On the Order of Accuracy of the Generalized Interpolation Material Point Method

Philip C. Wallstedt

25 April 2008, 10:00 AM

University of Utah

Contents

List of Figures

1 Project Summary

The Generalized Interpolation Material Point (GIMP) [7] method and its parent, the Material Point Method [35, 38], are investigated for accuracy and stability and several improvements are developed and verified. While displaying less accuracy than the Finite Element (FE) method, GIMP has applicability on a wide range of problems, especially those involving contact and complex domains such as foams and natural materials like wood or human tissue.

In order to confidently verify the GIMP algorithm, its implementation, and the improvements developed for it, several manufactured solutions are created that are compatible with the assumptions made in the GIMP algorithm, especially the implied zero normal stress at all material boundaries. The solutions are developed following the de-facto standard for verification: the Method of Manufactured Solutions (MMS) [20, 4, 32, 2].

A technique is developed that uses velocity gradient information that is already available (for the calculation of rate of deformation) within the GIMP algorithm to improve the accuracy of velocity projection, including the exact projection of linear velocity functions. The enhanced velocity projection reduces the error of a solution by about forty percent for the cases discussed here, at almost no additional computational cost.

A broad range of time integration schemes is explored and one critical change is investigated and ultimately implemented in a 3D parallel fluid and solid-mechanics solver. The new time integration strategy stores velocity information at half time steps, enabling central difference time updates of all variables in the method. It is shown that second order spatial convergence is attained by making this improvement to the temporal integration. Some analysis of the close link between temporal and spatial effects in GIMP is also included.

A meshfree method is developed that provides compatible integration for the particles-in-a-grid framework using a least-squares scheme, improving over the gaps and overlaps that occur in native GIMP integration. The method developed in this proposal differs from typical meshfree methods in that it discards nearest neighbor searches in favor of fast projections of information to the Cartesian grid of GIMP. Such an approach is more compatible with modern cache-based computer architectures and large scale parallel computing.

Intellectual Merit: For the first time manufactured solutions are developed that provide a means by which to make precise measurements of accuracy for MPM and GIMP. The accuracy and stability properties of several non-linear time integration schemes are demonstrated for the first time, and one scheme is presented that induces second order spatial convergence. A new meshfree method is developed that fits within the particles-in-a-grid framework of GIMP while bringing additional smoothness and accuracy.

Broader Impacts: With the greater understanding and improvement of accuracy brought by this proposal, GIMP can be used for larger classes of problems. While scores of researchers currently rely on the method for problems with complex domains and/or extreme impact-penetration-deformation events, it will be opened to a much wider audience when it is better characterized. The development of manufactured solutions for GIMP (submitted by the author for publication in [41]) will enable other researchers who use GIMP to verify their implementations and to make additional algorithmic improvements. The manufactured solutions are also suitable for use by the meshfree and general computational solid mechanics communities and represent fully-fledged examples for the newly-standardized topic of verification [2]. They are more general and described in greater detail than typical manufactured solutions that we can point to in the literature.

2 Project Description

2.1 Introduction

The Generalized Interpolation Material Point (GIMP) Method is a particle-in-cell method for solid mechanics applications, described by Bardenhagen and Kober [7], that is a generalization of the Material Point Method (MPM) of Sulsky et al. [35, 38]. MPM in turn was drawn from ideas in FLIP [12, 13] and PIC [18, 19].

MPM and GIMP have been studied and used by numerous investigators, a subset of these important contributions include: analysis of time integration properties by Bardenhagen [5]; membranes and fluid-structure interaction by York, Sulsky and Schreyer [42, 43]; implicit time integration by Guilkey and Weiss [17], as well as Sulsky and Kaul [36]; conservation properties and plasticity by Love and Sulsky [27, 26]; contact by Bardenhagen et al. [6]; cracks and fracture by Nairn [29], and local mesh refinement by Ma et al. [28].

MPM and GIMP are convenient because they allow easy discretization of complex geometries, fast and straightforward contact treatments, robustness under large deformations and relative ease of parallel implementation. However, GIMP has largely defied the types of rigorous analysis that have been applied to say, the Finite Element Method (FEM). This is due, at least in part, to the mixed Eulerian-Lagrangian nature of the method, in which particles carry all state data, while the advancement of that state is carried out on the underlying grid.

GIMP improves spatial integration over MPM by treating particles as small blocks of material, instead of as Dirac delta functions, which does a great deal to reduce the errors and instabilities that potentially arise as particle distributions become disordered. Several variations of GIMP shape functions $S_i(r)$ are used in practice; the variations can all be grouped together in the general 1D form of Eq. 1 where x is position, i is the grid node index, p is the particle index, $r = |x_p - x_i|$, h is the size of a grid cell, and l_p is half-width of each particle. The first available choice in the list is always used:

$$S_i = \begin{cases} 1 - \frac{r^2+l_p^2}{2hl_p} & r < l_p \\ 1 - \frac{r}{h} & r < h - l_p \\ \frac{(h+l_p-r)^2}{4hl_p} & r < h + l_p \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

For (contiguous particle) cpGIMP the particle size is continuously updated according to the local deformation gradient F on each particle: $l_p^n = F^n l_p^0$. For (uniform or Utah) UGIMP the particle size is not changed: $l_p^n = l_p^0$, while for original Sulsky MPM $l_p = 0$. Tensor products of Eq. 1 are used to generate shape functions in 2D and 3D.

For each GIMP time step, velocity and acceleration are projected to the grid based on current particle values of position, velocity, and deformation gradient.

$$\mathbf{v}_i^n = \frac{\sum_p S_{ip} \mathbf{v}_p^n m_p}{\sum_p S_{ip} m_p} \tag{2}$$

$$\mathbf{f}_i^{int} = -\sum_p V_p^0 |\mathbf{F}_p^n| \nabla S_{ip} \cdot \boldsymbol{\sigma}(\mathbf{F}_p^n) \tag{3}$$

$$\mathbf{a}_i^n = \frac{\mathbf{f}_i^{int} + \mathbf{f}_i^{ext}}{\sum_p S_{ip} m_p} \tag{4}$$

Then the grid velocity is updated in a forward Euler manner and the updated velocity gradient is found.

$$\mathbf{v}_i^{n+1} = \mathbf{v}_i^n + \mathbf{a}_i^n \Delta t \tag{5}$$

$$\nabla \mathbf{v}_p^{n+1} = \sum_i \nabla S_{ip} \mathbf{v}_i^{n+1} \tag{6}$$

Finally, particle values of position, velocity and deformation gradient are updated.

$$\mathbf{x}_p^{n+1} = \mathbf{x}_p^n + \sum_i S_{ip} \mathbf{v}_i^{n+1} \Delta t \tag{7}$$

$$\mathbf{F}_p^{n+1} = \mathbf{F}_p^n + \nabla \mathbf{v}_p^{n+1} \mathbf{F}_p^n \Delta t \tag{8}$$

$$\mathbf{v}_p^n = \mathbf{v}_p^n + \sum_i S_{ip} \mathbf{a}_i^n \Delta t \tag{9}$$

Tracking of particle edges, described by Ma et al. [28] offers a vehicle by which to improve the size estimates of GIMP particles, although tracking the particle corners with sufficient accuracy has proved challenging in general simulations. An enhanced scheme for projecting particle data to the grid is described by Wallstedt and Guilkey here and in [40] which both reduces the error in this operation, and also provides more predictable behavior.

