桥梁大作业报告

组长：贺琪
组员：陈煜 刘畅武 杨昊光 朱子霖

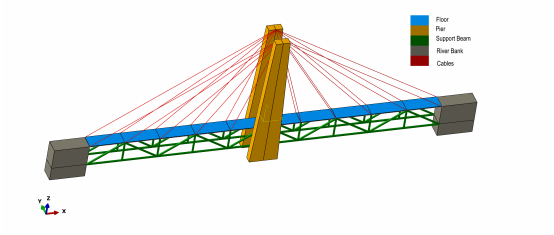
2016 年 12 月 25 日

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。

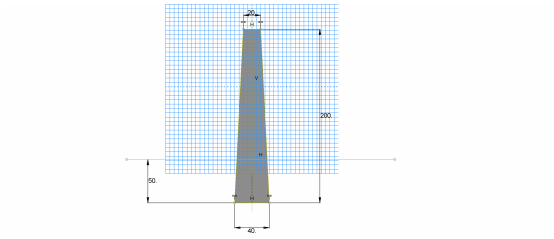


桥墩：

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



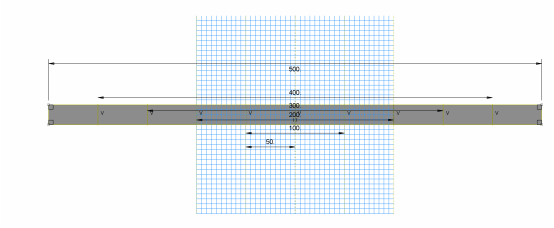
桥面：

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2×2×5=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50×50×20的立方体，如图1所示，变长为20的一边与桥面相铰接，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。

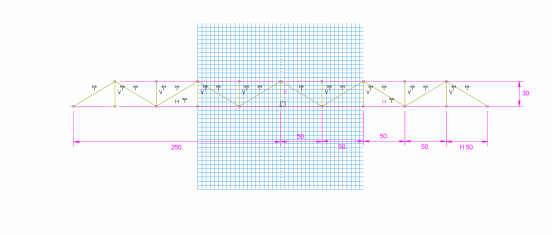
材料	：	Granite
弹性模量	：	60e9
泊松比	：	0.27
密度	：	2770

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2×9=18个结点。两端结点与河堤相铰接，共计2×2=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



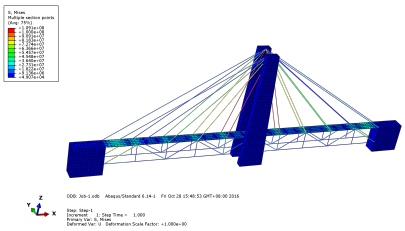
钢缆：

连接桥面与桥墩，共计2×2×5=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

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完备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：

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

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

最小二乘逼近为解系数矩阵$\boldsymbol{C}_{ij}$，使得：$\boldsymbol{A}\boldsymbol{C}_{ij}=\boldsymbol{S}_{ij}$，即$\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{C}_{ij}=\boldsymbol{A}^T\boldsymbol{S}_{ij}$。最后解$\boldsymbol{X}^{(k)}\boldsymbol{C}_{ij}=S_{ij}^{(k)}$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

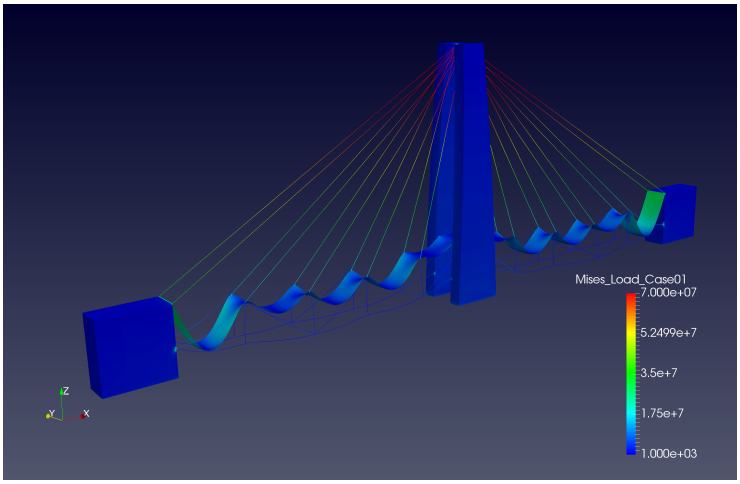
本程序后处理采用ParaView完成，在程序内包含输出为VTK文件的子程序VTKgenerate

