有限元大作业答辩

Report on the Final Assignment

第二组

组长：贺琪

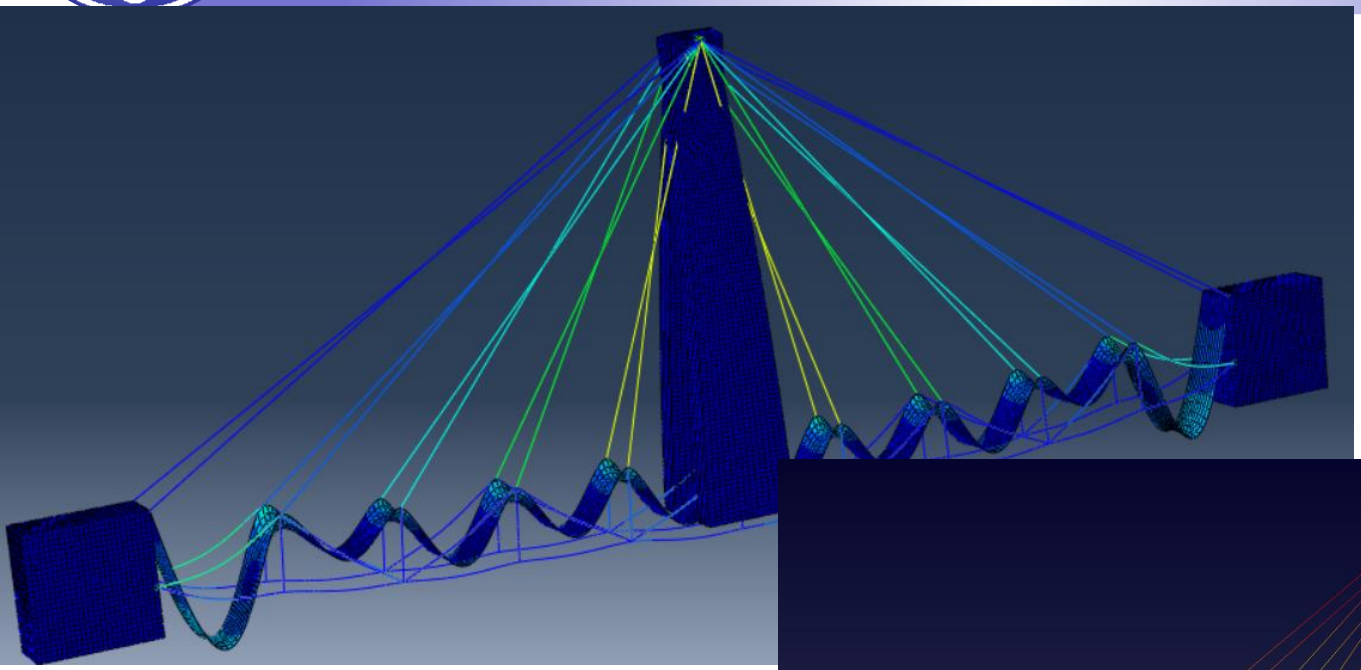
组员：陈煜，刘畅武，杨昊光，朱子霖

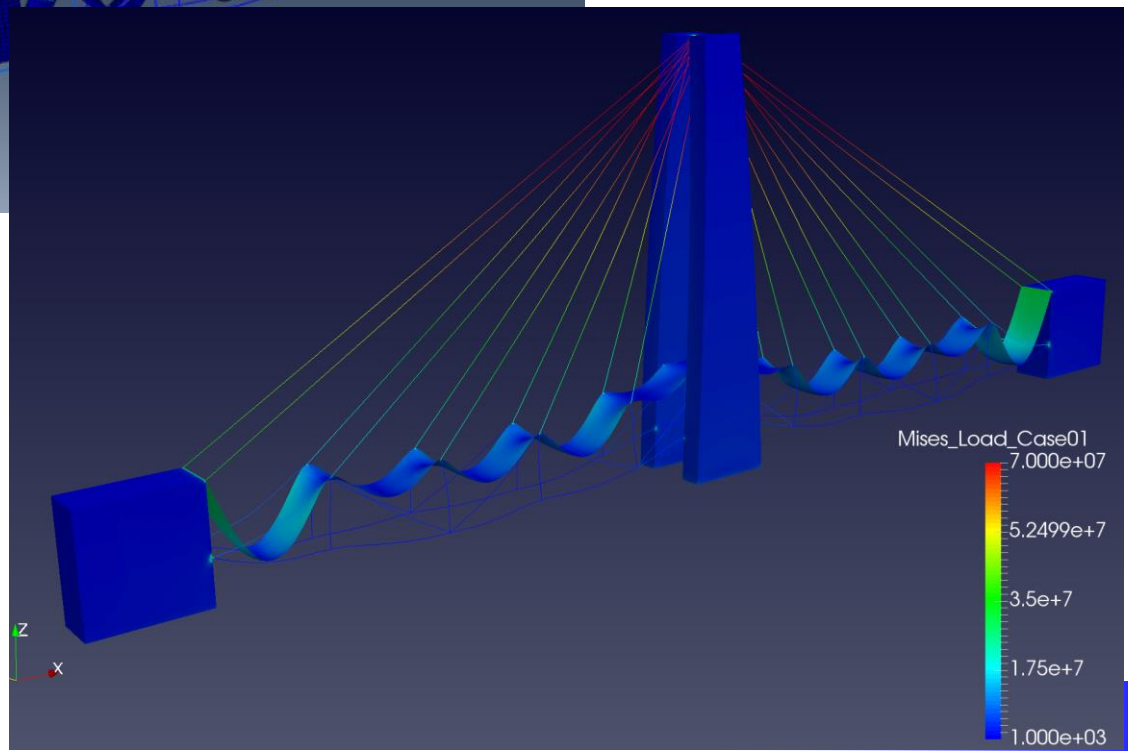
2016. 12. 27

成果汇报

- 通过一个月的努力，在小组成员的紧密配合之下，本组顺利完成了大作业的任务。
- 我们成功运行了前三个规模输入程序（Job-4受限于测试机内存，未能跑通），并在3个任务中都取得了最快的成绩。其中Job-1约0.2s，Job-2约3.5s，Job-3约35s。
- 于此同时。我们还是唯一使用ParaView进行后处理的小组。