Several types of time integration improvements have been explored in the literature such as implicit integration [17, 36], central differencing [37, 41], and energy conservation effects [5, 26, 27]. But fundamental questions regarding the accuracy and stability of MPM and GIMP remain unanswered. Indeed it has been shown that MPM and GIMP defy traditional linear stability analysis methods [41].

In the last fifteen years a new class of "meshfree" methods has surged in popularity. Originating with the work of Nayroles et al. [30], the Element Free Galerkin (EFG) method of Belytschko and coworkers [11] set the stage for a large family of methods that was to follow such as the Meshless Local Petrov-Galerkin method [3], Reproducing Kernel Particle Method [24], and hp-clouds [14], to cite only a few. Common to these methods is the use of moving least squares [21] to ensure that low-degree polynomials and other functions are reproduced exactly from scattered data points. Dirichlet boundary conditions are a noticeable difficulty encountered with these methods [15], and the methods are also considered to be about an order of magnitude slower computationally than FEM. In order to mitigate these factors a hybrid combination of EFG and FEM was developed, called XFEM [34], where meshfree sampling points are embedded within a typical FEM mesh in certain critical locations such as cracks. Reproducibility is a central theme of meshfree methods and its relation to consistency and completeness is discussed at length in [8]. Most meshfree papers mention reproducibility and virtually all of them demonstrate first or second order numerical convergence for their implementations. Overviews of meshfree methods are in [9, 16] and an in-depth treatment of many methods is found in the text by Liu [23]. As part of the work proposed herein certain ideas are taken from the meshfree literature, especially reproducibility through least squares interpolation, and incorporated into the GIMP particles-in-a-grid framework to create a more accurate method that still retains many of the advantages that GIMP has to offer.

Lastly we draw attention to the compelling need for verification of scientific computer codes [2]. Many classes of experiments cannot be performed, such as testing bullet impact damage on the

human body, and many are too expensive, such as shaking a building until it falls. The codes used to simulate such phenomena require extraordinary predictive ability. The codes cannot merely interpolate between existing experiments; rather they must provide reliable answers from first principles. Therefore the demonstration of accuracy must be strong, thorough and objective. While classic FEM enjoys this level of predictability for small deformations with smooth motion and well-defined domains, GIMP has not been sufficiently-tested in this regard. Most papers using MPM and/or GIMP have used "eye-ball norms": visual comparison with experimental data or exact solutions. We know of only a few spatial convergence measures in the literature and those were for problems that were unrealistically simple; for example [7].

Broadly speaking, we can separate GIMP into spatial and temporal components such that spatial components influence the integrals and derivatives in a particular time step and temporal components define how the solutions advance from one time step to the next. In this proposal we explore both spatial and temporal improvements and precisely measure the benefits of each by the Method of Manufactured Solutions (MMS) [20, 4, 32].

2.2 Verifying Accuracy with Manufactured Solutions

MMS 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 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 non-linearities for which exact solutions will never be known. It is argued [20] that MMS is sufficient to verify a code, not merely necessary.

Although the usual purpose of MMS is to verify that a code's stated order of accuracy is observed in a particular implementation, we instead question assumptions of order and use MMS here to discover what order of accuracy the GIMP method, as well as our codes, can achieve.

For this paper we define two non-linear 2D dynamic manufactured solutions, and use them for subsequent testing. The solutions exercise the mathematical and numerical capabilities of the code and provide reliable answers about its accuracy and stability.

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. Solutions from Updated and Total Lagrange formulations are equivalent [10].

However, it turns out that it is necessary, or at least convenient, to manufacture solutions in the Total Lagrange formulation. This might at first appear to conflict with the fact that GIMP is always implemented in the Updated Lagrange form. But 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 equation of motion is presented in Total and Updated Lagrange forms, respectively:

$$\nabla \mathbf{P} + \rho_0 \mathbf{b} = \rho_0 \mathbf{a} \tag{10}$$

$$\nabla \boldsymbol{\sigma} + \rho \mathbf{b} = \rho \mathbf{a} \tag{11}$$

where $\mathbf{P}$ is the 1^{st} Piola-Kirchoff Stress; $\boldsymbol{\sigma}$ is Cauchy Stress; ρ is density; $\mathbf{b}$ is acceleration due to body forces; and $\mathbf{a}$ is 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:

$$\mathbf{P} = \lambda ln J \mathbf{F}^{-1} + \mu \mathbf{F}^{-1} \left(\mathbf{F}\mathbf{F}^T - \mathbf{I} \right) \tag{12}$$

$$\boldsymbol{\sigma} = \frac{\lambda ln J}{J} \mathbf{I} + \frac{\mu}{J} \left(\mathbf{F}\mathbf{F}^T - \mathbf{I} \right) \tag{13}$$

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

The acceleration $\mathbf{b}$ due to body forces is used as the MMS source term. The source term is manufactured such that the equations of motion are satisfied. We simply declare that the displacement will follow some reasonable but probably non-physical path, such as a sine function, and then determine the body force throughout the object that causes the assumed displacement to occur.

The definition of error used later in this proposal is chosen with the Total versus Updated Lagrange formulations in mind. On each particle the exact displacement in the Total Lagrange form is related to the computed displacement in the Updated Lagrange form by measuring the error δ as:

$$\delta_p = ||(\mathbf{x}_p - \mathbf{X}_p) - \mathbf{u}_{exact}(\mathbf{X}_p, t)||. \tag{14}$$

We define a single pessimistic, but trustworthy, measure of error for a complete solution as the L_∞ norm over all particles and all time steps:

$$L_\infty = max(\delta_p). \tag{15}$$

Two 2D cases are drawn from the equation of motion and discussed in detail in the next two sections.

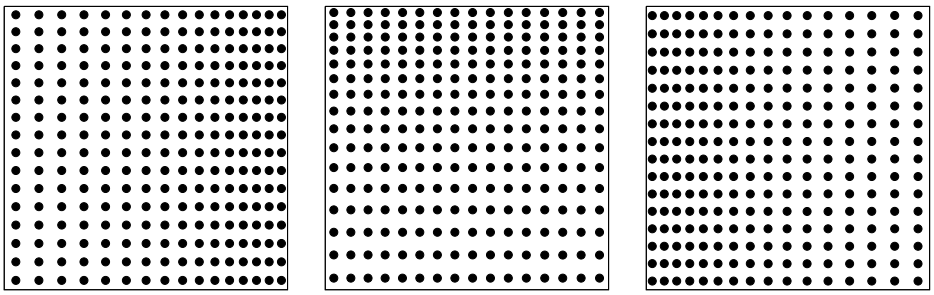
2.2.1 Axis-Aligned Displacement in a Unit Square

Displacement in a unit square is prescribed with normal components only. Through this choice, the corners and edges of GIMP particles are coincident and co-linear. 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 [7]. While it is not representative of general material deformations, it does allow characterization of the error introduced via the inexact approximations to GIMP, e.g., use of a constant sized particle characteristic function.

The plane strain displacement field is chosen to be:

$$\mathbf{u} = \begin{pmatrix} Asin(\pi X)cos(c\pi t) \\ Asin(\pi Y)sin(c\pi t) \\ 0 \end{pmatrix} \tag{16}$$

where X and Y are the scalar components of position in the reference configuration, t is time, A is the maximum amplitude of displacement and c is wave speed such that $c^2 = \frac{E}{\rho_0}$ where E is Young's modulus; see Figure 1.



