An Evaluation of Explicit Time Integration Schemes For Use with the Generalized Interpolation Material Point Method

P. C. Wallstedt J. E. Guilkey *

Department of Mechanical Engineering, U of U, Salt Lake City, UT 84112

Abstract

The stability and accuracy of the Generalized Interpolation Material Point (GIMP) Method is measured directly through carefully-formulated manufactured solutions over wide ranges of CFL numbers and mesh sizes. The manufactured solutions are described in detail. The accuracy and stability of several time integration schemes are compared via numerical experiments. The effect of various treatment of particle "size" are also considered. The hypothesis that GIMP is most accurate when particles remain contiguous and non-overlapping is confirmed by comparing manufactured solutions with and without this property.

Key words: material point method, manufactured solutions, time integration, MPM, GIMP, MMS, PIC
PACS: 02.70.Ns, 02.70.Dh, 52.65.Rr, 07.05.Tp

1 Introduction

The Generalized Interpolation Material Point (GIMP) Method is a particle-in-cell method for solid mechanics applications, described by Bardenhagen and Kober (1), that is an extension of the Material Point Method (MPM) of Sulsky et al. (2). 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 (3); membranes and fluid-structure interaction by York, Sulsky and Schreyer (4; 5); implicit time integration by

* Corresponding Author
Email address: james.guilkey@utah.edu (J. E. Guilkey).

Guilkey and Weiss (6), as well as Sulsky and Kaul (7); conservation properties and plasticity by Love and Sulsky (8; 9); contact by Bardenhagen et al. (10); cracks and fracture by Nairn (11).

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, the family of GIMP methods, including MPM, 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, often referred to as a computational "scratchpad".

The standard GIMP implementation, in which particles are treated as blocks of material, as opposed to Dirac delta functions, does a great deal to reduce the errors and instabilies that potentially arise as particle distributions become disordered. Tracking of particle "corners", described by Ma et al. (12) offers a vehicle by which to improve the size estimates of GIMP particles. An enhanced scheme for projecting particle data to the grid was described by Wallstedt and Guilkey (13) which both reduces the error in this operation, and also provides more predictable behavior.

Despite the significant value of these works, they do little to explore certain fundamental questions regarding the accuracy and stability of MPM and GIMP. The work of Bardenhagen (3) in particular, considers energy conservation in the face of a particular choice of time integration strategy, but it does not consider how that choice affects accuracy or stability. Here, we seek to build on that work, by studying the accuracy and stability of the time integration strategies described in (3) as well as a centered difference scheme as described in (14).

This paper is organized as follows. We first present a brief overview of MPM and GIMP, followed by a description of the choices of time integration strategy. This includes a detailed exposition of the time evolution of a single vibrating particle, which illustrates the non-linear nature of the discrete equations. Next, we describe the vehicle by which we have studied the behavior of the strategies described here, namely, the Method of Manufactured Solutions (MMS) (15; 16; 17). Finally, we present results from a series of numerical experiments, and from these, draw conclusions about the efficacies of the various approaches.

2 Review of the Generalized Interpolation Material Point Method

The material point method (MPM) was described by Sulsky et al. (2; 22) as an extension to the FLIP (Fluid-Implicit Particle) method of Brackbill (18), which itself is an extension of the particle-in-cell (PIC) method of Harlow (19). Interestingly, the name "material point method" first appeared in the literature two years later in a description of an axisymmetric form of the method (20). In both FLIP and MPM, the basic idea is the same: objects are discretized into particles, or material points, each of which contains all state data for the small region of material that it represents. These particles are spatially Dirac delta functions, meaning that the material that each represents is assumed to exist at a single point in space, namely the position of the particle. A subset of the particle data, minimally mass and velocity, are projected onto a background grid that is usually, although not necessarily, Cartesian. This projection is accomplished using weighting functions, also known as shape functions or interpolation functions. These are typically, but not necessarily, linear, bilinear or trilinear in one, two and three dimensions, respectively.

More recently, Bardenhagen and Kober (1) generalized the development that gives rise to MPM, and suggested that MPM may be thought of as a subset of their "Generalized Interpolation Material Point" (GIMP) method. In the family of GIMP methods one chooses a characteristic function χ_p to represent the particles and a shape function S_i as a basis of support on the computational nodes. An effective shape function $\bar{S}_{ip}$ is found by the convolution of the χ_p and S_i which is written as:

$$\bar{S}_{ip}(\mathbf{x}_p) = \frac{1}{V_p} \int_{\Omega_p \cap \Omega} \chi_p(\mathbf{x} - \mathbf{x}_p) S_i(\mathbf{x}) \, \mathrm{d}\mathbf{x}. \tag{1}$$

While the user has significant latitude in choosing these two functions, in practice, the choice of S_i is usually given (in one-dimension) as,

$$S_i(x) = \begin{cases} 1 + (x - x_i)/h & -h < x - x_i \leq 0 \\ 1 - (x - x_i)/h & 0 < x - x_i \leq h \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

where x_i is the vertex location, and h is the cell width, assumed to be constant in this investigation, although this is not a general restriction on the method. Multi-dimensional versions are constructed by forming tensor products of the one-dimensional version in the orthogonal directions.

When the choice of characteristic function is the Dirac delta,

$$\chi_p(\mathbf{x}) = \delta(\mathbf{x} - \mathbf{x}_p) V_p, \tag{3}$$

where $\mathbf{x}_p$ is the particle position, and V_p is the particle volume, then traditional MPM is recovered. Typically, when an analyst indicates that they are "using GIMP" this implies use of the linear grid basis function given in Eq. 2 and a "top-hat" characteristic function, given by (in one-dimension),

$$\chi_p(x) = H(x - (x_p - l_p)) - H(x - (x_p + l_p)), \tag{4}$$

where $H(x)$ is the Heaviside function ($H(x) = 0$ if $x < 0$ and $H(x) = 1$ if $x \geq 0$) and l_p is the half-length of the particle. When the convolution indicated in Eq. 1 is carried out using the expressions in Eqns. 2 and 4, a closed form for the effective shape function can be written as:

$$S_i\left(x_p\right) = \begin{cases} \frac{(h+l_p+(x_p-x_i))^2}{4hl_p} & -h-l_p < x_p - x_i \leq -h+l_p \\ 1 + \frac{(x_p-x_i)}{h} & -h+l_p < x_p - x_i \leq -l_p \\ 1 - \frac{(x_p-x_i)^2+l_p^2}{2hl_p} & -l_p < x_p - x_i \leq l_p \\ 1 - \frac{(x_p-x_i)}{h} & l_p < x_p - x_i \leq h-l_p \\ \frac{(h+l_p-(x_p-x_i))^2}{4hl_p} & h-l_p < x_p - x_i \leq h+l_p \\ 0 & \text{otherwise,} \end{cases} \tag{5}$$

