

桥梁大作业报告

组长：贺琪

组员：陈煜 刘畅武 杨昊光 朱子霖

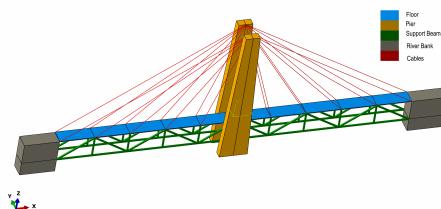
2016年12月25日

目录

0.1 总述

0.1.1 项目描述

桥模型由桥墩（pier），桥面（floor）, 支撑梁（support beam），河堤（river bank）和钢缆（cables）五部分组成，其中桥墩和河堤用实体单元建模，桥面用板单元建模，支撑梁用梁单元建模，钢缆用杆单元建模，如图1所示。

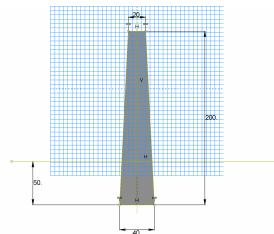


桥墩:

桥墩在XZ方向为左右对称梯形，如图2所示，梯形高200，上底为20，下底40，在y方向上厚度为10。桥面位于距桥墩底50处。两个桥墩顶面内侧中点为所有钢缆的与桥墩的连接点。（参照图1）

采用实体单元建模

材料	:	Concrete
弹性模量	:	25e9
泊松比	:	0.3
密度	:	2320



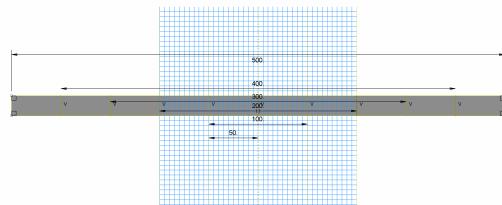
桥面:

桥面位于z=0的平面内，如图3所示，为长方形，长为500，宽为20，厚度为1。在桥面上下边对称地布置钢缆连接点，每个钢缆连接点相距50，共

计 $2 \times 2 \times 5 = 20$ 个钢缆连接点。每根钢缆另一端连接桥墩顶面内侧中点。(参
照图1)

采用板单元建模

材料	:	Concrete
弹性模量	:	25e9
泊松比	:	0.3
密度	:	2320



河堤:

河堤为 $50 \times 50 \times 20$ 的立方体, 如图1所示, 变长为20的一边与桥面相接
接, 另外在距离底面20处与支撑梁相铰接。(铰接指对应结点平动自由度相
同, 转动自由度自由)

采用实体单元建模。

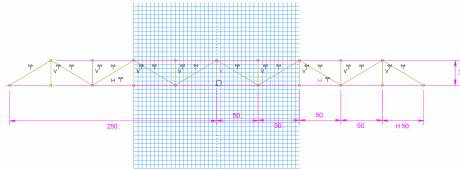
材料	:	Granite
弹性模量	:	60e9
泊松比	:	0.27
密度	:	2770

支撑梁:

支撑梁共有两组, 分别位于桥面两侧下方, 其结构左右对称, 如图4所
示。支撑梁上部每个结点与桥面相铰接, 两组共计 $2 \times 9 = 18$ 个结点。两端结
点与河堤相铰接, 共计 $2 \times 2 = 4$ 个结点。

采用梁单元建模, 梁截面为正方形筒, 边长为2, 厚度为0.1。

材料	:	Aluminum
弹性模量	:	70e9
泊松比	:	0.346
密度	:	2710

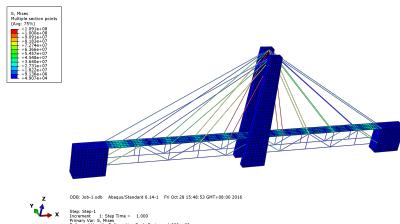


钢缆:

连接桥面与桥墩，共计 $2 \times 2 \times 5 = 20$ 根。每根截面积为0.25。
采用杆单元建模

材料	:	Steel
弹性模量	:	117e9
泊松比	:	0.266
密度	:	7860

0.1.2 Abaqus结果



0.2 桥梁功能实现方案

0.2.1 前处理

前处理使用python与abaqus二次开发进行对INP文件的预处理，将其变成stap90能读入的标准文件。

读入文件格式

功能开关: 本小组采用的标准文件读入格式在规定的stap90固定格式基础上添加：

Bridge 0110									
	4087	4	1	1	0	0	0	200.000	-10.000
1	0	0	0	0	0	0	0	200.000	10.000
2	0	0	0	0	0	0	0	150.000	10.000
3	0	0	0	0	0	0	0	150.000	-10.000
4	0	0	0	0	0	0	0	100.000	10.000
5	0	0	0	0	0	0	0	100.000	-10.000
6	0	0	0	0	0	0	0	50.000	10.000
7	0	0	0	0	0	0	0	50.000	-10.000
8	0	0	0	0	0	0	0	0.000	10.000
9	0	0	0	0	0	0	0	0.000	-10.000
10	0	0	0	0	0	0	0	0.000	3425
11	0	0	0	0	0	0	0	-50.000	10.000
12	0	0	0	0	0	0	0	-50.000	-10.000
13	0	0	0	0	0	0	0	-100.000	10.000

对应的数字分别是以下功能开关:

BANDWIDTHOPT,PARDISODOOR,LOADANALYSIS,DYNANALYSIS,

Pardiso: 同时, 我们需要在后面添加Pardiso所需的单元节点号 (重复一遍)

1759	3127	3128	2915	2914	3424
1760	1.总单元数	2910	2834	2915	3425
2884					
4	2.此单元对应节点数				3.此单元对应节点号
1	23	236	46		
4					
23	24	237	236		
4					
24	25	238	237		
4					
25	2	26	238		
4					
46	236	239	45		
4					
236	237	240	239		

I10: 为了求解大规模问题, 我们将所有I5输入改为I10输入

Python与abaqus二次开发

以下三个步骤:

Mesh on part – Mesh on instance

Merge Nodes

Renumber Nodes

前两者通过python完成, 后一步直接可在abaqus操作:

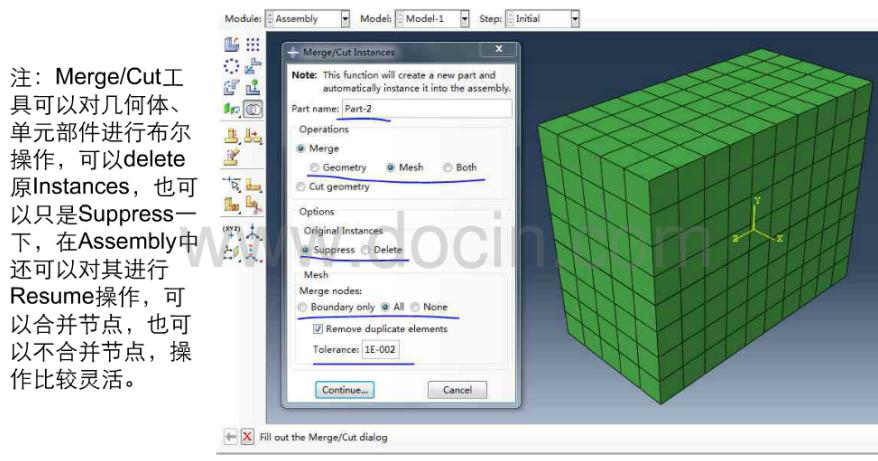
1. mdb.models['Model-1'].setValues(noPartsInputFile=ON)

2. mdb.models['Model-1'].rootAssembly.InstanceFromBooleanMerge
 (name='PartMerge',instances=('Part-1-1','Part-2-1','Part-3-2','Part-3-2-lin-
 1-2')domain=MESH,mergeNodes=ALL)
 (注：具体的models和name需要改成自己相对应的名字)

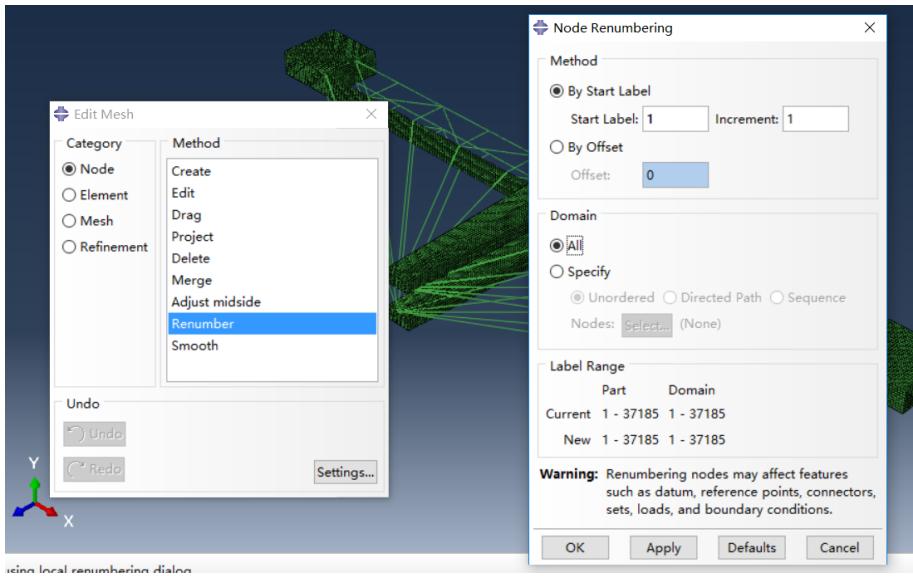
在abaqus也可操作merge命令

2.3操作步骤如下：

2.3.1在Assembly模块中，单击工具区的Merge工具，弹出Merge/Cut Instances对话框，如下图所示，设置好对话框的选项后单击Continue按钮。



3. 在abaqus中选择mesh，左栏功能最下面一个有对mesh操作的选项，选择renumber（注意需在mesh on part形式下，这就需要先按前两步生成所有instance变成一个part的操作）



前面三步分别生成merge-1文件，与merge-3文件，PreINP.py需要在相同的路径下找到这两个文件。

PreINP.py实现

基本原理就是，通过查找关键字来定位，其中具体的操作见PreINP.py源代码

```
#关键词位置信息
keylocation = [0 for x in range(line_len)]

for i in range(0, line_len-1):
    if line[i].find('*Part') != -1:          #'Part' --1
        keylocation[i] = 1
    if line[i].find('*Node') != -1:           #'Node' --2
        keylocation[i] = 2
    if line[i].find('*Element') != -1:         #'Element' --3
        keylocation[i] = 3
    if line[i].find('*Material') != -1:
        keylocation[i] = 5
    if line[i].find('*Boundary') != -1:
        keylocation[i] = 6
#继续添加关键字-----
```

