Examination and Analysis of Implementation Choices within the Material Point Method (MPM)

M. Steffen[1], **P.C. Wallstedt**[2], **J.E. Guilkey**[2,3], **R.M. Kirby**[1] **and M. Berzins**[1]

Abstract: The Material Point Method (MPM) has shown itself to be a powerful tool in the simulation of large deformation problems, especially those involving complex geometries and contact where typical finite element type methods frequently fail. While these large complex problems lead to some impressive simulations and solutions, there has been a lack of basic analysis characterizing the errors present in the method, even on the simplest of problems. The large number of choices one has when implementing the method, such as the choice of basis functions and boundary treatments, further complicates this error analysis. In this paper we explore some of the many choices one can make when implementing an MPM algorithm and the numerical ramifications of these choices. Specifically, we analyze and demonstrate how the smoothing length within the Generalized Interpolation Material Point Method (GIMP) can affect the error and stability properties of the method. We also demonstrate how various choices of basis functions and boundary treatments affect the spatial convergence properties of MPM.

Keyword: Material Point Method, GIMP, Meshfree Methods, Meshless Methods, Particle Methods, Smoothed Particle Hydrodynamics, Quadrature

1 Introduction

The Material Point Method (MPM) [Sulsky, Chen, and Schreyer (1994); Sulsky, Zhou, and Schreyer (1995)] is a mixed Lagrangian and Eulerian method utilizing Lagrangian particles to carry history-dependent material properties and an Eulerian background mesh to calculate derivatives and solve the equations of motion.

MPM and its variants have been shown to be extremely successful and robust in simulating a large number of complicated engineering problems (see for example [Bardenhagen, Brydon, and Guilkey (2005); Nairn (2006); Sulsky, Schreyer, Peterson, Kwok, and Coon (2007)]). The most well known of these variants is the Generalized Interpolation Material Point (GIMP) Method [Bardenhagen and Kober (2004)], of which traditional MPM is a special case. GIMP provides improved accuracy, stability and robustness to simulations through the introduction of particle characteristic functions, which in most cases have the effect of smoothing the grid basis functions. The ability to handle solid mechanics problems involving large deformations and/or fragmentation of structures, which are sometimes problematic for finite element methods, has led, in part, to the method's success.

MPM, and later, GIMP, was chosen as the solid mechanics component for fluid-structure interaction simulations within the Center for the Simulation of Accidental Fires and Explosions (C-SAFE). The goal of C-SAFE has been the development of a capability to simulate the response of a metal container filled with explosives to a large hydrocarbon pool fire, including heat up, ignition and rupture of the container. The pioneering work of Kashiwa and co-workers [Kashiwa, Lewis, and Wilson (1996)] inspired this choice as they had demonstrated many of the capabilities that would be required for such simulations, including material failure and solid-to-gas phase

[1] School of Computing, University of Utah, Salt Lake City, UT, USA. {msteffen,kirby,berzins}@cs.utah.edu
[2] Department of Mechanical Engineering, University of Utah, Salt Lake City, UT, USA. {philip.wallstedt,james.guilkey}@utah.edu
[3] Corresponding Author

transition. To achieve the required level of parallelization and to provide a platform for Adaptive Mesh Refinement, C-SAFE investigators created the Uintah Computational Framework (UCF) [Parker, Guilkey, and Harman (2006)]. It is within this software environment that the implementations of MPM and GIMP under consideration here exist, along with components for fire simulation, compressible reacting flow, and fluid-structure interaction.

The main goal of this paper is to examine some of the implementation choices within GIMP in a multi-dimensional simulation setting and to understand the algorithmic and numerical ramifications of those choices. Specifically, we will focus on the smoothing length parameter (or the particle characteristic function) and examine a few choices for evolving the smoothing length in time which have been implemented within the UCF. We will perform analysis and carry out simulations in both 1-D and 3-D in order to shed light on the error and stability properties that result from the various choices.

This paper is organized as follows. Section 2 provides background to give context concerning where and how MPM fits into the family of particle and meshfree methods and introduces previous analysis performed on MPM. Section 3 gives an algorithmic overview of MPM and GIMP, focusing on some of the choices made when implementing MPM within the UCF. Section 4 provides an analysis and interpretation of some of the spatial errors present in MPM and GIMP, building on previous analysis by the authors. In the process, we investigate the relationship between GIMP as implemented in the UCF and MPM using B-spline basis functions. Section 5 overviews the process for developing interesting problems with analytical solutions which can be used to test our methods and measure errors in our solutions. In Section 6 we present numerical results and discuss the differences that result from the aforementioned choices. Lastly, Section 7 is a summary of our findings and our conclusions.

2 Background

The Material Point Method was introduced by Sulsky, Chen, and Schreyer (1994); Sulsky, Zhou, and Schreyer (1995) as a solid mechanics extension to FLIP (Fluid-Implicit Particle) [Brackbill and Ruppel (1986); Brackbill, Kothe, and Ruppel (1988)], a "full-particle" Particle In Cell (PIC) fluids simulation method.

More recently, Bardenhagen and Kober (2004) generalized the development that gives rise to MPM and showed that MPM can be considered a subset of their "Generalized Interpolation Material Point" (GIMP) method.

Although not derived directly from what are classically considered as meshfree or meshless methods, MPM falls within a general class of meshfree methods and is discussed within the meshfree community since it has both many of the same advantages and many of the same challenges as other meshfree methods [Li and Liu (2004)]. Like many meshfree methods, the primary partitioning of the material does not involve a polygonal tessellation (as in finite elements), but rather some alternative non-mesh-based unstructured representation. However, unlike fully mesh-free methods, such as the Meshless Local Petrov-Galerkin Method (MLPG) [Atluri and Zhu (1998); Han, Rajendran, and Atluri (2005); Han, Liu, Rajendran, and Atluri (2006); Atluri (2006); Atluri, Liu, and Han (2006)], MPM utilizes a background mesh to perform differentiation, integration, and solve the equations of motion. The use of a background mesh is still similar to other meshfree methods such as the Element Free Galerkin Method (EFGM) [Belytschko, Lu, and Gu (1994)]. While the background mesh is formally free to take any form, it is most often chosen for computational efficiency to be a Cartesian lattice (*i.e.* segments, quadrilaterals and hexahedra in 1-D, 2-D and 3-D respectively). These functions are used, in essence, as a means of discretizing the continuum equations, with the domain of these functions being an alternative (in the sense of versus particles) representation of the deformed configuration of the material. Nodal integration based upon particle positions as is used in other particle methods such as PIC methods[Grigoryev, Vshivkov, and Fedoruk (2002)] is employed during the solution process.

Spatial integration errors were quickly determined to be the limiting factor in the accuracy and stability of MPM. One proposed way to ameliorate the convergence problems found in MPM was to move away from the idea of nodal integration and instead think of the particles as having extent within the quadrature scheme. Bardenhagen and Kober (2004) accomplished this with GIMP by adding particle characteristic functions. There were questions, however, on how to evolve these functions in time within a multi-D simulation. Since the deformation gradient is only maintained at one point within a particle's voxel, it is unclear that the use of this information to deform particles' voxels is sufficient to maintain a partition of the deformed domain. And, if it were sufficient, it is even more unclear how to accomplish the accurate spatial integration of these deformed voxels. Ma, Lu, and Komanduri (2006) proposed another approach for evolving the particle characteristic functions by adding massless corner particles to explicitly track the deformation of a particle's voxel, or integration domain. Steffen, Kirby, and Berzins (2008) analyzed the case where nodal integration was still used, but looked at how the use of smoother basis functions drastically reduced the nodal integration quadrature errors.

3 Overview of the Material Point Method

MPM is a mixed Lagrangian and Eulerian method with particles representing the discrete Lagrangian state of a material. The history dependent properties of a material are carried and updated on the particles. A background mesh is also used, in part to solve the equations of motion. This background mesh can be non-uniform and be comprised of elements of various shapes; however for computational efficiency a uniform Cartesian grid is almost always employed. Among other benefits, a uniform Cartesian grid eliminates the need for computationally expensive neighborhood searches during particle-mesh interaction. Particle information is projected to this background mesh, from which gradients required for constitutive model evaluation (at the particles) are calculated and the equations of motion are solved. Using the solution to the equations of motion on the grid, the material state, minimally velocities and positions, is then updated at the particles.

As the above procedure is similar for nearly all variants of MPM, the main distinguishing feature between the different MPM methods in this paper is the choice of basis functions. We will start by giving an overview of standard MPM, or MPM using standard piecewise-linear basis functions. Next, we will show how MPM can be easily implemented using B-spline basis functions and mention the benefits of these smoother basis functions. The Generalized Interpolation Material Point Method (GIMP) will then be reviewed. Lastly, various options for implementing kinematic boundary conditions will be presented.

3.1 Standard Material Point Method

The MPM procedure begins by discretizing the problem domain Ω with a set of material points, or particles. The particles are assigned initial values of position, velocity, mass, volume, and deformation gradient, denoted $\mathbf{x}_p$, $\mathbf{v}_p$, m_p, V_p, and $\mathbf{F}_p$ (subscript index p is used to distinguish particle values versus an index of i for grid node values). Alternatively, instead of velocity and mass, momentum and mass density may be prescribed at the particle location, from which m_p and $\mathbf{v}_p$ can be calculated. Depending on the simulation, other quantities may be required at the material points as well, such as temperature. The particles are then considered to exist within a computational grid, which for ease of computation is usually a regular Cartesian lattice. Fig. 1 depicts a representation of a typical 2-D MPM problem.

