ANALYSIS-GUIDED IMPROVEMENTS OF THE MATERIAL POINT METHOD

by

Michael Dietel Steffen

A dissertation submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Computing

School of Computing

The University of Utah

December 2009

THE UNIVERSITY OF UTAH GRADUATE SCHOOL

SUPERVISORY COMMITTEE APPROVAL

of a dissertation submitted by

Michael Dietel Steffen

This dissertation has been read by each member of the following supervisory committee and by majority vote has been found to be satisfactory.

	Chair: Robert M. Kirby
	Martin Berzins
	Christopher R. Johnson
	Steven G. Parker
	James E. Guilkey

THE UNIVERSITY OF UTAH GRADUATE SCHOOL

FINAL READING APPROVAL

To the Graduate Council of the University of Utah:

I have read the dissertation of Michael Dietel Steffen in its final form and have found that (1) its format, citations, and bibliographic style are consistent and acceptable; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the Supervisory Committee and is ready for submission to The Graduate School.

Date

Robert M. Kirby
Chair, Supervisory Committee

Approved for the Major Department

Martin Berzins
Chair/Dean

Approved for the Graduate Council

Charles A. Wight
Dean of The Graduate School

ABSTRACT

The Material Point Method (MPM) has shown itself to be a powerful tool in the simulation of large deformation problems, especially those involving complex geometries and contact where typical finite element type methods frequently fail. While these large complex problems lead to some impressive simulations and solutions, there has been a lack of basic analysis characterizing the errors present in the method, even on the simplest of problems. However, like most methods which employ mixed Lagrangian (particle) and Eulerian strategies, analysis of the method is not straightforward. The lack of an analysis framework for MPM, as is found in finite element methods, makes it challenging to explain anomalies found in its employment and makes it difficult to propose methodology improvements with predictable outcomes.

In this dissertation, we provide a formal analysis of the errors in MPM and use this analysis to direct proposed improvements. In particular, we will focus on how the lack of regularity in the grid functions used for representing the solution can hamper both spatial and temporal convergence of the method. We will show how the use of smoother basis functions, such as B-splines, can improve the accuracy of the method. An in-depth analysis of the current time stepping methods will help to explain behavior currently demonstrated numerically in the literature and will allow users of the method to understand the balance of spatial and temporal errors in MPM. Lastly, extrapolation techniques will be proposed to improve quadrature errors in the method.

CONTENTS

ACKNOWLEDGMENTS

First and foremost, I would like to express immense gratitude to my advisor, Mike Kirby, for expending the tremendous time and effort to help me through this process. Mike's involvement in my graduate school career provided me the guidance and motivation to ask, explore, and answer the tough questions required in the research process. His guidance has afforded me the tools and skills to continue in a research career. Numerous life events can occur in the span of five years, and Mike showed more sympathy than anyone could have asked for when school work took a back seat to other more pressing affairs. For that I am also extremely grateful.

I would also like to acknowledge my committee members, especially Jim Guilkey and Martin Berzins, whose unique perspectives kept Mike and myself in check when our narrow view of the world at times lost sight of the big picture. In particular, Jim brought an engineering viewpoint that was crucial in keeping our research grounded and more fully applicable to the engineering community. The Utah MPM group, including Philip Wallstedt, and Jeff Weiss also provided critical input that helped direct our efforts.

There are countless Utah friends which added to the "well rounded" aspect of my education. Though the time spent cycling, camping, canoeing, bowling, go-kart racing, crosswording, barbecuing, fishing, sailing, and snowboarding was not itself productive with respect to my research, it did provide sanity in a world otherwise filled with Greek letters and equations, without which I would never had finished. I have made some lifelong friends along the way that are vastly more important to me than any degree.

I would never have reached this milestone if it were not for my family, especially my Mother, Father, and Sister. From financial assistance to loving support, they were there every step of the way. No mere paragraph can do justice to the

appreciation I have for them. And to Kelly: Thank you. For everything.

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.

CHAPTER 1

INTRODUCTION

The Material Point Method (MPM) [47, 50] is a particle method which represents a material as a collection of *material points* (hereafter referenced as particles) whose deformation is determined by solving Newton's laws of motion for the internal force due to particle interaction. As with all methods of this form, the challenge (and novelty) of the method often comes from the means by which one defines approximations of differential and integral operators given particle data. Although similar in nature to Smoothed Particle Hydrodynamics (SPH) as used in fluid mechanics and to meshfree (or meshless) methods as used in solid mechanics, MPM distinguishes itself as a mixed Lagrangian-Eulerian method which utilizes a regular lattice "background" grid for solving the equations of motion. The material point method attempts to marry the best of both worlds – use of Lagrangian particles for representing material (and its corresponding kinematic and dynamic properties) and use of an Eulerian grid upon which efficient numerical solvers can be built.

MPM and its variants have been shown to be extremely successful and robust in simulating a large number of high-deformation and otherwise complicated engineering problems such as densification of foam [7], compression of wood [35], sea ice dynamics [49], and energetic device explosions [21], to name a few. The most well known of these variants is the Generalized Interpolation Material Point (GIMP) Method [8], of which traditional MPM is a special case. GIMP provides improved accuracy, stability and robustness to simulations through the introduction of particle characteristic functions, which in most cases have the effect of smoothing the grid basis functions. The ability to handle solid mechanics problems involving large deformations and/or fragmentation of structures, which are sometimes problematic

for finite element methods, has lead, in part, to the method's success.

While these simulations are impressive and have pushed the boundaries of high-deformation simulation science where finite element methods often fail, there has been a relative lack of basic error analysis of the method. It is our belief that this lack of formal analysis has hindered the adoption of MPM by a larger audience. This dissertation develops an analysis framework and uses that framework to provide a better understanding of the error properties of the method. The results from this analysis are used to drive improvements to the method. Aside from the improvements detailed in this dissertation, this expanded understanding of the method will provide further benefits, including assisting users of the method in making well informed choices when implementing the method and helping MPM become further accepted by the simulation community.

1.1 Contributions

The goal of this dissertation is to provide an analysis of errors of the Material Point Method and to use knowledge gained from that analysis to drive improvements of the method. In meeting the goal the following contributions have been made:

- *A spatial error analysis framework for the Material Point Method.* By equating the quadrature errors in MPM to integration errors when using a composite midpoint rule with breaks in continuity of the integrand, a new analysis framework is presented which helps understand spatial errors in MPM. This analysis led us to the use of smoother basis functions to improve spatial convergence of the method. These contributions are documented in Chapter 4 and reported in the published peer-reviewed journal article: "Analysis and reduction of quadrature errors in the Material Point Method (MPM)", M. Steffen, R. M. Kirby, and M. Berzins, International Journal for Numerical Methods in Engineering, Volume 76, Number 6, Copyright © 2008, John Wiley & Sons, Ltd.[44]

- *An analysis and demonstration of how implementation details affect the error properties of MPM.* Previously undocumented and unexplored implementation details of MPM, such as boundary conditions and smoothing length parameters within GIMP, can have a major impact on the order of accuracy of the method. The previous analysis techniques were applied to the method to better understand these impacts. This helps the practitioner understand the numerical ramifications of their implementation choices. These contributions are documented in Chapter 5 and reported in the published peer-reviewed journal article: "Examination and Analysis of Implementation Choices within the Material Point Method (MPM)", M. Steffen, P. C. Wallstedt, J. E. Guilkey, R. M. Kirby, and M. Berzins, Computer Modeling in Engineering & Sciences, Volume 32, Number 2, Copyright © 2008, Tech Science Press.[46]

- *An understanding of the balance of errors in MPM through the decoupling of space and time.* The use of techniques in Chapters 4 and 5 combined with the use of moving mesh MPM and measurement of local truncation errors allows a detailed understand of both spatial and temporal errors and how these errors are balanced. This aids practitioners in understanding and deciding where to focus their refinement efforts–in space or in time. These contributions are documented in Chapter 6 and reported in the submitted journal article [45].

- *Improvement of spatial accuracy in the MPM framework.* Until now, the most successful improvements to the accuracy of MPM have come from the use of smoother basis functions and centered difference time-stepping schemes. Here, extrapolation techniques are used to further improve the spatial integration scheme accuracy which provides improvement to overall spatial errors in MPM. These contributions are reported in Chapter 7.

1.2 Organization

This dissertation will proceed as follows: Chapter 2 will present background and relevant work to help give context about where and how MPM fits into the family of particle and meshfree methods. Chapter 3 will provide an overview of MPM,

starting with an explanation of the relevant solid mechanics, showing how MPM discretizes the equations of motion, and finally presenting the variants of MPM and many of the choices one has when implementing the algorithm. Chapter 4 will focus on the spatial errors in MPM. Specifically an analysis of quadrature errors will show how smoother basis functions can affect the spatial convergence rates of the method. Chapter 5 will extend much of this analysis to various flavors of the GIMP algorithm currently implemented in Uintah. An examination of many of the implementation details will show how these choices can affect spatial errors in the method. Chapter 6 will take a close look at the details of the time-stepping algorithms currently used within the MPM framework. An analysis of temporal errors will not only provide a mathematical understanding of behaviors acknowledged in the literature, but will also help practitioners select time-steps which not only satisfy stability constraints, but also balance space and time errors. Chapter 7 will take our enhanced understanding of errors in MPM and develop extrapolation techniques to further improve spatial accuracy. Lastly, Chapter 8 will provide some extensions to the current ideas and future work.

CHAPTER 2

RELEVANT WORK

This chapter provides some of the historic background supporting the Material Point Method. Recently, Brackbill provided an overview of a number of related particle methods [14] which is helpful in understanding where MPM fits in with the particle community. The purpose of this section is not to provide an exhaustive biography of MPM with comparison to all predecessors and methodological siblings. Rather, an attempt will be made to hit the salient points of comparison and contrast with respect to this work.

