A 3D collision scheme for compressible media in a general connectivity lagrangian formulation

N. Bar-Gill, J. Nemirovsky, N. Har'El, O. Agmon
Rafael, Israel

Abstract

In multi-material hydrodynamic problems described using Lagrangian formulation, the interaction between different material meshes is carried out using collision schemes. The basic schemes simulate interactions using momentum transfer equations, which are solved for every pair of colliding meshes. In this paper we present a novel collision scheme, which extends the basic ones. After completing the momentum transfer stage, our scheme prevents penetration between meshes gradually, while detecting future collisions, ensuring that the penetration depth is kept under a specified limit. Collisions of more than two meshes at the same point are treated specifically, by solving vectorized momentum transfer equations. This approach, which suppresses sliding for these points, prevents numerical penetrations of mesh vertices into the interface between two other meshes. Since this special treatment is invoked only when it is necessary, it does not significantly affect the efficiency of the scheme. We demonstrate the generality of our scheme in a problem in which one mesh is calculated using a Lagrangian code, while the other mesh is calculated concurrently using an ALE code. The two meshes are coupled using our collision scheme, by passing information between them (and the codes), and by updating their boundaries' position and velocity.
Keywords: collision schemes, hydrodynamic flow, lagrangian formulation, numerical calculation.

1 Introduction

The Lagrangian formulation is commonly used in the numerical study of hydrodynamic phenomena (e.g. Wilkins [1]). In modeling multiple material problems, this formulation is supplemented by a collision scheme, which describes the interaction between the different materials. Many such schemes are known, and in this paper we present a scheme based on the vertex-face collision method (Belytschko, [2]).
Our collision scheme refines several aspects of Belytschko's scheme, by anticipating the occurrence of collisions, and limiting the hydrodynamic time-step, such that penetration of one mesh into the other is kept within a given tolerance. Occasional penetrations are corrected in several time-steps, in order to maintain hydrodynamic stability and smoothness of the solution.
A new aspect of our scheme deals with the case of a junction of three or more meshes. In the basic collision schemes, the interaction between meshes is carried out between pairs of meshes. This approach is both simple to implement, intuitive for user input, and efficient as well. However, this methodology is not rigorous enough for situations in which more than two meshes collide at a certain point (these points usually form a curve in three dimensional space). In our approach we offer special treatment for this case, such that the advantages of the simplicity of the basic scheme are mostly retained, and the rare, problematic aspects are corrected when needed.
An added benefit of our general mesh interaction scheme, is the ability to couple meshes calculated using different hydrodynamic schemes. Using a standard parallelization library (PVM, see Sunderam [3]), we are able to couple meshes calculated using entirely different codes for hydrodynamic flow.
The paper is organized as follows: Section 2 provides a brief description of the basic vertex-face collision scheme, and the refinements added here. Section 3 details the special treatment given for junctions of three or more meshes. In section 4 we describe the method in which our scheme is used to couple meshes calculated using different hydrodynamic models or codes. We then conclude, presenting possibilities for future research.

2 The Vertex-Face Collision Scheme

2.1 The Basic Algorithms

Most Lagrangian hydrodynamic formulations advance the solution by discrete time-steps, repeatedly solving the equations every time-step. The calculations carried out inside a time-step are described in Fig. 1: the accelerations at each vertex are derived from the forces acting on them (we associate mass with vertices). Then, velocities at each vertex are computed, and the vertices are then moved accordingly. At this point, the vertex movement is

translated into a volume change of the cells, and thus a change in the material density. From this, the energy equations are solved, and new energy and pressure values are given to the cells. These are then used in order to calculate the forces acting on the vertices for finding the accelerations at the beginning of the next time-step.
Our collision scheme enters this time-step cycle at two points - just after the derivation of the vertex velocities, and again after the vertices are moved.
The scheme first identifies the collisions, and solves momentum transfer equations by changing the velocities of the vertices which take part in the collision. Throughout the collision algorithm, the basic scheme deals with pairs of meshes, one of which is treated as the vertex mesh, and the other as the face mesh. (For symmetry reasons, each pair is then calculated again, with the roles of the meshes reversed).
The algorithm traverses the list of vertices of the vertex mesh, and executes for each of them. A collision is identified as follows:

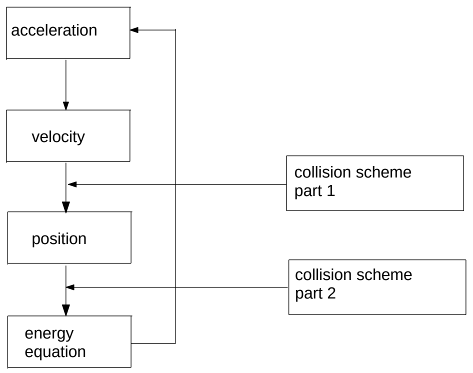

Figure 1: Lagrangian Time-Step - Inclusion of Collision Scheme

- The cells of the face mesh are geometrically arranged in a hash table, which helps to find the cells in the face mesh that might be colliding with the current vertex (of the vertex mesh).
- For each cell found in the previous step, we check whether the current vertex is in it.

After a collision is detected, the penetration point on the face of the penetrated cell is found. For brevity, we simply state that is accomplished by projecting the penetrating vertex onto the face.
Now that a collision between a vertex and a face has been pin-pointed, the momentum transfer equation between them is solved such that the penetration velocity (velocity in the direction of the normal of the face) is set equal to zero. This is done as follows:

- Since velocity and mass are associated with vertices, the momentum equation must be solved between the penetrating vertex and the vertices of the penetrated face. Therefore, we must define the penetration point as a weighted average of the face vertices. To this end, the face is divided into triangles, by connecting every vertex with the center of the face. (The face center is not a vertex of the mesh, and is used here as an auxiliary point). The weights of the vertices of the triangle to which the penetration point belongs are given by:

$$\boldsymbol{r_p} = \sum_{i=1}^{3} \gamma_i \boldsymbol{r_i} \tag{1}$$

  where $\boldsymbol{r_p}$ is the position vector of the penetration point, $\boldsymbol{r_i}$ is the position vector of the i^{th} vertex of the triangle, and γ_i is the weight of vertex i. The weight assigned to the vertex which is the center of the face, is divided equally between the vertices of the face. From this point forward, the face center is not needed, and γ_i relates to the weights of the actual vertices of the face $(i = 1, .., n)$.
- The velocity of the penetration point $\boldsymbol{v_p}$ can be linearly interpolated by:

$$\boldsymbol{v_p} = \sum_{i=1}^{n} \gamma_i \boldsymbol{v_i} \tag{2}$$

  where $\boldsymbol{v_i}$ is the velocity of vertex i of the penetrated face. If we denote the velocity of the penetrating vertex as $\boldsymbol{v_{n+1}}$, we can write the requirement that the penetration velocity must become zero as:

$$(\boldsymbol{v_{n+1}} + \Delta\boldsymbol{v_{n+1}}) \cdot \hat{n} = \sum_{i=1}^{n} \gamma_i (\boldsymbol{v_i} + \Delta\boldsymbol{v_i}) \cdot \hat{n} \tag{3}$$

  where $\hat{n}$ is the direction of the normal to the face (the direction of penetration), and $\Delta\boldsymbol{v_i}$ is the required change in velocity of vertex i.
- The momentum transfer equation between the penetrating vertex and the center-of-mass of the penetrated face can also be written as:

$$m_{n+1}\Delta\boldsymbol{v_{n+1}} \cdot \hat{n} = M\Delta\boldsymbol{v_{cm}} \cdot \hat{n} \tag{4}$$

where $M = \sum_{i=1}^{n} m_i$ and v_{cm} is the velocity of the center-of-mass of the face. The momentum of the center-of-mass can be distributed between the vertices of the face by some arbitrary coefficients β_i.

- From eqn (2),(3) and (4), we find that the change of velocity for the center-of-mass of the face is given by a **scalar** equation in the direction $\hat{n}$:

$$\Delta v_{cm} = \frac{v_{n+1} - v_p}{M\left(\frac{1}{m_{n+1}} + \sum_{i=1}^{n} \gamma_i \beta_i \frac{1}{m_i}\right)} \tag{5}$$

From the equation of angular momentum conservation,

$$\begin{aligned} M\Delta v_{cm}\hat{n} \times \left[\sum_{i=1}^{n} \beta_i \boldsymbol{r_i} - \boldsymbol{r_p} - \Delta \boldsymbol{r}\right] &= \\ M\Delta v_{cm}\hat{n} \times \left[\sum_{i=1}^{n} (\beta_i - \gamma_i)\, \boldsymbol{r_i} - \Delta r\hat{n}\right] &= 0, \end{aligned} \tag{6}$$

where $\Delta \boldsymbol{r}$ is the vector connecting the penetrating vertex and the penetration point (its projection onto the face), it follows that $\beta_i = \gamma_i$ is a suitable solution of the equations. Now eqn (5) can be solved, transferring momentum such that the penetration velocity is nullified.