0.2.2 SPR

采用SPR方法进行节点应力恢复。对于一个内部节点，若与8个8H单元相连，则有 $n = 8 \times 8 = 64$ 个高斯点用于应力恢复；对于一个边界面上的节点，则与4个单元相连，有32个高斯点用于应力恢复；同理，对于边界棱上的节点有16个，边界顶点上的节点有8个高斯点。为防止过拟合，对于边界面上的节点和内部节点采用二阶多项式进行最小二乘逼近，对于边界棱和边界顶点上的节点采用完备一阶多项式进行最小二乘逼近。以三维单元为例：

- 对二阶逼近：

$$\mathbf{A} = \begin{bmatrix} 1 & x_1 & y_1 & z_1 & x_1y_1 & y_1z_1 & z_1x_1 \\ 1 & x_2 & y_2 & z_2 & x_2y_2 & y_2z_2 & z_2x_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & y_n & z_n & x_ny_n & y_nz_n & z_nx_n \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} 1 & x & y & z & xy & yz & zx \end{bmatrix}$$

$$\mathbf{C}_{ij} = \begin{bmatrix} c_0 & c_1 & c_2 & c_3 & c_4 & c_5 & c_6 \end{bmatrix}^T, \quad \mathbf{S}_{ij} = \begin{bmatrix} \sigma_{ij}^{(1)} & \sigma_{ij}^{(2)} & \cdots & \sigma_{ij}^{(n)} \end{bmatrix}^T$$

最小二乘逼近为解系数矩阵 \mathbf{C}_{ij} ，使得： $\mathbf{AC}_{ij} = \mathbf{S}_{ij}$ ，即 $\mathbf{A}^T \mathbf{AC}_{ij} = \mathbf{A}^T \mathbf{S}_{ij}$ 。最后解 $\mathbf{X}^{(k)} \mathbf{C}_{ij} = \mathbf{S}_{ij}^{(k)}$ 为第 k 个节点上的应力。

- 对一阶逼近：

同理，但只取 $1, x, y, z$ 的项使用。

- 当连接到一个节点的所有单元具有的高斯点都不足以满足系数要求，或单元采用的为直接刚度法时，转而采用节点平均法恢复应力。这种情况常见于梁、杆、3T单元等。
- 之前尝试过采用完备二阶多项式进行逼近，然而基本上都会出现过拟合现象。于是从逼近函数里删去平方项，得到的结果相对理想。
- 算法分析

为得到每个节点对应的Patch，构建节点关系矩阵NodeRelationFlag，为NUMNP行矩阵，列数由单元类型以一般情况下够用来决定。对组

内所有单元的连接矩阵作循环，除最后两列外，之前列按顺序存储循环得到的该节点连接的单元编号。倒数第二列指示该节点共连接的单元数，最后一列指示哪个单元的1号节点对应本节点，用于提取节点坐标。之后为按节点循环，得到所连接的单元的高斯点坐标及应力情况，对其按之前所述的方法进行最小二乘逼近，得到系数矩阵。最后根据系数矩阵和本节点的坐标得到本节点的应力情况。

- 本程序对每一个单元组都采用这种思路进行节点应力恢复，并且按单元组顺序将包含节点的各应力分量输出至.OUT文件中

0.2.3 后处理

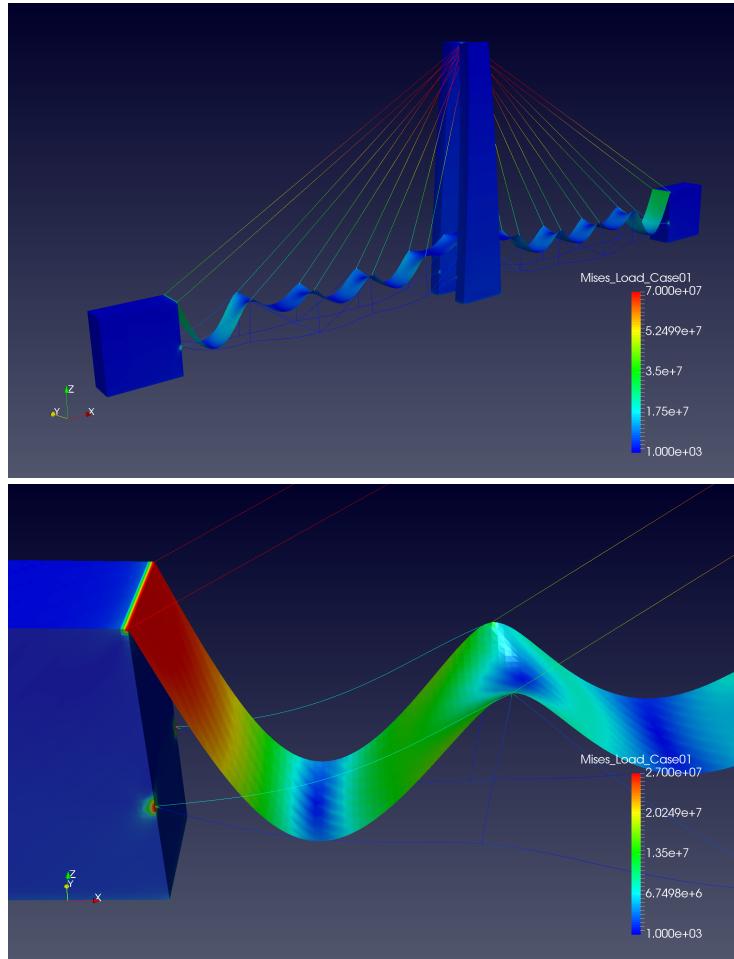
本程序后处理采用ParaView完成，在程序内包含输出为VTK文件的子程序VTKgenerate

在IND=0阶段，对VTK文件进行初始化，按VTK要求的格式写入文件头和各节点坐标。

在程序执行过程中，会将需要后处理而且要从内存中清除的数据写入3个临时文件中。分别为：

- *VTK.tmp*: 存储位移和应力恢复信息，在WRITED（位移）和PostProcessor（应力）子程序处写入；
- *VTKNode.tmp*: 存储连接矩阵信息，在PostProcessor子程序处写入；
- *VTKElTyp.tmp*: 存储单元类型信息，在ElCal子程序处写入。

最后在IND=3阶段读取临时文件中的信息，按顺序整合为单一vtk格式的输出文件。网格形式为UNSTRUCTURED GRID，格式为ASCII，所记录的数据类型为double，包含三个场量：位移场、应力场和Von Mises应力场。以下为通过ParaView渲染生成的桥梁变形及应力分布及局部细节（变形放大倍率100，着色为Von Mises应力）：



0.2.4 8H

- 算法分析

给出一个8H单元，其节点坐标矩阵为 $\mathbf{X} = \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ \vdots & \vdots & \vdots \\ x_8 & y_8 & z_8 \end{bmatrix}$ ，每条边的

方向上采用2点高斯积分，共8个高斯点，并采用三线性函数作为形函数。作前处理时，将坐标变换到 ξ, η, ζ 下求形函数的梯度，再对8个高斯点作加权求和，得到单元刚度阵：

$$N_i = \frac{1}{8}(1 + \xi_i \xi)(1 + \eta_i \eta)(1 + \zeta_i \zeta), \quad i = 1, 2, \dots, 8$$

$$\mathbf{N} = [\mathbf{N}_1 \quad \mathbf{N}_2 \quad \mathbf{N}_3 \quad \mathbf{N}_4 \quad \mathbf{N}_5 \quad \mathbf{N}_6 \quad \mathbf{N}_7 \quad \mathbf{N}_8], \quad \mathbf{N}_i = N_i \begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix}$$

$$GN = \begin{bmatrix} \frac{\partial N_1}{\partial \xi} & \frac{\partial N_2}{\partial \xi} & \cdots & \frac{\partial N_8}{\partial \xi} \\ \frac{\partial N_1}{\partial \eta} & \frac{\partial N_2}{\partial \eta} & \cdots & \frac{\partial N_8}{\partial \eta} \\ \frac{\partial N_1}{\partial \zeta} & \frac{\partial N_2}{\partial \zeta} & \cdots & \frac{\partial N_8}{\partial \zeta} \end{bmatrix}$$

$$\mathbf{B} = \nabla_s \mathbf{N} = [\mathbf{B}_1 \quad \mathbf{B}_2 \quad \mathbf{B}_3 \quad \mathbf{B}_4 \quad \mathbf{B}_5 \quad \mathbf{B}_6 \quad \mathbf{B}_7 \quad \mathbf{B}_8], \quad \mathbf{B}_i = \begin{bmatrix} B_{ix} & 0 & 0 \\ 0 & B_{iy} & 0 \\ 0 & 0 & B_{iz} \\ B_{iy} & B_{ix} & 0 \\ 0 & B_{iz} & B_{iy} \\ B_{iz} & 0 & B_{ix} \end{bmatrix}, \quad i = 1, 2, \dots, 8$$

\mathbf{B} 矩阵的各量由单元Jacobian矩阵和形函数对各坐标的导数得到:

$$\mathbf{J}^{-1} \cdot GN = \begin{bmatrix} B_{1x} & B_{2x} & \cdots & B_{8x} \\ B_{1y} & B_{2y} & \cdots & B_{8y} \\ B_{1z} & B_{2z} & \cdots & B_{8z} \end{bmatrix}, \quad \mathbf{J} = GN \cdot \mathbf{X}^T$$

\mathbf{D} 矩阵由本构关系得到, 在此认为材料是各向同性的:

$$\sigma_{ij} = 2G\varepsilon_{ij} + \lambda\varepsilon_{kk}\delta_{ij}$$

$$\Rightarrow \mathbf{D} = \begin{bmatrix} 2G + \lambda & \lambda & \lambda \\ \lambda & 2G + \lambda & \lambda \\ \lambda & \lambda & 2G + \lambda \\ & & 2G \\ & & & 2G \\ & & & & 2G \end{bmatrix}, \quad \text{s.t. } \boldsymbol{\sigma} = \mathbf{D}\boldsymbol{\varepsilon}, \text{ where } \boldsymbol{\varepsilon} =$$

$$\begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \varepsilon_{xy} \\ \varepsilon_{yz} \\ \varepsilon_{zx} \end{bmatrix}, \boldsymbol{\sigma} = \begin{bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \tau_{xy} \\ \tau_{yz} \\ \tau_{zx} \end{bmatrix}.$$

$$\mathbf{K}^e = \sum_{i=1}^2 \sum_{j=1}^2 \sum_{k=1}^2 w_i w_j w_k \mathbf{B}^T(\xi_i, \eta_j, \zeta_k) \mathbf{D} \mathbf{B}(\xi_i, \eta_j, \zeta_k)$$

生成单元刚度阵通过Location Matrix进行组装，得到全局的刚度阵，再解 $\mathbf{Kd} = \mathbf{f}$.