The Material Point Method (MPM) is a mixed Lagrangian-Eulerian method with moving particles on a background grid. MPM [47, 50] 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 [16, 15]. More recently, Bardenhagen and Kober [8] generalized the development that gives rise to MPM and showed that MPM can be considered a subset of their "Generalized Interpolation Material Point" (GIMP) method. These methods use similar approaches to Smoothed Particle Hydrodynamics (SPH), namely to use an integral representation of field variables, or kernel approximation, when solving the governing equations. For example, in SPH, the evaluation of a field variable at a position $\mathbf{x}$ involves a weighted sum of particle data multiplied by a smoothing kernel function in the neighborhood of $\mathbf{x}$. The extent of the neighborhood, or influence domain, is determined by the smoothing lengths of the particles. MPM similarly employs the idea of particles and particle smoothing kernel functions. However, unlike some particle methods where each particle represents a specific object, or collection of objects, such as electrons or stars, MPM primarily uses particles and their associated volumes to

partition a continuum. In MPM, particles are used to represent the Lagrangian state of a material. To solve the equations of motion, particle functions spread (or project) information to a background grid on which the equations of motion are solved. The background grid cells implicitly define influence domains, effectively eliminating the need for neighbor searches when a fixed Cartesian mesh is used. As mentioned earlier, the Generalized Interpolation Material Point method (GIMP) [8] was developed as an extension to MPM that modifies the type of particle functions used.

Although not derived directly from what are classically considered as meshfree or meshless methods, MPM falls within a general class of meshfree methods and is discussed within the meshfree community since it has both many of the same advantages and many of the same challenges as other meshfree methods [31]. Like many meshfree methods, the primary partitioning of the material does not involve a polygonal tessellation (as in finite elements), but rather some alternative non-mesh-based unstructured representation. However, unlike fully mesh-free methods, such as the Meshless Local Petrov-Galerkin Method (MLPG) [4, 26, 25, 2, 3], MPM utilizes a background mesh to perform differentiation, integration, and solve the equations of motion. The use of a background mesh is still similar to other meshfree methods such as the Element Free Galerkin Method (EFGM) [11]. While the background mesh is formally free to take any form (such as in the Particle Finite Element Method [36], where the mesh is created by a Delaunay triangulation of particles), it is most often chosen for computational efficiency to be a Cartesian lattice (i.e., segments, quadrilaterals and hexahedra in 1-D, 2-D and 3-D respectively). These functions are used, in essence, as a means of discretizing the continuum equations, with the domain of these functions being an alternative (in the sense of versus particles) representation of the deformed configuration of the material. Nodal integration based upon particle positions as is used in other particle methods such as PIC methods [20] is employed during the solution process.

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 has 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 starting to form.

With all numerical methods, the accuracy of the method can depend highly on the accuracy of the numerical quadrature used. For some methods, like the finite element method, defining regions on which to perform integration and the act of numerical integration itself are well established tasks. In meshfree methods, however, information is often stored at seemingly "random" positions, and hence the act of numerical integration becomes more difficult. The issue of numerical integration within meshfree methods has received much attention and is well studied (e.g., see Dolbow and Belytschko [19] where the effect of meshfree shape functions on integration error is reviewed).

The difficulty of various quadrature schemes, especially when using background grid cells as integration domains within Galerkin implementations, is one reason that collocation methods and nodal integration have been explored [9, 17]. It is these collocation schemes that particle methods such as PIC, SPH, and MPM most resemble. Unfortunately, detailed analysis of numerical integration errors within particle methods is limited in the literature. Vshivkov provides a detailed analysis of the projection errors in PIC [52], showing that the error depends both on the grid spacing and the mean number of particles in a cell. Recently, Quinlan et al. [38] looked at the truncation error when approximating spatial derivatives within SPH, showing that the error depends on smoothing length and the ratio of particle spacing to smoothing length. Both the particle-centric nature of the shape functions in both PIC and SPH, and the difference between shape functions in SPH and MPM, make the straightforward application of these results to MPM difficult. It is because of this that an analysis of integration errors specific to MPM is appropriate.

Spatial integration errors were quickly determined to be the limiting factor in the accuracy and stability of MPM. One proposed way to ameliorate the convergence

problems found in MPM was to move away from the idea of nodal integration and instead think of the particles as having extent within the quadrature scheme. Bardenhagen and Kober [8] accomplished this with GIMP by adding particle characteristic functions. There were questions, however, on how to evolve these functions in time within a multidimensional simulation. Since the deformation gradient is only maintained at one point within a particle's voxel, it is unclear that the use of this information to deform particles' voxels is sufficient to maintain a partition of the deformed domain. And, if it were sufficient, it is even more unclear how to accomplish the accurate spatial integration of these deformed voxels. Ma et al. [34] proposed another approach for evolving the particle characteristic functions by adding massless corner particles to explicitly track the deformation of a particle's voxel, or integration domain.

Temporal errors have also received little formal analysis directly applicable to the original MPM method as described by Sulsky et al. Outside of some work by Love and Sulsky [32, 33] analysing an energy consistent implementation of MPM (the second of these papers showing an implicit implementation to be unconditionally stable and energy-momentum consistent), the most comprehensive look at temporal errors in MPM thus far has been by Bardenhagen [6] and subsequently Wallstedt and Guilkey [55] in which rigorous tests were performed comparing various explicit time-stepping algorithms which 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 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. While the comparisons shed much light on expected behavior of the various time-stepping schemes, a detailed analysis of the temporal errors withing the MPM framework still did not exist, nor had the expected temporal convergence properties of the methods been fully demonstrated.

Tran et al. [51] did analyze temporal errors in a variation of MPM developed for the simulation of gas dynamics problems; however, their use of a particle volume normalization procedure precludes the direct application of their conclusions to the original MPM method. Many of their analysis techniques are applicable, and Chapter 6 expands on their analysis to help understand these errors in the context of the original MPM method.

CHAPTER 3

OVERVIEW OF THE MATERIAL POINT METHOD

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

We will begin this chapter with a brief overview of the solid mechanics required to understand the development of the method, including the Galerkin discretization of equations of motion and the constitutive models used. Next, we will show how various discretizations and approximations lead to what is known as MPM. Standard MPM will be reviewed followed by moving-mesh MPM. Various choices for grid basis functions will be reviewed since following chapters devote much attention to the consequences of their use. The Generalized Interpolation Material Point Method (GIMP), an extension to MPM, will then be reviewed. Lastly, various options for implementing kinematic boundary conditions are presented.

3.1 Solid Mechanics Overview

This section provides a brief overview of the basic mechanics and governing equations required to understand the development of the Material Point Method with respect to solid mechanics problems. A basic Galerkin discretization of the equation of motion is presented, followed by a brief explanation of the constitutive model used for all of the problems in this work. A good overview of continuum mechanics, including numerous constitutive equations, is provided by Spencer [43]. Galerkin discretizations are the basis for finite element methods and a full treatment of these techniques can be found in most any finite element method text [57, 27].

3.1.1 Galerkin Discretization of Equation of Motion

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

$$\rho \mathbf{a} = \nabla \cdot \boldsymbol{\sigma} + \rho \mathbf{b}. \tag{3.1}$$

Here, ρ is the material density, $\mathbf{a}$ is acceleration, $\boldsymbol{\sigma}$ is Cauchy stress (assumed to be symmetric in this work), and $\mathbf{b}$ is the acceleration due to body forces (such as gravity). Next, we write acceleration as a linear combination of trial basis functions $\{\phi_i\}$, where $\mathbf{a}(x) = \sum_{i=1}^{N} \mathbf{a}_i \phi_i(x)$. Substituting this into (3.1) and taking the inner product of each term with a test function ϕ_j (ϕ_j is selected from the set of trial basis functions above, and thus the range of j is also from 1 to N. From here on out, the range of the indices are tacit so that the notation can be simplified) leaves us with the Galerkin weak-form of the equation of motion:

$$(\rho \sum_i \mathbf{a}_i \phi_i, \phi_j) = -(\boldsymbol{\sigma}, \nabla \phi_j) + (\rho \mathbf{b}, \phi_j), \tag{3.2}$$

where the notation (a, b) represents the inner product of the functions a and b over our domain Ω, i.e., $(a, b) = \int_\Omega a \cdot b \, d\Omega$. Equation (3.2) represents a linear system written as the following matrix equation[1]:

$$\mathbf{M}\mathbf{a} = \mathbf{f}^{int} + \mathbf{f}^{ext}, \tag{3.3}$$

[1]When written as a linear system, it is tacitly understood that lower case terms are arrays of values, as in (3.3).

where

$$M_{ij} = \int_\Omega \rho \phi_i \phi_j \, d\Omega, \tag{3.4}$$

$$\mathbf{f}_i^{int} = -\int_\Omega \boldsymbol{\sigma} \cdot \nabla \phi_i \, d\Omega, \tag{3.5}$$

and

$$\mathbf{f}_i^{ext} = \int_\Omega \rho \mathbf{b} \phi_i \, d\Omega. \tag{3.6}$$

One method for simplifying (3.3) such that solving a linear system is no longer required is 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 $\mathbf{M}$ [27]; however, we will only consider mass lumping using the row-sum technique, as it is the most prevalent 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 (3.3) reduces to:

$$\mathbf{a}_i = (\mathbf{f}_i^{int} + \mathbf{f}_i^{ext})/m_i. \tag{3.7}$$

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

$$m_i = \sum_j M_{ij} = \int_\Omega \rho \phi_i \sum_j \phi_j \, d\Omega = \int_\Omega \rho \phi_i \, d\Omega. \tag{3.8}$$

The above gives us a procedure to solve for $\mathbf{a}$. While not technically part of a Galerkin discretization, another set of equations are required to describe the behavior of the material in time. Simply put, the acceleration of the material is equivalent to the second time derivative of the displacement of the material $\mathbf{u}$. Or,

$$\ddot{\mathbf{u}}(\mathbf{x}, t) = \mathbf{a}(\mathbf{x}, t). \tag{3.9}$$

One could also solve the equivalent set of coupled first-order ODEs:

$$\dot{\mathbf{v}}(\mathbf{x}, t) = \mathbf{a}(\mathbf{x}, t), \tag{3.10}$$

$$\dot{\mathbf{u}}(\mathbf{x}, t) = \mathbf{v}(\mathbf{x}, t). \tag{3.11}$$

The temporal discretization of the above equations are referred to as the time-stepping method and is discussed in future sections.