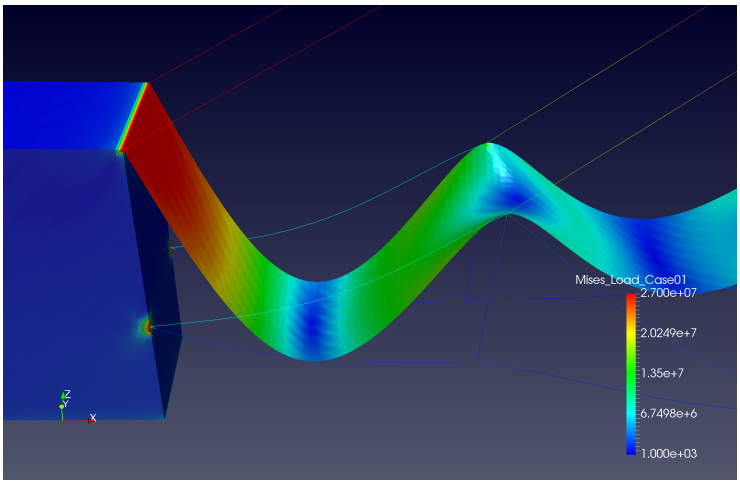
在IND=0阶段，对VTK文件进行初始化，按VTK要求的格式写入文件头和各节点坐标。

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$\boldsymbol{X}=\begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ \vdots & \vdots & \vdots \\ x_8 & y_8 & z_8 \end{bmatrix}$，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$\xi,\ \eta,\ \zeta$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：

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

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

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

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

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

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

$\boldsymbol{D}$矩阵由本构关系得到，在此认为材料是各向同性的：

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

$$\Rightarrow \boldsymbol{D} = \begin{bmatrix} 2G+\lambda & \lambda & \lambda & & & & \\ \lambda & 2G+\lambda & \lambda & & & & \\ \lambda & \lambda & 2G+\lambda & & & & \\ & & & 2G & & & \\ & & & & 2G & & \\ & & & & & 2G & \end{bmatrix},\ \text{s.t. } \boldsymbol{\sigma} = \boldsymbol{D}\boldsymbol{\varepsilon},\ \text{where } \boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \varepsilon_{xy} \\ \varepsilon_{yz} \\ \varepsilon_{zx} \end{bmatrix},\ \boldsymbol{\sigma} = \begin{bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \tau_{xy} \\ \tau_{yz} \\ \tau_{zx} \end{bmatrix}.$$

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

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$\boldsymbol{K}\boldsymbol{d} = \boldsymbol{f}$.

- 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$\boldsymbol{D}$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$\boldsymbol{A}$、$\boldsymbol{S}$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个点来将其分划为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.

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

$$F^{(e)} = \left\{ F_{Ni} \quad F_{Qiy} \quad F_{Qiz} \quad M_{ix} \quad M_{iy} \quad M_{iz} \vdots \quad F_{Nj} \quad F_{Qjy} \quad F_{Qjz} \quad M_{jx} \quad M_{jy} \quad M_{jz} \right\}^T$$

$$\mathrm{d}^{(e)} = \left\{ u_i \quad v_i \quad w_i \quad \theta_{ix} \quad \theta_{iy} \quad \theta_{iz} \vdots \quad u_j \quad v_j \quad w_j \quad \theta_{jx} \quad \theta_{jy} \quad \theta_{jz} \right\}^T$$

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

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

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.

$$\mathbf{K^{(e)\prime} = T^T K^{(e)} T} \tag{2}$$

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 lateral,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 ≤ N ≤ 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 ≤ NOD ≤ 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 ≤ N ≤ 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:

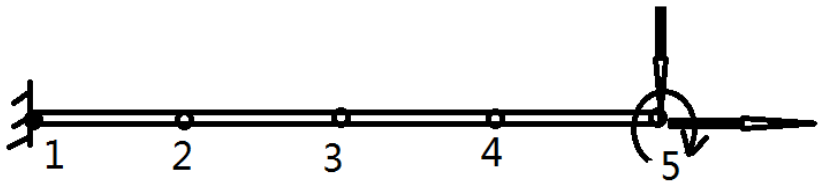
列。	变量。	意义。
1-5。	M。	单元号（1 ≤ M ≤ NPAR(2) ）。
6-10。	II。	单元左端节点号（1 ≤ II ≤ NUMNP）。
11-15。	JJ。	单元右端节点号（1 ≤ JJ ≤ NUMNP）。
16-20。	MTYP。	本单元的材料/截面性质组号。
21-25。	KG。	单元自动生成增量。