- 算法实现

位移计算部分按照变换将坐标变换到 $[-1, 1] \times [-1, 1] \times [-1, 1]$ 后即可得到形函数和各导数。取得高斯点位置，代入 \mathbf{D} 矩阵即可以产生单元刚度阵，调用 COLHT 和 ADDBAN 生成总刚度矩阵并求解。

SPR 部分，在前处理时保存单元连接矩阵到临时文件中，在后处理中调取，计算每个节点连接的单元数量和编号，以及用于恢复节点位置的信息，生成为 NodeRelationFlag 数组。对每个节点，根据该数组选取逼近的阶次，计算 \mathbf{A} 、 \mathbf{S} 矩阵，并传入最小二乘子程序 LeastSquare 中，得到系数数组，并代入节点位置信息，得到恢复的节点应力。

- 算法测试

Patch Test 选取了 7 单元模型和 8 单元模型进行测试，测试形状为单位立方体， $E = 1000$, $\nu = 0.25$ ，通过内部确定 1 个或 8 个点来将其分割为 8 个或 7 个单元。施加线性位移场：

$$u_x = 0.001x, u_y = 0.002y, u_z = 0.003z$$

得到各点需要施加的外力，并采用最少的边界条件对单元进行测试。测试结果如 data/8H 中各输入输出文件所示。

测试结果表明，对于线性场，输出误差在机器 ε 量级，单元设计能精确重构线性位移场，为一阶收敛的。

对SPR的测试，其输出结果误差也在机器 ϵ 量级，因而从算法设计的角度来说是合理的。

0.2.5 Beam

Basic principles

For a beam element, the equation for a single beam element is quite simple, for the reason that the calculation of the element stiffness matrix doesn't require numerical integral. It can be done manually. The stiffness matrix for a beam element is as follow.

$$K^{(e)} = \begin{bmatrix} \frac{EA}{l} & & & & & & & \\ & \frac{12EI_z}{l^3} & & & & & & \\ 0 & & \frac{12EI_y}{l^3} & & & & & \\ 0 & 0 & & \frac{GJ_z}{l} & & & & \\ 0 & 0 & & 0 & \frac{4EI_Y}{l} & & & \\ 0 & 0 & & \frac{-6EI_z}{l^2} & 0 & \frac{4EI_Y}{l} & & \\ 0 & 0 & & \frac{6EI_z}{l^2} & 0 & 0 & \frac{4EI_z}{l} & \\ \hline -\frac{EA}{l} & 0 & 0 & 0 & 0 & 0 & EA & \\ 0 & -\frac{12EI_z}{l^3} & 0 & 0 & 0 & \frac{-6EI_z}{l^2} & 0 & \frac{12EI_z}{l^3} \\ 0 & 0 & -\frac{12EI_y}{l^3} & 0 & \frac{6EI_y}{l^2} & 0 & 0 & \frac{12EI_y}{l^3} \\ 0 & 0 & 0 & \frac{-GJ_x}{l} & 0 & 0 & 0 & \frac{GJ_z}{l} \\ 0 & 0 & \frac{-6EI_y}{l^2} & 0 & \frac{2EI_y}{l} & 0 & 0 & \frac{6EI_y}{l^2} \\ 0 & 0 & \frac{6EI_z}{l^2} & 0 & 0 & \frac{2EI_z}{l} & 0 & \frac{-6EI_z}{l^2} \end{bmatrix} \quad \text{symmetry}$$

$$\begin{aligned} F^{(e)} &= \{F_{N_i} \ F_{Q_{iy}} \ F_{Q_{iz}} \ M_{ix} \ M_{iy} \ M_{iz} \ F_{N_j} \ F_{Q_{jy}} \ F_{Q_{jz}} \ M_{jx} \ M_{jy} \ M_{jz}\}^T \\ \mathbf{d}^{(e)} &= \{u_i \ v_i \ w_i \ \theta_{ix} \ \theta_{iy} \ \theta_{iz} \ u_j \ v_j \ w_j \ \theta_{jx} \ \theta_{jy} \ \theta_{jz}\}^T \end{aligned}$$

In all, we can get the equation for the element itself.

$$\mathbf{K}^{(e)} \mathbf{d}^{(e)} = \mathbf{F}^{(e)} \quad (1)$$

The process of assembly and solution of the linear equation is just the same as the former elements.

ATTENTION The stiffness matrix for beams point to arbitrary directions needs the translation matrix to do the coordinates translation.

$$\mathbf{K}^{(e)'} = \mathbf{T}^T \mathbf{K}^{(e)} \mathbf{T} \quad (2)$$

Programming procedures

The most difficult part for the programming is the generation of the ID array without interfering with other elements. The structure of the STAP90 program is very terrible for this kind of modification, given the reason that the original program read the element controlling messages after the nodes, making it complex for the dealt for nodes. The only choice left for me is whether changing the whole blueprint for the entire STAP 90 program or just writing the individual input or output subroutine for the beam. In terms of the deadline requirements, I choose the latter, leaving the former mission to the following weeks. As you can see in the source files, the input and output subroutine for the beam is in the 'BEAM.f90'.

Input file format

For controlling messages of nodals:

列	变量	意义
1-5	N	节点号 ($1 \leq N \leq \text{NUMNP}$)。
6-10	ID(1,N)	x-平动方向边界条件代码。 0-自由; 1-固定。
11-15	ID(2,N)	y-平动方向边界条件代码。 0-自由; 1-固定。
16-20	ID(3,N)	z-平动方向边界条件代码。 0-自由; 1-固定。
21-25	ID(4,N)	x-转动方向(转角)边界条件代码。 0-自由; 1-固定。
26-30	ID(5,N)	y-转动方向(转角)边界条件代码。 0-自由; 1-固定。
31-35	ID(6,N)	z-转动方向(转角)边界条件代码。 0-自由; 1-固定。
36-45	X(N)	x-坐标。
46-55	Y(N)	y-坐标。
56-65	Z(N)	z-坐标。

For information about the loads:

列。	变量。	意义。
1-5。	NOD。	集中载荷作用的节点号 ($1 \leq NOD \leq NUMNP$)。
6-10。	IDIRN。	载荷作用方向 (1-x 方向力, 2-y 方向力, 3-z 方向力, 4-x 方向力矩, 5-y 方向力矩, 6-z 方向力矩)。
11-20。	FLOAD。	载荷值。

For information about the materials:

列。	变量。	意义。
1-5。	N。	材料/截面性质组号 ($1 \leq N \leq NPAR(3)$)。
6-15。	E(N)。	杨氏模量。
16-25。	G(N)。	剪切模量。
26-35。	AREA(N)。	截面面积。
36-45。	I _x (N)。	对局部 x 轴截面二阶惯性矩 (形心主轴系)。
46-55。	I _y (N)。	对局部 y 轴截面二阶惯性矩 (形心主轴系)。
56-65。	I _z (N)。	对局部 z 轴截面二阶惯性矩 (形心主轴系)。
66-75。	J _x (N)。	对局部 x 轴的截面极矩 (形心主轴系)。
76-85。	J _y (N)。	对局部 y 轴的截面极矩 (形心主轴系)。
86-95。	J _z (N)。	对局部 z 轴的截面极矩 (形心主轴系)。

For information about the element connectivity:

列。	变量。	意义。
1-5。	M。	单元号 ($1 \leq M \leq NPAR(2)$)。
6-10。	II。	单元左端节点号 ($1 \leq II \leq NUMNP$)。
11-15。	JJ。	单元右端节点号 ($1 \leq JJ \leq NUMNP$)。
16-20。	MTYP。	本单元的材料/截面性质组号。
21-25。	KG。	单元自动生成增量。

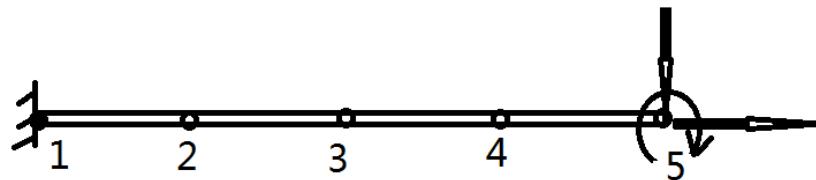
The rate of convergence

For the beam, the FEM guarantees the continuity and the completeness. The completeness will be verified in the patchtest. For continuity, the method for the approximation of the function is 3-order Hermite interpolation, maintaining the continuity for both the displacement and the angle. It is obvious that the FEM will give the exact answer at nodes if we give concentration forces or momentum at the nodes. For the distribution forces, the FEM will not give the exact results. The reason for this is that the order of the moment is at least 2 for the distribution forces. The results will

be better and will converge to the exact solution with the mesh refined to infinite decimal. The convergence rate for the displacement is at the order of 2, while the rate for the energy norm is at the order of 1.

Patchtest and the results

I use an example for the patchtest. The situation of the load and the structure is as follow.



The input files and the results for the patchtest in the folder for the beam.

0.2.6 Shell

0.2.7 半带宽优化

简介

首先需要注意的是，半带宽优化已经被证明是一个NP完全问题，所以无法在多项式时间得到最优解。现在的算法主要是近似的优化算法，目前主流的算法有CM算法，GPS算法，tabu搜索算法，GRASP算法，模拟冷却算法等。由于程序的最初设计以速度优先，故在选择的时候排除了耗时巨大的模拟冷却算法，而是采用了偏于经验类的算法。加之使用FORTRAN语言实现图结构困难，所以半带宽部分选择在stap90程序中实现最为原始的CM算法。而对于更加强力一点的算法，则选择用C++予以实现GRASP算法的部分。

算法介绍 a)CM算法

CM算法全称Cuthill-McKee算法在这两人于1969年名为Reducing the band-width of sparse symmetric matrices的论文中提出，是最先被提出的半带宽优化算法。其主要实现较为简单，主要就是通过一个广度优先搜索，每次把入度最小的点标号，并把该点的邻居加入队列中，通过循环得到最终的

编号。

b)GRASP算法

GRASP算法是在2004年由西班牙Valencia大学的Estefania Piñana教授等人在其文章GRASP and path relinking for matrix bandwidth minimization中提出的。其吸取了前人(CM, GPS, Tabu)的优点，在试验中有着非常优秀的表现。其主要内容分为三个阶段，构建阶段，提升阶段和路径重连阶段。构建阶段提出了5中不同的方案，在保证优质初始解的同时也考虑了其随机性，避免其落入局部最优点。在大作业的C++部分实现中，我才用了C1部分，主要原因还是比较易于操作。提升阶段参考于tabu算法的交换节点编号的部分，不过为了提升性能，GRASP算法仅仅考虑在交换时会改变最大带宽以及最大带宽数的个数的点。路径重连部分适用于极大规模问题，而且需要有多局部极值，并不适用于本问题，故忽略。

