Improved Velocity Projection for the Material Point Method

P. C. Wallstedt[1] and J. E. Guilkey[1]

Abstract The standard velocity projection scheme for the Material Point Method (MPM) and a typical form of the GIMP Method are examined. It is demonstrated that the fidelity of information transfer from a particle representation to the computational grid is strongly dependent on particle density and location. In addition, use of non-uniform grids and even non-uniform particle sizes are shown to introduce error. An enhancement to the projection operation is developed which makes use of already available velocity gradient information. This enhancement facilitates exact projection of linear functions and reduces the dependence of projection accuracy on particle location and density for non-linear functions. The efficacy of this formulation for reducing error is demonstrated in solid mechanics simulations in one and two dimensions.

keywords: Material Point Method, MPM, PIC, GIMP, meshless methods.

1 Introduction

Continual increases in computational power have enabled simulations of increasingly complex physical systems. Meshless and quasi-meshless methods often provide acceptable solutions for problems where the venerable Finite Element Method (FEM) fails due to pathologies associated with the mesh. The Material Point Method (MPM) [Sulsky, Chen and Schreyer (1994); Sulsky, Zhou and Schreyer (1995)] is one such quasi-meshless method. MPM may be considered a Particle-In-Cell (PIC) method that has been extended for use in solid mechanics.

MPM has advantages in modeling geometrically complex domains such as those found in biological systems [Guilkey, Hoying, Weiss (2005)], as well as scenarios involving large deformations [Brydon, Bardenhagen, Miller, Seidler (2005)], contact [Bardenhagen, Guilkey, Roessig, Brackbill, Witzel, Foster (2001)], fracture [Guo and Nairn (2004)], molecular dynamics [Ma, Lu, Wang, Roy, Hornung, Wissink, Komanduri (2005); Ma, Liu, Lu, Komanduri (2006a); Ma, Liu, Komanduri (2006b); Ma, Lu, Wang, Roy, Hornung, Wissink, Komanduri (2006c)], and delamination [Shen, L.; Chen, Z.; (2005)]. MPM is relatively easy to parallelize because of its use of a Cartesian background grid, or "scratchpad", which eliminates neighbor searches and provides for straightforward domain decomposition. In addition, it has been integrated with a compressible CFD solver to model fluid-structure interactions in the simulations of exploding containers [Parker, Guilkey, Harman (2005)].

Several researchers have strengthened and extended the mathematical underpinnings of MPM in areas such as conservation [Bardenhagen (2002), Love and Sulsky (2006)] and completeness [Bardenhagen (2006)]. Others have made algorithmic improvements including implicit time integration [Guilkey and Weiss (2003); Burgess, Sulsky, and Brackbill (1992); Love and Sulsky (2006)] and the aforementioned contributions to contact and fracture.

Bardenhagen and Kober (2004) revisited the formulation of MPM by recognizing that the transfer of information between particles and grid is best represented as the inner product of grid and particle functions. A finite sized particle was developed whose volume can be split among the cells that it overlaps, thus providing C^1 differentiability, which in turn reduces the deleterious effects of cell face crossing. Furthermore, their work provides a framework for future accuracy improvements and error estimation. This framework is generically known as the Generalized Interpolation Material Point Method, or GIMP. Some of the advantages of GIMP over traditional MPM, beyond those previously reported, will be demonstrated here.

Along with particular advantages, MPM has some notable limitations. One of these regards the transfer of data from particles to the computational scratchpad. Here we demonstrate that the degree to which information is preserved in this operation is strongly dependent on particle density (number of particles per computational cell) and location. In what follows, this is first demonstrated via simple examples, after which a systematic means for measuring this error is introduced. Current observations are placed in the context of previous work in this area, after which a solution to this problem is proposed and

[1] University of Utah, U.S.A.

demonstrated. Other sources of error, such as those due to time integration and particle cell-crossing, are not considered here.

2 Motivation

A full description of the MPM algorithm can be found in the references cited in the introduction. In the interest of brevity, we forego such a description here, and concentrate on new findings.

The Material Point Method (MPM) features a fixed Eulerian grid within which particles are free to move. The 100 particles in each panel of Figure 1 may start in idealized locations relative to the grid (left), or they could be less ideally located as depicted in the radial pattern (right). The radial particle pattern is non-random and includes a nearly even spacing between particles. Yet the fortunate effects of symmetry and central differencing that apply to the ideal pattern will not apply to the radial pattern. Standard MPM codes produce much less accurate answers for the radial pattern than for the ideal. And of course, even if the initial particle distribution is ideal, as simulations evolve, the particles will generally move into a less favorable configuration.