(a) $t = 0$ (b) $t = \frac{1}{2c}$ (c) $t = \frac{1}{c}$

Figure 1: Illustration of particle displacements at three representative times.

The stress is found by substituting deformation gradient into Eq. 12:

$$\mathbf{P} = \begin{pmatrix} \frac{\lambda}{F_{XX}}K + \frac{\mu}{F_{XX}}(F_{XX}^2 - 1) & 0 & 0 \\ 0 & \frac{\lambda}{F_{YY}}K + \frac{\mu}{F_{YY}}(F_{YY}^2 - 1) & 0 \\ 0 & 0 & \lambda K \end{pmatrix} \tag{17}$$

where $K = ln(F_{XX}F_{YY})$ and the subscripts on $\mathbf{u}$ and $\mathbf{F}$ indicate individual terms of displacement and deformation gradient (which is only non-zero on the diagonals) equations.

Acceleration is found by twice differentiating displacement Eq. 16 in time. Finally, substituting stress $\mathbf{P}$ into Eq. 10 and solving for the body force $\mathbf{b}$ (used as the MMS source term) it is found that:

$$\mathbf{b} = \begin{pmatrix} \frac{\pi^2 u_X}{\rho_0}\left[\frac{\lambda}{F_{XX}^2}(1-K) + \mu\left(1 + \frac{1}{F_{XX}^2}\right) - E\right] \\ \frac{\pi^2 u_Y}{\rho_0}\left[\frac{\lambda}{F_{YY}^2}(1-K) + \mu\left(1 + \frac{1}{F_{YY}^2}\right) - E\right] \\ 0 \end{pmatrix}. \tag{18}$$

A 3D version of this problem has been developed and used to verify the correctness of the University of Utah Uintah-GIMP implementation.

2.2.2 Radial Expansion of a Ring

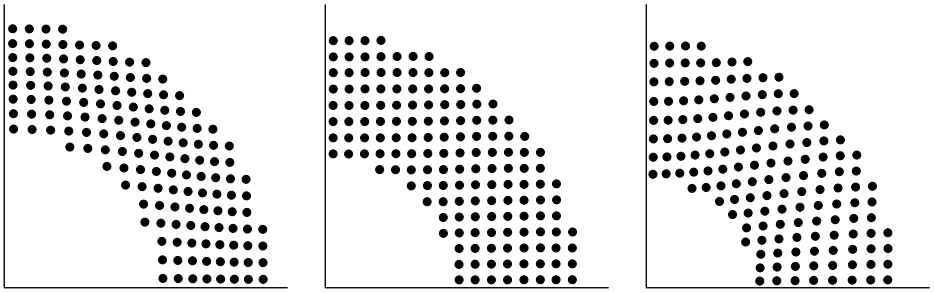
Displacement is prescribed with radial symmetry for a ring as

$$u(R) = Acos(c\pi t)(c_3R^3 + c_2R^2 + c_1R) \tag{19}$$

where R (and θ) represent cylindrical coordinates in the reference configuration. A is the maximum magnitude of displacement (10% of R_O in this case), t is time, and c is the wave speed in the material. See Figure 2 for an illustration of displacements.

The constants c_3, c_2, and c_1 are chosen so that the field always provides for zero normal stress on the inner (R_I) and outer (R_O) surfaces of the ring and so that $u(R_O) = A$:

$$c_3 = \frac{-2}{R_O^2(R_O - 3R_I)}, \quad c_2 = \frac{3(R_O + R_I)}{R_O^2(R_O - 3R_I)}, \quad c_1 = \frac{-6R_I}{R_O(R_O - 3R_I)} \tag{20}$$



(a) $t = 0$ (b) $t = \frac{1}{2c}$ (c) $t = \frac{1}{c}$

Figure 2: Illustration of particle displacements at three representative times.

The neo-Hookean constitutive model of Eq. 12 with zero Poisson's ratio is used to make the manufactured solution tractable. This is deemed acceptable because behavior for non-zero Poisson's ratio has already been represented by the axis-aligned problem. Free surfaces provide the complicating factor for this scenario.

We relate the Cartesian components of displacement in terms of the reference coordinates X and Y where $R^2 = X^2 + Y^2$:

$$\mathbf{u} = \begin{pmatrix} Acos(c\pi t)(c_3R^3 + c_2R^2 + c_1R)\frac{X}{R} \\ Acos(c\pi t)(c_3R^3 + c_2R^2 + c_1R)\frac{Y}{R} \\ 0 \end{pmatrix} \tag{21}$$

Equations for velocity, acceleration, and deformation gradient are straightforward to find by differentiating Eq. 21 with respect to time and position. It is more difficult to solve Eq. 12 (with zero Poisson's ratio) for the MMS source term, and the resultant equations are quite unwieldy. We use the Maple symbolic manipulation package to achieve a solution, and to generate C-compatible source code. Additional detail can be found in [41].

2.3 Gradient Enhancement

Let us begin by examining the effect of particle position within grid cells. GIMP particles are always initialized so they fill a cell in an equally-spaced manner with particle positions at particle centers. As a solution progresses the particles move from their initial positions and some of them cross into neighboring cells. The asymmetry of general particle positions introduces error into larger-displacement solutions that does not show up for quasi-static problems.

Now we set aside the remainder of the GIMP time step and just focus on the velocity projection of Eq. 2 By way of illustration the velocity projection errors for several simple 1D cases are related in Figure 3. None of the four cases is able to correctly project a simple linear velocity function (nor any higher degree function), which is analogous to a lack of consistency or completeness. Although GIMP is a non-linear method and cannot be expected to strictly obey the Lax theorem, its inability to exactly reproduce the simplest of functions is annoying, if not troubling.

In order to address this issue a method is developed in [40] and summarized here that uses the velocity gradient information that is already available (for the calculation of rate of deformation)

within the GIMP algorithm to improve the accuracy of the velocity 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 $\frac{\partial v_p}{\partial x}$.

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 in 1D as v^e_{ip}.

$$v^e_{ip} = v^e - \frac{\partial v_p}{\partial x}(x_p - x_i) \tag{22}$$

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 $x_p = \sum_i x_i S_{ip}$, 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 is then modified to use the extrapolated velocities and the remainder of the time step proceeds as usual:

$$v_i = \frac{\sum_p v^e_{ip} m_p S_{ip}}{\sum_p m_p S_{ip}} \tag{23}$$

The 3D form for gradient-enhanced particles is simply:

$$\mathbf{v}^e_{ip} = \mathbf{v}^e - \nabla \mathbf{v} \cdot (\mathbf{x}_p - \mathbf{x}_i) \tag{24}$$

The advantage of being able to reproduce linear velocity functions is highlighted with a 1D study of velocity projection error. A global particles-per-cell (PPC) number is defined as the total number of particles divided by the total number of cells in a 1D grid. For this test the particles are always evenly spaced, but one by one additional particles are "squeezed" into the grid which gradually raises the PPC. A plot is made of the maximum errors resulting from velocity projection of a linear function over a range of global PPC numbers; see Fig. 4. When the number of particles within one cell is an integer value then the velocity projection is exact. But for all other particle arrangements some error is observed in the form of distinct "humps" where particles are arranged in realistic (non-ideal) positions. However the error is substantially less for GIMP than for MPM.

