

自动求解有侧移框架柱计算长度系数的算法与实现

摘要: 主要介绍了如何编程接续结构分析软件 STAAD 自动求解钢柱的计算长度的算法和编程思路。

关键词: STAAD, 标准模版库, 有侧移柱计算长度系数

The algorithm and application to calculate K value automatically for sideway uninhabited frame column

ABSTRACT: The algorithm that how to calculate K value for sideway uninhabited frame column automatically is introduced.

KYE WORDS: STAAD, STL, K value for Sideway uninhabited frames

计算柱子的计算长度系数是一个比较繁琐的过程, 如何快速计算整个计算模型中所有柱子的计算长度系数就更加让工程师头疼。本文介绍如何编写程序读取 STAAD 的输入文件和输出文件, 自动计算有侧移框架柱的计算长度系数的思路。

美国规范有侧移框架柱的计算长度系数的计算方法

在美国规范 ANSI/AISC 360-05 中, 有侧移框架柱的计算长度系数 K, 是通过以下的超越方程来计算的:

$$\frac{G_A G_B \left(\frac{\pi}{K}\right)^2 - 36}{6(G_A + G_B)} - \frac{\frac{\pi}{K}}{\tan\left(\frac{\pi}{K}\right)} = 0, \quad G = \frac{\sum\left(\frac{E_c I_c}{L_c}\right)}{\sum\left(\frac{E_g I_g}{L_g}\right)} = \frac{\sum\left(\frac{EI}{L}\right)_c}{\sum\left(\frac{EI}{L}\right)_g}$$

其中, G_A , G_B 分别为相交于柱上端、柱下端的柱 (Column) 线刚度之和与梁 (Girder) 线刚度之和的比值。

中国规范有侧移框架柱的计算长度系数的计算方法

中国规范 GB50017-2003 表 D-2 之注 1 对有侧移框架柱的计算长度系数, 按下式计算

$$\left[36K_1 K_2 - \left(\frac{\pi}{\mu}\right)^2 \right] \sin \frac{\pi}{\mu} + 6(K_1 + K_2) \frac{\pi}{\mu} \cdot \cos \frac{\pi}{\mu} = 0$$

其中, K_1 , K_2 分别为相交于柱上端、柱下端的横梁线刚度之和与柱线刚度之和的比值。显然, 中国规范中的 K_1 , K_2 分别为美国规范中 G_A , G_B 的倒数。分别令 $K_1=1/G_A$, $K_2=1/G_B$ 并经过简单的三角函数的变换, 易证美国规范中计算有侧移框架柱的超越方程与中国规范计算有侧移框架柱的超越方程完全一致。有侧移框架柱的计算长度系数都大于 1。

在与柱翼缘平行的平面内, 通常布置斜撑构成无侧移体系, 此种体系的柱子的计算长度系数都小于 1。本文只讨论在与柱腹板平行的平面内的柱子的计算长度系数 (即 STAAD 软件中的参数“KZ”) 的自动求解。

基本假定

- 梁、柱都是等截面;
- 斜柱的计算长度系数不予计算;
- 当横梁与柱铰接时, 取横梁线刚度为 0;

