

有限元课程论文

康金梁，郑蕴哲，司马锲，吉首瑞

2019 年 6 月 19 日

(清华大学航天航空学院，北京 100084)

1 引言

在本学期的有限元课程中，我们学习了有限元的相关知识。为了将所学知识应用于实际应用中，增强编程能力，同时满足课程的考核要求，我们以老师发布的 STAPpp 程序为基础，添加新的模块和功能，编写完成新版本的 STAPpp 程序。程序最后要求能够完成构建桥梁的有限元模拟。

并且在课程对最后，应用我们所学知识，完成了桥梁竞赛，设计自己的满足条件的桥梁，并且通过 STAPpp 程序模拟，对桥梁设计进行优化，以在评估中取得更好的效果。

在整个任务的完成过程中，离不开小组成员的付出，更离不开助教学长和老师的辛勤付出。感谢老师的悉心教导和学长加班加点的答疑与测试！

2 STAPpp 程序设计与实现

2.1 前处理

本次前处理采用 Python3.7 环境开发完成，调用了 numpy 数学库进行解析求解。考虑到本次 stapp 输入文件格式较为单一，而 ABAQUS 的 inp 文件中又由一系列关键字打头的 instance 组成，因此我们采用了较为简单的前处理设计思路。即编写一个解析器顺序读取 inp 文件中的各个关键字字头，再根据每一类关键字在解析器中定义一系列对应的类如 node, material, elementgroup 等来分别处理 inp 文件中的不同信息。我们在 ABAQUSparser 中定义了每个关键字对应类来读入数据再将数据传输到 calculate 文件中，完成单元的编号，节点的坐标旋转，合并，去重，材料截面转换等操作，再导入到 outputter 文件中以 stapp 要求进行输出。同时计算部分还可以由单元和节点数据和材料特性利用高斯积分计算体力，考虑到本次任务中重力以 loadcase 形式导入，该功能并没有起到作用。由于本次前处理结构思路较为简单，使得文件输出较为稳定，但也存在通用性不足的缺陷，当进行不同类型的转换时，需要补充定义更多关键词类，这给实际应用带来了较多困难。

2.2 Bar 单元

对 Bar 单元的处理上，除了在前一阶段的作业中，添加了 Gravity 函数来计算单元自重外，其余代码基本和源代码保持一致。

2.3 4Q 单元

4Q 单元由于已经作为作业出现，故在此不再赘述。详情请查看源代码与郑蕴哲的 4Q 设计报告。

2.4 3T 单元

2.4.1 3T 单元求解过程

3T-1 求形函数 N

引入面积坐标对求 3T 单元很有效

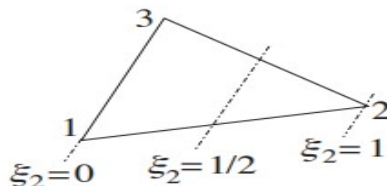


图 1: 面积坐标

通过面积坐标的公式:

$$\begin{bmatrix} 1 \\ x \\ y \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ x_1^e & x_2^e & x_3^e \\ y_1^e & y_2^e & y_3^e \end{bmatrix} \begin{bmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \end{bmatrix}$$

图 2: 变换公式

并且 N= 得到 B 函数如下:

$$\mathbf{B}^e = \frac{1}{2A^e} \begin{bmatrix} y_{23}^e & 0 & y_{31}^e & 0 & y_{12}^e & 0 \\ 0 & x_{32}^e & 0 & x_{13}^e & 0 & x_{21}^e \\ x_{32}^e & y_{23}^e & x_{13}^e & y_{31}^e & x_{21}^e & y_{12}^e \end{bmatrix}$$

图 3: B 矩阵

因此，通过公式可得到单元 K 矩阵:

$$\mathbf{K}^e = \int_{\Omega^e} \mathbf{B}^{eT} \mathbf{D} \mathbf{B}^e d\Omega = t^e \mathbf{A}^e \mathbf{B}^{eT} \mathbf{D} \mathbf{B}^e$$

图 4: 单元 K 矩阵

3T-2 外力矩阵

集中力仍然不需要修改（未进行坐标变换）体作用力公式如下：

$$\begin{aligned} \mathbf{f}_{\Omega}^e &= \int_{\Omega^e} \mathbf{N}^{eT} \mathbf{b} d\Omega & \mathbf{b} &= \sum_{I=1}^3 \mathbf{N}_I^{3T} \mathbf{b}_I = \mathbf{N}^e \mathbf{b}^e \\ &= \int_{\Omega^e} \begin{bmatrix} \mathbf{N}_1^{3T} & 0 \\ 0 & \mathbf{N}_1^{3T} \\ \mathbf{N}_2^{3T} & 0 \\ 0 & \mathbf{N}_2^{3T} \\ \mathbf{N}_3^{3T} & 0 \\ 0 & \mathbf{N}_3^{3T} \end{bmatrix} \sum_{I=1}^3 \mathbf{N}_I^{3T} \begin{bmatrix} b_{xI} \\ b_{yI} \end{bmatrix} d\Omega = \frac{A^e}{12} \begin{bmatrix} 2b_{x1} + b_{x2} + b_{x3} \\ 2b_{y1} + b_{y2} + b_{y3} \\ b_{x1} + 2b_{x2} + b_{x3} \\ b_{y1} + 2b_{y2} + b_{y3} \\ b_{x1} + b_{x2} + 2b_{x3} \\ b_{y1} + b_{y2} + 2b_{y3} \end{bmatrix} \end{aligned}$$

图 5: 体作用力公式

以及面作用力如下：（通过一点高斯积分获得）

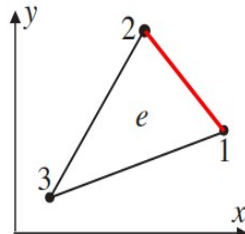
$$\begin{aligned} \mathbf{f}_{\Gamma}^e &= \int_{\Gamma^e} \mathbf{N}^{eT} \bar{\mathbf{t}} d\Gamma & N_1^{2L} &= 1-\xi, \quad N_2^{2L} = \xi & \bar{\mathbf{t}} &= (1-\xi)\bar{\mathbf{t}}_1 + \xi\bar{\mathbf{t}}_2 & d\Gamma &= l d\xi \\ &= \int_0^1 \begin{bmatrix} 1-\xi & 0 \\ 0 & 1-\xi \\ \xi & 0 \\ 0 & \xi \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} t_{x1}(1-\xi) + t_{x2}\xi \\ t_{y1}(1-\xi) + t_{y2}\xi \end{bmatrix} l d\xi = \frac{l}{6} \begin{bmatrix} 2t_{x1} + t_{x2} \\ 2t_{y1} + t_{y2} \\ t_{x1} + 2t_{x2} \\ t_{y1} + 2t_{y2} \\ 0 \\ 0 \end{bmatrix} \end{aligned}$$


图 6: 面作用力公式

如此获得了单元刚度阵与外力阵

3T-3 组装

按照事先划好的网格进行组装（LM 阵已知）

$$\mathbf{K} = \sum_{e=1}^{n_{el}} \mathbf{L}^{eT} \mathbf{K}^e \mathbf{L}^e, \quad \mathbf{f} = \sum_{e=1}^{n_{el}} \mathbf{L}^{eT} \mathbf{f}^e$$

图 7: 组装公式

3T-4 求解

最后通过矩阵的直接求解法（LDLT 分解法）求解。反作用力由以下公式求出：

$$\sum_{e=1}^{n_{el}} \mathbf{L}^{eT} \mathbf{F}^e = \mathbf{f} + \mathbf{r}$$

图 8: 反作用力公式

位移可直接求出。通过位移，按照以下公式可得应变、应力：

$$\boldsymbol{\varepsilon}^e(\xi_i, \eta_i) = \begin{bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \gamma_{xy} \end{bmatrix}_{(\xi_i, \eta_i)}^e = \mathbf{B}^e(\xi_i, \eta_i) \mathbf{d}^e \quad \boldsymbol{\sigma}^e(\xi_i, \eta_i) = \begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{bmatrix}_{(\xi_i, \eta_i)}^e = \mathbf{D}^e \boldsymbol{\varepsilon}^e(\xi_i, \eta_i)$$

图 9: 应变应力公式

按照以上思路完成 3T cpp 程序的写作。

2.4.2 Patch test 与收敛性分析

Patch Test 部分:

通过单项给定的线性位移场进行: 图形如下: (材料参数 E=1000, v=0.3)

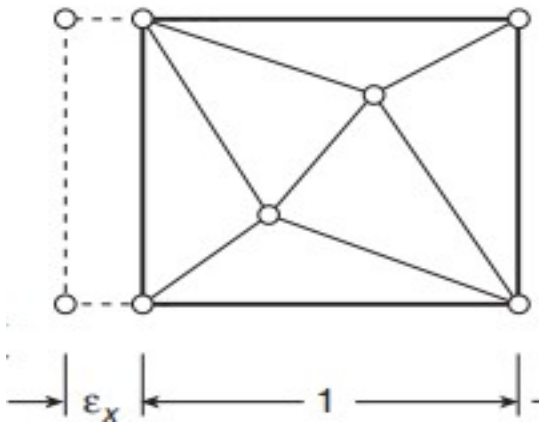


图 10: 测试图形

给定位移场为 (单向拉伸):

Consider the solution ($E = 1000, \nu = 0.3$)
$$\begin{matrix} u = 0.002x \\ v = -0.0006y \end{matrix} \Rightarrow \begin{matrix} b = 0 \\ \sigma_x = 2 \end{matrix}$$

图 11: 线性位移

输入参数设定顺时针, 且固定左下点与左上 x 方向):
获得的位移结果为:

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	-6.00000e-03	0.00000e+00
3	2.00000e-02	-6.00000e-03	0.00000e+00
4	2.00000e-02	1.92988e-17	0.00000e+00
5	4.00000e-03	-2.40000e-03	0.00000e+00
6	1.40000e-02	-4.80000e-03	0.00000e+00

图 12: 位移结果

可见误差项为 10-17 量级，属于机器误差范围，因此模型通过了 Patch test. 故本程序是收敛的。
收敛率分析部分

对左下角固定的正方形材料进行拉伸: (E=1000,v=0.03)

