Fast algorithm for contact detection based on classification by cubes.

M. Puigpinos

Keywords: contact detection, contact mechanics, finite elements, contact problems.

In this article a low computational cost algorithm for contact detection is proposed. The motivation of this work is to reduce drastically the computational time of computing the closest projection in a pair of contact. Comparing the proposed method with the direct method (compare each node of the master with all the elements of the slave) the time is reduced in five orders of magnitude for big meshes (nodes of Master plus nodes of Slave more than one million). Another interesting result is that in all tested cases, the computational time does not exceed the one hundred of seconds. Taking into account these results, the implementation of this method in a finite element code will result in a faster computation or in an increase of precision since the contact pair could be computed more frequently.

1 Introduction

In the numerical resolution of mechanical problems is really important to compute contacts, due to they are present in multiple practical aplications. In the computation of contacts, two stages can be differentiated: the search of contact pairs and the reaction forces computation. This work is focused on the first stage.

The search of contact pairs could seem at first glance an easy task. The appearence of several articles about contact detection, shows that there is a lot of research about it around the computational mechanics community. General algorithms for contact detection such Oct-tree, were developed by the video games industry and are useful to detect contact. Most of the articles, study algorithms for contact detection related with particle methods (DEM), since this topic is one of the weak points of DEM based methods [1],[2].

When solving a contact problem, there are two main concepts that must be defined: the master is the node of the geometry that produces the repulsive force. The slave is the surface that contacts the node of the master. Therefore, the statment of the problem to solve can be written as: find the node i of the master, that has the closest projection to the surface j of the slave. A direct and easy method is to compare all the surfaces of the slave to each node of the master and compute the projections for each case and select the minimum. In a finite element mesh, the order of operations will depend on the number of nodes of the master and the number of elements of the slave. Therefore, working with big meshes will arise high computational cost. Avoiding this computational cost by selecting just the nodes and surfaces that are most probable to be in contact is the main objective of this work. Let's consider two geometries, G_1 and G_2. Let's define the master as the geometry G_1 and the slave as G_2. The number of operations of the direct method proposed before is of order $\mathcal{O}(nn_{G_1} ne G_2)$. Figure 1 shows that for 10^6 nodes of the master and 10^5 elements of the slave, the number of operations is about 10^{11}. This is a very big number of operations. Thus, if it were possible to classify the nodes of the master and the elements of the slave in 'bags' and then just take into account the bags that both have, nodes of the master and elements of the slave, then the direct method could be just applied to that contacting bags. Defining n_b as the total number of bags and n_b^c as the number of bags that both have, nodes and elements, figure 2 shows how when the percentage of $\frac{n_b^c}{n_b}$ decreases, the number of operations decreases too.

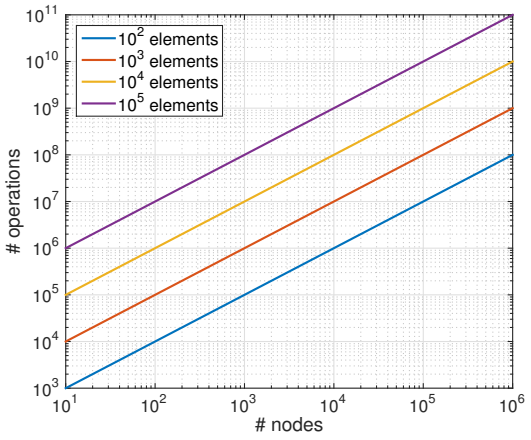

Fig. 1 Number of operations needed to compute a contact pair as a function of nodes of the master and elements of the slave.

