3 = 1/4*(1+xi)*(1+eta);
N4 = 1/4*(1-xi)*(1+eta);
N = [eye(3)*N1;eye(3)*N2;eye(3)*N3;eye(3)*N4];
sigmainner = smnode(e,:)*N;
sigmam=1/3*(sigmainner(1)+sigmainner(2));
Sx = sigmainner(1)-sigmam;
Sy = sigmainner(2)-sigmam;
Sz = 0-sigmam;
tauxy=sigmainner(3);
sigmahat = sqrt(3)*sqrt(1/2*(Sx^2+Sy^2+Sz^2)+tauxy^2);
end
```

```
function [ estress,estrain ] = stress_strain(e, tota,connect,D,B)
%该函数用来计算应力值和应变值
```

```
% 计算任意位置的应力和应变，xi 和 eta 坐标任意，由 B 的值来体现
nodes = connect(e,:);
unodes = zeros(2*length(nodes),1);
for i = 1:length(nodes)
    unodes(2*i-1) = 2*nodes(i)-1;
    unodes(2*i) = 2*nodes(i);
end
ae = tota(unodes);
estress = D*B*ae;
estrain = B*ae;
end
```

```
function [B,J] = B(e,coord,connect,gaussxi,gausseta)
%本函数用于计算 gauss 点处的应变矩阵 B 的值。
nodes = connect(e,:);%相关形函数（节点）编号
xe    = coord(nodes,:); %整体坐标系下，单元节点的坐标
% 构建局部坐标系下的形函数 N{i}
  J_par=zeros(2,4);
    %计算雅各比矩阵 J  %%另外一中采用元胞方法进行计算的方法：N{i} =@(xi,eta)
1/4*(1+Ixi(i)*xi)*(1+Ieta(i)*eta); %形函数   注意：调用的时候要用{}
  B=zeros(3,8); %定义初始的应变矩阵 B
  Ixi=[-1,1,1,-1];
  Ieta=[-1,-1,1,1]; %符号控制
              for k = 1:4
                  eta =gausseta ;%gauss 点对应的局部坐标 eta
                  xi = gaussxi;%gauss 点对应的局部坐标 xi
                  J_par(1,k) = Ixi(k)*1/4*(1+Ieta(k)*eta);  %雅各比矩阵偏导系数第一行，第 i 个高斯点的值
                  J_par(2,k) = Ieta(k)*1/4*(1+Ixi(k)*xi);  %雅各矩阵比偏导系数第二行，第 i 个高斯点的值
              end
%               Jgauss = permute(J_par(i,:,:),[2 3 1]);  %第 i 个高斯点处的雅各比矩阵
                  J =  J_par*xe;
              for i = 1:4%第 i 个高斯点的雅各比矩阵 J
                  N_ipar(:,i) = inv(J)*J_par(:,i);
                  B(1,2*i-1)=N_ipar(1,i);
                  B(2,2*i)=N_ipar(2,i);
                  B(3,2*i-1:2*i)=[N_ipar(2,i),N_ipar(1,i)];
              end
return
end
```

```
function [ D ] = e_plasm( sigmas,E,Et,nu,sigma )
%该函数用来生成弹塑性矩阵函数，用于增量法的计算
% 第一步：构建应力偏量,函数需要输入的是应力向量 sigma=（sigmax,sigmay,tauxy）
```

```
sigmam=1/3*(sigma(1)+sigma(2));
Sx = sigma(1)-sigmam;
Sy = sigma(2)-sigmam;
tauxy=sigma(3);
G=E/(2*(1+nu));
A=E*Et/(E-Et);
B = Sx^2+Sy^2+2*nu*Sx*Sy+2*(1-nu)*tauxy^2+2*(1-nu)*A*sigmas^2/(9*G);
Dp = E/(B*(1-nu^2))*[(Sx+nu*Sy)^2,(Sx+nu*Sy)*(Sy+nu*Sx),(1-nu)*(Sx+nu*Sy)*tauxy;
    (Sx+nu*Sy)*(Sy+nu*Sx),(Sy+nu*Sx)^2,(1-nu)*(Sy+nu*Sx)*tauxy;
    (1-nu)*(Sx+nu*Sy)*tauxy,(1-nu)*(Sy+nu*Sx)*tauxy,(1-nu)^2*tauxy^2];
D = elasm(E,nu) - Dp ;
return
end
```

```
function [ De ] = elasm(E,nu  )
%UNTITLED3 此处显示有关此函数的摘要
%   此处显示详细说明
De=E/(1-nu^2)*[1,nu,0;nu,1,0;0,0,(1-nu)/2];
return
end
```

