INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING
Int. J. Numer. Meth. Engng 2010; **82**:1207–1243
Published online 9 December 2009 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/nme.2787

Decoupling and balancing of space and time errors in the material point method (MPM)

Michael Steffen, Robert M. Kirby*, † and Martin Berzins

School of Computing and Scientific Computing and Imaging Institute, University of Utah, Salt Lake City, UT 84112, U.S.A.

SUMMARY

The material point method (MPM) is a computationally effective particle method with mathematical roots in both particle-in-cell and finite element-type methods. The method has proven to be extremely useful in solving solid mechanics problems involving large deformations and/or fragmentation of structures, problem domains that are sometimes problematic for finite element-type methods. Recently, the MPM community has focused significant attention on understanding the basic mathematical error properties of the method.

Complementary to this thrust, in this paper we show how spatial and temporal errors are typically coupled within the MPM framework. In an attempt to overcome the challenge to analysis that this coupling poses, we take advantage of MPM's connection to finite element methods by developing a 'moving-mesh' variant of MPM that allows us to use finite element-type error analysis to demonstrate and understand the spatial and temporal error behaviors of MPM. We then provide an analysis and demonstration of various spatial and temporal errors in MPM and in simplified MPM-type simulations.

Our analysis allows us to anticipate the global error behavior in MPM-type methods and allows us to estimate the time-step where spatial and temporal errors are balanced. Larger time-steps result in solutions dominated by temporal errors and show second-order temporal error convergence. Smaller time-steps result in solutions dominated by spatial errors, and hence temporal refinement produces no appreciative change in the solution. Based upon our understanding of MPM from both analysis and numerical experimentation, we are able to provide to MPM practitioners a collection of guidelines to be used in the selection of simulation parameters that respect the interplay between spatial (grid) resolution, number of particles and time-step. Copyright © 2009 John Wiley & Sons, Ltd.

Received 10 July 2009; Revised 27 September 2009; Accepted 30 September 2009

KEY WORDS: material point method; meshfree methods; meshless methods; particle methods; smoothed particle hydrodynamics; quadrature; time stepping; nodal quadrature

*Correspondence to: Robert M. Kirby, School of Computing and Scientific Computing and Imaging Institute, University of Utah, 50 S. Central Campus Drive, Salt Lake City, UT 84112, U.S.A.
†E-mail: kirby@cs.utah.edu

Contract/grant sponsor: U.S. Department of Energy through the Center for the Simulation of Accidental Fires and Explosions (C-SAFE); contract/grant number: W-7405-ENG-48

1. INTRODUCTION

The material point method (MPM) [1, 2] is a mixed Lagrangian and Eulerian method utilizing a collection of Lagrangian particles to discretize a material and an Eulerian background mesh on which to calculate derivatives and solve equations of motion. MPM has proven to be extremely successful in simulating high-deformation and otherwise complicated engineering problems such as densification of foam [3], compression of wood [4], sea ice dynamics [5], and energetic device explosions [6], to name a few.

While these simulations are impressive and have pushed the boundaries of high-deformation simulation science where other finite element methods often fail, there has been a relative lack of basic error analysis of the method. For example, time-stepping algorithms within the method have received little attention. While the centered difference time-stepping scheme often used for advancing velocities and displacements is well explained within the ODE literature, the complicated interconnection between spatial and temporal errors in MPM makes quantifying the error behavior more complex. In particular, the motivation of this paper is to reconcile through analysis and numerical experimentation statements that the time-stepping method used in MPM is 'formally second-order' [5] with the recent and detailed convergence tests showing 'zero-order' temporal convergence [7].

In this paper we give a detailed explanation of both standard MPM and a variant of MPM to which we refer to as 'moving-mesh MPM' and provide an analysis and demonstration of the spatial and temporal errors of the method. Moving-mesh MPM is a fully Lagrangian method that helps to control some of the more complicated sources of errors within MPM—quadrature and grid crossing errors—thereby allowing us to construct computational experiments that help to ferret out the mathematical and algorithmic choices within MPM which violate the mathematical assumptions upon which time-stepping algorithms are based. A simplified non-physical mathematical problem with similar error characteristics as MPM helps us to both analyze and demonstrate the expected error behaviors in MPM-type simulations.

We then extend this work to provide intuition and guidelines by which the MPM practitioner can select time-step sizes that balance space and time errors. In particular, we help the practitioner to understand the trade-offs between increasing the spatial resolution through increasing grid spacing and the number of particles and the corresponding impact on temporal errors. In the case in which explicit time-stepping algorithms are used (as is often the case in the MPM community and as is analyzed in this paper), the practitioner can also further appreciate the trade-offs the between temporal accuracy and stability as dictated by their time-step choice.

This paper is organized as follows. Section 2 gives a brief historical background to provide the context as to where and how MPM fits into the family of particle methods. Previous results of MPM error analysis and demonstrations are reviewed, focusing on previous analysis of spatial and temporal error behaviors. Section 3 provides an overview of the MPM method, beginning with a review of how MPM comes about through a collection of approximations and assumptions injected into the standard Galerkin approximation process applied to the equations of motion. With this algorithmic background in place, moving-mesh MPM is then fully described. Section 4 provides an explanation of the coupling of spatial and temporal errors within MPM. Section 5 provides three studies of various error behaviors for both a simplified non-physical problem with MPM-type characteristics, and a single step standard and moving-mesh MPM. Section 6 gives a demonstration of the errors analyzed in Section 5, this time in the full MPM

DOI: 10.1002/nme

framework. Section 7 provides some guidelines to the practitioner on how algorithm parameters affect various errors. And finally, Section 8 is a summary of our findings, our conclusions, and future work.

2. BACKGROUND

MPM is a mixed Lagrangian–Eulerian method with moving particles that store history-dependent variables and a fixed background grid used for calculating derivatives and for solving equations of motion. MPM [1, 2] descends from a long line of particle-in-cell (PIC) methods, specifically as a solid mechanics extension to the 'full particle' formulation of PIC called FLIP [8, 9]. More recently, the generalized interpolation material point method (GIMP) [10] was developed as a generalization of MPM where particles are represented by particle-characteristic functions, of which the Dirac delta function $\delta(x-x_p)$ results in the original MPM method. These methods share the same general framework—namely that the solution of the equations of motion can be accomplished through a discretization of the solution domain with a set of particles, projection of particle information to a background mesh, solving of the equations of motion on the background mesh, and then by using the mesh solution to both move and update the particles' history-dependent variables.

As was previously stated, these algorithms have enabled, from the engineering perspective, complicated large deformation simulations where finite element-type methods often fail due to numerical issues such as mesh-entanglement. While the broad applicability and robustness of these methods have been used to encourage their adoption within the engineering community, the final critique consisting of a detailed understanding of the basic error properties of the method is just beginning to form.

As an example of such a critique, the recently published work has focused on understanding the impact of quadrature choices within the MPM framework. It is well acknowledged that within almost all numerical methods, the accuracy of the method can depend highly on the accuracy of the numerical quadrature used. Recently, Steffen *et al.* [11] performed an analysis of the spatial quadrature errors in MPM, equating the quadrature errors in MPM to integration errors when using a composite midpoint rule with breaks in the continuity of the integrand. This analysis helped to explain why second-order spatial convergence, as one would expect in finite element methods, is not possible when piecewise-linear basis functions are used for representing field quantities on the Eulerian mesh within MPM. The simple adaptation to quadratic B-spline basis functions (also detailed in [12]) allowed the demonstration of second-order spatial convergence of full MPM simulations. Midpoint integration errors are also second-order and therefore, whereas higher-order basis functions may improve the overall error further, spatial convergence rates will not improve with the current integration strategies. For higher than second-order spatial convergence, more advanced techniques than nodal integration as currently employed would be required.

The analysis in [11, 12] assumed the use of a fixed background grid—a grid that 'resets' back to the starting position after each time-step. While particles may start in ideal positions with particle voxel boundaries aligned with grid cell boundaries, any motion will quickly lead to an arrangement where the particles overlap the grid cell boundaries. It is this overlap that leads to the largest quadrature errors. Another option is to use moving-mesh MPM, where the background mesh moves with the particles and is never reset. The particles will remain in their ideal positions, eliminating the errors associated with particle voxel and grid cell boundary overlap. While this technique may seem contrary to the spirit of MPM, it remains effective for small deformation problems

DOI: 10.1002/nme

and completely eliminates grid-crossing errors, allowing for simpler analysis and demonstration of temporal errors. Moving-mesh MPM has previously been used to model the biological mechanics of cells [13] and in studying the texture evolution in polychrystalline nickel [14].

Another example within the MPM literature of the trend to analyze the mathematical algorithmic properties of the choices made within the MPM framework is given by the work of Love and Sulsky [15, 16], in which the selection of the time-stepping algorithms employed within MPM were scrutinized. Love and Sulsky [15, 16] analyzed an energy consistent implementation of MPM, the second aspect of these papers showing an implicit implementation to be unconditionally stable and energy-momentum consistent. We note that their use of full consistent mass matrices and implicit time-integration strategies add considerable computational complexity to the original explicit algorithm, and hence are not often used in practice. Our study will remain focused on the more ubiquitous second-order scheme used in engineering practice.

A third example within the MPM literature of the aforementioned trend is the work of Bardenhagen [17] and subsequently Wallstedt and Guilkey [7] in which rigorous tests were performed comparing the various explicit time-stepping algorithms that have appeared in the literature. Specifically 'update stress first' (USF), 'update stress last' (USL), and centered difference (CD) methods were compared with USL and CD showing superiority with respect to the overall error magnitudes. While CD was shown to have the lowest error of the time-stepping methods in their tests, the method showed no temporal error convergence in the regions of time-step selection where their simulations were stable.

This paper seeks to use and extend the perspective on the spatial errors gained in [11, 12] to understand the lack of temporal convergence demonstrated in [7]. The inspiration for connecting the spatial error characteristics with the temporal error characteristics lies outside the MPM literature. Lawson *et al.* [18] demonstrate a method of error control in solving parabolic equations, and is the basis on which we formulate our analysis. In this work, we do not go as far as attempting to control errors in MPM, but as in the work of Lawson *et al.*, we model our time-update equation for an ODE of the form $\dot{v} = a$ as,

$$v^{k+1} = v^k + (a^k + c_1 h^p)\Delta t + c_2 \Delta t^q \tag{1}$$

where the spatial errors in a are assumed to be $\mathcal{O}(h^p)$ and the time-stepping method has temporal errors of $\mathcal{O}(\Delta t^q)$. Here, h represents our spatial discretization spacing and Δt is our time-step size. Constants c_1 and c_2 are problem dependent, but once determined can be used to find the location where spatial and temporal errors are balanced (i.e. where $c_1 h^p \Delta t = c_2 \Delta t^q$). The confluence of these perspectives allows us to both appreciate and explain why MPM exhibits the temporal convergence behavior as reported in the literature, and more importantly, allows us to provide guidelines to the practitioner concerning the interplay between space and time errors.

3. OVERVIEW OF MPM

In this section, we will begin with a very short review of the Galerkin discretization of equations of motion. Next, we will show how MPM can be derived from various approximations while solving the Galerkin discretization. Moving-mesh MPM will then be outlined and the differences between standard and moving-mesh MPM will be highlighted.

DOI: 10.1002/nme

3.1. Galerkin discretization of equations of motion

The equation of motion for a continuum in the updated Lagrangian frame is given by:

$$\rho\mathbf{a} = \nabla\cdot\boldsymbol{\sigma} + \rho\mathbf{b}. \tag{2}$$

Here, ρ is the material density, $\mathbf{a}$ is acceleration, $\boldsymbol{\sigma}$ is Cauchy stress (assumed to be symmetric in this paper), and $\mathbf{b}$ is the acceleration due to body forces. Next, we write acceleration as a linear combination of basis functions $\{\phi_i\}$, where $\mathbf{a}(x) = \sum_i \mathbf{a}_i \phi_i(x)$. Substituting this into (2) and taking the inner product of each term with a test function ϕ_j leaves us with the Galerkin weak form of the equation of motion:

$$\left(\rho \sum_i \mathbf{a}_i \phi_i, \phi_j\right) = -(\boldsymbol{\sigma}, \nabla\phi_j) + (\rho\mathbf{b}, \phi_j), \tag{3}$$

where the notation (a, b) represents the inner product of the functions a and b over our domain Ω, i.e. $(a, b) = \int_\Omega a\cdot b\,\mathrm{d}\Omega$. Equation (3) represents a linear system written as the following matrix equation[‡]:

$$\mathbf{M}\mathbf{a} = \mathbf{f}^{\text{int}} + \mathbf{f}^{\text{ext}} \tag{4}$$

where

$$M_{ij} = \int_\Omega \rho\phi_i\phi_j\,\mathrm{d}\Omega, \tag{5}$$

$$\mathbf{f}_i^{\text{int}} = -\int_\Omega \boldsymbol{\sigma}\cdot\nabla\phi_i\,\mathrm{d}\Omega, \tag{6}$$

and

$$\mathbf{f}_i^{\text{ext}} = \int_\Omega \rho\mathbf{b}\phi_i\,\mathrm{d}\Omega. \tag{7}$$

One method for simplifying (4), such that solving a linear system is no longer required to lump the mass matrix $\mathbf{M}$—that is, substitute $\mathbf{M}$ with a diagonal matrix $\tilde{\mathbf{M}}$. There are a number of methods to mass lump $\mathbf{M}$ [19]; however, we will only consider mass lumping using the row-sum technique, as it is the most prevalently used method employed in practice within the MPM community. The row-sum technique is particularly simple: $\tilde{M}_{ii} = m_i = \sum_j M_{ij}$. Once $\mathbf{M}$ has been mass lumped, the solution to (4) reduces to:

$$\mathbf{a}_i = (\mathbf{f}_i^{\text{int}} + \mathbf{f}_i^{\text{ext}})/m_i. \tag{8}$$

If the basis functions maintain a partition of unity within the domain, $\sum_i \phi_i(x) = 1$ for all $x \in \Omega$, and the diagonal term m_i can be calculated directly and efficiently, without generating all the terms in $\mathbf{M}$, since

$$m_i = \sum_j M_{ij} = \int_\Omega \rho\phi_i \sum_j \phi_j\,\mathrm{d}\Omega = \int_\Omega \rho\phi_i\,\mathrm{d}\Omega. \tag{9}$$

[‡]When written as a linear system, it is tacitly understood that lower case terms are arrays of values, as in (4).