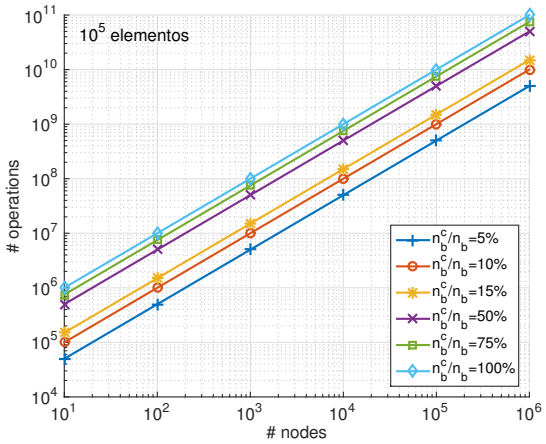

Fig. 2 Number of operations needed to compute a contact pair as a function of nodes of the master and elements of the slave, using the classification in bags.

2 Methodology

The algorithm to compute the contact pair follows the next steps:

- Define the domain.
- Compute the number of divisions.
- Generate the cube mesh.
- Classify the nodes of the master.
- Classify the elements of the slave.
- Check the bags with nodes and elements.

Henceforth, everything will be developed for a three dimensional space, therefore the bags will be called cubes, since these are the shape they adopt.

2.1 Step 1. Define the domain

Let's define the domain of geometries as the addition of both domains, the domain of the master and the domain of the slave. Then $\Omega_{ms} = \Omega_m \oplus \Omega_s$. The domain Ω is defined by the minimum and maximum components of all points in Ω_{ms}; see figure 3.

$$\begin{aligned} \Omega &:= \{\mathbf{P}_{min}, \mathbf{P}_{max}\} \\ \mathbf{P}_{min} &:= (x_{min}, y_{min}, z_{min}) \\ \mathbf{P}_{max} &:= (x_{max}, y_{max}, z_{max}) \end{aligned} \tag{1}$$

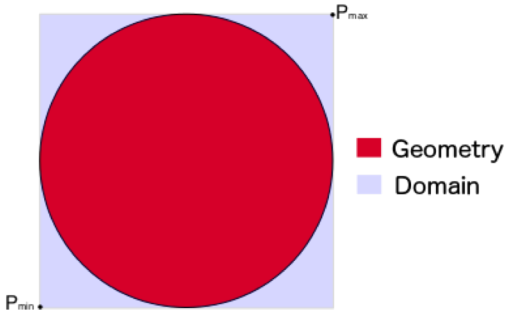

Fig. 3 Graphic representation of the rectangular domain (2D), that evolves the geometry.

2.2 Step 2. Compute the number of divisions

The number of divisions are computed by the user definition of the parameter influence radius (rI). The influence radius is the maximum distance at which a node of the master can see an element of the slave. That means, that only the elements that are closer than this distance will be included in the future computation on the closest contact pair. The computation of the number of divisions is shown in equation 2, being L_x, L_y, L_z the lengths of each side of the domain respectively. The number of cubes is the product of the number of divisions in each direction.

$$\begin{aligned} nDiv_i &= int(\frac{L_i}{rI}) \; for \; i = \{x, y, z\} \\ n_{cubes} &= nDiv_x \cdot nDiv_y \cdot nDiv_z \end{aligned} \tag{2}$$

2.3 Step 3. Generate the cube mesh

The cube mesh is defined by the minimum and maximum corner of the cubes, similarly to the domain creation; see figure 4.

$$\begin{aligned} \mathbf{C}_{min}^{icube} &= (x_{min}, y_{min}, z_{min}) \\ \mathbf{C}_{max}^{icube} &= (x_{max}, y_{max}, z_{max}) \end{aligned} \tag{3}$$

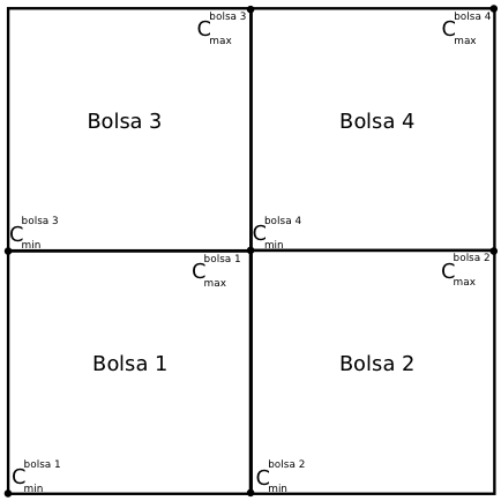

Fig. 4 Cube mesh with the correspondant corner's points that defines the mesh (2D).