算法实现

在stap90中，如果要使用半带宽优化，为了简便操作，我选择了在输入文件中加入部分信息，也就是在输入ID之后就输入整体的连接矩阵。再次说明一下这一点的必要性，因为半带宽优化关心的是图的整体结构，所以整体的连接矩阵必不可少，而新产生的ID矩阵又必须在组装过程中发挥作用，所以选择了在输入原先的ID之后直接输入连接矩阵。在具体实现中，因为FORTRAN没有可变数组的支持，我先是构建了一个node模板和一个list模板，具体接口请详见list.f90与node.f90。通过循环将与自由度i有关的所有自由度都放在了链表数组lists(i)之中，然而由于链表的搜索功能是其软肋，所以在实现搜索的时候还是比较麻烦的。而C++的部分则是比较轻松，同时实现了类似的接口完成了算法实现。Stap90的实现位于solvermode.f90文件中的bdopt函数，C++的实现位于BandwidthOpt文件夹中。

在实验中，我发现由于决定skyline算法时间的主要因素是nwk，也就是skyline存储中的元素个数，而不是最大带宽，加上桥问题的主要问题在于其部分元素带宽很大，而大多数很小，导致使用了优化虽然可以使带宽从1492降至1356，但是总元素个数增加，时间反而从12s增长至18s，得不偿失。这说明半带宽优化并不适用于所有问题。

0.2.8 稀疏存储求解器

简介

首先对稀疏矩阵进行一些简单介绍。稀疏矩阵主要指非零元素为 $O(n)$ 的矩阵。其没有必要使用以往的 $O(n^2)$ 求解，而是可以利用特殊的储存方式，大大缩减存储空间。与此同时，因为实际上的计算大多仅仅与非零元素有关，所以有算法可以将求解线性方程 $Lx = y$ 的时间降至 $O(\text{flop})$ (其中flop为非零元素个数)大大降低其运算时间，是求解有稀疏性质的线性系统的一把利器。本次采用intel MKL中集成的pardiso求解器，因为其为市面上速度最快的求解器。矩阵的存储方式采用默认的CSC格式。值得一提的是，相比于stap90中原有的skyline求解器，pardiso在空间上有着劣势。因为skyline求解器可以在原地进行 L^TDL 分解，而CSC存储的矩阵，由于不知道分解后非零元素的位置，需要在过程中被迫开一份 $O(n^2)$ 的空间。

实现

由于具体的函数调用都是相同的MKL中的pardiso，所以实现中更重要的一些小细节，而这些小细节就是保证本组成为性能第一的根本。实现中最主要的部分是如何构建CSC存储所需要的3个数组。当时考虑了是否要利用stap90原有的maxa来改写得到CSC格式，但是仔细思考后，我认为那种方式在初始的时候开辟了过多的内存，所以舍弃，而是采用直接从连接矩阵入手的想法。而对于这种想法，我采用了与半带宽优化同样的策略，使用链表得到所欲求的数组。但是在后续的优化中，我发现大量的指针不利于编译器优化，在大规模问题中大大拖累了时间，所以毅然决定改为使用可分配数组。但是由于需要整体的大小，只好牺牲额外的空间，不过好在牺牲应该不大。另外一点，对于稀疏矩阵来说，最重要的性能参数应当是稀疏度，所以我在初步组装之后设计了一种 $O(n)$ 数量级的crop算法，将组装时加入的0元素去掉，从而将稀疏度降低到0.0036最后一个设计上的优势就是我们发现，windows的磁盘设置导致我们无法一次开辟存储超过2G的数组，所以当规模非常大的时候，使用原有的memalloc已经无法满足Job - 4的需求，所以我通过使用两个可分配数组来存放CSC中大小为nwk的数组以克服这个问题。

实验效果

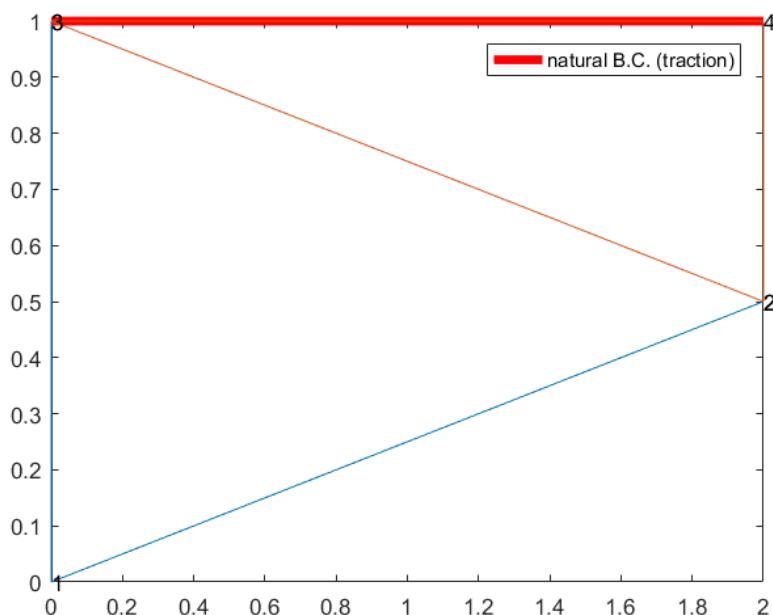
在测试中，我们以 $Job - 10.25s$ 比 $Job - 23s, Job - 330s$ 的成绩，比第二名在每一项都快至少一倍轻松得到了第一名。

感受

在本组的分配中，我主动选择了速度优化这一个方面，在时间的过程中我学到了很多东西。最重要的两点是：第一点，是在优化并不完全是设计算法，而是一个对整个问题的把控。第二点，对于编写软件的人，硬件知识也是必须的。我曾经因为没有把编译器调到x64而导致内存被限制在了4G以内而困扰了很久，包括前文提到的windows不能开2G的数组都是一些常见的硬件知识。最后，必须要说能够看到本组的程序跑 $Job - 1$ 从最开始的40s到0.2s真的是非常自豪，非常激动。

0.3 其他单元

0.3.1 3T



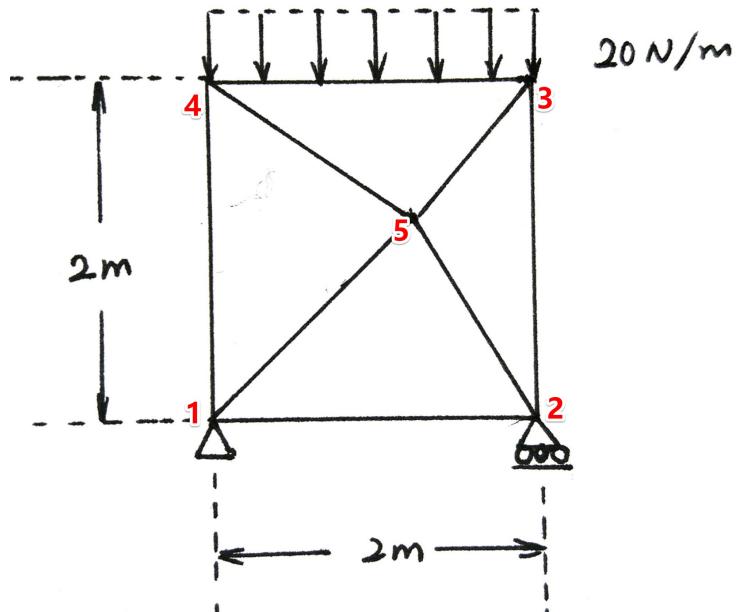
Answer in Matlab

Stress at Gauss Points					
	x-coord	y-coord	s_xx	s_yy	s_xy
Element 1					
33	1.333333	0.500000	-6.380783	-1.914235	-38.404804
34	0.333333	0.750000	-6.380783	-1.914235	-38.404804
35	0.333333	0.250000	-6.380783	-1.914235	-38.404804
Element 2					
39	1.666667	0.916667	12.761565	-19.202402	-3.190391
40	0.666667	0.916667	12.761565	-19.202402	-3.190391
41	1.666667	0.666667	12.761565	-19.202402	-3.190391
42					

Answer in Staq90

STRESS CALCULATIONS FOR ELEMENT GROUP 1					
ELEMENT	X-CORRD	Y-CORRD	STRESS_XX	STRESS YY	STRESS_XY
1	0.133333E+01	0.500000E+00	-0.638078E+01	-0.191423E+01	-0.384048E+02
1	0.333333E+00	0.250000E+00	-0.638078E+01	-0.191423E+01	-0.384048E+02
1	0.333333E+00	0.750000E+00	-0.638078E+01	-0.191423E+01	-0.384048E+02
2	0.166667E+01	0.916667E+00	0.127616E+02	-0.192024E+02	-0.319039E+01
2	0.166667E+01	0.666667E+00	0.127616E+02	-0.192024E+02	-0.319039E+01
2	0.666667E+00	0.916667E+00	0.127616E+02	-0.192024E+02	-0.319039E+01

0.3.2 PatchTest3T



We constrain the node 1 in xy direction and node 2 in y direction. With the pressure added on the top surface of plate, we get the constant stress that

$$s_{xx} = 0$$

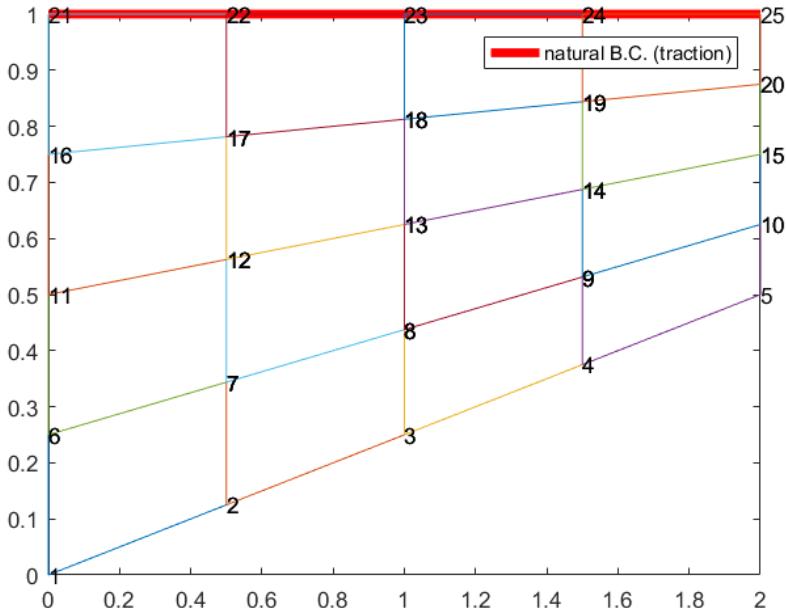
$$s_{yy} = -20$$

$$s_{xy} = 0$$

Answer in Stap90

	ELEMENT	X-CORRD	Y-CORRD	STRESS_XX	STRESS_YY	STRESS_XY
96	1	0.211325E+00	0.211325E+00	-0.177636E-14	-0.200000E+02	0.305421E-15
97	1	0.788675E+00	0.211325E+00	-0.266454E-14	-0.200000E+02	0.916262E-15
98	1	0.211325E+00	0.788675E+00	0.888178E-15	-0.200000E+02	-0.122168E-14
99	1	0.788675E+00	0.788675E+00	0.000000E+00	-0.200000E+02	0.000000E+00
100	1	0.788675E+00	0.788675E+00	0.000000E+00	0.000000E+00	0.000000E+00