The gradient of the shape is:

$$\nabla S_i(x_p) = \begin{cases} \frac{h+l_p+(x_p-x_i)}{2hl_p} & -h-l_p < x_p - x_i \leq -h+l_p \\ \frac{1}{h} & -h+l_p < x_p - x_i \leq -l_p \\ -\frac{(x_p-x_i)}{hl_p} & -l_p < x_p - x_i \leq l_p \\ -\frac{1}{h} & l_p < x_p - x_i \leq h-l_p \\ -\frac{h+l_p-(x_p-x_i)}{2hl_p} & h-l_p < x_p - x_i \leq h+l_p \\ 0 & \text{otherwise,} \end{cases} \tag{6}$$

There is one further consideration in defining the effective shape function, and that is whether or not the size (length in 1-D) of the particle is kept fixed (denoted as "UGIMP" here) or is allowed to evolve due to material deformations ("Finite GIMP" or "Contiguous GIMP" in (1) and "cpGIMP" here). In one-dimensional simulations, evolution of the particle (half-)length is straightforward,

$$l_p^n = F_p^n l_p^0, \tag{7}$$

where F_p^n is the deformation gradient at time n. In multi-dimensional simulations, a similar approach can be used, assuming an initially rectangular or cuboid particle, to find the current particle shape. The difficulty arises in evaluating Eq. 1 for these general shapes. One approach, apparently effective, has been to create a cuboid that circumscribes the deformed particle shape (12). Alternatively, one can assume that the particle size remains constant (insofar as it applies to the effective shape function evaluations only). The error

that this assumption introduces was demonstrated in (1) and will be further explored below.

Regardless of the choice of particle characteristic function, the timestepping algorithm is an independent choice. In his paper exploring energy conservation error, Bardenhagen (3) considered two possibilities which he denoted USF for "update stress first" and USL for "update stress last". As the names imply, these refer to the point within a timestep at which the particle stress is computed. While the original publications describing MPM used the USL scheme, USF came into use because it had a particular practical advantage. Namely, the velocity field from which gradients are taken for use in computing the stress is smoothly varying, having just been projected to the grid via interpolation functions. This improved the robustness of the method. A general overview of an MPM (or GIMP) timestep is given here, advancing from time n to $n+1$, in which the USF/USL distinction is described where appropriate.

The algorithm begins by a projection of the particle mass and momentum to the grid to form nodal masses and velocities. If we adopt the shorthand that $S_{ip} = S_i(\mathbf{x}_p)$, these can be written as:

$$m_i = \sum_p S_{ip} m_p, \tag{8}$$

$$\mathbf{v}_i^n = \frac{\sum_p S_{ip} \mathbf{v}_p^n m_p}{m_i}. \tag{9}$$

If the USF option is chosen, then gradients of $\mathbf{v}_i^n$ are computed. These can be used to compute a strain increment for use in a hypoelastic constitutive model. Alternatively, a deformation gradient on the particle can be updated for use in a hyperelastic model. Additionally, the particle volume V_p is updated by multiplying the initial volume by the determinant of the deformation gradient.

$$\nabla \mathbf{v}_p = \sum_i \nabla S_{ip} \mathbf{v}_i^n, \tag{10}$$

$$\mathbf{F}_p^{n+1} = (1 + \nabla \mathbf{v}_p \Delta t)\, \mathbf{F}_p^n, \tag{11}$$

$$\boldsymbol{\sigma}_p^{n+1} = \boldsymbol{\sigma}(\mathbf{F}_p^{n+1}), \tag{12}$$

$$V_p^{n+1} = V_p^0 |\mathbf{F}_p^{n+1}|. \tag{13}$$

The internal force, $\mathbf{f}_i^{int}$, is computed at the nodes from the volume integral of the divergence of the particle stress. In the USF case, this is based on the stress and volume that were just computed, while in the USL case, they are the values computed at the end of the prior timestep. The time superscript is omitted here for generality,

$$\mathbf{f}_i^{int} = -\sum_p \nabla S_{ip} \cdot \boldsymbol{\sigma}_p V_p. \tag{14}$$

Body forces and tractions are lumped into an external force term denoted by $\mathbf{f}_i^{ext}$, and with this we can compute acceleration on the grid by:

$$\mathbf{a}_i = \frac{\mathbf{f}_i^{int} + \mathbf{f}_i^{ext}}{m_i}. \tag{15}$$

This acceleration is used to update the grid velocity:

$$\mathbf{v}_i^* = \mathbf{v}_i^n + \mathbf{a}_i \Delta t. \tag{16}$$

Material point positions and velocities are updated by:

$$\mathbf{x}_p^{n+1} = \mathbf{x}_p^n + \sum_i S_{ip} \mathbf{v}_i^* \Delta t, \tag{17}$$

$$\mathbf{v}_p^{n+1} = \mathbf{v}_p^n + \sum_i S_{ip} \mathbf{a}_i \Delta t, \tag{18}$$

Finally, if the USL algorithm is chosen, the velocity gradients at the particles are computed based on the v_i^* values:

$$\nabla \mathbf{v}_p = \sum_i \nabla S_{ip} \mathbf{v}_i^*, \tag{19}$$

and Eqns. (11-13) are evaluated here. An important alternate method for finding velocity gradients in MPM is discussed in section 4.

3 Development of Discrete Equations for a One-Particle System

In attempting to analyze the stability and accuracy characteristics of competing time integration schemes, we construct discrete equations for $\mathbf{x}_p^{n+1}, \mathbf{v}_p^{n+1}$ and $\mathbf{F}_p^{n+1}$ in terms of these same time n quantities. We choose nearly the simplest possible system, a single one-dimensional particle in a single computational cell, constrained on the left side. This is the same problem analyzed by Bardenhagen (3), and is depicted here in Fig. 1.