DOI: 10.1002/nme

Once $\mathbf{a}_i$ is determined, the rate equations for velocity ($\dot{\mathbf{v}}=\mathbf{a}$) and position ($\dot{\mathbf{x}}=\mathbf{v}$) can be integrated and updated with standard ODE time-stepping algorithms.

3.2. Standard MPM

The MPM procedure begins by discretizing the problem domain Ω with a set of material points or particles. These particles are assigned initial values of position (in the reference or material frame), displacement, velocity, mass, volume, and deformation gradient, denoted $\mathbf{X}_p$, $\mathbf{u}_p$, $\mathbf{v}_p$, m_p, V_p, and $\mathbf{F}_p$, respectively. The subscript index p is used to distinguish particle values from an index of i for grid node values. The current position of a particle in the deformed configuration can easily be calculated as $\mathbf{x}_p=\mathbf{X}_p+\mathbf{u}_p$, where $\mathbf{X}_p$ is the initial position in the material frame and $\mathbf{u}_p$ is the displacement vector. Alternatively, instead of velocity and mass, momentum and mass density may be prescribed at the particle location, from which m_p and $\mathbf{v}_p$ can be calculated. Depending on the simulation, other quantities may be required at the material points as well, such as temperature. A computational background mesh fully encompassing the simulated objects is constructed, which for ease of computation is usually chosen to be a regular Cartesian lattice.

In order to advance from time-level t^k to t^{k+1} (all of the following quantities will be assumed to be at time-level t^k unless otherwise noted), the first step in the MPM computational algorithm involves projecting (or spreading) data from the material points to the grid. An initial Galerkin projection of particle momentum allows grid velocity to be calculated:

$$\left(\rho\sum_i \mathbf{v}_i\phi_i,\phi_j\right)=(\rho\mathbf{v},\phi_j). \tag{10}$$

Solving (10) for all i and j is again equivalent to solving the following linear system:

$$\mathbf{M}\mathbf{v}=\mathbf{p}, \tag{11}$$

where $\mathbf{v}_i$ is the velocity associated with node i, $\mathbf{M}$ is the mass matrix (5), and

$$\mathbf{p}_i=\int_\Omega \rho\mathbf{v}\phi_i\,\mathrm{d}\Omega. \tag{12}$$

To avoid an expensive linear solve, we again mass lump our matrix $\mathbf{M}$, in which case $\mathbf{v}_i$ is now found by solving

$$\mathbf{v}_i=\frac{\mathbf{p}_i}{m_i}=\frac{\int_\Omega \rho\mathbf{v}\phi_i\,\mathrm{d}\Omega}{\int_\Omega \rho\phi_i\,\mathrm{d}\Omega}. \tag{13}$$

A defining feature of the MPM algorithm is the use of nodal integration to approximate the integrals in equations such as (13). Given an initial undeformed particle volume V_p^0 and its current deformation gradient $\mathbf{F}_p$, the current particle volume is calculated as

$$V_p=\det(\mathbf{F}_p)V_p^0. \tag{14}$$

DOI: 10.1002/nme

Using this updated volume, (13) is approximated with nodal integration (a quasi-midpoint rule) where field quantities are assumed to be sampled by particle values as follows:

$$\mathbf{v}_i = \frac{\mathbf{p}_i}{m_i} \approx \frac{\sum_p \rho_p \mathbf{v}_p \phi_{ip} V_p}{\sum_p \rho_p \phi_{ip} V_p} = \frac{\sum_p \frac{m_p}{V_p} \mathbf{v}_p \phi_{ip} V_p}{\sum_p \frac{m_p}{V_p} \phi_{ip} V_p} = \frac{\sum_p m_p \mathbf{v}_p \phi_{ip}}{\sum_p m_p \phi_{ip}} \tag{15}$$

where $\phi_{ip} = \phi_i(\mathbf{x}_p)$ is the basis function centered at grid node i evaluated at the particle position $\mathbf{x}_p$. We will define $m_i = \sum_p m_p \phi_{ip}$ as nodal mass, which also represents the mass-lumped version of what Sulsky and Kaul [20] describe as the consistent mass matrix $M_{ij} = \sum_p \phi_{ip} \phi_{jp} m_p$. Next, the internal force term (6) is found by first calculating stress as a function of the constitutive model and the deformation gradient stored with each particle, then by multiplying stress by the gradient of ϕ_i. Again, nodal integration is used as a means of approximating the integral in the expression

$$\mathbf{f}_i^{\text{int}} = -\int_\Omega \boldsymbol{\sigma} \cdot \nabla \phi_i \, \mathrm{d}\Omega \approx -\sum_p \boldsymbol{\sigma}_p \cdot \nabla \phi_{ip} V_p, \tag{16}$$

where the stress is a function of the deformation gradient, $\boldsymbol{\sigma}_p = \sigma(\mathbf{F}_p)$, and where $\nabla \phi_{ip} = \nabla \phi_i(\mathbf{x}_p)$. The external force term (7) is then calculated given any body force as follows:

$$\mathbf{f}_i^{\text{ext}} = \int_\Omega \rho \mathbf{b} \phi_i \, \mathrm{d}\Omega \approx \sum_p \frac{m_p}{V_p} \mathbf{b}_p \phi_{ip} V_p \approx \sum_p m_p \mathbf{b}_p \phi_{ip}. \tag{17}$$

Using nodal mass m_i and the internal and external forces from (16) and (17), respectively, we can now calculate nodal accelerations $\mathbf{a}_i$ using (8). Grid velocities are then updated with an appropriate time-stepping scheme. Implicit time-stepping schemes exist for MPM [16, 20, 21]; however, we choose to use the explicit Euler-Forward time discretization presented within the original MPM algorithm, which has the following expression for the update of velocity:

$$\mathbf{v}_i^{k+1} = \mathbf{v}_i^k + \mathbf{a}_i \Delta t. \tag{18}$$

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

$$\nabla \mathbf{v}_p^{k+1} = \sum_i \nabla \phi_{ip} \mathbf{v}_i^{k+1}. \tag{19}$$

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

$$\mathbf{F}_p^{k+1} = (\mathbf{I} + \nabla \mathbf{v}_p^{k+1} \Delta t) \mathbf{F}_p^k, \tag{20}$$

$$\mathbf{v}_p^{k+1} = \mathbf{v}_p^k + \sum_i \phi_{ip} \mathbf{a}_i \Delta t, \tag{21}$$

and

$$\mathbf{u}_p^{k+1} = \mathbf{u}_p^k + \sum_i \phi_{ip} \mathbf{v}_i^{k+1} \Delta t. \tag{22}$$

DOI: 10.1002/nme

Equations (15)–(22) outline one time-step of MPM and assume initialization of particle values at time t^0: $\mathbf{u}_p^0$, $\mathbf{v}_p^0$, $\mathbf{F}_p^0$, and V_p^0. If possible, a simple change of initializing particle velocities a half time-step earlier, i.e. $\mathbf{v}_p^{-1/2}$, and using the same MPM algorithmic procedure outlined above leads to the following set of staggered central-difference update equations:

$$\mathbf{v}_i^{k+1/2} = \mathbf{v}_i^{k-1/2} + \mathbf{a}_i \Delta t, \tag{23}$$

$$\nabla \mathbf{v}_p^{k+1/2} = \sum_i \nabla \phi_{ip} \mathbf{v}_i^{k+1/2}, \tag{24}$$

$$\mathbf{F}_p^{k+1} = (\mathbf{I} + \nabla \mathbf{v}_p^{k+1/2} \Delta t) \mathbf{F}_p^k, \tag{25}$$

$$\mathbf{v}_p^{k+1/2} = \mathbf{v}_p^{k-1/2} + \sum_i \phi_{ip} \mathbf{a}_i \Delta t, \tag{26}$$

and

$$\mathbf{u}_p^{k+1} = \mathbf{u}_p^k + \sum_i \phi_{ip} \mathbf{v}_i^{k+1/2} \Delta t. \tag{27}$$

A similar staggered central difference method is used for MPM by Sulsky *et al.* [5], the benefits of which are reviewed in detail by Wallstedt and Guilkey [7].

The calculation of $\boldsymbol{\sigma}_p$ involves a constitutive model evaluation and is specific for different material models. The neo-Hookean elastic constitutive model used in this paper is detailed in Section 6.1.

Most standard MPM implementations use piecewise-linear basis functions for ϕ_i due to their ease of implementation and small local support. The one-dimensional form of the piecewise-linear basis function is given by

$$\phi(x) = \begin{cases} 1 - |x|/h & : \quad |x| < h \\ 0 & : \quad \text{otherwise}, \end{cases} \tag{28}$$

where h is the grid spacing. The basis function associated with grid node i at position x_i is then $\phi_i = \phi(x - x_i)$. The basis functions in multiple dimensions are separable functions, constructed using (28) in each dimension. For example, in three-dimensions, we have:

$$\phi_i(\mathbf{x}) = \phi_i^x(x) \phi_i^y(y) \phi_i^z(z). \tag{29}$$

Recently, the benefits of smoother basis functions have been explored within the MPM framework. For example, B-splines have been shown to decrease quadrature errors and improve the spatial convergence rates for many MPM problems [11]. A typical one-dimensional quadratic B-spline can be constructed by convolving piecewise-constant basis functions with themselves:

$$\phi = \chi * \chi * \chi / (|\chi|)^2, \tag{30}$$

where χ is the piecewise-constant basis function:

$$\chi(x) = \begin{cases} 1 & : \quad |x| < \frac{1}{2} l \\ 0 & : \quad \text{otherwise}, \end{cases} \tag{31}$$

DOI: 10.1002/nme

and l is the width of χ. A separable three-dimensional B-spline basis function can then be constructed using (29).

If we depart from the idea that each grid node corresponds to a single basis function, we can discretize our one-dimensional domain of length L with n knots, and construct quadratic B-spline basis functions from the open knot vector

$$[x_0, x_0, x_1, \ldots, x_i, \ldots, x_{n-2}, x_{n-1}, x_{n-1}], \tag{32}$$

where $x_i = x_0 + i \cdot h$, and the knot spacing $h = L/(n-1)$. For a k-order B-splines (for quadratic B-splines, $k=3$), there will be $n+k-2$ basis functions, which are calculated recursively as

$$\phi_{i,k} = \phi_{i,k-1} \frac{x - x_i}{x_{i+k-1} - x_i} + \phi_{i+1,k-1} \frac{x_{i+k} - x}{x_{i+k} - x_{i+1}}, \tag{33}$$

$$\phi_{i,1} = \begin{cases} 1 & : \quad x_i \leqslant x < x_{i+1} \\ 0 & : \quad \text{otherwise.} \end{cases} \tag{34}$$

This is more akin to high-order finite elements, where the number of degrees of freedom remain constant within each grid cell. However, unlike high-order finite elements, these B-spline basis functions maintain the partition of unity property required for the simple mass-lumping implicit in the MPM algorithm. These B-spline basis functions are also C^1 continuous at the grid node boundaries, allowing for reduced quadrature and grid crossing errors [11]. More details regarding the use of B-spline basis functions within MPM, including boundary condition choices, are outlined in [12].

3.3. Moving-mesh MPM

The term 'moving-mesh MPM' which we (and others) employ denotes an MPM method that is fully Lagrangian, where the mesh 'moves' with the particles. However, moving-mesh MPM is actually implemented by keeping both the mesh and particles stationary in the reference configuration and keeping track of displacements for the particles and grid nodes. This is similar to what is done in standard FEM methods with the major difference being that particle locations essentially define the quadrature point locations. Moving-mesh MPM may seem contrary to the spirit of MPM, in that typical FEM difficulties such as mesh-entanglement can occur. However, many of the benefits of standard MPM are still present in moving-mesh MPM, such as ease of initial discretization of complex geometries using techniques similar to those used by Brydon *et al.* in the simulation of foam [3].

To help to understand the mathematical and algorithmic differences between standard MPM and moving-mesh MPM, we start by examining the calculation of mass at the grid nodes within the standard MPM algorithm:

$$m_i = \int_\Omega \rho(\mathbf{x}) \phi_i(\mathbf{x}) \, \mathrm{d}\Omega \tag{35}$$

$$\approx \sum_p \rho_p \phi_i(\mathbf{x}_p) V_p \tag{36}$$

$$= \sum_p \frac{m_p}{V_p} \phi_{ip} V_p \tag{37}$$

$$= \sum_p m_p \phi_{ip}, \tag{38}$$

DOI: 10.1002/nme

where $\rho_p \equiv m_p / V_p$. If instead of the position of the particles, we keep track of displacements $\mathbf{u}$, such that $\mathbf{x} = \mathbf{X} + \mathbf{u}(\mathbf{X})$, we can represent this as

$$m_i = \int_{\Omega_0} \rho(\mathbf{X}) \Phi_i(\mathbf{X}) J \, \mathrm{d}\Omega_0 \tag{39}$$

$$\approx \sum_p \rho_p \Phi_i(\mathbf{X}_p) \det(\mathbf{F}_p) V_p^0 \tag{40}$$

$$= \sum_p \frac{m_p}{V_p} \Phi_{ip} \det(\mathbf{F}_p) V_p^0 \tag{41}$$

$$= \sum_p \frac{m_p}{\det(\mathbf{F}_p) V_p^0} \Phi_{ip} \det(\mathbf{F}_p) V_p^0 \tag{42}$$

$$= \sum_p m_p \Phi_{ip}. \tag{43}$$

Here, $J = \det(\mathbf{F})$ is the Jacobian of the mapping from Ω_0 to Ω. Therefore, algorithmically, mass and velocity projections in moving-mesh MPM are very similar to mass projections in standard MPM, except that Φ_i is evaluated in the reference configuration at $\mathbf{X}_p$ instead of the deformed configuration at $\mathbf{x}_p$. The first algorithmic difference between moving-mesh MPM and standard MPM then comes when calculating the deformation gradients $\mathbf{F}_p$. In standard MPM, deformation gradients are time-integrated as in (20). However, the definition of the deformation gradient is $\mathbf{F} = \mathbf{I} + \partial \mathbf{u} / \partial \mathbf{X}$ and as displacements are maintained on the grid, $\mathbf{F}_p$ can be directly calculated from $\mathbf{u}_i$

$$\mathbf{F}_p = \mathbf{I} + \sum_i \nabla_0 \Phi_i(\mathbf{X}_p) \mathbf{u}_i, \tag{44}$$

where ∇_0 denotes the gradient with respect to coordinates in the reference frame and an outer product is implied. Stress can then be calculated from $\mathbf{F}_p$.

