INDUSTRIAL TRAINING REPORT

Samuel Canadell

On the occasion of the industrial training course I choose QUANTECH ATZ SA as a suitable place to perform my internship. I was especially interested in get in contact with engineer simulation company because I would like to see how the numerical methods works on real engineer world. QUANTECH ATZ are specialized in engineer simulation processes, and therefore this experience could satisfy my necessities as well as the company can take profit of my time with them.

QUANTECH provides the industry with state-of-the-art CAE simulation software applications to solve engineering problems in the fields of metal forming, optimization of casting processes, virtual testing, damage evolution of composites parts, and others, by using intensive computer-based simulation techniques. QUANTECH, provides several products, I joined in the STAMAPCK solver team. STAMPACK is a program of sheet metal forming simulation software with more than 15 years performing process simulations in the metal forming industry. Nowadays is a multipurpose and multistage simulation software into a single software suite. Sectors covered are automotive, aeronautics/aerospace, transport, metal packaging, home appliance, electronic instruments and other sectors.

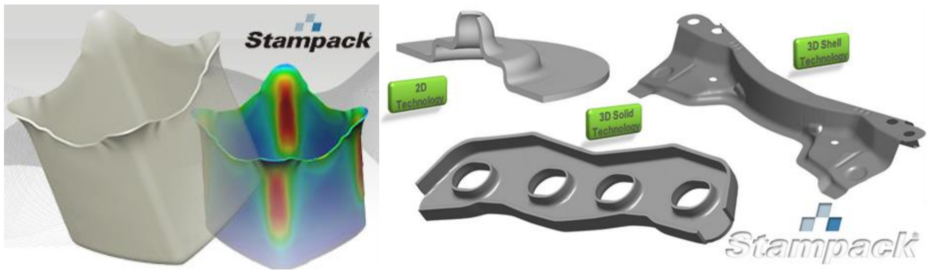


Fig.1 Industrial applications of the Stampack code (http://www.quantech.es)

Notice that Marc Puigpinós, which is a classmate, also gets involved with the company, and therefore we together develop some of the assigned tasks. My internal tutor Dr. FERNANDO RASTELLINI, show me a list of topics which I could performed as internship, and finally we decided that me and Marc could work together and improve the contact search code. I found that it was a good opportunity to participate not just in the design of the method in addition programing the implementation and testing it, which definitely was what I was highly interested.

Problem statement

The contact detection problem statement, in terms of the Stampack application, is to find which nodes of the master set, are close to the elements of the slave set. Fernando told me that the code was taking too much time for finding the first contact, and he explained that this was due to the current method is all to all, which it is straight forward to implement and it is robust, but the point is that it is extremely slow even more for large problems.

In order to improve the performance on the first contact finding, Fernando suggest to consider the Octree method, because it is extensively used in other commercial codes. An octree is a tree data structure in which each internal node has exactly eight children. Octrees are most often used to partition a three dimensional space by recursively (until some condition is achieved, for instance that the existence of two materials inside a subspace) subdividing it into eight octants.

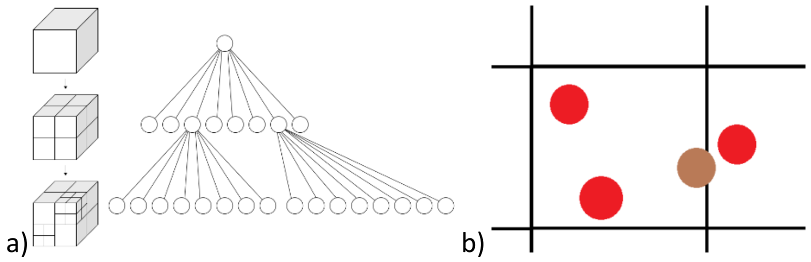


Fig3. a) Octree recursive subdivision of a cube into octants, b) and the bordering problem

I spend few days reviewing on Octree information, and we found plenty of examples showing good performance and the application in several fields. Therefore we decided to write a pseudocode, but then we figured out that we were not taking profit of the materials sets connectivity's. The octree algorithm has the inconvenient that may one particle is located on the border of a subspace (Fig 3b), and then may it is not considered to be in contact with a very close particle of the bordering subspace. The solution of this can be solved just considering that every particle can belong to more than one subspace, but this involves more complexity at implementation level. Those are the reasons why we focused on improve the Octree method. After some cost estimations, we decided that definitely we were more interested in design a particular method for solve the contact search.

Contact Search Cube Method (CSCM)

We follow the idea of a fixed subspace grid in order to just create this mesh space once as well as the classification of the entities (nodes and elements). Finally the idea of CSCM becomes to create a cube mesh and then classify nodes on the cubes where they belong, but regarding the elements classification do it also on the envelope of the cubes where they belong to. In that way, we overcome the problem of the bordering particles, because the elements are classified in such a way that they are "belonging" also into a bordering cubes.