0.3.3 4Q



Answer in Matlab

Element 1					

81	x-coord	y-coord	s_xx	s_yy	s_xy
82	0.105662	0.077851	-152.576347	-39.555167	-61.888824
83	0.394338	0.146207	-173.071977	-26.858861	-24.127911
84	0.105662	0.218376	-101.740111	-24.304296	-58.622060
85	0.394338	0.276315	-118.165710	-10.386981	-20.599604
Element 2					

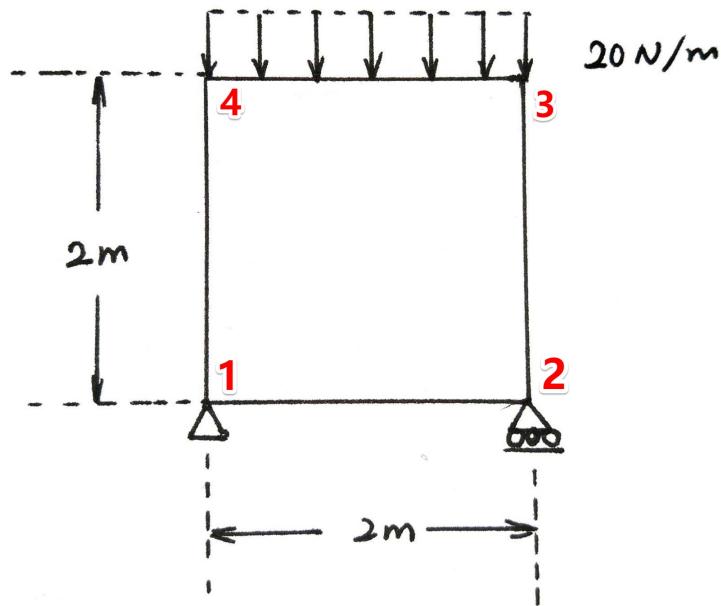
88	x-coord	y-coord	s_xx	s_yy	s_xy
89	0.605662	0.196247	-102.830368	-2.425610	-54.956151
90	0.894338	0.264603	-132.351283	-24.564884	-16.584438
91	0.605662	0.318730	-61.611597	9.940022	-56.939445
92	0.894338	0.376669	-87.301176	-11.049852	-18.752082
Element 3					

95	x-coord	y-coord	s_xx	s_yy	s_xy
96	1.105662	0.314643	-51.452830	-7.077900	-37.005831
97	1.394338	0.382999	-68.908892	-16.345880	-12.801393
98	1.105662	0.419084	-29.269984	-0.423046	-37.510824
99	1.394338	0.477023	-44.268461	-8.953750	-13.362333

Answer in Stapp90

177	ELEMENT	X-CORRD	Y-CORRD	STRESS_XX	STRESS YY	STRESS_XY
178	1	0.105662E+00	0.778513E-01	-0.152576E+03	-0.395552E+02	-0.618888E+02
179	1	0.394338E+00	0.146207E+00	-0.173072E+03	-0.268589E+02	-0.241279E+02
180	1	0.105662E+00	0.218376E+00	-0.101740E+03	-0.243043E+02	-0.586221E+02
181	1	0.394338E+00	0.276315E+00	-0.118166E+03	-0.103870E+02	-0.205996E+02
182	2	0.605662E+00	0.196247E+00	-0.102830E+03	-0.242561E+01	-0.549562E+02
183	2	0.894338E+00	0.264603E+00	-0.132351E+03	-0.245649E+02	-0.165844E+02
184	2	0.605662E+00	0.318730E+00	-0.616116E+02	0.994002E+01	-0.569394E+02
185	2	0.894338E+00	0.376669E+00	-0.873012E+02	-0.110499E+02	-0.187521E+02
186	3	0.110566E+01	0.314643E+00	-0.514528E+02	-0.707790E+01	-0.370058E+02
187	3	0.139434E+01	0.382999E+00	-0.689089E+02	-0.163459E+02	-0.128014E+02
188	3	0.110566E+01	0.419084E+00	-0.292700E+02	-0.423046E+00	-0.375108E+02
189	3	0.139434E+01	0.477023E+00	-0.442685E+02	-0.895375E+01	-0.133623E+02

0.3.4 PatchTest4Q



We constrain the node 1 in xy direction and node 2 in y direction. With the pressure added on the top surface of plate, we get the constant stress that

$$s_{xx} = 0$$

$$s_{yy} = -20$$

$$s_{xy} = 0$$

Answer in Step90

	ELEMENT	X-CORRD	Y-CORRD	STRESS_XX	STRESS_YY	STRESS_XY
96	1	0.211325E+00	0.211325E+00	-0.177636E-14	-0.200000E+02	0.305421E-15
97	1	0.788675E+00	0.211325E+00	-0.266454E-14	-0.200000E+02	0.916262E-15
98	1	0.211325E+00	0.788675E+00	0.888178E-15	-0.200000E+02	-0.122168E-14
99	1	0.788675E+00	0.788675E+00	0.000000E+00	-0.200000E+02	0.000000E+00
100	1					

0.3.5 8Q

0.3.6 9Q

9Q的原理与4Q类似，均是用 $Kd = f$ 进行求解，只是其组装单元刚度阵所使用的形函数不同

$$N_i = \begin{cases} \frac{1}{4}\xi_i\eta_i\xi\eta(1 + \xi_i\xi)(1 + \eta_i\eta) & (i = 1, 2, 3, 4) \\ \frac{1}{2}\eta_i\eta(1 - \xi^2)(1 + \eta_i\eta) & (i = 5, 7) \\ \frac{1}{2}\xi_i\xi(1 - \eta^2)(1 + \xi_i\xi) & (i = 6, 8) \\ (1 - \xi^2)(1 - \eta^2) & (i = 9) \end{cases}$$

PatchTest所使用的构型仍为4Q使用的Patchtest方案，即约束左下角两个自由度的位移，以及右下角Y自由度的位移，在顶部施加向下为20N/m的均布力。

其结果如下：

NODE	X-DISPLACEMENT	Y-DISPLACEMENT	Z-DISPLACEMENT	X-ROTATION	Y-ROTATION	Z-ROTATION
1	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
2	0.600000E-06	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
3	0.600000E-06	-0.200000E-05	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
4	0.173949E-13	-0.200000E-05	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
5	0.300000E-06	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
6	0.600000E-06	-0.100000E-05	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
7	0.300000E-06	-0.200000E-05	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
8	-0.105922E-13	-0.100000E-05	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
9	0.300000E-06	-0.100000E-05	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00

ELEMENT	X-CORRD	Y-CORRD	STRESS_XX	STRESS YY	STRESS_XY
1	-0.861136E+00	-0.861136E+00	-0.653980E-07	-0.300000E+02	-0.214415E-06
1	-0.861136E+00	-0.339981E+00	0.289206E-06	-0.300000E+02	-0.550864E-07
1	-0.861136E+00	0.339981E+00	0.232497E-06	-0.300000E+02	-0.204030E-06
1	-0.861136E+00	0.861136E+00	-0.209034E-06	-0.300000E+02	-0.591672E-06
1	-0.339981E+00	-0.861136E+00	-0.186423E-07	-0.300000E+02	-0.846520E-07
1	-0.339981E+00	-0.339981E+00	0.194094E-06	-0.300000E+02	-0.217484E-07
1	-0.339981E+00	0.339981E+00	-0.477117E-07	-0.300000E+02	-0.805520E-07
1	-0.339981E+00	0.861136E+00	-0.631110E-06	-0.300000E+02	-0.233595E-06
1	0.339981E+00	-0.861136E+00	-0.186423E-07	-0.300000E+02	0.846521E-07
1	0.339981E+00	-0.339981E+00	0.194094E-06	-0.300000E+02	0.217484E-07
1	0.339981E+00	0.339981E+00	-0.477117E-07	-0.300000E+02	0.805520E-07
1	0.339981E+00	0.861136E+00	-0.631110E-06	-0.300000E+02	0.233595E-06
1	0.861136E+00	-0.861136E+00	-0.653980E-07	-0.300000E+02	0.214415E-06
1	0.861136E+00	-0.339981E+00	0.289206E-06	-0.300000E+02	0.550864E-07
1	0.861136E+00	0.339981E+00	0.232497E-06	-0.300000E+02	0.204030E-06
1	0.861136E+00	0.861136E+00	-0.209034E-06	-0.300000E+02	0.591672E-06

0.3.7 4T

类似于3T单元，在三维Simplex坐标系中，可以用相对体积来作为高斯点的确定依据。指定四个节点的编号顺序为，当右手四指绕底面三顶点方向旋转时，拇指指向第四顶点方向。有：

|||||| HEAD =====

$$N_i = \frac{1}{6V} (a_i + b_i x + c_i y + d_i z)$$

作 $\Lambda = \begin{bmatrix} 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \\ 1 & x_3 & y_3 & z_3 \\ 1 & x_4 & y_4 & z_4 \end{bmatrix}$, 有: $V = \frac{1}{6} |\Lambda|$, 且

$$a_1 = \begin{bmatrix} x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \\ x_4 & y_4 & z_4 \end{bmatrix}, b_1 = - \begin{bmatrix} 1 & y_2 & z_2 \\ 1 & y_3 & z_3 \\ 1 & y_4 & z_4 \end{bmatrix}, c_1 = \begin{bmatrix} 1 & x_2 & z_2 \\ 1 & x_3 & z_3 \\ 1 & x_4 & z_4 \end{bmatrix}, d_1 = - \begin{bmatrix} 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \\ 1 & x_4 & y_4 \end{bmatrix}$$