成果汇报

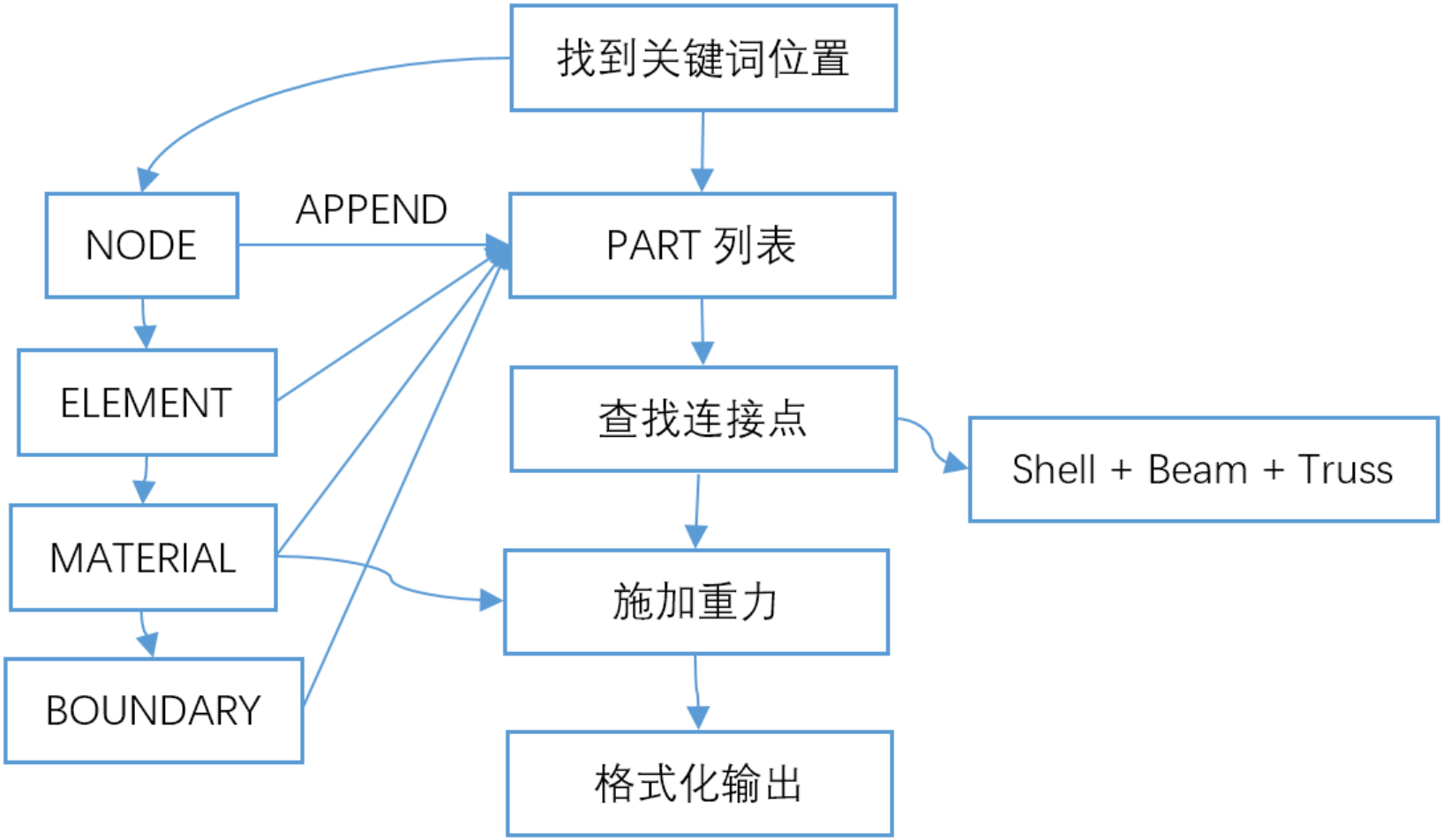




内容简介

- 前处理
- 后处理
- 基本单元（8H，梁，壳）
- 高级功能

前处理


找到关键词位置
NODE
APPEND
PART 列表
ELEMENT
查找连接点
Shell + Beam + Truss
MATERIAL
施加重力
BOUNDARY
格式化输出

前处理

输入文件

Bridge
0110

4087	4	1	1
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0
5	0	0	0
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
10	0	0	0
11	0	0	0
12	0	0	0
13	0	0	0

1759	3127	3128	2915	2914	3424
1760	1.总单元数	2910	2834	2915	3425
2884				3.此单元对应节点号	
4	2.此单元对应节点数				
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		

前处理

- Python-abaqus二次开发
  - Mesh on part -- Mesh on instance
    - mdb.models['Model-1'].setValues(noPartsInputFile=ON)
  - Merge Nodes
    - mdb.models['Model1'].rootAssembly.InstanceFromBooleanMerge(name='PartMerge',instances=('Part-1-1','Part-2-1','Part-3-2','Part-3-2-lin-1-2'domain=MESH,mergeNodes=ALL)
  - Renumber Nodes

后处理

- 后处理部分细分可以分为以下3个部分
  - SPR
  - 输出VTK数据文件格式
  - 使用ParaView绘图（唯一一组）

SPR

- 功能
  - 程序中所有单元都有相应的SPR调用支持。SPR根据单元的维数和需要恢复的应力分量个数自动分配数组，根据节点连接单元的情况自动决定恢复的阶数。
  - 目前支持二阶、一阶和零阶（节点平均法）恢复应力。

SPR

- 基本思路
  - 独立的最小二乘逼近子程序
  - 拟合的目标函数（以三维为例）
    - 二阶

$$f=a_0+a_1x+a_2y+a_3z+a_4xy+a_5yz+a_6zx$$

    - 一阶

$$f=a_0+a_1x+a_2y+a_3z$$

SPR

- 实现思路
  - 构建节点连接关系
  - NodeRelationFlag(NUMNP,NPAR(5)*2+12)
  - 最后两列为状态指示列
    - 一共有多少单元连接到该节点
    - 循环遇到的第一个连接到该节点的单元（用于恢复坐标）
  - 由节点连接关系得到Patch包含的高斯点及其应力状态。
  - 进而得到拟合所需的数据点和所需要恢复的点的坐标。
  - 代入程序里，得到NStress个系数向量。
  - 根据系数向量和节点坐标，得到恢复应力。

ParaView和VTK数据文件格式

- 使用ParaView做后处理和数据可视化。使用标准VTK格式数据文件。
- 数据文件所需信息多数在程序执行过程中从内存中清除
  - 使用了3个临时文件，分别记录
    - 位移和应力恢复信息
    - 连接矩阵信息
    - 单元类型信息
  - 添加全局变量记录总“连接数”
  - 最后在VTKgenerate子程序的最后一步读取临时文件进行整合。

```
# vtk DataFile Version 2.0                    ](1)
Really cool data                              ](2)
ASCII | BINARY                                ](3)
DATASET type                                  ]
...                                           ](4)
POINT_DATA n                                  ]
...                                           ](5)
CELL_DATA n                                   ]
...                                           ]
```

Part 1: Header

Part 2: Title (256 characters maximum, terminated with newline \n character)

Part 3: Data type, either ASCII or BINARY

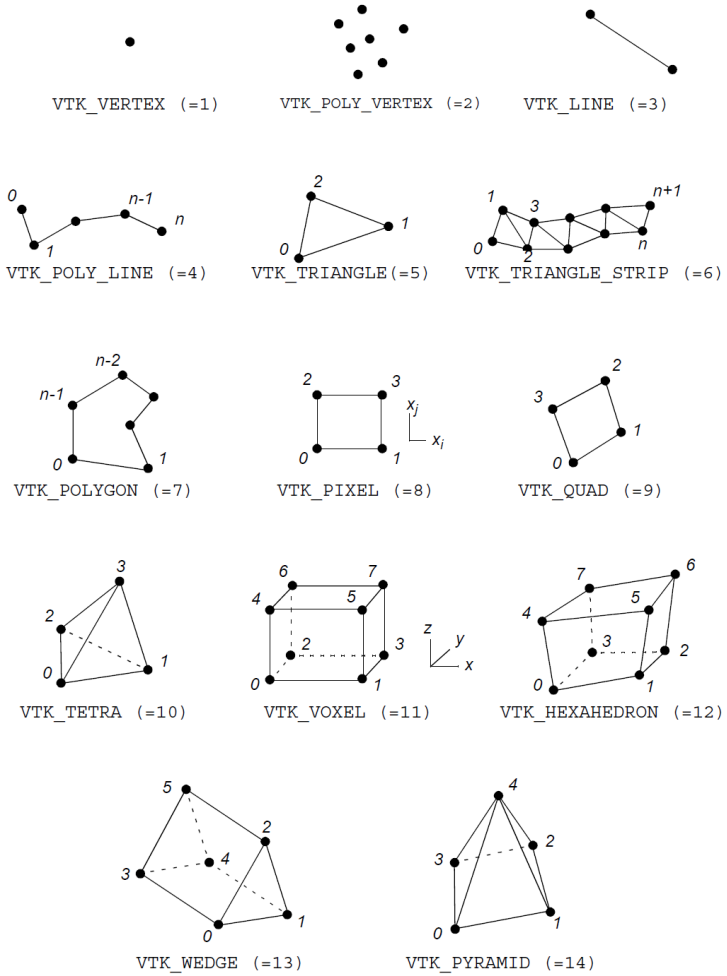
Part 4: Geometry/topology. *Type* is one of:

```
STRUCTURED_POINTS
STRUCTURED_GRID
UNSTRUCTURED_GRID
POLYDATA
RECTILINEAR_GRID
FIELD
```

Part 5: Dataset attributes. The number of data items *n* of each type must match the number of points or cells in the dataset. (If *type* is FIELD, point and cell data should be omitted.