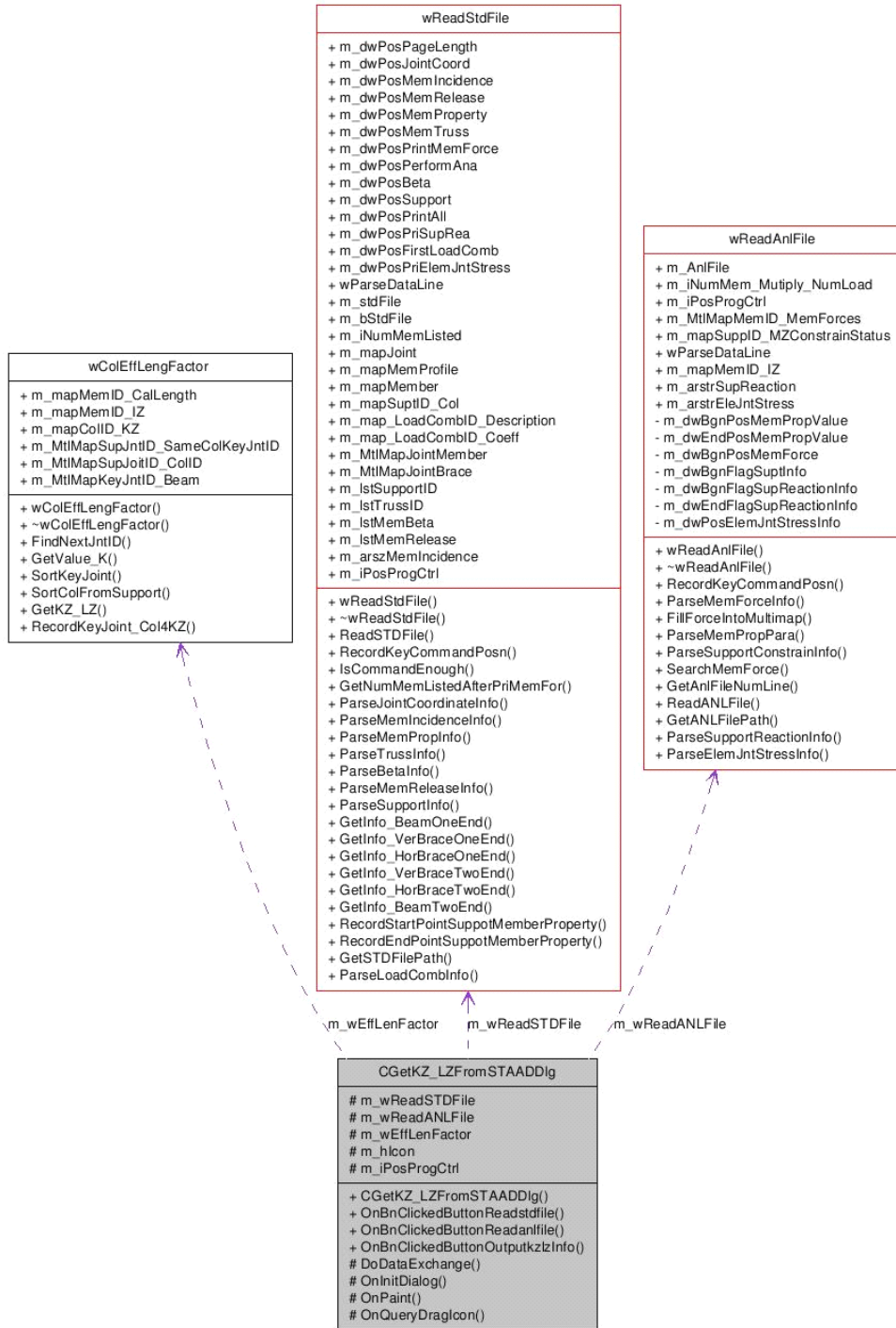
- 柱脚处的 G 值，对铰接柱脚取 10.0，对刚接柱脚取 1.0；
- 不考虑横梁轴力的影响。

思路介绍

对于计算机而言，求解超越方程并不是难点，主要的难点在于如何存储模型中的点和杆件的拓扑关系，并据此得到柱子和梁的线刚度之和。这个问题可以分解为以下几个子问题：

- 1) 节点坐标、节点属性（包括支座还是非支座、节点编号等）、杆件属性（包括两端约束情况、起点编号和终点编号、长度、BETA 角、截面等信息）以及杆件之间的相互位置关系等几何拓扑信息如何读取以及如何保存；
- 2) 在 STAAD 中，主梁被次梁打断成多段，柱子被铰接梁打断，求 K 值时，需要把这些梁段或柱段分别合并，记录真实的梁和柱子（physical member）的长度；
- 3) 梁和柱子的截面惯性矩如何取得；
- 4) 在侧移发生的平面内，柱子两端连接了哪几根梁，梁端约束情况如何，梁的惯性矩和长度分别是多少。

下图是读取 STAAD 文件进行柱计算长度系数的自动求解的程序中用到的几个类的关系图：点状线条的箭头用于显示两个类之间的关系——通过箭头旁边的变量可以访问到箭头所指的类，而这些变量正是箭头所指的类的一个实例：

