Node-To-Segment Contact Method

Ahmed Elgailani

June 7, 2019

1 NTS interaction description

The interaction is based on the penalty method in which we allow for interpenetration between particles and penalize it by a harmonic repulsion between the surface nodes and segments. It is customary to name the interacting surface node the "slave" node and the segment the "master".

Forces on the interacting nodes are derived from the potential $\phi = \frac{1}{2}kd^2$ (figure 1). Where k is the penalty parameter and d is the interference (the closest approach between the interfering node and the segment). In order to minimize the amount of energy contribution from these non-physical interferences, it is desirable to set k value as high as possible without causing numerical instabilities or necessitating going for practically too small timesteps. The forces on the slave node and the two master nodes (or the two nodes defining the master segment) are given by $\frac{\partial u}{\partial r_i}$ where r_i is the i-th component of the position vector of the node.

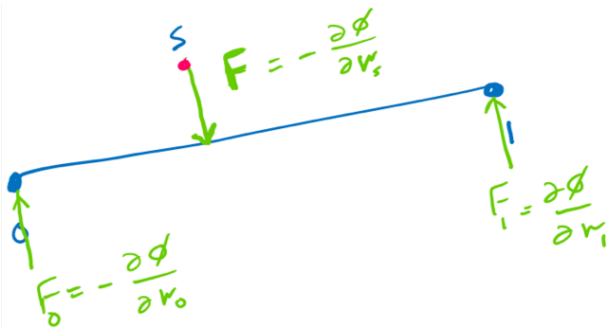


Figure 1: Repulsive forces on the slave node and on the two master nodes are derived from a harmonic potential

Consider the two meshes in figure 2, the two meshes clearly are in interference at multiple locations. The dashed black lines are the closest approaches between the slave nodes and the master surface or mesh. Notice the interchangeability of the slave-master relationship between the interacting meshes. **The idea is to find the closest distance between each slave node (any node of one mesh that lies inside another mesh) and its**

master mesh and apply a repulsive force in proportion to this closest distance. Notice that a slave node can have more than one master mesh but, clearly, it can not have more than one master segment (or node) on the same master mesh. In other words, **the closest approach between a slave node and a master mesh is the smallest d of all the d's between the slave node and the surface segments of the master mesh**.

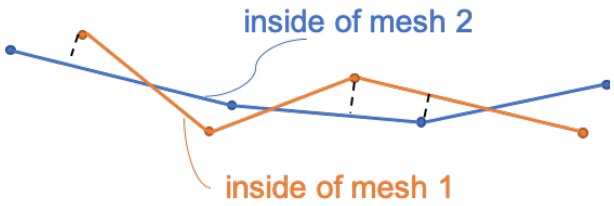


Figure 2: Two interacting meshes (particles in our case) 1 and 2. The black dashed lines indicate the closest approaches between the slave nodes and the master meshes. Notice that a mesh can be (and usually) a master and slave at the same time.

Now we know that for each slave node lies inside a master mesh, we need to loop over all (practically, only the close ones with the potential of being the closest) the surface segments of that master mesh and find the closest approach between the slave node and the surface segment. As illustrated in figure 3, a slave node can be in one of three possible situations: either the normally projected point of contact lies between the two nodes of the segment, or it lies outside on the extension of the segment to the left (closest point on the segment is node 0) or to the right (closest point on the segment is node 1). As shown in the figure 1, s is normalized parameter with a value between 0 and 1 if the normal projection of the slave node lies on the actual segment, greater than 1 if the projection lies on the extension of the segment to the right, or less than 0 if the projected point lies on the left extension. When $0 < s < 1$, the repulsive force $\boldsymbol{F}$ on the slave node is balanced by the sum of the two forces of the master nodes $\boldsymbol{F_0}$ and $\boldsymbol{F_1}$. Performing the calculations we find that $\boldsymbol{F_0} = (1-s)\boldsymbol{F}$ and $\boldsymbol{F_1} = s\boldsymbol{F}$. And in the cases $s <= 0$ or $s >= 1$ there will be only one interacting master node, node 0 or node 1 with a reaction force $\boldsymbol{F_0} = \boldsymbol{F}$ or $\boldsymbol{F_1} = \boldsymbol{F}$, respectively.