对2,3,4其它各项按 Λ 行列式的代数余子式推出。

类似于8H单元, 作B矩阵为 $B = \nabla_s N$, 其每项元素均为由节点坐标决定的常数。

对单元刚度阵进行组装, 最后解总刚度阵方程, 思路同8H。分片测试采用立方体, 分割为四个顶点对应的四面体和中心一个正四面体共5个单元, 执行C类分片测试, 结果通过, 输入输出文件如Samples文件夹内所示。

|||||| dedcc6447c9449ac1dd2040a1e7b5f4a73482371

0.3.8 铁木辛柯梁

基本原理

与欧拉梁不同的是, 铁木辛柯梁在转角和挠度上同时插值, 这样就考虑了剪切的影响。铁木辛柯梁的刚度矩阵分为剪切部分和弯曲部分, 为了避免剪切锁闭, 剪切部分采用减缩积分。在一次插值的情况下, 剪切部分采用1点高斯积分。铁木辛柯梁的难点还是三维情况下刚度阵的组装, 这里包含了材料力学意义上的扭转, 铁木辛柯梁意义上的弯曲, 和普通的简单拉伸。我们把这些刚度元素组装到合适的位置。铁木辛柯梁的刚度阵如下图。其中剪切部分有一个截面修正因子, 这里和课上讲的修整因子成倒数关系。取

$$k = \frac{10(1+\nu)}{12+11\nu}$$

$$K^{(e)} = \begin{bmatrix} \frac{EA}{l} & & & & & & & & \\ 0 & \frac{GAk}{l} & & & & & & & \\ 0 & 0 & \frac{GAk}{l} & & & & & & \\ 0 & 0 & 0 & \frac{GJ_z}{l} & & & & & \\ 0 & 0 & -\frac{GAk}{2} & 0 & \frac{EI_y}{l} + \frac{GAkl}{4} & & & & \\ 0 & \frac{GAk}{2} & 0 & 0 & 0 & \frac{EI_z}{l} + \frac{GAkl}{4} & & & \\ \hline -\frac{EA}{l} & 0 & 0 & 0 & 0 & 0 & \frac{EA}{l} & & \\ 0 & -\frac{GAk}{l} & 0 & 0 & 0 & -\frac{GAk}{2} & 0 & \frac{GAk}{l} & \\ 0 & 0 & -\frac{GAk}{l} & 0 & \frac{GAk}{2} & 0 & 0 & \frac{GAk}{l} & \\ 0 & 0 & 0 & -\frac{GJ_x}{l} & 0 & 0 & 0 & 0 & \frac{GJ_z}{l} \\ 0 & 0 & -\frac{GAk}{2} & 0 & \frac{EI_y}{l} + \frac{GAkl}{4} & 0 & 0 & \frac{GAk}{2} & 0 & \frac{EI_y}{l} + \frac{GAkl}{4} \\ 0 & \frac{GAk}{2} & 0 & 0 & 0 & -\frac{EI_z}{l} + \frac{GAkl}{4} & -\frac{GAk}{2} & 0 & 0 & \frac{EI_z}{l} + \frac{GAkl}{4} \end{bmatrix}$$

symmetry

载荷以及右端力向量如图。

$$\begin{aligned} F^{(e)} &= \{F_{N_i}, F_{Q_{iy}}, F_{Q_{iz}}, M_{ix}, M_{iy}, M_{iz}; F_{N_j}, F_{Q_{jy}}, F_{Q_{jz}}, M_{jx}, M_{jy}, M_{jz}\}^T \\ \mathbf{d}^{(e)} &= \{\mu_i, v_i, w_i, \theta_{ix}, \theta_{iy}, \theta_{iz}; u_j, v_j, w_j, \theta_{jx}, \theta_{jy}, \theta_{jz}\}^T \end{aligned}$$

输入输出

For controlling messages of nodals:

列	变量	意义
1-5	N	节点号 ($1 \leq N \leq \text{NUMNP}$)。
6-10	ID(1,N)	x-平动方向边界条件代码。 0-自由; 1-固定。
11-15	ID(2,N)	y-平动方向边界条件代码。 0-自由; 1-固定。
16-20	ID(3,N)	z-平动方向边界条件代码。 0-自由; 1-固定。
21-25	ID(4,N)	x-转动方向(转角)边界条件代码。 0-自由; 1-固定。
26-30	ID(5,N)	y-转动方向(转角)边界条件代码。 0-自由; 1-固定。
31-35	ID(6,N)	z-转动方向(转角)边界条件代码。 0-自由; 1-固定。
36-45	X(N)	x-坐标。
46-55	Y(N)	y-坐标。
56-65	Z(N)	z-坐标。

For information about the loads:

列。	变量。	意义。
1-5。	NOD。	集中载荷作用的节点号 ($1 \leq NOD \leq NUMNP$)。
6-10。	IDIRN。	载荷作用方向 (1-x 方向力, 2-y 方向力, 3-z 方向力, 4-x 方向力矩, 5-y 方向力矩, 6-z 方向力矩)。
11-20。	FLOAD。	载荷值。

For information about the materials:

列。	变量。	意义。
1-5。	N。	材料/截面性质组号 ($1 \leq N \leq NPAR(3)$)。
6-15。	E(N)。	杨氏模量。
16-25。	G(N)。	剪切模量。
26-35。	AREA(N)。	截面面积。
36-45。	I _x (N)。	对局部 x 轴截面二阶惯性矩 (形心主轴系)。
46-55。	I _y (N)。	对局部 y 轴截面二阶惯性矩 (形心主轴系)。
56-65。	I _z (N)。	对局部 z 轴截面二阶惯性矩 (形心主轴系)。
66-75。	J _x (N)。	对局部 x 轴的截面极矩 (形心主轴系)。
76-85。	J _y (N)。	对局部 y 轴的截面极矩 (形心主轴系)。
86-95。	J _z (N)。	对局部 z 轴的截面极矩 (形心主轴系)。

For information about the element connectivity:

列。	变量。	意义。
1-5。	M。	单元号 ($1 \leq M \leq NPAR(2)$)。
6-10。	II。	单元左端节点号 ($1 \leq II \leq NUMNP$)。
11-15。	JJ。	单元右端节点号 ($1 \leq JJ \leq NUMNP$)。
16-20。	MTYP。	本单元的材料/截面性质组号。
21-25。	KG。	单元自动生成增量。

分片试验结果收敛性分析

需要注意的是，铁木辛柯梁对一般截面并没有精确解，但是对于圆截面还是可以分析收敛性。误差的能量范数是一阶收敛的。并且减缩积分能获得更精确的结果。最重要的是，通过铁木辛柯梁，我对有限元中减缩积分的方法有了新的理解。铁木辛柯梁只是一种梁模型，通过铁木辛柯梁和欧拉梁的编写，我发现只要给定一种弯曲模型，就可以很轻松地算出这种模型的有限元刚度阵，进而进行计算。

```

TIMOSHENKE
0010
      5   1   1   1
      1   1   1   1   1   1   1   0.0   0.0   0.0   0
      2   0   0   0   0   0   0   0.0   0.5   0.0   0
      3   0   0   0   0   0   0   0.0   1.0   0.0   0
      4   0   0   0   0   0   0   0.0   1.5   0.0   0
      5   0   0   0   0   0   0   0.0   2.0   0.0   0
      1   2
      5   3   -1.0E3
      5   1   0.5E2
     12   4   1
     1   2.0E11   7.0E10   3.0E1   1.0E2   1.0E2   1.0E2   2.0E2   2.0E2
     1   1   2   1   0
     2   2   3   1   0
     3   3   4   1   0
     4   4   5   1   0
stop

DISPLACEMENTS

NODE   X-DISPLACEMENT   Y-DISPLACEMENT   Z-DISPLACEMENT   X-ROTATION   Y-ROTATION   Z-ROTATION
  1   0.00000E+00   0.00000E+00   0.00000E+00   0.00000E+00   0.00000E+00   0.00000E+00
  2   0.144754E-10   0.00000E+00   -0.289509E-09   -0.437500E-10   0.00000E+00   -0.218750E-11
  3   0.298884E-10   0.00000E+00   -0.597768E-09   -0.750000E-10   0.00000E+00   -0.375000E-11
  4   0.459263E-10   0.00000E+00   -0.918527E-09   -0.937500E-10   0.00000E+00   -0.468750E-11
  5   0.622768E-10   0.00000E+00   -0.124554E-08   -0.100000E-09   0.00000E+00   -0.500000E-11

```

0.3.9 Plate

0.3.10 Formula Derivation

In the stamp90, I create two kinds of plate elements, both of which are for thick plate.

The difference between them is the number of the nodes in each element. The first type has only 4 nodes each element, the same as 4Q, while the other has 8. I would explain the reason below.

Plate4Q

The Plate4Q element is almost the same as 4Q. We use isoparametric element by represent both the transformation of coordination and the w , β_x , β_y with the same bunch of bilinear shape function.

The only disparity is the formula of strain and stress, as shown below.

As for strain:

$$\begin{pmatrix} \epsilon_{xx} \\ \epsilon_{yy} \\ \gamma_{xy} \end{pmatrix} = -z \begin{pmatrix} \frac{\partial \beta_x}{\partial x} \\ \frac{\partial \beta_y}{\partial y} \\ \frac{\partial \beta_x}{\partial y} + \frac{\partial \beta_y}{\partial x} \end{pmatrix} \quad (3)$$

$$\begin{pmatrix} \gamma_{xz} \\ \gamma_{yz} \end{pmatrix} = \begin{pmatrix} \frac{\partial w}{\partial x} - \gamma_x \\ \frac{\partial \beta_y}{\partial y} - \gamma_y \end{pmatrix} \quad (4)$$

And for stress:

$$\begin{pmatrix} \tau_{xx} \\ \tau_{yy} \\ \tau_{xy} \end{pmatrix} = -z \frac{E}{1-\nu^2} \begin{pmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{pmatrix} \begin{pmatrix} \epsilon_{xx} \\ \epsilon_{yy} \\ \gamma_{xy} \end{pmatrix} \quad (5)$$

$$\begin{pmatrix} \tau_{xz} \\ \tau_{yz} \end{pmatrix} = \frac{E}{2(1+\nu)} \begin{pmatrix} \gamma_{xz} \\ \gamma_{yz} \end{pmatrix} \quad (6)$$

The last thing we need is the formula for total energy:

$$\begin{aligned} \Pi = & \frac{1}{2} \int_A \int_{-h/2}^{h/2} \begin{pmatrix} \epsilon_{xx} & \epsilon_{yy} & \gamma_{xy} \end{pmatrix} \begin{pmatrix} \tau_{xx} \\ \tau_{yy} \\ \tau_{xy} \end{pmatrix} dz dA \\ & + \frac{k}{2} \int_A \int_{-h/2}^{h/2} \begin{pmatrix} \epsilon_{xz} & \epsilon_{yz} \end{pmatrix} \begin{pmatrix} \tau_{xz} \\ \tau_{yz} \end{pmatrix} dz dA \end{aligned} \quad (7)$$

Then we could use the variational principle to gain the specific formula of each term in the equation below.

$$(K_\kappa + K_\gamma)d = f \quad (8)$$

However, I soon find out that this element could not pass any kinds of patch test, since it has represent bending. Therefore, I choose to modify the element to 8 nodes.

Plate8Q

The increase of the number of nodes barely change anything. The only alternation is using the serendipity shape function for 8 nodes instead of the bilinear one of 4 nodes.

0.3.11 Convergence Analysis

The convergence of the two elements are the same as 4Q and 8Q.

Both could exactly reconstruct linear field. And the plate8Q element could only reconstruct a quadratic field only is the coordination transformation is linear.