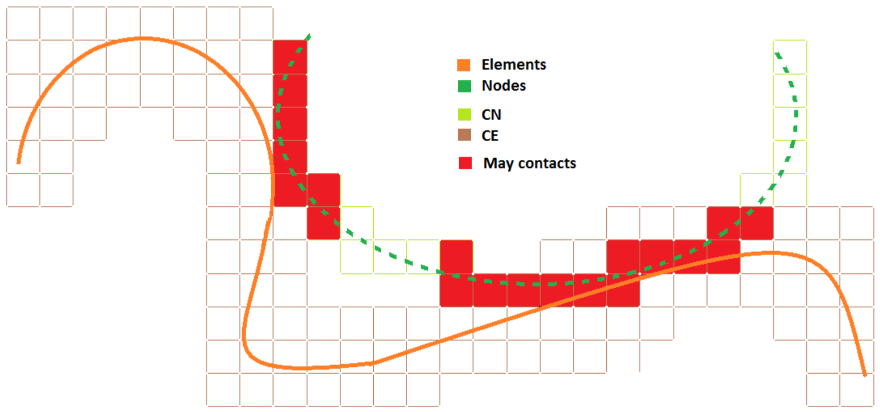


Fig2. CSCM strategy

CSCM implementation 2D

Regarding the implementation of the method, first of all, we implemented in 2D and in a *Matlab* environment, in order to test the idea and check the computation cost of the method.
The following result (Fig4) is an example where the blue dots represents that are in contact to the upper red ones. In Fig5 there is depicted the behavior of two methods: CS classifies the nodes using the x,y,z advancing comparison (explained in CSCM implementation 3D, which is the next section), and the R which classifies the nodes using the all-to-all comparison. The results depicted shows that the smart implementation (CS) present much better results than the robust approach, because the computational cost remains constant for large values of cubes.

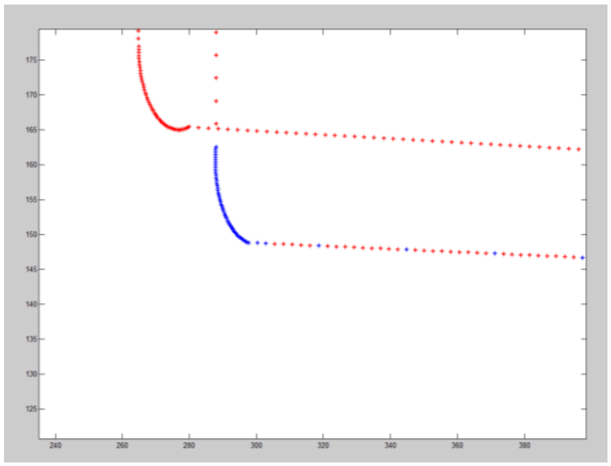


Fig4. Testing of the 2D Matlab implementation

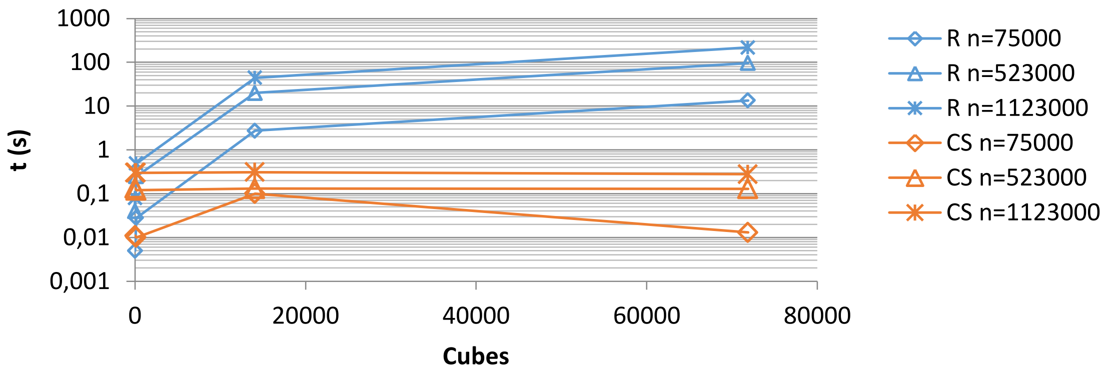


Fig5. Computational cost of the all-to-all method (R), and the Cube Search method (CS)

CSCM implementation 3D

As we were satisfied with the implementation in 2D, we begin the implementation of the 3D case in FORTRAN. As in the 2D case, me and Marc developed different functions and then we added to the code. The code follow the following structure:

1. Cube mesh generator
2. Master nodes classification
3. Slave elements classification
    a. Remesh method (Marc)
    b. Edge method (Samuel)
4. May contact
5. Nearest

This document just comments on the *nodes classification* and the *Edge method* for the elements classification. The first one is based on the classification of the *x, y, z* coordinates of the node, and comparing it with the cube ones.