At each time-step t^k (all of the following quantities will be assumed to be at time t^k unless otherwise noted), the first step in the MPM computational cycle involves projecting (or spreading) data from the material points to the grid. Specifically, we are interested in projecting particle mass and momentum to the grid to calculate mass and

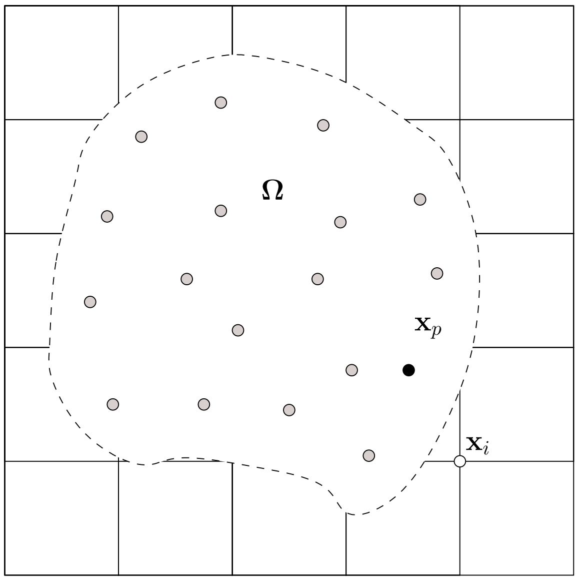


Figure 1: Typical 2-D MPM problem setup. The dotted line represents the boundary of the simulated object Ω and each closed point represents a material point used to discretize Ω. The square mesh represents the background grid. Each square in the background grid is a grid cell, and grid nodes are located at the corners of grid cells.

velocity at the grid nodes in the following way:

$$m_i = \sum_p \phi_{ip} m_p \tag{1}$$

$$\mathbf{v}_i = (\sum_p \phi_{ip} m_p \mathbf{v}_p)/m_i, \tag{2}$$

where $\phi_{ip} = \phi_i(\mathbf{x}_p)$ is the basis function centered at grid node i evaluated at the position $\mathbf{x}_p$. Note that Eq. 1 represents the mass-lumped version of what Sulsky and Kaul (2004) describe as the consistent mass matrix $M_{ij} = \sum_p \phi_{ip}\phi_{jp} m_p$. Next, internal force on the grid is found by taking the divergence of stress as a function of the constitutive model and the deformation gradient stored with each particle.

$$\sigma_p = \sigma(\mathbf{F}_p) \tag{3}$$

$$\mathbf{f}_i^{int} = -\sum_p \nabla\phi_{ip} \cdot \sigma_p V_p, \tag{4}$$

where $\nabla\phi_{ip} = \nabla\phi_i(\mathbf{x}_p)$ and $V_p = \det(\mathbf{F}_p)V_p^0$ denotes the volume of the particle voxel (in its deformed configuration). Combining the internal grid force with any external forces $\mathbf{f}_i^{ext}$, grid accelerations are then calculated as:

$$\mathbf{a}_i = (\mathbf{f}_i^{int} + \mathbf{f}_i^{ext})/m_i. \tag{5}$$

Next, grid velocities are updated with an appropriate time stepping scheme. Implicit time stepping schemes exist for MPM [Guilkey and Weiss (2003); Sulsky and Kaul (2004); Love and Sulsky (2006b)], however we choose to use the explicit Forward-Euler time discretization presented within the original MPM algorithm:

$$\mathbf{v}_i^{k+1} = \mathbf{v}_i^k + \mathbf{a}_i \Delta t. \tag{6}$$

Velocity gradients are then calculated at the particle positions using the updated grid velocities:

$$\nabla\mathbf{v}_p^{k+1} = \sum_i \nabla\phi_{ip} \mathbf{v}_i^{k+1}. \tag{7}$$

Lastly, the history-dependent particle quantities are time-advanced. Particle deformation gradients, velocities, and positions are updated using calculated velocity gradients, grid accelerations, and grid velocities:

$$\mathbf{F}_p^{k+1} = (\mathbf{I} + \nabla\mathbf{v}_p^{k+1}\Delta t)\mathbf{F}_p^k \tag{8}$$

$$\mathbf{v}_p^{k+1} = \mathbf{v}_p^k + \sum_i \phi_{ip}\mathbf{a}_i \Delta t \tag{9}$$

$$\mathbf{x}_p^{k+1} = \mathbf{x}_p^k + \sum_i \phi_{ip}\mathbf{v}_i^{k+1}\Delta t. \tag{10}$$

Eqs. 1-10 outline one time-step of MPM and assume initialization of particle values at time t^0: $\mathbf{x}_p^0$, $\mathbf{v}_p^0$, $\mathbf{F}_p^0$, and V_p^0. Non-linear finite element codes often use a staggered central difference method [Belytschko, Liu, and Moran (2000)]. If possible, a simple change of initializing particle velocities a half time step earlier, *i.e.* $\mathbf{v}_p^{-1/2}$, and using the same MPM algorithmic procedure outlined above leads to the following set of staggered central-difference update equations:

$$\mathbf{v}_i^{k+\frac{1}{2}} = \mathbf{v}_i^{k-\frac{1}{2}} + \mathbf{a}_i \Delta t \tag{11}$$

$$\nabla \mathbf{v}_p^{k+\frac{1}{2}} = \sum_i \nabla \phi_{ip} \mathbf{v}_i^{k+\frac{1}{2}} \tag{12}$$

$$\mathbf{F}_p^{k+1} = (\mathbf{I} + \nabla \mathbf{v}_p^{k+\frac{1}{2}} \Delta t)\mathbf{F}_p^k \tag{13}$$

$$\mathbf{v}_p^{k+\frac{1}{2}} = \mathbf{v}_p^{k-\frac{1}{2}} + \sum_i \phi_{ip} \mathbf{a}_i \Delta t \tag{14}$$

$$\mathbf{x}_p^{k+1} = \mathbf{x}_p^k + \sum_i \phi_{ip} \mathbf{v}_i^{k+\frac{1}{2}} \Delta t. \tag{15}$$

A similar staggered central difference method is used for MPM by Sulsky, Schreyer, Peterson, Kwok, and Coon (2007), the benefits of which are reviewed in detail by Wallstedt and Guilkey (2008).

The calculation of σ_p involves a constitutive model evaluation and is specific for different material models. The neo-Hookean elastic constitutive model used in this paper is more fully described in Section 5.1.

Calculating $\mathbf{f}_i^{ext}$ is another problem dependent procedure with several options. The first option is to calculate $\mathbf{f}_i^{ext}$ directly on the grid. This is easily done with body forces such as gravity where $\mathbf{f}_i^{ext} = m_i \mathbf{g}$. Another option is to calculate $\mathbf{f}_p^{ext}$ on the particles and project to the grid through the grid shape functions:

$$\mathbf{f}_i^{ext} = \sum_p \phi_{ip} \mathbf{f}_p^{ext}. \tag{16}$$

Moving forces, such as surface tractions can be implemented by associating the forces with a finite set of "surface" particles in conjunction with Eq. 16. Lastly, for performance reasons, MPM is typically implemented using a fixed, equally spaced Cartesian lattice, however nothing in the method requires the grid to be fixed. Moving grid nodes can be implemented to track boundary forces, calculating $\mathbf{f}_i^{ext}$ directly on the boundary nodes, however implementing this for complex geometries in multiple dimensions is not trivial.

Most standard MPM implementations use piecewise-linear basis functions for ϕ_i due to their ease of implementation, small local support, and familiarity to those in the finite element community. The 1-D form of the piecewise-linear basis function is given by:

$$\phi(x) = \begin{cases} 1 - |x|/h & : \quad |x| < h \\ 0 & : \quad \text{otherwise,} \end{cases} \tag{17}$$

where h is the grid spacing. The basis function associated with grid node i at position x_i is then $\phi_i = \phi(x - x_i)$. The basis functions in multi-D are separable functions, constructed using Eq. 17 in each dimension. e.g., in 3-D,

$$\phi_i(\mathbf{x}) = \phi_i^x(x)\phi_i^y(y)\phi_i^z(z). \tag{18}$$

3.2 B-Spline Material Point Method

As Bardenhagen and Kober (2004) described in the development of GIMP, lack of regularity in $\nabla\phi$ is conjectured to be the root cause of what is referred to as "grid cell crossing instabilities". As can be seen in Fig. 2(a), piecewise-linear basis functions are only C_0 continuous at cell boundaries. Tran, Kim, and Berzins (2007) performed a detailed analysis concerning temporal errors within an MPM fluids framework in which the grid crossing errors arising from use of piecewise-linear basis functions were precisely determined. An analysis by Steffen, Kirby, and Berzins (2008) shows how the lack of smoothness of the standard piecewise-linear basis functions (Eq. 17) causes significant spatial quadrature errors, and the use of smoother basis functions, such as B-splines, significantly reduces these errors.

A typical one-dimensional quadratic B-spline can be constructed by convolving piecewise-constant basis functions with themselves:

$$\phi = \chi * \chi * \chi / (|\chi|)^2 \tag{19}$$

where $\chi(r)$ is the piecewise-constant basis function:

$$\chi(x) = \begin{cases} 1 & : \quad |x| < \frac{1}{2}l \\ 0 & : \quad \text{otherwise} \end{cases} \tag{20}$$

and l is width of χ. The B-spline basis function associated with node i, is then $\phi_i(x) = \phi(x - x_i)$. The multi-D B-spline basis functions in MPM are not radial basis functions, as in other meshfree methods, but rather are separable, constructed using Eq. 19 in each dimension–the same way as with piecewise-linear basis functions in Eq. 18. Evaluating Eq. 19 gives the 1-D B-spline basis function:

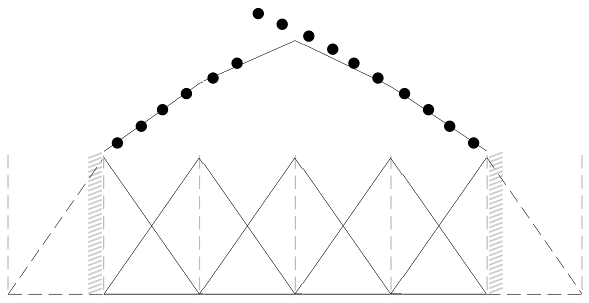
$$\phi(x) = \begin{cases} \frac{1}{2h^2}x^2 + \frac{3}{2h}x + \frac{9}{8} & : \quad -\frac{3}{2}h \le x \le -\frac{1}{2}h \\ -\frac{1}{h^2}x^2 + \frac{3}{4} & : \quad -\frac{1}{2}h \le x \le \frac{1}{2}h \\ \frac{1}{2h^2}x^2 - \frac{3}{2h}x + \frac{9}{8} & : \quad \frac{1}{2}h \le x \le \frac{3}{2}h \\ 0 & : \quad \text{otherwise.} \end{cases} \tag{21}$$

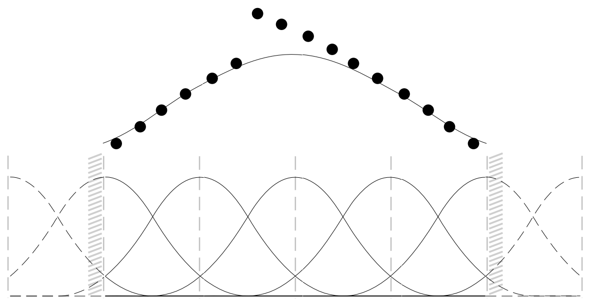
It is worth noting that a set of these quadratic B-spline basis functions maintain the partition of unity property required by the mass-lumping implicit in the MPM projection functions such as Eq. 1 and Eq. 2. An example set of these basis functions is shown in Fig. 2(b).