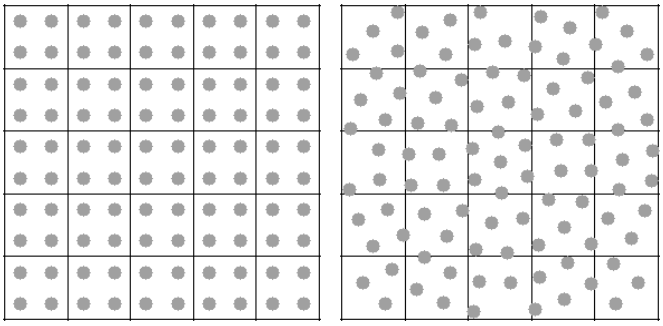

Figure 1: Ideal Versus General Particle Positions

Spatial error of individual MPM steps can be studied independently by measuring the accuracy with which each of the particle-grid interaction steps performs. Grid-to-particle operators behave as expected, with the properties of the interpolation function that is used. However, the transfer of information from particles to grid is a distinct operation with less obvious characteristics.

Field quantities on the MPM grid are found through a mass-weighted least squares procedure. For example, velocity projection is given by:

$$v_i = \frac{\sum_p (v_p m_p S_{ip})}{\sum_p (m_p S_{ip})} \qquad (2.1)$$

where v_i is the nodal velocity, v_p is the particle velocity, m_p is the particle mass and S_{ip} is the trial (or "shape") function.

A brief aside is in order at this point. Throughout this paper we refer to "MPM" and "GIMP", even though MPM can be considered a specific version of GIMP methods. GIMP refers to a class of methods for which one can make a choice of the grid trial function that is used, as well as the particle characteristic function. Throughout this work, the bilinear "tent" function is used as the grid trial function. When the particle characteristic function is chosen to be a Dirac delta function, MPM is the resulting method. When the particle characteristic function is the constant "top hat" function, the resulting form is called "contiguous particle GIMP" [Bardenhagen and Kober, (2004)]. This is the form used here, and it will be generically referred to as GIMP. This form of GIMP can be conveniently expressed in terms of a change in grid trial function. The MPM and (effective) GIMP trial functions are shown in Figure 2. The effective GIMP trial function is shown assuming a particle width equal to half that of a computational cell.

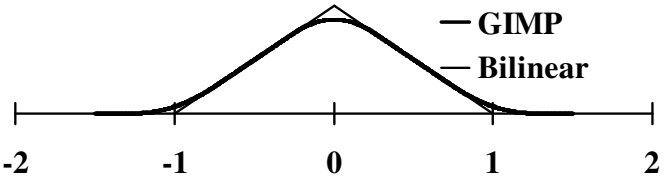

Figure 2: Trial Functions for MPM and GIMP

Although MPM uses bilinear trial functions and GIMP uses a more complex C^1 trial function, neither system is able to provide an exact projection of a linear velocity field for arbitrary particle positions. This is demonstrated by examples in Figure 3 where four arrangements of particles are shown in which the projection of the linear velocity function $v(x) = mx + b$ is performed, with the error incurred for each. The errors reported in each panel are for bilinear functions, but are non-zero for GIMP as well. The dashed line indicates the correct value for velocity, while the diamond symbols indicate the values of the nodal velocities computed using Eq. 2.1.

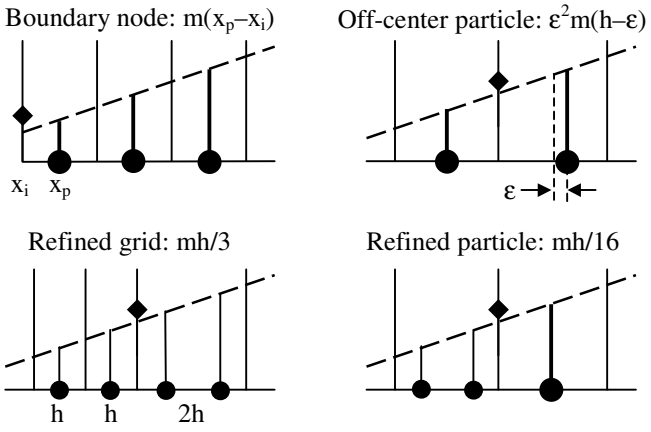

Figure 3: Velocity Projection Error for Degenerate Cases

Inspection of the upper right panel reveals that the error is zero for perfectly centered particles. In all of these cases, cancellation of error occurs due to symmetry. This symmetry is lost in each of the cases depicted above. This has negative implications for solutions involving non-uniform meshes, non-uniform particle sizes, and arbitrary particle positions.