3.1.2 Constitutive Models

The above equations of motion in a continuum apply equally to all materials, whether they be steel, plastic, or any other material. What makes various materials unique is their dependence of stress on various other kinematic properties of the body, such as the strain tensor. A set of constitutive equations define this dependence and will allow us to calculate the above stress $\boldsymbol{\sigma}$ given other material properties. Constitutive models are themselves a voluminous subject and are not a focus of this work. Numerous constitutive models can be found in continuum mechanics texts (see [43] for example); however, linear-elastic and neo-Hookean models are the only models considered here.

A multi-D linear elastic model can be given as

$$\boldsymbol{\sigma} = \lambda \mathbf{I}\, \mathrm{tr}\mathbf{E} + 2\mu\mathbf{E}, \tag{3.12}$$

where $\boldsymbol{\sigma}$ is the stress tensor, $\mathbf{I}$ is the identity matrix, λ is Lamé's first parameter, and μ is the shear modulus of the material. The strain tensor $\mathbf{E}$ can be calculated as

$$\mathbf{E} = \frac{1}{2}(\mathbf{F} + \mathbf{F}^T) - \mathbf{I}, \tag{3.13}$$

where $\mathbf{F}$ is the deformation gradient $F_{ij} = \partial x_i / \partial X_j$.

The material model most often used in this work is the 1-D neo-Hookean model which better approximates elasticity with large deformations. This model is written simply (assuming a Poisson's ratio of zero) as

$$\sigma = \frac{E}{2}(F - F^{-1}), \tag{3.14}$$

where E is Young's modulus.

3.2 MPM Discretization

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 versus 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. Figure 3.1 depicts a representation of a typical 2-D MPM problem.

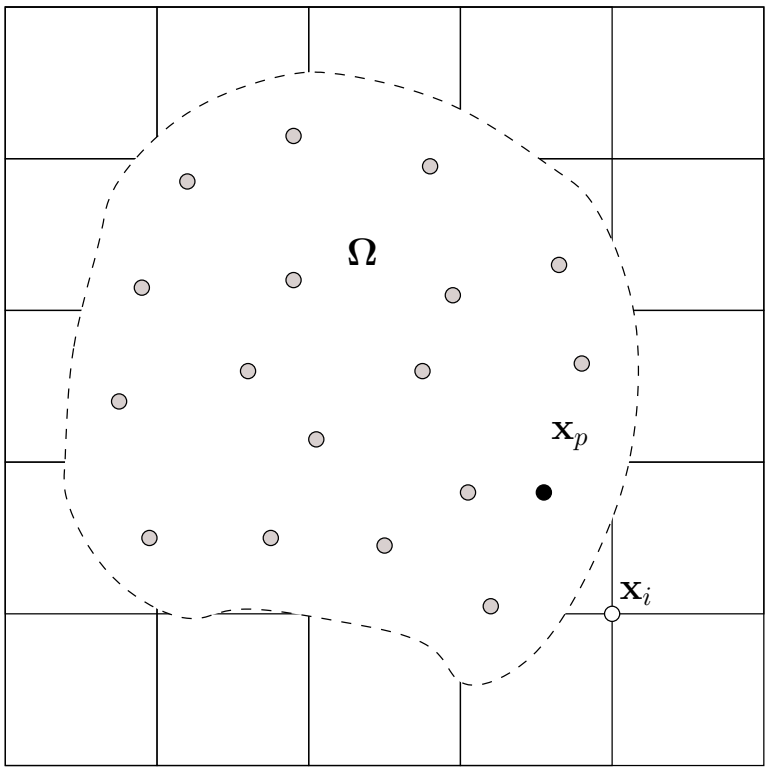


Figure 3.1. Typical 2-D MPM problem setup. The dotted line represents the boundary of the simulated object Ω and each closed point represents a material point used to discretize Ω. The square mesh represents the background grid. Each square in the background grid is a grid cell and grid nodes are located at the corners of grid cells.

3.2.1 Standard MPM

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:

$$(\rho \sum_i \mathbf{v}_i \phi_i, \phi_j) = (\rho \mathbf{v}, \phi_j). \tag{3.15}$$

Solving (3.15) for all i and j is again equivalent to solving the following linear system:

$$\mathbf{M}\mathbf{v} = \mathbf{p}, \tag{3.16}$$

where $\mathbf{v}_i$ is the velocity associated with node i, $\mathbf{M}$ is the mass matrix (3.4), and

$$\mathbf{p}_i = \int_\Omega \rho \mathbf{v} \phi_i \, d\Omega. \tag{3.17}$$

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

$$\mathbf{v}_i = \frac{\mathbf{p}_i}{m_i} = \frac{\int_\Omega \rho \mathbf{v} \phi_i \, d\Omega}{\int_\Omega \rho \phi_i \, d\Omega}. \tag{3.18}$$

A defining feature of the MPM algorithm is the use of nodal integration to approximate the integrals in equations such as (3.18). Given an initial undeformed particle volume V_p^0 and its current deformation gradient $\mathbf{F}_p$, the current particle volume is calculated as

$$V_p = \det(\mathbf{F}_p) V_p^0. \tag{3.19}$$

Using this updated volume, (3.18) is approximated with nodal integration (a quasi-midpoint rule) where field quantities are assumed to be sampled by particle values as follows:

$$\mathbf{v}_i = \frac{\mathbf{p}_i}{m_i} \approx \frac{\sum_p \rho_p \mathbf{v}_p \phi_{ip} V_p}{\sum_p \rho_p \phi_{ip} V_p} = \frac{\sum_p \frac{m_p}{V_p} \mathbf{v}_p \phi_{ip} V_p}{\sum_p \frac{m_p}{V_p} \phi_{ip} V_p} = \frac{\sum_p m_p \mathbf{v}_p \phi_{ip}}{\sum_p m_p \phi_{ip}}, \tag{3.20}$$

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 [48] describe as the consistent mass matrix $M_{ij} = \sum_p \phi_{ip}\phi_{jp}m_p$. Next, the internal force term (3.5) is found by first calculating stress as a function of the constitutive model and the deformation gradient stored with each particle, then by taking the divergence of that stress. Again, nodal integration is used as a means of approximating the integral in the expression:

$$\mathbf{f}_i^{int} = -\int_\Omega \boldsymbol{\sigma} \cdot \nabla\phi_i \, d\Omega \approx -\sum_p \boldsymbol{\sigma}_p \cdot \nabla\phi_{ip} V_p, \tag{3.21}$$

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 (3.6) is then calculated given any body forces as follows:

$$\mathbf{f}_i^{ext} = \int_\Omega \rho\mathbf{b}\phi_i \, d\Omega \approx \sum_p \frac{m_p}{V_p}\mathbf{b}_p\phi_{ip}V_p = \sum_p m_p\mathbf{b}_p\phi_{ip}. \tag{3.22}$$

Using nodal mass m_i and the internal and external forces from (3.21) and (3.22) respectively, we can now calculate nodal accelerations $\mathbf{a}_i$ using (3.7). Grid velocities are then updated with an appropriate time-stepping scheme. Implicit time-stepping schemes exist for MPM [23, 48, 33]; 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:

$$\mathbf{v}_i^{k+1} = \mathbf{v}_i^k + \mathbf{a}_i\Delta t. \tag{3.23}$$

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

$$\nabla\mathbf{v}_p^{k+1} = \sum_i \nabla\phi_{ip}\mathbf{v}_i^{k+1}. \tag{3.24}$$

Lastly, 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:

$$\mathbf{F}_p^{k+1} = (\mathbf{I} + \nabla\mathbf{v}_p^{k+1}\Delta t)\mathbf{F}_p^k, \tag{3.25}$$

$$\mathbf{v}_p^{k+1} = \mathbf{v}_p^k + \sum_i \phi_{ip}\mathbf{a}_i\Delta t, \tag{3.26}$$

and

$$\mathbf{u}_p^{k+1} = \mathbf{u}_p^k + \sum_i \phi_{ip}\mathbf{v}_i^{k+1}\Delta t. \tag{3.27}$$

Equations (3.20)-(3.27) 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:

$$\mathbf{v}_i^{k+\frac{1}{2}} = \mathbf{v}_i^{k-\frac{1}{2}} + \mathbf{a}_i\Delta t, \tag{3.28}$$

$$\nabla\mathbf{v}_p^{k+\frac{1}{2}} = \sum_i \nabla\phi_{ip}\mathbf{v}_i^{k+\frac{1}{2}}, \tag{3.29}$$

$$\mathbf{F}_p^{k+1} = (\mathbf{I} + \nabla\mathbf{v}_p^{k+\frac{1}{2}}\Delta t)\mathbf{F}_p^k, \tag{3.30}$$

$$\mathbf{v}_p^{k+\frac{1}{2}} = \mathbf{v}_p^{k-\frac{1}{2}} + \sum_i \phi_{ip}\mathbf{a}_i\Delta t, \tag{3.31}$$

and

$$\mathbf{u}_p^{k+1} = \mathbf{u}_p^k + \sum_i \phi_{ip}\mathbf{v}_i^{k+\frac{1}{2}}\Delta t. \tag{3.32}$$

A similar staggered central difference method is used for MPM by Sulsky et al. [49], the benefits of which are reviewed in detail by Wallstedt and Guilkey [55].

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 most often in this dissertation is more fully described in Section 3.1.2.

3.2.2 Moving Mesh MPM

The term "moving-mesh MPM" that 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 [7]. See Figure 3.2 for an illustration of the differences between standard and moving-mesh MPM.