The above construction of B-splines basis functions by the convolution of piecewise-constant functions is helpful in showing the connection between the various basis functions considered in the paper. This convolution construction has a number of consequences, including requiring the use of extra, or ghost nodes, beyond the boundary to maintain a partition of unity within the domain. This presents no problems when implementing periodic boundary conditions, however Dirichlet boundary conditions are non-trivial to implement. One such boundary treatment will be discussed in Section 3.4.

Another option, which requires a slight departure from the convolution construction, is to modify the boundary basis functions to still maintain a partition of unity within the domain but enforce that the boundary basis function evaluates exactly to one on the boundary (which, by construction, would require all other basis functions to evaluate to zero at that boundary). One such set of basis functions was used by Steffen, Kirby, and Berzins (2008) in their simulation of a one-dimensional bar with traction forces. The basis functions used in that paper, however, can only represent functions which have zero slope on the grid boundaries, which was the case in their simulation.

Another choice of B-spline basis construction which allows for the same partition of unity property, the boundary basis functions to exactly evaluate to one on the boundary, and which also allows representation of non-zero slope solutions is the use of an open knot vector to describe the basis functions. Again, for computational efficiency,


(a) Piecewise-Linear


(b) Quadratic B-Spline

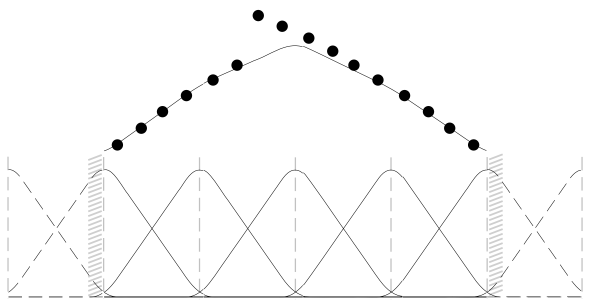

(c) GIMP

Figure 2: Example sets of 1-D basis functions used in MPM. Each set of basis functions shows an accompanying set of particles (with height representing velocity) and the corresponding velocity field on the grid after projecting particle values using Eq. 2. Piecewise-linear basis functions result in piecewise-linear velocity fields with a discontinuity of velocity gradients occurring at grid node locations. Both B-spline and GIMP basis functions result in smoother fields.

we choose the knot vector to be an open uniform knot vector with the end knots having a multiplicity of $k-1$ for a k-order B-spline. For example, if we discretize our 1-D domain of length l with N knots, the knot spacing would be $h = l/(N-1)$, and the knot vector for a set of fourth-order B-splines (consisting of third-order polynomials) would look like

$$[x_0, x_0, x_0, x_1, \dots, x_i, \dots, x_{n-2}, x_{n-1}, x_{n-1}, x_{n-1}],$$

where $x_i = x_0 + i \cdot h$. For a k-order B-spline, this results in $N+k-2$ basis functions which are calculated recursively as

$$\phi_{i,k} = \phi_{i,k-1}\frac{x - x_i}{x_{i+k-1} - x_i} + \phi_{i+1,k-1}\frac{x_{i+k} - x}{x_{i+k} - x_{i+1}} \tag{22}$$

$$\phi_{i,1} = \begin{cases} 1 & x_i \le x < x_{i+1} \\ 0 & \text{otherwise.} \end{cases} \tag{23}$$

If either denominator in Eq. 22 evaluates to zero (which only happens when knots are repeated in the knot vector), the entire term is set equal to zero.

Note that this is a slight departure from the previous MPM basis functions where there exists exactly one basis function for each grid node. Here, if grid nodes were used as knots, there are $k-2$ extra basis functions, or degrees of freedom in the system. However, the mechanics of the MPM algorithm remain exactly the same with the subscript i representing values associated with degrees of freedom, rather than values associated with grid points. Fig. 3 shows examples of these modified boundary B-spline basis functions.

3.3 Generalized Interpolation Material Point Method

The Generalized Interpolation Material Point (GIMP) Method [Bardenhagen and Kober (2004)] is an extension to MPM which takes advantage of the fact that equations such as Eq. 1 take the form:

$$\begin{aligned} g_i &= \sum_p g_p \phi_{ip} \\ &= \sum_p g_p \frac{\int_\Omega \phi_i(\mathbf{x})\delta(\mathbf{x}-\mathbf{x}_p)\,d\Omega}{\int_\Omega \delta(\mathbf{x}-\mathbf{x}_p)\,d\Omega}, \end{aligned} \tag{24}$$

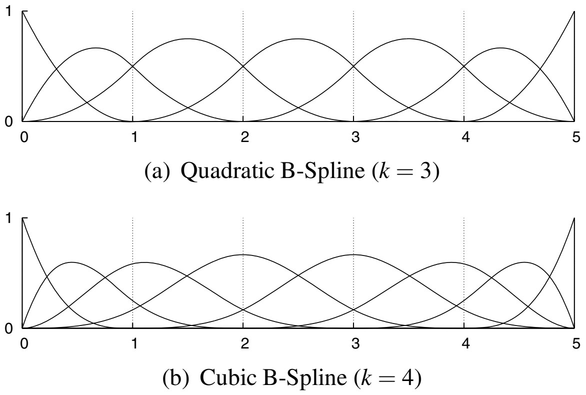


Figure 3: Example sets of modified boundary 1-D B-spline basis functions used in MPM.

with δ the Kronecker delta. GIMP then replaces δ with a general particle characteristic function $\chi_p(x)$ centered at the particle position x_p. This results in new projection equations of the form:

$$g_i = \sum_p g_p \overline{\phi}_{ip}, \tag{25}$$

where $\overline{\phi}_{ip}$ is the weighting function given by

$$\overline{\phi}_{ip} = \frac{1}{\int_\Omega \chi_p(\mathbf{x})\,d\Omega}\int_\Omega \phi_i(\mathbf{x})\chi_p(\mathbf{x})\,d\Omega. \tag{26}$$

Equations using $\nabla\phi_{ip}$, such as Eq. 4 are similarly modified to use a gradient weighting function:

$$\overline{\nabla\phi}_{ip} = \frac{1}{\int_\Omega \chi_p(\mathbf{x})\,d\Omega}\int_\Omega \nabla\phi_i(\mathbf{x})\chi_p(\mathbf{x})\,d\Omega. \tag{27}$$

GIMP is often implemented using standard piecewise-linear grid basis functions (Eq: 17) and piecewise-constant particle characteristic functions:

$$\chi_p = \begin{cases} 1 & : \quad |x| < \frac{1}{2}l_p \\ 0 & : \quad \text{otherwise,} \end{cases} \tag{28}$$

in which case the 1-D MPM and GIMP weighting functions can be grouped together in the following general form:

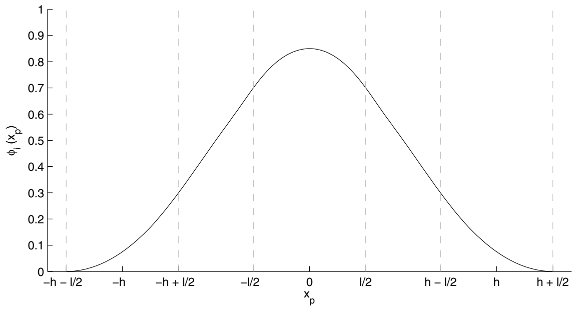
$$\phi = \begin{cases} 1-(4x^2+l_p^2)/(4hl_p) & : |x| < \frac{l_p}{2} \\ 1-|x|/h & : \frac{l_p}{2} \le |x| < h-\frac{l_p}{2} \\ \left(h+\frac{l_p}{2}-|x|\right)^2/(2hl_p) & : h-\frac{l_p}{2} \le |x| < h+\frac{l_p}{2} \\ 0 & : \text{otherwise,} \end{cases} \quad (29)$$

where l_p is the width of the particle characteristic function χ_p. Again, the basis function associated with node i located at position x_i is $\phi_i(x) = \phi(x - x_i)$ and the multi-D weighting function is constructed as the tensor product of the 1-D weighting functions in each direction.

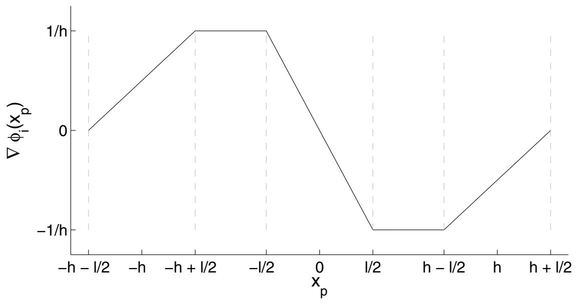
Since a particle moves and its voxel deforms in time, the question then becomes how to handle $\mathbf{l}_p$, the vector of widths of particle p's voxel in a multi-D simulation. Ideally, we would like the particles' voxels to deform and tile space for all time. In 1-D, this was accomplished by setting l_p equal to the particle's time-updated volume V_p. This scheme results in particle specific, time-dependent weighting functions $\overline{\phi}_{ip}$ and was referred to as contiguous-particle GIMP (cpGIMP). For general multi-D simulations, however, the use of rectilinear χ_p will not allow a perfect tiling to occur.

One choice for handling $\mathbf{l}_p$ in a multi-D simulation, and what we will refer to as standard GIMP, is to leave the particle lengths unchanged for all time, *i.e.* $\mathbf{l}_p = \mathbf{l}_p^0$, where the superscript ^{0} indicates initial particle size. Standard GIMP weighting functions are then particle specific, but not time-dependent. Another option is uniform GIMP (uGIMP), where a single smoothing length l is used for all particles, for all time. Note that in the case where the initial discretization was performed using uniform particles, standard GIMP would be the same as uGIMP. And lastly, while space cannot be tiled in a general multi-D simulation using rectilinear χ_p, updating $\mathbf{l}_p$ in time to give a rough approximation to the particle's deformed voxel will still be referred to as cpGIMP. The cpGIMP approximation used in this paper is $\mathbf{l}_p = \mathbf{l}_p^0\text{diag}(\mathbf{F})$, such that the particle size varies through time as dictated by the appropriate diagonal term of the deformation gradient $\mathbf{F}$. Note that l_p here refers to the full particle width, and not the half-width as used in the original GIMP formulation.