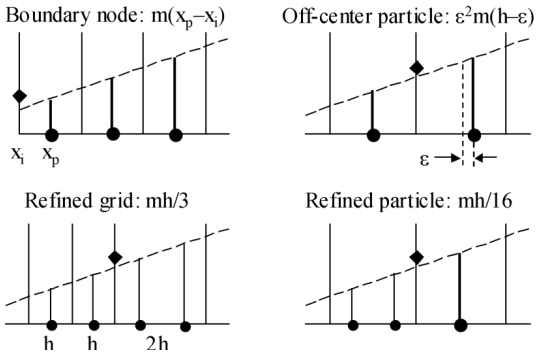


Figure 3: Error for Degenerate Cases

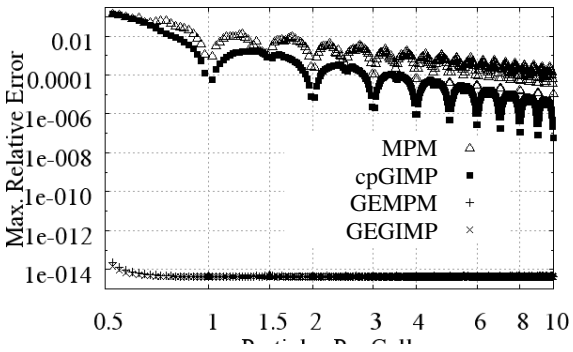


Figure 4: Error for Linear Function

It was found in [40] that the upper bound of error for the projection of charge density in PIC, as derived by Vshivkov [39] resembles several of the trends observed in velocity projection in GIMP. The upper bound is:

$$\delta_k \leq \left(\frac{3\rho_{avg}^2}{\rho_{min}} + h\frac{\rho_{avg}^2 \rho_{max}}{\rho_{min}^3} \left| \frac{\partial \rho}{\partial x} \right|_{max} \right) \frac{1}{PPC^2} + \frac{h^2}{12} \left| \frac{\partial^2 \rho}{\partial x^2} \right|_{max} \tag{25}$$

where ρ is charge density and h is cell size. Vshivkov's result suggests that the error occurs as a sum of two terms that can act independently and which must both be driven toward zero. However, when gradient enhancement is used (denoted by the GEMPM and GEGIMP series) then a linear function is reproduced exactly, regardless of particle placement, as evidenced by the machine precision error for the gradient enhanced cases.

Gradients of velocity are already present within the GIMP algorithm; they are calculated in order to update deformation gradient. Instead of being discarded at the end of a cycle, the velocity gradients can be saved and used as input for the next cycle at a very modest computational cost.

2.4 Alternative Time Integration Algorithms

The choice of time integration strategy originally implemented in Uintah was based on the performance of MPM. It was chosen to offset discontinuities in the velocity field by updating stress first (called the USF method) - before grid velocity is updated by the current acceleration. Subsequent experience has shown that, with GIMP's increased accuracy, a smooth velocity field is produced and more accurate time integration results from updating stress last (USL) as described in Section 2.1 and originally used in [35, 38]. A detailed analysis and comparison of USF and USL was performed in [5]. By initializing velocity to a negative half time step the algorithm becomes a centered-difference (CD) method [36] which gives second order performance for smooth problems. The methods are compared briefly in Table 1.

For problems that start with non-zero acceleration, the initialization of velocity to a negative half time step becomes non-trivial; if omitted, accuracy cannot be better than first order. If the true solution is known at the negative half time step then it should be used. However, we have found that it is convenient to simply halve the grid acceleration of Eq. 35 for the first time step: $\mathbf{a}_i^0 = \frac{1}{2}\mathbf{a}_i^0$. This correctly propagates the negative half time step initialization throughout the algorithm and has not displayed any loss of accuracy in the tests that we have performed.

2.5 Results of Convergence Analysis

The order of accuracy of GIMP is measured using results from the manufactured solutions of Section 2.2. This is the first time such data has been available for GIMP or MPM. The convergence measures that follow do not merely verify a coded implementation of GIMP. They also help to suggest a hypothesis for the accuracy of GIMP in general scenarios and illustrate the critical role played by the contiguity or compatibility of integration.

Solution accuracy is examined by measuring the error in displacement, as defined by Eq. 15, of computed solutions over ranges of mesh sizes and CFL numbers. The test code is configured so that an error of one is returned whenever a particular solution crashes.

2.4.1 USF - Update Stress First

Velocity and its gradient are found at the current time and used to update deformation gradient to the next time step:

$$\mathbf{v}_i = \frac{\sum_p S_{ip}\mathbf{v}_p^n m_p}{\sum_p S_{ip} m_p} \tag{26}$$

$$\nabla \mathbf{v}_p^n = \sum_i \nabla S_{ip}\mathbf{v}_i \tag{27}$$

$$\mathbf{F}_p^{n+1} = \mathbf{F}_p^n + \nabla \mathbf{v}_p^n \mathbf{F}_p^n \Delta t \tag{28}$$

A constitutive model finds stress as a function of the new deformation gradient, enabling calculation of internal force:

$$\mathbf{f}_i^{int} = -\sum_p \nabla S_{ip} \cdot \boldsymbol{\sigma}(\mathbf{F}_p^{n+1}) V_p^0 |\mathbf{F}_p^{n+1}| \tag{29}$$

Finally, acceleration is calculated and used to update the position and velocity on the particles:

$$\mathbf{a}_i = \frac{\mathbf{f}_i^{int} + \mathbf{f}_i^{ext}}{\sum_p S_{ip} m_p} \tag{30}$$

$$\mathbf{x}_p^{n+1} = \mathbf{x}_p^n + \sum_i S_{ip}(\mathbf{v}_i + \mathbf{a}_i \Delta t)\Delta t \tag{31}$$

$$\mathbf{v}_p^{n+1} = \mathbf{v}_p^n + \sum_i S_{ip}\mathbf{a}_i \Delta t \tag{32}$$

2.4.2 CD - Centered Difference

Velocity and acceleration are projected to the grid based on current information. The particle velocity lags behind by a half time step [36].

$$\mathbf{v}_i^{n-\frac{1}{2}} = \frac{\sum_p S_{ip}\mathbf{v}_p^{n-\frac{1}{2}} m_p}{\sum_p S_{ip} m_p} \tag{33}$$

$$\mathbf{f}_i^{int} = -\sum_p V_p^0 |\mathbf{F}_p^n| \nabla S_{ip} \cdot \boldsymbol{\sigma}(\mathbf{F}_p^n) \tag{34}$$

$$\mathbf{a}_i^n = \frac{\mathbf{f}_i^{int} + \mathbf{f}_i^{ext}}{\sum_p S_{ip} m_p} \tag{35}$$

Now the grid velocity is updated in a central difference fashion:

$$\mathbf{v}_i^{n+\frac{1}{2}} = \mathbf{v}_i^{n-\frac{1}{2}} + \mathbf{a}_i^n \Delta t \tag{36}$$

With smoother GIMP shape functions the gradient of velocity can be found from updated grid velocity:

$$\nabla \mathbf{v}_p^{n+\frac{1}{2}} = \sum_i \nabla S_{ip}\mathbf{v}_i^{n+\frac{1}{2}} \tag{37}$$

Position and deformation gradient can now be updated in a central-difference manner because the updated grid velocity is ahead by a half time step.

$$\mathbf{x}_p^{n+1} = \mathbf{x}_p^n + \sum_i S_{ip}\mathbf{v}_i^{n+\frac{1}{2}} \Delta t \tag{38}$$

$$\mathbf{F}_p^{n+1} = \mathbf{F}_p^n + \nabla \mathbf{v}_p^{n+\frac{1}{2}} \mathbf{F}_p^n \Delta t \tag{39}$$

The velocity update is also central difference; velocity goes from time step $n-1/2$ to $n+1/2$ from the midpoint acceleration at n.

$$\mathbf{v}_p^{n+\frac{1}{2}} = \mathbf{v}_p^{n-\frac{1}{2}} + \sum_i S_{ip}\mathbf{a}_i^n \Delta t \tag{40}$$

Table 1: Comparison of USF and CD Time-Stepping Algorithms

2.5.1 Convergence for Axis-Aligned Displacement

Temporal convergence results for several interesting combinations of time integration algorithm and shape function are plotted in Fig. 5.