VTK数据文件

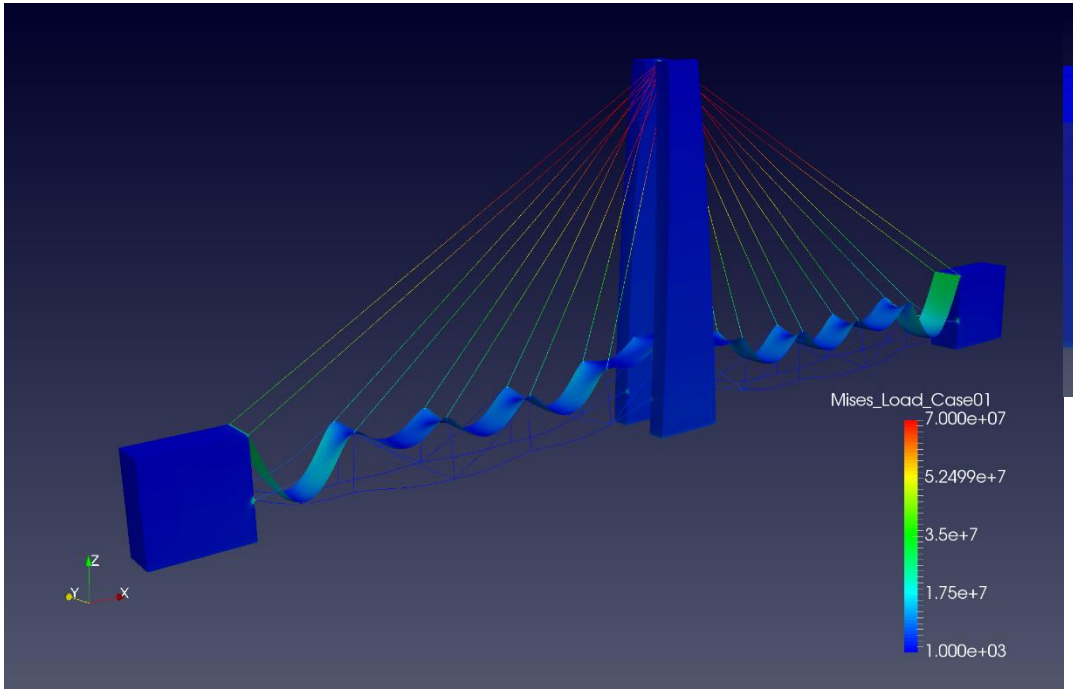
- 由于单元类型众多，采用UNSTRUCTURED_GRID 作为拓扑类型
- 输出三个场量（FIELD_DATA）
  - 位移（三元，向量）
  - 应力（六元，张量）
  - Von Mises应力（一元，标量）
  - 在ParaView中可以自动识别三元位移向量场并把它叠加到本体上，自动生成变形后的图像。

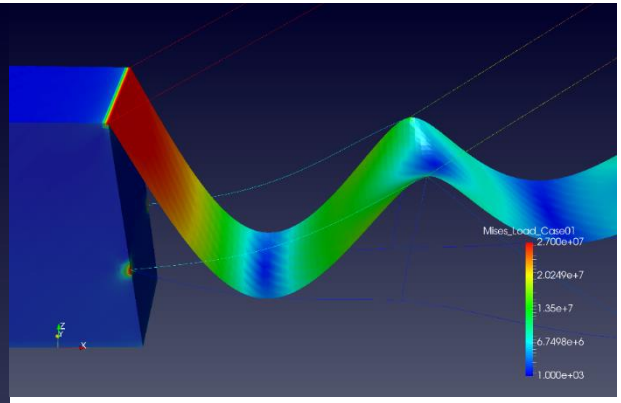


VTK数据文件示例

```
# vtk DataFile Version 3.0
 Bridge-2
ASCII
DATASET UNSTRUCTURED_GRID
POINTS       37067 double
   200.000000000000       -10.0000000000000
0.000000000000000E+000
   200.000000000000        10.0000000000000
0.000000000000000E+000
… … … … … …
CELLS        30370      255110
           4           0          22         601          87
           4          22          23         602         601
… … … … … …
CELL_TYPES        30370
           9           9           9           9           9
9
           9           9           9           9           9
9
… … … … … …
CELL_DATA       30370
POINT_DATA       37067
FIELD Result             3
Displacement_Load_Case01            3       37067 double
 -1.041396576690393E-002  1.633312869524324E-004 -
9.600850712844450E-002
… … … … … …
Stress_Load_Case01             6       37067 double
   2838072.96290648        150454.911686954
2528120.49159094
… … … … … …
Mises_Load_Case01              1       37067 double
   17184828.3301651        17705078.1532487
23082108.9025568
… … … … … …
```

ParaView效果展示





8H

- 给出一个8H单元，其节点坐标矩阵为$\boldsymbol{X}$，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。有：

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

- 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。

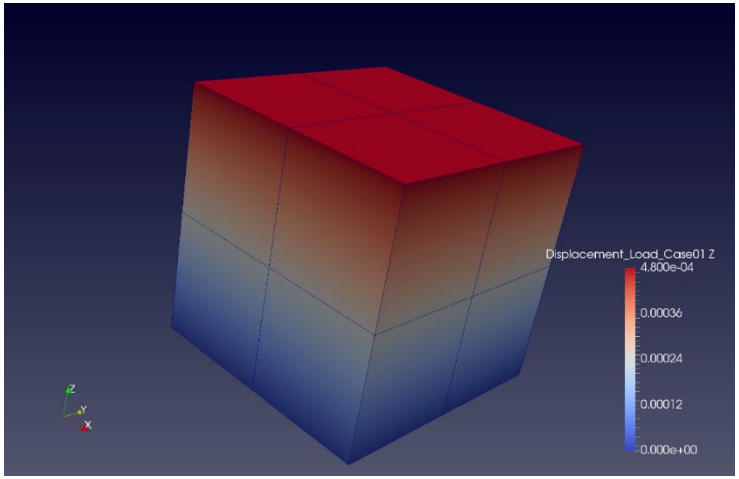
$$\boldsymbol{B}_i = \begin{bmatrix} B_{ix} & 0 & 0 \\ 0 & B_{iy} & 0 \\ 0 & 0 & B_{iz} \\ B_{iy} & B_{ix} & 0 \\ 0 & B_{iz} & B_{iy} \\ B_{iz} & 0 & B_{ix} \end{bmatrix},\ i=1,2,...,8 \qquad \boldsymbol{K}^e = \sum_{i=1}^{2}\sum_{j=1}^{2}\sum_{k=1}^{2} w_i w_j w_k \boldsymbol{B}^T(\xi_i,\eta_j,\zeta_k)\boldsymbol{D}\boldsymbol{B}(\xi_i,\eta_j,\zeta_k)$$

8H

- 分片试验做了7单元和8单元两种，设置线性位移场和C类边界条件，均得到了精度在机器ε量级的结果。（开发时使用的编译器为GFortran，没有IVF对精度的各种优化。后来在IVF上算出的误差均为0）

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

NODE	X-DISPLACEMENT	Y-DISPLACEMENT	Z-DISPLACEMENT
1	0.58775E-19	0.14908E-18	0.48000E-03
2	-0.60000E-04	0.00000E+00	0.48000E-03
3	-0.12000E-03	-0.27105E-19	0.48000E-03
4	0.53522E-19	-0.60000E-04	0.48000E-03
5	-0.60000E-04	-0.60000E-04	0.48000E-03
6	-0.12000E-03	-0.60000E-04	0.48000E-03
7	0.14908E-18	-0.12000E-03	0.48000E-03
8	-0.60000E-04	-0.12000E-03	0.48000E-03
9	-0.12000E-03	-0.12000E-03	0.48000E-03
10	-0.10435E-19	0.94868E-19	0.24000E-03
11	-0.60000E-04	0.13553E-19	0.24000E-03
12	-0.12000E-03	-0.25411E-19	0.24000E-03
13	0.84703E-21	-0.60000E-04	0.24000E-03
14	-0.48000E-04	-0.72000E-04	0.33600E-03
15	-0.12000E-03	-0.60000E-04	0.24000E-03
16	0.60986E-19	-0.12000E-03	0.24000E-03
…	…	…	…