Fig. 4 shows an example 1-D GIMP weighting function $\overline{\phi}_{ip}$ and gradient weighting function $\overline{\nabla\phi}_{ip}$ for a piecewise-constant χ_p with a characteristic length of l. Notice that that while $\overline{\phi}_{ip}$ looks smooth in Fig. 4(a), the dashed lines show locations of breaks in continuity which become apparent in $\overline{\nabla\phi}_{ip}$ in Fig. 4(b). These breaks in continuity will become important in later analysis.



(a) GIMP Weighting Function



(b) GIMP Gradient Weighting Function

Figure 4: Example GIMP weighting function $\overline{\phi}_{ip}$, and gradient weighting function $\overline{\nabla\phi}_{ip}$ centered at 0 using piecewise linear grid basis functions and piecewise constant particle characteristic functions χ_p. Dotted lines denote breaks in the continuity of the functions.

3.4 Kinematic Boundary Conditions

One of the conveniences afforded by the use of a Cartesian background grid is the ease of application of kinematic boundary conditions. That is, Dirichlet or Neumann conditions, or a combination, on the velocity field. Note that if no treatment is given to the boundary nodes, then particles are able to freely advect from the computational domain in what could be considered a zero gradient Neumann condition. This important part of the algorithm has received scant treatment in the literature (although it is very relevant when one is actually implementing MPM), so we turn attention to it here.

3.4.1 Traditional MPM

In traditional MPM, boundary conditions only need be applied on those nodes which coincide with the extents of the computational domain. As illustrated in Fig. 2(a) nodes beyond those boundaries are not influenced by particles within the domain. This can be considered a result of the zero width of the Dirac delta characteristic functions. Boundary conditions must be applied to the velocity that has been projected to the nodes (Eq. 2), the time advanced velocity (Eq. 6), and the acceleration (Eq. 5). For Dirichlet conditions, this simply means overwriting the calculated values for the velocities with the prescribed values. For the acceleration, some debate exists regarding the proper means of treatment. The usual approach has been to assume that a Dirichlet condition for velocity implies that the acceleration should be zero on those boundary nodes. However, it is also possible that if Eq. 6 were solved for acceleration:

$$\mathbf{a}_i = (\mathbf{v}_i^{k+1} - \mathbf{v}_i^k)/\Delta t, \tag{30}$$

the proper value for the acceleration at the boundary nodes would be computed based on the difference between the time advanced velocity (after application of boundary conditions) and the projected velocity (without the application of boundary conditions). Put another way, acceleration on the boundary nodes can be considered to reflect the force required to bring the velocity at those nodes from the projected value to the prescribed value.

While these two approaches to the acceleration seem substantially different, the difference in simulation results is very subtle. Indeed, when both approaches are tested with the manufactured solution described in Sec. 5.1, the superiority of either is not apparent. Currently, the UCF implementation uses the boundary treatment given in Eq. 30.

In addition to prescribed velocity boundary conditions, "symmetry" boundary conditions are also frequently useful. Symmetry BCs are used to represent a plane of symmetry, which allows the use of a reduced computational domain, or a frictionless surface. They are achieved by simply applying a zero velocity Dirichlet condition on the component of velocity normal to a boundary, while allowing the other components to remain at their computed values. Acceleration is handled in the same manner, with the normal component either zeroed out, or computed as in Eq. 30.

3.4.2 GIMP and B-Spline MPM

When using GIMP or B-Spline MPM, there are additional considerations in the applications of the boundary conditions. Namely, because of their increased extents, it is possible for particles to influence, and be influenced by, nodes that lie outside of the simulation domain, (see Figures 2(b) and 2(c)). In the UCF, these are referred to as "extra" nodes, but may also be called "ghost" nodes by other investigators. Boundary condition treatment of these nodes for Dirichlet conditions is the same as for the regular boundary nodes, namely, their computed values are replaced by prescribed values as described above.

In treating symmetry boundaries, the extra nodes require special care. In particular, the normal component of velocity for these nodes is no longer set to zero, but rather should be set to the negative of the value of the node opposite the boundary. The need to do so is apparent if one considers two objects approaching a collision plane symmetrically. The normal component of velocity on the opposite sides of that plane will have opposing signs.

4 Analysis and Interpretation

MPM is a fairly new method and thus there has been a recent push by the MPM community to provide a more formal analysis of errors in the method. Bardenhagen (2002) looked at energy conservation errors in MPM, focusing on the effects of the choice of two time-stepping algorithms. Recently, Wallstedt and Guilkey (2008) expanded on the analysis of those time-stepping algorithms. Love and Sulsky (2006a) and Love and Sulsky (2006b) analyze an energy consistent implementation of MPM, the second of these showing an implicit implementation to be unconditionally stable and energy-momentum consistent. Wallstedt and Guilkey (2007) focus on velocity projection errors and present a scheme which helps ameliorate these errors. Steffen, Kirby, and Berzins (2008) perform an analysis on some of the spatial integration errors present within MPM.

In this section, we continue adding a few more pieces to the error analysis of MPM. Specifically we will look at integration errors which are affected by the smoothing of the piecewise-linear basis functions.

4.1 The Relationship between GIMP and B-Splines

Taking a closer look at the weighting function (Eq. 26), we see that the construction is essentially a convolution of the grid basis functions ϕ_i and the particle basis function χ_p. Since a standard piecewise-linear ϕ_i can also be represented as the convolution of piecewise-constant basis functions, we can rewrite the GIMP weighting function as:

$$\overline{\phi} = \chi_g * \chi_g * \chi_p / (|\chi_g||\chi_p|) \tag{31}$$

where the width of χ_g is h (the grid spacing), and the width of χ_p is l_p, as described in the GIMP methods. The equivalent GIMP basis function would then come from evaluating Eq. 31:

$$\overline{\phi}_i(x) = \overline{\phi}(x - x_i) \tag{32}$$

with the GIMP weighting function equivalent to evaluating Eq. 32 at the particle position, x_p. The reason for rewriting the GIMP basis functions in this way is to demonstrate the similarities between the construction of GIMP basis functions and the construction of B-spline basis functions as in Eq. 19. Both basis functions are constructed by convolving piecewise-constant basis functions with themselves; however all of the χ in the B-spline basis are of width h while one of the χ functions used in the GIMP method has width l_p.

In cpGIMP, the particle characteristic length l_p (of which there may be different lengths for different directions) is updated in time, meaning the cpGIMP weighting function (Eq. 29) is time dependent, and is different for each particle p. The ideal case would be that the updating of l_p in time will cause the set of particle characteristic functions χ_p to perfectly tile space, but due to the rectilinear constraints of χ_p, this is not possible in general multi-D simulations. Because of this inability to tile space, and the recognition that the major benefit of GIMP is the smoother equivalent basis functions, a simplified standard GIMP is used in which $l_p = l_p^0$ for all time. Furthermore, if $l_p = l$ is constant for all particles p in a simulation (uGIMP), the effect is truly equivalent to using standard MPM with a smoother set of basis functions. In fact, if one were to disassociate the smoothing length, l, from the particles in a uGIMP formulation and instead leave l as a free parameter, the effect is to create quadratic B-spline-like basis functions, with l determining the maximum extent of the functions. Choosing $l = l_p^0$ would give standard GIMP. Choosing $l = h$ would give quadratic B-spline basis functions. Choosing $l = 0$ would lead to the degenerate case of $\chi_p = \delta(x - x_p)$, leaving us with the standard piecewise-linear basis functions.

It has been our decision to leave the smoothing length l as a free parameter the UCF, allowing for various options when running simulations. We will explore various choices of l in the sections to follow.

4.2 Smoothing Length Dependent Integration Errors

Spatial integrals within MPM are performed using nodal integration – an approximation which takes the form:

$$\int_{\Omega} f(x)\, d\Omega \approx \sum_p f(x_p) V_p. \tag{33}$$

An analysis of errors in the above approximation within the MPM framework was performed by Steffen, Kirby, and Berzins (2008). There, the nodal integration approximations in MPM were equated to non-uniformly-spaced midpoint integration of functions with discontinuities in various derivatives. In particular, the analysis focused on the errors when calculating the internal force (Eq. 4), which involves the following approximation:

$$f_i^{int} = -\int_{\Omega} \sigma(x) \cdot \nabla \phi_i(x)\, d\Omega \approx -\sum_p \sigma_p \cdot \nabla \phi_{ip} V_p. \tag{34}$$

The main result from that analysis showed that if the particle arrangements did not respect the discontinuities which arise from the basis functions (*i.e.* a particle's voxel overhangs node boundaries), an extra integration, or "jump" error can arise in the above approximation which is of the order $C[[f^{(p+1)}]]\Delta x^{p+2}$, where the function, f, being integrated is C_p continuous (with $p = -1$ for discontinuous functions). Here, $[[\cdot]]$ represents the jump in the $p+1$ derivative of f at the discontinuity and Δx is the particle spacing. Note that the function $\nabla\phi_i$ in Eq. 34 is discontinuous, thus a jump error of $\mathscr{O}(\Delta x)$ can arise in MPM when using standard piecewise-linear basis functions, depending on particle spacing. Numerical examples of this error were shown in Steffen, Kirby, and Berzins (2008).