The next departure from standard MPM is the calculation of internal force. Using the relation between the first Piola–Kirkhhoff and Cauchy stress tensors: $\mathbf{P} = J \boldsymbol{\sigma} \mathbf{F}^{-T}$, and the appropriate transformation of test functions (via the deformation gradient), one arrives at the equivalent force calculation and approximation (16) in the reference frame:

$$\mathbf{f}_i^{\text{int}} = -\int_\Omega \boldsymbol{\sigma}(\mathbf{x}) \cdot \nabla \phi_i(\mathbf{x}) \, \mathrm{d}\Omega = -\int_{\Omega_0} \mathbf{P}(\mathbf{X}) \cdot \nabla_0 \Phi_i(\mathbf{X}) \, \mathrm{d}\Omega_0 \approx -\sum_p \mathbf{P}_p \cdot \nabla_0 \Phi_{ip} V_p^0. \tag{45}$$

This internal force calculation differs from standard MPM in that $\nabla_0 \Phi$ is evaluated at $\mathbf{X}_p$ in the reference configuration, the first Piola–Kirkhhoff stress is used instead of the Cauchy stress, and the initial undeformed particle volume V_p^0 is used instead of the updated volume V_p.

The initialization of moving-mesh MPM is similar to standard MPM, discretizing the problem domain with a set of material points and assigning those points initial particle values, including displacements $\mathbf{u}_p = \mathbf{u}_0(\mathbf{X}_p)$, with $\mathbf{u}_0(\mathbf{X})$ the initial displacement field. Particles should be equally spaced and aligned with the grid cell boundaries as the major benefits of moving-mesh MPM are only obtained when particles are in these 'ideal' positions.

As grid displacements are maintained and used in (44), initialization also requires a projection of $\mathbf{u}_0$ onto the grid. This is accomplished by initializing $\mathbf{u}_i$ through an approximate L_2 projection of $\mathbf{u}_0$ onto $\{\Phi_i\}$. Again, the full L_2 projection would come from solving the following equation:

$$\mathbf{A}\mathbf{u} = \mathbf{b}, \tag{46}$$

DOI: 10.1002/nme

where $A_{ij} = (\Phi_i, \Phi_j)$ and $\mathbf{b}_i = \int_{\Omega_0} \Phi_i(\mathbf{X})\mathbf{u}_0(\mathbf{X})\,\mathrm{d}\Omega_0$. Continuing with the MPM philosophy, we solve the above equation first by mass lumping $\mathbf{A}$, then by nodal integration. Thus we obtain

$$\mathbf{u}_i = \frac{\mathbf{b}_i}{\sum_j A_{ij}} \tag{47}$$

$$= \frac{\int_{\Omega_0} \Phi_i(\mathbf{X})\mathbf{u}_0(\mathbf{X})\,\mathrm{d}\Omega_0}{\int_{\Omega_0} \Phi_i\,\mathrm{d}\Omega_0} \tag{48}$$

$$\approx \frac{\sum_p \Phi_{ip}\mathbf{u}_p V_p^0}{\sum_p \Phi_{ip} V_p^0}. \tag{49}$$

A typical moving-mesh MPM algorithm would then proceed as follows: during initialization, grid displacements are initialized from particle displacements:

$$\mathbf{u}_i = \frac{\sum_p \Phi_{ip}\mathbf{u}_p V_p^0}{\sum_p \Phi_{ip} V_p^0}. \tag{50}$$

Then, for each time-step, perform the following operations:

Solve for mass at grid	$m_i = \sum_p m_p \Phi_{ip}$	(51)
Solve for grid velocity	$\mathbf{v}_i^k = \sum_p m_p \mathbf{v}_p^k \Phi_{ip}/m_i$	(52)
Solve for external forces	$\mathbf{f}_i^{\text{ext}} = \sum_p m_p \mathbf{b}_p \Phi_{ip}$	(53)
Solve for internal forces	$\mathbf{f}_i^{\text{int}} = -\sum_p \mathbf{P}_p^k \cdot \nabla_0 \Phi_{ip} V_p^0$	(54)
Solve for grid acceleration	$\mathbf{a}_i^k = (\mathbf{f}_i^{\text{int}} + \mathbf{f}_i^{\text{ext}})/m_i$	(55)
Time advance grid velocity	$\mathbf{v}_i^{k+1} = \mathbf{v}_i^k + \mathbf{a}_i^k \Delta t$	(56)
Time advance grid displacements	$\mathbf{u}_i^{k+1} = \mathbf{u}_i^k + \mathbf{v}_i^{k+1} \Delta t$	(57)
Time advance particle deformation gradient	$\mathbf{F}_p^{k+1} = \mathbf{I} + \sum_i \nabla_0 \Phi_{ip} \mathbf{u}_i^{k+1}$	(58)
Solve constitutive model	$\mathbf{P}_p^{k+1} = \mathbf{P}(\mathbf{F}_p^{k+1})$	(59)
Time advance particle velocities	$\mathbf{v}_p^{k+1} = \mathbf{v}_p^k + \Delta t \sum_i \mathbf{a}_i^k \Phi_{ip}$	(60)
Time advance particle displacements	$\mathbf{u}_p^{k+1} = \mathbf{u}_p^k + \Delta t \sum_i \mathbf{v}_i^{k+1} \Phi_{ip}.$	(61)

Another significant difference between standard MPM and moving-mesh MPM is how $\mathbf{F}_p$ is calculated. Standard MPM time integrates $\mathbf{F}$ as in (20), whereas moving-mesh MPM calculates $\mathbf{F}_p$ directly from grid displacements in (58).

DOI: 10.1002/nme

4. INTERPRETING THE COUPLING OF LAGRANGIAN AND EULERIAN SIMULATIONS

Although MPM involves numerous discretization and approximation choices in the simulation of physical and mathematical problems, many of the errors previously observed in MPM, including grid crossing errors, can be viewed as quadrature errors in integrating spatial quantities. Specifically, Steffen *et al.* [11] shows how nodal integration in MPM is essentially a midpoint integration-type scheme, where discontinuities in spatial quantities (at the grid nodes, in particular) are not respected within the integration scheme, as one would normally do when integrating discontinuous functions with the midpoint rule. This occurs because particle voxels may not be aligned with the grid cells. It is this overhanging of particle voxels with the grid cell boundaries that result in errors greater than what would normally be expected with the midpoint integration rule.

Quadrature errors are unique in MPM, in that they are fairly low order and time-dependent, or coupled, in standard Eulerian MPM. In standard MPM, a simulation may be initially discretized with particle voxels aligned with the grid cells; however, as the simulation progresses, particles move with respect to the grid (or in an alternate view, the grid is reset, which still causes the particles to be displaced with respect to their original grid positions), and these particle voxels overlap with the grid cell boundaries that begin to develop. Furthermore, this quadrature error will generate errors in acceleration, and in turn cause errors in velocity and position, changing again the particle positions with respect to the grid cells, and thus influencing future quadrature errors. This is to say, quadrature errors have a compounding effect on MPM.

One time-stepping algorithm currently employed in MPM to solve the two coupled first-order ODEs

$$\dot{\mathbf{v}}(\mathbf{x}, t) = \mathbf{a}(\mathbf{x}, t) \tag{62}$$

$$\dot{\mathbf{u}}(\mathbf{x}, t) = \mathbf{v}(\mathbf{x}, t). \tag{63}$$

is the centered difference time-integration method:

$$\mathbf{v}^{k+1/2} = \mathbf{v}^{k-1/2} + \mathbf{a}^k \Delta t \tag{64}$$

$$\mathbf{u}^{k+1} = \mathbf{u}^k + \mathbf{v}^{k+1/2} \Delta t. \tag{65}$$

As has been pointed out in the MPM literature [5], this method is 'formally' second-order in time. However, this formal analysis carries with it assumptions regarding the smoothness and accuracy of $\mathbf{a}$, assumptions that do not hold within the MPM framework. In particular, the acceleration calculated using the MPM algorithm may have significant quadrature errors in space and discontinuities in time [22], both of which make second-order temporal convergence unrealizable to the MPM practitioner. Figure 2 shows a sample of a typical grid acceleration field $\mathbf{a}(\mathbf{x}) = \sum_i \phi_i(\mathbf{x})\mathbf{a}_i$ encountered in standard MPM when piecewise-linear basis functions are used.[§] The jump in the acceleration occurs when a particle's position in space crosses a grid cell boundary. The calculated acceleration is obviously not smooth in this case, the impact of which has repercussions on the updated velocity and displacement of the particle.

One way to decouple and alleviate these errors is to employ Lagrangian, or moving-mesh MPM, as outlined in Section 3.3. As can be seen in Figure 1, the particles will remain fixed with respect

[§] The actual problem being simulated in Figure 2 is the 1-D elastic bar detailed in Section 6.1 but is only used as a qualitative motivating example here.

DOI: 10.1002/nme

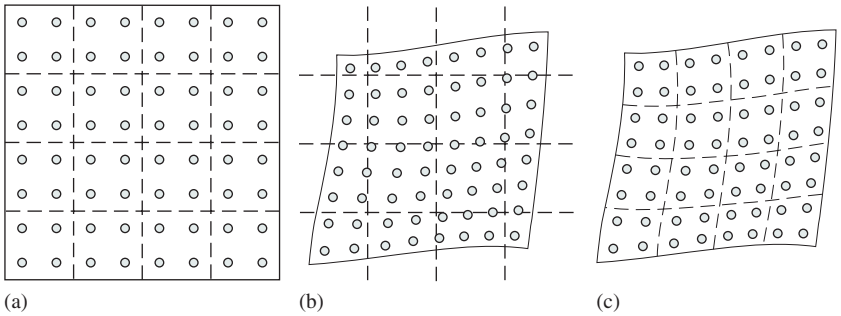

Figure 1. Standard MPM versus moving-mesh MPM. In moving-mesh MPM, particles remain at their ideal positions within grid cells (in the reference configuration). In standard MPM, particles change locations and cross grid cells leading to larger quadrature errors: (a) reference; (b) standard MPM; and (c) moving-mesh MPM.

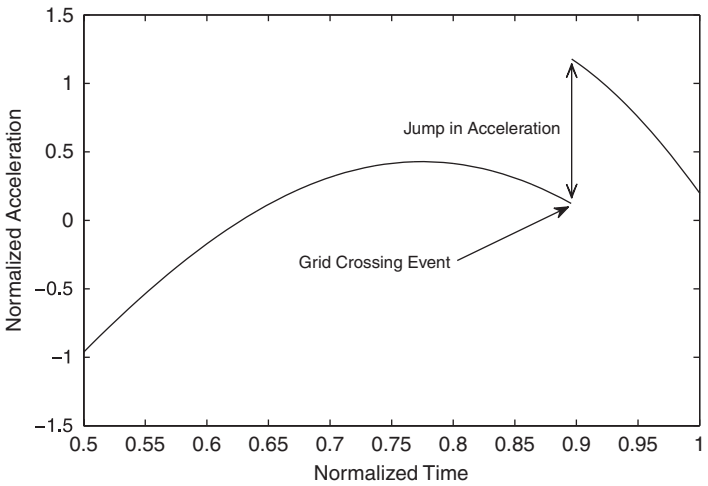

Figure 2. Grid acceleration field over time sampled by following the displacement of one particle. Standard MPM and piecewise-linear basis functions were used. The jump in acceleration occurs when a particle crosses a grid cell.

to the grid for all time. As particles do not move with respect to the grid, quadrature errors, while still present, are not time-dependent. Furthermore, if particles are initially grid-cell aligned, they will remain so as the simulation progresses, allowing for decreased quadrature errors as no particle-voxel and grid-cell boundary overlaps occur.

Moving-mesh MPM is very similar to standard FEM methods and thus suffers from many of the same problems. In particular, moving-mesh MPM is not well suited for large deformation problems and can experience mesh entanglement issues. The use of moving-mesh MPM may seem counter intuitive as large deformation problems are one of the main strengths of MPM, however our use of moving-mesh MPM will allow for the decoupling of spatial and temporal errors and aid in analysis and demonstration of these errors in the following sections.

DOI: 10.1002/nme

5. STUDIES OF SIMPLIFIED DECOUPLED PROBLEMS

The entire MPM algorithm, whether we consider standard MPM or moving mesh MPM as outlined in Equations (50)–(61), involves many steps and many approximations. Furthermore, as argued in Section 4, spatial and temporal errors are interconnected and exhibit compounding behavior, making analysis of full MPM simulations difficult. In this section, we start by presenting our decoupling strategy that allows us to study and analyze simpler problems that still demonstrate many of the numerical errors present in a full MPM simulation. Next, we will study the impact of spatial discontinuities on time-stepping by performing an analysis and showing demonstrations of the time-stepping jump error, where we look at the error associated with time-integrating past discontinuities in the velocity field. A study on the impact of quadrature errors on time-stepping follows. We conclude this section by examining the balance between spatial and temporal errors.

5.1. Decoupling strategy

If we consider the MPM algorithm in a reverse order of operations, our final goal is to time-integrate particle information, including the particle position:

$$\frac{\mathrm{d}x_p}{\mathrm{d}t} = v(x_p(t)). \tag{66}$$

MPM most often uses a Forward-Euler, or centered difference scheme to integrate the above equation, and again, the errors associated with these schemes are well understood [23]. Most previous analysis, however, assumes some level of continuity of the function $v(x)$. In standard MPM, the velocity field v is generated as a linear combination of piecewise-linear basis functions, giving rise to a piecewise-linear velocity field v. The integration of particle position (or displacement) in standard MPM is akin to performing streamline integration through a time-dependent piecewise-linear field in which the velocity field v is created from information about the particles. The errors arising in this situation will be illustrated in Section 5.2 by fixing a piecewise-linear velocity field $v(x)$, and performing streamline integration through this fixed velocity field to demonstrate the resulting jump errors.