0.3.12 Patch Test

After discussion, our group decides to use a simple quadratic w field to do the patch test.

As we have no idea the deformation formula for a plate, we simplified it to a beam by restricting the β_y for every node and fixed one end of the board to make it a cantilever.

From the formula of a cantilever in the situation that only a torque acts on the free end.

$$\begin{aligned} w &= \frac{Mx^2}{2EI} \\ \beta_x &= \frac{Mx}{EI} \end{aligned} \quad (9)$$

In the patch test, we set the length of the board as 40m, the width as 2m, and the thickness as 0.1m, E as 10^{10} , ν as 0, and the total M in one end as 80 N.M.

Therefore,

$$I = 2 \times \int_{-0.05}^{0.05} z^2 dz = \frac{1}{6000} \quad (10)$$

Put those numbers back to (7), we have,

$$\begin{aligned} w &= 2.4 \times 10^{-5} x^2 \\ \beta_x &= 4.8 \times 10^{-5} x \end{aligned} \quad (11)$$

And the result from the two kinds of elements are:

The 4 nodes element:

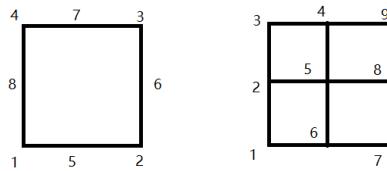
D I S P L A C E M E N T S

NODE	X-DISPLACEMENT	Y-DISPLACEMENT	Z-DISPLACEMENT
1	0.00000E+00	0.00000E+00	0.00000E+00
2	0.00000E+00	0.00000E+00	0.00000E+00
3	0.00000E+00	0.00000E+00	0.00000E+00
4	0.57597E-06	0.57597E-07	0.00000E+00
5	0.57597E-06	0.57597E-07	0.00000E+00
6	0.57597E-06	0.57597E-07	0.00000E+00
7	0.23039E-05	0.11519E-06	0.00000E+00
8	0.23039E-05	0.11519E-06	0.00000E+00
9	0.23039E-05	0.11519E-06	0.00000E+00

The 8 nodes elements:

D I S P L A C E M E N T S			
NODE	X-DISPLACEMENT	Y-DISPLACEMENT	Z-DISPLACEMENT
1	0.00000E+00	0.00000E+00	0.00000E+00
2	0.38400E-01	0.19200E-02	0.00000E+00
3	0.38400E-01	0.19200E-02	0.00000E+00
4	0.00000E+00	0.00000E+00	0.00000E+00
5	0.96000E-02	0.96000E-03	0.00000E+00
6	0.38400E-01	0.19200E-02	0.00000E+00
7	0.96000E-02	0.96000E-03	0.00000E+00
8	0.00000E+00	0.00000E+00	0.00000E+00

The nodes are indexed in the following way.

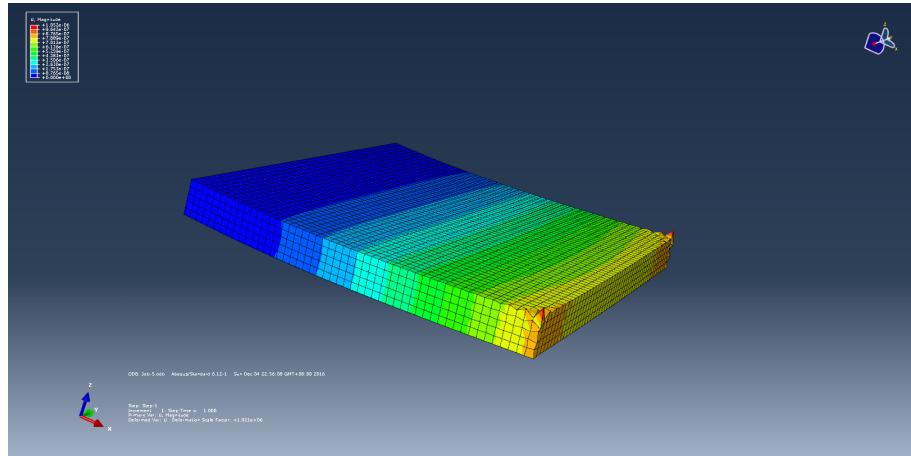


As we could see, since the 4 nodes element could not represent bending, the result of it was poor, while the 8 nodes element successfully passed the patch test.

0.3.13 Application

I use the Abaqus to check the 2 element.

The problem is that two concentrate force act on the corner of a square board.



The maxima of displacement is 1.05E-6

In the plate4Q element, I use 16 elements. The answer is shown below.

DISPLACEMENTS			
NODE	W	BETA_X	BETA_Y
1	0.0000E+00	0.0000E+00	0.0000E+00
2	0.21448E-06	0.42636E-07	0.25127E-08
3	0.21975E-06	0.40684E-07	-0.25891E-21
4	0.0000E+00	0.0000E+00	0.0000E+00
5	0.53146E-07	0.21840E-07	0.22433E-08
6	0.22105E-06	0.41457E-07	0.41694E-09
7	0.60161E-07	0.22520E-07	-0.74446E-22
8	0.0000E+00	0.0000E+00	0.0000E+00
9	0.74715E-06	0.59310E-07	-0.85034E-08
10	0.69327E-06	0.51158E-07	-0.24485E-21
11	0.46051E-06	0.52729E-07	-0.23656E-08
12	0.70919E-06	0.53985E-07	-0.48149E-08
13	0.44462E-06	0.47277E-07	-0.26801E-21
14	0.74715E-06	0.59310E-07	0.85034E-08
15	0.21448E-06	0.42636E-07	-0.25127E-08
16	0.70919E-06	0.53985E-07	0.48149E-08
17	0.46051E-06	0.52729E-07	0.23656E-08
18	0.22105E-06	0.41457E-07	-0.41694E-09
19	0.0000E+00	0.0000E+00	0.0000E+00
20	0.53146E-07	0.21840E-07	-0.22433E-08
21	0.0000E+00	0.0000E+00	0.0000E+00

In the plate8Q element, I use 4 elements. The answer is shown below.

D I S P L A C E M E N T S			
NODE	W	BETA_X	BETA_Y
1	0.00000E+00	0.00000E+00	0.00000E+00
2	0.21345E-07	0.83680E-08	0.76905E-09
3	0.81883E-07	0.15238E-07	0.22491E-09
4	0.17267E-06	0.20285E-07	-0.18742E-08
5	0.28399E-06	0.22341E-07	-0.57413E-08
6	0.25545E-06	0.18556E-07	-0.38969E-08
7	0.16413E-06	0.17803E-07	-0.15525E-08
8	0.82044E-07	0.14474E-07	-0.10578E-09
9	0.23186E-07	0.85076E-08	0.11981E-09
10	0.00000E+00	0.00000E+00	0.00000E+00
11	0.00000E+00	0.00000E+00	0.00000E+00
12	0.23345E-07	0.85465E-08	0.82718E-24
13	0.81689E-07	0.14027E-07	0.00000E+00
14	0.15986E-06	0.16652E-07	-0.66174E-23
15	0.24493E-06	0.17056E-07	-0.26470E-22
16	0.25545E-06	0.18556E-07	0.38969E-08
17	0.16413E-06	0.17803E-07	0.15525E-08
18	0.82044E-07	0.14474E-07	0.10578E-09
19	0.23186E-07	0.85076E-08	-0.11981E-09
20	0.00000E+00	0.00000E+00	0.00000E+00
21	0.00000E+00	0.00000E+00	0.00000E+00
22	0.21345E-07	0.83680E-08	-0.76905E-09
23	0.81883E-07	0.15238E-07	-0.22491E-09
24	0.17267E-06	0.20285E-07	0.18742E-08
25	0.28399E-06	0.22341E-07	0.57413E-08

Since the elements are still few, it is nice to have the same scale. And the plate8Q is better.

0.3.14 无限单元

0.3.15 超级单元

0.3.16 过渡单元

此处的过渡单元为5节点，过渡4Q与9Q单元，使其衔接起来

PatchTest所使用的构型仍为4Q使用的Patchtest方案，即约束左下角两个自由度的位移，以及右下角Y自由度的位移，在顶部施加向下为20N/m的均布力。

其结果如下：

NODE	X-DISPLACEMENT	Y-DISPLACEMENT	Z-DISPLACEMENT	X-ROTATION	Y-ROTATION	Z-ROTATION
1	0. 00000E+00	0. 00000E+00	0. 00000E+00	0. 00000E+00	0. 00000E+00	0. 00000E+00
2	0. 60000E-06	0. 00000E+00	0. 00000E+00	0. 00000E+00	0. 00000E+00	0. 00000E+00
3	0. 60000E-06	-0. 20000E-05	0. 00000E+00	0. 00000E+00	0. 00000E+00	0. 00000E+00
4	0. 173949E-13	-0. 20000E-05	0. 00000E+00	0. 00000E+00	0. 00000E+00	0. 00000E+00
5	0. 30000E-06	0. 00000E+00	0. 00000E+00	0. 00000E+00	0. 00000E+00	0. 00000E+00
6	0. 60000E-06	-0. 10000E-05	0. 00000E+00	0. 00000E+00	0. 00000E+00	0. 00000E+00
7	0. 30000E-06	-0. 20000E-05	0. 00000E+00	0. 00000E+00	0. 00000E+00	0. 00000E+00
8	-0. 105922E-13	-0. 10000E-05	0. 00000E+00	0. 00000E+00	0. 00000E+00	0. 00000E+00
9	0. 30000E-06	-0. 10000E-05	0. 00000E+00	0. 00000E+00	0. 00000E+00	0. 00000E+00

ELEMENT	X-CORRD	Y-CORRD	STRESS_XX	STRESS_VY	STRESS_XY
1	-0.861136E+00	-0.861136E+00	-0.653980E-07	-0.300000E+02	-0.214415E-06
1	-0.861136E+00	-0.339981E+00	0.289206E-06	-0.300000E+02	-0.550864E-07
1	-0.861136E+00	0.339981E+00	0.232497E-06	-0.300000E+02	-0.204030E-06
1	-0.861136E+00	0.861136E+00	-0.209034E-06	-0.300000E+02	-0.591672E-06
1	-0.339981E+00	-0.861136E+00	-0.186423E-07	-0.300000E+02	-0.846520E-07
1	-0.339981E+00	-0.339981E+00	0.194094E-06	-0.300000E+02	-0.217484E-07
1	-0.339981E+00	0.339981E+00	-0.477117E-07	-0.300000E+02	-0.805520E-07
1	-0.339981E+00	0.861136E+00	-0.631110E-06	-0.300000E+02	-0.233595E-06
1	0.339981E+00	-0.861136E+00	-0.186423E-07	-0.300000E+02	0.846521E-07
1	0.339981E+00	-0.339981E+00	0.194094E-06	-0.300000E+02	0.217484E-07
1	0.339981E+00	0.339981E+00	-0.477117E-07	-0.300000E+02	0.805520E-07
1	0.339981E+00	0.861136E+00	-0.631110E-06	-0.300000E+02	0.233595E-06
1	0.861136E+00	-0.861136E+00	-0.653980E-07	-0.300000E+02	0.214415E-06
1	0.861136E+00	-0.339981E+00	0.289206E-06	-0.300000E+02	0.550864E-07
1	0.861136E+00	0.339981E+00	0.232497E-06	-0.300000E+02	0.204030E-06
1	0.861136E+00	0.861136E+00	-0.209034E-06	-0.300000E+02	0.591672E-06