Mathematical models that are well-regarded in the scientific community include some kind of specific and rigorous estimation of error. For PIC methods, to which MPM is closely related, the following relation for the upper bound of error of the projection of charge density, ρ, was developed by V. A. Vshivkov:

$$\delta_k \le \left(\frac{3\rho_{av}^2}{\rho_{min}} + h \frac{\rho_{av}^2 \rho_{max}}{\rho_{min}^3} \left| \frac{\partial \rho}{\partial x} \right|_{max} \right) \frac{1}{PPC^2} + \frac{h^2}{12} \left| \frac{\partial^2 \rho}{\partial x^2} \right|_{max} \tag{2.2}$$

where h is the grid spacing and PPC is the ratio of the global number of particles to the global number of cells. It is found in subsequent sections that error measurements for MPM fit within the form of this equation – that is to say, a sum of two terms which can act independently and which must both be driven toward zero.

3 Velocity Projection Error

Here, the error due to the projection of velocity as given by Eq. 2.1 is studied and an approach by which this error may be reduced is proposed. While Eq 2.2 provides guidance to this investigation, there is much about the structure of the error that it does not reveal. To learn more, a number of numerical experiments are performed on carefully chosen special cases and guidelines are inferred based on analysis of those results.

The error due to velocity projection can be measured by prescribing velocity on the particles according to a linear, quadratic, or other function. In this context, the error is defined as the difference between velocity on each grid node and velocity from the known function evaluated at the nodal position. For a linear function $v(x) = 1 + x$ the velocity on each particle is assigned as $v_p(x) = v(x_p)$, the mass of each particle is the same, and the velocity at each grid node is found by equation 2.1. Then the error is defined as:

$$\delta^v = \max\left(\frac{v_i - v(x_i)}{v(x_i)} \right) \tag{3.1}$$

Arbitrary particle position is simulated in 1D by a so-called "squeeze test" wherein successively increasing integer numbers of particles are packed into the same grid and the error is measured for each arrangement (Figure 4). A global particles-per-cell ratio is thus defined for each set of grid and particles:

$$PPC = \frac{\text{total number of particles}}{\text{total number of cells}} \tag{3.2}$$

Figure 4: Global Particles-Per-Cell Ratio

Using a 1D grid with 100 cells, and starting at PPC=.5, one particle at a time is added, and the error computed for each arrangement, up to PPC=10 (1000 particles in all). This series of tests is carried out for both the MPM and the GIMP trial functions. The error from this series of tests is shown in Figure 5.

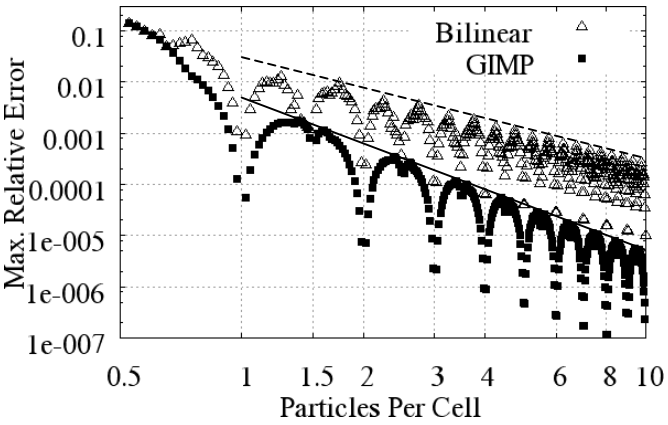

Figure 5: Velocity Projection Error - Linear

Significant error is found even for the projection of a simple linear function $v(x) = 1 + x$, except at integer (and half integer) values of PPC. While the error does not decrease monotonically with increasing PPC, the values of the local maxima display a downward trend. The GIMP trial functions perform better in general and the values of the local maxima converge as PPC^{-3} (solid line) while the bilinear trial functions converge as PPC^{-2} (dashed line).

When a quadratic velocity function, $v(x)=(1+x)^2$, is prescribed on the particles, the error shows somewhat different behavior. Local error maxima are observed in Figure 6 as well, but here they descend to a plateau. No arrangements of particles report zero error, which is no surprise considering the use of bilinear trial functions. But it is initially curious that further increase in PPC produces no decrease in error. The GIMP trial functions descend to the plateau more quickly and more nearly monotonically, but the use of GIMP does not reduce the level of the plateau.