Following the steps outlined in Section 2, we begin by projecting the particle data to the computational nodes. Since this is a one-dimensional scenario, bold facing of the variables is omitted. Because the left node is constrained, we can neglect it and only consider the quantities on the right node, which will be denoted here by a subscript i. In this analysis, the standard linear shape functions are used, which can be simplified for the current system as follows:

$$S_{ip} = \frac{x_p^n}{h} \qquad x < h \tag{20}$$

Thus the lumped mass on the right node is:

$$m_i = m_p \left(\frac{x_p^n}{h} \right) \tag{21}$$

and the velocity is:

$$v_i = \frac{m_p v_p \left(\frac{x_p^n}{h} \right)}{m_i} = v_p \tag{22}$$

At this point, we will assume a USF formulation, and compute the new deformation gradient. This can be found via a recursion relation:

$$F_p^{n+1} = F_n^{n+1} F_p^n \tag{23}$$

where F_n^{n+1} is the incremental deformation gradient from time n to n+1. Equation 23 can be rewritten as:

$$F_p^{n+1} = \left(1 + \nabla v_p \Delta t\right) F_p^n \tag{24}$$

where the gradient of velocity on the particle, ∇v_p, is as given by Eq. 10. For the present case, this is:

$$\nabla v_p = \sum_p G_{ip} v_i = \frac{v_p^n}{h} \tag{25}$$

where $G_{ip} = \frac{1}{h}$ is the gradient of the linear shape function on the right node, and the above accounts for the fixed boundary condition on the left node. Thus, Eq. 24 can be written in terms of time n values:

$$F_p^{n+1} = \left(1 + \frac{v_p^n}{h} \Delta t\right) F_p^n \tag{26}$$

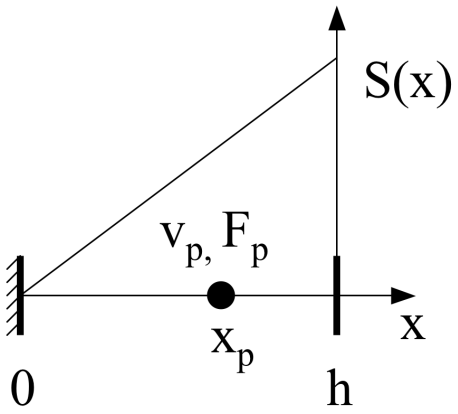


Fig. 1. Time n configuration of a single particle system. Linear shape function for the rightmost node is also depicted.

Moving forward in the timestep, the nodal force at the grid is the volume integral of the divergence of stress, computed as:

$$f_i^{int} = \sum_p G_{ip}\sigma_p^{n+1}V_p^{n+1} \tag{27}$$

where V_p^{n+1} is the current particle volume. In one dimension, $V_p^{n+1} = F_p^{n+1}V_p^0$. In addition, we have chosen the Neo-Hookean constitutive relation with zero Poisson's ratio:

$$\sigma = \frac{E}{2}\left(F - \frac{1}{F}\right), \tag{28}$$

where E is Young's modulus. Combining these gives:

$$f_i^{int} = \left(\frac{E}{2h}\right)\left(\left(F_p^{n+1}\right)^2 - 1\right)V_p^0 \tag{29}$$

Next, the nodal acceleration is the quotient of the internal force divided by the nodal mass, or Eq. 29 divided by Eq. 21. Simplification gives:

$$a_i = \frac{\frac{E}{2}\left(\left(F_p^{n+1}\right)^2 - 1\right)V_p^0}{m_p x_p^n} \tag{30}$$

We can now integrate the velocity at the node according to Eq. 16, using Eqns. 22, 26 and 30.

$$v_i^* = v_i^n + \frac{\frac{E}{2}\left(\left(\left(1 + \frac{v_p^n}{h}\Delta t\right)F_p^n\right)^2 - 1\right)V_p^0}{m_p x_p^n}\Delta t. \tag{31}$$

Finally, we can update the particle position and velocity according to Eqns. 17 and 18.

$$x_p^{n+1} = x_p^n + \frac{x_p^n}{h}\left(v_p^n + \frac{\frac{E}{2}\left(\left(\left(1 + \frac{v_p^n}{h}\Delta t\right)F_p^n\right)^2 - 1\right)V_p^0}{m_p x_p^n}\Delta t\right)\Delta t, \tag{32}$$

$$v_p^{n+1} = v_p^n + \frac{x_p^n}{h}\frac{\frac{E}{2}\left(\left(F_p^{n+1}\right)^2 - 1\right)V_p^0}{m_p x_p^n}\Delta t, \tag{33}$$

Simplifiying gives expressions for position, velocity and deformation gradient at time n+1 in terms of those quantities at time n for the USF approach.

$$x_p^{n+1} = x_p^n + v_p^n\left(\frac{x_p^n}{h}\right)\Delta t + \frac{E}{2h}\left(\left(\left(1 + \frac{v_p^n}{h}\Delta t\right)F_p^n\right)^2 - 1\right)\left(\frac{V_p^0}{m_p}\right)\Delta t^2 \tag{34}$$

$$v_p^{n+1} = v_p^n + \frac{E}{2h}\left(\left(\left(1 + \frac{v_p^n}{h}\Delta t\right)F_p^n\right)^2 - 1\right)\left(\frac{V_p^0}{m_p}\right)\Delta t \tag{35}$$

$$F_p^{n+1} = \left(1 + \frac{v_p^n}{h}\Delta t\right) F_p^n \tag{36}$$

By following the same procedure, similar expressions can be arrived at for the USL scheme. These are just stated here:

$$x_p^{n+1} = x_p^n + v_p^n \left(\frac{x_p^n}{h}\right) \Delta t + \frac{E}{2h}\left(\left(F_p^n\right)^2 - 1\right)\left(\frac{V_p^0}{m_p}\right)\Delta t^2 \tag{37}$$

$$v_p^{n+1} = v_p^n + \frac{E}{2h}\left(\left(F_p^n\right)^2 - 1\right)\left(\frac{V_p^0}{m_p}\right)\Delta t \tag{38}$$

$$F_p^{n+1} = \left(1 + \left(\frac{v_p^n}{h} + \frac{\frac{E}{2}\left(\left(F_p^n\right)^2 - 1\right)V_p^0}{m_p x_p^n h}\Delta t\right)\Delta t\right) F_p^n \tag{39}$$

The conclusion of this development is that, even for the simplest possible simulation, the resulting discrete equations are non-linear in several variables. As such, classical stability analysis is not feasible. For this reason, we have turned to the Method of Manufactured Solutions to generate exact solutions for non-linear problems. By comparing algorithmic performance against these, we can characterize the efficacy of the various approaches. First, we consider other candidate schemes for time integration.

4 Centered-Difference Time Integration

A number of families of time integration schemes have been investigated for use with GIMP including Runge Kutta, Runge-Kutta-Nystrom, Adams-Bashforth-Moulton (ABM), and Predictor-Corrector Newmark methods. In the authors' experience, few of these methods have been able to achieve their formal orders of accuracy. For example, the Runge-Kutta family is stable but offers no additional accuracy while incurring significantly greater computational cost. Not only is GIMP used for highly discontinuous and nonlinear problems (for which the ABM family is ill-suited) but the spatial idiosyncracies of the method tend to overwhelm any improvement that a temporally high order method might offer.

Significant trial-and-error experience has shown that successful algorithms for GIMP are 1. made from low order versions of the given family of time integration schemes and 2. involve a mixture of explicit and implicit forms. The successful USL and USF methods update grid velocity or particle stress explicitly (based on the current time step) then update remaining variables based on the new time step.