In standard MPM, the velocity field is also time-integrated using an acceleration field a, which is also calculated using information from the particles:

$$\frac{\mathrm{d}v_p}{\mathrm{d}t} = a(x_p(t)) \tag{67}$$

$$a_p = a(x_p) = \sum_i a_i \phi_i(x_p) \tag{68}$$

$$a_i = f_i/m_i = \frac{1}{m_i}\int_\Omega \nabla\phi_i(x)\sigma(x)\,\mathrm{d}\Omega \approx \frac{1}{m_i}\sum_p \nabla\phi_{ip}\sigma(x_p)V_p. \tag{69}$$

The previous work [11] analyzed the quadrature errors that come about from the nodal integration approximation in Equation (69). Our next decoupling strategy which will allow us to look at the impact of these spatial quadrature errors on time-stepping is to specify a discontinuous field $g(x)$ (as $\nabla\phi_i(x)\sigma(x)$ is discontinuous when piecewise-linear basis functions are used) and define acceleration as $a_e = \int_\Omega g(x)\,\mathrm{d}\Omega$. This integral will be approximated in a similar fashion to the approximations in MPM. The resulting acceleration will not be the same as the acceleration

DOI: 10.1002/nme

calculated in (69), however the integration will be over a similarly discontinuous function, and thus we will see similar error behaviors. To avoid confusion, we will refer to a_e as 'external acceleration'. Employing this strategy will lead us to global error approximations in position resulting from spatial quadrature errors at each time step. Analysis and results of this problem follow in Section 5.3.

And finally, in Section 5.4 we will consider all the errors in the problem. With a better understanding of both spatial and temporal error behaviors, we will be able to predict and demonstrate where these spatial and temporal errors are balanced.

5.2. *Impact of spatial discontinuities on time-stepping*

The recent work by Tran *et al.* [22] analyzed errors in an MPM algorithm with respect to a gas dynamics problem. One feature of their MPM implementation which differs from most other implementations is a volume normalization step. While most implementations of MPM for solid mechanics define particle volume at time t^k as $V_p^k = \det(\mathbf{F}_p^k) V_p^0$, the algorithm used in Tran *et al.* defines particle volume (in 1-D) as $V_p^k = h/n_i^k$, where h is the grid spacing and n_i^k is the number of particles in the grid cell i (of which particle p also belongs to). Therefore, much of their analysis relating to spatial errors is not directly applicable to the variants of MPM presented here. They do, however, consider temporal errors when integrating past a jump in the continuity of the velocity field. This error is present in the standard MPM algorithm and we will consider it here.

5.2.1. Simplified problem. Before we proceed with an analysis, we wish to devise a simplified non-physical problem that exhibits many of the similar mathematical approximations and traits as the full MPM algorithm. The errors in this simplified problem will display similar characteristics to errors in the full MPM algorithm, but will be easier to analyze and will provide us insight into the expected error behavior in MPM.

The main mathematical features we wish to preserve from the full MPM algorithm are the evaluation of a piecewise-linear velocity field when time-integrating particle positions and the integration of a discontinuous field in the acceleration calculation. In doing so, we will consider a single particle p, starting at $x=0$ at time $t=0$. We will fix a piecewise-linear velocity field $v(x)$ on the domain $\Omega=[0,1]$, as shown in Figure 3(a). This velocity field is not time-dependent and is defined by:

$$v(x) = \begin{cases} 3x+1 & : \quad x \in [0, 1/3] \\ 6x & : \quad x \in [1/3, 2/3] \\ 12x-4 & : \quad x \in [2/3, 5/6] \\ 18x-9 & : \quad x \in [5/6, 1]. \end{cases} \tag{70}$$

With a particle initiating at $x=0$, the particle position can be determined by solving the following equation for $x(t)$:

$$\frac{\partial x}{\partial t} = v(x(t)). \tag{71}$$

DOI: 10.1002/nme

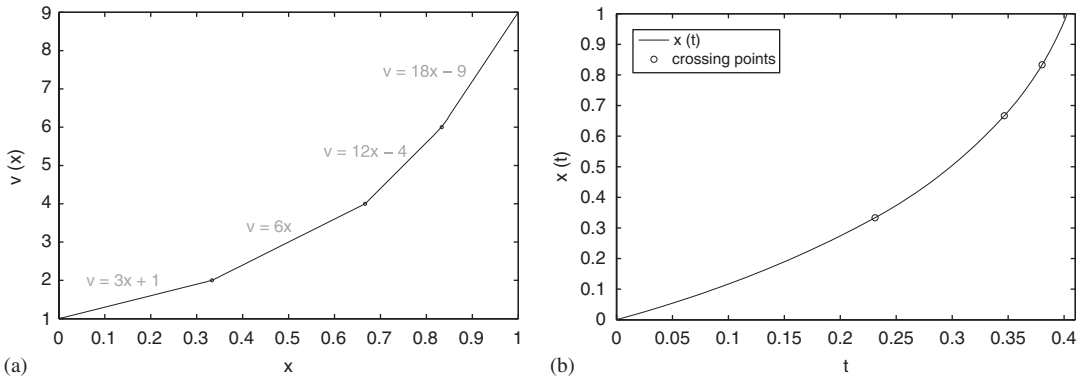

Figure 3. Fixed piecewise-linear velocity field and resulting $x(t)$ for our simplified problem: (a) velocity field and (b) particle position.

As our velocity field is piecewise-linear, this function can be solved analytically. For a linear velocity field $v(x)=ax+b$, the solution to this equation is

$$x(t)=\frac{b+ax_0}{ae^{at_0}}e^{at}-\frac{b}{a}. \tag{72}$$

The solution for $x\in[0,1/3]$, with $a=3$, $b=1$, $t_0=0$, and $x_0=0$, is then

$$x(t)=\tfrac{1}{3}e^{3t}-\tfrac{1}{3}. \tag{73}$$

This solution is valid only for $x\in[0,1/3]$. We can find at which times these are valid by solving the inverse equation with $x=x_1^{\text{cross}}=1/3$ for t_1^{cross}:

$$t_1^{\text{cross}}=\frac{1}{a}\ln\left[\frac{x_1^{\text{cross}}+b/a}{b+ax_0}ae^{at_0}\right]. \tag{74}$$

Therefore, (73) is valid for $t=[0,t_1^{\text{cross}}]$. The second segment, valid for $x\in[1/3,2/3]$, is calculated in a similar manner, with $a=6$, $b=0$, $t_0=t_1^{\text{cross}}$, and $x_0=1/3$. The second crossing time t_2^{cross} is calculated in a manner similar to (74), with $x=x_2^{\text{cross}}=2/3$. The resulting piecewise-exponential position function $x(t)$ is shown in Figure 3(b).

We can see the error behavior of this system by performing the following Forward-Euler time-integration strategy:

$$x_p^{k+1}=x_p^k+v(x_p^k)\Delta t. \tag{75}$$

The full MPM algorithm exhibits similar jump errors as the above problem due to the similarities in the piecewise-linear velocity fields.

5.2.2. Analysis. Figure 4 demonstrates a situation where a particle p samples a piecewise-linear velocity field $v(x)$ at time t^k. The particle position is then time-integrated to t^{k+1} using the standard forward-Euler scheme $x_p^{k+1}=x_p^k+\Delta t\,v(x_p^k)$. In this scenario, a grid crossing has occurred, i.e. $x_p^k<x_i<x_p^{k+1}$. As the velocity field $v(x)$ has a jump in continuity at x_i, standard ODE error bounds do not necessarily apply.

DOI: 10.1002/nme

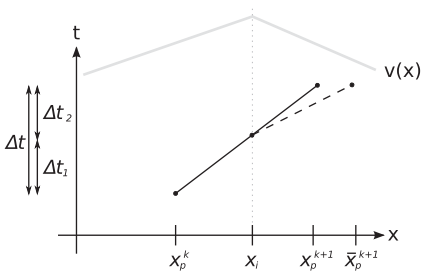

Figure 4. One-step versus two-step method for crossing a discontinuity in a velocity field.

One method for handling this situation is to perform a two-step time-integration strategy, where a time-step of Δt_1 is determined, which will bring the particle to the discontinuity, then a second time-step of $\Delta t_2 = \Delta t - \Delta t_1$ is taken, reevaluating the velocity field for the second time-step.

The algorithm would then be to calculate $x_p^{k+1} = x_p^k + \Delta t\, v(x_p^k)$ as normal. If a grid crossing has occurred where $x_p^k < x_i^{k+1} < x_p^{k+1}$, calculate the first time-step $\Delta t_1 = (x_i - x_p^k)/v(x_p^k)$ which will advance the particle to the grid node x_i. Next, calculate an adjusted two-step particle position as $\bar{x}_p^{k+1} = x_i + \Delta t_2 v(x_i)$.

The difference between the two-step and one-step particle positions, $\bar{x}_p^{k+1} - x_p^{k+1}$, or the time-stepping jump error, was calculated in [22]. They showed this difference to be:

$$\bar{x}_p^{n+1} - x_p^{n+1} = (v_i^{n+1} - v_{i-1}^{n+1}) \left[\frac{x_i - x_p^n}{x_i - x_{i-1}} \right] \Delta t_2 + \left[a_{i-1}^n + \frac{x_p^n - x_{i-1}}{x_i - x_{i-1}} (a_i^n - a_{i-1}^n) \right] \Delta t_1 \Delta t_2. \tag{76}$$

Here, we continue to expand on the analysis in [22] to help to understand the relationship between decreasing Δt and the expected behavior of the difference $\bar{x}_p^{n+1} - x_p^{n+1}$ in (76).

To simplify, the second term (in the square brackets) is merely the projection of grid acceleration onto the particle at time n: a_p^n. Therefore, we can rewrite this as:

$$\bar{x}_p^{n+1} - x_p^{n+1} = (v_i^{n+1} - v_{i-1}^{n+1}) \left[\frac{x_i - x_p^n}{x_i - x_{i-1}} \right] \Delta t_2 + a_p^n \Delta t_1 \Delta t_2. \tag{77}$$

Rearranging the first term gives:

$$\frac{v_i^{n+1} - v_{i-1}^{n+1}}{x_i - x_{i-1}} (x_i - x_p^n) \Delta t_2 \approx \frac{\partial v}{\partial x} (x_i - x_p^n) \Delta t_2. \tag{78}$$

To arrive at this point, we have assumed that the particle has crossed the grid node x_i during a full time-step. i.e. $x_p^n < x_i < x_p^{n+1}$. Furthermore, we know that $x_i = x_p^n + v_p \Delta t_1$, where v_p is the projection of grid velocities to the particle position. Therefore, $x_i - x_p^n = v_p \Delta t_1$. Plugging this into the above, we see that the first term looks like:

$$\frac{\partial v}{\partial x} v_p \Delta t_1 \Delta t_2. \tag{79}$$

DOI: 10.1002/nme

Thus, we get an error in position of the form:

$$\bar{x}_p^{n+1} - x_p^{n+1} = \left[\frac{\partial v}{\partial x} v_p + a_p^n\right] \Delta t_1 \Delta t_2 \tag{80}$$

As $\Delta t_2 = \Delta t - \Delta t_1$ with $\Delta t_1 < \Delta t$, we can rewrite these time-steps as $\Delta t_1 = \alpha \Delta t$ with $0 < \alpha < 1$ and $\Delta t_2 = (1-\alpha)\Delta t$. Thus

$$\Delta t_1 \Delta t_2 = \alpha(1-\alpha)\Delta t^2. \tag{81}$$

The term $\alpha(1-\alpha)$ has a maximum of $\frac{1}{4}$ at $\alpha = \frac{1}{2}$, thus the error in position is bounded by

$$\bar{x}_p^{n+1} - x_p^{n+1} \leqslant \frac{1}{4}\left[\frac{\partial v}{\partial x} v_p + a_p^n\right] \Delta t^2. \tag{82}$$

Therefore, the time-stepping jump error, or the error between the two-step and the one-step methods, is $\mathcal{O}(\Delta t^2)$. The following section will show the results demonstrating this second-order error behavior.

5.2.3. Results. The following is a test with a piecewise-linear velocity field (arising from piecewise-linear basis functions). Given a time-step Δt, the equation

$$x_i = x_p + \Delta t_1 v(x_p) \tag{83}$$

was solved for x_p with $\Delta t_1 = \Delta t/2$. This gives us a starting position, such that the velocity field will move the particle such that x_i is halfway between x_p^n and $\bar{x}_p^{n+1}$. Next, x_p^{n+1} is calculated in the two-step method, i.e.:

$$x_p^{n+1} = x_p^n + v_p \Delta t_1 + v_i \Delta t_2 = x_i + \frac{\Delta t}{2} v_i. \tag{84}$$

The difference $\bar{x}_p^{n+1} - x_p^{n+1}$ is calculated and plotted in Figure 5. Here, we can see the $\mathcal{O}(\Delta t^2)$ convergence we expect.

Our estimate for the jump error in Section 5.2.2 was

$$\varepsilon_{\text{jump}} = \frac{1}{4}\left[\frac{\partial v}{\partial x_-} v_p + a_p^n\right] \Delta t^2. \tag{85}$$

The terms $\partial v/\partial x$, v_p, and a_p are easily calculated for the simplified problem in Section 5.2.1. Using an initial time-step (before refinement) of $\Delta t_0 = 0.01$, we can measure the jump errors and compare against our estimates. The jump error is estimated using (85) and calculated as the difference between performing the time-integration strategy in the standard fashion (giving x_p) and performing the time-integration utilizing the two-step strategy to obtain $\bar{x}_p$:

$$\varepsilon_{\text{jump}}^k = x_p^k - \bar{x}_p^k. \tag{86}$$

Figure 6(a) and (b) show the calculated jump errors for various time-step selections. Table I shows the estimated and calculated jump errors for a particular time-step, demonstrating that the error bounds are tight.

