INDUSTRIAL TRAINING REPORT

1 Motivation.

In this section I want to explain why I choose Quantech ATZ SA to do my industrial internship. When I was searching for a place to do the internship, I was convinced to choose some job in the developer side of numerical methods, and if it was possible in a company related with industry. My background is industrial engineering, and I have been working on automotive industry as process engineer for almost two years. I was interested then, in how numerical methods relates with real life problems far away from the academic we were used to see in the master. Quantech ATZ SA, not only accomplished all these demands but they offered the opportunity to solve a really interesting problem: design and implement in Stampack (the commercial code) a fast algorithm for contact detection. The work has been developed by Samuel Canadell and me with the help of our advisor Dr. Fernando Rastellini and all the people of the Stampack team.

2 Problem proposed.

The proposed problem consists on develop a fast algorithm for contact detection. The importance of this work relies in the fact that find contact in a FE mesh requires a lot of computational time, and it is a problem in the industry. Before stating the problem is important to define two basic concepts: master and slave surfaces. The slave surface is defined by the set of nodes of the surface that “see” the contact, that means, where we want to compute the forces. The master surface is the set of elements used to compute the repulsion forces over the slave surface.

The problem statement reads: Find the minimum distance between the nodes of the slave and the elements of the master. This problem can be solved easily by doing an all-to-all algorithm. The problem of this method is that takes a lot of time when the number of nodes and elements increase.

2.1 First proposal: Octree.

The first proposal was to develop an octree algorithm. The octree algorithm is commonly used to classify nodes or particles in mesh-free methods. The method consists on finding the closest point to another one (the root) by generating successive subdivisions of the

domain and erasing the points that are not in the same division that the root. After some criteria is accomplished, the subdivisions are stopped.

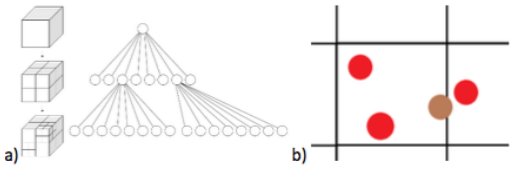

Figure 1 a) Octree structure b) Boundary problem

Figure 1 a) shows that octree methods can be interpreted as a data structure with hierarchy. The problem of this method is that it can lead to boundary problems as it is shown in Figure 1 b). Where do the brown node belong? If it belongs to the left square, the brown node will not detect the red one in its right, but if we decide that the brown node belong to both squares, then we have duplication of nodes, and we are not solving the problem stated.

Another problem arises from that method. As we are doing subdivisions recursively, we have to reclassify each time the nodes belonging to the same parent square of the root. In order to fix this problem we though about doing the octree for all the geometry and miss al the nodes and elements that belong to squares that are only fulfilled by topologies of one surface or the other (master or slave). The good thing of this idea is that we do not keep those nodes and elements that are far away, thus it should be fast. The drawbacks are that the border problems is not solved and that it will consume a lot of time generating and reclassifying the nodes and the elements. Finally we decided to use another method developed by us: CubeSearch method.

2.2 Second and final proposal: CubeSearch method.

Following the explanation of the previous section, we though in using a fixed grid instead of using a grid which is subdivided recursively. The side size of each cube of the grid is called influence radius and it is a geometrical parameter that has the physical meaning of the distance from which we consider both surfaces close enough to start contact calculations. The good thing of this method is that the computational cost is dependent on the number of cubes which is much smaller than the number of nodes or elements in a big mesh (interesting problems for industry). Another advantage is that the border problem is solved by including in the search of contact, not only the cube where both, master and surface topologies are present, but also in the neighbours cubes. That also avoids the bling points produced in convex corners. The only problem was that we should

classify nodes and elements in those cubes, and this is not fast. By the way, the nodes classification was solved easily taking profit of connectivities. As soon as we know at what cube the first node belong, we can use the connectivity matrix to classify the near nodes that will probably belong to the same cube or to the neighbours.