梁单元

- 欧拉梁
- 铁木辛柯梁

欧拉梁

- 难点在于刚度阵的组装
  - 三维情况是拉压和弯扭的组合变形状态，虽然相互解耦，但是得组装在同一个12阶矩阵里。

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

symmetry

欧拉梁

- 上述刚度阵在形心主轴的局部坐标系中，还需坐标变换。

$$K = T^T K T$$

其中　T=diag{t,t,t,t},每个t为一个3阶方向余弦矩阵

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	0.572917E-12	0.000000E+00	-0.114583E-10	-0.437500E-10	0.000000E+00	-0.218750E-11
3	0.208333E-11	0.000000E+00	-0.416667E-10	-0.750000E-10	0.000000E+00	-0.375000E-11
4	0.421875E-11	0.000000E+00	-0.843750E-10	-0.937500E-10	0.000000E+00	-0.468750E-11
5	0.666667E-11	0.000000E+00	-0.133333E-09	-0.100000E-09	0.000000E+00	-0.500000E-11

- 欧拉梁的改进和缺陷：
  - 采用Hermite插值，转角连续，无法处理梁之间铰接的情况——不传递弯矩和转角，只传递力。
  - 梁在三维空间中仅由两个坐标确定，只给出了一个主轴方向，剩余两个主轴方向未知，只能人为指定——可以采用一些自动求最大惯性矩的算法？

铁木辛柯梁

- 铁木辛柯梁和欧拉梁的唯一区别就是他们基于的弯曲理论不同，给一种弯曲理论，就可以组装一种梁的刚度阵。
- 因此，只需要在欧拉梁刚度阵的弯曲部分改动即可。
- 一维铁木辛柯梁的刚度阵如下:

$$[k_b]=\frac{EI}{l}\begin{bmatrix}0&0&0&0\\0&1&0&-1\\0&0&0&0\\0&-1&0&1\end{bmatrix}$$

$$[k_s]=\frac{GAk}{l}\begin{bmatrix}1&\frac{l}{2}&-1&\frac{l}{2}\\\frac{l}{2}&\frac{l^2}{3}&-\frac{l}{2}&\frac{l^2}{6}\\-1&-\frac{l}{2}&1&-\frac{l}{2}\\\frac{l}{2}&\frac{l^2}{6}&-\frac{l}{2}&\frac{l^2}{3}\end{bmatrix}$$

铁木辛柯梁

- 由于转角和挠度同阶插值，但是它们互为导数关系，因此引起剪切锁闭！这个刚度阵不能用，所以，对于剪切变形部分我们采用减缩积分。

$$\left(\frac{EI}{l}\begin{bmatrix}0&0&0&0\\0&1&0&-1\\0&0&0&0\\0&-1&0&1\end{bmatrix}+\frac{GAk}{l}\begin{bmatrix}1&l/2&-1&l/2\\l/2&l^2/4&-l/2&l^2/4\\-1&-l/2&1&-l/2\\l/2&l^2/4&-l/2&l^2/4\end{bmatrix}\right)\begin{Bmatrix}w_1\\\psi_1\\w_2\\\psi_2\end{Bmatrix}=\begin{Bmatrix}P_1\\M_1\\P\\M\end{Bmatrix}$$

铁木辛柯梁

- 采用铁木辛柯梁理论的刚度阵：

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

symmetry

铁木辛柯梁

- 计算结果展示：

```
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      2.0E2
    1    1    2    1    0
    2    2    3    1    0
    3    3    4    1    0
    4    4    5    1    0
stop
```

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	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
2	0.144754E-10	0.000000E+00	-0.289509E-09	-0.437500E-10	0.000000E+00	-0.218750E-11
3	0.298884E-10	0.000000E+00	-0.597768E-09	-0.750000E-10	0.000000E+00	-0.375000E-11
4	0.459263E-10	0.000000E+00	-0.918527E-09	-0.937500E-10	0.000000E+00	-0.468750E-11
5	0.622768E-10	0.000000E+00	-0.124554E-08	-0.100000E-09	0.000000E+00	-0.500000E-11

壳单元

程序设计思路

由平板弯曲造成应变：

$$\begin{bmatrix} \tau_{xx1} \\ \tau_{yy1} \\ \tau_{xy1} \end{bmatrix} = -z \times \frac{E}{1-\nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} \begin{bmatrix} \frac{\partial \beta_x}{\partial x} \\ \frac{\partial \beta_y}{\partial y} \\ \frac{\partial \beta_y}{\partial x} + \frac{\partial \beta_x}{\partial y} \end{bmatrix}$$

$$\begin{bmatrix} \tau_{xz} \\ \tau_{yz} \end{bmatrix} = \frac{E}{2(1+\nu)} \begin{bmatrix} \frac{\partial w}{\partial x} - \beta_x \\ \frac{\partial w}{\partial y} - \beta_y \end{bmatrix}$$

由薄膜应力造成的应变：

$$\begin{bmatrix} \tau_{xx2} \\ \tau_{yy2} \\ \tau_{xy2} \end{bmatrix} = \frac{E}{1-\nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} \begin{bmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial v}{\partial y} \\ \frac{\partial v}{\partial x} + \frac{\partial u}{\partial y} \end{bmatrix}$$

壳单元

程序设计思路

对其进行虚功原理分析，得到方程：

$$\int_A \int_{-h/2}^{h/2} \begin{bmatrix} \varepsilon_{xx1} & \varepsilon_{yy1} & \varepsilon_{xy1} \end{bmatrix} \begin{bmatrix} \tau_{xx1} \\ \tau_{yy1} \\ \tau_{xy1} \end{bmatrix} dzdA + k\int_A \int_{-h/2}^{h/2} \begin{bmatrix} \varepsilon_{xz} & \varepsilon_{yz} \end{bmatrix} \begin{bmatrix} \tau_{xz} \\ \tau_{yz} \end{bmatrix} dzdA + \int_A \int_{-h/2}^{h/2} \begin{bmatrix} \varepsilon_{xx2} & \varepsilon_{yy2} & \varepsilon_{xy2} \end{bmatrix} \begin{bmatrix} \tau_{xx2} \\ \tau_{yy2} \\ \tau_{xy2} \end{bmatrix} dzdA$$

$$= \int_A wpdA$$

(注：由于选取的是中性面，平板弯曲和薄膜应力在交叉项积分为0，因而可以直接解耦)

所以类比之前的刚度矩阵构造，我们可以构造单元刚度矩阵为：

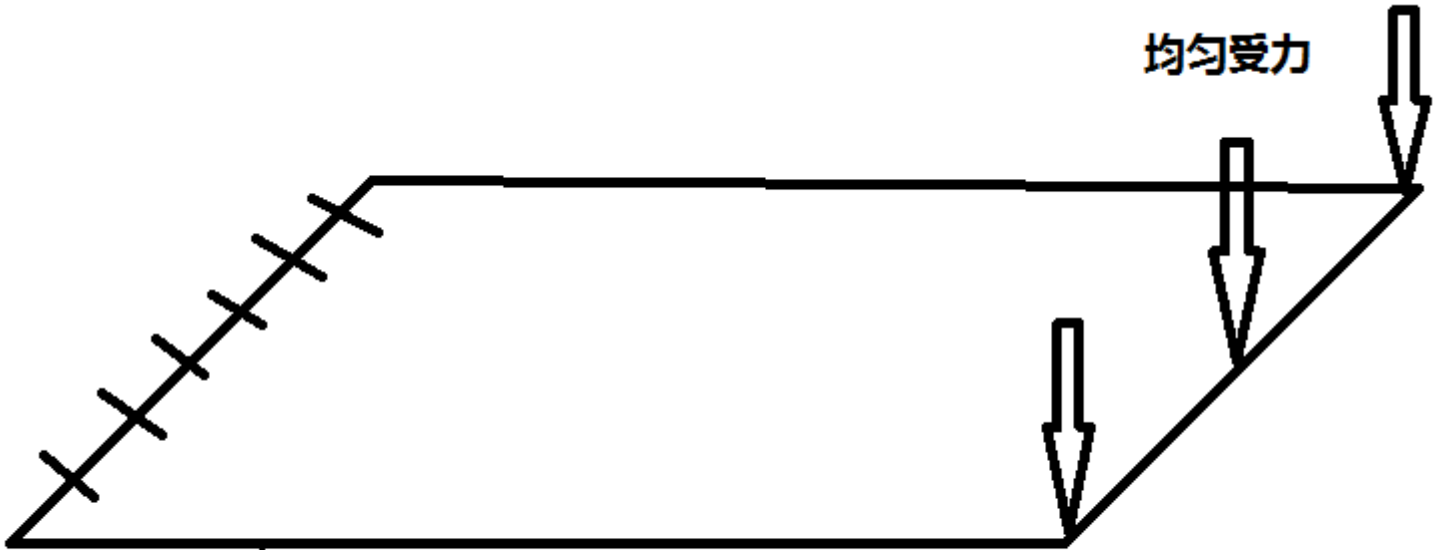
$$K = \int_A (B_\kappa^T C_b B_\kappa + B_\gamma^T C_s B_\gamma + B_p^T C_p B_p) dA$$