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

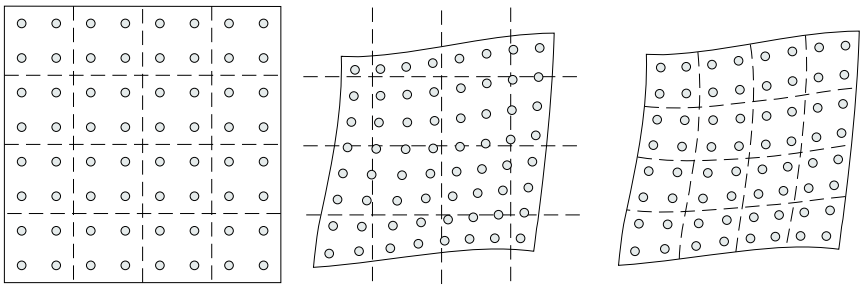
$$m_i = \int_\Omega \rho(\mathbf{x})\phi_i(\mathbf{x})\, d\mathbf{x} \tag{3.33}$$

$$\approx \sum_p \rho_p \phi_i(\mathbf{x}_p) V_p \tag{3.34}$$

$$= \sum_p \frac{m_p}{V_p} \phi_{ip} V_p \tag{3.35}$$

$$= \sum_p m_p \phi_{ip}, \tag{3.36}$$

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



(a) Reference (b) Standard MPM (c) Moving-mesh MPM

Figure 3.2. 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.

$$m_i = \int_{\Omega^0} \rho(\mathbf{X})\Phi_i(\mathbf{X})\det(\mathbf{J})\, d\mathbf{X} \tag{3.37}$$

$$\approx \sum_p \rho_p \Phi_i(\mathbf{X}_p)\det(\frac{\partial \mathbf{x}}{\partial \mathbf{X}})V_p^0 \tag{3.38}$$

$$= \sum_p \frac{m_p}{V_p}\Phi_{ip}\det(\mathbf{I} + \frac{\partial \mathbf{u}}{\partial \mathbf{X}})V_p^0 \tag{3.39}$$

$$= \sum_p \frac{m_p}{\det(\mathbf{F}_p)V_p^0}\Phi_{ip}\det(\mathbf{F}_p)V_p^0 \tag{3.40}$$

$$= \sum_p m_p \Phi_{ip}. \tag{3.41}$$

Here, $\mathbf{J}$ 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 (3.25). However, the definition of the deformation gradient is $\mathbf{F} = \mathbf{I} + \frac{\partial \mathbf{u}}{\partial \mathbf{X}}$ and since displacements are maintained on the grid, $\mathbf{F}_p$ can be directly calculated from $\mathbf{u}_i$:

$$\mathbf{F}_p = \mathbf{I} + \sum_i \nabla_0 \Phi_i(\mathbf{X}_p)\mathbf{u}_i, \tag{3.42}$$

where ∇_0 denotes the gradient with respect to coordinates in the reference frame. 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 1^{st} 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 (3.21) in the reference frame:

$$\mathbf{f}_i^{int} = -\int_\Omega \boldsymbol{\sigma}(\mathbf{x}) \cdot \nabla \phi_i(\mathbf{x})\, d\Omega = -\int_{\Omega_0} \mathbf{P}(\mathbf{X}) \cdot \nabla_0 \Phi_i(\mathbf{X})\, d\Omega_0 \approx -\sum_p \mathbf{P}_p \cdot \nabla_0 \Phi_{ip} V_p^0. \tag{3.43}$$

This internal force calculation differs from standard MPM in that $\nabla_0\Phi$ is evaluated at $\mathbf{X}_p$ in the reference configuration, the 1^{st} 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 particles values, including displacement $\mathbf{u}_p = \mathbf{u}_0(\mathbf{X}_p)$, with $\mathbf{u}_0(\mathbf{X})$ the initial displacement field. Particles should be equally-spaced and aligned with the grid cell boundaries since the major benefits of moving-mesh MPM are only obtained when particles are in these "ideal" positions.

Since grid displacements are maintained and used in (3.42), 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:

$$\mathbf{A}\mathbf{u} = \mathbf{b}, \tag{3.44}$$

where $A_{ij} = (\Phi_i, \Phi_j)$ and $\mathbf{b}_i = \int_{\Omega^0} \Phi_i(\mathbf{X})\mathbf{u}(\mathbf{X})\, d\mathbf{X}$. Continuing with the MPM philosophy, we solve the above equation first by mass lumping $\mathbf{A}$, then by nodal integration. Thus we obtain

$$\mathbf{u}_i = \frac{\mathbf{b}_i}{\sum_j A_{ij}} \tag{3.45}$$

$$= \frac{\int_{\Omega^0} \Phi_i(\mathbf{X})\mathbf{u}(\mathbf{X})\, d\mathbf{X}}{\int_{\Omega^0} \Phi_i\, d\mathbf{X}} \tag{3.46}$$

$$\approx \frac{\sum_p \Phi_{ip}\mathbf{u}_p V_p^0}{\sum_p \Phi_{ip} V_p^0}. \tag{3.47}$$

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

$$\mathbf{u}_i = \frac{\sum_p \Phi_{ip}\mathbf{u}_p V_p^0}{\sum_p \Phi_{ip} V_p^0}. \tag{3.48}$$

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

Solve for mass at grid $$m_i = \sum_p m_p \Phi_{ip} \tag{3.49}$$

Solve for grid velocity $$\mathbf{v}_i^k = \sum_p m_p \mathbf{v}_p^k \Phi_{ip} / m_i \tag{3.50}$$

Solve for external forces $$\mathbf{f}_i^{ext} = \sum_p m_p \mathbf{b}_p \Phi_{ip} \tag{3.51}$$

Solve for internal forces $$\mathbf{f}_i^{int} = -\sum_p \mathbf{P}_p^k \cdot \nabla \Phi_{ip} V_p^0 \tag{3.52}$$

Solve for grid acceleration $$\mathbf{a}_i^k = (\mathbf{f}_i^{int} + \mathbf{f}_i^{ext})/m_i \tag{3.53}$$

Time advance grid velocity $$\mathbf{v}_i^{k+1} = \mathbf{v}_i^k + \mathbf{a}_i^k \Delta t \tag{3.54}$$

Time advance grid displacements $$\mathbf{u}_i^{k+1} = \mathbf{u}_i^k + \mathbf{v}_i^{k+1} \Delta t \tag{3.55}$$

Time advance particle deformation gradient $$\mathbf{F}_p^{k+1} = 1 + \sum_i \mathbf{u}_i^{k+1} \cdot \nabla \Phi_{ip} \tag{3.56}$$

Solve constitutive model $$\mathbf{P}_p^{k+1} = \mathbf{P}(\mathbf{F}_p^{k+1}) \tag{3.57}$$

Time advance particle velocities $$\mathbf{v}_p^{k+1} = \mathbf{v}_p^k + \Delta t \sum_i \mathbf{a}_i^k \Phi_{ip} \tag{3.58}$$

Time advance particle displacements $$\mathbf{u}_p^{k+1} = \mathbf{u}_p^k + \Delta t \sum_i \mathbf{v}_i^{k+1} \Phi_{ip}. \tag{3.59}$$

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 (3.25), where moving-mesh MPM calculates $\mathbf{F}_p$ directly from grid displacements in (3.56).

As in standard MPM, there are many options for implementation of a moving-mesh MPM algorithm. Zhang [56], for example, utilizes a version of moving-mesh MPM which does not re-project the velocity from particles to grid (3.50), but instead uses the time-updated grid velocity from the previous step (3.54).

3.2.3 Choice of Grid Basis Functions

In the discussion above, we purposely did not define precisely what grid basis functions one should use for MPM. In the MPM algorithm outlined above, the choice of ϕ can be considered another option of the method.

Due to their compact support, ease of computation, and partition of unity property, piecewise-linear basis functions are probably the most commonly used choice (with their natural extensions to bi-linear and tri-linear basis functions in two dimensions and three dimensions respectively). The one-dimensional form of the piecewise-linear basis function, and its gradient are given by:

$$\phi(x) = \begin{cases} 1 - |x|/h & : \quad |x| < h \\ 0 & : \quad \text{otherwise,} \end{cases} \tag{3.60}$$

$$\nabla\phi(x) = \begin{cases} h & : \quad -h < x < 0 \\ -h & : \quad 0 < x < h \\ 0 & : \quad \text{otherwise,} \end{cases} \tag{3.61}$$

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 (3.60) and (3.61) in each dimension. For example, in three dimensions, we have:

$$\phi_i(\mathbf{x}) = \phi_i^x(x)\phi_i^y(y)\phi_i^z(z) \tag{3.62}$$

and

$$\nabla\phi_i(\mathbf{x}) = \nabla\phi_i^x(x)\nabla\phi_i^y(y)\nabla\phi_i^z(z). \tag{3.63}$$

It is worth noting that while these piecewise linear basis functions are the same as those used in many low-order finite element methods, the discontinuous nature of $\nabla\phi$ is important in the analysis of MPM since it is a mixed Lagrangian-Eulerian method. Therefore, example versions of Equations (3.60) and (3.61) are explicitly shown in Figure 3.3 (top). In finite element methods, integration over the domain is decomposed into the sum of integrals over elements with quadrature points remaining fixed within elements. In MPM, however, particles act as integration points and are allowed to advect through the domain and across these discontinuities in $\nabla\phi$. The consequences of using particles to integrate discontinuous functions will be explored in Chapter 4.