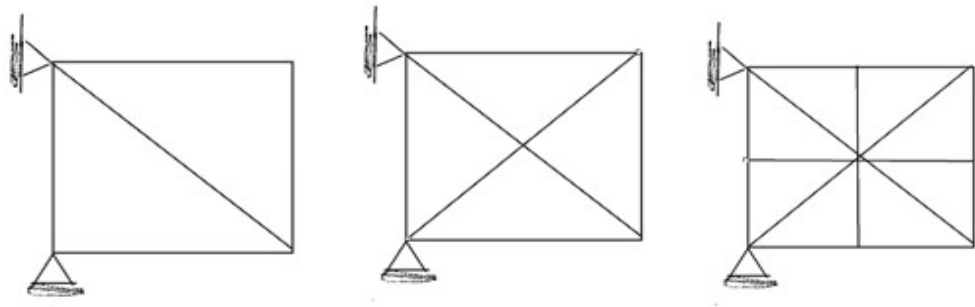


图 13: 加密图形

采取细分网格如图，单元给定线性位移场 $u=0.002x$ 单向拉伸的正方形单元的收敛率分析：精确解的：普通单元的常数应变为：

$E_x=0.02$
 $E_y=E_z=-0.006$

$$||e|| = \frac{1}{2} E \sum (\varepsilon_{ex} - \varepsilon_h)^2 A_i$$

图 14: 计算能量范数下误差公式

a.2 单元

STRESS CALCULATIONS FOR ELEMENT GROUP 1							
ELEMENT NUMBER	GAUSS POINT	X-COORD	Y-COORD	Z-COORD	SXX	SYX	TXY
1	1	0.00000e+00	0.00000e+00	0.00000e+00	-1.94407e+01	-5.83222e+00	-5.59254e-01
1	2	0.00000e+00	1.00000e+00	1.00000e+00	-1.94407e+01	-5.83222e+00	-5.59254e-01
1	3	1.00000e+00	0.00000e+00	0.00000e+00	-1.94407e+01	-5.83222e+00	-5.59254e-01
2	1	1.00000e+00	0.00000e+00	1.00000e+00	-2.05593e+01	-5.59254e-01	5.59254e-01
2	2	0.00000e+00	1.00000e+00	1.00000e+00	-2.05593e+01	-5.59254e-01	5.59254e-01
2	3	1.00000e+00	1.00000e+00	0.00000e+00	-2.05593e+01	-5.59254e-01	5.59254e-01

图 15: 2 单元应力数据

得到应变为：
element Ex Ey Ez
[1] 0.01944 -0.00583 -0.00559
[2] 0.02056 -0.00559 -0.00559
ABS(e)=0.0328
b.4 单元

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		1.93466e-02	-3.82214e-03	0.00000e+00			
4		1.93466e-02	3.82214e-03	0.00000e+00			
5		8.75318e-03	2.45671e-19	0.00000e+00			
S T R E S S C A L C U L A T I O N S F O R E L E M E N T G R O U P 1							
ELEMENT NUMBER	GAUSS POINT	X-COORD	Y-COORD	Z-COORD	SXX	SYX	TXY
1	1	0.00000e+00	0.00000e+00	0.00000e+00	-1.92377e+01	-5.77132e+00	-1.88978e-16
1	2	0.00000e+00	1.00000e+00	1.00000e+00	-1.92377e+01	-5.77132e+00	-1.88978e-16
1	3	5.00000e-01	5.00000e-01	0.00000e+00	-1.92377e+01	-5.77132e+00	-1.88978e-16
2	1	0.00000e+00	0.00000e+00	0.00000e+00	-2.00000e+01	-2.17786e+00	7.62250e-01
2	2	1.00000e+00	1.00000e+00	0.00000e+00	-2.00000e+01	-2.17786e+00	7.62250e-01
2	3	5.00000e-01	5.00000e-01	0.00000e+00	-2.07623e+01	1.41561e+00	-8.67201e-16
3	1	0.00000e+00	0.00000e+00	1.00000e+00	-2.07623e+01	1.41561e+00	-8.67201e-16
3	2	1.00000e+00	1.00000e+00	0.00000e+00	-2.07623e+01	1.41561e+00	-8.67201e-16
3	3	5.00000e-01	5.00000e-01	0.00000e+00	-2.00000e+01	-2.17786e+00	7.62250e-01
4	1	0.00000e+00	0.00000e+00	1.00000e+00	-2.00000e+01	-2.17786e+00	7.62250e-01
4	2	0.00000e+00	0.00000e+00	0.00000e+00	-2.00000e+01	-2.17786e+00	7.62250e-01
4	3	5.00000e-01	5.00000e-01	0.00000e+00	-2.00000e+01	-2.17786e+00	7.62250e-01

图 16: 4 单元应力数据

element Ex Ey Ez

[1] 0.01924 -0.00577 0

[2] 0.02 -0.00217 0.00762

[4] 0.02076 0.001415 0

[4] 0.02 -0.00217 -0.00762

ABS(e)=0.0162

c.8 单元

S T R E S S C A L C U L A T I O N S F O R E L E M E N T G R O U P 1							
ELEMENT NUMBER	GAUSS POINT	X-COORD	Y-COORD	Z-COORD	SXX	SYX	TXY
1	1	0.00000e+00	0.00000e+00	0.00000e+00	-1.82830e+01	-5.48490e+00	-6.50704e-15
1	2	0.00000e+00	5.00000e-01	5.00000e-01	-1.82830e+01	-5.48490e+00	-6.50704e-15
1	3	5.00000e-01	5.00000e-01	0.00000e+00	-1.82830e+01	-5.48490e+00	-6.50704e-15
2	1	0.00000e+00	5.00000e-01	0.00000e+00	-1.82830e+01	-5.48490e+00	-6.50704e-15
2	2	0.00000e+00	1.00000e+00	1.00000e+00	-1.82830e+01	-5.48490e+00	-6.50704e-15
2	3	5.00000e-01	5.00000e-01	0.00000e+00	-1.82830e+01	-5.48490e+00	-6.50704e-15
3	1	0.00000e+00	5.00000e-01	5.00000e-01	-2.17130e+01	-7.58596e-01	3.48997e-01
3	2	0.00000e+00	1.00000e+00	1.00000e+00	-2.17130e+01	-7.58596e-01	3.48997e-01
3	3	5.00000e-01	1.00000e+00	0.00000e+00	-2.17130e+01	-7.58596e-01	3.48997e-01
4	1	1.00000e+00	1.00000e+00	1.00000e+00	-2.28833e+01	-1.10968e+00	-1.51928e+00
4	2	5.00000e-01	5.00000e-01	5.00000e-01	-2.28833e+01	-1.10968e+00	-1.51928e+00
4	3	5.00000e-01	1.00000e+00	0.00000e+00	-2.28833e+01	-1.10968e+00	-1.51928e+00
5	1	1.00000e+00	5.00000e-01	1.00000e+00	-1.71127e+01	1.51928e+00	-3.78070e+00
5	2	5.00000e-01	5.00000e-01	1.00000e+00	-1.71127e+01	1.51928e+00	-3.78070e+00
5	3	1.00000e+00	1.00000e+00	0.00000e+00	-1.71127e+01	1.51928e+00	-3.78070e+00
6	1	1.00000e+00	0.00000e+00	1.00000e+00	-1.71127e+01	1.51928e+00	-3.78070e+00
6	2	5.00000e-01	5.00000e-01	5.00000e-01	-1.71127e+01	1.51928e+00	-3.78070e+00
6	3	1.00000e+00	5.00000e-01	0.00000e+00	-1.71127e+01	1.51928e+00	-3.78070e+00
7	1	1.00000e+00	0.00000e+00	1.00000e+00	-2.28833e+01	-1.10968e+00	-1.51928e+00
7	2	5.00000e-01	0.00000e+00	0.00000e+00	-2.28833e+01	-1.10968e+00	-1.51928e+00
7	3	5.00000e-01	5.00000e-01	0.00000e+00	-2.28833e+01	-1.10968e+00	-1.51928e+00
8	1	0.00000e+00	0.00000e+00	5.00000e-01	-2.17130e+01	-7.58596e-01	3.48997e-01
8	2	0.00000e+00	0.00000e+00	0.00000e+00	-2.17130e+01	-7.58596e-01	3.48997e-01
8	3	5.00000e-01	5.00000e-01	0.00000e+00	-2.17130e+01	-7.58596e-01	3.48997e-01

图 17: 8 单元应力数据

element Ex Ey Ez

[1] V -0.00548 0

[2] 0.0183 -0.00548 0

[3] 0.0217 -0.000758 0.000349

[4] 0.0218 -0.00111 -0.00152

[5] 0.0191 0.00152 -0.00378

[6] 0.0191 0.00152 0.00378

[7] 0.0218 -0.00111 0.00152

[8] 0.0217 -0.000758 -0.000349

ABS(e)=0.0090

对结果进行线性拟合 (e 的对数) 得到:

P=1.11

故最后该 3T 单元的收敛率为 1.11，大于 1 因此收敛（通过 Patch Test 已经收敛。）

2.5 8H 单元

在 8H 单元的构建中，我们从高级单元的基本构造方法出发。考虑到 8H 单元是线性单元，我们在 4Q 单元的位移函数上拓展得到 8H 单元的位移函数，引入 ψ, η, ζ 作为母单元坐标，由于 8H 单元有 24 个自由度，我们在三维二阶完备多项式中去除三项平方项再加入一个三次项得到 8H 单元位移函数如下。

$$\begin{aligned} u &= \alpha_1 + \alpha_2\psi + \alpha_3\eta + \alpha_4\zeta + \alpha_5\psi\eta + \alpha_6\eta\zeta + \alpha_7\psi\zeta + \alpha_8\psi\eta\zeta, \\ v &= \alpha_9 + \alpha_{10}\psi + \alpha_{11}\eta + \alpha_{12}\zeta + \alpha_{13}\psi\eta + \alpha_{14}\eta\zeta + \alpha_{15}\psi\zeta + \alpha_{16}\psi\eta\zeta, \\ w &= \alpha_{17} + \alpha_{18}\psi + \alpha_{19}\eta + \alpha_{20}\zeta + \alpha_{21}\psi\eta + \alpha_{22}\eta\zeta + \alpha_{23}\psi\zeta + \alpha_{24}\psi\eta\zeta. \end{aligned}$$