DOI: 10.1002/nme

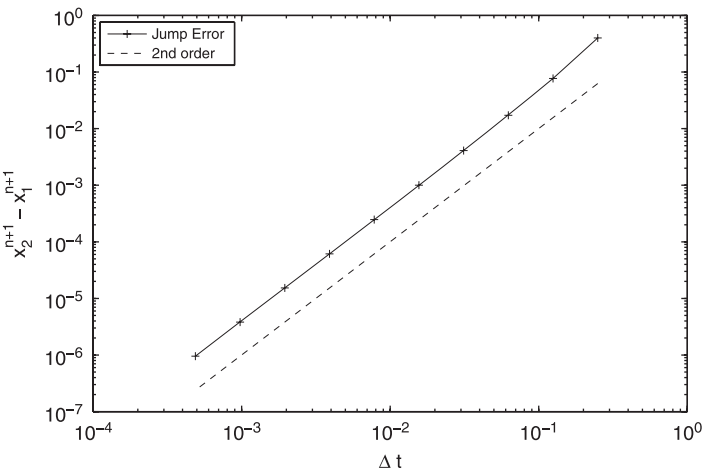

Figure 5. Convergence of the jump error ($\bar{x}_p^{n+1} - x_p^{n+1}$) when x_i is half the distance between x_p^n and x_p^{n+1}.

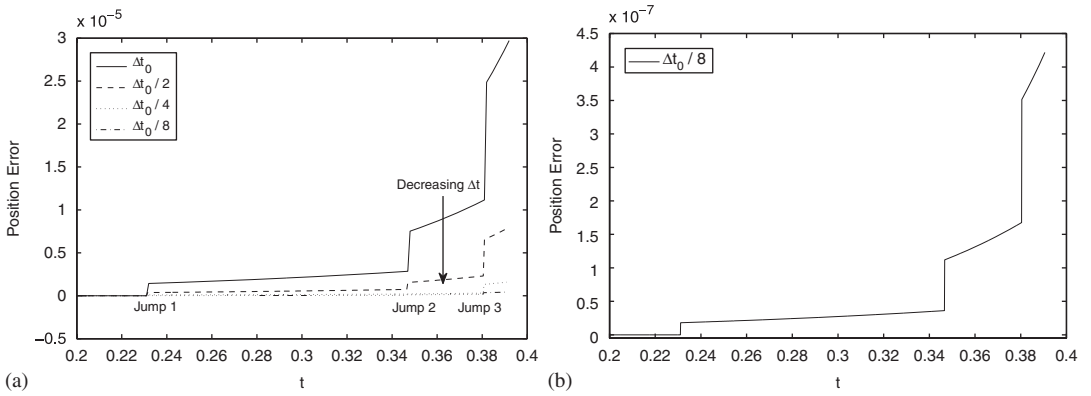

Figure 6. Measured jump errors for simplified problem: (a) multiple time-step sizes and (b) single time-step size.

Table I. Estimated and calculated values for all three jumps in the simplified problem, showing tight bounds for the estimated jump error.

Jump	Estimated jump	Calculated jump
1	2.34×10^{-8}	1.81×10^{-8}
2	9.36×10^{-8}	7.60×10^{-8}
3	2.81×10^{-7}	1.84×10^{-7}

DOI: 10.1002/nme

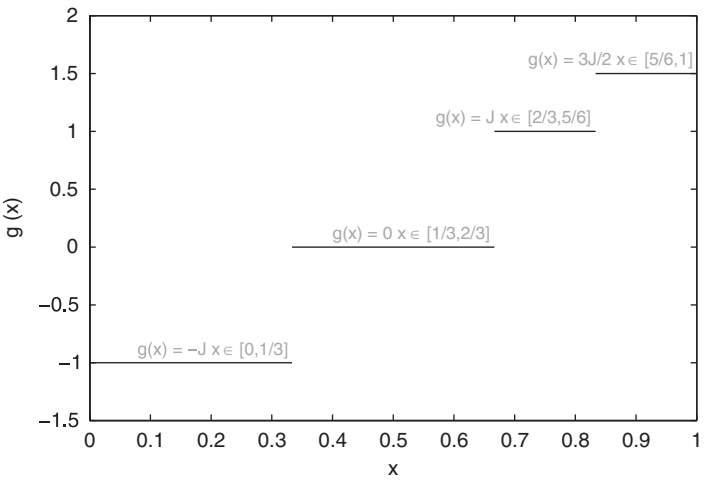

Figure 7. Function $g(x)$ used in calculating external acceleration for a simplified problem.

5.3. *Impact of spatial quadrature errors on time stepping*

The recent work [11] analyzed quadrature errors in the MPM framework but did not extend to take into account the feedback that occurs between spatial and temporal errors as a simulation progresses. In this section we will introduce a modification to our simplified problem which will exhibit similar quadrature errors as found in standard MPM. We will follow this with an analysis and demonstration of these errors.

5.3.1. Simplified problem. We will use the same prescribed velocity field v as developed in Section 5.2.1, however, we will also define an external acceleration $a_e(t)$, defined by integrating a given function $g(x)$ over the domain Ω. The function $g(x)$, shown in Figure 7, is not time-dependent and is defined as:

$$g(x) = \begin{cases} -J & : \quad x \in [0, 1/3] \\ 0 & : \quad x \in [1/3, 2/3] \\ J & : \quad x \in [2/3, 5/6] \\ 3J/2 & : \quad x \in [5/6, 1]. \end{cases} \tag{87}$$

The function $g(x)$ was chosen such that $\int_\Omega g(x)\,\mathrm{d}x = 0$. However, similar to the nodal integration in MPM, we will approximate this integral with midpoint quadrature over the particle domain $\Omega_p = [x_p - \Delta x/2, x_p + \Delta x/2]$, i.e. acceleration is approximated as

$$a_e(t) = \int_\Omega g(x)\,\mathrm{d}x \approx \int_{\Omega - \Omega_p(t)} g(x)\,\mathrm{d}x + g(x_p)\Delta x. \tag{88}$$

Again, the inclusion of a_e is not meant to be physical, but to include a forcing term that exhibits quadrature errors similar to those that occur when calculating the acceleration in MPM.

DOI: 10.1002/nme

This is accomplished as the integrands in the calculation of our simplified external acceleration a_e and the full MPM acceleration (69) (when piecewise-linear basis functions are used) are both discontinuous.

If the calculation of $a_e(t)$ can be carried out exactly, external acceleration should be zero for all time and the behavior of the particle should be the same as in Figure 3(b). Any error in the integration will result in non-zero external accelerations. The errors in the integration result from errors in the above midpoint approximation. The particle position x_p changes with time, and thus this error is time-dependent, hence $a_e(t)$ is time-dependent, even though $g(x)$ is not.

Finally, we can see the behavior of this system by performing the following Forward-Euler time-integration strategy

$$a_e^k = \int_{\Omega-\Omega_p} g(x)\,\mathrm{d}x + g(x_p^k)\Delta x \tag{89}$$

$$v_e^{k+1} = v_e^k + a_e^k \Delta t \tag{90}$$

$$v^{k+1} = v(x_p^k) + v_e^{k+1} \tag{91}$$

$$x_p^{k+1} = x_p^k + v^{k+1}\Delta t. \tag{92}$$

The MPM algorithm exhibits similar behavior as seen in this simplified problem due to the quadrature errors in calculating internal forces (16). This complicated interplay between spatial and temporal errors is one reason why analysis of MPM is not straightforward.

5.3.2. Analysis. Analysis of quadrature errors in the MPM framework [11] calculated errors in internal force when evenly spaced particles sample a material with constant stress:

$$E_f = \int_\Omega \sigma(x)\cdot\nabla\phi_i\,\mathrm{d}\Omega - \sum_p \sigma_p\cdot\nabla\phi_{ip}V_p = \sigma\cdot\left[\int_\Omega \nabla\phi_i\,\mathrm{d}\Omega - \Delta x\sum_p \nabla\phi_{ip}\right], \tag{93}$$

where Δx is the particle spacing, or volume. Here, $\nabla\phi_i$ is either piecewise-constant or piecewise-linear depending on whether piecewise-linear or quadratic B-spline basis functions are used. The bracketed term is equivalent to the error in integrating a piecewise-constant or piecewise-linear function using a composite midpoint rule. This error should be zero if particle voxels align with breaks in the continuity of the integrand; however, in general this is not the case with MPM. Figure 8 shows an example of a particle spanning breaks in the continuity.

The maximum internal force error from Equation (93) when using piecewise-linear basis functions is due to integrating over breaks in the continuity of the piecewise-constant function $\nabla\phi$, as can be seen in Figure 8. This error looks like $E_{\text{jump}} = C_1[[\phi'(0)]]\Delta x$, where $[[\cdot]]$ denotes the jump condition, and C_1 is a constant depending on the integrand. For the case of piecewise-linear basis functions, $C_1 = \frac{1}{2}$. Evaluating the entire integral in (93), taking into account each continuity jump, the upper bound on the total force error E_f (denoted as E_{total}) is

$$E_f \leqslant E_{\text{total}} = 2\sigma\frac{\Delta x}{h}. \tag{94}$$

Performing the same analysis when using quadratic B-splines leads to a jump error of the form $E_{\text{jump}} = C_2[[\phi''(0)]]\Delta x^2$. For our quadratic B-splines, C_2 when integrating $\nabla\phi$ is $\frac{1}{8}$. This leads to

DOI: 10.1002/nme

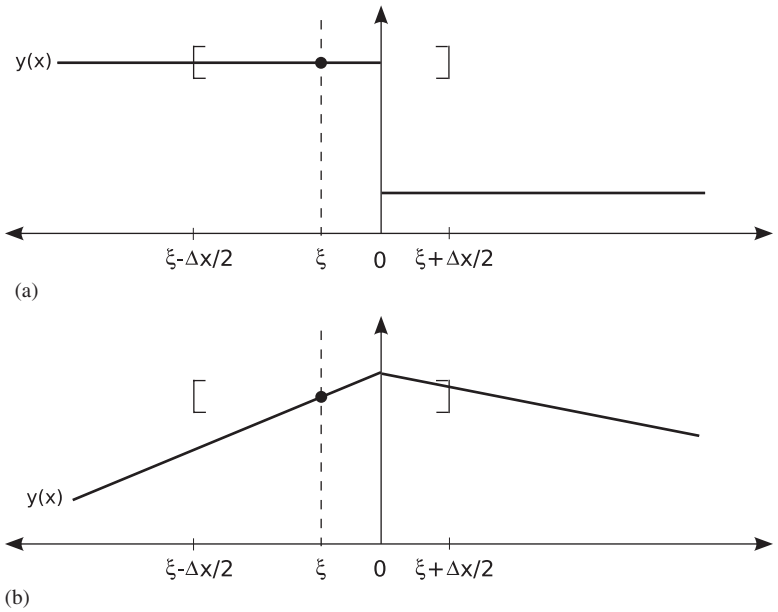

Figure 8. Examples of particles spanning breaks in continuity. Here (a) shows a piecewise-constant integrand, which occurs when integrating $\nabla\phi$ with piecewise-linear basis functions, where (b) shows a piecewise-linear integrand, arising from integrating $\nabla\phi$ with quadratic B-spline basis functions.

an upper bound on the total force error E_f of

$$E_f \leqslant E_{\text{total}} = \sigma \frac{\Delta x^2}{h^2}. \tag{95}$$

This leads to an acceleration error for piecewise-linear on each time-step that looks like

$$\varepsilon = C\alpha\gamma(t), \tag{96}$$

where C is a constant, α is $\Delta x/h$, or the inverse of the number of particles-per-cell (PPC), and $\gamma(t)$ is a function between -1 and 1, specifying how much of the maximum quadrature error is added. The time-update equation is then

$$v^{k+1} = v^k + (a^k + C\alpha\gamma(t^k))\Delta t \tag{97}$$

$$x^{k+1} = x^k + v^{k+1}\Delta t \tag{98}$$

$$= x^k + v^k\Delta t + a^k\Delta t^2 + C\alpha\gamma(t^k)\Delta t^2 \tag{99}$$

where the term $C\alpha\gamma(t)\Delta t^2$ is the error term. Continuing, assuming another error in the acceleration on the next time-step, we get the following:

$$v^{k+2} = v^{k+1} + (a^{k+1} + C\alpha\gamma(t^{k+1}))\Delta t \tag{100}$$

$$= v^k + a^k\Delta t + C\alpha\gamma(t^k)\Delta t + a^{k+1}\Delta t + C\alpha\gamma(t^{k+1})\Delta t. \tag{101}$$

DOI: 10.1002/nme

Now, let us assume that $\gamma(t)$ is the worst possible case for all t, that is $|\gamma(t)|=1$. Then

$$v^{k+2} = v^k + a^k\Delta t + a^{k+1}\Delta t + 2C\alpha\Delta t \tag{102}$$

$$x^{k+2} = x^{k+1} + v^{k+2}\Delta t \tag{103}$$

$$= x^k + v^k\Delta t + a^k\Delta t^2 + C\alpha\Delta t^2 + v^k\Delta t + a^k\Delta t^2 + a^{k+1}\Delta t^2 + 2C\alpha\Delta t^2 \tag{104}$$

$$= x^k + 2v^k\Delta t + 2a^k\Delta t^2 + 3C\alpha\Delta t^2. \tag{105}$$

If we continue our time-steps inductively, we get the following after N steps

$$v^N = v^0 + \sum_{i=1}^{N} a^i\Delta t + NC\alpha\Delta t \tag{106}$$

$$x^N = x^0 + \sum_{j=1}^{N} v^j\Delta t \tag{107}$$

$$= x^0 + Nv_0\Delta t + \sum_{j=1}^{N}\left[\left(\sum_{i=1}^{j} a^i\Delta t\right) + jC\alpha\Delta t\right]\Delta t \tag{108}$$

$$= x^0 + Tv_0 + \sum_{j=1}^{N}\sum_{i=1}^{j} a^i\Delta t^2 + \sum_{j=1}^{N} jC\alpha\Delta t^2 \tag{109}$$

$$= x^0 + Tv_0 + \sum_{i=1}^{N}(N-i+1)a^i\Delta t^2 + \frac{N(N+1)}{2}C\alpha\Delta t^2 \tag{110}$$

$$= x^0 + Tv_0 + \sum_{i=1}^{N}(N-i+1)a^i\Delta t^2 + \frac{1}{2}TC\alpha\Delta t + \frac{1}{2}T^2C\alpha, \tag{111}$$

where T is the final time $T=t_0+N\Delta t$. The global quadrature errors with piecewise-constant $g(x)$ is then

$$E_q = \tfrac{1}{2}TC\alpha\Delta t + \tfrac{1}{2}T^2C\alpha. \tag{112}$$

When piecewise-quadratic basis functions are used, such as B-splines or GIMP functions, the analysis is similar, leading to global quadrature errors of the form

$$E_q = \tfrac{1}{2}TC\alpha^2\Delta t + \tfrac{1}{2}T^2C\alpha^2. \tag{113}$$

Our simplified problem with piecewise-linear f will exhibit similar error behavior as the analysis above for MPM with piecewise-linear basis functions. The following section will give a demonstration of these errors in the simplified problem.