Recently, the benefits of smoother basis functions have been explored within the MPM framework. As Bardenhagen and Kober [8] described in the development

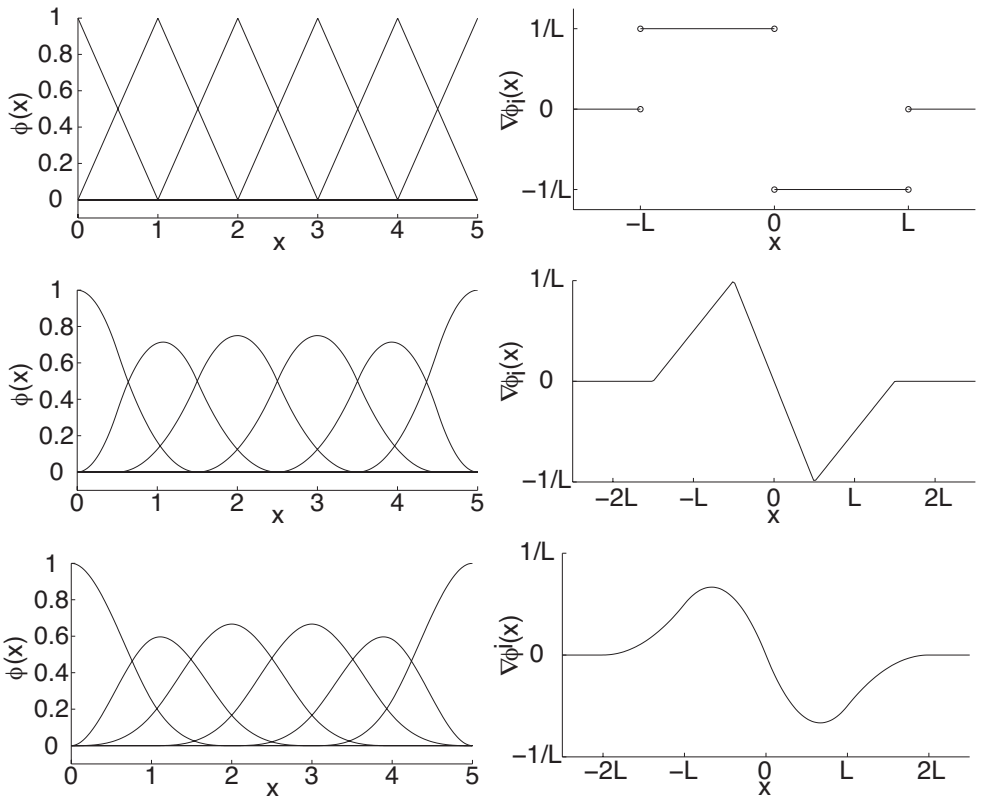


Figure 3.3. Example set of six equally-spaced 1-D basis functions (left column) and corresponding gradients for a selected grid node (right column) for piecewise-linear (top row), quadratic B-spline (middle row), and cubic B-spline (bottom row) basis functions.

of GIMP, lack of regularity in $\nabla\phi$ is conjectured to be the root cause of what is referred to as "grid cell crossing instabilities". As can be seen in Figure 3.4(a), piecewise-linear basis functions are only C_0 continuous at cell boundaries. Tran et al. [51] performed a detailed analysis concerning temporal errors within an MPM fluids framework in which the grid crossing errors arising from use of piecewise-linear basis functions were precisely determined. Chapter 4 will focus on how the lack of smoothness of the standard piecewise-linear basis functions (3.60) causes significant spatial quadrature errors, and how the use of smoother basis functions, such as B-splines, significantly reduces these errors.

In one method to construct the quadratic B-spline basis function for grid node i, the knot vector $\{x_{i-3/2}, x_{i-1/2}, x_{i+1/2}, x_{i+3/2}\}$ is used, where x_i is the position of

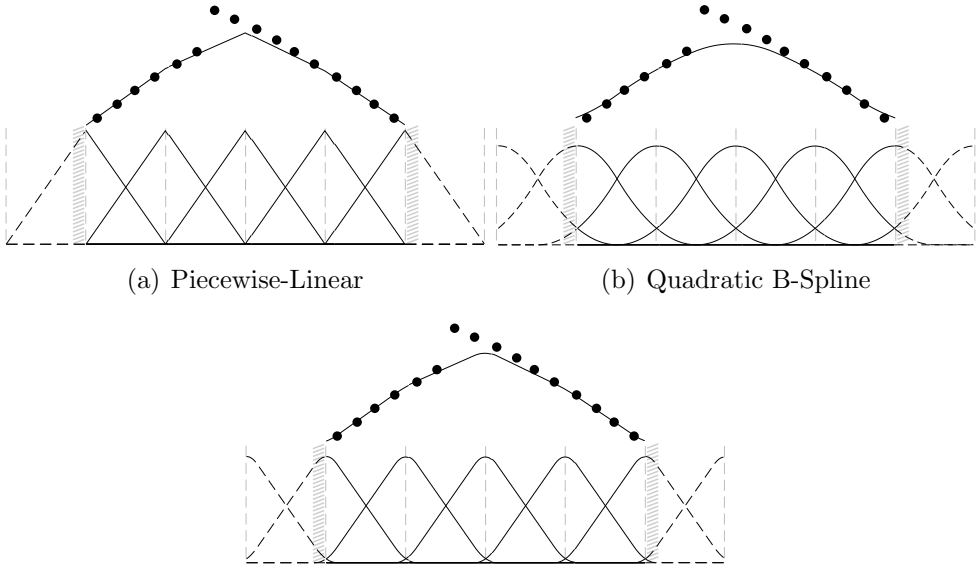
node i, and $x_{i+1/2} = \frac{1}{2}(x_i + x_{i+1})$. However, if node $i = 1$ is one node away from the left boundary x_0, the knot vector $\{x_0, x_{i-1/2}, x_{i+1/2}, x_{i+3/2}\}$ is used. And lastly, the basis function for the border node $i = 0$ is defined by adding two B-splines defined by the knot vectors $\{x_i, x_i, x_{i+1/2}, x_{i+3/2}\}$ and $\{x_i, x_i, x_i, x_{i+1/2}\}$. A similar technique is used for the right boundary. As an example, an internal zero-centered quadratic B-spline has the form

$$\phi(x) = \begin{cases} \frac{1}{2h^2}x^2 + \frac{3}{2h}x + \frac{9}{8} & : \quad -\frac{3}{2}h \leq x \leq -\frac{1}{2}h \\ -\frac{1}{h^2}x^2 + \frac{3}{4} & : \quad -\frac{1}{2}h \leq x \leq \frac{1}{2}h \\ \frac{1}{2h^2}x^2 - \frac{3}{2h}x + \frac{9}{8} & : \quad \frac{1}{2}h \leq x \leq \frac{3}{2}h \\ 0 & : \quad \text{otherwise.} \end{cases} \tag{3.64}$$

This method of constructing quadratic B-spline basis functions not only guarantees the peak of the basis functions to be centered over the grid nodes when nodes are equally spaced, but also guarantees the partition of unity property, $\sum_i \phi_i(x) = 1$ for any $x \in \Omega$, required for the form of the implicit mass lumping often used in MPM. This method also generates interior basis functions that are zero on the boundary, i.e., $\phi_{i\neq 0}(x_0) = 0$. This boundary property combined with the partition of unity property combine such that a boundary basis function evaluates exactly to 1 on the boundary, allowing for easier application of boundary conditions. As with the standard piecewise-linear basis functions, multidimensional basis functions can be created by tensor products of the 1-D constructs.

It is worth noting that a set of these quadratic B-spline basis functions maintain the partition of unity property required by the mass-lumping implicit in the MPM projection functions such as is (3.20). An example set of these basis functions is shown in Figure 3.4(b).

Cubic B-splines may also be used and are, perhaps, somewhat more intuitive since the nodal positions are the values used in the knot vectors. Cubic B-splines are C_2 continuous and span four grid cells in each dimension. The knot vector $\{x_{i-2}, x_{i-1}, x_i, x_{i+1}, x_{i+2}\}$ is used for internal nodes. When node i is one node away from the left boundary, the vector $\{x_{i-1}, x_{i-1}, x_i, x_{i+1}, x_{i+2}\}$ is used. And when node i is the left boundary, the basis function is defined by adding the two B-splines defined by the vectors $\{x_i, x_i, x_i, x_{i+1}, x_{i+2}\}$ and $\{x_i, x_i, x_i, x_i, x_{i+1}\}$. Again,



(a) Piecewise-Linear (b) Quadratic B-Spline

(c) GIMP

Figure 3.4. Example sets of 1-D basis functions used in MPM. Each set of basis functions shows an accompanying set of particles (with height representing velocity) and the corresponding velocity field on the grid after projecting particle values using (3.20). Piecewise-linear basis functions result in piecewise-linear velocity fields with a discontinuity of velocity gradients occurring at grid node locations. Both B-spline and GIMP basis functions result in smoother fields.

construction of right boundary basis functions are similar. An internal zero-centered cubic B-spline has the following form:

$$\phi(x) = \begin{cases} \frac{1}{6h^3}x^3 + \frac{1}{h^2}x^2 + \frac{2}{h}x + \frac{4}{3} & : \quad -2h \leq x \leq -h \\ -\frac{1}{2h^3}x^3 - \frac{1}{h^2}x^2 + \frac{2}{3} & : \quad -h \leq x \leq 0 \\ \frac{1}{2h^3}x^3 - \frac{1}{h^2}x^2 + \frac{2}{3} & : \quad 0 \leq x \leq h \\ -\frac{1}{6h^3}x^3 + \frac{1}{h^2}x^2 - \frac{2}{h}x + \frac{4}{3} & : \quad h \leq x \leq 2h \\ 0 & : \quad \text{otherwise.} \end{cases} \tag{3.65}$$

Figure 3.3 (bottom) shows an example set of cubic B-spline basis functions for six equally spaced grid nodes (left) and their derivative functions (right). Multi-dimensional B-spline basis functions are created by taking the tensor product of 1-D basis functions. These multidimensional spline basis functions will have a rectilinear footprint on the grid and are not the same as the radial spline basis functions used in other mesh-free methods, such as SPH.

The above methods for constructing quadratic and cubic B-splines are used primarily in Chapter 4. One downside to these "boundary" B-spline functions is they do not allow the representation of nonzero slope solutions at the boundary. Another method for constructing B-spline basis functions used in Chapters 5 and beyond is by convolving piecewise-constant basis functions with themselves:

$$\phi = \chi * \chi * \chi / (|\chi|)^2, \tag{3.66}$$

where χ is the piecewise constant basis function:

$$\chi(x) = \begin{cases} 1 & : \quad |x| < \frac{1}{2}l \\ 0 & : \quad \text{otherwise,} \end{cases} \tag{3.67}$$

and l is the width of χ. A separable 3-D B-spline basis function can then be constructed using (3.62).