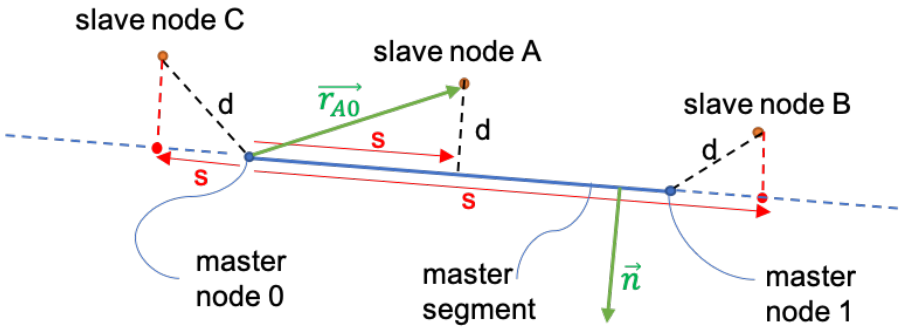


Figure 3: The slave node has one of three possible situations when interacting with a segment. Case 1: the projection of the slave node (e.g. node A) lies on the actual segment ($0 < s < 1$). Case 2: the projection of the slave node (e.g. node B) lies on the right extension of the master segment ($s >= 1$). Case 3: the projection of the slave node (e.g. node C) lies on the left extension of the master segment ($s <= 0$). In case 1, the signed closest approach d can be given by the dot product of the vector $\vec{\boldsymbol{r}}_{A0}$ (directed from node 0 to node A) and the outward unit normal $\vec{\boldsymbol{n}}$ on the master segment. For the other two cases, d is given by the magnitude of the vectors connecting the slave node and the master node (i.e. $\vec{\boldsymbol{r}}_{A0}$ or $\vec{\boldsymbol{r}}_{A1}$). In all of the cases, based on the sign of the product $\vec{\boldsymbol{r}}_{A0} \cdot \vec{\boldsymbol{n}}$, the interaction is determined as interference (-ve sign) or a gap (no contact, +ve sign). Note: the analysis is sensitive to the choice of which of the two nodes of the segment is node 0 and which is node 1. My convention is that if you walk from node 0 toward node 1 the interior of the mesh will always be to your left.

2 Linked-lists and Verlet cells

On the previous code, with node-to-node interactions, we needed only one linked-list whose size is equal to the number of surface nodes. The fact that any surface node can belong to one and only one Verlet cell made it possible to use one linked-list vector. Consider the cells stencil in figure 4a. If we tried to build the nodes linked-list, for cell E for example, as shown in figure 4b, it will be finite with a start node and end node. However, if we tried to do the same for the segments (figure 4c), we will end up with a one list connecting all the segments in the simulation because some of the segments span over multiple cells, as indicated by the red links in figure 4c, which connect th segments of cell E with the segments in other cells.

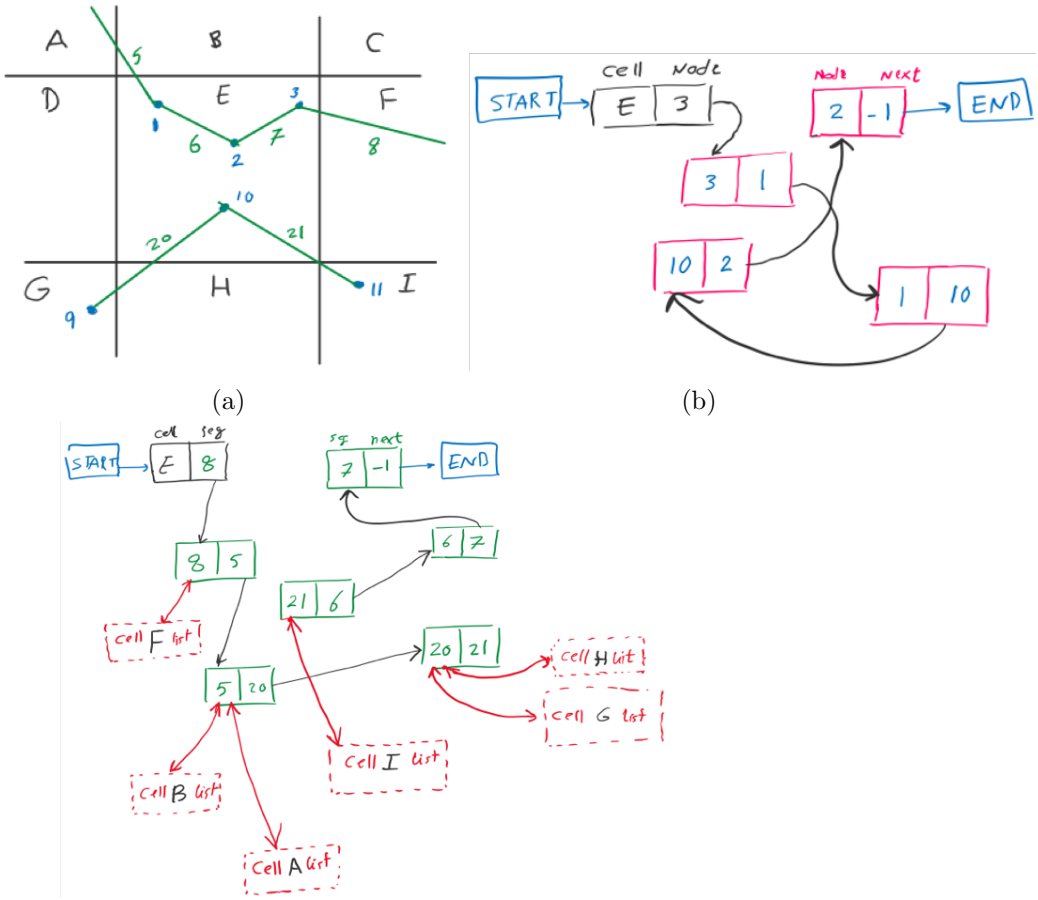
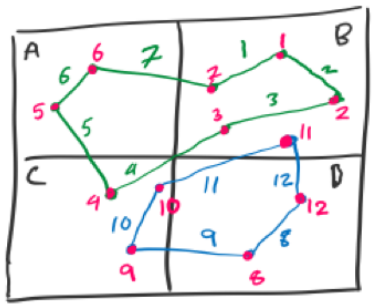