5.3.3. Results. In Section 5.3, the global error for our simplified problem at final time $T=t_0+N\Delta t$, including the effect of quadrature errors, is given by:

$$x^N = x^0 + Tv_0 + \sum_{i=1}^{N}(N-i+1)a^i\Delta t^2 + \frac{1}{2}TC\Delta x\Delta t + \frac{1}{2}T^2C\Delta x. \tag{114}$$

DOI: 10.1002/nme

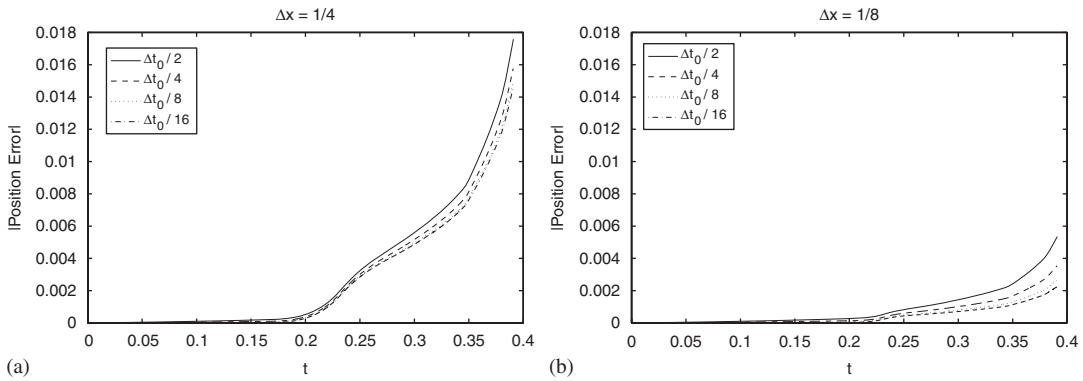

Figure 9. Global errors for two values of Δx and numerous values of Δt plotted on the same axes: (a) $\Delta x = \frac{1}{4}$ and (b) $\Delta x = \frac{1}{8}$.

The first three terms represent the standard Forward-Euler time-stepping method, and the last two terms represent the estimate of quadrature errors on the global position error:

$$\varepsilon = \tfrac{1}{2} T C \Delta x \Delta t + \tfrac{1}{2} T^2 C \Delta x. \tag{115}$$

From this estimate, we would expect global errors to decrease with the decreasing time-step Δt, but to be limited by the last term, which has no dependence on the time-step. We also expect global errors to increase quadratically with final time T and to decrease with the decreasing particle spacing Δx as seen in Figure 9.

While the error estimate in (115) shows the error growing quadratically as the final time T increases, and while the simplified problem was designed to demonstrate this behavior and the results in Figure 9 exhibit this unbounded error, it is worth noting that the global error in many simulations oscillate around the true solution.

5.4. Balancing space and time errors

Until now, we have focused on analysis of errors in simplified problems that demonstrate similar error behaviors as full MPM. Through a better understanding of these component errors, and through numerical demonstrations, we can gain insight concerning the spatial and temporal convergence properties of the method. In this section we will eliminate the compounding of errors in MPM by focusing on single time-step, local truncation errors in the full MPM framework. Using models for the expected behavior of spatial and temporal errors, we will be able to estimate the balancing point (a particular time-step Δt) where these two errors are equal.

5.4.1. Moving-mesh MPM. The velocity update equation in Equation (64) is a straightforward second-order discretization of $\dot{\mathbf{v}} = \mathbf{a}$. This can be seen by performing Taylor series expansions of $\mathbf{v}$ about time t^k:

$$v^{k+1/2} = v^k + \dot{v}^k(\Delta t/2) + \tfrac{1}{2}\ddot{v}^k(\Delta t/2)^2 + \tfrac{1}{6}\dddot{v}^k(\Delta t/2)^3 + \mathcal{O}(\Delta t)^4 \tag{116}$$

$$v^{k-1/2} = v^k - \dot{v}^k(\Delta t/2) + \tfrac{1}{2}\ddot{v}^k(\Delta t/2)^2 - \tfrac{1}{6}\dddot{v}^k(\Delta t/2)^3 + \mathcal{O}(\Delta t)^4. \tag{117}$$

DOI: 10.1002/nme

Subtracting (117) from (116) yields:

$$v^{k+1/2} - v^{k-1/2} = \Delta t\, \dot{v}^k + \tfrac{1}{24}\Delta t^3\, \dddot{v}^k + \cdots. \tag{118}$$

Rearranging terms elucidates to us how this discretization is second-order in time, assuming a is sufficiently smooth:

$$a^k = \dot{v}^k = \frac{v^{k+1/2} - v^{k-1/2}}{\Delta t} + \mathcal{O}(\Delta t^2). \tag{119}$$

And finally, if we measure local truncation errors, we would expect to see third-order behavior:

$$v^{k+1/2} = v^{k-1/2} + \Delta t a^k + \mathcal{O}(\Delta t^3). \tag{120}$$

Again, these are the standard ODE theory results and assume that **a** is known and sufficiently smooth [23]. However, as was shown above, significant spatial errors can exist in MPM. In fact, assuming second-order spatial errors, acceleration will take the form

$$a^k = \tilde{a}^k + c_1 h^2, \tag{121}$$

where $\tilde{a}$ is our calculated acceleration and c_1 is a constant not dependent on h. Substituting (121) into (120) gives us our MPM time-update equation for the centered difference velocity update scheme:

$$v^{k+1/2} = v^{k-1/2} + \Delta t(\tilde{a}^k + c_1 h^2) + c_2 \Delta t^3. \tag{122}$$

Here, the term $c_1 h^2 \Delta t$ represents the spatial contribution to the local truncation error. The term $c_2 \Delta t^3$ is the temporal contribution to the local truncation error. Thus, we would expect a transition point between spatial and temporal errors dominating when

$$c_1 h^2 = c_2 \Delta t^2 \tag{123}$$

which occurs when

$$\Delta t = Ch, \tag{124}$$

with $C = \sqrt{c_1/c_2}$.

5.4.2. Standard MPM. When using standard MPM with piecewise-linear basis functions, we expect first-order spatial errors. Therefore, instead of (121), acceleration will now be

$$a^k = \tilde{a}^k + c_1 h, \tag{125}$$

where $\tilde{a}$ is the calculated acceleration. Substituting (125) into (120) gives us our MPM time-update equation for the centered-difference velocity update scheme within the standard MPM framework:

$$v^{k+1/2} = v^{k-1/2} + \Delta t(\tilde{a}^k + c_1 h) + c_2 \Delta t^3. \tag{126}$$

This differs from (122) in that the spatial error term is first-order, rather than second-order. We would now expect the transition point between spatial and temporal errors dominating at

$$c_1 h = c_2 \Delta t^2, \tag{127}$$

DOI: 10.1002/nme

which occurs when

$$\Delta t = C\sqrt{h}, \tag{128}$$

with $C = \sqrt{c_1/c_2}$.

Quadratic B-spline basis functions, however, still exhibit second-order spatial errors for this problem, even with standard MPM. Therefore, instead of (128), the transition point for standard MPM with B-spline basis functions should still occur when $\Delta t = Ch$, as in (124).

Demonstrations of these transition, or balancing points will be shown in Section 6.4 for both moving-mesh and standard MPM.

6. RESULTS FOR FULL MPM SIMULATIONS

In Section 5, we studied, analyzed, and demonstrated various errors on simplified and decoupled problems. These problems were chosen due to their relative ease of analysis and because they exhibit similar errors to those that exist in a full MPM simulation. In this section, we demonstrate that these same errors exist in a full MPM simulation and have similar behaviors.

6.1. One-dimensional periodic bar

To allow for quantitative measurements of errors, a one-dimensional transient problem with an analytic solution will be used to perform numerical tests. We will use the same one-dimensional periodic bar that we have used in previous MPM tests [12]. The problem we are considering has an assumed analytical displacement on the domain $[0, 1]$, and the resultant deformation gradient of

$$u(X,t) = A\sin(2\pi X)\cos(C\pi t), \tag{129}$$

$$F(X,t) = 1 + 2A\pi\cos(2\pi X)\cos(C\pi t), \tag{130}$$

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

$$b(X,t) = C^2\pi^2 u(X,t)(2F(X,t)^{-2} + 1). \tag{131}$$

The functions u and F are included in (131) only to simplify notation. The constitutive model is a simple 1-D neo-Hookean model, assuming zero Poisson's ratio:

$$\sigma = \frac{E}{2}\left(F - \frac{1}{F}\right). \tag{132}$$

This constitutive model, when combined with the body force given by (131) will lead to the analytical displacement solution in (129). We take advantage of the equivalence between the first Piola–Kirkhhoff and Cauchy stresses in 1-D in the implementation of this test problem.

Figure 10 is a demonstration of what an MPM simulation is expected to produce for the above 1-D periodic bar at various times. The points represent the particle positions, and the updated particle volumes V_p^k are depicted as the width of the surrounding white boxes.

DOI: 10.1002/nme

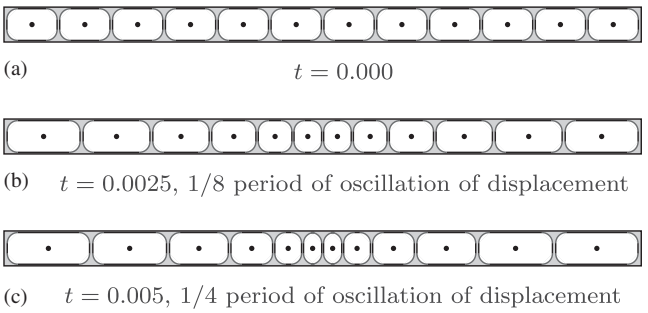

Figure 10. Diagrams of an MPM solution for the 1-D periodic bar at different times with the parameters $E = 10^4$, $\rho_0 = 1$, and $A = 0.1$. The points represent particle positions, and the white boxes show the particle volumes, or widths in 1-D. (a) $t = 0.000$; (b) $t = 0.0025$, $\frac{1}{8}$ period of oscillation of displacement; and (c) $t = 0.005$, $\frac{1}{4}$ period of oscillation of displacement.

6.2. Impact of spatial discontinuities on time stepping

In an attempt to reduce the spatial quadrature errors to a point where the jump error from Section 5.2 can be seen, the 1-D periodic bar was solved using 64 grid cells and 100 PPC. The parameters used were $A = 0.05$ (five percent maximum displacement), $E = 10^4$, $\rho_0 = 1.0$, and a time-step that corresponds to a CFL of 0.1. The problem was solved with a periodic MPM using standard piecewise-linear basis functions.

Standard one-step Forward-Euler time-stepping was used to update the particle positions x_p^{k+1}, but on each step, the two-step method for handling grid crossings (outlined in Section 5.2.2) was used to calculate the two-step particle position $\bar{x}_p^{k+1}$. The single-step jump error was then calculated as $\varepsilon_{\text{jump}}^k = x_p^k - \bar{x}_p^k$. These jump errors were then accumulated to obtain the global jump error $E_{\text{jump}}^k = \sum_k \varepsilon_{\text{jump}}^k$.

Figure 11 shows the result of this simulation. Both the global displacement error and the global jump error were plotted over time. As can be seen in the figure, the global jump error is a relatively small percentage of the overall error, even with 100 PPC.

6.3. Impact of spatial quadrature errors on time stepping

The 1-D periodic bar was simulated again, this time with a more realistic choice for the numbers of PPC. Figure 12 shows an example of the acceleration $a_p = \sum_i a_i \phi_i(x_p)$ felt by a typical particle p for both piecewise-linear and B-spline basis functions when the domain was discretized with four PPC. The particle acceleration a_p is clearly discontinuous for piecewise-linear basis functions and continuous for B-spline basis functions. A close inspection shows that a_i is only C_0 continuous for quadratic B-spline basis functions. This behavior is mainly due to the quadrature errors generated as particles cross grid nodes. This jump in the acceleration is unaffected by the time-step selection.

Spatial convergence studies were then performed on the 1-D bar with various numbers of PPC, and the RMS displacement error was calculated after one full period of oscillation. When the number of PPC is held constant, standard MPM initially converges at $\mathcal{O}(h^2)$, as would be expected in standard finite elements with piecewise-linear basis functions, but soon reaches a point where the quadrature error starts to dominate, and convergence is lost. Increasing the number

DOI: 10.1002/nme

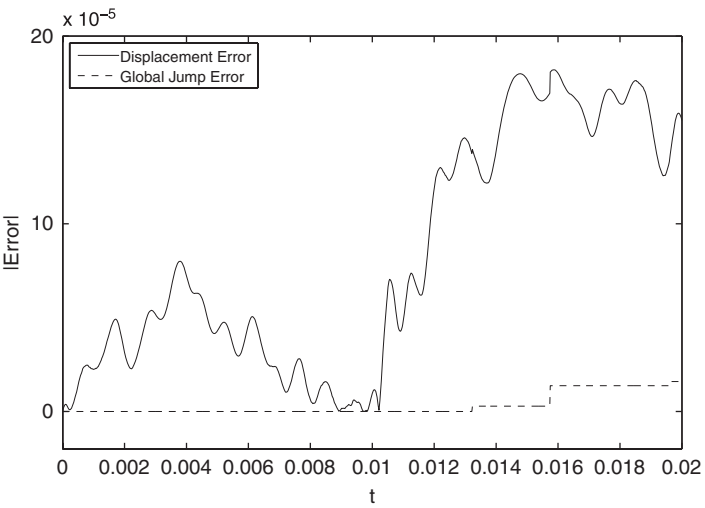

Figure 11. Displacement error and cumulative jump error (calculated as the sum of the difference between single step and two-step time integration) for a single typical particle in a simulation with 64 grid cells and 100 particles-per-cell.