If we depart from the idea that each grid node corresponds to a single basis function, we can discretize our 1-D domain of length L with n knots, and construct quadratic B-spline basis functions from the open knot vector:

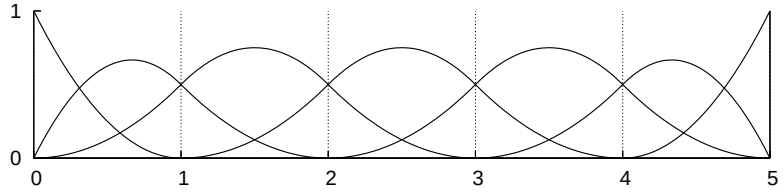
$$[x_0, x_0, x_1, \ldots, x_i, \ldots, x_{n-2}, x_{n-1}, x_{n-1}], \tag{3.68}$$

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

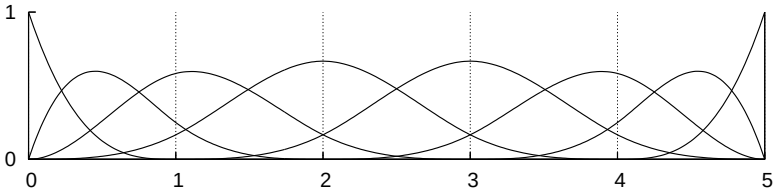
$$\phi_{i,k} = \phi_{i,k-1} \frac{x - x_i}{x_{i+k-1} - x_i} + \phi_{i+1,k-1} \frac{x_{i+k} - x}{x_{i+k} - x_{i+1}}, \tag{3.69}$$

$$\phi_{i,1} = \begin{cases} 1 & : \quad x_i \leq x < x_{i+1} \\ 0 & : \quad \text{otherwise.} \end{cases} \tag{3.70}$$

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 grid node boundaries, allowing for reduced quadrature and grid crossing errors [44]. Figure 3.5 shows examples of these modified boundary B-spline basis functions.



(a) Quadratic B-Spline ($k = 3$)



(b) Cubic B-Spline ($k = 4$)

Figure 3.5. Example sets of modified boundary 1-D B-spline basis functions used in MPM.

3.2.4 Generalized Interpolation Material Point Method (GIMP)

The Generalized Interpolation Material Point (GIMP) Method [8] is an extension to MPM, which takes advantage of the fact that integral equations in MPM take the form:

$$g_i = \sum_p g_p \phi_{ip} = \sum_p g_p \frac{\int_\Omega \phi_i(\mathbf{x}) \delta(\mathbf{x} - \mathbf{x}_p)\, d\Omega}{\int_\Omega \delta(\mathbf{x} - \mathbf{x}_p)\, d\Omega}, \tag{3.71}$$

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

$$g_i = \sum_p g_p \overline{\phi}_{ip}, \tag{3.72}$$

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

$$\overline{\phi}_{ip} = \frac{1}{\int_\Omega \chi_p(\mathbf{x})\, d\Omega} \int_\Omega \phi_i(\mathbf{x}) \chi_p(\mathbf{x})\, d\Omega. \tag{3.73}$$

Equations using $\boldsymbol{\nabla \phi}_{ip}$, such as (3.21) are similarly modified to use a gradient weighting function:

$$\overline{\boldsymbol{\nabla \phi}}_{ip} = \frac{1}{\int_\Omega \chi_p(\mathbf{x})\, d\Omega} \int_\Omega \boldsymbol{\nabla \phi}_i(\mathbf{x}) \chi_p(\mathbf{x})\, d\Omega. \tag{3.74}$$

GIMP is often implemented using standard piecewise-linear grid basis functions (3.60) and piecewise-constant particle characteristic functions:

$$\chi_p = \begin{cases} 1 & : \quad |x| < \frac{1}{2} l_p \\ 0 & : \quad \text{otherwise,} \end{cases} \tag{3.75}$$

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

$$\phi = \begin{cases} 1 - (4x^2 + l_p^2)/(4hl_p) & : \quad |x| < \frac{l_p}{2} \\ 1 - |x|/h & : \quad \frac{l_p}{2} \leq |x| < h - \frac{l_p}{2} \\ \left(h + \frac{l_p}{2} - |x|\right)^2 /(2hl_p) & : \quad h - \frac{l_p}{2} \leq |x| < h + \frac{l_p}{2} \\ 0 & : \quad \text{otherwise,} \end{cases} \tag{3.76}$$

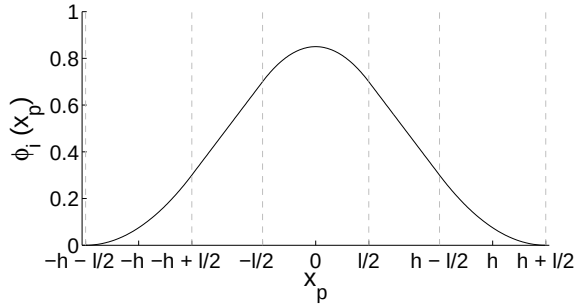
where l_p is the width of the particle characteristic function χ_p. Again, the basis function associated with node i located at position x_i is $\phi_i(x) = \phi(x - x_i)$ and the multi-D weighting function is constructed as the tensor product of the 1-D weighting functions in each direction.

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

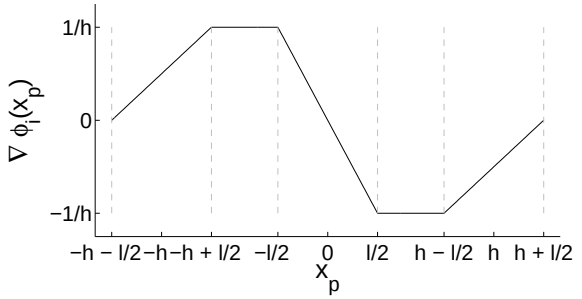
One choice for handling $\mathbf{l}_p$ in a multi-D simulation, and what we will refer to as standard GIMP, is to leave the particle lengths unchanged for all time, i.e., $\mathbf{l}_p = \mathbf{l}_p^0$, where the superscript indicates initial particle size. Standard GIMP weighting functions are then particle specific, but not time-dependent. Another option is uniform GIMP (uGIMP), where a single smoothing length l is used for all particles, for all time. Note that in the case where the initial discretization was performed using uniform particles, standard GIMP would be the same as uGIMP. And lastly, while space can not be tiled in a general multi-D simulation using rectilinear χ_p,

updating $\mathbf{l}_p$ in time to give a rough approximation to the particle's deformed voxel will still be referred to as cpGIMP. The cpGIMP approximation used in this work is $\mathbf{l}_p = \mathbf{l}_p^0 \cdot \text{diag}(\mathbf{F})$, such that the particle size varies through time as dictated by the appropriate diagonal term of the deformation gradient $\mathbf{F}$. Note that l_p here refers to the full particle width, and not the half-width as used in the original GIMP formulation.

Figure 3.6 shows an example 1-D GIMP weighting function $\overline{\phi}_{ip}$ and gradient weighting function $\overline{\nabla\phi}_{ip}$ for a piecewise-constant χ_p with a characteristic length of l. Notice that that while $\overline{\phi}_{ip}$ looks smooth in Figure 3.6(a), the dashed lines show locations of breaks in continuity which become apparent in $\overline{\nabla\phi}_{ip}$ in Figure 3.6(b). These breaks in continuity will become important in later analysis.



(a) GIMP Weighting Function



(b) GIMP Gradient Weighting Function

Figure 3.6. Example GIMP weighting function $\overline{\phi}_{ip}$, and gradient weighting function $\overline{\nabla\phi}_{ip}$ centered at zero using piecewise linear grid basis functions and piecewise constant particle characteristic functions χ_p. Dotted lines denote breaks in the continuity of the functions.

3.2.5 Kinematic Boundary Conditions

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

In traditional MPM, boundary conditions only need be applied on those nodes which coincide with the extents of the computational domain. As illustrated in Figure 3.4(a) nodes beyond those boundaries are not influenced by particles within the domain. This can be considered a result of the zero width of the Dirac delta characteristic functions. Boundary conditions must be applied to the velocity that has been projected to the nodes (3.20), the time advanced velocity (3.23), and the acceleration (3.7). For Dirichlet conditions, this simply means overwriting the calculated values for the velocities with the prescribed values. For the acceleration, some debate exists regarding the proper means of treatment. The usual approach has been to assume that a Dirichlet condition for velocity implies that the acceleration should be zero on those boundary nodes. However, it is also possible that if (3.23) were solved for acceleration:

$$\mathbf{a}_i = (\mathbf{v}_i^{k+1} - \mathbf{v}_i^k)/\Delta t, \tag{3.77}$$

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

While these two approaches to the acceleration seem substantially different, the difference in simulation results is very subtle. Indeed, when both approaches are

tested with the manufactured solution described in Section 5.2.1, the superiority of either is not apparent. Currently, the implementation in Uintah uses the boundary treatment given in (3.77).

In addition to prescribed velocity boundary conditions, "symmetry" boundary conditions are also frequently useful. Symmetry BCs are used to represent a plane of symmetry, which allows the use of a reduced computational domain, or a frictionless surface. They are achieved by simply applying a zero velocity Dirichlet condition on the component of velocity normal to a boundary, while allowing the other components to remain at their computed values. Acceleration is handled in the same manner, with the normal component either zeroed out, or computed as in (3.77).

When using GIMP or B-Spline MPM, there are additional considerations in the applications of the boundary conditions. Namely, because of their increased extents, it is possible for particles to influence, and be influenced by, nodes that lie outside of the simulation domain, (see Figures 3.4(b) and 3.4(c)). In Uintah, these are referred to as "extra" nodes, but may also be called "ghost" nodes by other investigators. Boundary condition treatment of these nodes for Dirichlet conditions is the same as for the regular boundary nodes, namely, their computed values are replaced by prescribed values as described above.