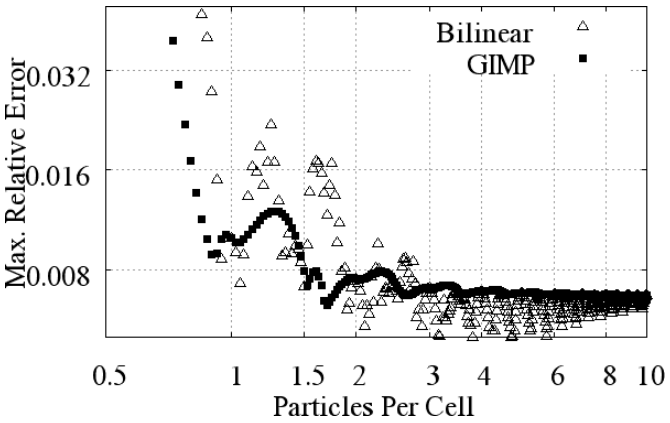

Figure 6: Velocity Projection Error - Quadratic

The existence of the plateau in Figure 6 can be attributed to the fact that, regardless of how many particles are used, ultimately, all of the information is transferred to the two nodes (in 1D) that bound each cell. This is akin to trapezoidal integration, which is known to only be capable of exactly integrating a linear function.

Relating observed error in MPM to the analytical expression for PIC given by Equation (2.2), it is clear that the plateau in error remains after reducing the left term to zero (by making PPC large) without altering the right term. The right term, and the associated plateau, could be reduced independently by fixing PPC and reducing grid cell size, indicating that both grid resolution and particle density are important to accuracy. An example of this is shown in Section 5.

4 Gradient Enhancement

A method has been developed that uses the velocity gradient information that is already available (for the calculation of rate of deformation) within the MPM algorithm to improve the accuracy of the projection, including the exact projection of linear functions. Within this method, each particle acts as though it is the only particle within a cell and assumes that it must exactly project a linear function of velocity. Let each particle carry velocity v_p and velocity gradient $\partial v_p/\partial x$. Particle p uses this information to suggest an extrapolated nodal velocity v^e for each node to which it would ordinarily contribute. Conceptually, these suggestions form a table in which the extrapolation of velocity on particle p to node i is referred to as v^e_{ip}.

$$v^e_{ip} = v_p - \frac{\partial v_p}{\partial x}(x_p - x_i) \tag{4.1}$$

Each row of the extrapolation table contains 2 entries for the 1D tent functions, 3 entries for the 1D GIMP functions, 4 entries for the 2D tent functions or 9 entries for the 2D GIMP functions, and so on. The scheme continues to conserve momentum if $x_p = \Sigma x_i S_{ip}$, i.e., for isoparametric trial functions. The table of extrapolated velocities is not stored in practice; rather each entry is computed on-the-fly. The original velocity projection (Eq. 2.1) is modified to use the extrapolated velocities:

$$v_i = \frac{\sum_p (v^e_{ip} m_p S_{ip})}{\sum_p (m_p S_{ip})} \tag{4.2}$$

A visual representation of the change that occurs due to use of the extrapolated v^e_{ip} is given in Figure 7. The original v_p values are extended straight to the nodes (solid lines) and weighted according to the particles' locations. The extrapolated v^e_{ip} extend to the nodes differently (dashed lines) and are also weighted by the particles' locations.

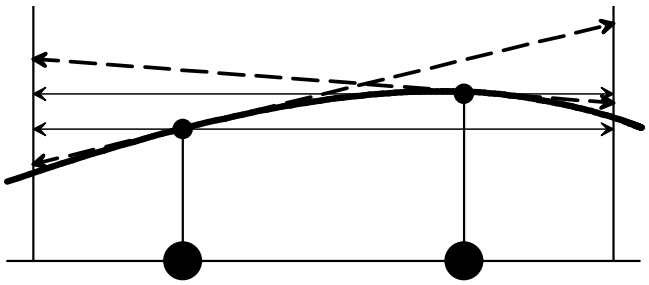

Figure 7: Extrapolated velocities

Multi-dimensional forms of gradient-enhanced particles can be developed in an analogous manner. The 2D equations are written here as:

$$u^e_{ip} = u_p - \frac{\partial u_p}{\partial x}(x_p - x_i) - \frac{\partial u_p}{\partial y}(y_p - y_i) \tag{4.3a}$$

$$v^e_{ip} = v_p - \frac{\partial v_p}{\partial x}(x_p - x_i) - \frac{\partial v_p}{\partial y}(y_p - y_i) \tag{4.3b}$$

where u and v are the x and y components of velocity, respectively. These provide the extrapolated nodal velocities based on information from each particle. The extrapolated velocity components can then be used in Eq. 4.2 to compute nodal velocities.

5 Results

5.1 1D Velocity Projection

Projection error is measured again using the gradient enhanced bilinear and GIMP trial functions and it is found that a linear function is projected exactly for any particle distribution; see Figure 8.