The jump error for a single particle is calculated as $E_{jump} = \int_{\Omega_p} f(x)\,dx - f(x_p)\Delta x$, where Ω_p spans a discontinuity, or jump, and Δx is the width of the particle. This consists of measuring the midpoint integration error for the single interval spanning the jumps. Integration approximations, such as in Eq. 33, involve integration over the whole domain, using multiple intervals, leading to a composite midpoint rule integration error which is $\mathscr{O}(\Delta x^2)$. These two errors are additive, giving a total error of the form

$$E_{total} = \int_{\Omega} f(x)\, d\Omega - \sum_p f(x_p) V_p = E_{MP} + E_{jump}, \tag{35}$$

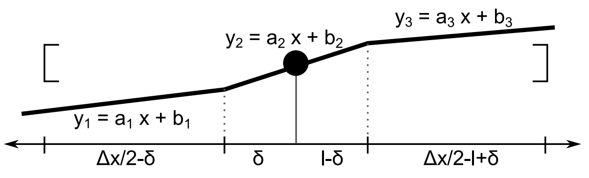
where E_{MP} is the composite midpoint error and E_{jump} is any errors arising from integrating across jumps. Note that if we assume particles are non-overlapping and fill space, this equation can also be written as

$$E_{total} = \sum_p [\int_{\Omega_p} f(x)\, d\Omega - f(x_p)V_p]. \tag{36}$$

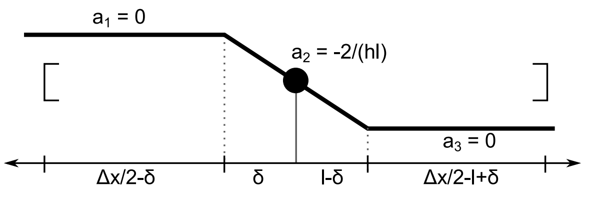
Since the errors are additive and since E_{MP} is always $\mathscr{O}(\Delta x^2)$, C_0 and higher continuous functions exhibit an overall integration error which is $\mathscr{O}(\Delta x^2)$, while discontinuous functions have an error which is $\mathscr{O}(\Delta x)$. Again, this is important because the nodal integration for the internal force calculation in Eq. 4 involves the gradients of the basis functions, which are discontinuous at grid cell boundaries when the standard piecewise-linear basis functions are used.

In uGIMP, we have the choice of a smoothing parameter l (the width of our general particle characteristic function χ), independent of the individual particle sizes, which ensures us C_1 continuous basis functions but leads to a situation which was not analyzed in Steffen, Kirby, and Berzins (2008) – the case where the width of the particle is greater than the smoothing length. In such cases (as illustrated in Fig. 5), a particle can span two jumps in the continuity of the basis functions.

Consider a general case from Eq. 36 where a single particle, or Ω_p, spans three regions of a piecewise linear function. The first region (R_1) is defined by the equation $y_1 = a_1 x + b_1$, the second (R_2) will be $y_2 = a_2 x + b_2$, and the third (R_3) is $y_3 = a_3 x + b_3$ with the particle located a distance δ inside the second region (see Fig. 5). For this



(a) General Two-Jump Function



(b) GIMP Specific Two-Jump Situation

Figure 5: Example cases of a particle's volume (the area between the square brackets) spanning two jumps in a piecewise-linear function: (a) shows a particle spanning two jumps in a general piecewise-linear function with a_i and b_i, $i = 1,2,3$ the parameters describing the linear segments, while (b) shows the specific piecewise-linear uGIMP gradient function $\nabla\phi$ and a situation where the particle size Δx is greater than the smoothing length l.

case, the integration error is given by:

$$E_{jump} = \int_{\Omega_p} f(x)\,dx - f(x_p)V_p \tag{37}$$

$$\begin{aligned} &= \int_{\Omega_p\cap R_1} y_1\,dx + \int_{\Omega_p\cap R_2} y_2\,dx + \\ &\quad \int_{\Omega_p\cap R_3} y_3\,dx - y_2(x_p)\Delta x \end{aligned} \tag{38}$$

$$\begin{aligned} &= \frac{1}{2}(a_3-a_2)l^2 + \frac{1}{2}(a_2-a_3)l\Delta x + \\ &\quad (a_2-a_3)l\delta + \frac{1}{2}(a_3-a_1)\delta^2 + \\ &\quad \frac{1}{2}(a_1-2a_2+a_3)\delta\Delta x + \frac{1}{8}(a_3-a_1)\Delta x^2 \end{aligned} \tag{39}$$

where l is the width of the center region and δ is the particle offset into the center region. Since l and δ are both less than Δx, this whole expression appears to be $\mathscr{O}(\Delta x^2)$. However, when we consider the specific case of measuring the integration error in internal force (Eq. 34 with $\sigma = 1$) when a particle spans the center region of $\nabla\phi$ as in uGIMP (see Fig. 4(b) and Fig. 5(b)), the slopes of the left and right regions in Fig. 5(b) are zero, while the slope of the center region is dependent on the smoothing length l and the grid spacing h. Specifically, for these regions, $a_1 = 0$, $a_3 = 0$ and $a_2 = -2/(hl)$. When we substitute these parameters into Eq. 39, we are left with

$$E_{jump} = \frac{-1}{h}\left[\Delta x - l + 2\left(1 - \frac{\Delta x}{l}\right)\delta\right]. \tag{40}$$

Here, it is clear that the jump error in this case is $\mathscr{O}(\Delta x)$. If instead, $\Delta x < l$ and the particle only spans one of the uGIMP jumps, the error then takes the form:

$$E_{jump} = \frac{-1}{hl}[\delta^2 - \delta\Delta x + \frac{1}{4}\Delta x^2]. \tag{41}$$

Since l no longer depends on Δx, this is now $\mathscr{O}(\Delta x^2)$.

To test this analysis, we calculate the force on a single node for a set of particles with constant stress σ. This is the same test performed by Steffen, Kirby, and Berzins (2008) when looking at a particle spanning a single jump instead of the two-jumps analyzed above. In this case, the internal force on a node is calculated as

$$f_i^{int} = -\sum_p \nabla\phi_{ip}\cdot\sigma_p V_p = -\sigma\sum_p \nabla\phi_i(x_p)V_p. \tag{42}$$

For a constant stress, internal force should be zero, so any errors are from integrating $\nabla\phi_i$. Fig. 6 shows the errors for various particle spacings when the smoothing length $l = 1/10$. As expected, when $\Delta x < l$ the error converges as $\mathscr{O}(\Delta x^2)$. When Δx becomes greater than l, the error tends towards $\mathscr{O}(\Delta x)$.

Here, we have shown errors in the internal force which are either $\mathscr{O}(\Delta x)$ or $\mathscr{O}(\Delta x^2)$, depending on the relationship between the smoothing length l and the particle widths Δx. For stability, in addition to the typical CFL constraints one needs when smooth forces exist, we need to consider further time step restrictions when force kicks arise from these integration errors. These time step restrictions would be similar to those required, as shown by Tran, Kim, and Berzins

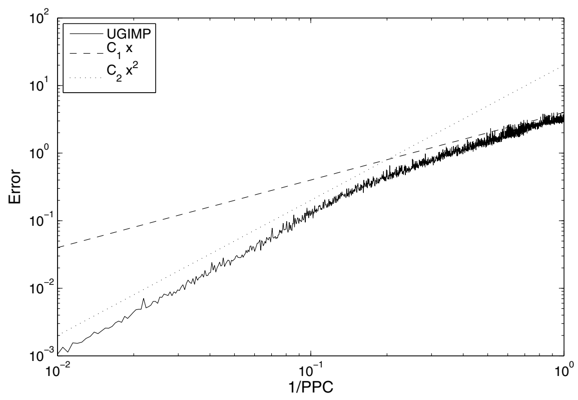


Figure 6: Errors in internal force vs. particle spacing for constant $\sigma = 1$, grid spacing $h = 1$, and smoothing length $l = 1/10$. The particles have a global uniform density, however they have a locally non-uniform spacing. Otherwise, superconvergent results are observed. Note that when particle spacing is less than the smoothing length l, the error converges as second-order. As the particle spacing becomes greater than the smoothing length, the error tends towards first-order.

(2007), due to force kicks arising from grid crossing errors. While a time step of Δt_1 may be sufficient when we are in the $\mathscr{O}(\Delta x^2)$ error region, a smaller Δt_2 may be required to control stability when we are in the $\mathscr{O}(\Delta x)$ error region.

4.3 Impact of Boundary Treatments

In MPM, the union of the particles' voxels are assumed to fill space and define the material of interest. However, many calculations are not performed directly on the particles, but rather on the background grid to which the particle information is projected. This projection of particle information leads to a set of "active" basis functions and grid cells (those which have particles in their support) which, in general, will differ geometrically than the union of particles' voxels. This can, and does, lead to a further errors in many MPM simulations.

In standard MPM with piecewise-linear basis functions, the active grid cells are those which contain a particle. One could argue that a grid cell which contains no particles but still overlaps with a particle voxel (from a particle in a neighboring cell) should also be active, but is not considered so in the current MPM framework. In either case, when considering active cells on the grid, there may be a geometric error of up to h in each direction. When moving to uGIMP, or B-splines, this geometric error can become worse since the support of these basis functions are larger. Cubic B-splines, quadratic B-splines, and uGIMP can experience geometric errors of up to $2h$, $3h/2$ and $h+l/2$, respectively. All of these errors are $\mathscr{O}(h)$; however it is important to note that these geometric errors are not only a function of how well the object of interest is aligned with grid cells, but they are also a function of basis function choice.

Some work has been performed on MPM background grids which more closely represent the material of interest. For example, Wallstedt (2008) has worked on an MLS representation of a material boundary and incorporates this boundary into the MPM integration routines. Here, we sidestep part of the issue by developing test problems in Section 5 whose boundaries are perfectly aligned with the grid boundaries (such as a fixed-fixed elastic bar). Even with these aligned test problems, geometric errors can still exist since information is projected to extra nodes outside the domain, as is shown in Fig. 2(b) and Fig. 2(c); information which is still used in standard kinematic boundary treatments.

To illustrate this geometric error, Fig. 7 shows an example of the density field resulting from projecting particles with constant mass (a discretization of a constant density field) to the grid. The density field is calculated as $\rho(x) = \sum_i \rho_i \phi_i(x)$ with $\rho_i = m_i / (\sum_p \phi_{ip} V_p)$. In this example, the constant density field spans the region $[0,1]$ which is embedded in a grid covering the region $[-0.2,1.2]$. Since the deformed configuration of the material with respect to the grid is effectively the support of the fields of interest, we can see from Fig. 7 how implementing boundary conditions and modeling contact can present a challenge when wider basis functions are used.