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

CHAPTER 4

ANALYSIS AND REDUCTION OF QUADRATURE ERRORS IN THE MATERIAL POINT METHOD

While MPM and GIMP have been shown to be extremely robust, detailed analysis of the errors in MPM, even for simple problems, is lacking. This is not surprising, as MPM suffers the same challenges as experienced in almost all particle and meshfree methods – finding a common framework or point of reference from which to define quantities like truncation error and quadrature error. Such analysis is not only needed as part of the classic numerical verification process advocated in the engineering sciences, but also for driving improvements of the methodology.

Consider particle grid crossing – one of the motivations given for the development of GIMP. Bardenhagen et al. [8] demonstrate numerically that particles crossing grid cell boundaries cause unexpected computational artifacts, especially in the computation of internal forces. In a similar fashion as is often done in the SPH literature, the problem was ascribed to the choice of particle representation. Not based upon direct analysis of MPM but upon exploiting the analogies between MPM and other particle methodologies, GIMP introduces the idea of particle characteristic functions which have the effect of smoothing the impact of a particle's information on the underlying grid. The hope was that through this generalization one could eliminate these grid crossing artifacts. Although GIMP greatly reduced the impact of grid crossing, it did not eliminate it. The improved computational results of GIMP merit further investigation to explain why it provided tremendous benefit or to postulate why it did not completely solve the problem.

We believe that many of the numerical artifacts seen when employing MPM

can be understood as being the result of the nature of the quadrature rules built into the methodology. This chapter will examine the errors in internal force due to quadrature errors in the MPM framework. Detailed analysis will be performed using the standard piecewise-linear basis functions often used in MPM simulations. The hypothesis of smoother grid basis functions eliminating internal force errors will be tested by extending the analysis to both quadratic and cubic B-spline basis functions within the original MPM framework, showing that they do indeed reduce errors in internal forces. Lastly, we will perform full dynamic simulations, showing that MPM with B-spline basis functions provide convergence properties not achievable with the standard piecewise-linear basis functions.

4.1 Interpretation of Particle Volume

It is our contention that understanding how particle volumes are handled in MPM is necessary for developing a coherent analysis of the method. Every MPM simulation is initialized by discretizing the problem domain Ω with a set of particles. The exact method of discretization can vary depending on the situation, but normally consists of placing particles so that one obtains coverage of the material configuration. One example of a variation might be whether particles reside on the trace of the material configuration or not (which might be preferable for handling boundary conditions).

It is often convenient to imagine partitioning the material frame Ω into a set of initial voxels Ω_p^0 (we use the term voxel to denote the volumetric subset of the domain and reserve the use of the term volume to denote the scalar quantity describing the integral over a voxel) such that $\Omega = \bigcup_p \Omega_p^0$. The superscript is used to emphasize that this occurs at time level zero (i.e., in the material or reference configuration). The concept of a "particle" in MPM is that of one of these voxels; however, the geometric information of the voxel itself is normally not maintained. A position $x_p \in \Omega_p^0$ (usually, but not always, consisting of the geometric centroid of the voxel) and volume $V_p^0 = \int_{\Omega_p^0} d\Omega$ are held by each particle. With each time-step of the MPM algorithm presented in Section 3.2.1, a particle's position and volume

are updated. As the material deforms, the voxels tacitly deform as denoted in Figure 4.1. It is assumed that at any time level, the deformed configuration of the material is represented by the union of the deformed voxels and that the sum of the volumes equals the volume of the deformed material.

Integration within MPM relies upon a particle's position and volume; it is used directly in calculating $\mathbf{f}_i^p$ in (3.21) and is used tacitly in the mass projection since the mass at a grid node i is given by the following relation:

$$m_i = \int_\Omega \rho(x)\phi_i(x)\, d\Omega \approx \sum_p \rho_p \phi_{ip} V_p \approx \sum_p \frac{1}{V_p} m_p \phi_{ip} V_p = \sum_p m_p \phi_{ip}. \tag{4.1}$$

It is important to appreciate that the choice within MPM to only maintain a particle's volume and a single sample point (particle position) dictates the quadrature approximation properties of the method. MPM effectively employs, in the worst case, first-order Riemann integration of field quantities. In the 1-D case where the voxel consists of a line segment of length L and the sample point can be maintained at the center of the interval, the form of the integration reduces to the familiar midpoint rule. In the multi-dimensional case, it is more difficult to show that one can do better than first-order if one only monitors volume and a single position, as the general shape of the voxel is only constrained by the laws of motion and the sampling point is not required to be maintained at the geometric

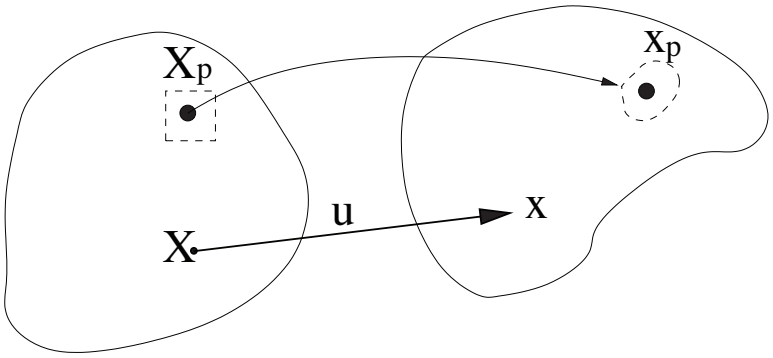


Figure 4.1. Reference (left) and deformed (right) configurations. The dotted lines represent the voxel associated with a particle in both the reference and deformed configurations. Note that the voxel of a particle does not maintain its shape in the deformed configuration, but space can still be tiled.

centroid. *Much of the anomalous behavior exhibited by MPM can be attributed to the quadrature approximation properties of the method.* In fact, many of the proposed improvements to MPM either explicitly or tacitly attempt to control and improve MPM's quadrature behavior.

For example, Bardenhagen et al. [8] try to address the problem of tracing deformed particle voxels with contiguous particle GIMP by using updated volumes in the weighting function $\overline{S}_{ip}$. This is only used in 1-D and assumes space remains tiled with the updated set of Ω_p, x_p is in the center of the update voxel Ω_p, and the width of Ω_p is given by the updated particle volume V_p. This technique still uses information at the particle position to approximate the deformed shape of Ω_p.

Recently, Ma et al. [34] have modified GIMP to approximate the deformed state of each Ω_p, solving the problem of tiling space by placing massless tracking particles at the corner of the initial square voxels. These particles are advected with the grid velocity and are used to define the deformed voxel shape.

One interpretation of both of these previous efforts is that both have focused on improving the means of computing (or maintaining) the measure of integration – that is, attempting to more faithfully represent the voxel. Both, however, accomplish this at a computational cost which degrades the raw efficiency one can gain from the algorithm as presented in Section 3.2.1. In this work, we have taken an alternative view – to acknowledge the errors introduced by the quadrature employed in the current MPM algorithm, to attempt to understand the different contributing factors in that error, and to modify the integrand to help minimize the impact of the error. In particular, our focus is on the impact of changing the grid basis functions used within MPM in a way that reduces the quadrature error and provides consistent convergence results. In the next section, we will present our analysis and our suggested improvements based on our findings.

4.2 Analysis

Most of the grid values in MPM are calculated as approximations to the mass-lumped L^2 projections of data onto the grid basis functions or the gradients of the

basis functions as in (3.20). For example, if a field function $g(x)$ existed over the domain Ω, the values of g_i and ∇g_i at grid node i would be calculated as:

$$g_i = \int_\Omega g(x)\phi_i(x)\,d\Omega \approx \sum_p g_p \phi_{ip} V_p \tag{4.2}$$

$$\nabla g_i = -\int_\Omega g(x)\nabla\phi_i(x)\,d\Omega \approx -\sum_p g_p \nabla\phi_{ip} V_p, \tag{4.3}$$

where g_p is a sample of $g(x)$ at the particle position x_p and V_p is the particle volume. The function g may be of the form $g(x) = f(x)\rho(x)$, with ρ the density, leading to a mass weighted projection of f as in (3.20) with $f_i = g_i/m_i$. If $g(x)$ was a vector valued function, the divergence of $g(x)$ at a grid node would be calculated as follows:

$$(\nabla \cdot g)_i = -\int_\Omega g(x)\cdot\nabla\phi_i(x)\,d\Omega \approx -\sum_p g_p \cdot \nabla\phi_{ip} V_p. \tag{4.4}$$

Note that 1-D Equations (4.2) - (4.4) all have the form $\int_\Omega f(x)\,dx \approx \sum_p f(x_p)V_p$ – that is, they represent quadrature approximations of the integral. In this section, we will discuss the case of 1-D MPM in which we track the interval length L and maintain the sample point (*particle position*) at the centroid of the voxel. In this case, the quadrature in MPM reduces to the midpoint rule.

As a review, the midpoint rule for approximating the integral of $f(x)$ is typically written as $\int_\Omega f(x)\,dx \approx h\sum_{i=1}^N f(x_i)$ where the domain Ω has been subdivided into N regions of size h and x_i is located in the center of region i. The midpoint rule, however, does not require each region to be the same size. If the domain is divided into N regions with individual sizes h_i, the midpoint rule with uneven spacing is written as $\int_\Omega f(x)\,dx \approx \sum_{i=1}^N f(x_i)h_i$ where again, x_i is located in the center of region i.

It should be clear that if x_p is located in the center of the voxel defined by the volume V_p, and if the set of particle voxels tile the domain (or at least tile the non-zero regions of the function being integrated), that the MPM approximation to the integrals in Equations (4.2) - (4.4) are equivalent to a midpoint rule with uneven spacing. There is a problem, however, with applying the standard midpoint rule error analysis to this problem. The standard error analysis assumes continuity

of f in each interval i. Depending on the choice of basis functions ϕ_i, the functions being integrated will not satisfy this continuity condition over the entire domain Ω. Integrating discontinuous functions with the midpoint rule is valid when the division of Ω into regions respects these discontinuities. In finite elements, for example, discontinuities occur at element boundaries, however integration is always performed over individual elements, thus these discontinuities are respected in FEM integration schemes. In MPM, however, the particle voxels will not, in general, respect these discontinuities for all time as particles advect through the domain. Figure 4.2 shows how a particle configuration may respect spatial discontinuities caused by the gradient of basis functions at time t, but will not at time $t + \Delta t$ once the particles have advected through the domain. Therefore, one way to understand the integration errors in MPM is to understand the errors in using the midpoint rule when integrating across discontinuities.