此时便按照原STAP90程序计算结果即可

本题使用的是八节点单元，坐标变化为四节点的线性变换。

壳单元

- 分片试验



高级功能

- 半带宽优化
- 稀疏矩阵求解器
- 模态分析
- 动态分析
- 弹塑性杆分析
- 过渡单元
- 无限单元

半带宽优化

- 算法选择
  - 处于节约时间考虑，首先舍弃模拟冷却方法。
  - 而由于FORTRAN不宜实现图结构，故采用最简单的CM算法进行半带宽优化。
  - CM算法的具体为一个广度优先搜索(dfs)，其以入度为优先度维护一优先队列从而进行排序。
- 实现思路
  - 思路较为朴素，首先用FORTRAN实现链表模块，后通过链表长度等接口来直接实现算法

半带宽优化

- 效果
  - 不得不提，在采用CM算法进行半带宽优化之后不仅没有降低skyline求解器所需要的时间，反而将时间从12s提升到了18s（x64编译器 release模式，输入文件为Job-1)
  - 经分析后发现，虽然CM算法着实减小了半带宽，但却增加了skyline存储的元素总个数(nwk)，可谓得不偿失。

稀疏矩阵求解器

- 稀疏矩阵求解器的使用可以说是本次桥任务中提高速度的最重要的一环。
- 本次采用的是*Intel® Math Kernel Library* (MKL)中的pardiso求解器，其为目前市面上最优质的稀疏矩阵求解器。
- 稀疏矩阵的存储方式为CSC，是一种常见的存储方式

稀疏矩阵求解器

- 实现思路
  - 由于大家都是使用pardiso，而其包装好的接口都是相同的，所以在相同配置下，决定速度的主要是生成CSC格式的算法。
  - 本程序的实现主要分为3部分
    - pardiso_input (初步生成CSC)
    - pardiso_assem (装配)
    - pardiso_crop (修建整理CSC)

稀疏矩阵求解器

- 实现思路
  - pardiso_input
    - 在这部分我通过和半带宽优化类似的手段通过外加输入内容的方式得到CSC中想要的columns和rowIndex
    - 在实验中我发现，采用链表的方式不利于编译器优化，于是将编写vector模块，通过牺牲少量内存的方式大幅提高速度

稀疏矩阵求解器

- 实现思路
  - pardiso_assem
    - 具体的组装方式和skyline类似，唯一需要注意的是truss部分在原先的程序中局部刚度阵只计算了上半三角部分。

稀疏矩阵求解器

- 实现思路
  - pardiso_crop
    - 由于局部刚度阵中很多元素为0，导致组装出来的CSC格式中的value部分含有大量0.为了提高速度，加入crop函数，通过一次对value的遍历，同时更新columns，rowIndex，nwk和value，使最后的稀疏度达到了0.0036%（Job-3数据）,从而把pardiso解算所需要的时间控制到最低

模态分析与动力学分析

- 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：

$$M\ddot{x} + Kx = 0$$

  - 求解模态实质为求解广义特征值问题：

$$K - \lambda M = 0, \qquad \omega = \lambda^{1/2}$$

  - 往往只需求解出最小的几个正特征值。

模态分析与动力学分析

- 实现方法
- 组装质量矩阵
  - 添加全局变量DYNANALYSIS，若设置为.TRUE.则组装质量阵，存放在DA(NP(10))。
  - 如果采用稀疏矩阵求解器，则同时组装Row Index和Column Index向量，存放在IA(NP(9))，IA(NP(8))。

模态分析

- 采用基于Skyline储存方法的LANCZOS算法
- Intel MKL Extended Eigensolver 使用FESAT算法（基于Rayleigh-Ritz法）
  - 采用CSR存储格式
  - 由于在位移求解器存储时刚度阵只存储了上三角，因此对于CSR形式，作转置为输入下三角，其余寻址向量不需改变。

模态分析

- 结果示例

```
 a test to STAP90
… … … … … … …
 N O D A L   P O I N T   D A T A

  NODE          BOUNDARY                              NODAL POINT
 NUMBER     CONDITION  CODES                          COORDINATES
              X    Y    Z   RX   RY   RZ              X           Y           Z
    1         1    1    1    1    1    1          -0.300       0.520       0.000
    2         1    1    1    1    1    1           0.520       0.520       0.000
    3         0    0    1    1    1    1           0.000       0.000       0.000
… … … … … … …

  SET       YOUNG'S     CROSS-SECTIONAL   DENSITY
 NUMBER     MODULUS          AREA
              E               A               ρ
    1   0.20700E+12     0.120000E-03     0.400000E+04


 E L E M E N T   I N F O R M A T I O N

 ELEMENT     NODE     NODE       MATERIAL
 NUMBER-N      I        J       SET NUMBER
        1              1             3              1
        2              3             2              1
… … … … … … …
 ------------------------------------------------------------------------------
           E I G E N   V A L U E   C A L C U L A T I O N   R E S U L T S

 ERROR BOUNDS REACHED ON EIGENVALUES AT RESTART OF    1
  0.1191E-24 0.5006E-24 0.0000E+00

 DEGREES OF EIGEN IS    2 THE NUMBER OF CONVERGENCD IS    2

 THE CALCULATED EIGENVALUES ARE
   0.15524999978419E+09  0.20321854319299E+09

 THE CALCULATED EIGENVECTORS ARE

  0.1921E+01 0.1109E+01
  0.0000E+00 0.2509E+01

 ERROR MEASURES ON THE EIGENVALUES
  0.2909E-15 0.4864E+00

 UPPER BOUNDS ON EIGENVALUE CLUSTERS
   0.15680249974733E+09  0.00000000000000E+00

 NO. OF EIGENVALUES IN EACH CLUSTER
                    1                    2
 NO. OF EIGENVALUES LESS THAN UPPER BOUNDS
                    1                    3

 CHECK APPLIED AT SHIFT   0.00000000000000E+00
 THERE ARE       0 EIGENVALUES MISSING
 ------------------------------------------------------------------------------
… … … … … … …
```

动态分析

程序设计思路

根据弹性力学和牛顿定律，我们有公式：

$$M\ddot{U} + C\dot{U} + KU = R$$

于是我们根据中心查分法得到以下方程（无阻尼）：

$$(\frac{M}{\triangle t^2})U^{t+\triangle t} = R^t - (K - \frac{2}{\triangle t^2}M)U^t - (\frac{M}{\triangle t^2})U^{t-\triangle t}$$