```
function [sigma0_max,sigma0,tota0,smnode0] =
solver( K,P0,D,numnod,numel,nel,bnoderow,outp,Ixi,Ieta,coord,connect,gaussxi,gausseta )
%该函数用来求解给定总刚度，荷载和本构关系时的各单元最大等效应力
%   除了以上的变量以外，其余均为网格划分时的常数。
%该方程的缺点：不是变刚度的方程。
a0=K\P0;%计算 P0 作用时弹性解
tota0 = zeros(2*numnod,1);
rowk0=setdiff(1:2*numnod,bnoderow);
tota0(rowk0)=a0;
u0=zeros(numnod,1); %P0 荷载对应的水平位移
v0=zeros(numnod,1); %P0 荷载对应的纵向位移
for i = 1:numnod
u0(i) = tota0(2*i-1);%P0 荷载对应取出水平位移
v0(i) = tota0(2*i); %P0 荷载对应取出纵向位移
end
sigma0=zeros(numel,nel*3);
strain0=zeros(numel,nel*3);
for e = 1:numel
%计算 gauss 点处的应力和应变
for i_gauss = 1:4 %逐个取高斯点
[Bgauss,~] = B(e,coord,connect,gaussxi(i_gauss),gausseta(i_gauss));%计算各单元高斯点处的应变矩阵
[sigma0(e,3*i_gauss-2:3*i_gauss),~]=stress_strain(e,tota0,connect,D,Bgauss);
```

```
end
end
smnode0=sigma0*outp; %得到节点处磨平后的应力值
sigma0_max = zeros(numel,1);
for e =1:numel
sigma0_i = zeros(nel,1);
for i =1:nel
[~,sigma0_i(i)] = sigmainner(smnode0, e,Ixi(i),Ieta(i));
end
sigma0_max(e) =max(sigma0_i);  %取得单元的最大等效应力，行数对应单元数
end
end
```

```
function [ K ] = totalstiff( D,connect,coord,numel,nel,numnod )
%计算单元刚度矩阵Ke，并集成总刚度K
%connect 是单元节点编号 coord是单元节点对应的坐标   D 是本构方程系数矩阵 numel是单元总数量 nel单元节点数
% numnod是总结点数
  gaussxi=1/sqrt(3)*[-1,1,1,-1];  %高斯点xi坐标
  gausseta=1/sqrt(3)*[-1,-1,1,1];  %高斯点eta坐标
  K = sparse(2*numnod,2*numnod); % 刚度矩阵[K]，初始化为0，使用稀疏矩阵存储
for e = 1:numel %同一维的情况，依然按单元来扫描
    Ke = zeros(8,8);
    for i_gauss = 1:4 %采用高斯积分方案
        [Bgauss,J] = B(e,coord,connect,gaussxi(i_gauss),gausseta(i_gauss));%计算各单元高斯点处的应变矩阵
        Ke = Ke+ Bgauss'*D*Bgauss*det(J); %单元刚度矩阵
    end
  totn = connect(e,:); %单元节点编号
  totn2 = zeros(2*length(totn),1);  %有水平和竖直位移，矩阵扩大一倍
  for i =1:nel
      totn2(2*i-1:2*i,1) =  [2*totn(i)-1;2*totn(i)];  %扩大后单元节点对应的行号
  end
  K(totn2,totn2) = K(totn2,totn2)+Ke;  %组装整体刚度
end
end
```

另附，结果处理 von-mises 应力计算程序，弹塑性部分只需将弹塑性计算结果导入即可。

```
pp=zeros(numnod,1);
n=1;
for e =1:numel
    aa=connect(e,:);
    for j=1:nel
        m=aa(j);
        pp(m,n)=sigplotf(j,e);
    end
```

```
     n=n+1;
end
for i = 1:numnod
    py(i,1) = sum(pp(i,:))/length(find(pp(i,:)));
end
sig = reshape(py,numnody,numnodx);   %重构横向位移矩阵
surface(coordx,coordy,sig);colorbar;  %绘制位移表面
shading interp
ylim([0,1])    %控制轴坐标范围
xlim([0,8])
axis equal
```

计算给定点的位移值：直接使用单元边界的形函数插值。

```
for ani=1:4
    xint=fix(numelx/5*ani+1);
    yint=fix(numely/2+1);
    rateright=numelx/5*ani+1-xint;
    vpoint(ani)=(1-rateright)*v(yint,xint)+rateright*v(yint,xint+1);
    veppoint(ani)=(1-rateright)*vep(yint,xint)+rateright*vep(yint,xint+1);
end
```