2.4 Step 4/5. Classify the nodes of the master and the elements of the slave

The classification of both, nodes of the master and elements of the slave, are the key points when the algorithm is implemented in a code. Depending on how those methods are implemented the code would be very time consuming. The key point to classify the nodes of the masters consists on taking profit of connectivies. The first node has to be classified by comparing its coordinates with the coordinates of the cube and see if it is in between. Then the second node may be the neighbour of the first one (using connectivities), thus the first cube to check is the cube where the first node was classified. If the node is not in that cube, a second level search is done in the neighbours cubes. If the node is not in the neighbours cubes, then the method starts from the beginning comparing the coordinates of the second node to all cubes. The experience shows that for dense meshes, most of the nodes are classified within the neighbours. Therefore, it is not necessary to go through the whole matrix of cubes several times.

The classification of elements is more difficult since one element, if it is big enough, may belong to more than one cube. Since this subject may need a whole article, just will be mentioned the two different solutions found to classifying the elements. The first one (and a bit faster) consists on classifying the three nodes of the triangle by the use of the method that classifies the nodes of the master. After that, and always if the cube structure is known, it is possible to interpolate cubes along two edges and then project cubes to the interior. The second one consists on remeshing the element (only if the element size is bigger than the cube size), and then classify the nodes obtained from the remeshed element.

2.5 Step 6. Check the cubes with nodes and elements

The last step consists on checking all the cubes and find those that share nodes and elements. These set of nodes and elements are the most likely to be in contact and the ones used to find the minimum with the closest projection.

Figure 5 shows the theoretical number of operations needed to compute the closest projection using different methods for node classification. The results obtained are for one million nodes of the master and one hundred thousand elements. The dashed line refers to the direct method that computes the projections element by element for every node. This figure is equivalent to figure 2 but with the intelligence of the method included. The node vs. cubes method increases the number of operations much faster when the number of cubes increases. It even reaches the number of operations of the direct method for one hundred thousand elements. On the other hand, the other methods have similar cost in terms of number of operations and never exceed the number of operations of the direct method in the range of study. This result confirms the fact that the algorithm is sensitive to the implementation.

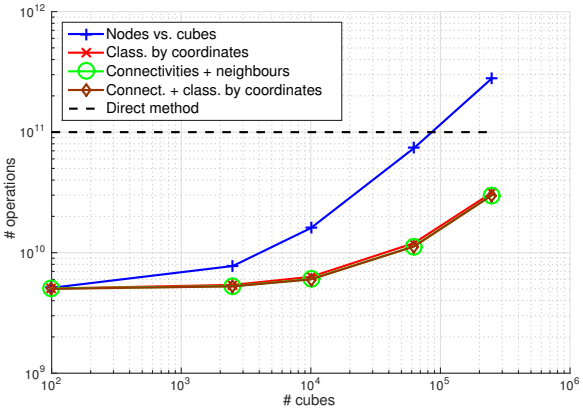

Fig. 5 Number of operations for the method of classification in cubes, including the cost of the method itself.

3 Results

In this section there are the results obtained from the tests done in geometry of figure 6. The figure 7 shows the experimental results for different meshes when increasing the number of cubes. This graphic can be compared with figure 5. The tests show the behaviour expected by the theoretical model, which is to increase the computational cost while increasing the number of cubes. Recent tests (results not avaliable in this article) show that the number of operations also increase when the number of cubes is small. The interest of this result relies on the fact that exist a number of cubes that minimize the

computational cost. The last comment about figure 7 is about the maximum computational time for one million nodes of the master and six hundred thousand nodes of the slave is about thirty seconds. Thus the results not only match the theory in a qualitative way but the results show that the algorithm is very fast too.