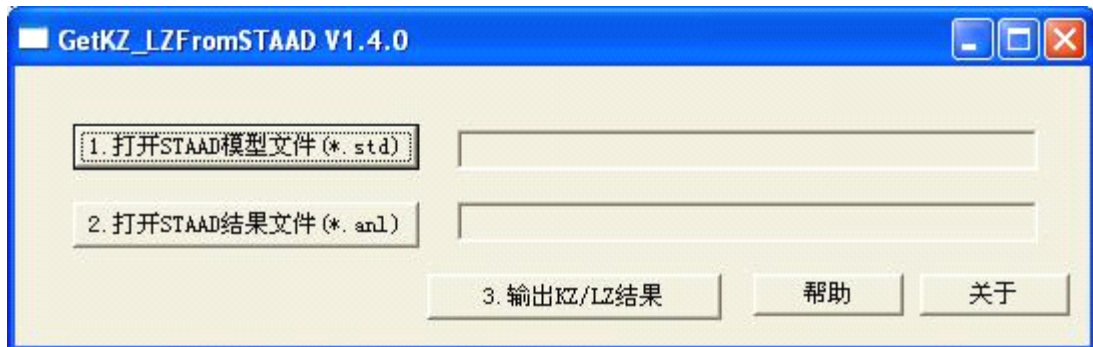


其中，

- 类wReadStdFile主要完成读取STAAD的输入文件 (*. std) 并存储相关信息并对这些信息进行进一步加工的工作，主要解决问题1)和4)；
- 类wReadAnlFile主要完成从STAAD的输出文件 (*. anl) 中读取各个杆件的截面惯性矩并存储相应信息的工作，主要解决问题3)。
- 类wColEffLengFactor主要将上述两个类中存储的信息进行整理，并遍历模型中的所有柱子，按美国规范中的超越方程以及相关的修正条文求解有侧移框架柱的计算长度系数。
- 类 CGetKZ_LZFromSTAADDlg 的工作比较简单，主要负责界面，对应如下图所示的对话框，

该类中有三个重要的成员变量，它们分别是：

- 类 `wReadStdFile` 的实例 `m_wReadSTDFile`；
- 类 `wReadAnlFile` 的实例 `m_wReadANLFile`；
- 类 `wColEffLengFactor` 的实例 `m_wEffLenFactor`。



类 `wReadStdFile` 以及类 `wReadAnlFile` 在《钢结构》（2007年6月）的《STAAD 二次开发的经验》一文中有比较详细的阐述，此处不再赘述。这两个类的通用性较强，把它们封装为两个动态链接库，可以降低各个类之间的耦合度，使得程序架构更加清晰，也易于扩展，易于进行团队开发：

- 事实上，利用这两个动态链接库，我们还开发了自动判断节点形式进行节点设计、读取有限单元节点内力进行配筋计算等几个后处理程序；
- 如果接续其他软件进行柱子的计算长度系数的计算，只需要把类 `wReadStdFile` 以及类 `wReadAnlFile` 换掉即可。

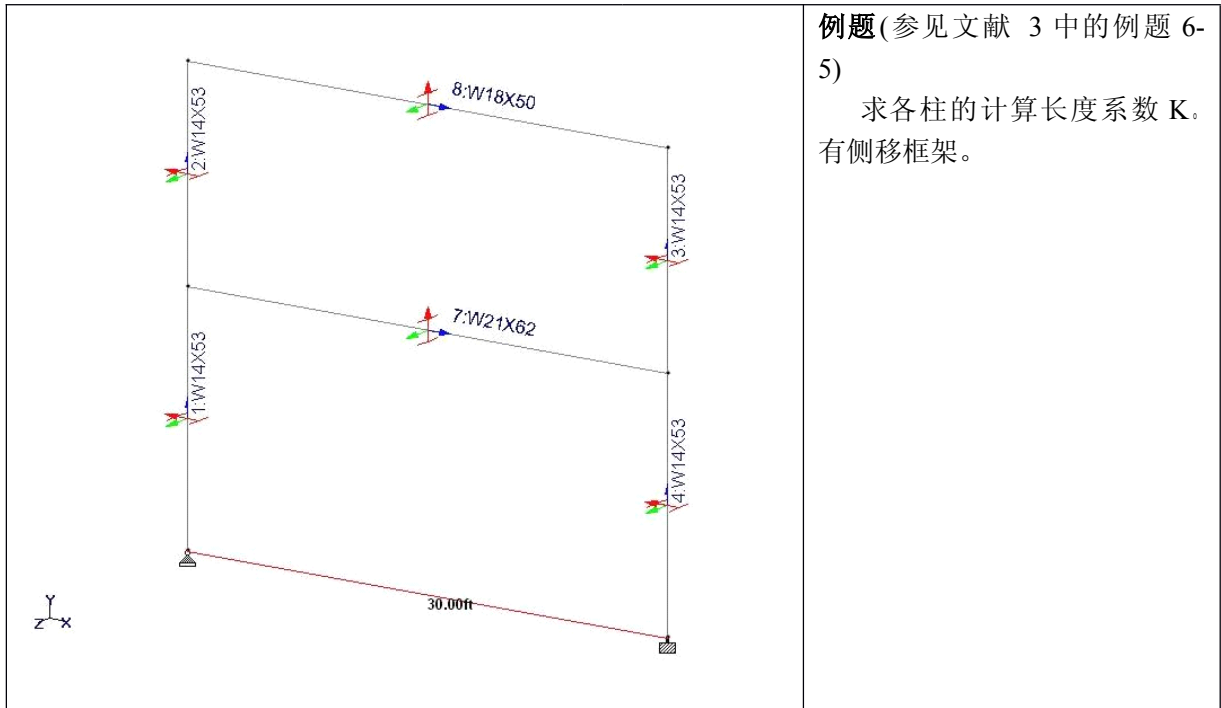
对工程师而言，问题 2) 比较简单，但是对于计算机程序，它却是求解柱计算长度系数的难点所在，这个问题主要在类 `wColEffLengFactor` 中解决，其中的几个函数的调用关系如下图所示，思路如下：