Figure 4: (a) A stencil of nine Verlet cells. Nodes belong to one and only one cell. Segments can be part of multiple cells (e.g. segments 5 and 21). (b) The nodes linked-list are one-dimensional array connecting all the nodes in a particular cell together. The blocks represent memory addresses which have two slots: the first store a node index and in the other is a pointer to address of the next node in the cell. To loop over the nodes in a particular cell you access the pointer for the first element stored in the memory address associated with that particular cell which in its turn point you to the next element until you reach the last one and exit the loop. (c) An attempt to build a one-dimensional linked-list for the segments. We see the problem created by the segments that belong to more than one cell (indicated by the blocks connected with red links in the figure) that they connect the segments of one cell with the segments of the other and the loop over one cell will not get to its end.

To avoid that, I introduced a new multidimensional linked-lists for the segments. Basically, I created a separate vector of linked-list for each cell. Let's explain that by looking at

the example system given in figure 5a. As shown in figure 5b, the nodes linked-list is a 1D vector with a finite list for any cell. Figure 5c shows how we fill the 2D linked-lists array of the system.



(a)

1	-1
2	3
3	7
4	10
5	6
6	-1
7	1
8	-1
9	-1
10	9
11	2
12	8
A	5
B	11
C	4
D	12

(b)

	A	B	C	D
1	-2	2	-2	-2
2	-2	-1	-2	-2
3	-2	1	-2	-2
4	-2	7	-1	-2
5	6	-2	4	-2
6	7	-2	-2	-2
7	-1	3	-2	-2
8	-2	-2	-2	-1
9	-2	-2	10	8
10	-2	-2	5	-2
11	-2	-2	9	-2
12	-2	-2	-2	9
	5	4	11	12

(c)

Figure 5: (a) A small system of four Verlet cells containing two meshes with a total of 12 nodes and 12 segments. (b) The nodes linked-list. (c) Segments linked-lists. They are arranged in a 2D array with one row for each surface segment and one last row reserved for the first segments in the cells. Every cell has its own column that has no connection with the other cells (this is the main purpose of having this array). Initially, the rows of the segments are filled with an arbitrary value (I use -2) to indicate emptiness. For each cell column, if the segment happened to be the last one in that cell, its corresponding slot will take the value -1 to indicate that it is the last, otherwise it will be filled with the index of the next segment in the cell or left with the value -2 to indicate that it is not part of that cell. Note: the purpose of filling the table initially with -2 instead of -1 as in the nodes linked-list is useful when populating the table. The way I populate the segments of a particular cell is by looking for the nodes in the cell and then check the two segments associated with that node. If the segment slot value is -2, it means the segment has not been listed yet and I go ahead and fill it with the index of the next segment if there is one, or with -1 if it is the last segment in the cell. If it is not -2, it means the segment has been visited before which happens frequently because every segment is associated with two nodes and in this case, we leave it as is and move on to the next segment.

In finding the cell(s) to which a segment belongs, I use a simplified method. I simply find the cells of the two lateral nodes of the segment and list the segment under both of them if they are different. However, it is possible that the segment crosses more than two cells (e.g. segment 11 in figure 5a), but here I appeal to the "right" choice of cell size. If the cell size is set correctly, even if we miss listing the segment on the cell of the slave node, it will still be part of the Verlet stencil (the group of the nine closest neighboring

cells) and would be looped over. In figure 4 example, we find that when we loop over potential interacting segments around node 8 in cell D (suppose for simplicity that we allow for interaction between members of the same mesh), although segment 11 is in cell D but is not listed so because neither its lateral nodes is inside cell D, the segment is still part of the stencil and will be visited, twice in this case.

3 Algorithm

On every step:

- Calculate the forces caused by material internal deformation on all nodes.
- Calculate surface interactions forces on surface nodes:
  - First, update Velret linked-cells of nodes and segments.
  - For each Verlet cell, loop over surface nodes in that cell (they are the slave nodes).
  - For each one of the slave nodes, loop over all master segments (they might belong to multiple meshes) in the nine cells stencil and calculate d (the closest approach) of every segment. Pick the segment with smallest d for every mesh involved separately (to allow for multiple master meshes) and save them in a map that has node-mesh pairs as the keys, and master segments for the values.
  - loop over the map of closest approaches (created in the previous step) and check for the sign of d: if -ve, calculate and add the forces; else, ignore and continue.