随后，由以下形式的形函数进行插值：

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

继而由

$$B_i = \nabla_s N_i$$

得到应变矩阵

$$B_i = \begin{pmatrix} \frac{\partial N_i}{\partial x} & 0 & 0 \\ 0 & \frac{\partial N_i}{\partial y} & 0 \\ 0 & 0 & \frac{\partial N_i}{\partial z} \\ \frac{\partial N_i}{\partial y} & \frac{\partial N_i}{\partial x} & 0 \\ 0 & \frac{\partial N_i}{\partial z} & \frac{\partial N_i}{\partial y} \\ \frac{\partial N_i}{\partial z} & 0 & \frac{\partial N_i}{\partial x} \end{pmatrix}.$$

其中， $B = [B_1 \ B_2 \ B_3 \ B_4 \ B_5 \ B_6 \ B_7 \ B_8]$ 。再通过 Jacobi 矩阵将形函数映射到物理坐标系中得到单元刚度阵：

$$K^e = \int_{-1}^1 \int_{-1}^1 \int_{-1}^1 B^T DB |J| d\psi d\eta d\zeta.$$

最后组装得到总体刚度阵，得到系统方程。为了验证所构造的 8H 单元，我们采用一个线性场的分片试验进行 test C。由于线性位移场实质上只包括常正应变与常剪应变两种情形，因此我们下面分别构造了单轴拉伸与纯剪载荷的算例进行验证。分片试验由 7 个单元组成，单元划分方式如图18所示。

首先是单轴拉伸的分片试验，施加如图2.5所示的载荷。相应的位移场精确解应为：

$$\begin{aligned} u_x &= -\nu \frac{4P}{El_z l_x} x, \\ u_y &= -\nu \frac{4P}{El_z l_y} y, \\ u_z &= \frac{4P}{El_z^2} z. \end{aligned}$$

与图20所示的计算值对比，可以看到符合得很好，这说明所构造的 8H 单元具有至少二阶精度。实际上，由理论分析易知，8H 单元确实只具有二阶精度。考虑到分片试验是单轴拉伸，再构造一个纯剪的算例，加载方式如图2.5所示。进而得到了如图21所示的计算结果，这一结果与理论解符合得很好。这里的 8H 单元使用的是应力平滑（即 Gauss 点的应力外推）后的应力。

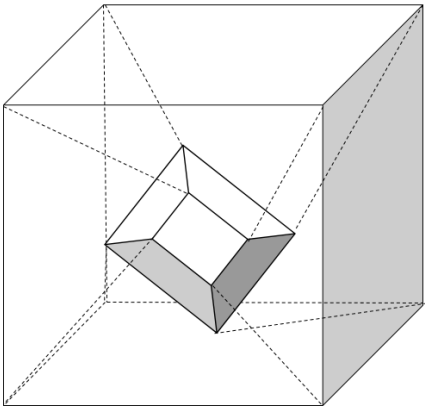


图 18: 分片试验单元划分示意图

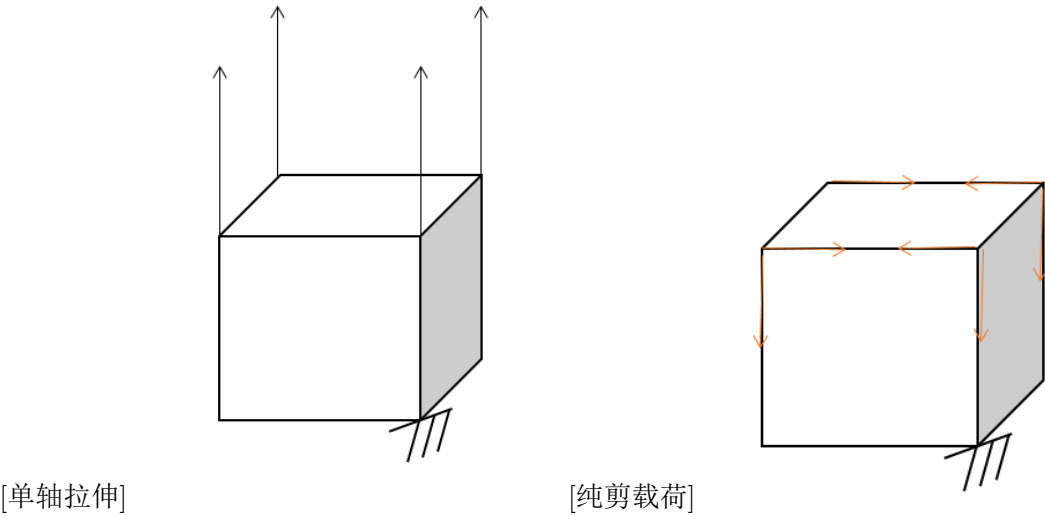


图 19: 分片试验载荷与约束示意图

D I S P L A C E M E N T S			
NODE	X-DISPLACEMENT	Y-DISPLACEMENT	Z-DISPLACEMENT
1	-4.00000e-002	0.00000e+000	0.00000e+000
2	-4.00000e-002	-4.00000e-002	0.00000e+000
3	2.08167e-017	-4.00000e-002	0.00000e+000
4	0.00000e+000	0.00000e+000	0.00000e+000
5	-4.00000e-002	-9.02056e-017	2.00000e-001
6	-4.00000e-002	-4.00000e-002	2.00000e-001
7	-1.45717e-016	-4.00000e-002	2.00000e-001
8	-2.05630e-016	-4.85723e-017	2.00000e-001
9	-2.40000e-002	-8.00000e-003	4.00000e-002
10	-2.40000e-002	-2.40000e-002	8.00000e-002
11	-8.00000e-003	-2.40000e-002	4.00000e-002
12	-8.00000e-003	-8.00000e-003	4.00000e-002
13	-2.40000e-002	-8.00000e-003	1.20000e-001
14	-2.40000e-002	-2.40000e-002	1.20000e-001
15	-8.00000e-003	-2.40000e-002	1.20000e-001
16	-8.00000e-003	-8.00000e-003	1.20000e-001

图 20: 单轴拉伸位移场计算结果

D I S P L A C E M E N T S			
NODE	X-DISPLACEMENT	Y-DISPLACEMENT	Z-DISPLACEMENT
1	0.00000e+000	0.00000e+000	0.00000e+000
2	0.00000e+000	0.00000e+000	0.00000e+000
3	0.00000e+000	0.00000e+000	0.00000e+000
4	0.00000e+000	0.00000e+000	0.00000e+000
5	0.00000e+000	2.40000e+000	4.99600e-016
6	0.00000e+000	2.40000e+000	-5.37672e-016
7	-3.33067e-016	2.40000e+000	-2.70884e-016
8	-1.11022e-016	2.40000e+000	-1.01581e-016

图 21: 纯剪载荷位移场计算结果

2.6 Beam 单元

由于 Beam 单元与 Bar 单元十分类似, 因此在 Beam 单元实现的过程中很多地方都借鉴了 Bar 单元的思路和细节。

首先 Beam 的刚度阵如下图所示

$$\begin{bmatrix}
 \frac{EA}{l} & 0 & 0 & 0 & 0 & 0 & -\frac{EA}{l} & 0 & 0 & 0 & 0 & 0 \\
 0 & \frac{12EJ_z}{l^3} & 0 & 0 & 0 & 0 & 0 & \frac{12EJ_z}{l^3} & 0 & 0 & 0 & 0 \\
 0 & 0 & \frac{12EJ_y}{l^3} & 0 & 0 & 0 & 0 & 0 & \frac{12EJ_y}{l^3} & 0 & 0 & 0 \\
 0 & 0 & 0 & \frac{GJ_x}{l} & 0 & 0 & 0 & 0 & 0 & \frac{GJ_x}{l} & 0 & 0 \\
 0 & 0 & \frac{-6EJ_y}{l^2} & 0 & \frac{4EJ_z}{l} & 0 & 0 & \frac{-6EJ_y}{l^2} & 0 & \frac{4EJ_z}{l} & 0 & 0 \\
 0 & \frac{6EJ_z}{l^2} & 0 & 0 & 0 & \frac{4EJ_x}{l} & 0 & \frac{6EJ_z}{l^2} & 0 & 0 & \frac{4EJ_x}{l} & 0 \\
 -\frac{EA}{l} & 0 & 0 & 0 & 0 & 0 & \frac{EA}{l} & 0 & 0 & 0 & 0 & 0 \\
 0 & \frac{-12EJ_z}{l^3} & 0 & 0 & 0 & \frac{-6EJ_x}{l^2} & 0 & 0 & \frac{12EJ_z}{l^3} & 0 & 0 & 0 \\
 0 & 0 & \frac{-12EJ_y}{l^3} & 0 & \frac{6EJ_z}{l^2} & 0 & 0 & 0 & 0 & \frac{12EJ_y}{l^3} & 0 & 0 \\
 0 & 0 & 0 & \frac{-GJ_x}{l} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{GJ_x}{l} & 0 \\
 0 & 0 & \frac{-6EJ_z}{l^2} & 0 & \frac{2EJ_x}{l} & 0 & 0 & 0 & \frac{6EJ_z}{l^2} & 0 & \frac{4EJ_y}{l} & 0 \\
 0 & \frac{6EJ_x}{l^2} & 0 & 0 & 0 & \frac{2EJ_z}{l} & 0 & \frac{-6EJ_x}{l^2} & 0 & 0 & 0 & \frac{4EJ_z}{l}
 \end{bmatrix}$$

图 22: Beam 刚度阵

对于 Beam 单元, 类似 Bar 单元, 我们可以写出其位移与外力形式。

$$d^e = \begin{pmatrix} u_{x1} & u_{y1} & u_{z1} & \theta_{x1} & \theta_{y1} & \theta_{z1} & u_{x2} & u_{y2} & u_{z2} & \theta_{x2} \\ \theta_{y2} & \theta_{y3} \end{pmatrix}^T$$

$$f^e = \begin{pmatrix} F_{x1} & F_{y1} & F_{z1} & M_{x1} & M_{y1} & M_{z1} & F_{x2} & F_{y2} & F_{z2} & M_{x2} \\ M_{y2} & M_{y3} \end{pmatrix}^T$$