We postulate that, in general, all of the methods here can suffer from $\mathscr{O}(h)$ geometric errors. In the special case of boundary aligned problems,

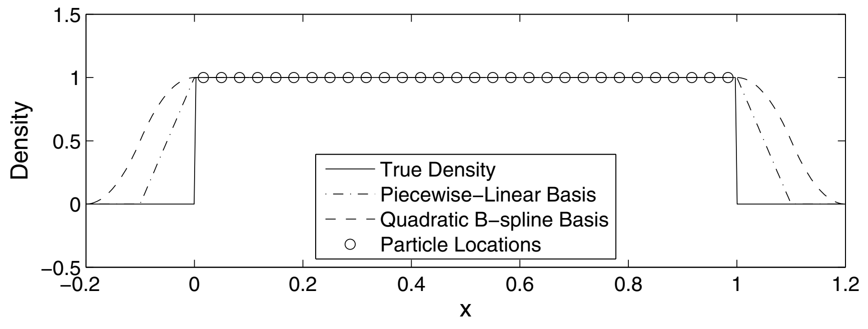


Figure 7: Density fields resulting from projecting particle mass to the background grid. The true density field is shown, along with density fields calculated with piecewise-linear and quadratic B-spline basis functions. Here we can see that the geometric extent of information projected to the grid not only depends on which grid cells contain material, but also on basis function choice.

methods where information is projected to extra nodes, such as uGIMP and standard B-splines, will still be affected by this $\mathscr{O}(h)$ geometric error when Neumann or Dirichlet boundary conditions are applied. These methods should not be affected by this error when periodic boundary conditions are used. Methods which do not require the use of extra nodes, such as standard MPM with piecewise-linear basis functions (Fig. 2(a)), modified boundary B-splines (Fig. 3), and cpGIMP will not be affected by this geometric error.

It is worth noting that while cpGIMP is implemented in the UCF using extra boundary nodes, information is not projected to these extra nodes in well-behaved boundary aligned simulations. This is because the particle p that is closest to the boundary has width l_p, and is located at a position of $l_p/2$ to the inside of the boundary, and the closest extra node is at a distance of $h+l_p/2$, which is the exact location where the extra node's basis function goes to zero (see Fig. 4).

5 Test Problem Development

Code verification has gained renewed importance in recent decades as costly projects rely more heavily on computer simulations. Full time-dependent test problems with analytical solutions are desired so that simulation errors can be assessed. The Method of Manufactured solutions [Schwer (2002); Knupp and Salari (2003); Banerjee (2006)] begins with an assumed solution to the model equations and analytically determines the external force required to achieve that solution. This allows the user to verify the accuracy of numerical implementations, understand the effects of parameter choices within the code, and to find where bugs may exist or improvements can be made. The critical advantage afforded by MMS is the ability to test codes with boundaries or nonlinearities for which exact solutions will never be known. It is argued [Knupp and Salari (2003)] that MMS is sufficient to verify a code, not merely necessary.

Since full transient mechanics solutions are often difficult to find in the literature, we will first present an overview of the method of manufactured solutions with which we will then develop both 1-D and 3-D test problems.

5.1 Method of Manufactured Solutions Overview

For this paper we define several non-linear dynamic manufactured solutions and use them for subsequent testing. The solutions exercise the mathematical and numerical capabilities of the code and provide reliable test problems for ascertaining a simulation's accuracy and stability properties.

Finite Element Method (FEM) texts often present Total Lagrange and Updated Lagrange forms of the equations of motion. The Total Lagrange form is written in terms of the reference configuration of the material whereas the Updated Lagrange form is written in terms of the current configuration. Either form can be used successfully in a FEM algorithm, and solutions from Updated and Total Lagrange formulations are equivalent [Belytschko, Liu, and Moran (2000)].

However, within GIMP it is necessary to manufacture solutions in the Total Lagrange formulation so that zero normal stress can be applied to free surfaces as a boundary condition. This might at first appear to conflict with the fact that GIMP is always implemented in the Updated Lagrange form. The equivalence of the two forms and the ability to map back and forth between them allows a manufactured solution in the Total Lagrange form to be validly compared to a numerical solution in the Updated Lagrange form.

The equations of motion are presented in Total and Updated Lagrangian forms, respectively:

$$\nabla \mathbf{P} + \rho_0 \mathbf{b} = \rho_0 \mathbf{a} \tag{43}$$

$$\nabla \sigma + \rho \mathbf{b} = \rho \mathbf{a} \tag{44}$$

where

$\mathbf{P}$	1^{st} Piola-Kirchoff Stress,
σ	Cauchy Stress,
ρ	density,
$\mathbf{b}$	acceleration due to body forces, and
$\mathbf{a}$	acceleration.

Many complicated constitutive models are used successfully with GIMP, but for our purposes the simple neo-Hookean is sufficient to test the nonlinear capabilities of the algorithm. The stress is related in Total and Updated Lagrangian forms, respectively:

$$\mathbf{P} = \lambda \ln J \mathbf{F}^{-1} + \mu \mathbf{F}^{-1} \left(\mathbf{F}\mathbf{F}^T - \mathbf{I}\right) \tag{45}$$

$$\sigma = \frac{\lambda \ln J}{J}\mathbf{I} + \frac{\mu}{J}\left(\mathbf{F}\mathbf{F}^T - \mathbf{I}\right) \tag{46}$$

where

$\mathbf{u}$	displacement,
$\mathbf{X}$	position in the reference configuration,
$\mathbf{F} = \mathbf{I} + \frac{\partial \mathbf{u}}{\partial \mathbf{X}}$	deformation gradient,
$J = \|\mathbf{F}\|$	Jacobian,
μ	shear modulus, and
λ	Lamé constant.

The acceleration **b** due to body forces is used as the MMS source term. The source term is "manufactured" in such a fashion that the equations of motion are satisfied for the particular input fields. We select as an ansatz the displacement field, such as a sine function, and then apply a special body force throughout the object that causes the displacement to occur.

5.2 One-Dimensional Periodic Bar

To understand the effect of smoothing length on errors within MPM, we start by simulating a one-dimensional periodic bar on the domain $[0,1]$. The problem we are considering has an assumed analytical displacement and resultant deformation gradient of:

$$u(X,t) = A\sin(2\pi X)\cos(C\pi t), \tag{47}$$

$$F(X,t) = 1 + 2A\pi\cos(2\pi X)\cos(C\pi t), \tag{48}$$

where X is the material position in the reference configuration, A is the maximum deformation percentage, and $C = \sqrt{E/\rho_0}$ is the wave speed. The bar is subjected to a body force of

$$b(X,t) = C^2\pi^2 u(X,t)(2F(X,t)^{-2} + 1). \tag{49}$$

The functions u and F are included in Eq. 49 only to simplify notation. The constitutive model is drawn from Eq. 46 in 1-D with zero Poisson's ratio:

$$\sigma = \frac{E}{2}\left(F - \frac{1}{F}\right). \tag{50}$$

This constitutive model, when combined with the body force given by Eq. 49 will lead to the analytical displacement solution in Eq. 47.

While this one-dimensional bar has a periodic solution, the manufactured solution was chosen such that the velocity and displacements are both zero on the boundaries of our simulation domain $[0,1]$. This allows us to test our simulation with both Dirichlet and periodic boundary treatments of the same problem.

5.3 Axis-Aligned Displacement in a Unit Cube

Displacement in a unit cube is prescribed with normal components such that the corners and edges of GIMP particles are coincident and collinear. This choice allows direct demonstration that GIMP can achieve the same spatial accuracy characteristics in multiple dimensions that have been shown in a single dimension. It is not, however, representative of general material deformations usually found in most realistic engineering scenarios.

The displacement field is chosen to be:

$$\mathbf{u} = \begin{pmatrix} A\sin(2\pi X)\sin(C\pi t) \\ A\sin(2\pi Y)\sin(\frac{2}{3}\pi + C\pi t) \\ A\sin(2\pi Z)\sin(\frac{4}{3}\pi + C\pi t) \end{pmatrix} \tag{51}$$

where X, Y, and Z are the scalar components of position in the reference configuration, t is time, A is the maximum amplitude of displacement, and $C = \sqrt{E/\rho_0}$ is the wave speed, where E is Young's modulus. The factors of two are chosen so that a periodic boundary condition can be used if desired.

The deformation gradient tensor is found by taking derivatives with respect to position, but for the axis-aligned problem only the diagonal terms are non-zero. Therefore:

$$\begin{aligned} \mathbf{F}_{XX} &= 1 + 2A\pi\cos(2\pi X)\sin(C\pi t) \\ \mathbf{F}_{YY} &= 1 + 2A\pi\cos(2\pi Y)\sin(\tfrac{2}{3}\pi + C\pi t) \\ \mathbf{F}_{ZZ} &= 1 + 2A\pi\cos(2\pi Z)\sin(\tfrac{4}{3}\pi + C\pi t) \end{aligned} \tag{52}$$

Acceleration is found by twice differentiating displacement Eq. 51 in time. Then substituting stress $\mathbf{P}$ into Eq. 43 and solving for the body force $\mathbf{b}$ (used as the MMS source term) it is found that:

$$\mathbf{b} = \pi^2 \begin{pmatrix} u_X\left(\frac{4\mu}{\rho_0} - C^2 - 4\frac{\lambda(K-1)-\mu}{\rho_0 F_{XX}^2}\right) \\ u_Y\left(\frac{4\mu}{\rho_0} - C^2 - 4\frac{\lambda(K-1)-\mu}{\rho_0 F_{YY}^2}\right) \\ u_Z\left(\frac{4\mu}{\rho_0} - C^2 - 4\frac{\lambda(K-1)-\mu}{\rho_0 F_{ZZ}^2}\right) \end{pmatrix} \tag{53}$$

where $K = \ln(F_{XX}F_{YY}F_{ZZ})$ and the subscripts on $\mathbf{u}$ and $\mathbf{F}$ indicate individual terms of displacement and deformation gradient equations.

6 Results

6.1 One-D Smoothing Length Experiments

We simulated the one-dimensional periodic bar developed in Section 5.2 on the domain $[0,1]$ to understand the effect of smoothing length on errors within MPM.

The bar is initially discretized with an even sampling of points with initial positions X_p^0. The particle positions are then adjusted to $x_p = X_p^0 + u(X_p^0, 0)$, and deformation gradients set to $F_p = F(X_p^0, 0)$. The simulation is run to a final time T and errors in the particle positions are calculated as

$$\text{Error} = |x_p^T - X_p^0 - u(X_p^0, T)|. \tag{54}$$

This simulation was run with the parameters $A = 0.02$, $E = 10^4$, and $\rho_0 = 1.0$ to a final time $T = 2/C$ (one full period of oscillation) using uGIMP with various numbers of particles-per-cell (PPC) and various smoothing lengths. Fig. 8 shows how errors depend on smoothing length for different numbers of PPC when we run at a relatively large time step corresponding to a CFL number of 0.8. We see that the simulations go unstable when the smoothing length is close to, or less than the initial width of the particles. The 2 PPC simulation goes unstable for $L < h/2$, the 3 PPC simulation goes unstable for $L < h/3$, *etc*.