Once we decided to develop the method, I carried out a basic theoretical analysis, and prove that this method was faster than the so called robust (all-to-all method).

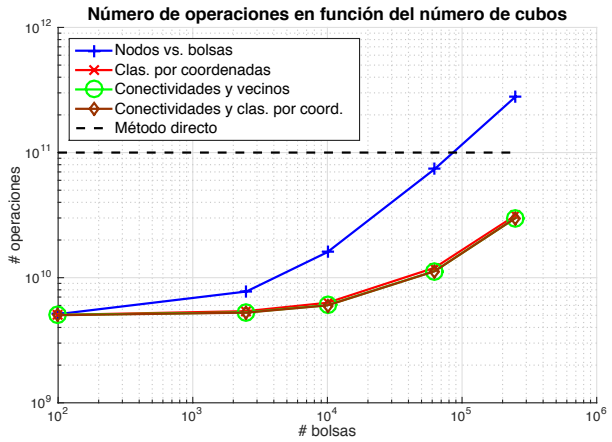

Figure 2 Number of operations as a function of number of cubes using different methods of node classification

Figure 2 show for different methods of node classification how much computational cost takes. As we can see, the computational cost depends on the number of cubes (bolsas) and we can observe that only in one case, the CubeSearch method is slower than the direct method. Using all the other methods to classify nodes, the CubeSearch method is faster. That were good news.

3 CubeSearch implementation.

The CubeSearch method was implemented in Matlab (2D case) and Fortran 90 (3D). The 2D Matlab implementation result is shown in Figure 3. As we can see, the slave surface combine blue and red points. The blue points are the ones that are close to the ones of the master surface. This exercise was done by the all-to-all method and the results were the same.

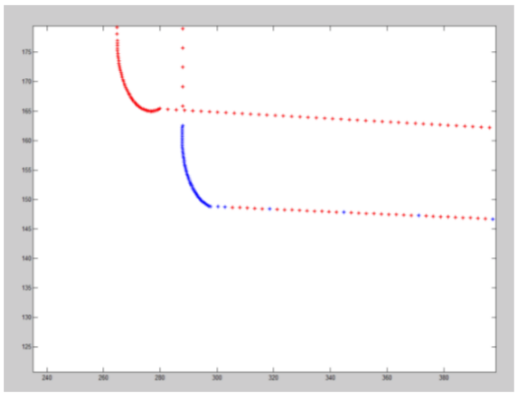

Figure 3 Contact points (blue) between two 2D surfaces.

The 3D Fortran implementation was carried out similarly to the 2D. The main difference was the element classification. The steps that the program follow to compute the possible contacts are:

1. Generate the cubes grid
2. Slave nodes classification
3. Master elements (nodes in 2D Matlab implementation) classification
    a. Remesh (Marc)
    b. Edge method (Samuel)
4. May contact
5. Nearest

The remesh method I have developed consists on compare the size of one element to the size of the cubes. If the cubes are smaller than the element, the element is remeshed. The size comparison might be fast, thus the largest edge of the element was divided by the influence radius to have a relation between sizes. The hardest developing part of this method was to create the mesh generator. The mesh generator refines a triangular element and the output is a coordinates list and a connectivity matrix. The good thing of this method was that the mesh could be non-conformal, therefore there is no need to have information about near elements.

Finally both implementations of elements classification ended and after some tests we concluded that Edge method was faster, and is the one that the CubeSearch method is using to classify elements.

4 Testing the implementation and assembling the code to Stampack.

After the code implementation was done in Fortran 90, we create some input files to visualize some examples. We did several tests with different mesh sizes and the performance with the all-to-all method was compared. We saw that our methods takes more profit compared with the all-to-all method when the number of nodes and elements is really high (which is what we are interested). In particular, we get a very good result for one million nodes of the slave and six hundred thousand elements of the muster, with a computational time of 27 seconds while the all-to-all method lasts hours.