因而我们得到了刚度方程实现的具体细节。当然为了能够在三维计算, 我们还需要对杆进行一个旋转变换。

材料的输入属性包括弹性模量和泊松比。由于很多情况下梁是空心的，因此我们除了长、宽外还允许输出内部空心矩阵的参数。具体请参照源代码。

2.7 Plate 单元

2.7.1 单元设计思路

plate 单元是一种一个方向的尺度远小于另外两个方向尺度的单元。在本 STAPpp 程序设计中，采用了薄板假设，即 Kirchhoff-love 理论，包含平截面假设和中性面垂直假设。根据以上假设，我们可以得到平板弯曲的结果，

$$\boldsymbol{\varepsilon} = \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{Bmatrix} = z \begin{Bmatrix} -\frac{\partial^2 w}{\partial x^2} \\ -\frac{\partial^2 w}{\partial y^2} \\ -2\frac{\partial^2 w}{\partial x \partial y} \end{Bmatrix} = z \mathbf{L} w = z \boldsymbol{\kappa}$$

其中，矩阵 \mathbf{L} 为，

$$\mathbf{L} = \begin{Bmatrix} -\frac{\partial^2 w}{\partial x^2} \\ -\frac{\partial^2 w}{\partial y^2} \\ -2\frac{\partial^2 w}{\partial x \partial y} \end{Bmatrix}$$

然后，因为平板单元含有四个节点，共有十二个方程，所以我们取帕斯卡三角形中前十二项，得到 w 的表达式，代入后，得到 \mathbf{L} 的表达式为，

$$\mathbf{L} w = \begin{Bmatrix} 2\alpha_4 + 6\alpha_7 x + 2\alpha_8 y + 6\alpha_1 xy \\ 2\alpha_6 + 2\alpha_9 x + 6\alpha_1 0y + 6\alpha_1 2xy \\ 2\alpha_5 + 4\alpha_8 x + 4\alpha_9 y + 6\alpha_1 1x^2 + 6\alpha_1 2y^2 \end{Bmatrix}$$

且

$$\mathbf{L} w = \mathbf{Q} \boldsymbol{\alpha}^e = \mathbf{Q} \mathbf{M}^{-1} \mathbf{d}^e$$

则可以得到矩阵 \mathbf{Q} 的表达式为，

$$\mathbf{Q} = \begin{bmatrix} 0 & 0 & 0 & 2 & 0 & 0 & 6x & 2y & 0 & 0 & 6xy & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 2x & 6y & 0 & 6xy \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 4x & 4y & 0 & 6x^2 & 6y^2 \end{bmatrix} \quad (1)$$

这样我们就可以得到单元刚度阵的积分表达式，为

$$\mathbf{K}^e = \int_{\Omega^e} \mathbf{B}^T \mathbf{D} \mathbf{B} dx dy = \mathbf{M}^{-T} \left(\int_{\Omega^e} \mathbf{Q}^T \mathbf{D} \mathbf{Q} dx dy \right) \mathbf{M}^{-1}$$

同理，我们可以按照和之前类似的组装方式得到系统刚度阵和系统的应力矩阵，然后代入总的方程求解，即可得出相应的位移场结果。

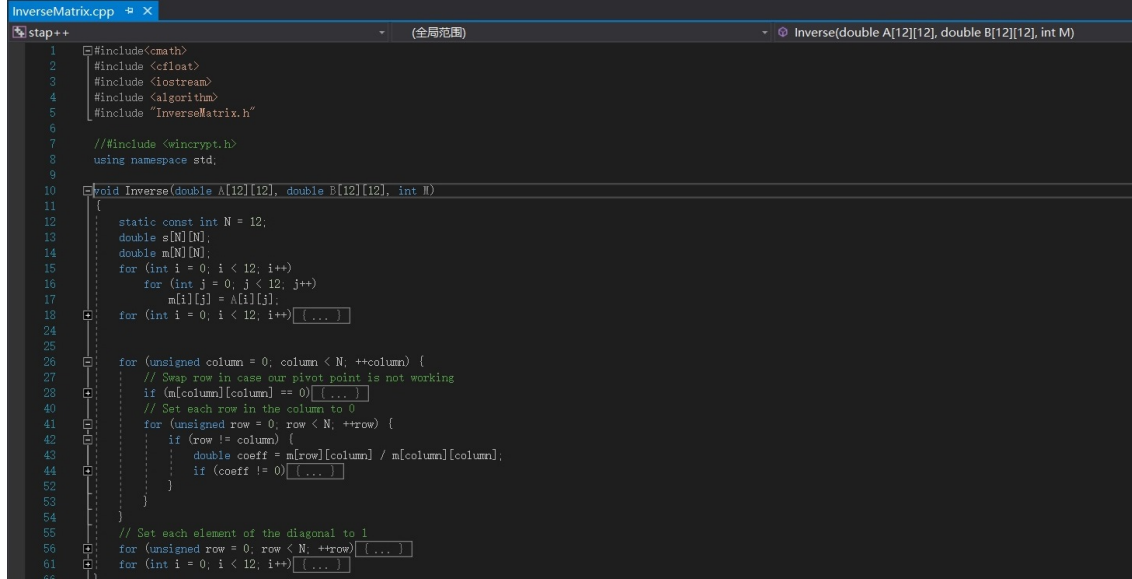
接下来较为重要和难以处理的是自由度问题。严格来讲，三维问题每个节点应有六个自由度，分别为三个平动自由度和三个转动自由度。但是由于 plate 单元中不存在围绕 z 轴的转动，从而第三个转动自由度是不存在的，所以每个节点应该开放五个自由度。然而考虑到嵌入 STAPpp 程序的需要，同意默认只开放三个平动自由度，在随后从前处理文件中读取节点信息时，再将两个转动自由度打开。

2.7.2 程序实现

但是在程序实现的过程中,我们发现,像之前单元一样利用 MATLAB 计算单元刚度阵解析表达式的方法已经不再适用。原因是 \mathbf{M} 过于复杂,无法在可期的时间内求出其逆的解析表达式。针对这点问题,在接下来的两个星期内,我提出了三种算法。

算法一

考虑到无法求得解析表达式,我们采用数值积分的方法求出每一个单元的数值的刚度矩阵。为了尽可能地提高运算速度,降低不能求解解析表达式所带来的计算复杂度,我们在 STAPpp 程序中添加了“InverseMatrix”类,用于矩阵求逆。该算法利用主元的 Gauss 方法求解矩阵逆的数值解。



```

1 #include <cmath>
2 #include <float>
3 #include <iostream>
4 #include <algorithm>
5 #include "InverseMatrix.h"
6
7 // #include <wincrypt.h>
8 using namespace std;
9
10 void Inverse(double A[12][12], double B[12][12], int M)
11 {
12     static const int N = 12;
13     double s[N][N];
14     double m[N][N];
15     for (int i = 0; i < 12; i++)
16         for (int j = 0; j < 12; j++)
17             m[i][j] = A[i][j];
18     for (int i = 0; i < 12; i++) {
19         // ...
20     }
21     for (unsigned column = 0; column < N; ++column) {
22         // Swap row in case our pivot point is not working
23         if (m[column][column] == 0) {
24             // ...
25         }
26         // Set each row in the column to 0
27         for (unsigned row = 0; row < N; ++row) {
28             if (row != column) {
29                 double coeff = m[row][column] / m[column][column];
30                 if (coeff != 0) {
31                     // ...
32                 }
33             }
34         }
35     }
36     // Set each element of the diagonal to 1
37     for (unsigned row = 0; row < N; ++row) {
38         // ...
39     }
40     for (int i = 0; i < 12; i++) {
41         // ...
42     }
43 }

```

图 23: 第一种算法

但是在完成后的验证过程中,我们发现,随着求解规模的不断增大,不断进行求逆所消耗的时间也大大增加,严重降低了程序的效率。所以在此基础上,我们编写了第二种算法。

算法二

通过查阅资料,我们发现了一种可以绕过求解矩阵 \mathbf{M} 的逆的方法。该方法的核心思想和之前 4Q 单元类似,通过坐标变换引入 ξ, η 坐标,然后给出相应的 \mathbf{N} 矩阵的表达式,通过对 \mathbf{N} 矩阵进行二阶求导,从而得出单元刚度阵的解析表达式。主要的计算过程如下,

形函数矩阵

$$\mathbf{N} = \begin{bmatrix} N_1 & \dots & N_{yn} \\ \frac{\partial N_1}{\partial y} & \dots & \frac{\partial N_{yn}}{\partial y} \\ \frac{\partial N_1}{\partial x} & \dots & \frac{\partial N_{yn}}{\partial x} \end{bmatrix}$$