One significant pattern is common to all the data series in the figure: Hardly any of the combinations of time and space schemes displays temporal convergence. Most seem unaffected by changes in CFL until some maximum stable CFL is reached after which the solution simply crashes.

However, USL-cpGIMP deviates from this pattern and displays clear first order temporal convergence for CFL > 0.2. But for CFL < 0.2 the rate of convergence changes to zero. Later on we will observe that spatial convergence for USL-cpGIMP also displays a change in convergence rate.

USF-cpGIMP converges in a crude way, but this signifies that it only holds together for very low values of CFL. Although CD-MPM is able to solve the problem over a range of CFL values, its error is more than two orders of magnitude greater than the CD-cpGIMP baseline. It has been our experience that MPM produces poor quantitative convergence when high stress and large deformation occur together. However, MPM performs acceptably with either high stress and infinitesimal deformation or with low stress and high advection, deformation and contact.

The best algorithms from the figure all show up as horizontal lines: USL-cpGIMP, CD-cpGIMP, and CD-GEGIMP. The slight difference between USL and CD – the half time-step initialization – produces an order of magnitude reduction of error. GEGIMP improves over cpGIMP with a roughly forty percent decrease in error, although at a cost of slightly reduced stability.

By improving the time integration algorithm to CD, temporal convergence is entirely eliminated; the rate of convergence becomes zero. Our explanation for this unexpected behavior is that the CD scheme reduces error from temporal sources to a low enough level that spatial error dominates the results over the entire range of valid CFL numbers. Therefore we resolve that additional improvements to GIMP should concentrate on improving the accuracy of spatial elements of the algorithm. Although we have tried several higher-order time integration schemes (not described here), none has improved overall accuracy above that provided by CD. We believe this strengthens the resolution that spatial effects, and element integration in particular, should be the focus of future research efforts. Later in this proposal we present a significant advance to spatial integration using least squares interpolation.

The temporal convergence results suggest a strong coupling between temporal and spatial effects that has been observed in GIMP in many ways. We next examine spatial convergence and how it is influenced by the choice of time integration. Spatial convergence results are plotted in Fig. 6 using the same symbols as in Fig. 5.

Both USF-cpGIMP and CD-MPM display unsatisfactory performance in that they fail to show convergence with decreasing cell size. However, we note with interest that neither method crashes, rather both continue to provide solutions that are visually acceptable even when their fundamental accuracy falters.

CD-UGIMP displays initially promising second order convergence, but accuracy is lost as the mesh is refined, evidently due to the spatial integration error that results from constant sized particles not filling the domain exactly.

Gradient Enhancement generates a reduction of error of roughly forty percent. However, this is accompanied by reduced stability and a loss of accuracy at fine mesh resolutions. The cause for this is not entirely explained although it is believed to be related to the gradients of velocity becoming more delicate with higher resolution.

Finally, the CD-cpGIMP combination is satisfyingly second order in space. We are reminded

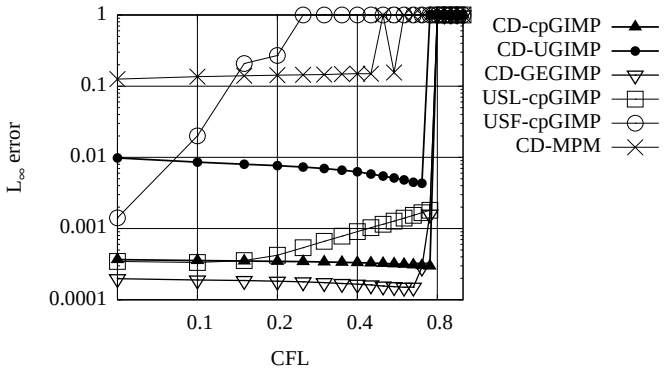


Figure 5: Temporal Convergence for Axis-Aligned Displacement

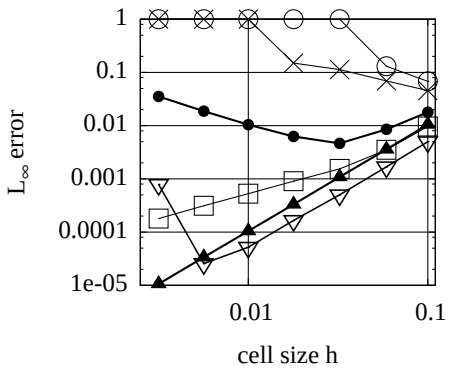


Figure 6: Spatial Convergence for Axis-Aligned Displacement

that this excellent behavior is only displayed for the special circumstances of the axis-aligned problem where no gaps or overlaps exist in the cell integration. The behavior of a more realistic problem is assessed in the next section.

2.5.2 Convergence for Expanding Ring

Temporal convergence results are generated for the expanding ring in Fig. 7. All solutions use 56^2 cells and $A = 0.1$ from Eq. 21. The curved surfaces of the ring are "stair-stepped" approximations in the particle representation.

Temporal convergence trends are somewhat different for the expanding ring as compared to the axis-aligned problem. UGIMP and Gradient Enhancement perform just as well as cpGIMP and USL performs better, compared to CD, than it did for the axis-aligned problem. This suggests that common factors dominate the results for all the methods and we believe that the most important of these is the gaps and overlaps in the particle representation that cause inaccuracies in the spatial integration due to non axis-aligned displacements in the ring.

The CD-MPM trend uses 25 particles per cell rather than four and requires significantly more computational effort. Even with this extra advantage it only gives acceptable results where CFL < 0.1. For reasons of poor performance the USF and MPM options are omitted from the spatial convergence results shown next.

Spatial convergence for the ring is shown in Fig. 8 using the same symbols as in Fig. 7. All solutions use four initially equally-spaced particles per cell with CFL $= 0.4$ and $A = 0.1$.

For the most part the trends display nominally first order convergence as compared to the second order convergence for the axis-aligned solutions. USL has more error than CD simply because of its first order initialization error, which is shown to decrease as time step sizes decrease. The loss of convergence that we expect to see with UGIMP occurs only at high resolution – the finest mesh size of the series.

The second order effects, seen in the axis-aligned problem, that differentiate USL, GEGIMP and UGIMP from the CD-cpGIMP baseline are less evident as the error is now dominated by the stair-stepped surface approximation and by gaps and overlaps among adjacent particles in the spatial integration. Due to the deformation, the latter of these is not eliminated by a cpGIMP treatment of the particle sizes.

