Hacking structural analysis. Join the crew.

Luis C. Pérez Tato
l.pereztato@gmail.com

February 16, 2016

1 About (your) heroes

First of all in this post the word *hacking* has nothing to do with cyber-terrorism or any other way the bad guys use computers to achieve their evil objectives. Here a *hacker* is someone that *wants to know* and refuses the idea of being a simple user.

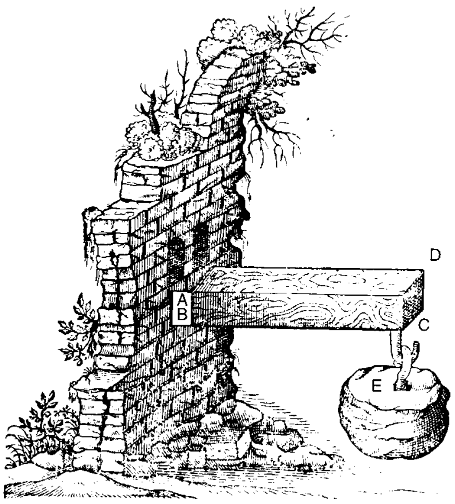
Having said that, if your heroes are Mark Zuckerberg, Steve Jobs, Bill Gates,... then this is probably not a post written for you.

But,... if the names of Steve Wozniak, Leonhardt Euler, Hardy Cross, O. C. Zienkiewicz or Klaus Jürgen Bathe make you feel motivated then maybe you can find here some ideas that whet your appetite (intellectual appetite).

2 Computers & structures, at a glance

Structural engineering is not only about structural mechanics, there are many other disciplines that an structural engineer must to know to do his job. However, it's structural mechanics (the scientific part of our knowledge) what gives meaning and value to our work as engineers.

From Galileo's *Two new sciences* until the early applications of numerical analysis to solve structural problems in the 50's of the last century, structural analysis methods and procedures are beign expressed (as in any other branch of physics) using mathematics.



With the arrival of computers the analysis procedures began to be expressed not only in mathematical language but also in the form of programs written in Fortran or Basic.

In the 80's, the popularization of personal computers and the ability to distribute programs in the form of executable code made possible the creation of a computer industry that initiates the *computing for the masses* era. In engineering offices hand calculators begun to be replaced by personal computers[1]

3 Ok, so where is the problem?

This exciting development has also its pitfalls:

1. Structural engineers become *users* with very little control (if any) over the sofware behavior. In a short time, practical implementation of the algorithms they study at the school, becomes beyond their reach.

2. Before the emergence of commercial software the methods for structural analysis were *ready to use* by engineering community as soon as they were published. Consider, for example, the impact that the publication of the moment distribution method[2] took on the project of statically indeterminate structures. On the contrary, commercial programs are *products* (not scientific papers) so their creation, documentation, distribution and (most of all) advertising are made with commercial criteria.

3. As a consequence of the commercial spirit mentioned in the preceding paragraph, many of these programs are designed to make things faster[3], not better, not in a new way.

[1]To illustrate the scenario: a PC version of ANSYS, is released about 1984, SAP-90 is released in 1989 and a myriad of programs with very limited capabilities grown like mushrooms at these days.

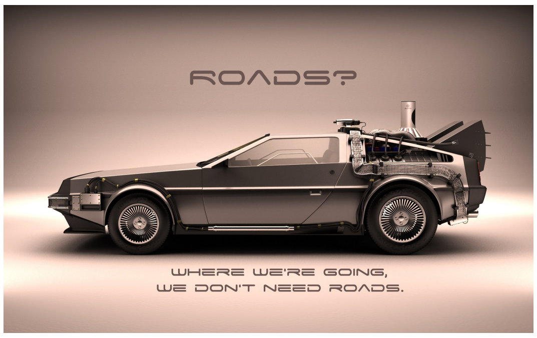
[2]It is hard to imagine that professor H. Cross thought getting rich with the publication of his method.

[3]This way, the analysis becomes as boring as filling the forms on a tax preparation program.

4 So... what's it all about?

The idea is quite simple: assuming we succeed in our mission[4], this alternate 2016 will be changed into the correct 2016, allowing structural engineers community to regain leadershin over their discipline. In other words, we must go **Back to the future** to transform how we work with computing to do it in the same way we do with physics, mathematics, geology,...

Like with maths or any other academic subjet, **there is no need to reinvent the wheel**; the foundation libraries (and some whole applications) for finite element analysis, computer graphics, geometry modeling,... are out there.



5 Join the crew

Well this is it. In case you've become interested, you can take a look to the following links:

- XC home page. XC is the translation into practice of the ideas exposed in this document.
- XC on GitHub. This is the site where we share the code.

If you want to collaborate you can do it in many ways:

Free software project management: if you are an experienced free software project manager it would be great if you help us to develop this one.

[4]Let's make a little joke by paraphrasing Doctor Emmet Lathrop.

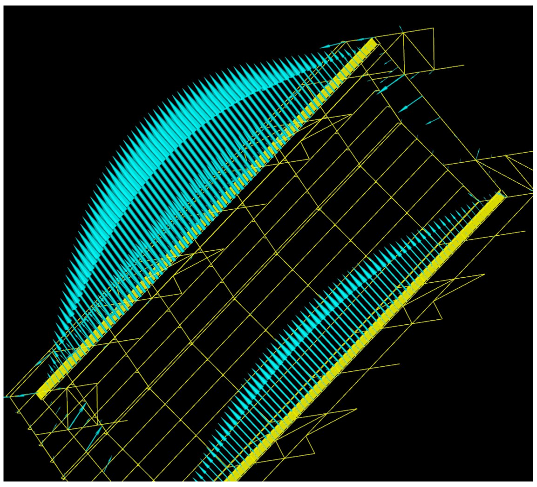
Write documentation: if you can write good English[5] and you have a good knowledge of FEA maybe you want to help us to write the documentation.

If you can read Spanish then you can help us to translate into English the comments in many C++ source files. There are also some documents written in Spanish that wait for translation.

Test code: if you know how a finite element analysis program works, you can help us to test the code. By now there are 383 verification tests that need to be documented and many verification tests that need to be written.

Write code: if you can write Python and/or C++ code maybe you can take a look here.

Good advice: maybe you can simply tell us what do you think/like/dislike about our project.



[5]At least better than me ;-)