其中形函数的表达式为,

$$N_i = \frac{1}{8}(1 + \xi_i \xi)(1 + \eta_i \eta)(2 + \xi_i \xi + \eta_i \eta - \xi^2 - \eta^2)$$

$$N_{xi} = -\frac{1}{8}b\eta_i(1 + \xi_i \xi)(1 + \eta_i \eta)(1 - \eta^2)$$

$$N_{yi} = \frac{1}{8}a\eta_i(1 + \xi_i \xi)(1 + \eta_i \eta)(1 - \xi^2)$$

这样，通过对 N 矩阵求二阶导，可以得到矩阵 B 的表达式，代入积分中，即可求得单元刚度阵的解析表达式。

```

599 void CPlate::ElementStiffness(double* Matrix)
600 {
601     clear(Matrix, sizeofStiffnessMatrix());
602
603     //calculate Ke
604     double B[3][12];
605     double D[3][12]; //不含常系数的D矩阵
606
607     double d = 0; //真正的D前的常数
608     double BD[12][3]; //B' * D 的中间项
609     double Ke[12][12]; //未积分的Ke
610     double A0[3][12];
611
612     for (int i = 0; i < 3; i++)
613         for (int j = 0; j < 12; j++)
614             A0[i][j] = 0;
615
616     for (int i = 0; i < 3; i++)
617         for (int j = 0; j < 12; j++)
618             B[i][j] = 0;
619     for (int i = 0; i < 3; i++)
620         for (int j = 0; j < 12; j++)
621             D[i][j] = 0;
622
623     //BD and Ke置零
624     for (int i = 0; i < 12; i++)
625         for (int j = 0; j < 3; j++)
626             BD[i][j] = 0;
627     for (int i = 0; i < 12; i++)
628         for (int j = 0; j < 12; j++)
629             Ke[i][j] = 0;
630
631     //D matrix
632     CPlateMaterial* material_ = dynamic_cast<CPlateMaterial*>(ElementMaterial_); // Pointer to material of the element
633     double E = material_>E; //板单元E
634     double nu = material_>nu; //板单元nu

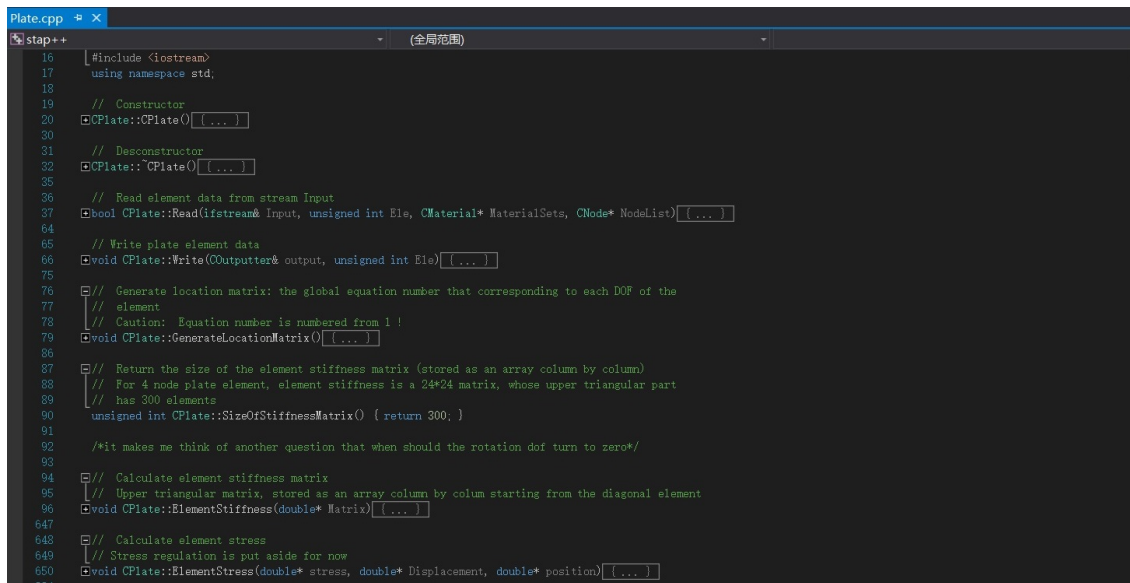
```

图 24: 第二种算法

我们可以发现，4Q 单元算法之所以能够对任意的（凸四边形）单元成立，主要原因就是引入了坐标变换。那么，将左边变换引入 plate 单元后，plate 单元算法能否和 4Q 单元一样，可以适用于任意形状的 plate 单元呢？在进行数学推导后，我们发现，原本在变换求导过程中出现的系数在后面逆变换中得到了抵消，简单地从推导过程中看，是可以应用到任意四边形形状的 plate 单元中的。但是在之后的 patch test 中，我们发现，对于特定形状的 plate 单元，计算结果出现“NAN”的情况。分析其原因，可能是因为坐标变换的雅克比矩阵并非常数，甚至连多项式都不是，程序无法正确计算。于是我推出了第三种算法。

算法三

第三种算法主要基于第二种算法，保留了矩阵求逆的解析解，保证了程序的运算效率。同时，不再追求能够求解任意的四边形 plate 单元，仅仅针对标准矩形单元和十分规则的平行四边形单元。经过验证，满足了 STAPpp 程序要求，可以用于实际求解中。



```

16 #include <iostream>
17 using namespace std;
18
19 // Constructor
20 CPlate::CPlate() {...}
21
22 // Destructor
23 CPlate::~CPlate() {...}
24
25 // Read element data from stream input
26 bool CPlate::Read(IStream& Input, unsigned int Ele, CMaterial* MaterialSets, CNode* NodeList) {...}
27
28 // Write plate element data
29 void CPlate::Write(OStream& output, unsigned int Ele) {...}
30
31 // Generate location matrix: the global equation number that corresponding to each DOF of the
32 // element
33 // Caution: Equation number is numbered from 1 !
34 void CPlate::GenerateLocationMatrix() {...}
35
36 // Return the size of the element stiffness matrix (stored as an array column by column)
37 // For 4 node plate element, element stiffness is a 24*24 matrix, whose upper triangular part
38 // has 300 elements
39 unsigned int CPlate::SizeOfStiffnessMatrix() { return 300; }
40
41 /*it makes me think of another question that when should the rotation dof turn to zero*/
42
43 // Calculate element stiffness matrix
44 // Upper triangular matrix, stored as an array column by column starting from the diagonal element
45 void CPlate::ElementStiffness(double* Matrix) {...}
46
47 // Calculate element stress
48 // Stress regulation is put aside for now
49 void CPlate::ElementStress(double* stress, double* Displacement, double* position) {...}
50
51

```

图 25: 最终算法 cpp 文件结构

2.7.3 patch test 及收敛性分析

之后，我们对最终版本的 plate 单元进行了 patch test。较为复杂的 patch test 采用了四乘四共计十六个单元，如下图所示，

```
patch.dat  + ×
1  A_patch_test_of_plate
2  16 1 1 1
3  1 1 1 0 0 0 1 -1.0 -1.0 0.0
4  2 1 1 0 0 0 1 0.0 -1.0 0.0
5  3 1 1 0 0 0 1 0.3 -1.0 0.0
6  4 1 1 0 0 0 1 1.0 -1.0 0.0
7  5 1 1 0 0 0 1 -1.0 -0.2 0.0
8  6 1 1 0 0 0 1 0.0 -0.2 0.0
9  7 1 1 0 0 0 1 0.3 -0.2 0.0
10 8 1 1 0 0 0 1 1.0 -0.2 0.0
11 9 1 1 0 0 0 1 -1.0 0.0 0.0
12 10 1 1 1 1 1 1 0.0 0.0 0.0
13 11 1 1 0 0 0 1 0.3 0.0 0.0
14 12 1 1 0 0 0 1 1.0 0.0 0.0
15 13 1 1 0 0 0 1 -1.0 1.0 0.0
16 14 1 1 0 0 0 1 0.0 1.0 0.0
17 15 1 1 0 0 0 1 0.3 1.0 0.0
18 16 1 1 0 0 0 1 1.0 1.0 0.0
19 1 8
20 1 5 1.6
21 5 5 2
22 9 5 2.4
23 13 5 2
24 4 5 -1.6
25 8 5 -2
26 12 5 -2.4
27 16 5 -2
28 6 9 1
29 1 24 0.2 1
30 1 1 2 6 5 1
31 2 2 3 7 6 1
32 3 3 4 8 7 1
33 4 5 6 10 9 1
34 5 6 7 11 10 1
35 6 7 8 12 11 1
36 7 9 10 14 13 1
37 8 10 11 15 14 1
38 9 11 12 16 15 1
```

图 26: plate 单元 patch test

得到结果为，

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	8.00000e-01	
4.00000e-01	2.00000e+00	0.00000e+00		
2	0.00000e+00	0.00000e+00	-2.00000e-01	
4.00000e-01	5.33532e-14	0.00000e+00		
3	0.00000e+00	0.00000e+00	-1.10000e-01	
4.00000e-01	-6.00000e-01	0.00000e+00		
4	0.00000e+00	0.00000e+00	8.00000e-01	
4.00000e-01	-2.00000e+00	0.00000e+00		
5	0.00000e+00	0.00000e+00	9.92000e-01	
8.00000e-02	2.00000e+00	0.00000e+00		
6	0.00000e+00	0.00000e+00	-8.00000e-03	
8.00000e-02	3.06699e-14	0.00000e+00		
7	0.00000e+00	0.00000e+00	8.20000e-02	
8.00000e-02	-6.00000e-01	0.00000e+00		
8	0.00000e+00	0.00000e+00	9.92000e-01	
8.00000e-02	-2.00000e+00	0.00000e+00		
9	0.00000e+00	0.00000e+00	1.00000e+	
00	-3.41949e-14	2.00000e+00	0.00000e+00	
10	0.00000e+00	0.00000e+00	0.00000e+00	
0.00000e+00	0.00000e+00	0.00000e+00		
11	0.00000e+00	0.00000e+00		
9.00000e-02	-2.52836e-15	-6.00000e-01	0.00000e+00	
12	0.00000e+00	0.00000e+00	1.00000e+	
00	-8.27116e-15	-2.00000e+00	0.00000e+00	
13	0.00000e+00	0.00000e+00		
8.00000e-01	-4.00000e-01	2.00000e+00	0.00000e+00	
14	0.00000e+00	0.00000e+		
00	-2.00000e-01	-4.00000e-01	4.92991e-14	0.00000e+00
15	0.00000e+00	0.00000e+		
00	-1.10000e-01	-4.00000e-01	-6.00000e-01	0.00000e+00

图 27: plate 单元 patch test 结果

在机器精度上和准确解完全符合，可以认为 plate 单元通过了 patch test，是收敛的。

接下来是进行收敛性分析。考虑到 plate 单元实际上只能在单元平面外产生变形，所以设计的检验模型为在 z 方向施加体力，使得位移场呈现正弦函数型。然后分别对一个单元、四个单元和十六个单元计算，画出基于 L2 范数的误差和分片大小的对数坐标图，并且进行线性函数拟合，得到的结果如下，