Nonlinear finite element codes often use a staggered central difference (CD) scheme (21) and such an approach is used for MPM by Sulsky (14). For the sake of clarity the method is written out in full using the notation of section 2. In practice the CD scheme is exactly the same as USL but for one crucial difference: initialization of particle velocity to a negative half time step.

$$m_i = \sum_p S_{ip} m_p, \tag{40}$$

$$\mathbf{v}_i^{n-\frac{1}{2}} = \frac{\sum_p S_{ip} \mathbf{v}_p^{n-\frac{1}{2}} m_p}{m_i} \tag{41}$$

$$\boldsymbol{\sigma}_p^n = \boldsymbol{\sigma}(\mathbf{F}_p^n) \tag{42}$$

$$V_p^n = V_p^0 |\mathbf{F}_p^n| \tag{43}$$

$$\mathbf{f}_i^{int} = -\sum_p \nabla S_{ip} \cdot \boldsymbol{\sigma}_p V_p \tag{44}$$

$$\mathbf{a}_i = \frac{\mathbf{f}_i^{int} + \mathbf{f}_i^{ext}}{m_i} \tag{45}$$

$$\mathbf{v}_p^{n+\frac{1}{2}} = \mathbf{v}_p^{n-\frac{1}{2}} + \sum_i S_{ip} \mathbf{a}_i \Delta t \tag{46}$$

$$\mathbf{v}_i^{n+\frac{1}{2}} = \mathbf{v}_i^{n-\frac{1}{2}} + \mathbf{a}_i \Delta t \tag{47}$$

$$\nabla \mathbf{v}_p^{n+\frac{1}{2}} = \sum_i \nabla S_{ip} \mathbf{v}_i^{n+\frac{1}{2}} \tag{48}$$

$$\mathbf{x}_p^{n+1} = \mathbf{x}_p^n + \sum_i S_{ip} \mathbf{v}_i^{n+\frac{1}{2}} \Delta t \tag{49}$$

$$\mathbf{F}_p^{n+1} = \left(1 + \nabla \mathbf{v}_p^{n+\frac{1}{2}} \Delta t\right) \mathbf{F}_p^n \tag{50}$$

For MPM as described in (22), an extra integration step is performed on the updated particle velocities to enhance stability. Equation 47 is replaced by:

$$\mathbf{v}_i^{n+\frac{1}{2}} = \frac{\sum_p S_{ip} \mathbf{v}_p^{n+\frac{1}{2}} m_p}{m_i} \tag{51}$$

We designate this modification to central difference time integration as "Update Velocity First", or "UVF-MPM" in the subsequent results section.

The negative 1/2 step data are available for known solutions, but for typical simulations they may be impossible to find. An easier approach that works nearly as well, and is used for all of the cases in this paper, is to multiply the grid acceleration values by 1/2 for the first time step only. This propagates the 1/2 through the algorithm correctly and fixes the first order error that would otherwise be incurred.

Although a number of additional variations of explicit time integration algorithms have been investigated (11) we limit our analysis to the methods

discussed here that appear to be in widest use. Additionally, by providing details of the manufactured solutions, a framework exists by which interested readers may test their own variations

5 Method of Manufactured Solutions

Code verification has gained importance in recent decades as costly projects rely more heavily on computer simulations. The Method of Manufactured Solutions (MMS) (15; 16; 17) 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 nonlinearities for which exact solutions will never be known. It is argued (15) that MMS is sufficient to verify a code, not merely necessary.

For this paper we define two non-linear large deformation dynamic manufactured solutions, and use both of them for subsequent testing. The two solutions exercize 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. Both forms can be used successfully in a FEM algorithm, and solutions from both forms are equivalent (21). 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:

$$\nabla \mathbf{P} + \rho_0 \mathbf{b} = \rho_0 \mathbf{a} \tag{52}$$

$$\nabla \boldsymbol{\sigma} + \rho \mathbf{b} = \rho \mathbf{a} \tag{53}$$

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 Lan-

grangean forms, respectively:

$$\mathbf{P} = \lambda ln J \mathbf{F}^{-1} + \mu \mathbf{F}^{-1} \left(\mathbf{F}\mathbf{F}^T - \mathbf{I} \right) \tag{54}$$

$$\boldsymbol{\sigma} = \frac{\lambda ln J}{J} \mathbf{I} + \frac{\mu}{J} \left(\mathbf{F}\mathbf{F}^T - \mathbf{I} \right) \tag{55}$$

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 the 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.

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

5.1 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 cpGIMP particles are coincident and colinear. 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 (1). While it is not representative of general material deformations, it does allow characterization of the error introduced via the inexact approximations to cpGIMP, e.g., use of a constant sized particle characteristic function.

The plane strain displacement field is chosen to be:

$$\mathbf{u} = \begin{pmatrix} Asin(\pi X)cos(c\pi t) \\ Asin(\pi Y)sin(c\pi t) \\ 0 \end{pmatrix} \tag{56}$$

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.

The deformation gradient is found by taking derivatives with respect to posi-

tion:

$$\mathbf{F} = \begin{pmatrix} 1 + A\pi cos(\pi X)cos(c\pi t) & 0 & 0 \\ 0 & 1 + A\pi cos(\pi Y)sin(c\pi t) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{57}$$

The stress is found by substituting Eq. 57 into 54:

$$\mathbf{P} = \begin{pmatrix} \frac{\lambda}{F_{XX}}K + \frac{\mu}{F_{XX}}(F_{XX}^2 - 1) & 0 & 0 \\ 0 & \frac{\lambda}{F_{YY}}K + \frac{\mu}{F_{YY}}(F_{YY}^2 - 1) & 0 \\ 0 & 0 & \lambda K \end{pmatrix} \tag{58}$$

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

Acceleration is found by twice differentiating displacement Eq. 56 in time. Finally, substituting stress $\mathbf{P}$ into Eq. 52 and solving for the body force $\mathbf{b}$ (used as the MMS source term) it is found that:

$$\mathbf{b} = \begin{pmatrix} \frac{\pi^2 u_X}{\rho_0}\left[\frac{\lambda}{F_{XX}^2}(1-K) + \mu\left(1 + \frac{1}{F_{XX}^2}\right) - E\right] \\ \frac{\pi^2 u_Y}{\rho_0}\left[\frac{\lambda}{F_{YY}^2}(1-K) + \mu\left(1 + \frac{1}{F_{YY}^2}\right) - E\right] \\ 0 \end{pmatrix}. \tag{59}$$