- 调用函数 `wColEffLengFactor::RecordKeyJoint_Col4KZ(wReadStdFile &wSTDFileInfo)` 完成以下工作：
 - 记录与支座节点在同一柱线上的对计算 KZ 有用的关键点的编号，存储在 `multimap<int,int> m_MtlMapSupJntID_SameColKeyJntID` 中（此处所谓的关键点，是指与支座在同一柱线上，且在与柱腹板平行的平面内有梁刚接于该点）；
 - 记录与支座在同一柱线上的所有杆件编号，存储在 `multimap<int,int> m_MtlMapSupJntID_ColID` 中；
 - 记录连在每个关键点且与柱翼缘刚接的梁的信息，存储到 `multimap<int,WMember> m_MtlMapKeyJntID_Beam` 中；
- 遍历每个柱脚，完成以下工作：
 - 调用函数 `wColEffLengFactor::SortKeyJoint (CList< WJoint, WJoint > & lstJoint)`，按节点纵坐标由小到大排序同一柱线上的关键点，存储在临时链表 `lstKeyJnt` 中；
 - 计算相邻两个关键点间的距离，存到动态数组 `arfLen_y` 中；
 - 调用函数 `wColEffLengFactor::SortColFromSupport (CList< WMember, WMember > & lstCol)`，按杆件中点纵坐标排序同一柱线上的杆件，存储在临时链表 `lstCol` 中；
 - 当柱子的最高点不在关键点链表 `lstKeyJnt` 中时，把它加入 `lstKeyJnt` 尾端；
 - 从头遍历存放关键点的链表 `lstKeyJnt`，计算每个关键点处的 G 值（G：此点处“柱

的线刚度之和”跟“梁的线刚度之和”的比值)：

- ◆ 从 `wSTDFileInfo` 的成员变量 `multimap<int,WMember> m_MtlMapJointMember` 中取得与关键点相连的上柱、下柱的 ID 号；
- ◆ 从动态数组 `arfLen_y` 中，找出此关键点与距其最近且纵坐标更高的另一个关键点间的柱的长度；
- ◆ 从 `wANLFileInfo` 的成员变量 `CMap<int,int,float,float> m_mapMemID_IZ` 中找出柱子的截面惯性矩 I_z ；
- ◆ 如果是支座，直接根据规范给出 G_A 值（柱脚铰接取 10.0，刚接取 1.0），退出本层循环，取链表 `lstKeyJnt` 中的下一个关键点重新开始本层循环进行 G 值计算的准备工作；如果不是支座，则继续进行下面的工作；
- ◆ 关键点连接的所有关键梁的计算长度是靠两层循环：
 - 外层循环是水平搜寻，从 `m_MtlMapKeyJntID_Beam` 中找到与关键点相连的梁，沿着该梁的另一端节点（记做 `iIgnoreBeamJntID`）水平顺藤摸瓜，找到相邻的共线梁段的下一节点；
 - 内层循环是对于每个节点 `iIgnoreBeamJntID` 沿纵坐标负方向顺次搜寻连在此点上的柱的另一端节点，看其是否是支座，如果不是支座，继续向下找相连的柱并判断其远端是否是支座，如此循环。循环完毕后，如果不是支座则回到外层循环；如果是支座，说明此时外层循环中的梁的末点就是靠人眼能一眼看穿而靠程序却要苦苦追寻的真实的梁（physical member）的终点而非与次梁相连的中间点，求出梁长度并记录之，保存在类 `wColEffLengFactor` 的成员变量 `CMap<int,int,float,float> m_mapMemID_CalLength` 中；
- ◆ 根据类 `wANLFileInfo` 的成员变量 `CMap<int,int,float,float> m_mapMemID_IZ` 和类 `wColEffLengFactor` 的成员变量 `CMap<int,int,float,float> m_mapMemID_CalLength` 求柱的另一个关键点处的 G_B ；
- ◆ 调用类 `wColEffLengFactor` 的函数 `GetValue_K(float fGA,float fGB)` 求解超越方程，求得该柱的 KZ 值，记录在 `CMap<int,int,float,float> m_mapCollID_KZ` 中；
- ◆ 令 “ $G_A = G_B$ ”，继续进行下一个关键点处的 K 值计算；
- 输出本柱线上所有柱的 KZ , LZ 的值。

