Supplemental Document: An Adaptive Generalized Interpolation Material Point Method for Simulating Elastoplastic Materials

Ming Gao, Andre Pradhana Tampubolon, Chenfanfu Jiang, Eftychios Sifakis

September 11, 2017

From MPM to adaptive GIMP

Since the introduction of the material point method to the graphics community, researchers tend to focus on defining the weighting function as a direct relation between a particle and the grid nodes in its immediate vicinity, e.g. a quadratic B-spline function, as shown in Figure 1a. GIMP is another alternative which downgrades the C^1 requirement to C^0 and then uses a convolution to resume the C^1 continuity. We assign an axis aligned range to each particle as in Figure 1b and the general form for the weight is

$$w_{ip} = \frac{1}{V_p} \int_{\Omega} \chi_p(\mathbf{x}) N_i(\mathbf{x}) d\mathbf{x} \tag{1}$$

where V_p is the volume of the dashed blue box Ω, χ_p is usually an indicator function, and $N_i(\mathbf{x})$ is the basis function (e.g. a simple hat function) .

The conventional GIMP is simply to do an integration for each affected grid node. However, we propose a new interpretation which is quite essential for all the following procedures. We divide the particle range into regions which intersect with each single grid cell as $\Omega_j = \Omega \cap C_j$ and then $\Omega = \sum_j \Omega_j$. So 1 becomes

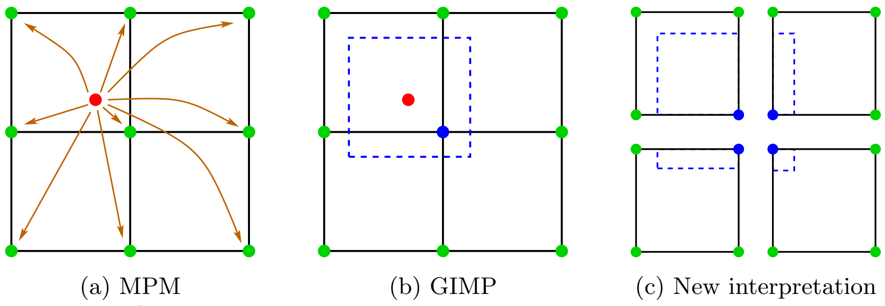
$$w_{ip} = \frac{1}{V_p} \sum_{\Omega_j} \int_{\Omega_j} N_i(\mathbf{x}) d\mathbf{x} \tag{2}$$

For example, the weight of the center grid node in Figure 1c is now the sum of the contributions computed from all four cells. When we switch to quadtrees, the corner nodes are easier to handle as in Figure 1d, and the same interpretation works in exactly the same way except that the basis function used in the right top cell is different.

For dealing with T-junction nodes (blue node in Figure 1e), we take the classical constrained hanging node treatment from octree FEM. The blue node is initially regarded as a real degree of freedom (DOF) in both smaller cells (Figure 1f), thus contributions from those two cells can be computed. As to the coarser cell, the T-junction is constrained and its property is completely determined by its parents (brown nodes). Instead of doing any weight computations in the coarser cell, we just need to redistribute the contributions collected from finer cells to its parents.

This new approach of computing weights for adaptive grids fulfills all the aforementioned requirements, and it especially enforces both partition of unity and C^1 continuity at the T-junctions, which are proved in the next sections. Furthermore, there is no principal or practical restrictions for its extension to non-graded trees. However, one obvious concern is that for dealing with one T-junction node, weight computations can go across several levels (with different basis functions, as in Figure 1g). And the corresponding data retrieval from several levels can soon jeopardize the memory bandwidth. For this performance related issue, we propose an equivalent way to convert all cross-level computations into uniform kernels, and the details are presented in the paper.

Lemma 1. ***Partition of unity*** *: given a particle, the weights of all grid nodes sum up to one* $\sum_i w_i = 1$


(a) MPM (b) GIMP (c) New interpretation

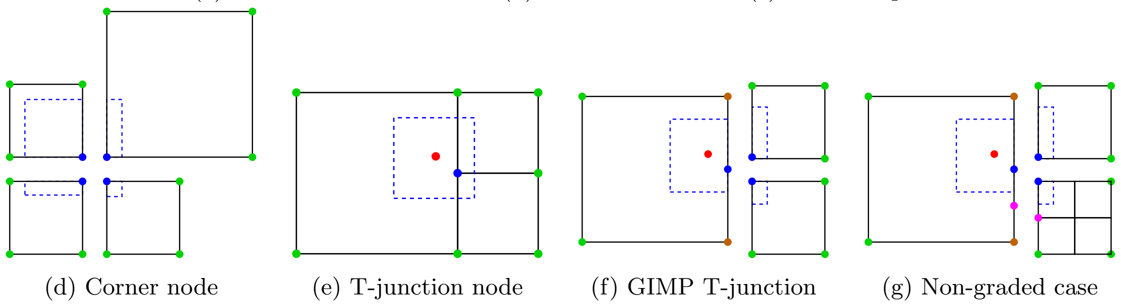

(d) Corner node (e) T-junction node (f) GIMP T-junction (g) Non-graded case

Figure 1: From MPM to Adaptive GIMP

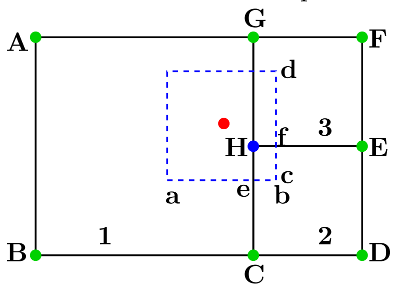


Figure 2: Quadtree

Proof. Consider the case in Figure 2, the T-junction (blue dot H) is constrained by its parents G and C. In this proof, we concern about the sum of the weights (not the continuity), thus regarding H as a real DOF (in cell 2 and 3 only) can ease the reasoning.