使用连接算例如图：

0.4 高级功能

0.4.1 弹塑性杆分析

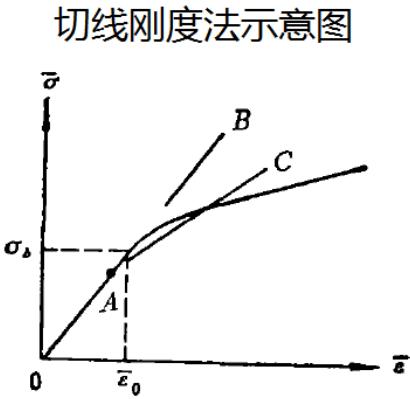
基本原理

弹塑性杆是一种最简单的材料非线性问题。在这个功能中，我使用切线刚度法来进行求解。我把杆的塑性问题简化成弹性段和塑形段，其中每段近似成线性函数，并使用相应的弹性模量和塑形模量来表示两端线段的斜率。切线刚度法的基本思想就是：假使载荷为 P_A 时，相应的位移为 u_A ，切线刚度为 K_A ，那么对于下一次载荷增量产生的位移是，

$$K_A \Delta u_{AB} = \Delta f_{AB}$$

$$u_B = u_A + \Delta u_{AB}$$

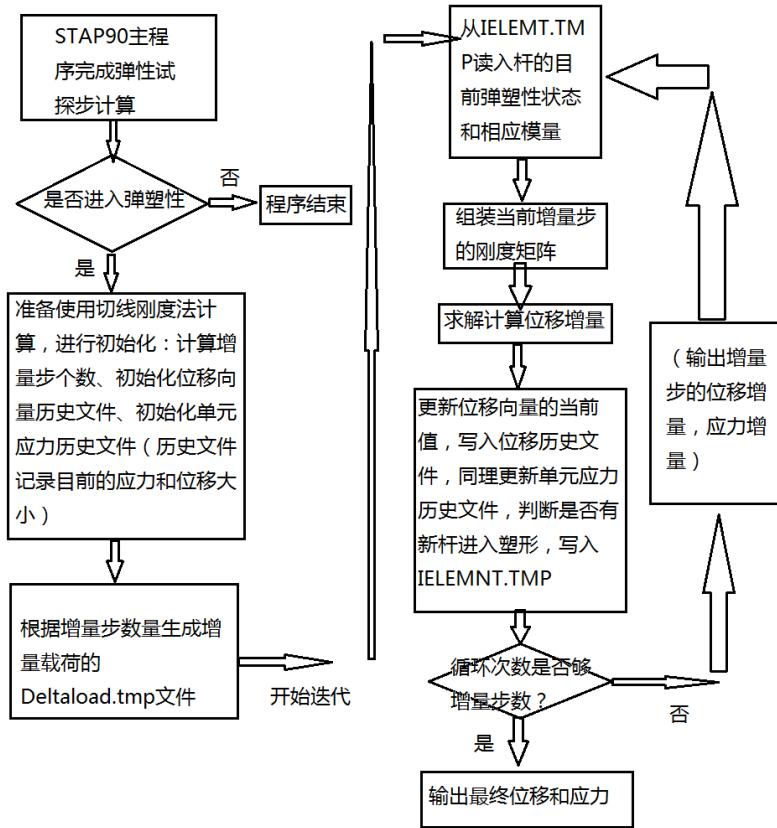
我们在使用切线刚度法进行计算时，都是从A点的应力状态出发，最后就得到载荷B情况下的各项参数，本方法的特点是，对于一次载荷分量，只需要求解一次即可。示意图如下。



该方法参考了1982年固体力学学报《弹塑性有限元一些解法的比较》，由中科院力学研究所吴永礼先生著。

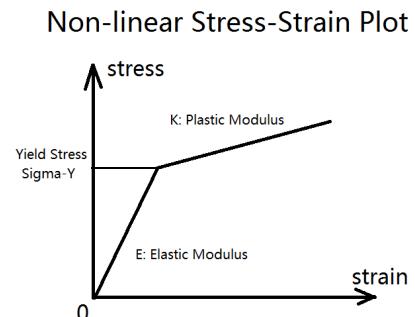
编程思路

首先进行弹性试探步的计算，确定是否需要进行弹塑性分析。若进行弹塑性分析，则使用切线刚度法求解。首先把载荷分成很多细份，从零开始逐渐加载。使用切线刚度，算出每一步的位移增量和应力增量，计算每步加载后的位移和应力，同时判断是否有杆进入弹塑性，若有弹塑性变化，则下一次组装刚度阵时要更新。这种增量步的方法把一个非线性的问题线性化，实质上是微分的思路——把复杂的函数简单化，理论上，这种求解方式可以进行任何材料非线性问题的计算。在这个方法中，每一步更新需要前一步的历史位移和应力信息，这些都记录在临时文件里，每一步均更新。整个程序的流程图如下。



输入和输出结果

在一根杆上加了100000的力，材料的截面性质如下，各输入和输出如下，结果符合理论值。



SET NUMBER	YOUNG' S MODULUS E	CROSS-SECTİONAL AREA A	YIELD-STRESS SIGMA-Y	PLASTIC-MODULES K
1	0. 10000E+12	0. 10000E-01	0. 10000E+06	0. 10000E+11

弹性试探步结果

D I S P L A C E M E N T S

NODE	X-DISPLACEMENT	Y-DISPLACEMENT	Z-DISPLACEMENT	X-ROTATION	Y-ROTATION	Z-ROTATION
1	0. 00000E+00	0. 00000E+00	0. 00000E+00	0. 00000E+00	0. 00000E+00	0. 00000E+00
2	0. 10000E-04	0. 00000E+00	0. 00000E+00	0. 00000E+00	0. 00000E+00	0. 00000E+00

ELASTIC TRIAL SOLUTION F O R E L E M E N T G R O U P 1

ELEMENT NUMBER	FORCE	STRESS
1	0. 10000E+05	0. 10000E+07

增量步计算过程中截图 (这里展现的是弹性屈服的时刻)

NODE	X-DISPLACEMENT	Y-DISPLACEMENT	Z-DISPLACEMENT	X-ROTATION	Y-ROTATION	Z-ROTATION
1	0. 00000E+00	0. 00000E+00	0. 00000E+00	0. 00000E+00	0. 00000E+00	0. 00000E+00
2	0. 10000E-07	0. 00000E+00	0. 00000E+00	0. 00000E+00	0. 00000E+00	0. 00000E+00

位移增量

ELEMENT NUMBER	FORCE	STRESS
1	0. 101000E+04	0. 101000E+06

D I S P L A C E M E N T S

NODE	X-DISPLACEMENT	Y-DISPLACEMENT	Z-DISPLACEMENT	X-ROTATION	Y-ROTATION	Z-ROTATION
1	0. 00000E+00	0. 00000E+00	0. 00000E+00	0. 00000E+00	0. 00000E+00	0. 00000E+00
2	0. 10000E-06	0. 00000E+00	0. 00000E+00	0. 00000E+00	0. 00000E+00	0. 00000E+00

ELASTIC TRIAL SOLUTION F O R E L E M E N T G R O U P 1

ELEMENT NUMBER	FORCE	STRESS
1	0. 102000E+04	0. 102000E+06

最终计算结果

D I S P L A C E M E N T S

NODE	X-DISPLACEMENT	Y-DISPLACEMENT	Z-DISPLACEMENT	X-ROTATION	Y-ROTATION	Z-ROTATION
1	0. 00000E+00	0. 00000E+00	0. 00000E+00	0. 00000E+00	0. 00000E+00	0. 00000E+00
2	0. 909100E-04	0. 00000E+00	0. 00000E+00	0. 00000E+00	0. 00000E+00	0. 00000E+00

F O R E L E M E N T G R O U P 1

ELEMENT NUMBER	STRESS
1	0. 10000E+07

0.4.2 模态分析

背景

N自由度无阻尼线性系统作微幅振动，最后可得到振动微分方程组： $\mathbf{M}\ddot{\mathbf{x}} + \mathbf{K}\mathbf{x} = \mathbf{0}$ 。分析一个弹性体的模态，实质是求解广义特征值问题 $\mathbf{K} - \lambda\mathbf{M} = \mathbf{0}$ ，再由 $\omega = \lambda^{1/2}$ 得到系统的特征频率。对于多自由度系统，往往只有最小的数个特征值，即对应的几个频率最低的模态对结构具有关键作用。搜索最小数个特征值的算法常见的有LANCZOS迭代法，子空间迭代法等。本程序针对刚度阵、质量矩阵采用Skyline存储方式试图采用LANCZOS迭代法对广义特征值问题进行求解，另外针对稀疏矩阵的CSR存储方法，也提供了基于Intel MKL的feast算法库的调用。测试计算采用杆单元，但对大多数单元都加入了组装质量阵的模块，理论上可以进行模态分析。测试结果如Samples/Modal文件夹中所示。

算法分析

LANCZOS迭代法，其基本思路为利用里兹法求解特征值问题：

- 选取与所有已求得的特征向量正交的初始迭代向量 \hat{x}_1 ，并进行正则化：

$$x_1 = \frac{\hat{x}_1}{\beta_1}, \quad \beta_1 = (\hat{x}_1^T M \hat{x}_1)^{1/2}$$

- 选取一移轴量 μ ，生成Lanczos基向量：

$$(K - \mu M)\tilde{x}_i = Mx_{i-1}, \quad a_{i-1} = \tilde{x}_i^T M x_{i-1}, \quad \tilde{x}'_i = \tilde{x}_i - a_{i-1}x_{i-1} - \beta_{i-1}x_{i-2}, \\ \hat{x}_i = \tilde{x}'_i - \sum_{k=1}^{i-1} (\hat{x}_i^T M x_k) x_k - \sum_{j=1}^{n_c} (\hat{x}_i^T M \phi_j) \phi_j, \quad \beta_i = (\hat{x}_i^T M \hat{x}_i)^{1/2}, \quad x_i = \frac{\hat{x}_i}{\beta_i}$$

- 求解 $Tz = \frac{1}{\lambda}z$ ，得到在本次迭代中收敛的特征值及对应的特征向量，如果收敛个数超过所需个数，则停止迭代，否则转1.

0.4.3 动力学响应分析