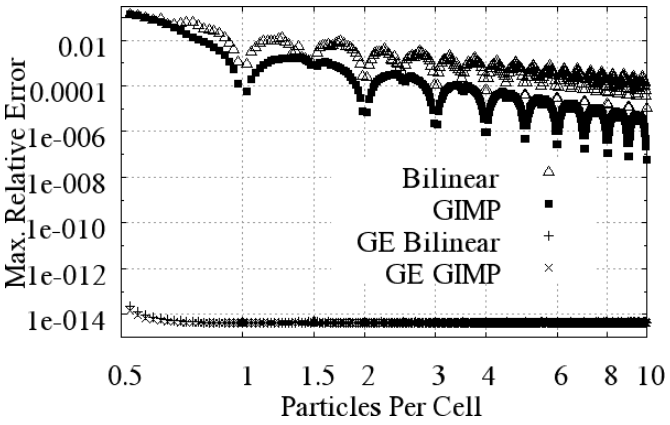

Figure 8: Gradient Enhanced Projection for Linear Function

Improved behavior is seen for higher-order functions as well where the grid refinement plateau is reached for very low PPC ratios. The data from Figure 6 are re-plotted along with errors measured when using the gradient enhanced particles in Figure 9. The low PPC value at which the plateau is reached has desirable implications for reducing computational effort required to reach a particular level of accuracy.

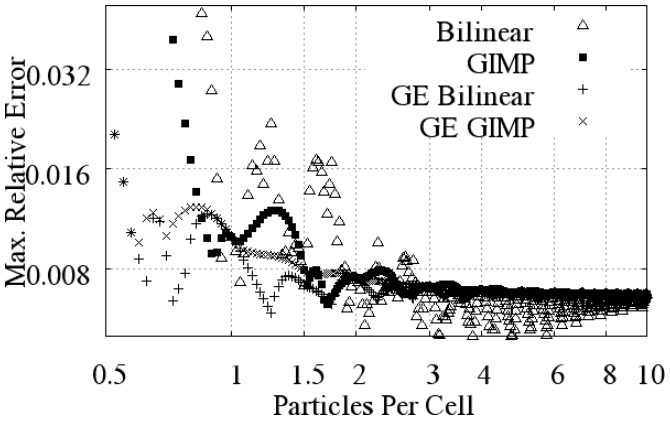

Figure 9: Gradient Enhanced Projection for Quadratic Function

To demonstrate the dependence of error on mesh refinement, the error is shown in Figure 10 for three meshes of different resolutions for Gradient-Enhanced GIMP.

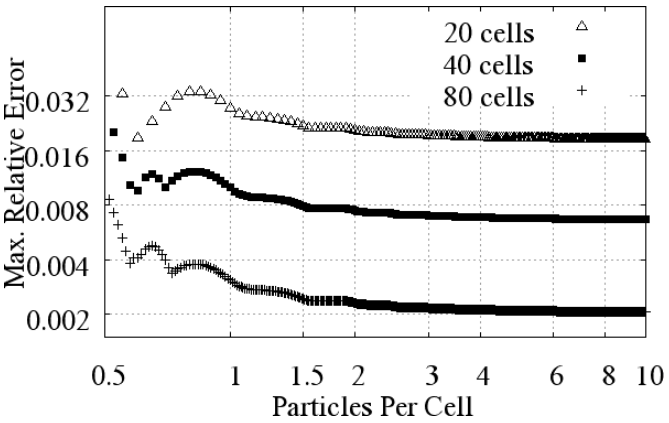

Figure 10: Dependence of Projection Error on Mesh Size using Gradient Enhancement

5.2 1D MPM Simulations

The benefit of using gradient enhancement is further explored with complete solutions using a 1D MPM code initialized with the non-ideal particle positions suggested by the "squeeze" test of Section 3.

A linear elastic bar moving with uniform initial velocity v_o has one end suddenly brought to rest at time zero. The MPM solution is compared to an exact solution of the wave equation

$$u_{tt} = (E/\rho)u_{xx} \tag{5.1}$$

where $u_t(x,0) = v_0$ and $u(0,t) = 0$. Density = area = bar length L = 1. CFL = kc/h = 0.1 where k is the time step, c is the wave speed $\sqrt{E/\rho}$, h is the cell size 1/40, and Young's Modulus E = 10000. The period and amplitude of the tip displacement are 4L/c and v_0L/c, respectively, where v_0 = 1. The exact solution for the displacement forms a series of clipped "saw-tooth" patterns. Figure 11 shows the exact and numerical solutions at four locations along the bar.