Fig. 9 shows the effect of smoothing length on the time-step stability restrictions. Fig. 9(a) shows larger smoothing lengths are stable for a wider range of CFL values when the problem is discretized with 4 PPC. While the simulation using a smoothing length of $h/8$ goes unstable at a CFL number of approximately 0.75, the same simulation with a smoothing length of h (equivalent to quadratic B-spline basis functions) is stable up to a CFL number of approximately 1.2. Fig. 9(b) shows similar behaviors for 8 PPC. Furthermore, the time-step stability restrictions do not change significantly between the 4 PPC and 8 PPC simulations, suggesting that the stability is more dependent on the smoothing length than the number of particles per cell.

6.2 One-D Spatial Convergence Results

To investigate the spatial convergence properties of the various MPM methods, we start by simulating the same one-dimensional periodic bar from Section 5.2, now focusing on the behavior of the

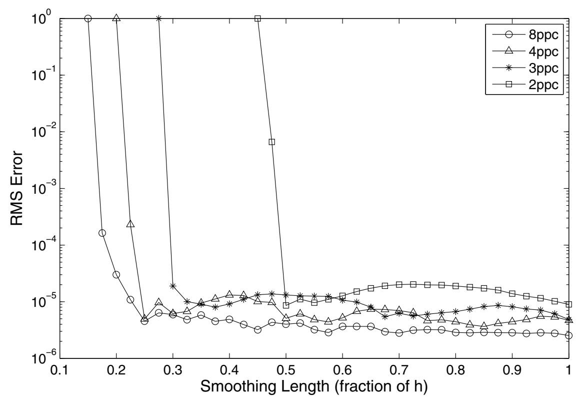
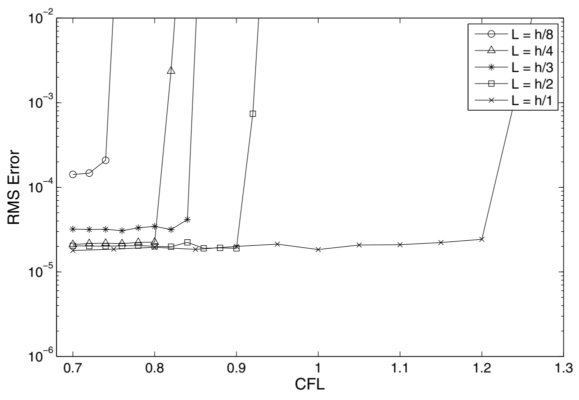


Figure 8: Stability analysis for uGIMP showing errors vs. uGIMP smoothing length. The simulations were run with a fixed time step corresponding to a CFL number of 0.8.

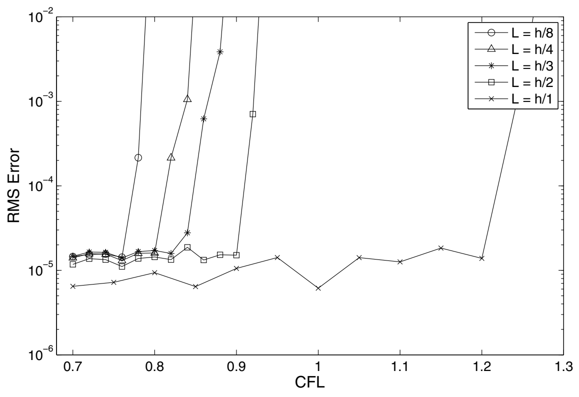
error with respect to grid resolution and how that error may differ using different choices of basis functions and boundary treatments. The simulations were run with the parameters $A = 0.05$ (5% maximum displacement), $E = 10^4$, and $\rho_0 = 1.0$ to a final time $T = 1/C$ (one-half period of oscillation). All simulations were run with a time step of $\Delta t = 4 \cdot 10^{-6}$, corresponding to a CFL number of approximately 0.2 for 512 grid cells–the highest resolution test case.

Fig. 10 shows results from simulations with both the standard MPM piecewise linear basis functions and quadratic B-splines. With an initial discretization of 4 PPC, we see the standard MPM piecewise linear basis functions showing no significant convergence beyond a modest 16 grid cells. Quadratic B-splines show a significant improvement, demonstrating $\mathscr{O}(h^2)$ convergence, with an error plateau occurring past 128 grid cells. The 4 PPC quadratic B-spline simulation was run with both periodic boundary conditions using standard splines (see Fig. 2(b)) and Dirichlet boundary conditions using the modified boundary splines (Fig. 3(a)). The results from the two boundary treatments are nearly identical. As was shown previously, using smoother basis functions greatly improves numerical quadrature errors and stability issues, however nodal integration will always give some quadrature error which can explain the error plateaus starting at 128 grid cells. The last set of simulations in Fig. 10 shows the same quadratic B-spline simulation with Dirichlet boundary conditions, except this time the problem has been discretized using 6 PPC. The extra particles helps lower the quadrature error and lower the error plateau.

To further illustrate errors stemming from boundary treatments, Fig. 11 shows errors for the same problem simulated with B-splines, using three distinct boundary treatments. Similar to Fig. 10, the Dirichlet boundary conditions with modified boundary B-splines and the periodic boundary conditions with standard B-splines show nearly identical results and demonstrate $\mathscr{O}(h^2)$ convergence. Also shown are results for standard



(a) 4 Particles Per Cell



(b) 8 Particles Per Cell

Figure 9: Examination of stability for uGIMP showing errors versus CFL number for various choices of smoothing length.

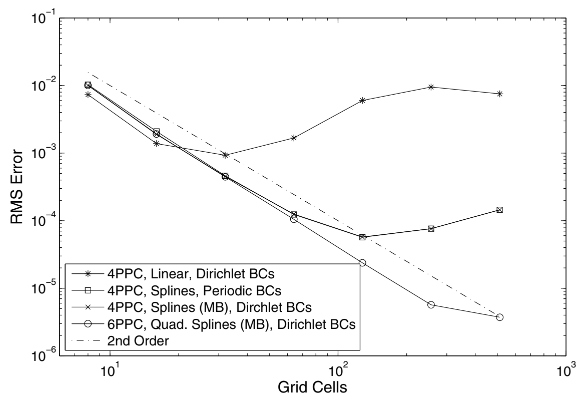


Figure 10: Spatial convergence on a one-dimensional bar manufactured solution problem.

B-splines with Dirichlet boundary conditions. This technique requires handling of the extra or "ghost" boundary nodes as explained in Section 3.4.2. As was discussed previously, this can lead to a geometric error on the grid which is $\mathscr{O}(h)$ and the results show that the error convergence is in fact reduced to $\mathscr{O}(h)$.

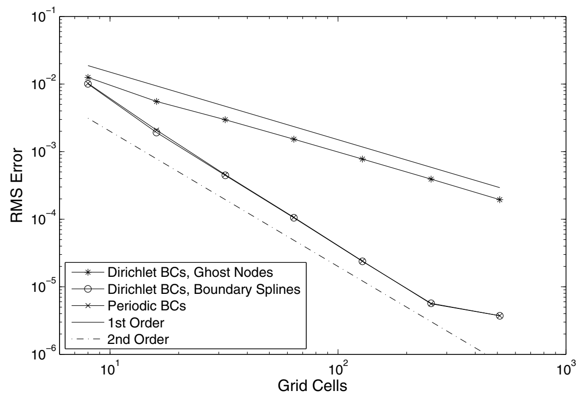


Figure 11: Spatial convergence on a one-dimensional bar manufactured solution using quadratic B-splines with various boundary treatments.

6.3 *Verification with the Method of Manufactured Solutions in Multi-D*

The full three-dimensional axis-aligned problem from Section 5.3 was implemented in the UCF to both demonstrate the validity of the multi-D manufactured solution and show that many of the 1-D convergence results from the previous section are also valid in 3-D. The simulation was run with the parameters $A = 0.05$ (5% maximum displacement), $\rho_0 = 1.0$, $E = 10^4$ and a Poisson's ratio of 0.3. The problem was discretized using 4 PPC in each dimension (64 total particles per cell). Both B-spline basis functions (with symmetric and periodic boundary conditions) and cpGIMP were used in the simulations. The study consisted of grid resolutions from $8\times8\times8$ cells (32768 particles) up to $64\times64\times64$ cells (16.8 million particles).

The results in Fig. 12 clearly show $\mathscr{O}(h^2)$ convergence with cpGIMP for all grid resolutions in the study. Using B-spline basis functions with periodic boundary conditions did nearly as well, with convergence rates trailing off at higher grid resolutions. Similar to the 1-D results, errors when using standard B-splines with the extra, or "ghost" boundary nodes (this time with symmetric boundary conditions) demonstrate the $\mathscr{O}(h)$ convergence we expect due to the geometric errors on the grid.

It is not surprising that cpGIMP outperforms other methods, as this problem is well suited for cpGIMP since particles remain axis-aligned and their voxels area a true partition of the domain. Fig. 13 is a visualization of a representative 2-D slice of the actual solution, showing the axis-aligned particle voxels and how they partition the domain. B-splines when using extra boundary nodes performed as expected, demonstrating the same $\mathscr{O}(h)$ error as the 1-D results. There is an obvious benefit to using periodic boundary conditions over symmetric boundary conditions for this problem since the errors are significantly lower. It is still unclear, however, why the convergence rate for the periodic boundary conditions trails off from the $\mathscr{O}(h^2)$ behavior we would expect from the 1-D results. There are a number of possibilities, including a more complicated quadrature error behavior in multi-D, or the buildup of grid crossing errors (similar to those analyzed by Tran, Kim, and Berzins (2007)), which may be more significant in multi-D since many more grid crossing events occur than in 1-D simulations using similar resolutions.