Taking the view of GIMP, the weight can be written as

$$w_i = \frac{1}{V}\int_a^b \int_c^d N_i \,\mathrm{dydx} \tag{3}$$

where $V = (b-a)\times(d-c)$ is the area of the particle range. Furthermore, we can divide the particle range into the intersections with each cell:

$$\begin{aligned} w_i &= w_i^1 + w_i^2 + w_i^3 && (4)\\ w_i^1 &= \frac{1}{V}\int_a^e \int_c^d N_i^1 \,\mathrm{dydx} && (5)\\ w_i^2 &= \frac{1}{V}\int_e^b \int_c^f N_i^2 \,\mathrm{dydx} && (6)\\ w_i^3 &= \frac{1}{V}\int_e^b \int_f^d N_i^3 \,\mathrm{dydx} && (7) \end{aligned}$$

In this way, each weight w_i can be splitted into three of them as $w_i^{1,2,3}$. It is clear that for some nodes, not all the three are non-zeros. For example, both w_A^2 and w_A^3 are zeros. We collect the splitted non-zero weights of cell 1 as

$$\begin{aligned}\sum_i w_i^1 &= w_A^1 + w_B^1 + w_C^1 + w_G^1 && (8)\\ &= \frac{1}{V}\int_a^e\int_c^d (N_A^1 + N_B^1 + N_C^1 + N_G^1)\,\mathrm{dydx} && (9)\\ &= \frac{1}{V}\int_a^e\int_c^d 1\,\mathrm{dydx} && (10)\\ &= \frac{(e-a)\times(d-c)}{V} && (11)\end{aligned}$$

where we used the identity $N_A^1 + N_B^1 + N_C^1 + N_G^1 = 1$ because all are standard hat functions. Similarly, we can get $\sum_i w_i^2 = \frac{(b-e)\times(f-c)}{V}$ and $\sum_i w_i^3 = \frac{(b-e)\times(d-f)}{V}$. Thus

$$\begin{aligned}\sum_i w_i &= \frac{(e-a)\times(d-c)}{V} + \frac{(b-e)\times(f-c)}{V} + \frac{(b-e)\times(d-f)}{V} && (12)\\ &= \frac{(b-a)\times(d-c)}{V} = 1 && (13)\end{aligned}$$

□

Lemma 2. C^0 ***continuity:*** *the shape function defined in the paper is* C^0 *continuous.*

Proof. In this section, we denote a modified basis function associated with grid node i by the notation $\hat{N}_i$. Consider the case in Figure 3, a particle (at position I) is moving across the interface edge GH between cell 1 and cell 3. The T-junction (blue dot H) is constrained by its parents G and C, thus it is not a real DOF anymore, and the weight from it should be redistributed to its parents. Let us focus on node G, since all

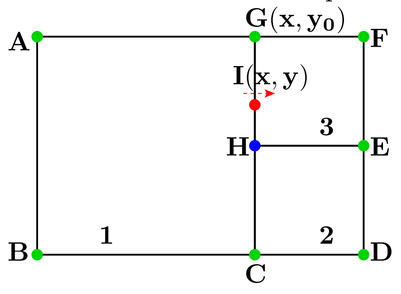


Figure 3: Quadtree

others are either the same or much easier cases.

When the particle is still in cell 1, the shape function can be computed :

$$\begin{aligned}\hat{N}_G^{1,x} &= 1 && (14)\\ \hat{N}_G^{1,y} &= 1 - \frac{y-y_0}{2\Delta x} && (15)\end{aligned}$$

where Δx is the cell size of finer cells. In this scenario, the shape function in the x-direction is always 1, we just ignore it for now.

After particle moves into cell 3, the shape function should be updated :

$$N_G^{3,y} = 1 - \frac{y-y_0}{\Delta x}. \quad (16)$$

Notice that the node H should also be computed and then redistributed to its parent as

$$\begin{aligned} N_H^{3,y} &= \frac{y-y_0}{\Delta x} && (17) \\ \hat{N}_G^{3,y} &= N_G^{3,y} + \frac{1}{2} N_H^{3,y} && (18) \\ &= 1 - \frac{y-y_0}{\Delta x} + \frac{1}{2}\frac{y-y_0}{\Delta x} && (19) \\ &= 1 - \frac{y-y_0}{2\Delta x} && (20) \end{aligned}$$

Thus $N_G^{1,y} = \hat{N}_G^{3,y}$, i.e. the shape function of node G is continuous when particle moves across the interface edge. Eventually the weight function can acquire C^1 continuity after computing the convolution of the shape function. $\square$