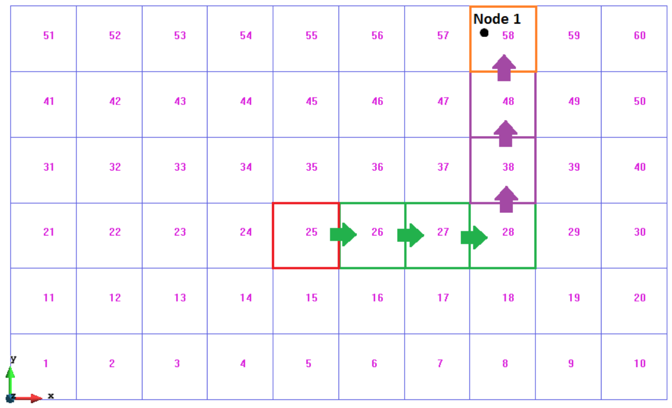
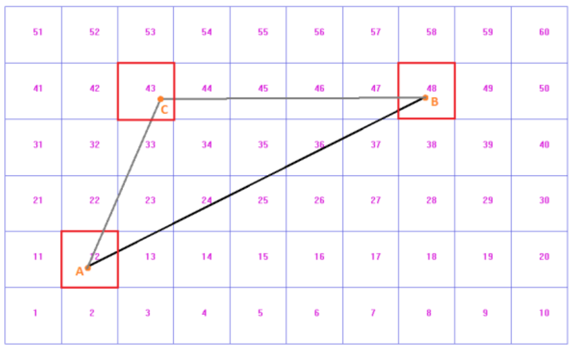
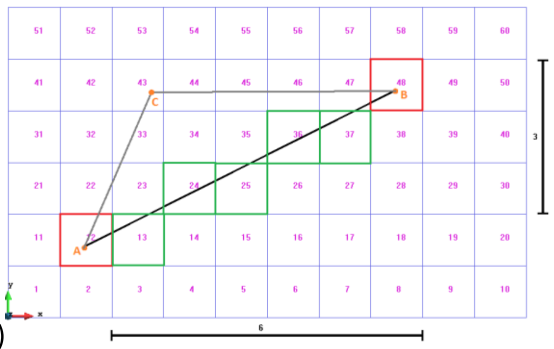


Fig6. Nodes classification procedure

We advance until the *x* is found, then the y and finally the z one, as it is shown in Fig7. The performance of this method is the best one of the methods considered because it takes benefit of the nodes connectivity's as well as the *x, y, z* advancing procedure reduces the number of comparisons.

Marc implemented a "remesh + nodes classification" in order to perform the elements classification, whereas I focused in a faster procedure. Interpolate the cube's classification along the edges as it is depicted in Fig7. The edge elements classification method has shown an excellent performance comparing with other elements classification approaches.

a) 

b) 

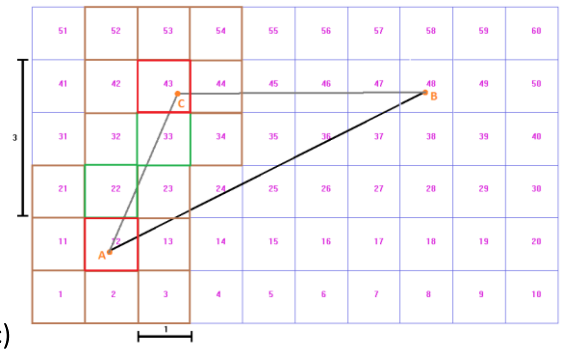
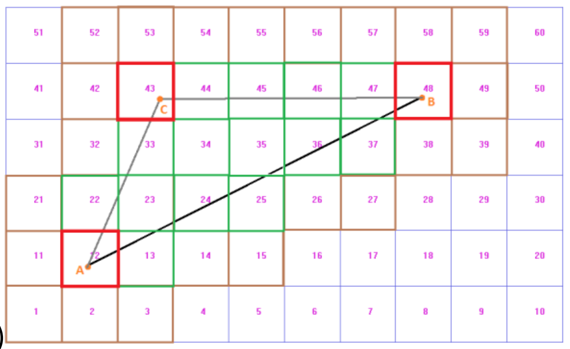
c)  d) 

Fig7. Edge method for the elements classification: a) classification of the 3 corners, b) advancing cubes along the main edge, c) advancing from the main edge cubes until the 3rd corner, d) the whole element domain is captured.