2.2 Our Refinements

The second part of the collision scheme moves penetrating vertices onto the boundary of the penetrated mesh. During this phase, future collisions are also detected, and the hydrodynamic time-step is limited accordingly in order to ensure that penetration does not exceed a given value.
Each penetrating vertex, which is identified as described above (in the first part of the collision scheme), is moved back toward the boundary of the penetrated mesh, along the line connecting it with the penetration point (also identified in the first part). In order to do that we propose a **gradual** movement, which limits the volume change invoked on the cell to which the penetrating vertex belongs. An abrupt change would cause a significant change in the cell's density, pressure and energy - resulting in numerical hydrodynamic instabilities. Using the nomenclature introduced above, we define,

$$l = \left|\boldsymbol{r_{n+1}} - \boldsymbol{r_p}\right|, \tag{7}$$

$$\Gamma = min\left(\alpha \frac{\boldsymbol{v_{n+1}} dt}{l}, 1\right), \tag{8}$$

where dt is the time-step, and α is a numerical parameter. The value Γ calculated above is used to determine the amount by which the penetrated vertex is moved, according to,

$$\boldsymbol{r'_{n+1}} = (1-\Gamma)\,\boldsymbol{r_p} + \Gamma \boldsymbol{r_{n+1}} \tag{9}$$

where $\boldsymbol{r'_{n+1}}$ is the new position of the penetrating vertex. Through numerical experimentation we have found that $\alpha = 2$ produces optimal results.
In calculating the time to a future collision, we find vertices and faces which are close to each other by using the hash-table approach described in the first part of the scheme. The faces are divided into triangles, and collisions are checked between a vertex and a triangle. This is done as follows:

- Using the symbols defined above, we can write the positions of the vertices at a future time t as:

  $$\boldsymbol{r_i(t)} = \boldsymbol{r_i} + \boldsymbol{v_i} \cdot t, \quad i = 1,..,4 \tag{10}$$

  Since the vertices do not necessarily retain constant velocities, this approach is only approximate. Therefore, the time of collision found here is used in order to **limit** the hydrodynamic time-step, and not to predict exactly **when** such a collision occurs.
- A collision will occur if:

  $$\boldsymbol{r_4(t)} = \alpha \boldsymbol{r_1(t)} + \beta \boldsymbol{r_2(t)} + (1-\alpha-\beta)\boldsymbol{r_3(t)} \tag{11}$$

  where $\boldsymbol{r_{i=1,2,3}(t)}$ are the vertices of the triangle, and $\boldsymbol{r_4(t)}$ is the penetrating vertex, with $\alpha, \beta \geq 0$ such that $\alpha + \beta \leq 1$. Defining, for example,

  $$\boldsymbol{A_i(t)} = \boldsymbol{r_i(t)} - \boldsymbol{r_3(t)}, \quad i = 1,2,4 \tag{12}$$

  we can write the condition for collision as

  $$\boldsymbol{A_4(t)} = \alpha' \boldsymbol{A_1} + \beta' \boldsymbol{A_2}, \tag{13}$$

  requiring that $\alpha', \beta' \geq 0, \alpha' + \beta' \leq 1$.
- The previous equation states that $\boldsymbol{A_4(t)}$ is linearly dependent on $\boldsymbol{A_{1,2}(t)}$. In **matrix** form,

  $$\underline{A(t)} = \{\boldsymbol{A_4(t)} | \boldsymbol{A_1(t)} | \boldsymbol{A_2(t)}\}$$

  $$det\left(\underline{A(t)}\right) = 0. \tag{14}$$

  Solving the above equation reduces to finding the roots of a 3^{rd} degree polynomial in t.

The approximate time for collision found above is used to limit the time-step, such that the penetration of one mesh into the other is kept under a given limit.
The above algorithms comprise the backbone of our collision scheme.