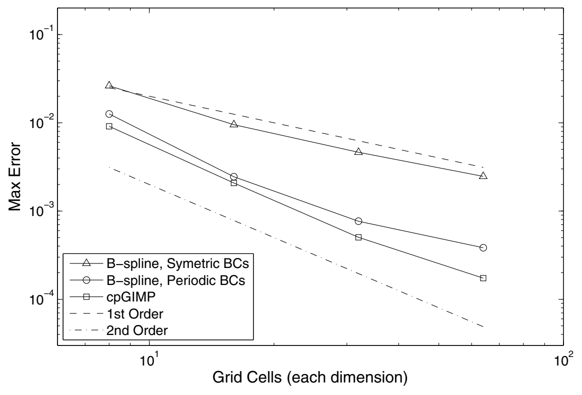


Figure 12: Spatial convergence on the three-dimensional axis-aligned manufactured solution problem.

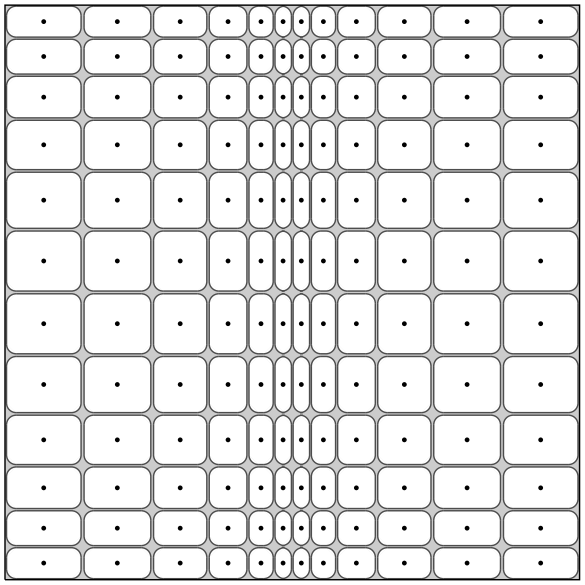


Figure 13: Visualization of a representative 2-D slice of the exact solution (Eq. 51) at time $t = .005$, showing deformed particle positions (black dots) and a conceptualization of the axis aligned particle voxels. The voxels have been rounded and their sizes slightly reduced for visual clarity.

7 Summary

In this paper we have considered some of the many choices one must consider when implementing the Material Point Method. Two of the design choices which have significant impact on error properties of the method are which grid basis functions to use and how to implement boundary conditions. We explored and analyzed the numerical impact of these algorithmic choices.

A number of basis functions were explored, including: standard piecewise-linear basis functions, B-spline basis functions, uniform GIMP (uGIMP), and contiguous particle GIMP (cpGIMP). All these functions were shown to be connected through a similar construction technique–the convolution of piecewise-constant functions of various lengths. Analysis of the uGIMP functions showed an integration, or quadrature error which was second order with respect to particle spacing when the basis function smoothing length is larger than the particle widths. When the smoothing length is smaller than the particle widths, this integration error becomes first order. The effects of this relationship between particle widths and smoothing lengths were demonstrated in simulations where instabilities occurred when the smoothing length was set smaller than the particle widths.

Boundary condition implementation also had an effect on the overall errors in the method. The geometric errors present in the grid representation of the deformed material can result in first order spatial errors when standard kinematic boundary conditions are applied. These geometric errors are exacerbated when smoother, and necessarily wider, basis functions are used, such as uGIMP, or B-splines. We were able to eliminate these first order errors when using periodic boundary treatments. Relaxing the requirement that each grid node correspond to a single basis function led us to a set of modified boundary B-spline basis functions which eliminated the geometric errors for our problem and allowed second order spatial convergence with standard Dirichlet boundary conditions.

Acknowledgement: This work was supported by the U.S. Department of Energy through the Center for the Simulation of Accidental Fires and Explosions (C-SAFE), under grant W-7405-ENG-48. In addition, the authors would like to acknowledge helpful discussions with members of the Utah MPM Group.

References

Atluri, S. N. (2006): Meshless Local Petrov-Galerkin (MLPG) mixed collocation method for elasticity problems. *CMES: Computer Modeling in Engineering & Sciences*, vol. 14, no. 3, pp. 141–152.

Atluri, S. N.; Liu, H. T.; Han, Z. D. (2006): Meshless Local Petrov-Galerkin (MLPG) mixed finite difference method for solid mechanics. *CMES: Computer Modeling in Engineering & Sciences*, vol. 15, no. 1, pp. 1–16.

Atluri, S. N.; Zhu, T. (1998): A new Meshless Local Petrov-Galerkin (MLPG) approach in computational mechanics. *Computational Mechanics*, vol. 22, no. 2, pp. 117–127.

Banerjee, B. (2006): Method of manufactured solutions. www.eng.utah.edu/ banerjee/Notes/MMS.pdf, 2006.

Bardenhagen, S. (2002): Energy conservation error in the material point method for solid mechanics. *Journal of Computational Physics*, vol. 180, pp. 383–403.

Bardenhagen, S.; Kober, E. (2004): The generalized interpolation material point method. *CMES: Computer Modeling in Engineering and Sciences*, vol. 5, pp. 477–495.

Bardenhagen, S. G.; Brydon, A. D.; Guilkey, J. E. (2005): Insight into the physics of foam densification via numerical simulation. *Journal of the Mechanics and Physics of Solids*, vol. 53, no. 3, pp. 597–617.

Belytschko, T.; Liu, W. K.; Moran, B. (2000): *Nonlinear Finite Elements for Continua and Structures*. John Wiley and Sons, LTD.

Belytschko, T.; Lu, Y. Y.; Gu, L. (1994): Element free Galerkin methods. *International Journal for Numerical Methods in Engineering*, vol. 37, no. 2, pp. 229–256.

Brackbill, J. U.; Kothe, D. B.; Ruppel, H. M. (1988): FLIP: a low-dissipation, particle-in-cell method for fluid flow. *Computer Physics Communications*, vol. 48, pp. 25–38.

Brackbill, J. U.; Ruppel, H. M. (1986): FLIP: a method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions. *Journal of Computational Physics*, vol. 65, pp. 314–343.

Grigoryev, Y. N.; Vshivkov, V. A.; Fedoruk, M. P. (2002): *Numerical "Particle-in-Cell" Methods*. VSP.

Guilkey, J. E.; Weiss, J. A. (2003): Implicit time integration for the material point method: Quantitative and algorithmic comparisons with the finite element method. *International Journal for Numerical Methods in Engineering*, vol. 57, no. 9, pp. 1323–1338.

Han, Z. D.; Liu, H. T.; Rajendran, A. M.; Atluri, S. N. (2006): The applications of Meshless Local Petrov-Galerkin (MLPG) approaches in high-speed impact, penetration and perforation problems. *CMES: Computer Modeling in Engineering & Sciences*, vol. 14, no. 2, pp. 119–128.

Han, Z. D.; Rajendran, A. M.; Atluri, S. N. (2005): Meshless Local Petrov-Galerkin (MLPG) approaches for solving nonlinear problems with large deformations and rotations. *CMES: Computer Modeling in Engineering & Sciences*, vol. 10, no. 1, pp. 1–12.

Kashiwa, B. A.; Lewis, M. L.; Wilson, T. (1996): Fluid-structure interaction modeling. Technical Report LA-13111-PR, Los Alamos National Laboratory, Los Alamos, 1996.

Knupp, P.; Salari, K. (2003): *Verification of Computer Codes in Computational Science and Engineering*. Chapman and Hall/CRC.

Li, S.; Liu, W. K. (2004): *Meshfree Particle Methods*. Springer.

Love, E.; Sulsky, D. L. (2006): An energy-consistent material-point method for dynamic finite deformation plasticity. *International Journal for Numerical Methods in Engineering*, vol. 65, no. 10, pp. 1608–1638.

Love, E.; Sulsky, D. L. (2006): An unconditionally stable, energy-momentum consistent implementation of the material-point method. *Computer Methods in Applied Mechanics and Engineering*, vol. 195, no. 33-36, pp. 3903–3925.

Ma, J.; Lu, H.; Komanduri, R. (2006): Structured mesh refinement in generalized interpolation material point method (GIMP) for simulation of dynamic problems. *CMES: Computer Modeling in Engineering and Sciences*, vol. 12, pp. 213–227.

Nairn, J. A. (2006): Numerical simulations of transverse compression and densification in wood. *Wood and Fiber Science*, vol. 38, no. 4, pp. 576–591.

Parker, S.; Guilkey, J.; Harman, T. (2006): A component-based parallel infrastructure for the simulation of fluid-structure interaction. *Engineering With Computers*, vol. 22, no. 3, pp. 277–292.

Schwer, L. (2002): Method of manufactured solutions: Demonstrations. www.usacm.org/vnvcsm/PDF_Documents/MMS-Demo-03Sep02.pdf, 2002.

Steffen, M.; Kirby, R. M.; Berzins, M. (2008): Analysis and reduction of quadrature errors in the material point method (MPM). *International Journal for Numerical Methods in Engineering*. DOI: 10.1002/nme.2360.

Sulsky, D.; Chen, A.; Schreyer, H. L. (1994): A particle method for history dependent materials. *Comput. Methods Appl. Mech. Engrg.*, vol. 118, pp. 179–196.

Sulsky, D.; Kaul, A. (2004): Implicit dynamics in the material-point method. *Computer Methods in Applied Mechanics and Engineering*, vol. 193, no. 12-14, pp. 1137–1170.

Sulsky, D.; Schreyer, H.; Peterson, K.; Kwok, R.; Coon, M. (2007): Using the material point method to model sea ice dynamics. *Journal of Geophysical Research*, vol. 112.

Sulsky, D.; Zhou, S.; Schreyer, H. L. (1995): Application of a particle-in-cell method to solid mechanics. *Computer Physics Communications*, vol. 87, pp. 236–252.

Tran, L. T.; Kim, J.; Berzins, M. (2007): Solving Time-Dependent PDEs using the Material Point Method, A Case Study from Gas Dynamics. Technical Report UUSCI-2007-010, SCI Institute, University of Utah, 2007.

Wallstedt, P. C. (2008): *On the Order of Accuracy of the Generalized Interpolation Material Point Method*. PhD thesis, University of Utah, 2008.

Wallstedt, P. C.; Guilkey, J. E. (2007): Improved velocity projection for the material point method. *CMES: Computer Modeling in Engineering and Sciences*, vol. 19, pp. 223–232.

Wallstedt, P. C.; Guilkey, J. E. (2008): An evaluation of explicit time integration schemes for use with the generalized interpolation material point method. *To Appear In Journal of Computational Physics*.