5.2 Radial Expansion of a Ring

Displacement is prescribed with radial symmetry for a ring as

$$u(R) = Acos(c\pi t)(c_3R^3 + c_2R^2 + c_1R) \tag{60}$$

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. 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 $u(R_O) = A$:

$$c_3 = \frac{-2}{R_O^2(R_O - 3R_I)},\ c_2 = \frac{3(R_O + R_I)}{R_O^2(R_O - 3R_I)},\ c_1 = \frac{-6R_I}{R_O(R_O - 3R_I)} \tag{61}$$

The neo-Hookean constitutive model of Eq. 54 with zero Poisson's ratio is used to make the manufactured solution tractible. 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 $R^2 = X^2 + Y^2$:

$$\mathbf{u} = \begin{pmatrix} A cos(c\pi t)(c_3 R^3 + c_2 R^2 + c_1 R)\frac{X}{R} \\ A cos(c\pi t)(c_3 R^3 + c_2 R^2 + c_1 R)\frac{Y}{R} \\ 0 \end{pmatrix} \tag{62}$$

Equations for velocity, acceleration, and deformation gradient are straightforward to find by differentiating Eq. 62 with respect to time and position. It is more difficult to solve Eq. 54 (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. The Maple commands that we used to do this are presented along with a brief description in Appendix B. This should allow the reader to reproduce these results for testing their own codes.

6 Numerical Results

As discussed above, GIMP does not lend itself to linear stability analysis and no analytical method has been seen by the authors for predicting the behavior of the time integration schemes in this paper when used with GIMP. In lieu of analysis, a series of numerical experiments are performed on cases chosen for their generality and applicability. A representative sampling of results obtained is presented here.

In order to save on computational effort various reduced forms of GIMP are sometimes used as shape functions. For cases denoted below as using cpGIMP, particle extents in each direction are approximated according to a three-dimensional version of Eq. 7:

$$\mathbf{l}_p^n = \mathbf{l}_p^0 diag(\mathbf{F}_p^n) \tag{63}$$

Note that only for the axis-aligned displacement problem does this result in the deformed particles filling the spatial domain exactly without any gaps or overlaps. For UGIMP the particle lengths are not changed: $\mathbf{l}_p^n = \mathbf{l}_p^0$, while for MPM $\mathbf{l}_p = 0$.

The definition of error 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 at each particle δ_p as the norm of the difference in computed displacement relative to the exact displacement:

$$\delta_p = ||(\mathbf{x}_p - \mathbf{X}_p) - \mathbf{u}_{exact}(\mathbf{X}_p, t)||. \tag{64}$$

The definition for error at a node is more difficult; see Appendix A.

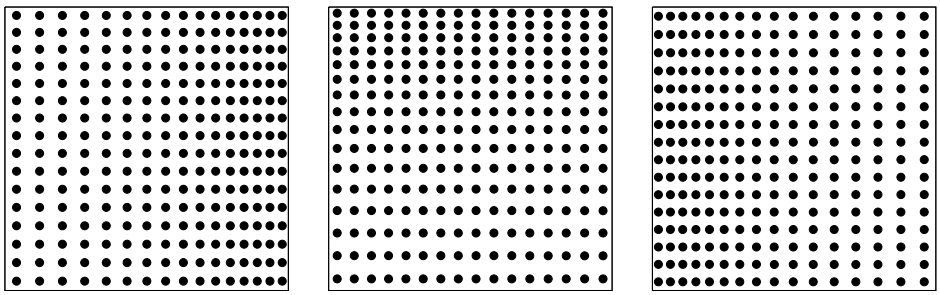
For the results of this paper we define a single pessimistic, but trustworthy, measure of error for a complete solution as the L_∞ norm over all particles and all time steps:

$$L_\infty = max(\delta_p). \tag{65}$$

During the computation of results we also calculated the L_1 and L_2 norms for all scenarios. However, we found that they indicated the same orders of accuracy as Eq. 65. For problems that are smooth in space and time, such as those used here, we prefer the L_∞ norm because it assures us that all particles are converging. However, for problems involving discontinuities such as contact or shocks, the L_1 and L_2 norms are appropriate.

6.1 Axis-Aligned Displacement

A series of simulations are carried out to measure temporal and spatial convergence. In each, the reference configuration contains four equally-spaced particles per cell. Example results for particle displacement from one such simulation, using a coarse 8x8 grid, are depicted in Fig. 2.



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

Fig. 2. Illustration of particle displacements at three representative times.

6.1.1 Temporal Convergence for Axis-Aligned Displacement

The temporal convergence behavior is examined by measuring the error in displacement, as defined by Eq. 65, of computed solutions over a range of CFL

numbers. All solutions use 56^2 cells and the maximum magnitude of displacement $A = 0.1$ (from Eq. 56) is large enough to cause the majority of particles to experience several cell crossings per period. The test code is configured so that an error of one is returned whenever a particular solution crashes. Results for several interesting combinations of time integration algorithm and shape function are plotted in Fig. 3. Below, we consider the results for each.

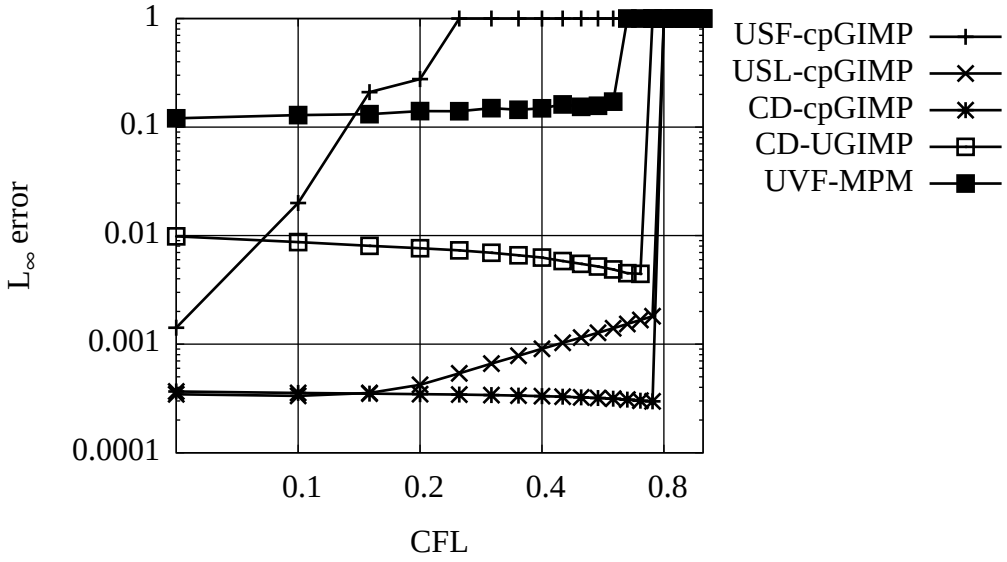


Fig. 3. Temporal Convergence for Axis-Aligned Displacement

The USF-cpGIMP combination displays uninspiring behavior for the Axis-Aligned problem. Although it completes the solution successfully for low CFL cases, its accuracy is poor and gets dramatically worse with increasing values of CFL. Although Bardenhagen (3) showed that USF conserves energy better for infinitesimal linear elastic problems, he did not recommend using USF and the results found here indicate that its lack of dissipation becomes problematic for non-linear large deformations. The algorithm becomes progressively less stable and imperfections in the solution are preserved and amplified. However, it must be noted that USF-MPM performs better than USL-MPM (neither of which are shown here, and neither of which performs better than the UVF-MPM in this simulation) because the velocity gradients are based on smooth values of velocity that have not yet been updated by potentially inaccurate grid accelerations. The smoother spatial gradients of GIMP enable the use of more delicate time integration schemes.