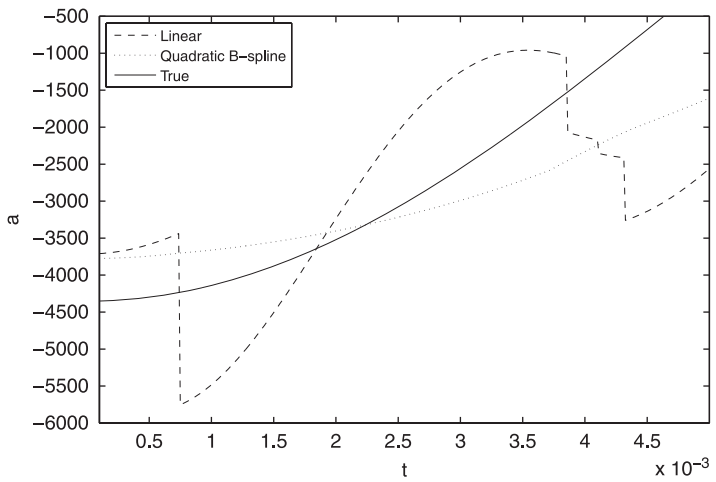

Figure 12. Acceleration felt by a particle for both piecewise-linear and B-spline basis functions in standard MPM. Discontinuities in accelerations occur at grid crossings with piecewise-linear basis functions. With B-spline basis functions, acceleration remains continuous when particles cross grid nodes.

of PPC lowers the point at which the quadrature errors start to dominate. Figure 13(a) shows these results.

Figure 13(b) shows the results of increasing the number of PPC at the same rate as the number of grid cells. In this case, the number of PPC was set to one-quarter the number of grid cells. This ever increasing number of PPC results in a seemingly consistent first-order method. This,

DOI: 10.1002/nme

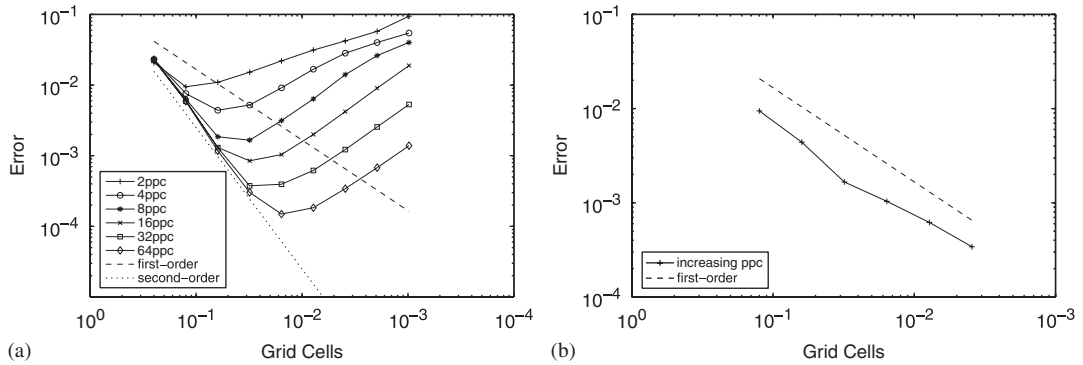

Figure 13. Displacement error with various numbers of particles using standard MPM with piecewise-linear basis functions. Figure (a) shows standard convergence plots with the number of particles-per-cell held constant in each convergence test. Figure (b) shows a convergence test where the number of particles-per-cell is increased along with the number of grid cells. In this case, the number of particles-per-cell was set to one-fourth the number of grid cells, resulting in a total of $N^2/4$ particles.

however, may be prohibitive in practice as the total number of particles is increasing quadratically with the number of grid cells. In other words, the total number of particles in the simulation is $\mathcal{O}(N^2)$, where N is the number of grid cells.

6.4. Balancing space and time errors

Temporal convergence studies so far have not demonstrated second-order convergence as we would expect [7, 17]. The assumption has been that spatial errors are dominating in the regimes being tested. One limitation on the previous studies has been the stability of the solution, requiring the CFL number to be less than unity. Global errors have been reported at given times T, which include the accumulation of errors up to that point. If a simulation fails due to stability reasons, the global error cannot be measured and no information can be gained.

To help to further understand the convergence properties of the scheme, we focus our attention on errors after one time-step—a measure of the local truncation errors. As overall stability of the simulation is not required when looking at a single time-step, we are free to operate in regimes we could not previously test.

The term $c_1 h^2$ in (121) assumes a second-order spatial error when calculating the acceleration. Analysis and demonstrations of second-order spatial errors in MPM arising from the numerical quadrature scheme has been previously shown [11]. To understand the constant associated with this term, we measure the L_2 error of the calculated acceleration on the grid

$$\varepsilon_a^2 = \int_\Omega (\tilde{a}(x) - a(x))^2 \, \mathrm{d}x, \tag{133}$$

where a is the true acceleration field from our manufactured solution and $\tilde{a}$ is the calculated field:

$$\tilde{a}(x) = \sum_i a_i \phi_i(x). \tag{134}$$

DOI: 10.1002/nme

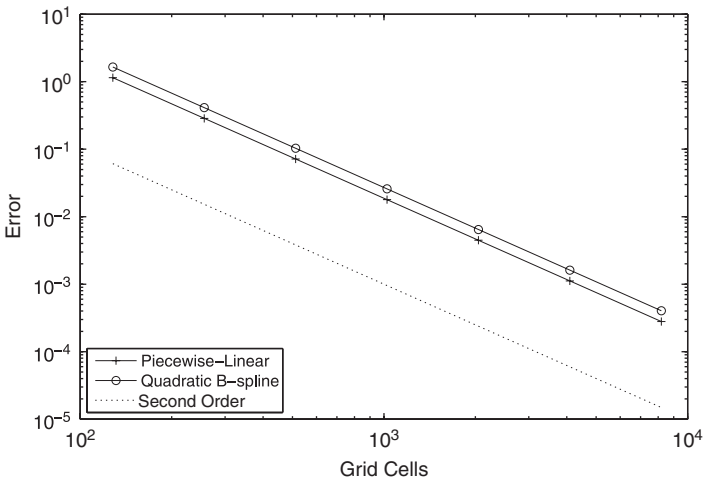

Figure 14. Spatial convergence of grid acceleration L2 errors at time $t = 0.2122 \cdot C$, with moving-mesh MPM, four particles-per-cell (PPC), and periodic boundary conditions for the 1-D axis aligned problem.

Figure 14 shows the spatial convergence of these acceleration errors with both piecewise-linear and quadratic B-spline basis functions at a time corresponding to approximately $\frac{1}{10}$ the period of oscillation (specifically at time $t = 0.2122 \cdot C$). This time was chosen such that particles will have non-zero displacements and velocities.

Again, assuming the acceleration error takes the form $\varepsilon_a = c_1 h^2$, and as the data in Figure 14 are clearly second order, we can choose one point to calculate the value of c_1 for our test case. For example, the acceleration error at 1024 grid cells with piecewise-linear basis functions is calculated to be 1.7895×10^{-2}. Thus, the constant c_1 (as our domain is of length $L = 1$) is

$$c_1 = \frac{\varepsilon_a}{h^2} = 1.8764 \times 10^4. \tag{135}$$

Next, to measure the constant c_2, we measured the L_2 errors in the grid velocity after a single time-step. The data in Figure 15 show third-order temporal convergence for the local truncation error as expected from (122). Knowing that the local velocity truncation error takes the form $\varepsilon_v = c_2 \Delta t^3$, we can choose one point to calculate the value of c_2 for our test case. With piecewise-linear basis functions, the velocity error corresponding to a time-step of $\Delta t = 1.0 \times 10^{-4}$ is 1.1277×10^{-5}. This leads to a value for c_2 of

$$c_2 = \frac{\varepsilon_v}{\Delta t^3} = 1.1277 \times 10^7. \tag{136}$$

Running a single time-step with 2048 grid cells ($h = 4.88 \times 10^{-4}$), we would expect the transition to occur from (124) at $\Delta t = 1.9917 \times 10^{-5}$. Figure 16 shows this experiment and the transition occurs precisely where we expect. Using the same techniques above with the data in Figures 14 and 15, we calculate the transition to occur at $\Delta t = 2.3908 \times 10^{-5}$ with B-spline basis functions. These transition points are very similar, which should not be surprising as the errors for piecewise-linear and quadratic B-splines are comparable.

DOI: 10.1002/nme

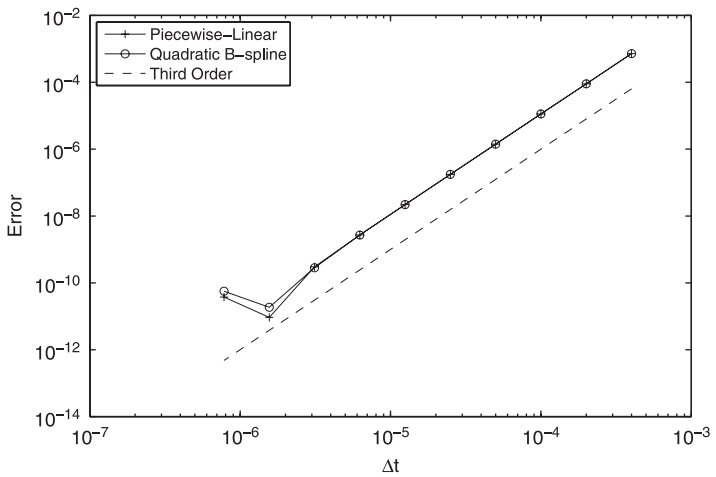

Figure 15. Temporal convergence of local truncation errors in grid velocity with moving-mesh MPM, 2^{20} grid cells, four particles-per-cell (PPC), and periodic boundary conditions for the 1-D axis aligned problem.

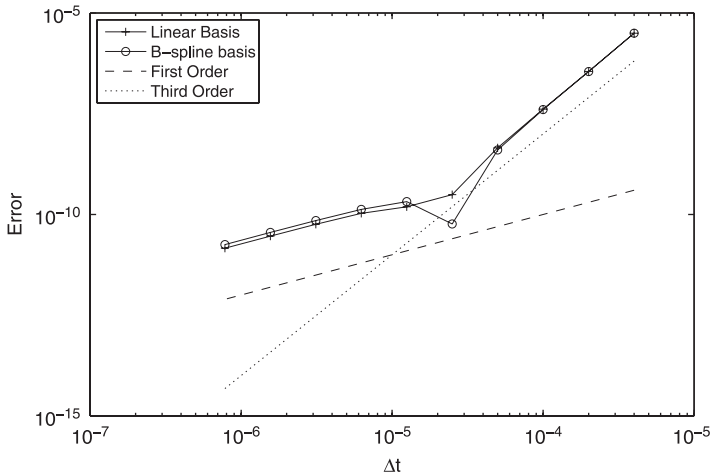

Figure 16. Temporal convergence for local truncation errors in grid velocity with moving-mesh MPM, 2048 grid cells, four particles-per-cell (PPC), and periodic boundary conditions for the 1-D axis aligned problem.

6.4.1. Standard MPM. Figure 17 shows the spatial convergence of acceleration errors with standard MPM. Here, piecewise-linear basis functions initially exhibit second-order spatial convergence with smaller numbers of grid cells when approximation or mass-lumping errors are dominating. However, the asymptotic $\mathcal{O}(h)$ quadrature errors eventually dominate when enough grid cells are used. Quadratic B-spline basis functions still exhibit the expected second-order spatial convergence with standard MPM.

DOI: 10.1002/nme

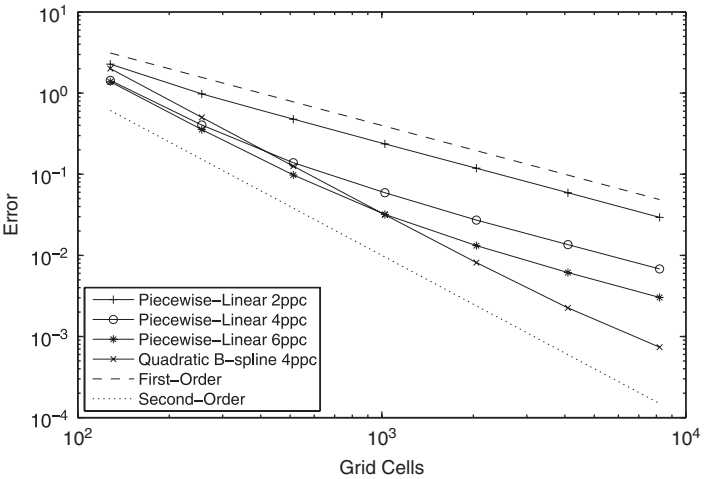

Figure 17. Spatial convergence of grid acceleration L2 errors with standard MPM and periodic boundary conditions for the 1-D axis aligned problem. Piecewise-linear basis functions with various numbers of particles-per-cell (PPC) and quadratic B-spline basis functions with four PPC are shown.

As the acceleration error for piecewise-linear basis functions now takes the form $\varepsilon_a = c_1 h$ in the asymptotic region, and as the data in Figure 17 demonstrate that behavior, we can choose one point to calculate the value of c_1 for our test case. For the four particle-per-cell case, the acceleration error at 1024 grid cells is calculated to be 5.9119×10^{-2}. Therefore, the constant c_1 is

$$c_1 = \frac{\varepsilon_a}{h} = 6.0538 \times 10^1. \tag{137}$$

Next, we again measure the constant c_2 using the errors in grid velocity after a single time-step. The data in Figure 18 show strong third-order local truncation error convergence with larger Δt, as expected from (126). With smaller Δt, a convergence plateau is reached, where the larger spatial errors in standard MPM are starting to dominate. As the local velocity truncation error takes the form $\varepsilon_v = c_2 \Delta t^3$ in the convergent region, we choose one point to calculate the value of c_2. The velocity error corresponding to a time-step of $\Delta t = 1.0 \times 10^{-4}$ is $\varepsilon_v = 1.1278 \times 10^{-5}$. This gives a value for c_2 of

$$c_2 = \frac{\varepsilon_v}{\Delta t^3} = 1.1278 \times 10^7. \tag{138}$$

Using the calculated constants c_1 and c_2 for a reasonable grid resolution of 2048 grid cells, our expected transition point between spatial and temporal errors dominating (128) when piecewise-linear basis functions are used (128) is at $\Delta t = 5.1196 \times 10^{-5}$. Figure 19 shows the results of this test. The transition point between spatial and temporal errors dominating occurs precisely at our estimate. Using a similar procedure as above, the data from Figures 17 and 18, along with (124), the transition point for standard MPM using quadratic B-splines is calculated as $\Delta t = 2.6507 \times 10^{-5}$.