We also prepared a document for the company in order to report all the work we had done, as well as a detailed explication of the code and other functions that were created but finally it were inefficient or just unnecessary.

Testing the 3D implementation

I performed several test, problems with different geometries, where the goal is to capture the nodes of the master surface which are "candidates" to be in contact to the elements of the slave surface. The first example is the one depicted on Fig8, where a sphere is almost to contact against two straight plates. From the results it can be seen that the CSCM method is working as expected.

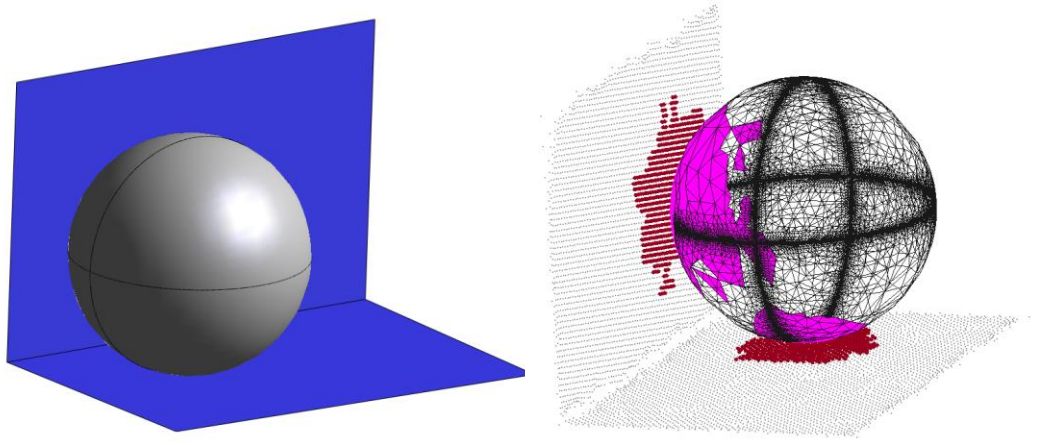


Fig8. a) Geometry statement, b) nodes in red are candidates to contact with pink elements.

The last part of the internship, we assembly the code on the STAMPACK program. This phase has not finished yet because the CSCM implementation is not optimized, and the performance is supposed to improve even more. As it can be seen in Fig9 there is an example performed on STAMPACK, the results from the old contact search and the new one are identical but with much less time cost. Note that the frequency for refreshing the contact must be lower in the CSCM version, this is due to a particular parameter of the method (influence radius) which currently we are studying in order to optimize it.

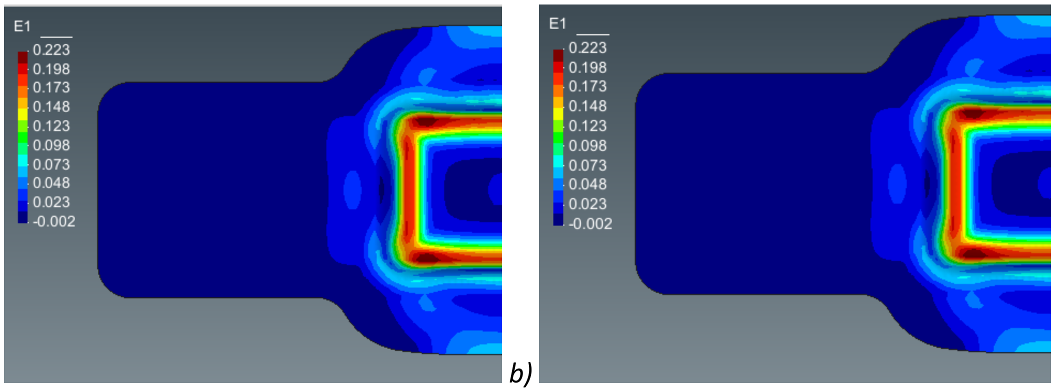


Fig9. Real STAMPACK application: a) old version with freq=10.000 took 0:24:23 s, b) CSCM version with freq=1.000 took 0:08:57 s

Conclusions

Along this experience I learned several thinks that I will not forget. One of the most valuable ideas which I would like to remark is the importance of get a powerful idea to design a method. The kernel of the method must be well defined from the beginning and try to consider the limitations.
Even that this is not a finished history, and probably we will continue finishing the CSCM implementation, I would like to acknowledge Dr. Fernando Rastellini and Dr. Ariel Eijo for the tasks supervising and drive this work with their experience and programming tips, which of course I will not forget easily. Finally, I would like to thank Marc Puigpinós for his patience, it is a pleasure share this experience with him, most especially helping each other with a widely smile.

Signed: Agreed:

Samuel Canadell Fernando Rastellini

21th January 2016.