于是通过该方程对于该情况不断迭代求解即可。

M有两种组装方法：

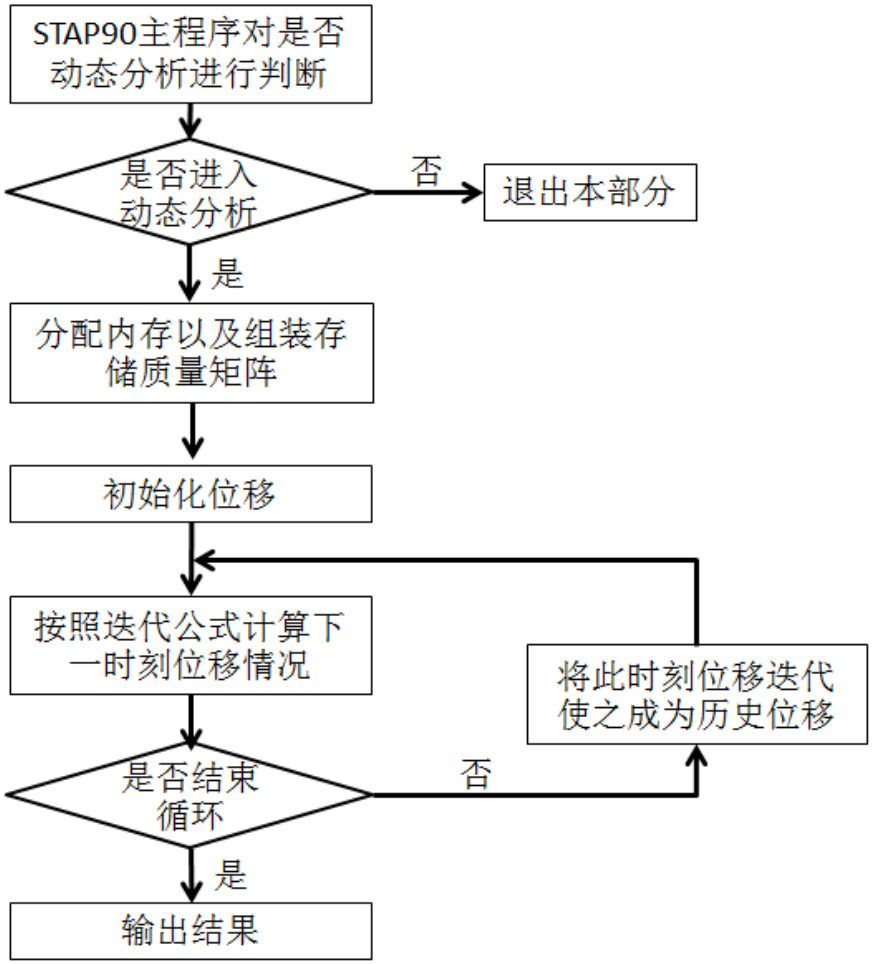
1、与K相同的组装方法，最终通过colsol求解器直接把M当做原Stap90中的K求解。

2、将初步组装的M矩阵按行叠加到对角元上，简化计算。

以下实例使用的是第二种组装方法，而桥使用的第一种组装方法。

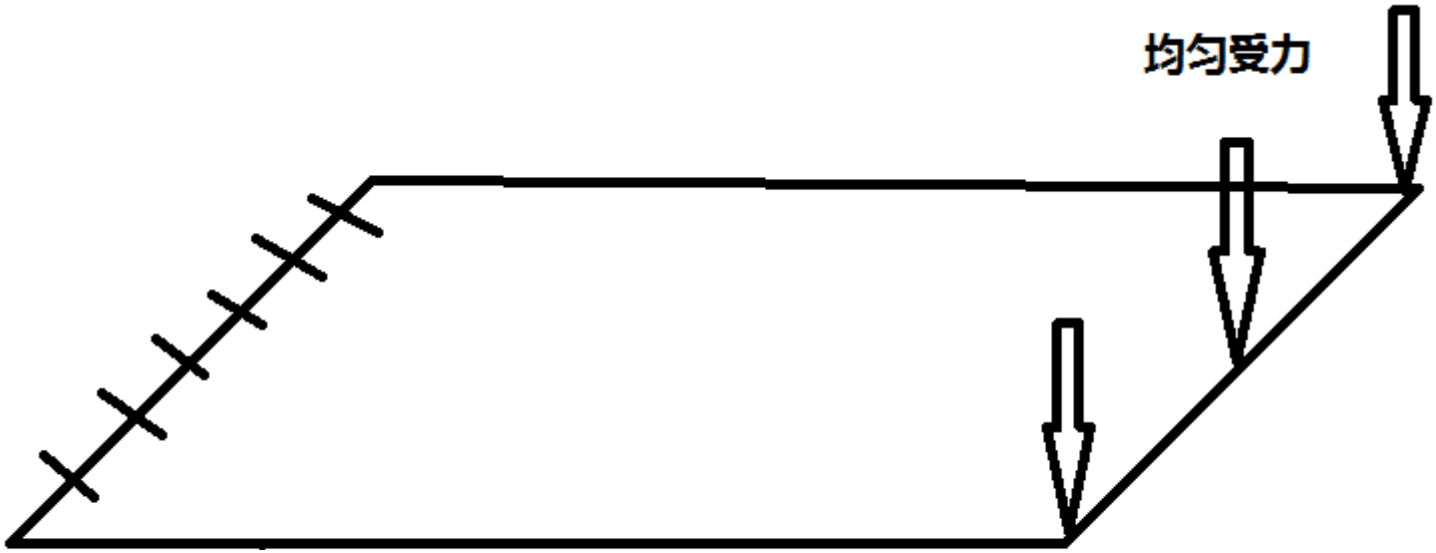
动态分析

- 程序设计思路


STAP90主程序对是否动态分析进行判断
是否进入动态分析
否
退出本部分
是
分配内存以及组装存储质量矩阵
初始化位移
按照迭代公式计算下一时刻位移情况
将此时刻位移迭代使之成为历史位移
是否结束循环
否
是
输出结果