Use of the USL-cpGIMP combination results in a dramatic improvement over USF-cpGIMP with a reduction of error of up to three orders of magnitude. Accuracy is preserved over a wide range of CFL conditions and the algorithm does not crash until CFL > 0.7 or thereabouts. We draw attention to the curious "elbow" that is observed at about CFL $= 0.2$ wherein the convergence rate changes from zero to one. Analysis of subsequent spatial convergence

results will suggest that a close link between spatial and temporal phenomena in GIMP causes the elbow. We believe the elbow indicates a shift of dominant error in USL-cpGIMP from spatial to temporal.

A minor modification to USL changes the algorithm to centered-difference (CD) (see Section 4) and eliminates the elbow. The CD-cpGIMP combination displays the same accuracy regardless of CFL right up until CFL exceeds the stable limit of roughly 0.7. By improving the time integration algorithm from USL to CD, temporal convergence is entirely eliminated. This is evidence that spatial error dominates the CD-cpGIMP combination, so temporal effects are not observed. Experience with a number of problems indicates that the CD-cpGIMP combination is the best all-around method; it is used as the benchmark throughout this paper.

The small algorithmic difference between cpGIMP and UGIMP has a substantial effect on accuracy as shown in the CD-UGIMP trend of Fig. 3. An order of magnitude in accuracy is lost by using UGIMP, but other traits of stability and general good behavior are retained. The UGIMP results are likely more representative of real world behavior, as general deformations do not retain the rectangular shape of the particles needed to fill the deformed domain exactly.

Lastly, we observe that the UVF-MPM of Sulsky et al. (22) is able to complete the solution over a range of CFL conditions but accuracy is an order of magnitude worse than CD-cpGIMP. It has been our experience that MPM produces poor quantitative convergence when high stress and large deformation occur together, but can perform acceptably under less demanding conditions.

6.1.2 Spatial Convergence for Axis-Aligned Displacement

The spatial convergence behavior is found by measuring the displacement error, as defined by Eq. 65, of computed solutions over a range of mesh sizes. All solutions use CFL $= 0.4$ and $A = 0.1$. Convergence results are plotted in Fig. 4.

The USF-cpGIMP and UVF-MPM combinations display similar trends of spatial accuracy despite being based on different time and spatial integration approaches. Both 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 plausible even when their fundamental accuracy falters.

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

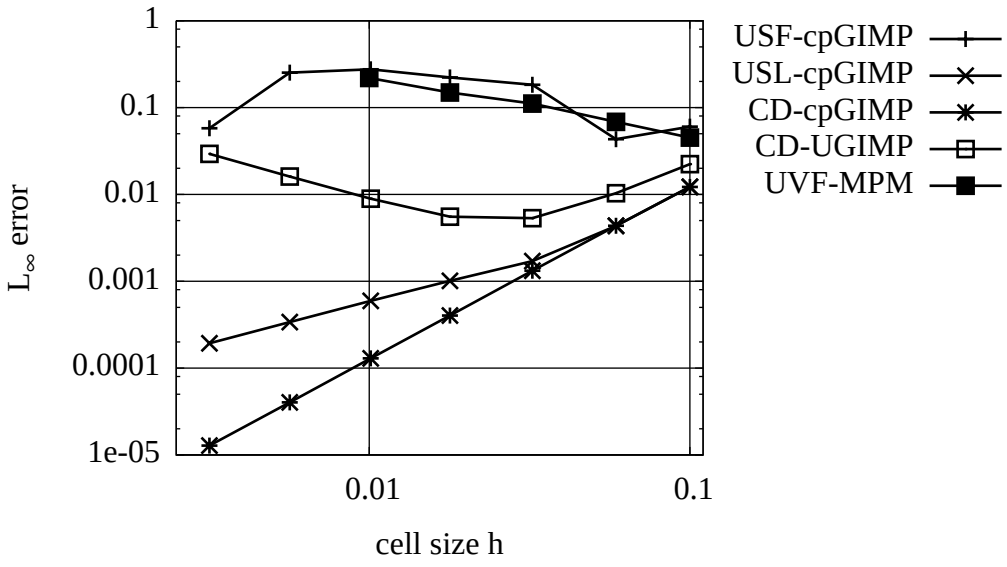


Fig. 4. Spatial Convergence for Axis-Aligned Displacement

to first order for finer meshes. We continue to suppose that this is due to the close coupling of spatial and temporal effects.

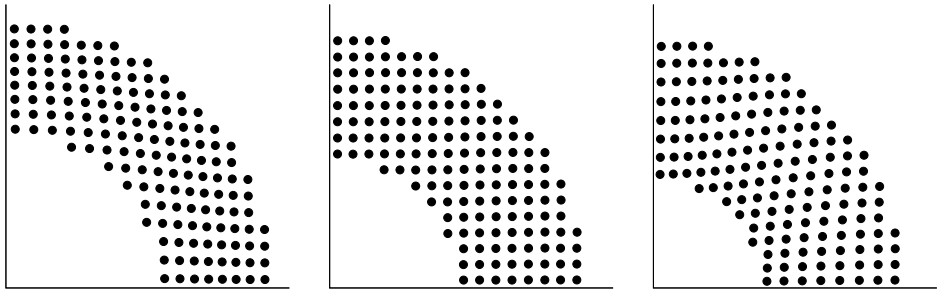
Finally, the CD-cpGIMP combination is satisfyingly second order in space. We are reminded that this excellent behavior is only displayed for the special circumstances of the axis-aligned problem where no gaps or overlaps exist in the cell integration. The behavior of a more realistic problem is assessed in the next section.

6.2 Expanding Ring

A series of simulations, each with four equally-spaced particles per cell, are carried out to measure temporal and spatial convergence. Example results for particle displacement from one such simulation, using a coarse 8x8 grid, are depicted in Fig. 5.