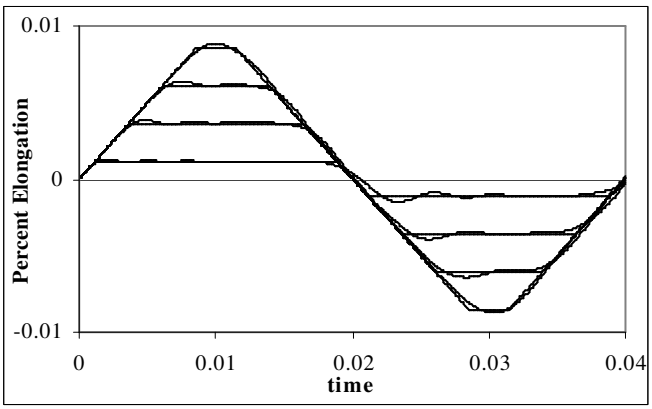

Figure 11: Bar Displacement History

The maximum relative error is retained for every particle at every time step and is reported as the error for the solution:

$$\delta^u = \max\left(\frac{u_p^k - u(X_p^k)}{v_0 L / c}\right) \quad (5.2)$$

The simulation is run for a range of global PPC values and the resulting pattern of error is plotted in Figure 12. The initial velocity is chosen such that the maximum deformation of the bar is one percent of its initial length, resulting in a solution which is beyond infinitesimal yet which still largely agrees with the linear solution. This provokes significant cell crossing of particles causing a few of the bilinear solutions to "crash" but GIMP handles the cell crossing without problem.

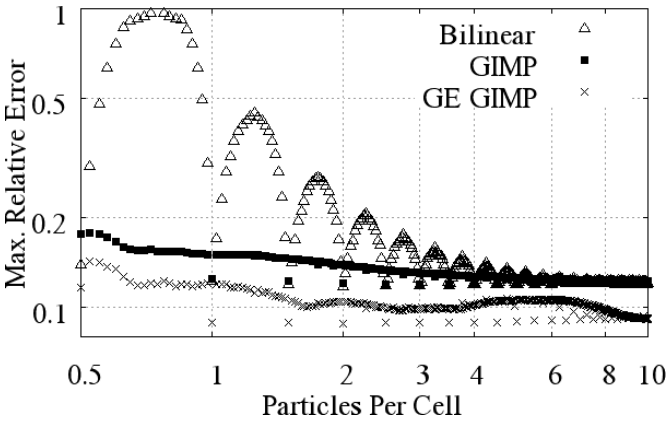

Figure 12: 1D Solution Accuracy with Increasing PPC

The power of the GIMP trial functions is made clear with this plot as they provide much more accuracy in the small PPC region and better reliability and consistency regardless of PPC.

Good accuracy for low PPC ratios is important to keep MPM competitive in terms of computational expense. Generally speaking, the computational cost is roughly proportional to the number of particles used, while the overall accuracy depends largely on grid resolution. Thus, the typical user of MPM cannot afford to use 100 or even 10 particles per cell in each dimension.

Figure 12 omits solutions that use gradient enhancement with bilinear trial functions. This is because they are pathologically unstable. In reality the gradient enhancement technique works only if good quality gradients are provided. Poor estimates of the gradient may cause the solution to be worse with gradient enhancement than without it. GIMP is superior to bilinear trial functions for providing reliably good gradients.

Gradients of velocity are already present within the MPM algorithm; they are calculated in order to update strain and volume. Instead of being discarded at the end of a cycle, the gradients can be saved and used as input for the next cycle at a very modest computational cost. In this case gradient enhancement is able to reliably reduce error by about forty percent when used within the GIMP trial functions.

Spatial convergence for the "saw-tooth" problem is only first order due to the sharp edges in the displacement and velocity fields. Second order convergence can be obtained by solving the wave equation (4.1) for a smooth initial displacement $u(x) = A\sin(\pi x)$ for $0 < x < 1$ and boundary conditions $u(0,t)=0$, $u(1,t) = 0$ which has the known solution $u(x,t) = A\sin(\pi x)\cos(c\pi t)$ where $A = 0.01$. The initial PPC ratio is 2, CFL = 0.2, and the time duration is a quarter period. Figure 13 depicts the spatial convergence with and without the use of gradient enhanced particles.