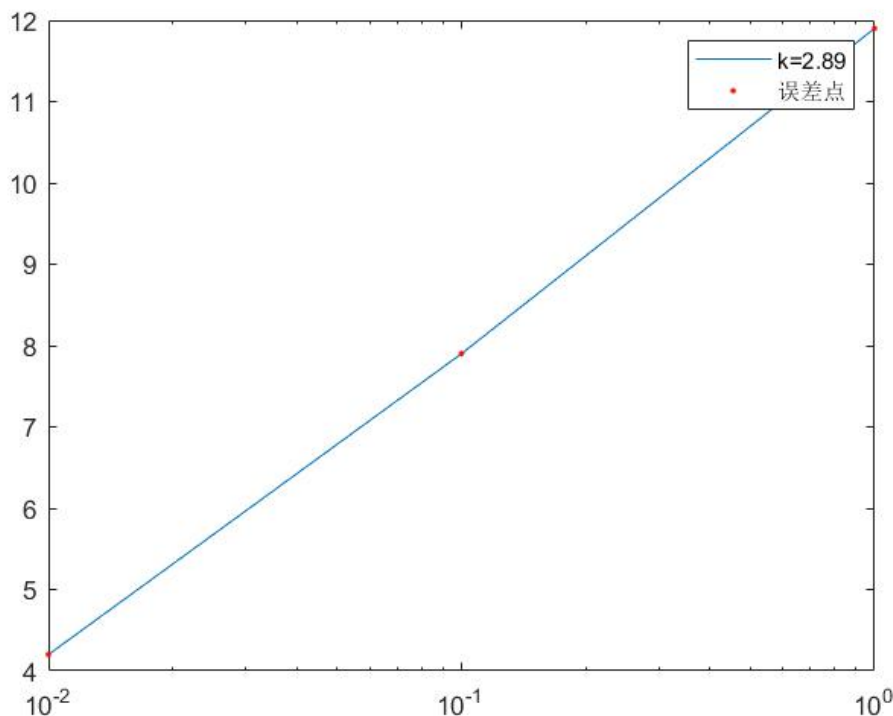


图 28: plate 单元收敛率分析

为了便于拟合等操作，误差值进行了等倍数放大。可以看到，直线斜率约为 3，考虑到随着 patch 数的增加，收敛效果会变好，所以基本满足收敛性分析要求。

2.8 Shell 单元

由于我们组有同学退课，所以在开始的两个星期没有写 shell 单元。在发现程序运行中，如果将 4Q 单元和 Plate 单元用指针链接，拼接成为 Shell 单元时，指针报错，所以临时补写了 Shell 单元。

实际 shell 单元的编写思路为 4Q+Plate 单元，但不是简单地链接合并。分别计算出中性面在平面内的位移和垂直于平面方向的位移，然后根据壳单元的厚度，得出上下两个面上节点的位移。同样为了避免求逆，参考了资料上的形函数，类似于 Plate 单元，得出最终结果。由于具体过程类似，本文不再赘述。

之后是对 Shell 单元进行分片实验，分片和 Plate 类似，都是有 16 个单元，但是不同的是增加了面内的变形和一些转角的自由度约束。如下图所示，


```
patch.dat * ×
1 A_patch_test_of_shell
2 16 1 1 1
3 1 1 1 0 0 0 1 -1.0 -1.0 0.0
4 2 0 0 0 0 0 1 0.0 -1.0 0.0
5 3 0 0 0 0 0 1 0.3 -1.0 0.0
6 4 0 1 0 0 0 1 1.0 -1.0 0.0
7 5 0 0 0 0 0 1 -1.0 -0.2 0.0
8 6 0 0 0 0 0 1 0.0 -0.2 0.0
9 7 0 0 0 0 0 1 0.3 -0.2 0.0
10 8 0 0 0 0 0 1 1.0 -0.2 0.0
11 9 0 0 0 0 0 1 -1.0 0.0 0.0
12 10 0 0 1 1 1 1 0.0 0.0 0.0
13 11 0 0 0 0 0 1 0.3 0.0 0.0
14 12 0 0 0 0 0 1 1.0 0.0 0.0
15 13 0 0 0 0 0 1 -1.0 1.0 0.0
16 14 0 0 0 0 0 1 0.0 1.0 0.0
17 15 0 0 0 0 0 1 0.3 1.0 0.0
18 16 0 0 0 0 0 1 1.0 1.0 0.0
19 1 16
20 1 5 1.6
21 5 5 2
22 9 5 2.4
23 13 5 2
24 4 5 -1.6
25 8 5 -2
26 12 5 -2.4
27 16 5 -2
28 1 2 -1
29 2 2 -1.3
30 3 2 -1.0
31 4 2 -0.7
32 13 2 1.0
33 14 2 1.3
34 15 2 1.0
35 16 2 0.7
36 7 9 1
37 1 24 0.2 1
38 1 1 2 6 5 1
39 2 2 3 7 6 1
40 3 3 4 8 7 1
41 4 5 6 10 9 1
42 5 6 7 11 10 1
43 6 7 8 12 11 1
44 7 9 10 14 13 1
45 8 10 11 15 14 1
```

图 29: shell 单元 patch test

程序运行得到的结果如下，

ELEMENT		GAUSS P		GAUSS POINTS DISPLACEMENTS		
STRESSES						
NUMBER	INDEX	W	Theta	ThetaY		
SX' X' _MAX	SY' Y' _MAX	SX' Y' _MAX				
1	1	8.00000e-01	4.00000e-01	2.00000e+00	-8.69042e+01	
1.33253e+01	3.21965e-13					
1	2	-1.66667e-02	3.25261e-16	-2.00000e-01	-3.01615e-01	
3.06458e+01	-1.19349e-13					
1	3	4.00000e-01	5.33532e-14	-2.16667e-02	-8.09584e+	
00 -8.32532e+00	-3.91354e-13					
1	4	2.86229e-16	-1.10000e-01	4.00000e-01	-9.46984e+	
01 -2.56458e+01	3.55271e-15					
1	5	-6.00000e-01	-3.33333e-02	8.00000e-01	-1.04083e-15	
2.00000e+00	7.50268e-16					
2	1	8.00000e-01	4.00000e-01	2.00000e+00	8.51043e+01	
4.77270e+01	-4.94197e-13					
2	2	-1.66667e-02	3.25261e-16	-2.00000e-01	-4.05643e+	
02 -5.04225e+01	9.93742e-14					
2	3	4.00000e-01	5.33532e-14	-2.16667e-02	-4.13438e+	
02 -8.93937e+01	1.47975e-16					
2	4	2.86229e-16	-1.10000e-01	4.00000e-01	7.73101e+01	
8.75587e+00	-4.75849e-14					
2	5	-6.00000e-01	-3.33333e-02	8.00000e-01	-1.90820e-15	
2.00000e+00	-3.83085e-16					
3	1	8.00000e-01	4.00000e-01	2.00000e+00	8.43137e+01	
4.75689e+01	-1.92901e-13					
3	2	-1.66667e-02	3.25261e-16	-2.00000e-01	-2.14377e+	
02 -1.21692e+01	1.17961e-13					
3	3	4.00000e-01	5.33532e-14	-2.16667e-02	-2.22171e+	
02 -5.11403e+01	2.15106e-13					
3	4	2.86229e-16	-1.10000e-01	4.00000e-01	7.65195e+01	

图 30: shell 单元 patch test 结果

在机器精度下，可以认为已经通过 patchtest，单元收敛。事实上，在进行桥梁计算时，也用到了 shell 单

元，在和 ABAQUS 的比较中，也可以发现，该单元基本满足收敛性要求。

2.9 IEM 单元

IEM 单元，是 STAP_{pp} 程序的一个扩展单元。IEM，是“无限元”的英文简称。该单元，主要用于解决无限大平板的有限元计算问题。

该单元的设计思路较为简单，主要按照有限元课堂上所讲的内容。例如，若在 x 方向趋于无穷，则对 x 建立映射，

$$\xi = 1 - \frac{x_Q - x_C}{x - x_C} = 1 - \frac{x_Q - x_C}{r}$$

则，当 ξ 趋于-1 时， x 在物理单元中取 x_P ；当 $\xi = 0$ 时， x 在物理空间中取 x_Q ；当 $\xi = 1$ 时， x 趋于正无穷。其对应的物理单元和母空间中的单元如下图所示，

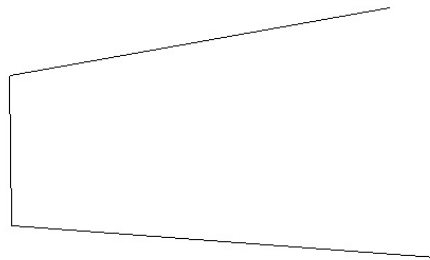


图 31: IEM 物理空间中的单元

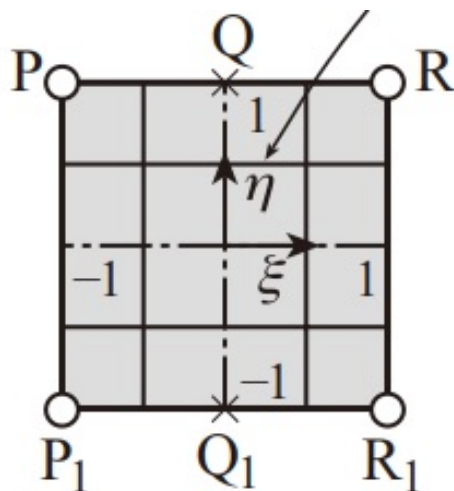


图 32: IEM 对应的母单元

但是，由于物理空间中的无限性，显然不可能求出对应节点的位移和应力，只能将对应高斯点的位移和应力求出，通过逆变换代入物理空间中，得到结果。除了映射有所不同，所取的近似函数有差别外，母空间中

问题的方程的建立和求解均和 4Q 单元相同。

接下来一个较为重要的问题就是如何做 IEM 单元的 patch test 和定义收敛率的问题。由于我们并没有找到很好的办法去证明某种 patch test 的有效性，在查阅一些文献的时候也没有找到统一的明确的定义，所以在这个问题中，我们认为按以上步骤构造的 IEM 单元在对应的 4Q 单元的意义下是否通过 patch test 可以视为 IEM 单元本身是否能够通过 patch test。所以我们认为这样的 IEM 单元是收敛的，尽管收敛率可能比对应的 4Q 单元收敛率较低。而在随后对算例的求解中也能够得到较为合理的结果。

2.10 若干“巧凑”边点元

这里的若干“巧凑”单元主要指从 5Q 到 9Q 的若干由“Serendipity”方法构造出来的单元。“Serendipity”方法，即在继承 4Q 单元四个角点的形函数后，针对位于单元边中点或者整个形状中心点，利用形函数在其他点值为零，在对应点值为一的特性，用因式相乘，最后乘上某一系数做归一化的方法，构造其他点的形函数。下图为 8Q 单元构造形函数的对应方法。