Based on the preceding results and on experience with many other solutions in GIMP an

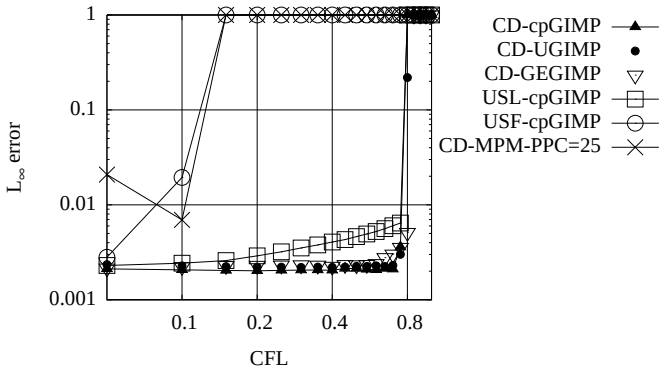


Figure 7: Temporal Convergence for Expanding Ring

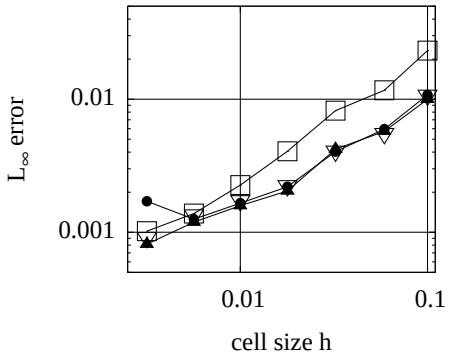


Figure 8: Spatial Convergence for Expanding Ring

informal theory for accuracy in GIMP is presented. We emphasize that this does not rise to the level of an estimate of theoretical accuracy and is not backed by analysis or discovery of an upper bound for error. Rather it is offered as a guide and as a rule-of-thumb that ties together all the results we have seen up until now. The informal theory is:

For large (and small) deformations GIMP is second order accurate in space when:

- Particles cover the material without gaps or overlaps
- Time integration is sufficiently accurate that spatial error dominates

For large deformations GIMP is up to first order accurate in space when:

- Particles initially cover the material but gaps and/or overlaps occur with deformation
- Particles are re-sized at every time step to better estimate the current deformation
- Time integration is sufficiently accurate that spatial error dominates

In the next section a method is presented that uses particle data within a weighted least squares framework to improve the way that integration is performed over the material domain.

2.6 Weighted Least Squares in the GIMP Framework

Significant analysis and testing (see [33] and Section 2.5) has indicated that much of the error that remains in GIMP, after temporal error is reduced to a subordinate role, comes from its spatial integration procedure. The sides of cpGIMP and UGIMP particles are always assumed to lay in coordinate planes even though this may allow gaps and overlaps to occur with realistic motion of particles. Whereas the FEM covers the material with a mesh throughout the solution, GIMP only covers the material at its initial configuration and thereafter the volume of material covered by cells and the volume of material tracked on particles may differ; see Figure 9 for a comparison of volume partitioning for FEM, GIMP, and the method of this proposal. Although this results in inferior accuracy for GIMP compared to FEM, it also results in the major advantage of enabling the method to work with very complex domains such as foams, composite structures, biological materials, and material accretion or disintegration.

Therefore a method is proposed that uses the sampling points of GIMP to approximate the surface of a material, then uses weighted least squares to generate a function over the grid which is integrated by grid cells rather than particle volumes. Cells on the material boundary are subdivided

into elements with special integration rules. A method like this may offer some advantages of both FEM and GIMP, or at least would allow easy transition between the two. For example, an over-pressurized tank can be modeled with the high-order weighted least squares scheme until the tank ruptures, and then the rupture surface and subsequent material disintegration can be modeled with GIMP.

The method developed in this proposal discards the nearest neighbor searches of typical meshfree methods in favor of the fast projections of information to a Cartesian grid used by GIMP. Such an approach is more compatible with modern cache-based computer architectures and large scale parallel computing.

The proposed method will retain desirable features of GIMP such as its smooth interface with existing Cartesian fluid mechanics codes and parallelization frameworks, and convenient handling of complex domains, extreme deformations, contact and material inter-penetration. Yet the new method will draw additional accuracy from the improved material cover scheme and least squares interpolation of meshfree methods.

The new method will be tested by comparison of results to known solutions, code verification via the Method of Manufactured Solutions, and stability testing for response to perturbation.

2.6.1 Details of the Method

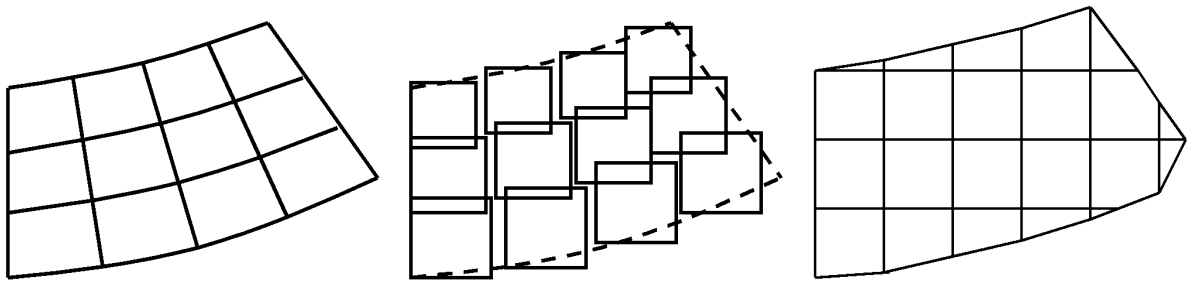
First, a conceptual overview of the method is presented and comparisons to GIMP and FEM are highlighted. Then details of key equations are related.

The method relies on many framework features of GIMP. Points of information are positioned within a fixed Cartesian grid; information is transferred to the grid in order to find gradients; then the points of information change their position and properties based on an explicit time integration update.

The key difference proposed is that cells are treated as finite elements and integration is performed over cells, rather than using a single number on each particle to represent its value and volume. Cells along the material boundary are divided into interior and exterior regions; the interior regions may be further split into triangles (or tetrahedra in 3D) and integrated by standard Gauss quadrature.

While function values in GIMP are only defined on particle centers, functions are defined throughout the material-covered region in the proposed method by means of weighted least squares [31]. This is in contrast to the moving least squares of meshfree methods [9, 16, 23, 21] which do not involve transfer of information to a grid.

Now we touch upon some of the mathematical details of the proposed method. We begin with



(a) Finite Element Volumes (b) GIMP Particle Volumes (c) WLS Cell Volumes

Figure 9: Comparison of Material Cover and Volume Partition Strategies

the WLS procedure, then show how it is used in a finite element context.

In 1D a quadratic function $f(r)$ (or any function), where $r = x_p - x_i$, can be written as a combination of a basis vector $\mathbf{b} = [1, r, r^2]$ and coefficient vector $\mathbf{c} = [c_0, c_1, c_2]$:

$$f(r) = c_0 + c_1 r + c_2 r^2 = \mathbf{b} \cdot \mathbf{c} \tag{41}$$

The function $f(r)$ represents a best approximation to the particle quantities f_p that fall within the local region of a node. To find the coefficients $\mathbf{c}$ we minimize, with respect to $\mathbf{c}$, the weighted discrete L_2 norm

$$L_2 = \sum_p^n W[f(r) - f_p]^2 \tag{42}$$

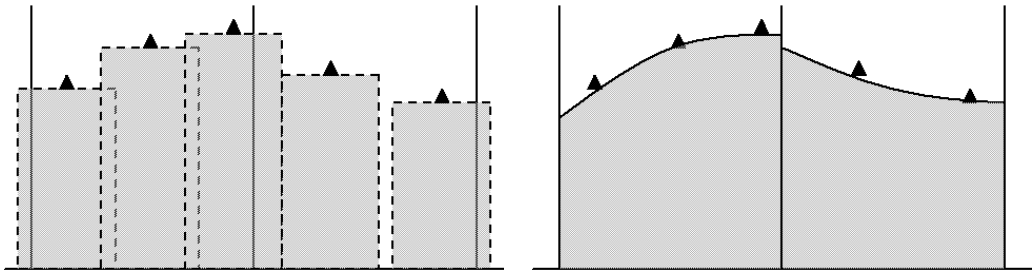
where the weight function W is a partition of unity with compact support, such as the typical piecewise-linear shape function of MPM, and n is the number of points with non-zero weight near node i. The weight function allows points to move from one cell to another smoothly.