动态分析

- 实例验证

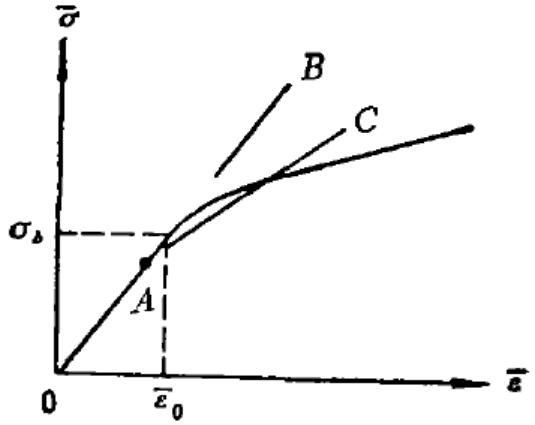


弹塑性杆分析

- 计算方法：切线刚度法
- 复杂问题简单化： 微分——线性化
- 虽然本构非线性，但是增量是线性的。

- 这个问题是从零应力开始的一次加载问题，是最简单的弹塑性情形。

切线刚度法示意图

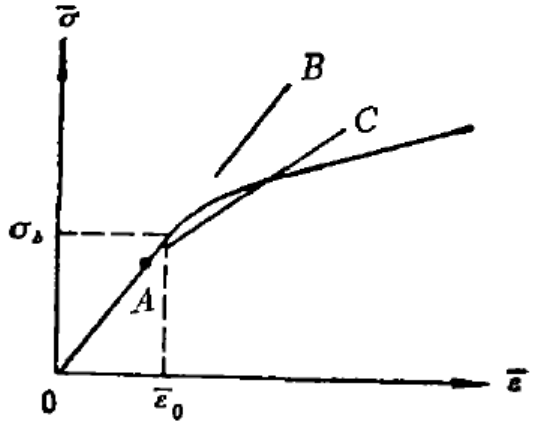


弹塑性杆分析

- 计算方法：切线刚度法
- 复杂问题简单化： 微分——线性化
- 虽然本构非线性，但是增量是线性的。

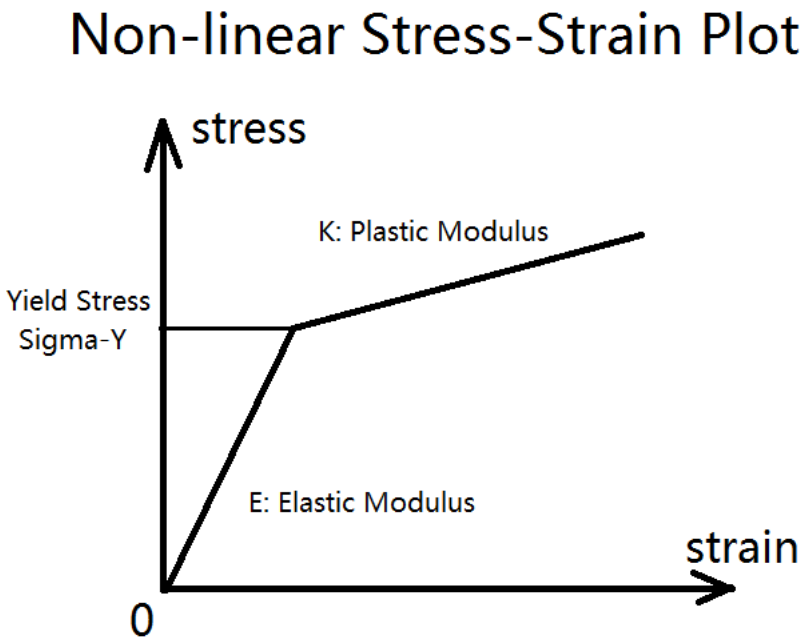
- 这个问题是从零应力开始的一次加载问题，是最简单的弹塑性情形。

切线刚度法示意图



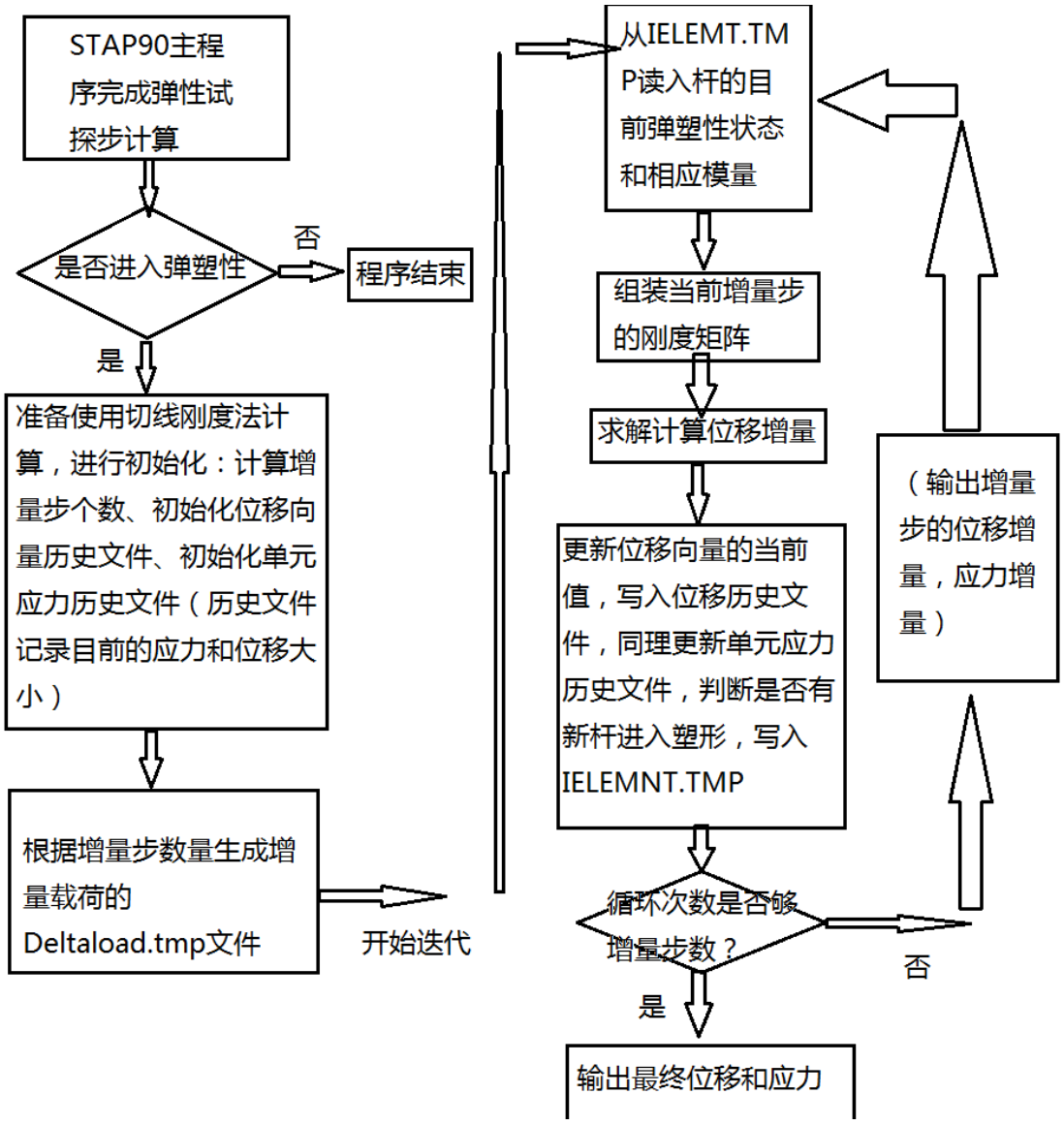
弹塑性杆分析

- 输入参数示例：



SET NUMBER	YOUNG'S MODULUS E	CROSS-SECTIONAL AREA A	YIELD-STRESS SIGMA-Y	PLASTIC-MODULES K
1	0.10000E+12	0.100000E-01	0.10000E+06	0.10000E+11

弹塑性杆分析


STAP90主程序完成弹性试探步计算
是否进入弹塑性
否
程序结束
是
准备使用切线刚度法计算，进行初始化：计算增量步个数、初始化位移向量历史文件、初始化单元应力历史文件（历史文件记录目前的应力和位移大小）
根据增量步数量生成增量载荷的Deltaload.tmp文件
开始迭代
从IELEMT.TMP读入杆的目前弹塑性状态和相应模量
组装当前增量步的刚度矩阵
求解计算位移增量
更新位移向量的当前值，写入位移历史文件，同理更新单元应力历史文件，判断是否有新杆进入塑形，写入IELEMNT.TMP
循环次数是否够增量步数？
否
是
输出最终位移和应力
（输出增量步的位移增量，应力增量）

弹塑性杆分析

- 弹性初步试探步结果：

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	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
2	0.100000E-04	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+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	0.100000E+05	0.100000E+07

弹塑性杆分析

- 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       0.000000E+00  0.000000E+00  0.000000E+00  0.000000E+00  0.000000E+00  0.000000E+00
        2       0.100000E-07  0.000000E+00  0.000000E+00  0.000000E+00  0.000000E+00  0.000000E+00
```

位移增量

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

 ELEMENT              FORCE             STRESS
 NUMBER
    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       0.000000E+00  0.000000E+00  0.000000E+00  0.000000E+00  0.000000E+00  0.000000E+00
        2       0.100000E-06  0.000000E+00  0.000000E+00  0.000000E+00  0.000000E+00  0.000000E+00

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

 ELEMENT              FORCE             STRESS
 NUMBER
    1            0.102000E+04      0.102000E+06
```

目前应力总值

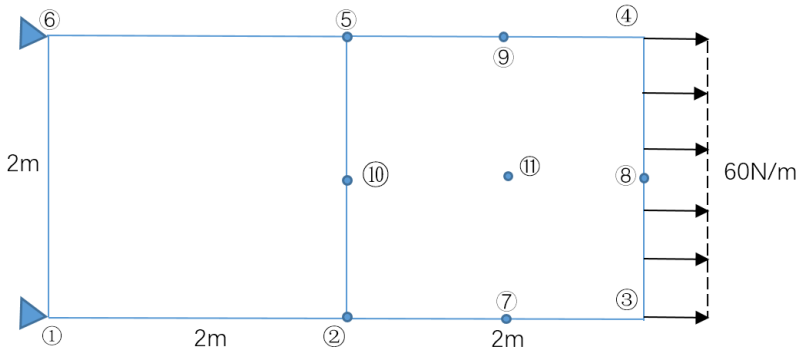
弹塑性杆分析

- 计算最终结果：

```
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       0.000000E+00  0.000000E+00  0.000000E+00  0.000000E+00  0.000000E+00  0.000000E+00
        2       0.909100E-04  0.000000E+00  0.000000E+00  0.000000E+00  0.000000E+00  0.000000E+00


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

 ELEMENT             STRESS
 NUMBER
    1             0.100000E+07
```