DOI: 10.1002/nme

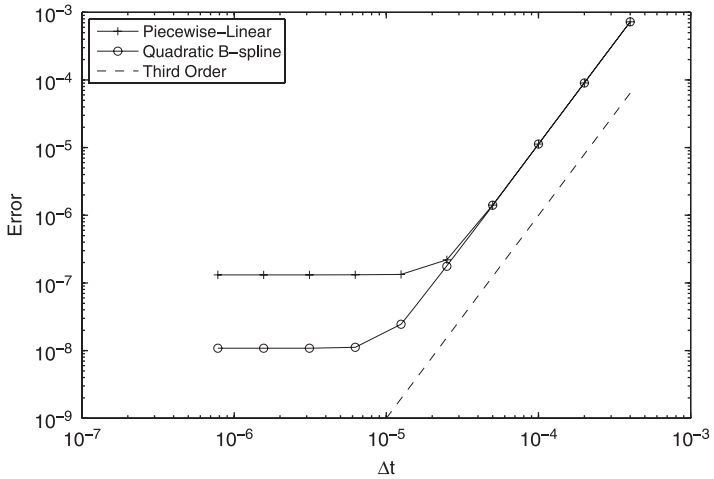

Figure 18. Temporal convergence of local truncation errors in grid velocity with standard MPM, 2^{20} grid cells, four particles-per-cell (PPC), and periodic boundary conditions for the 1-D axis aligned problem.

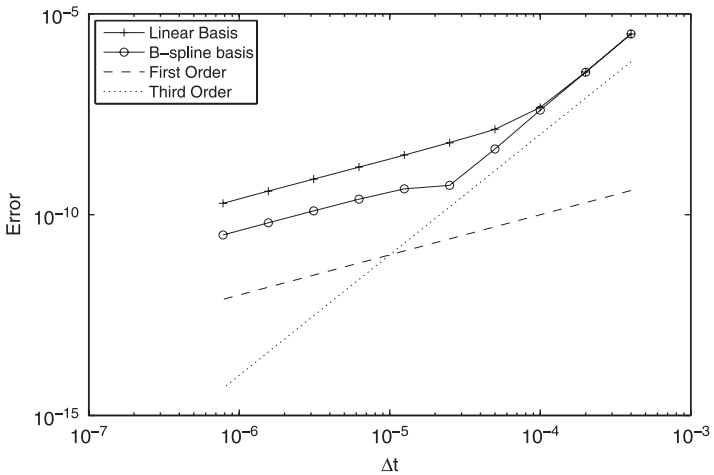

Figure 19. Temporal convergence for local truncation errors in grid velocity with standard MPM, 2048 grid cells, four particles-per-cell (PPC), and periodic boundary conditions for the 1-D axis aligned problem.

7. GUIDELINES

As with most numerical methods, there are numerous errors in MPM of which the practitioner must be aware. These errors include the interpolation errors, mass lumping errors, quadrature errors, standard time-integration errors, and time-integration jump errors. There are also a number of parameters that a practitioner must chose when discretizing a problem with MPM, and many

DOI: 10.1002/nme

of these parameter choices directly affect the above errors. These parameters include the choice of basis function ϕ_i, the grid resolution h, the particle widths Δx, and the time-step Δt.

The interpolation error can be thought of as the finite element-type error—the error associated with what function space $\mathscr{V}$ our set of basis functions $\{\phi_i\}$ spans. This error is well understood in the context of finite element methods and received little treatment in this paper. Generally, this error is $\mathcal{O}(h^k)$ where h is some measure of grid spacing, and k is some measure of the order of the basis functions. Thus, both the choice of basis functions and the grid spacing affect this error. However, even with the simplest piecewise-linear basis functions, this error is $\mathcal{O}(h^2)$. Increasing the order of basis functions will decrease this error, however the spatial error in MPM is already limited to $\mathcal{O}(h^2)$ due to other approximations. Therefore the interpolation error is rarely the limiting factor.

The mass lumping error is also well understood in the context of finite element frameworks [19]. The type of mass lumping employed in MPM is generally considered to provide a $\mathcal{O}(h^2)$ approximation to the complete (unlumped) mass matrix and is therefore rarely the limiting error in our simulations. It is important to note that mass lumping can provide positive benefits with respect to stabilization and monotonicity of the MPM method—things not explored in this work.

Temporal errors also exist in MPM. The standard time-stepping errors are affected by the choice of time-step Δt. In Section 5.2, we focused heavily on the time-stepping jump error—also a function of Δt, but the smoothness of which is also a function of the basis function choice. Smoother basis functions will change the order of this jump error as smoother basis functions result in smoother velocity fields that particles travel through.

In Section 5.3 we examined the impact of spatial quadrature errors on time-stepping. Here, the global displacement errors were a function of α^k, where $\alpha = \Delta x / h$, and k is again a measure relating to the smoothness of the basis functions. Increasing the smoothness of the basis functions from piecewise-linear to piecewise-quadratic (as is the case with the quadratic B-splines used in this paper) increases k from 1 to 2, and as α is typically much less than 1, this significantly reduces the overall effect of the quadrature errors on the final global displacement error. The choice of Δt does not affect these quadrature errors.

One interesting result is when quadrature errors are dominating (as is often the case when using piecewise-linear basis functions), the global displacement error looks like α^k. If the number of PPC remains constant, α also remains constant, and the global displacement error will not be reduced as grid resolution is increased. To have a consistent method, α must be reduced as grid resolution is increased, meaning the number of PPC must increase as the number of grid cells is increased. A demonstration of this is shown in Section 6.3.

The local truncation error results in Section 6.4 suggest that the time-step Δt, where spatial and temporal errors are balanced, is much higher than the stability limit when the explicit centered difference time-stepping scheme is used on this type of a problem. This explains why previous researchers [7] were not able to demonstrate the second-order temporal convergence expected from the method in full MPM simulations. Therefore, with a proper implementation of the centered difference time-stepping scheme, time-stepping errors will have generally converged to the remaining spatial errors by the stability limit, and thus running with a time-step significantly lower than the stability limit has no effect on reducing the temporal errors further.

To demonstrate the effects of following these guidelines while solving an engineering problem, the 1-D periodic bar from Section 6.1 was simulated with two different sets of parameters. In a naive attempt at reducing errors, the problem was simulated with piecewise-linear basis functions, 128 grid cells, four PPC, and a relatively small time-step corresponding to a CFL of 0.1. A second

DOI: 10.1002/nme

Table II. RMS error and run-time for two simulations of a 1-D periodic bar. The second simulation more closely follows the guidelines in this section, resulting in lower error and lower run-time.

	RMS error	Run-time (s)
Simulation 1	2.21×10^{-2}	9.62
Simulation 2	8.39×10^{-5}	0.50

simulation used more expensive quadratic B-spline basis functions, 64 grid cells, the same four PPC, and a larger time-step corresponding to a CFL of 0.9. The RMS error of particle positions and the corresponding run-times are shown in Table II. Here, we see that a selection of the parameters based upon the guidelines provided above results in a much lower error with significantly less computational effort.

8. SUMMARY AND CONCLUSIONS

In this paper we performed three studies on the various errors present in MPM. These studies, which included analysis and demonstrations, were performed on simplified problems that exhibit similar error characteristics to the full MPM framework. We also demonstrated these errors in a full MPM simulation. In the process, we have outlined and demonstrated the moving-mesh MPM algorithm—a fully Lagrangian method that helps to control some of the complexities of the MPM error analysis, mainly quadrature and grid crossing errors.

The first major error in consideration was the time-stepping jump error, and how spatial discontinuities impact these time-stepping errors. We showed that this jump error is second-order with respect to the time-step size, and was not a large contributor to the overall global errors. The second study focused on the impact of spatial quadrature errors on time-stepping. Building on the previous work by the authors, we were able to develop an estimate for global errors arising from the compounding effects of the quadrature errors. And finally, using a model for the error behavior in the method, we were able to estimate a time-step inflection point where spatial and temporal errors are balanced. Time-steps larger than the inflection point result in solutions dominated by temporal errors, whereas smaller time-steps lead to spatial error dominated solutions.

These studies allowed us to formulate guidelines for the practitioner when implementing a similar variant of MPM as used in this paper. The two main guidelines are that the use of smoother basis functions (such as quadratic B-splines) greatly reduces the quadrature errors and therefore reduces global errors in the method, and that time-steps near the stability limit are sufficient in most cases as time-steps near the stability limit already lead to solutions which are dominated by spatial errors. This helps to more fully explain the results showing zero-order global temporal convergence demonstrated by previous researchers [7]. Further guidelines were given, helping the practitioner to understand which algorithm parameters can be expected to affect the various errors in the method.

Previous analysis has shown that the nodal integration and the implicit mass lumping in MPM currently restrict the method to second-order in space [11]. To further reduce errors in the method, more advanced error control techniques beyond simple grid refinement may be required. Therefore,

DOI: 10.1002/nme

a detailed understanding of the balance of space and time errors in the method is key in driving these types of improvements.

ACKNOWLEDGEMENTS

This work was supported by the U.S. Department of Energy through the Center for the Simulation of Accidental Fires and Explosions (C-SAFE), under grant W-7405-ENG-48. In addition, the authors would like to acknowledge the helpful discussions with members of the Utah MPM Group—in particular Drs Jim Guilkey, Phil Wallstedt, and Rebecca Brannon.

REFERENCES

1. Sulsky D, Chen Z, Schreyer HL. A particle method for history-dependent materials. *Computer Methods in Applied Mechanics and Engineering* 1994; **118**:179–196.
2. Sulsky D, Zhou S, Schreyer HL. Application of a particle-in-cell method to solid mechanics. *Computer Physics Communications* 1995; **87**:236–252.
3. Brydon AD, Bardenhagen SG, Miller EA, Seidler GT. Simulation of the densification of real open-celled foam microstructures. *Journal of the Mechanics and Physics of Solids* 2005; **53**:2638–2660.
4. Nairn JA. Numerical simulations of transverse compression and densification in wood. *Wood and Fiber Science* 2006; **38**(4):576–591.
5. Sulsky D, Schreyer H, Peterson K, Kwok R, Coon M. Using the material point method to model sea ice dynamics. *Journal of Geophysical Research* 2007; **112**. DOI: 10.1029/2005JC003329.
6. Guilkey JE, Harman TB, Banerjee B. An Eulerian–Lagrangian approach for simulating explosions of energetic devices. *Computers and Structures* 2007; **85**(11–14):660–674.
7. Wallstedt PC, Guilkey JE. An evaluation of explicit time integration schemes for use with the generalized interpolation material point method. *Journal of Computational Physics* 2008; **227**(22):9628–9642.
8. Brackbill JU, Ruppel HM. FLIP: a method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions. *Journal of Computational Physics* 1986; **65**:314–343.
9. Brackbill JU, Kothe DB, Ruppel HM. FLIP: a low-dissipation, particle-in-cell method for fluid flow. *Computer Physics Communications* 1988; **48**:25–38.
10. Bardenhagen SG, Kober EM. The generalized interpolation material point method. *Computer Modeling in Engineering and Science* 2004; **5**(6):477–495.
11. Steffen M, Kirby RM, Berzins M. Analysis and reduction of quadrature errors in the material point method (MPM). *International Journal for Numerical Methods in Engineering* 2008; **76**(6):922–948. DOI: 10.1002/nme.2360.
12. Steffen M, Wallstedt PC, Guilkey JE, Kirby RM, Berzins M. Examination and analysis of implementation choices within the material point method (MPM). *Computer Modeling in Engineering and Sciences* 2008; **32**(2):107–127.
13. Guilkey JE, Hoying JB, Weiss JA. Computational modeling of multicellular constructs with the material point method. *Journal of Biomechanics* 2006; **39**(11):2074–2086.
14. Zhang L. Dynamic description of texture evolution in polycrystalline nickel under mechanical loading with elastic and plastic deformation via Monte Carlo and Material Point Method simulation. *Ph.D. Thesis*, Colorado School of Mines, 2008.
15. Love E, Sulsky DL. An energy-consistent material-point method for dynamic finite deformation plasticity. *International Journal for Numerical Methods in Engineering* 2006; **65**(10):1608–1638.
16. Love E, Sulsky DL. An unconditionally stable, energy-momentum consistent implementation of the material-point method. *Computer Methods in Applied Mechanics and Engineering* 2006; **195**(33–36):3903–3925.
17. Bardenhagen S. Energy conservation error in the material point method for solid mechanics. *Journal of Computational Physics* 2002; **180**:383–403.
18. Lawson J, Berzins M, Dew PM. Balancing space and time errors in the method of lines for parabolic equations. *SIAM Journal on Scientific and Statistical Computing* 1991; **12**(3):573–594.
19. Hughes TJR. *The Finite Element Method*: *Linear Static and Dynamic Finite Element Analysis*. Prentice-Hall: Englewood Cliffs, 1987.
20. Sulsky D, Kaul A. Implicit dynamics in the material-point method. *Computer Methods in Applied Mechanics and Engineering* 2004; **193**(12–14):1137–1170.

21. Guilkey JE, Weiss JA. Implicit time integration for the material point method: quantitative and algorithmic comparisons with the finite element method. *International Journal for Numerical Methods in Engineering* 2003; **57**(9):1323–1338.
22. Tran LT, Kim J, Berzins M. Solving time-dependent PDEs using the material point method, a case study from gas dynamics. *International Journal for Numerical Methods in Fluids* 2009; DOI: 10.1002/nme.2360.
23. Hairer E, Nrsett SP, Wanner G. *Solving Ordinary Differential Equations I*. Springer: Berlin, 1993.