The partial derivatives of L_2 are taken with respect to the coefficients $\mathbf{c}$, forming a small system of equations. The coefficients come out of the sums and the equations can be put in matrix form $\mathbf{Mc} = \mathbf{g}$ so the coefficients become $\mathbf{c} = \mathbf{M^{-1}g}$. The moment matrix $\mathbf{M}$ consists of weights and point locations only; the load vector $\mathbf{g}$ contains all information about the specific value f_p on each point that contributes to the system. In this way the inverse (or LU factorization) of the moment matrix can be stored for each point and then re-used to find the coefficients for several different load vectors.

Grid values for density, stress, external force and other possible quantities are approximated by WLS. Then for any quantity on the particles f_p the value of the function $f(r)$ on or near the material is interpolated from nodes in the support as: $f(r) = \sum_i W_i(r)\mathbf{b}(r) \cdot \mathbf{c}_i$. Gradients are calculated via the chain rule:

$$\frac{\partial f(r)}{\partial r} = \sum_i \left[\frac{\partial W_i(r)}{\partial r}\mathbf{b}(r) \cdot \mathbf{c}_i + W_i(r)\frac{\partial \mathbf{b}(r)}{\partial r} \cdot \mathbf{c}_i \right] \tag{43}$$

The deformation gradient $\mathbf{F}$ can be found from the displacement gradient of the current configuration $\mathbf{F} = [\mathbf{I} - \nabla \mathbf{u}(\mathbf{x})]^{-1}$ where $\mathbf{u}$ is displacement. It can also be carried forward in time in the manner of GIMP. Both options have been used successfully in 1D tests. The stress is drawn from the constitutive model and is calculated at each Gauss quadrature point $\boldsymbol{\sigma}_q = \boldsymbol{\sigma}(F_q)$. With the integrals defined the method proceeds to solve the weak form of Eq. 11 in the customary manner of GIMP and FEM.



(a) GIMP (b) WLS

Figure 10: Comparison of Integration over Particle Volumes and Cell Volumes

The new meshfree method is tested in 2D, but without the complication of a free surface, using the axis-aligned manufactured solution presented in Section 2.2. Initial results for the new meshfree method are excellent. The method demonstrates stability and converges as well as the CD-cpGIMP baseline; see Figure 11.

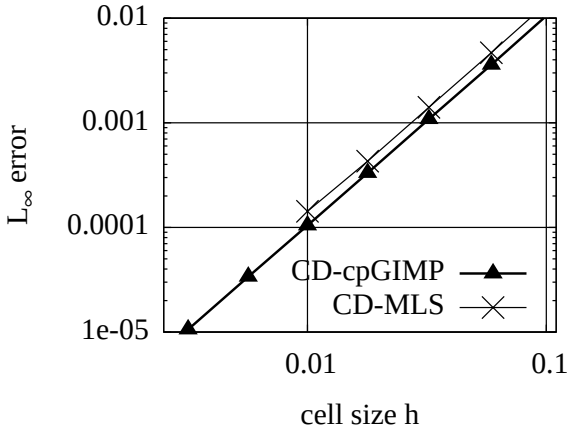


Figure 11: Weighted Least Squares Performance for Axis-Aligned Displacement

Lastly we take up the subject of surface-finding for the proposed method and the way in which surface cells are subdivided for integration purposes. This detail of the method is under active development and several variations of surface-finding methods have been or will be tried. The surface-finding routine related here works well in 1D; its application to multi-D is in progress.

A moving material boundary Γ is found by projecting the Jacobian $J = det(\mathbf{F})$ from the material points to the grid:

$$\Gamma_i = \frac{1}{PPC} \sum_p W_{ip} J_p \tag{44}$$

where PPC is the initial number of particles per cell. For $\Gamma > 1/2$ a point is inside the body; otherwise it is outside. If a cell has at least one node inside and one node outside then it is a boundary cell and is subdivided into interior and exterior pieces such that integrals of quantities involving partial cells are correct. For a 2D cell there are $2^4 = 16$ possible ways to subdivide it and for a 3D cell there are $2^8 = 256$ possible subdivisions. The Marching Cubes algorithm of Lorensen and Cline [25] describes a way to process these various combinations quickly. The adaptation required in this case is that volume regions are generated rather than the surface regions provided by Marching Cubes.

Future investigations of this method will concentrate on robust surface approximations. A wide of design choices is available, any of which will change the surface-finding directly or will allow a particular surface-finding method to achieve stability and accuracy. Some design choices that have been considered but not yet implemented are: use a 2D parabolic basis rather than the current planar basis; use splines with wider support as weight functions; sample integrals at particle positions rather than Gauss points; create special surface particles that carry nearest-neighbors information; define a surface via the Point Set Surfaces of [22, 1]; and investigate implicit surface and level set methods from the graphics community.

3 Activity Schedule

Figure 12 represents the time spent on each component of the research proposal. While most components are never really completed, the black coloring indicates when a component is in the forefront of research efforts and takes up a majority of my time.

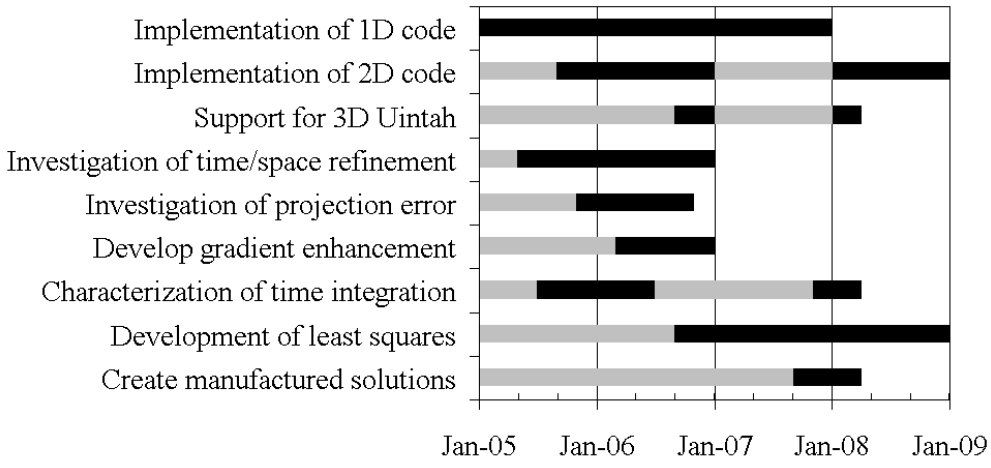


Figure 12: Schedule of Active Research Components

References

[1] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva. Computing and rendering point set surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 9, 2003.