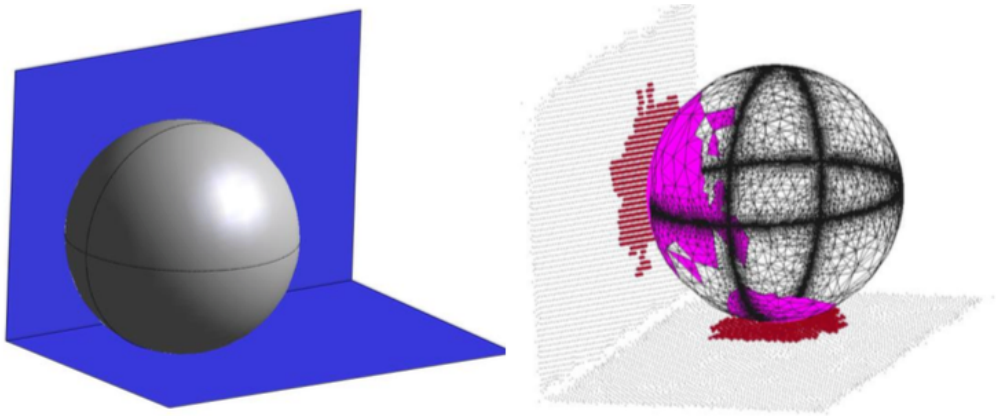

Figure 4 Output of CSM routine. Nodes of the slave in contact are colored in red and elements of the master in pink.

In the figure above, it is possible to see how the number of nodes and elements that play a role in the contact are very small compared with the total number of nodes and elements. This is what makes our method really fast.

Regarding the Stampack implementation we have to say that it is not still finished. It has been hard, since anyone of us know Fortran before starting the industrial training, and understand a commercial code like Stampack was and still is a challenge. At the moment we have implemented the our search algorithm but not in an efficient way (this is what we are doing now). The following figure shows that the results obtained are pretty similar and even the implementation is not efficient, it seems to save time with respect the present method implemented in Stampack.

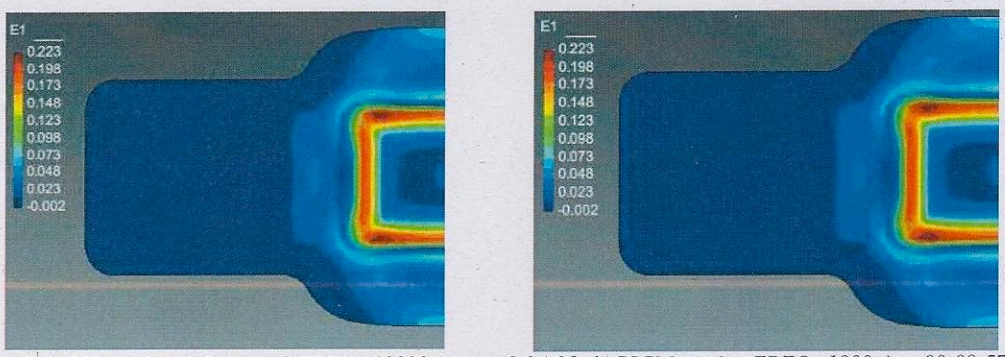

Figure 5 a) Old version of Stampack FREQ=10000, time = 0:24:23; b)CSCM version FREQ=1000 time 00:08:57

5 Conclusions

The main conclusion of the industrial training is that when coding for industry, it is not only important to have a good method, but a good implementation is mandatory. Thus, it is very important to have a good algorithm designed in paper before start coding a method directly. Finally I want to comment that I have learnt a lot of Fortran as well as code design.

6 Acknowledgements

I would like to acknowledge Dr. Fernando Rastellini and Dr. Ariel Eijo for giving me this opportunity and for all the time they have spend with me and Samuel. I would like to thank QUANTEZ ATZ SA also to trust in our job. Finally I want to thank Samuel Canadell to share this internship with me. We have had a good experience together and we also practiced and learnt how to work in group.

Signed:

Marc Puigpinós Blázquez

Agreed:

Dr. Fernando Rastellini

27-01-2016