3 Junctions of Three or More Meshes

In the previous section, our main collision scheme was presented. Note that in our scheme as well we traverse the list of colliding meshes, and treat them as pairs. This means that the algorithms deal with only two meshes at any given step of the solution. This approach has benefits in terms of intuitive user interface and computational efficiency. However, errors might occur in the occasion of collisions between more than two meshes at a certain point. Fig. 2 depicts a case in which there is no gap between meshes A and B (the gap in the figure appears for visualization purposes), and mesh C is colliding with them at their interface. It can be seen that the standard collision scheme, which collides mesh C with either mesh B or mesh A (but not simultaneously), does not prevent the penetrating vertex of mesh C from proceeding in between meshes A and B. Therefore, special treatment is needed in cases where more than two meshes collide simultaneously.
Detection of situations of multiple collisions is done as part of the standard

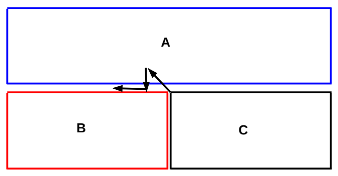

Figure 2: Schematic Example of a Vertex Sliding in between Two Meshes

collisions scheme. First, we record the number of meshes interacting with each vertex. In order to allow a vertex to be found inside more than one cell (such that it would appear to collide with more than one mesh), the cells are numerically inflated by 10% (a small factor), for this purpose alone. At the end of the standard algorithm, vertices which are found to collide with **more** than one other mesh are passed on to the correction algorithm. Since in most practical problems the number of these vertices is negligible compared to the total number of vertices, the added computation time needed for that is small as well.
The correction applied to the special vertices consists simply of nullifying their tangential velocity, in addition to the standard nullification of their penetration velocity. It is clear that this approach prevents the errors described earlier from occurring, although this solution is not rigorously exact. The desired result is achieved by repeating the collision algorithm for

the detected vertices, with the vectorized form of the momentum transfer equation (as opposed to solving the scalar equation, along the direction of the normal to the penetrated face). It should be noted that the angular momentum conservation equation must be vectorized as well:

$$M\Delta \boldsymbol{v_{cm}} \times \left[\sum_{i=1}^{n} (\beta_i - \gamma_i)\, \boldsymbol{r_i} - \Delta \boldsymbol{r}\hat{n}\right], \tag{15}$$

and from eqn (15) it follows that there is no value for β_i that will ensure conservation of angular momentum. Therefore, in lack of a better choice, we keep the definition $\beta_i = \gamma_i$ from our main algorithm for consistency. Since this solution is applied to a small number of vertices, and since the penetration depth allowed is small, the non-conservation of angular momentum is negligible.
In Fig. 3 we demonstrate the effect of the correction for multiple mesh collisions. We calculate the impact of a FSP (Fragment Simulating Projectile) on two adjacent armor plates (rolled homogeneous armor). Comparing the result on the left without the correction with the result on the right, we see numerical penetrations, which after applying the corrections, vanish.

4 Coupling Meshes Calculated separately using our Collision Scheme

The previous sections have described the algorithms that comprise our collision scheme. It should be noted that this scheme requires as input the boundary of the colliding meshes, along with velocity and mass values for the boundary vertices. Also, the scheme must be able to update the velocities and positions of the boundary vertices, such that the effect of the collisions in terms of momentum transfer and change of volume is taken into account correctly by the hydrodynamic scheme. These requirements are easily met by a variety of hydrodynamic schemes, which can be implemented even in different codes.
As an example of this approach, wrapper functions have been written, that allow the collision scheme to communicate through the parallelization library PVM with other programs running **concurrently**. Appropriate functions have been written for a Lagrangian code and for a different 3D ALE hydrodynamic code as well. This enabled us to couple a mesh calculated using the Lagrangian code for hydrodynamic flow, with another mesh calculated using the 3D ALE code. Fig. 4 shows a calculation of a sphere, hitting a plate at an oblique angle. In this example, the sphere is calculated using the Lagrangian code, while the plate is modeled using the ALE solver. Both codes are running in parallel, with the interaction between the two carried out by our collision scheme. Effectively, the scheme imposed boundary conditions (in terms of position and velocity) on the meshes.