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 bandwidth 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点的个数的点。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.f*90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存贮中的元素个数，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(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是一些小细节，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$Job-10.25s$fi$Job-23s, Job-330$s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

0.3.2 4Q

0.3.3 6T

0.3.4 8Q

0.3.5 9Q

0.3.6 4T

0.3.7 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下的扭转，铁木辛柯梁意义下的弯曲，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

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

0.3.8 Plate

0.3.9 无限单元

0.3.10 超级单元

0.3.11 过渡单元

0.4 高级功能

0.4.1 弹塑性杆分析

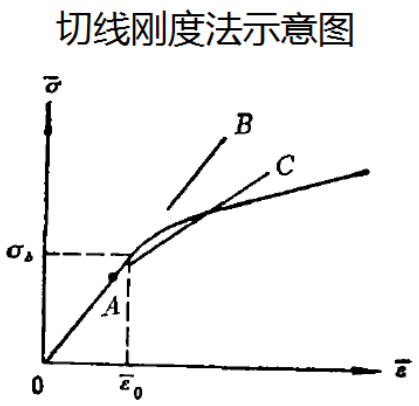
基本原理

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，

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

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

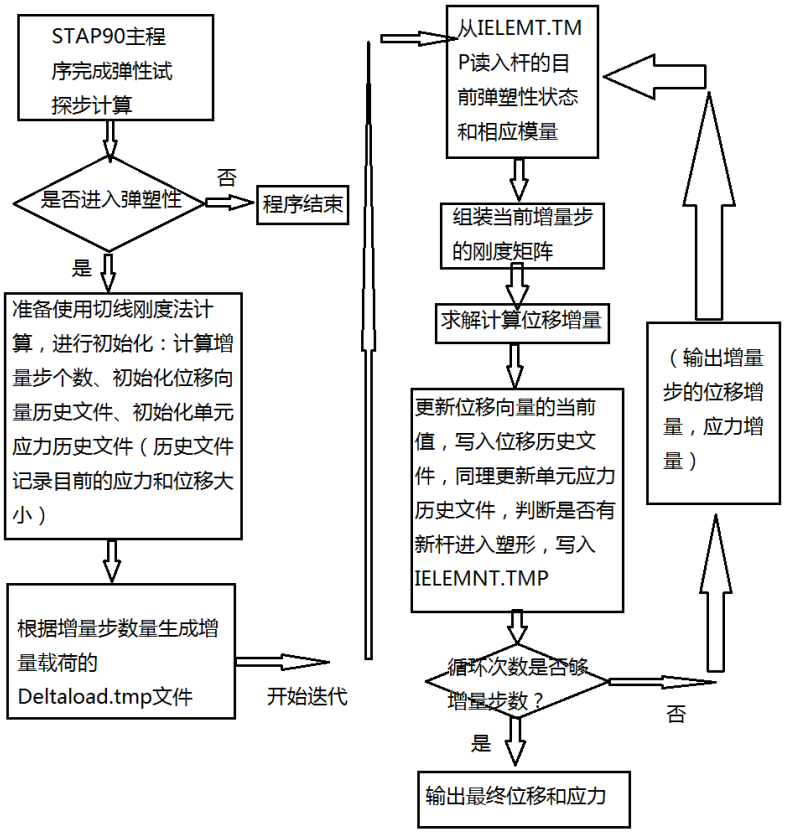
我们在使用切线刚度法进行计算时，都是从A点的应力状态出发，最后就得到载荷B情况下的各项参数，本方法的特点是，对于一次载荷分量，只需要求解一次即可。示意图如下。



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。

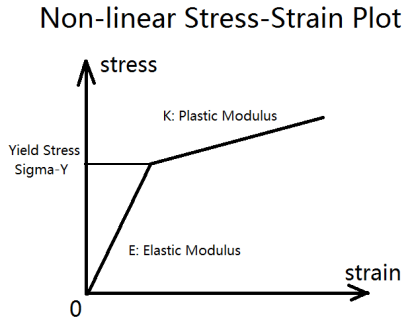
编程思路

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-SECTIONAL AREA A	YIELD-STRESS SIGMA-Y	PLASTIC-MODULES K
1	0.10000E+12	0.100000E-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	0.000000E+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

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 NUMBER	FORCE	STRESS
1	0.101000E+04	0.101000E+06

目前应力总值

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 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	0.000000E+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 NUMBER	STRESS
1	0.100000E+07

0.4.2 模态分析

0.4.3 动力学响应分析