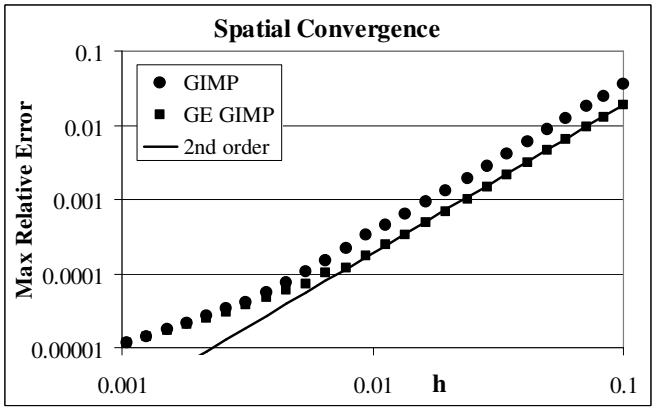

Figure 13: 1D Improvement due to Gradient Enhancement

5.3 2D MPM Simulations

Werner Soedel (2004) describes an exact solution for linear elastic in-plane plate vibration that is used to measure spatial convergence in 2D. For a full description of the problem and its exact solution the reader is urged to consult Soedel but briefly put, particles in a square domain are initially at positions shown in Figure 14 (left) and are released at time zero, after which they oscillate between the initial positions and those shown in the panel on the right. The displacements in the figure are exaggerated; the initial maximum displacement used in calculations was 0.004. Plate width, length, and height = 1; Young's Modulus = 1000; density = 1; Poisson's ratio = 0.3; PPC = 4. Velocity boundary conditions are set on the sides of the domain such that particles can move normal to the side, but not tangentially.

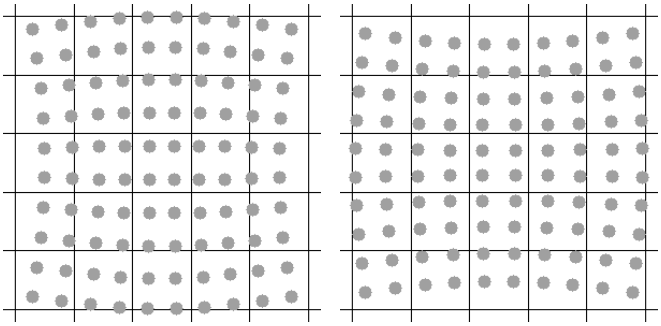

Figure 14: In-plane Vibration - Initial and Final Solutions

This problem is used to measure convergence in a 2D MPM code. Particles are initially positioned in an ideal Cartesian manner and the mesh is refined, keeping the PPC constant. A single maximum relative error is found for a particular solution and plotted versus the mesh size h in Figure 15.

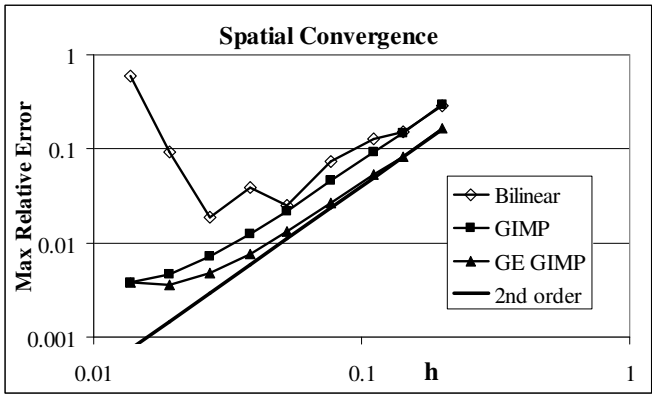

Figure 15: 2D Improvement due to Gradient Enhancement

Convergence for the bilinear case is unreliable, particular once particle cell crossings become frequent. GIMP maintains convergence to a higher level of resolution. The use of gradient enhanced particles maintains the same convergence characteristics, but at a lower overall error. Two reasons for the tail off in convergence are suspected. First, the 0.4% initial displacement simulation may depart from the conditions under which the infinitesimal exact solution is valid. This is also true for the example in Figure 13. Secondly, the vast improvement of GIMP over bilinear trial functions (see Figure 12) is slightly degraded in multiple dimensions. GIMP relies on exact tiling of space, which is significantly harder (bordering on intractable) for multiple dimensional simulations. An approximation that might improve this result by tracking particle corners is described by Ma, Lu, and Komanduri (2006), but was not implemented here. Despite this limitation the GIMP trial functions continue to perform better in multiple dimensions than the bilinear functions.

6 Conclusions

A systematic investigation into the accuracy properties of the projection of particle field data to a computational grid has been carried out for both bilinear and (effective) GIMP trial functions. It was demonstrated that the accuracy of this process is strongly dependent on particle density and location, as well as the resolution of the grid. Projection error in MPM shows characteristics similar to PIC. The error trends observed indicate a sum of two error terms; one dependent on the number of particles per cell, and one dependent on cell size. Improvement in accuracy requires driving both terms toward zero.

Velocity projection can be improved via re-use of existing gradient information at minimal cost. However, good gradients are required as input; indeed, the gradient enhancement can destabilize a solution if the input gradients are poor. Generally, the gradients provided by bilinear trial functions are not of sufficient quality to be of use. Fortunately, the use of GIMP trial functions provides gradients that are of high enough quality to result in a roughly 40% improvement in solution accuracy for each of the 3 cases investigated here.