[2] The American Society of Mechanical Engineers. *Guide for Verification and Validation in Computational Solid Mechanics*, 2006.

[3] S. N. Atluri and T. Zhu. A new meshless local petrov-galerkin (mlpg) approach in computational mechanics. *Comput. Mech.*, 22:117–127, 1998.

[4] B. Banerjee. Method of manufactured solutions. www.eng.utah.edu/ banerjee/Notes/MMS.pdf, October 2006.

[5] S. G. Bardenhagen. Energy conservation error in the material point method for solid mechanics. *Journal of Computational Physics*, 180:383–403, 2002.

[6] S. G. Bardenhagen, J. E. Guilkey, K. M. Roessig, J. U. Brackbill, W. M. Witzel, and J. C. Foster. An improved contact algorithm for the material point method and application to stress propagation in granular material. *Computer Modeling in Engineering and Sciences*, 2:509–522, 2001.

[7] S. G. Bardenhagen and E. M. Kober. The generalized interpolation material point method. *Computer Modeling in Engineering and Sciences*, 5:477–495, 2004.

[8] T. Belytschko, Y. Krongauz, J. Dolbow, and C. Gerlach. On the completeness of meshfree particle methods. *Int. J. Numer. Methods Eng.*, 43:785–819, 1998.

[9] T. Belytschko, Y. Krongauz, D. Organ, M. Fleming, and P. Krysl. Meshless methods: An overview and recent developments. *Comput. Methods Appl. Mech. Engrg.*, 139:3–47, 1996.

[10] T. Belytschko, W. K. Liu, and B. Moran. *Nonlinear Finite Elements for Continua and Structures*. John Wiley and Sons, LTD, 2000.

[11] T. Belytschko, Y. Y. Lu, and L. Gu. Element-free galerkin methods. *Int. J. Numer. Methods Eng.*, 37:229–256, 1994.

[12] J. U. Brackbill and H. M. Ruppel. Flip: A low-dissipation, particle-in-cell method for fluid flows in two dimensions. *J. Comp. Phys.*, 65:314–343, 1986.

[13] J. U. Brackbill and H. M. Ruppel. Flip mhd: A particle-in-cell method for magnetohydrodynamics. *J. Comp. Phys.*, 96:163–192, 1991.

[14] C. A. Duarte and J. T. Oden. An hp adaptive method using clouds. *Comput. Methods Appl. Mech. Eng.*, 139:237–262, 1996.

[15] S. Fernandez-Mendez and A. Huerta. Imposing essential boundary conditions in mesh-free methods. *submitted to Elsevier Science*, 2003.

[16] T. P. Fries and H. G. Matthies. Classification and overview of meshfree methods. Technical Report 2003-3, Institut fur Wissenschaftliches Rechnen, Technische Universitat Braunschweig, 2004.

[17] J. E. Guilkey and J. A. Weiss. Implicit time integration for the material point method: Quantitative and algorithmic comparisons with the finite element method. *International Journal for Numerical Methods in Engineering*, 57:1323–1338, 2003.

[18] F. H. Harlow. Hydrodynamic problems involving large fluid distortion. *J. Assoc. Comp. Mach.*, 4:137, 1957.

[19] F. H. Harlow. The particle-in-cell computing method for fluid dynamics. *Methods Comput. Phys.*, 3:319–343, 1963.

[20] P. Knupp and K. Salari. *Verification of Computer Codes in Computational Science and Engineering*. Chapman and Hall/CRC, 2003.

[21] P. Lancaster and K. Salkauskas. Surfaces generated by moving least squares methods. *Math. Comput.*, 37:141–158, 1981.

[22] D. Levin. Mesh-independent surface interpolation. *Advances in Computational Math.*, 2001.

[23] G. R. Liu. *Mesh free methods: moving beyond the finite element method*. CRC Press, 2003.

[24] W. K. Liu, S. Jun, and Y. Zhang. Reproducing kernel particle methods. *Int. J. Numer. Methods Fluids*, 20:1081–1106, 1995.

[25] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *Computer Graphics*, 21(4), 1987.

[26] E. Love and D. L. Sulsky. An energy-consistent material-point method for dynamic finite deformation plasticity. *International Journal for Numerical Methods in Engineering*, 65:1608–1638, 2005.

[27] E. Love and D. L. Sulsky. An unconditionally stable, energymomentum consistent implementation of the material-point method. *Computer Methods in Applied Mechanics and Engineering*, 195:3903–3925, 2006.

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

[29] J. A. Nairn. Material point method calculations with explicit cracks. *Computer Modeling in Engineering and Sciences*, 4:649–663, 2003.

[30] B. Nayroles, G. Touzot, and P. Villon. Generalizing the finite element method: diffuse approximation and diffuse elements. *Comput. Mech.*, 10:307–318, 1992.

[31] A. Nealen. An as-short-as-possible introduction to the least squares, weighted least squares and moving least squares methods for scattered data approximation and interpolation. Discrete Geometric Modeling Group TU Darmstadt, http://www.nealen.com/projects/mls/asapmls.pdf, May 2004.

[32] L. Schwer. Method of manufactured solutions: Demonstrations. www.usacm.org/vnvcsm/PDF_Documents/MMS-Demo-03Sep02.pdf, August 2002.

[33] M. Steffen, M. Berzins, and R. M. Kirby. Analysis and reduction of quadrature errors in the material point method (mpm). *International Journal for Numerical Methods in Engineering, accepted for publication*, 2000.

[34] N. Sukumar, N. Mos, B. Moran, and T. Belytschko. Extended finite element method for three-dimensional crack modeling. *International Journal for Numerical Methods in Engineering*, 48:1549–1570, 2000.

[35] D. L. Sulsky, Z. Chen, and H. L. Schreyer. A particle method for history dependent materials. *Computer Methods in Applied Mechanics and Engineering*, 118:179–196, 1994.

[36] D. L. Sulsky and A. Kaul. Implicit dynamics in the material-point method. *Computer Methods in Applied Mechanics and Engineering*, 193:1137–1170, 2004.

[37] D. L. Sulsky, H. L. Schreyer, K. Peterson, R. Kwok, and M. Coon. Using the material point method to model sea ice dynamics. *Journal of Geophysical Research*, 112:doi:10.1029/2005JC003329, 2007.

[38] D. L. Sulsky, S. Zhou, and H. L. Schreyer. Application of a particle-in-cell method to solid mechanics. *Computer Physics Communications*, 87:236–252, 1995.

[39] V. A. Vshivkov. The approximation properties of the particles-in-cells method. *Computational Mathematics and Mathematical Physics*, 36:509–515, 1996.

[40] P. C. Wallstedt and J. E. Guilkey. Improved velocity projection for the material point method. *Computer Modeling in Engineering and Sciences*, 19:223–232, 2007.

[41] P. C. Wallstedt and J. E. Guilkey. An evaluation of explicit time integration schemes for use with the generalized interpolation material point method. *Journal of Computational Physics*, under review, 2008.

[42] A. R. York, D. L. Sulsky, and H. L. Schreyer. The material point method for simulation of thin membranes. *International Journal for Numerical Methods in Engineering*, 44:1429–1456, 1999.

[43] A. R. York, D. L. Sulsky, and H. L. Schreyer. Fluid-membrane interaction based on the material point method. *International Journal for Numerical Methods in Engineering*, 48:901–924, 2000.