过渡单元

9Q过渡单元



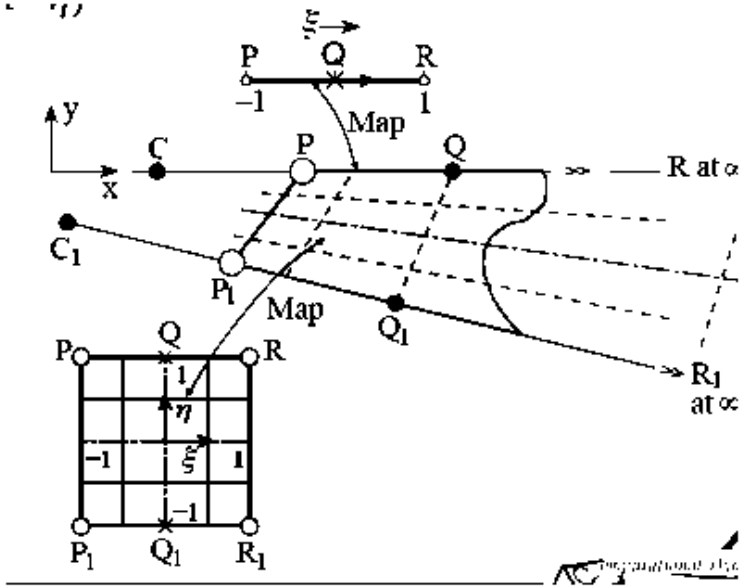
ELEMENT	X-CORRD	Y-CORRD	STRESS_XX	STRESS_YY	STRESS_XY
1	0.694318E-01	0.694318E-01	0.600000E+02	-0.193019E-07	-0.204958E-07
1	0.694318E-01	0.330009E+00	0.600000E+02	-0.227606E-07	-0.809190E-08
1	0.694318E-01	0.669991E+00	0.600000E+02	-0.227605E-07	0.809165E-08
1	0.694318E-01	0.930568E+00	0.600000E+02	-0.193017E-07	0.204955E-07
1	0.330009E+00	0.694318E-01	0.600000E+02	0.134719E-07	-0.262818E-07
1	0.330009E+00	0.330009E+00	0.600000E+02	0.100131E-07	-0.103762E-07
1	0.330009E+00	0.669991E+00	0.600000E+02	0.100132E-07	0.103761E-07
1	0.330009E+00	0.930568E+00	0.600000E+02	0.134720E-07	0.262817E-07
1	0.669991E+00	0.694318E-01	0.600000E+02	0.562324E-07	-0.338310E-07
1	0.669991E+00	0.330009E+00	0.600000E+02	0.527737E-07	-0.133566E-07
1	0.669991E+00	0.669991E+00	0.600000E+02	0.527737E-07	0.133567E-07
1	0.669991E+00	0.930568E+00	0.600000E+02	0.562325E-07	0.338310E-07
1	0.930568E+00	0.694318E-01	0.600000E+02	0.890062E-07	-0.396170E-07
1	0.930568E+00	0.330009E+00	0.600000E+02	0.855474E-07	-0.156409E-07
1	0.930568E+00	0.669991E+00	0.600000E+02	0.855474E-07	0.156411E-07
1	0.930568E+00	0.930568E+00	0.600000E+02	0.890062E-07	0.396171E-07

无限单元

程序设计思路

但是当设计到无限大的分析对象时，明显在无限远处的边界条件不好提供

于是我们对于形函数进行如下改动：



$$x = N_1(\eta)N_P(\xi)x_p + N_1(\eta)N_Q(\xi)x_Q + N_0(\eta)N_P(\xi)x_p + N_0(\eta)N_Q(\xi)x_Q$$

$$N_1 = (1+\eta)/2$$

$$N_0 = (1-\eta)/2$$

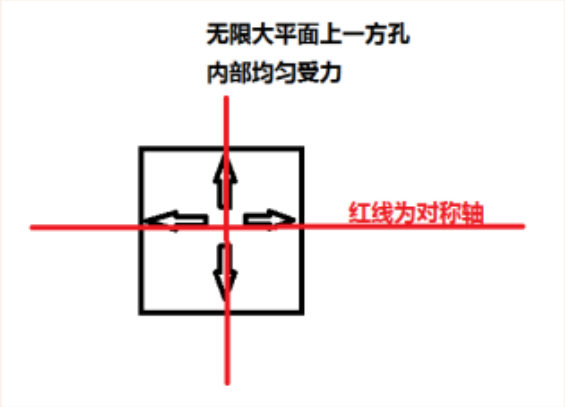
$$N_P = -(\frac{2\xi}{1-\xi})$$

$$N_Q = (1+\frac{2\xi}{1-\xi})$$

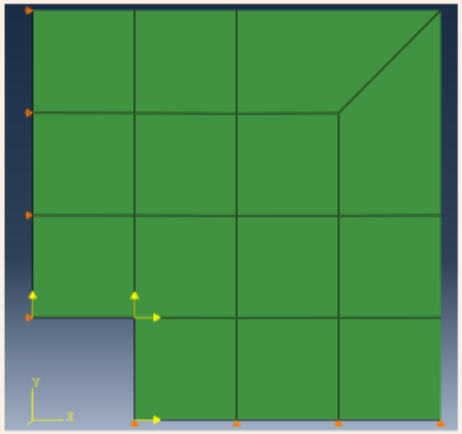
无限单元

实例验证

算一个如下实例：



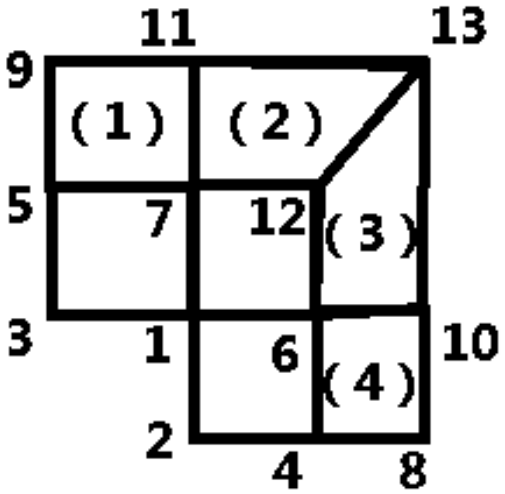
于是切成四分之一平面，在abaqus中进行如下网格划分：



无限单元

实例验证

而在使用STAP90划分时，运用如下的划分方法：



其中单元（1）、（2）、（3）、（4）使用的是无线单元，
其余为4Q单元。

无限单元

实例验证

最终位移计算结果如下：

Ux	2	7	8	6	17	19	24
Abaqus	5.45965E-09	1.28052E-09	1.44267E-09	0.92261E-09	2.05252E-09	1.03630E-09	1.05388E-09
采用无限单元	3.62256E-09	0.97195E-09	1.62307E-09	1.39779E-09	1.20308E-09	0.86413E-09	0.55678E-09
Ux	22	18	21	20	23		
Abaqus	0.50341E-09	1.00673E-09	0.19692E-09	0.77111E-09	0.49533E-09		
采用无限单元	0.69375E-09	0.61093E-09	0.22025E-09	0.32102E-09	0.38580E-09		
Uy	5	15	16	6	17	19	24
Abaqus	5.46315E-09	1.44497E-09	1.27938E-09	0.92150E-09	1.05233E-09	0.19520E-09	2.04885E-09
采用无限单元	3.62256E-09	1.62307E-09	0.97195E-09	1.39779E-09	0.55678E-09	0.22025E-09	1.20308E-09
Uy	22	18	21	20	23		
Abaqus	0.50346E-09	0.76915E-09	1.03866E-09	1.00645E-09	0.49553E-09		
采用无限单元	0.69375E-09	0.32102E-09	0.86413E-09	0.61093E-09	0.38580E-09		

可以看出，外围点位移不为0，无线原的算法明显优于有限元将最外围点锁死的算法，但是用无限原算法与abaqus实例中的解仍有一定差距，
预计是由于无限原的几个元素过于靠近原点所导致。

其余单元

- 除以上内容，本组还添加了3T、6T、4Q、9Q、8Q、27H、板单元（4节点和8节点）受限于时间不在此进行介绍

致谢

- 感谢张老师和宋言学长在整个项目完成工程中对本组全组同学的耐心指导。
- 感谢在部分任务中，其余组的部分同学与本组同学的讨论以及他们的帮助。

参考文献

- 张雄 王天舒 刘岩 著 《计算动力学（第2版）》
- *FEAST Eigenvalue Solver v3.0 User Guide* arXiv:1203.4031v3 [cs.MS]
- *VTK User's Guide* http://www.kitware.com/products/books/VTKUsersGuide.pdf, retrieved Dec. 26, 2016
- *Intel® Math Kernel Library Developer Reference*

谢谢大家！

ppt制作：朱子霖