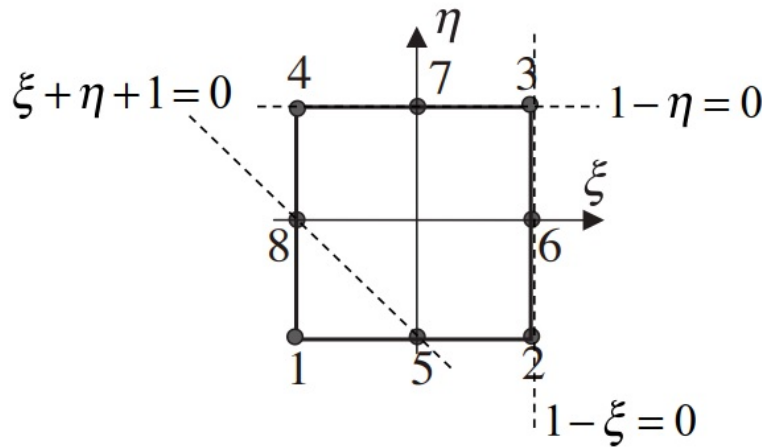


图 33: 用 Serendipity 方法构造 8Q 单元形函数

所有 serendipity 单元程序的具体书写，通过一种“程式化”的方式进行“批量生产”。首先通过在 2D 弹性问题的基础上改变来的 Matlab 程序，得出单元的刚度阵的解析表达式，通过 cout 等函数输出为 C++ 程序可以读入的形式，并同时利用 MATLAB 处理，得出计算所需要的中间变量，以加快计算速度。然后通过另外编写的 C++ 程序，以文本方式读入读出 MATLAB 结果，生成完全符合 C++ 语法的程序文本，然后贴入 STAPpp 程序中，修改其他头文件和 output 格式后，即可完成在 STAPpp 程序中添加对应的 Serendipity 单元模块。

在 patch test 方面，主要通过对应的 Serendipity 单元的拼接来实现 patch test。以 5Q 单元和 9Q 单元为例，如下图所示，

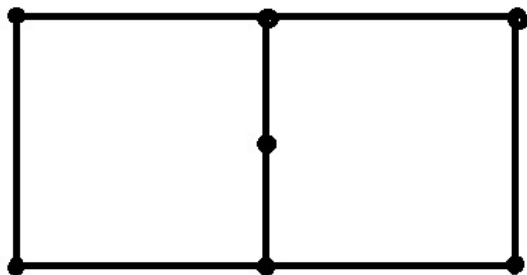


图 34: 用 Serendipity 方法构造 5Q 单元 patch test

其数值计算解在机器精度上和理论解相同，可以认为通过 patch test。在收敛性测试上，其收敛率和对应的 4Q 单元基本一致（由于单元数量较多的网格较难设计，而且没种不能保证结果一样，所以曲线拟合点数量较少）。

2.11 求解器优化

为优化计算，提高运算速度，我们在原有 LDLT 求解器的基础上导入了 Intel-mkl 库中的 pardiso 稀疏矩阵求解器。同原有 LDLT 求解器基于 skyline 矩阵存储格式一样，pardiso 稀疏矩阵求解器基于 CSR 稀疏矩阵压缩存储格式，我们在原有的程序基础上，模仿 skyline 格式的结构，对应地编写了 CSR 格式的 CSRMatrix: Element, CSRMatrix: Getdignoaladdress 等类来将输入数据转化为 CSR 格式并进行对应计算。并将 CSR 相关类用 ifdef 标注，这样可以在 cmake 步骤中选择不同宏定义来选择 LDLT 和稀疏矩阵求解两种求解模式。

2.12 后处理

由于并不需要我们单独画图而只需使用商用软件进行后处理，故后处理的任务简要可以认为将我们的计算结果转化为商用软件的输入形式。

首先是软件的选择。虽然 tecplot 并非开源（需要用盗版），但是在了解 tecplot 之后发现其对于有限元的后处理极为友好，输入格式简明。因而选择 tecplot 作为我们的后处理软件。

我们在 STAPPP 中后处理的思路如下：我们需要一个合适的.dat 的后处理文件，而原有的.out 输出文件由于格式已经固定，故唯一的办法是额外添加一个作为后处理的输出文件。我们在 STAPPP 中的 main 程序中创建一个新的.dat 文件按 tecplot 格式存储数据。具体的写入过程通过一个 PostOutputter 实现。类似于 Outputter 的实现方式，我们在每个单元中定义了专门计算后处理信息的函数 ElementPostInfo，然后在 PostOutputter 中调用，按照 tecplot 的格式输出。我们的后处理信息包括：位移后坐标、应力不变量、Mises

应力以及 6 个应力分量。具体代码请查看主程序文件。我们由下图可以看到我们的后处理最终效果和 abaqus 是十分相近的。

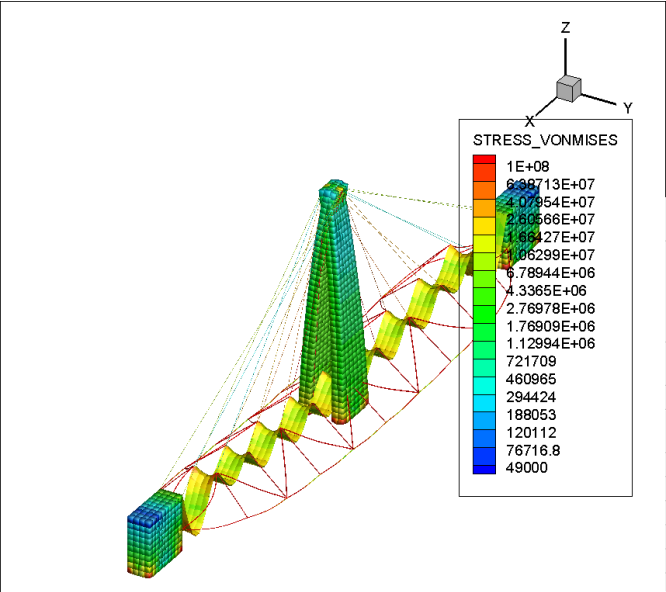


图 35: tecplot 后处理

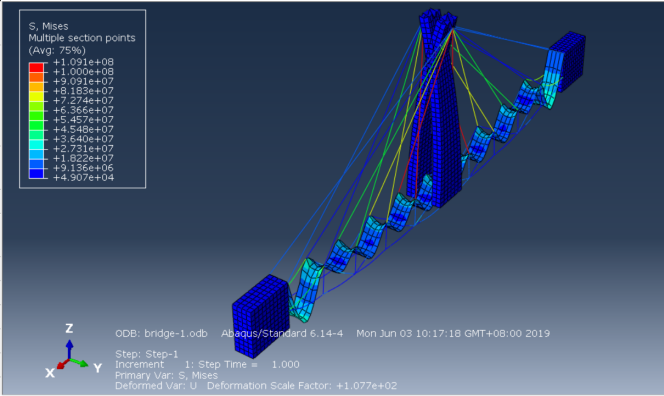


图 36: abaqus 后处理

3 桥梁设计、优化与评估

3.1 选题——为什么选斜拉桥

作为一种拉索体系，斜拉桥比梁式桥的跨越能力更大，是大跨度桥梁的最主要桥型。斜拉桥由许多直接连接到塔上的钢缆吊起桥面，斜拉桥主要由索塔、主梁、斜拉索组成。索塔型式有 A 型、倒 Y 型、H 型、独柱，材料有钢和混凝土的。斜拉索布置有单索面、平行双索面、斜索面等。第一座现代斜拉桥是 1955 年德国 DEMAG 公司在瑞典修建的主跨为 182.6 米的斯特伦松德（Stromsund）桥。目前世界上建成的最大跨径的斜拉桥为俄罗斯的俄罗斯岛大桥，主跨径为 1104 米，于 2012 年 7 月完工。

和拱桥相比，斜拉桥更为美观，也更加适用在长距离问题上。我们的算例要求 1500m 的跨度，因而一般情况下用斜拉桥更加合适。但是一个重要的问题是，我们对桥梁的参数比例一无所知。为此我们选取了一个典型的斜拉桥：苏通长江大桥作为我们的参考模型。通过对图片形状的估计，我们确定了我们桥梁的大致参数与设计方案。苏通长江大桥高度为 300m，跨度 1088m。因而对我们的桥，等比例地我们设定桥塔总高 117m，单跨 300m。

3.2 桥的设计

我们的桥主要由三部分组成：桥面，桥塔与钢索。

1. 桥面。桥面使用的是二维壳单元，长 1500m，宽 15m，正好满足设计要求。其中 mesh size 为 1。材料厚度参数设置为 0.1m。
2. 桥塔。桥塔采用的是二维草图拉伸的生成方式。桥塔主要由三个部分组成：最上方的拉索区，高为 30m，用于连接我们的拉索。中间的桥面区：包括承受桥面的部分。最下方的支撑区。尽管我们的算例中载荷是完全



图 37: 苏通长江大桥

稳定的，但是考虑到稳定性，我们的桥塔脚依然采取了斜的设计。这样我们的桥更能够抵御扰动的影响。

3. 桥索。桥面每隔 10m，桥塔每隔 2m 高设置一对桥索。为了保证单元的受力均匀，桥索被均匀分散连接在桥塔上。桥索共有 16 种规格，分别对应桥面距离 10m-160m。取桥索的 mesh size 极大以至于其为一个单元而不会产生额外自由度。

具体的细节请参照模型 Bridge0615.