The GIMP method is commonly understood to be a solution to the problem of particle cell-crossing. Here it was demonstrated that it is also considerably more accurate for field projection, particularly when used with the additional gradient information. GIMP also improved the accuracy and stability characteristics of full mechanics simulations as well.

Acknowledgements

The authors gratefully acknowledge helpful discussions with members of the MPM working group at the University of Utah, in particular Mike Kirby, Martin Berzins and Mike Steffen. Financial support for this work comes from: NSF-ITR Grant # CTS0218574 and DOE W-7405-ENG-48 (C-SAFE)

References

Bardenhagen, S. G.; Kober, E. M. (2004): The Generalized Interpolation Material Point Method. *Computer Modeling in Engineering and Science*., vol. 5, pp. 477-496.

Bardenhagen, S. G.; Brackbill, J. U.; Sulsky, D. (2000): The Material Point Method for Granular Materials. *Computer Methods in Applied Mechanics and Engineering*, vol. 187, pp. 529–541.

Bardenhagen, S. G.; Guilkey, J. E.; Roessig, K. M.; Brackbill, J. U.; Witzel, W. M.; Foster, J. C. (2001): An Improved Contact Algorithm for the Material Point Method and Application to Stress Propagation in Granular Material. *Computer Modeling in Engineering and Science*, vol. 2, pp. 509–522.

Bardenhagen, S. G. (2002): Energy Conservation Error in the Material Point Method for Solid Mechanics. *Journal of Computational Physics* vol. 180, pp. 383–403.

Bardenhagen, S. G. (2006): Completeness of the Material Point Method. *Second Annual MPM Workshop*, Tulsa, Oklahoma, March 6-7, 2006.

Brydon, A. D.; Bardenhagen, S. G.; Miller, E. A.; Seidler, G. T. (2005): Simulation of the densification of real open-celled foam microstructures. *Journal of the Mechanics and Physics of Solids*, vol. 53, pp. 2638-2660.

Burgess, D.; Sulsky, D.; Brackbill, J. U. (1992); Mass matrix formulation of the FLIP particle-in-cell method, *Journal of Computational Physics* vol. 103, pp. 1–15.

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, pp. 1323-1338.

Guilkey, J. E.; Hoying, J. B.; Weiss, J. A. (2006): Computational modeling of multicellular constructs with the material point method, *Journal of Biomechanics,* vol. 39, pp. 2074-2086.

Guo, Y.; Nairn, J. A. (2004): Calculation of J-Integral and Stress Intensity Factors using the Material Point Method. *Computer Modeling in Engineering and Science*, vol. 6, pp. 295-308.

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, pp. 3903–3925.

Ma, J.; Liu, Y.; Lu, H; Komanduri, R. (2006): Multiscale Simulation of Nanoindentation Using the Generalized Interpolation Material Point (GIMP) Method, Dislocation Dynamics (DD) and Molecular Dynamics (MD). *Computer Modeling in Engineering and Science*, vol. 16, pp. 41-56.

Ma, J.; Lu, H.; Wang, B.; Hornung, R; Wissink, A,; Komanduri, R. (2006): Multiscale Simulation Using Generalized Interpolation Material Point (GIMP) Method and Molecular Dynamics (MD). *Computer Modeling in Engineering and Science*, vol. 14, pp. 101-118.

Ma, J.; Lu, H.; Komanduri, R. (2006): Structured Mesh Refinement in Generalized Interpolation Material Point (GIMP) Method for Simulation of Dynamic Problems. *Computer Modeling in Engineering and Science*, vol. 12, pp. 213-227.

Ma, J.; Lu, H.; Wang, B.; Roy, S.; Hornung, R.; Wissink, A.; Komanduri, R. (2005): Multiscale Simulations Using Generalized Interpolation Material Point (GIMP) Method And SAMRAI Parallel Processing. *Computer Modeling in Engineering and Science*, vol. 8, pp. 135-152.

Parker, S. G.; Guilkey, J. E.; Harman, T. (2006): A component-based parallel infrastructure for the simulation of fluid-structure interaction. *Engineering with Computers*, To Appear.

Shen, L.; Chen, Z.; (2005): A Silent Boundary Scheme with the Material Point Method for Dynamic Analyses. *Computer Modeling in Engineering and Science*, vol. 7, pp. 305-320.

Soedel, W. (2004): *Vibrations of Shells and Plates 3rd ed. rev. and expanded*. Marcel Dekker, Inc. New York.

Sulsky, D.; Chen, Z.; Schreyer, H. L. (1994): A particle method for history-dependent materials. *Computer Methods in Applied Mechanics and Engineering*, vol. 118, pp. 179-196.

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

Vshivkov, V. A. (1996): The approximation properties of the particles-in-cells method. *Computational Mathematics and Mathematical Physics*, vol. 36, pp. 509-515.