6.2.1 Temporal Convergence for Expanding Ring

The temporal convergence behavior is found by measuring the error in displacement, as defined by Eq. 65, of computed solutions over a range of CFL numbers. All solutions use 56^2 cells and $A = 0.1$ from Eq. 62. The curved surfaces of the ring are "stair-stepped" approximations in the particle representation. Temporal convergence results are generated for the expanding ring in Fig. 6.



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

Fig. 5. Illustration of particle displacements at three representative times.

Temporal convergence trends are somewhat different for the expanding ring as compared to the axis-aligned problem. UGIMP performs 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 causes inaccuracies in the spatial integration due to non axis-aligned displacements in the ring.

While UVF-MPM appears to be the most stable algorithm, it is also significantly less accurate than the GIMP variations. Lastly we note that USF displayed very poor performance and for this reason is omitted from the results shown in the next section for spatial convergence.

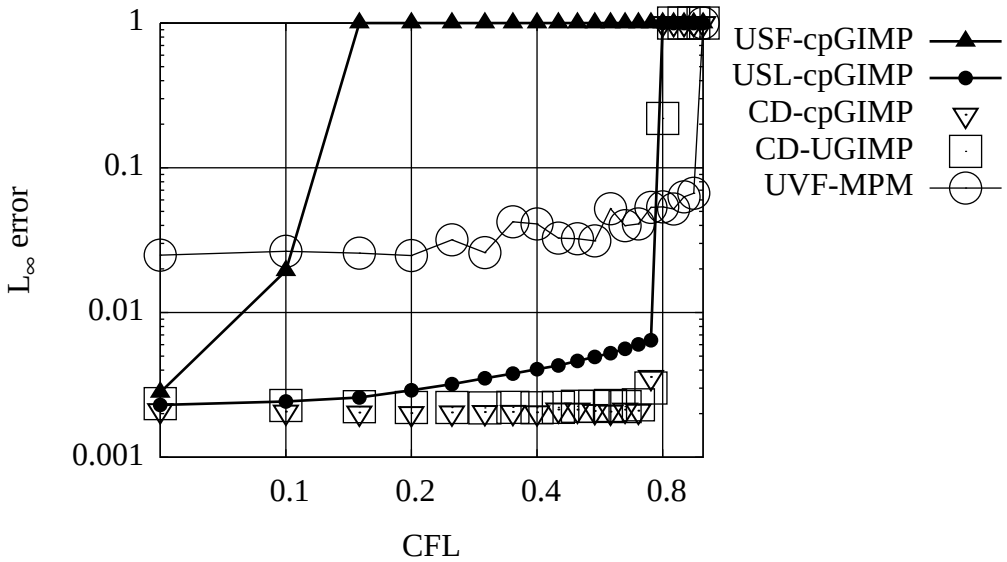


Fig. 6. Temporal Convergence for Expanding Ring

6.2.2 Spatial Convergence for Expanding Ring

Spatial convergence behavior, as presented in Fig. 7, is found by measuring the error, as defined by Eq. 65, of computed solutions over a range of mesh sizes. All solutions use four initially equally-spaced particles per cell with CFL = 0.4 and $A = 0.1$.

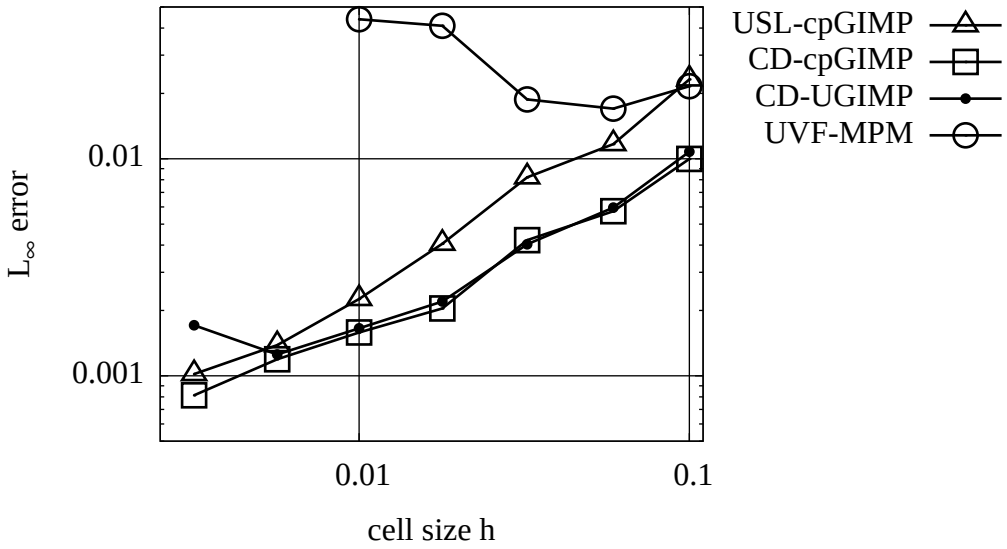


Fig. 7. Spatial Convergence for Expanding Ring

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 last point of the trend. UVF-MPM provides visually satisfactory solutions but it fails to converge.

The second order effects, seen in the axis-aligned problem, that differentiate USL 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 full cpGIMP treatment of the particle sizes.

7 Conclusions

As demonstrated in Section 3 a formal stability analysis for MPM (or GIMP) is difficult, if not impossible, due to the non-linear nature of the equations governing the advancement in time of position, velocity and deformation gradient. In lieu of formal analysis, the Method of Manufactured Solutions provides a suitable platform for testing stability and convergence behavior. MMS was used here to determine this behavior for a selection of time integration schemes suitable for use with GIMP. In addition, MMS allows investigation of non-linear, large deformation simulations for which GIMP is frequently employed.

Centered difference (CD) time integration was shown to be closely related to the "Update Stress Last" (USL) scheme, both of which performed significantly better than the "Update Stress First" (USF) scheme when used with cpGIMP. Evidence of the superiority of the CD and USL schemes is most apparent in light of stability and spatial convergence behavior. However, none of these schemes were able to achieve their formal orders of accuracy for the simulations considered here. (Note that for small deformation versions of these cases, convergence is significantly better.) This is evidently due to the comparatively large amount of spatial error inherent in GIMP, even for the manufactured solution for which GIMP is ideally suited.

Via the axis-aligned displacement problem we show that the UGIMP approximation, in which particle sizes are assumed to remain constant, eventually causes the accuracy to diverge with mesh refinement. Note that the strategy of Ma, et al., (12) for evolving particle shapes was not investigated here. Cursory testing of this idea showed some promise, but tracking the particle corners with sufficient accuracy has proved challenging in general simulations. Specifically, near the edges of objects, particle corners will eventually migrate into cells, some of the nodes of which don't have well defined velocities. In our experience, this often leads those corners into non-physical territory. Presumably, a solution to this difficulty exists, but our efforts in this arena have been minimal.