算例

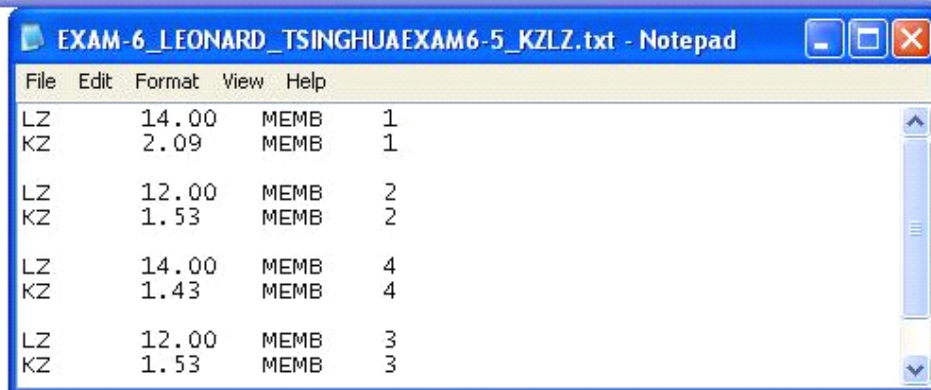
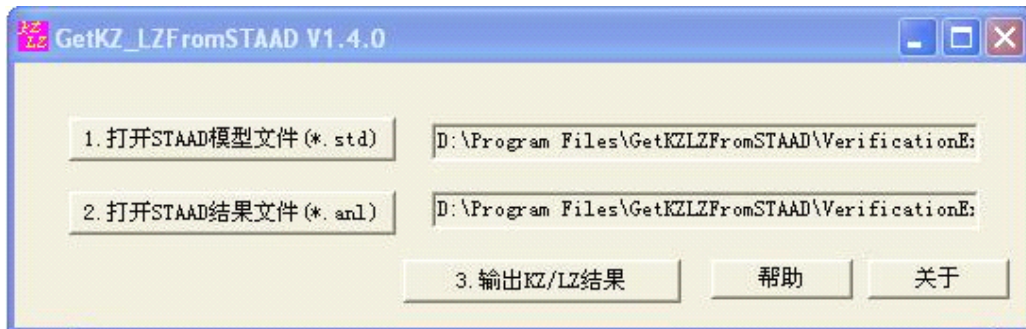


例题(参见文献 3 中的例题 6-5)

求各柱的计算长度系数 K。
有侧移框架。

柱编号	1	2	3	4	7	8
截面	W14x53	W14x53	W14x53	W14x53	W21x62	W18x50
长度	14	12	12	14	30	30
K 值(文献)	2.08	1.54	1.54	1.44	/	/
K 值(本程序)	2.09	1.53	1.53	1.43	/	/

本程序输出的结果如下图所示，输出的文本正是 STAAD 中设置柱计算长度系数的命令行，可以直接拷贝到 STAAD 的输入文件中：



结语

本程序开发过程中大量采用了 C++ 标准程序库 (Standard Library) 的标准模版库 (Standard Template Library) 中的算法 (如排序、查找等) 和数据结构 (如 map, multimap, list, vector, array 等)。STL 模版库是泛型编程 (generic programming) 的一个典范, 几乎所有被广泛运用的数据结构和算法 (algorithms), STL 都已高效而稳定地实现, 在这方面, 古老的 Fortran 是难以望其项背的。

采用这个具有工业强度的模版库, 我们可以从更高的抽象层次来谋划解决问题的思路, 而不必过早在类似“如何实现一个双向链表”、“如何排序才能保证稳定高效”这种细节层面的问题上花费心思。选择优秀的模版库, 能大大提高软件开发的效率, 增强代码的可读性、可维护性, 保证软件产品的稳定性、可复用性以及质量。

参考文献:

1. Nicolai M. Josuttis. The C++ Standard Library: A Tutorial and Reference. Addison Wesley, 1999
2. COMMENTARY on the Specification for Structural Steel buildings, AISC, 2005: 240-243
3. Leonard Spiegel, George F. Limbrunner. Applied Structural Steel Design (Fourth Edition) 北京: 清华大学出版社, 2005: 223-224