3.3 装配

整体装配通过镜像阵列简化操作。具体细节不再赘述。最终装配图如下

我们两类约束：1. 钢索约束。把我们的钢索和桥塔、桥面相连接。使用 Tie 约束，Node Region 方式将钢索约束在桥面的边缘与桥塔上端的边缘。2. 桥面约束。使用 Tie 约束将桥面约束在桥塔的承受桥面的部位。载荷为工况下载荷。

3.4 优化

为了改善我们的桥面设计，我们采取了如下方案优化。首先我们尽量去除了冗余材料：为此我们将桥塔上端设计为空心。其次，我们发现最大位移点在桥塔与桥塔的中间，因而我们在这个区域额外增加了钢索以减小最大位移。然后我们还发现钢索的承受的拉力与其位置有关。越靠近桥塔的地方其所承受的拉力越小，对最大位移的抑制作用也越小。而远离桥塔的钢索则承受更大的拉力，其等效弹簧系数对最大位移的影响也越大。因此，靠近桥塔和远离桥塔的钢索所需要的截面面积是不同的。因而我们根据离桥塔的距离设计了 4 组不同

截面面积的钢索材料属性。越靠近桥塔的钢索越细，反之越粗。最后的计算结果表明这种方式有效地在同样钢索用料的情况下减小了最大位移。

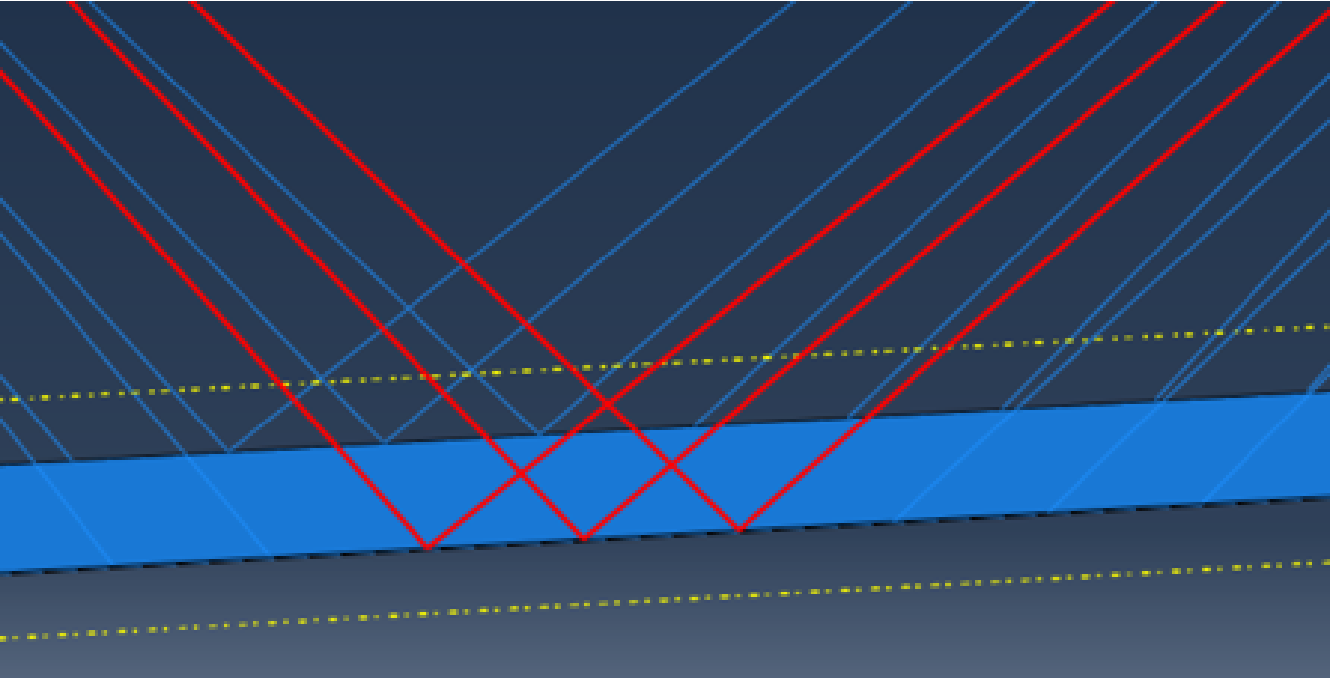


图 38: 桥面中段额外钢索

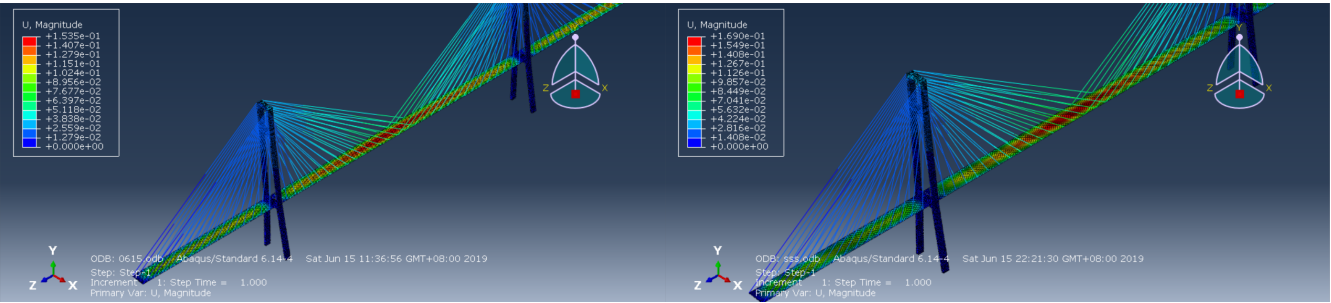


图 39: 同材料优化后最大位移

图 40: 优化前最大位移

3.5 最终结果

首先在后处理的 Visualization 部分我们可以直接得到最大位移数据。结果显示最大位移点在桥塔与桥塔中间，最大位移 15.35cm。

然后我们可以看到我们的 Mises 应力全部单元都在许可应力内，因此全部单元合格。

然后我们利用 Abaqus 的 Tool 可以得到我们的材料用量。Steel 用量 $2.043\text{e}+007\text{kg}$ ，Concrete 用量 $2.170\text{e}+004$ 立方米。最终的花费为 1.24 亿 RMB

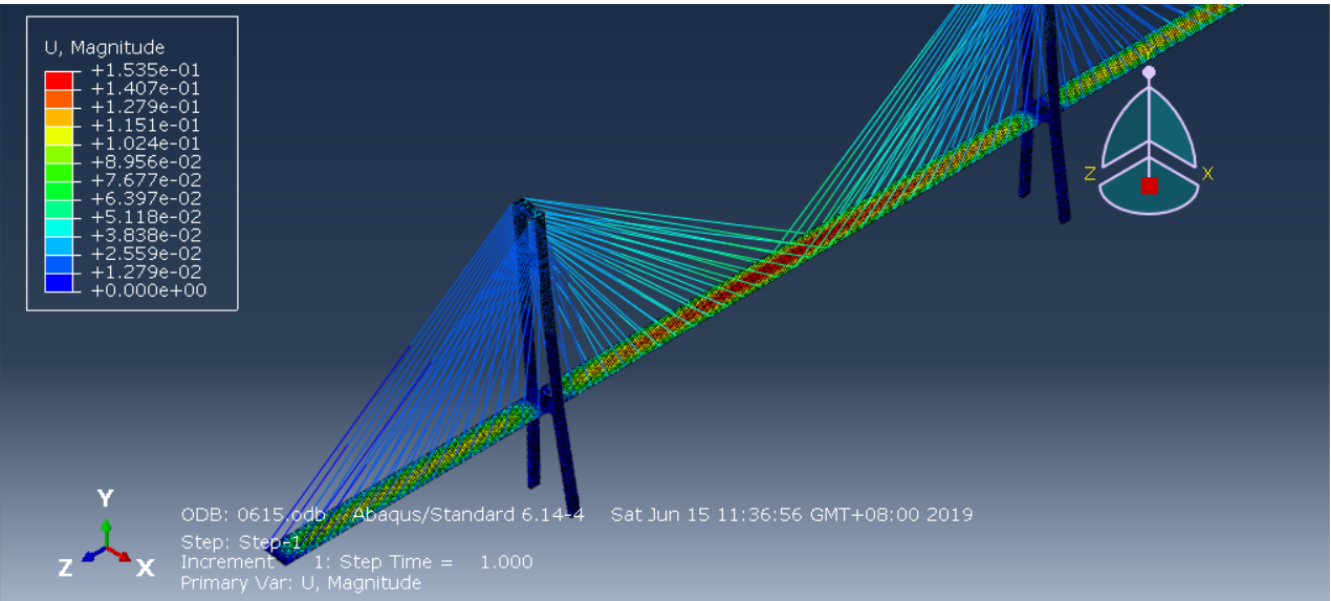


图 41: 最大桥面位移

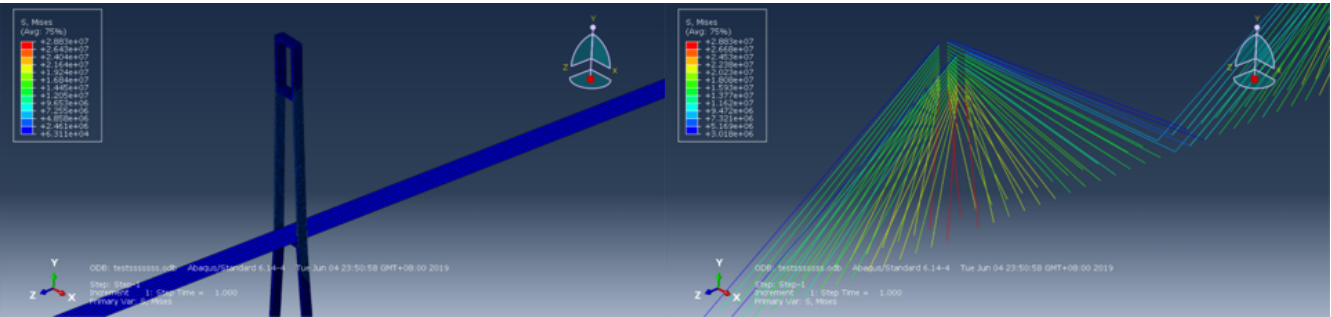


图 42: 混凝土应力情况

图 43: 钢索应力情况

4 结束语

程序的功能还有很多不完善的地方。例如，到底如何才能做到用 Plate 单元计算任意形状四边形单元？如何进一步提高单元的收敛率和程序的运行效率？……我们希望能接下来在今后的学习和科研中国，尽可能探究，并对程序进行进一步的完善。

在本文的末尾，再次感谢老师的教导和学长的辛苦付出！