While choice of time integration schemes has a large impact on the overall accuracy of a simulation, the ultimate conclusion of this work is that, when the best of these choices is made, spatial error remains dominant. As such, future work will concentrate on identification and reduction of specific spatial error sources.

8 Acknowledgements

The authors gratefully acknowledge valuable discussions with Mike Steffen, Mike Kirby, and Martin Berzins, as well as input from Comer Duncan. This work was supported by the National Science Foundation's Information Technology Research Program under grant CTS0218574, and the U.S. Department of Energy through the Center for the Simulation of Accidental Fires and Explosions, under grant W-7405-ENG-48.

A Nodal Error in the Current Configuration

The definition for error at a node is complicated by the fact that the nodes of the computational "scratch pad" are stationary with respect to the current configuration, but move with respect to the reference configuration. The reference position of a node is found by solving the following implicit equation for $\mathbf{X}_i$ using a simple root-finding subroutine:

$$\mathbf{u}_{exact}(\mathbf{X}_i, t) = \mathbf{x}_i - \mathbf{X}_i \tag{A.1}$$

where $\mathbf{x}_i$ is the known position of the node $_i$. Then the error in acceleration on a node, for example, could be defined as:

$$\delta_i = ||\mathbf{a}_i - \mathbf{a}_{exact}(\mathbf{X}_i, t)||. \tag{A.2}$$

B Example Maple Commands for Ring Body Force

The following Maple commands allow one to generate the body force for the ring problem in Sec. 5.2, and should be generally useful in developing other manufactured solutions.

```
with(linalg);
```

Displacement is defined to be radially symmetric; `R` is radius in the reference configuration and `T` is a placeholder for some function of time, which is later assumed to be trigonometric:

```
u:=T*(c3*R^3+c2*R^2+c1*R);
```

We first set out to find expressions for the coefficients $c_1 - c_3$. Displacement in cartesian coordinates in terms of radius `R` and angle `H` can be written:

```
uc:=<u*cos(H),u*sin(H),0>;
```

Displacement gradient with respect to cartesian coordinates in terms of `R` and `H` via the chain rule:

```
Gu:=evalm(matrix([
   [diff(uc[1],R)*cos(H)-diff(uc[1],H)*sin(H)/R,
    diff(uc[1],R)*sin(H)+diff(uc[1],H)*cos(H)/R,0],
   [diff(uc[2],R)*cos(H)-diff(uc[2],H)*sin(H)/R,
    diff(uc[2],R)*sin(H)+diff(uc[2],H)*cos(H)/R,0],
   [0,0,0]]));
```

Forming the deformation gradient:

```
I3:=matrix([[1,0,0],[0,1,0],[0,0,1]]);
F:=simplify(evalm(I3+Gu));
```

Evaluate stress assuming zero Poisson's ratio. At this point the stress matrix, if written densely, fills nearly a page and is difficult to handle. Symbolic math manipulation software is extremely helpful:

```
P:=simplify(evalm(E/2*inverse(F)&*(F&*transpose(F)-I3)));
```

We find the constants of Eq. 61 by rotating the stress to an arbitrary angle `H`, which leaves just two diagonal components in the stress tensor: normal stress and hoop stress.

```
Q:=matrix([[ cos(H),sin(H),0],
           [-sin(H),cos(H),0],
           [0,0,1]]);
PQ:=simplify(evalm(Q&*P&*transpose(Q)));
```

We set the normal stress at the inner and outer radii to zero and scale displacement at the outer radius. The software finds constants that satisfy these conditions. However, if Poisson's ratio is not zero then closed forms for the constants cannot be found.

```
Pb:=unapply(PQ[1,1],R);
ub:=unapply(u,R);
assume(Ri>0);
assume(Ro>Ri);
interface(showassumed=2);
solve({Pb(Ro)=0,Pb(Ri)=0,ub(Ro)=T},{c1,c2,c3});
```

With the coefficients of the displacement equation in hand, we find the divergence of stress in cartesian coordinates in terms of `R` and `H` via the same chain

rule operator used above:

```
dP:=simplify(<diff(P[1,1],R)*cos(H)-diff(P[1,1],H)*sin(H)/R
             +diff(P[2,1],R)*sin(H)+diff(P[2,1],H)*cos(H)/R,
              diff(P[1,2],R)*cos(H)-diff(P[1,2],H)*sin(H)/R
             +diff(P[2,2],R)*sin(H)+diff(P[2,2],H)*cos(H)/R,
              0>);
```

Finally we solve the momentum equation for `b` (assuming that `T` is of the form used in Eq. 60) and generate optimized C-style code (about a page and a half of it) that can be used to check our implementation of GIMP.

```
b:=evalm(-pi^2*E/rho*uc-1/rho*dP);
with(codegen,C):
C(b,optimized,mode=double);
```

References

[1] S. Bardenhagen, E. Kober, The generalized interpolation material point method, Computer Modeling in Engineering and Sciences 5 (2004) 477–495.

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

[3] S. Bardenhagen, Energy conservation error in the material point method for solid mechanics, Journal of Computational Physics 180 (2002) 383–403.

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

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

[6] J. E. Guilkey, 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 (2003) 1323–1338.

[7] D. Sulsky, A. Kaul, Implicit dynamics in the material-point method, Computer Methods in Applied Mechanics and Engineering 193 (2004) 1137–1170.

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

[9] E. Love, D. L. Sulsky, An energy-consistent material-point method for

dynamic finite deformation plasticity, International Journal for Numerical Methods in Engineering 65 (2005) 1608–1638.

[10] S. Bardenhagen, J. Guilkey, K. Roessig, J. Brackbill, W. Witzel, J. 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 (2001) 509–522.

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

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

[13] P. Wallstedt, J. Guilkey, Improved velocity projection for the material point method, Computer Modeling in Engineering and Sciences 19 (2007) 223–232.

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

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

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

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

[18] J. Brackbill, H. Ruppel, Flip: A low-dissipation, particle-in-cell method for fluid flows in two dimensions, J. Comp. Phys. 65 (1986) 314–343.

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

[20] D. Sulsky, H. Schreyer, Axisymmetric form of the material point method with applications to upsetting and taylor impact problems, Computer Methods in Applied Mechanics and Engineering 139 (1996) 409–429.

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

[22] D. Sulsky, S. Zhou, H. Schreyer, Application of a particle-in-cell method to solid mechanics, Computer Physics Communications 87 (1995) 236–252.