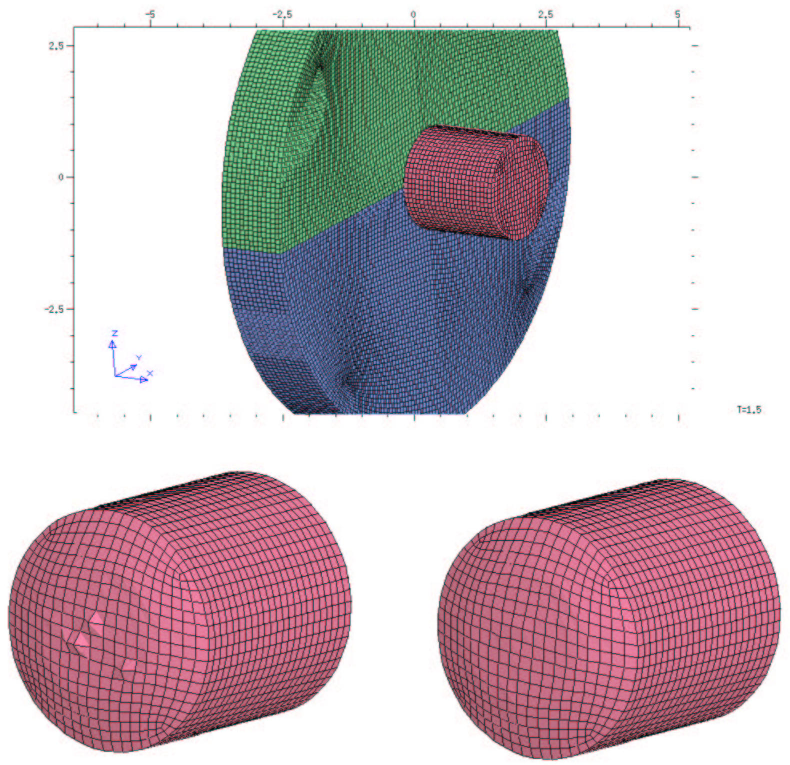

Figure 3: Impact of a FSP on two adjacent armor plates.

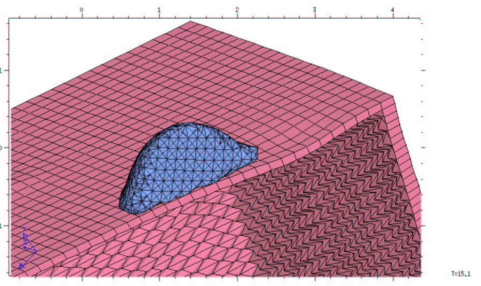

Figure 4: Coupled Lagrangian/ALE Calculation - Impact of Sphere on Plate

5 Conclusion

In describing hydrodynamic flow using the Lagrangian formulation, the boundary of meshes of different materials is well defined. The interaction

between such meshes is described by collision schemes. These schemes are well known, and usually solve momentum transfer equations between every pair of colliding meshes.
In this paper we have presented a novel collision scheme which elaborates on the basic notions mentioned above. Our scheme uses momentum transfer equations to nullify the penetration velocity of one mesh relative to the other. As an added step, the scheme corrects the position of vertices that have penetrated another mesh, by gradually moving them toward the penetration point. This is done before the energy equation is solved during the time-step, such that the solution is consistent, and numerical stability is maintained. In order to ensure that the amount of penetration allowed is kept within a given tolerance, future collisions are detected in advance, and the time of the expected collision is calculated approximately, assuming constant vertex velocities. The time-step is limited according to this result, keeping the amount of penetration small.
The basic collision scheme, in which pairs of meshes are collided at a time, allows for penetration of a vertex into a numerical gap between two meshes. In order to benefit from the computational efficiency of the standard scheme, our scheme detects situations in which more than two meshes collide at the same point (curve in 3D), and applies special treatment for these cases. For these vertices the momentum transfer equation is solved in vector form, such that sliding is suppressed. We have demonstrated that this approach solves the numerical error, with negligible effect on the efficiency of the algorithm. As an idea for future research, we propose allowing sliding of only **one** mesh relative to all others at these junctions. This should produce more accurate results for these special cases.
The general form of our scheme, which requires information regarding the position, velocity and mass of the boundaries of the colliding meshes, is suitable not only for Lagrangian hydrocodes. It has been shown that using the parallelization library PVM, we have been able to couple a mesh calculated using a Lagrangian code with a mesh calculated concurrently using a 3D ALE solver. This is done through wrapper functions which apply the collision scheme to both meshes.

References

[1] Wilkins, M.L., Calculation of elastic-plastic flow. *Methods of Computational Physics*, **3**, p. 211, 1964.

[2] Belytschko, T. & Lin, J.I., A three-dimensional impact-penetration algorithm with erosion. *Computers and Structures*, **25(1)**, pp. 95–104, 1987.

[3] Sunderam, V.S., Pvm: A framework for parallel distributed computing. *Concurrency: Practice and Experience*, **2(4)**, pp. 315–339, 1990.