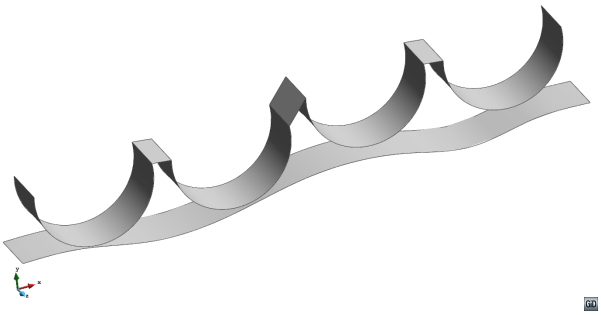

Fig. 6 Geometry curve: this geometry contains a surface with strong curvatures that are close to the master surface. The local minimums of the geomety are not anywere at the same distance from the master surface.

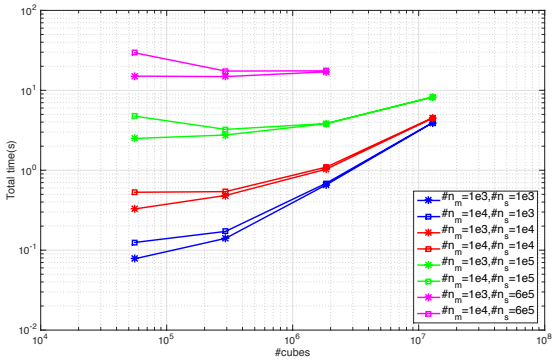

Fig. 7 Computational cost in terms of cpu time (s), for different number of nodes in the master (n_m) and slave (n_s) surfaces.

To conclude this section, the output of the method before computing the closest projection is presented. Figure 8 shows in pink, the elements and nodes that will be compared to compute the closest projection, instead of comparing the whole geometry. It can be seen that the number is much smaller and how last two concave curves are not included in that computation, since the distance between them and the master surface is higher than the parameter influence radius.

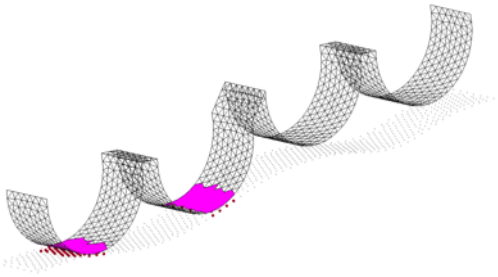

Fig. 8 Postprocessed results. Comparison between the actual nodes and elements to be compared with the total geometry.

4 Conclusions

The method proposed in this article reduces the number of candidates to be compared for contact detection by the use of a cube mesh. The main conclusions of this work are:

- The computational cost in terms of time as well as in term of number of operations is highly reduced depending on the number of cubes.
- There is a number of cubes that minimizes the computational time. It is important to select the influence radius because it defines the number of cubes and the method is very sensitive to this parameter.
- The classification of elements is solved by using two different strategies: cube interpolation and remeshing. Both methods have similar cost, being a bit faster the cube interpolation method (12%-18%).
- The theoretic results matches the experimental for moderate and high number of cubes ($> 10^4$) cubes (number of cubes tested)

The future work to be done:

- Study the influence of small number of cubes ($< 10^4$) cubes in the computational time.
- Study the memory consumption of the method, for different nodes of the master, nodes of the slave and number of cubes.
- Estimate the number of cubes for a given number of nodes and elements in order to optimize the computational time.

5 References

[1] V. Ogarko, S. Luding, A fast multilevel algorithm for contact detection of arbitrarily polydisperse objects, Multi Scale Mechanics (MSM), CTW, UTwente, Netherlands.
[2] Erfan G. Nezami, Youssef M.A. Hashash *, Dawei Zhao, Jamshid Ghaboussi, A fast contact detection algorithm for 3-D discrete element method, Elsevier, Computers and Geotechnics 31 (2004) 575587.
[3] Eric Perkins John R. Williams, A fast contact detection algorithm insensitive to object sizes, Engineering Computations, Vol. 18 Iss 1/2 (2001) 48 - 62