In this section, we will first lay out the simple test problem presented in [8] for understanding errors in internal force computation of MPM. We will then examine the interplay between the midpoint rule nature of MPM quadrature and the choice of the grid basis functions – in particular, examining the commonly used piecewise-linear functions as well as the B-splines we introduced in Section 3.2.3.

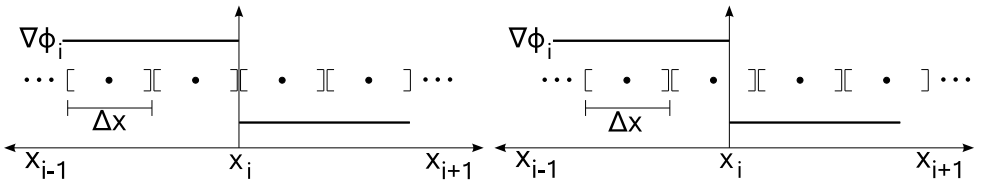


Figure 4.2. Particle configuration at time t (left) respects discontinuities in the gradients of piecewise-linear basis functions, allowing for exact integration using the midpoint rule. Advected particles at time $t + \Delta t$ (right) no longer respect the discontinuities, leading to quadrature errors.

4.2.1 Uniformly Stressed Body in MPM

In MPM, the internal force calculated by (3.21) is an approximation to

$$f_i^{int} = -\int_\Omega \sigma(x) \cdot \nabla\phi_i(x)\, d\Omega \approx -\sum_p \sigma_p \cdot \nabla\phi_{ip} V_p, \tag{4.5}$$

which is similar in form to (4.4). The stress field $\sigma(x)$ can in general take any form, making error analysis difficult. One form of $\sigma(x)$ that allows for easy analysis is the case of a uniformly stressed body with constant particle spacing Δx, where $\sigma_p = \sigma$ for all particles. Bardenhagen et al. [8] note that force imbalances can develop with uniformly stressed bodies when different numbers of particles are in adjacent cells. Since MPM can be thought of as using a midpoint-rule approximation to expressions such as (4.5), the errors in the internal force calculation can be analyzed by looking at errors in midpoint integration when integrating across discontinuities.

For this uniformly stressed body problem, the error in internal force can be evaluated as

$$E_f = \int_\Omega \sigma(x) \cdot \nabla\phi_i\, d\Omega - \sum_p \sigma_p \cdot \nabla\phi_{ip} V_p = \sigma \cdot \left[\int_\Omega \nabla\phi_i\, d\Omega - \Delta x \sum_p \nabla\phi_{ip}\right]. \tag{4.6}$$

The bracketed term in (4.6) not only corresponds to the error in internal force, but also represents the error in integrating $\nabla\phi_i$ using the midpoint rule. Since we are using piecewise polynomial basis functions, $\nabla\phi_i$ is also a piecewise polynomial and the internal force errors can be analyzed by looking at the midpoint integration errors present in (4.6). We will now examine those errors for the three basis function choices presented in Section 3.2.3.

4.2.2 Piecewise-linear Basis Functions

Consider the scenario depicted in Figure 4.2 where the particle configuration does not respect discontinuities in the underlying integrand. Figure 4.3 shows this scenario in more detail, focusing on the particle overlapping the discontinuity. When a background grid with cell width of h is discretized using piecewise-linear basis functions, $\nabla\phi$ is piecewise constant, and discontinuities in $\nabla\phi$ occur at $-h$, 0, and h, with jumps in the polynomial's leading coefficient (see Equation 3.61) of $1/h$, $-2/h$,

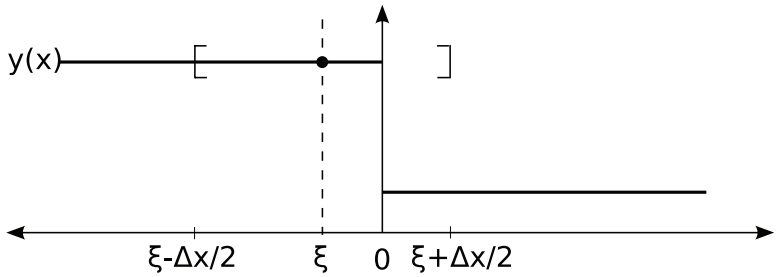


Figure 4.3. Piecewise constant function with a midpoint region spanning a discontinuity at $x = 0$. This is representative of a particle of width Δx centered at ξ whose voxel, Ω_p, crosses a grid cell boundary.

and $1/h$, respectively. The midpoint integration error from integrating across a discontinuity in a piecewise-constant function is shown in detail in the Appendix for the case of the uniformly stressed body above. As is shown by (A.6), the maximum error due to integrating over a discontinuity is given by $E_{jump} = C_1[[\phi'(0)]]\Delta x$, where $[[\cdot]]$ denotes the jump condition, Δx is the particle width, and C_1 is a constant depending on the polynomial. With piecewise-linear basis functions used in MPM, the coefficient C_1 for integrating $\nabla\phi$ is $1/2$.

For particles not crossing over a discontinuity, there will be no error contribution since the midpoint rule can integrate constant functions exactly. Therefore, the only integration intervals contributing to the total error are those which cross the discontinuities. Substituting these individual E_{jump} errors into the bracketed term in (4.6) leads to an upper bound on the total force error E_f of:

$$E_{total} = \sigma \left[\frac{1}{2} \left(\frac{1}{h} + \frac{2}{h} + \frac{1}{h} \right) \Delta x \right] = 2\sigma \frac{\Delta x}{h}. \tag{4.7}$$

The previous analysis only considered the magnitude of the jump when calculating an upper bound on the error. If the sign of the jump is taken into account, the maximum positive error occurs when integration over the discontinuity at 0 is respected by the particle distribution (i.e., no particles cross the discontinuity at 0), but the discontinuities at $-h$ and h are not. Similarly, the maximum negative error occurs when the discontinuities at $-h$ and h are respected, but the discontinuity at 0 is not. These cases lead to errors of

$$E_{total} = \pm\sigma\frac{\Delta x}{h}. \tag{4.8}$$

4.2.3 Quadratic B-spline Basis Functions

Consider the scenario depicted in Figure 4.4. For quadratic B-spline basis functions, $\nabla\phi$ is piecewise-linear and discontinuities in $\nabla^2\phi$ at $-3h/2$, $-h/2$, $h/2$, and $3h/2$, with jumps in second derivative (from Equation 3.64) of $1/h^2$, $-3/h^2$, $3/h^2$, and $-1/h^2$, respectively. The midpoint integration error from integrating across a discontinuity in a piecewise-linear function is shown in detail in the Appendix. As is shown by (A.14) the maximum error due to integrating over a discontinuity is given by $E_{jump} = C_2[[\phi''(0)]]\Delta x^2$, where $[[\cdot]]$ denotes the jump, Δx is the particle width, and C_2 is a constant depending on the polynomial. With the quadratic B-spline basis functions used in MPM, the coefficient C_2 for integrating $\nabla\phi$ is 1/8.

Again, for particles not crossing over a discontinuity, there will be no error contribution since the midpoint rule can integrate linear functions exactly. Therefore, the only integration intervals contributing to the total error are those crossing discontinuities. Substituting these individual E_{jump} errors into the bracketed term in (4.6) leads to an upper bound on the total force error E_f of

$$E_{total} = \sigma\left[\frac{1}{8}\left(\frac{1}{h^2} + \frac{3}{h^2} + \frac{3}{h^2} + \frac{1}{h^2}\right)\Delta x^2\right] = \sigma\frac{\Delta x^2}{h^2}. \tag{4.9}$$

Taking the signs of the jump in second derivatives into consideration, the

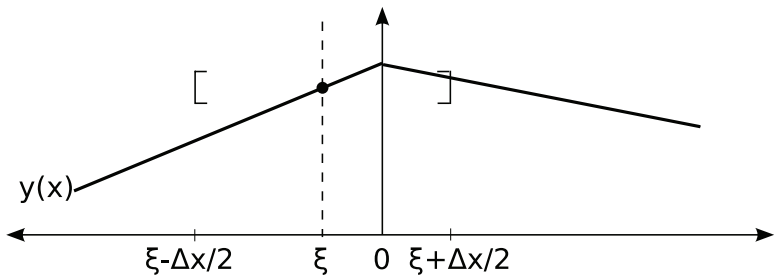


Figure 4.4. Piecewise-linear function with a midpoint region spanning a discontinuity in y' at $x = 0$. This is representative of a particle of width Δx centered at ξ whose voxel, Ω_p, crosses a grid cell boundary.

maximum positive error occurs when the particle distribution is respectful of discontinuities when integrating over the discontinuities at $-h/2$ and $3h/2$ (those corresponding to a negative a^*), but not respectful at $-3h/2$ and $h/2$. The maximum negative error occurs when the opposite is true. Once again, these cases lead to errors which are half of the maximum error calculated using the magnitude of the jump, and are given by:

$$E_{total} = \pm\sigma\frac{\Delta x^2}{2h^2}. \tag{4.10}$$

4.2.4 Cubic B-spline Basis Functions

Consider the scenario depicted in Figure 4.5. For cubic B-splines, $\nabla\phi$ is piecewise-quadratic and discontinuities in $\nabla^3\phi$ occur at $-2h, -h, 0, h, 2h$, with jumps in the third derivative (see Equation 3.65) of $1/2h^3$, $-2/h^3$, $3/h^3$, $-2/h^3$, and $1/2h^3$, respectively. The midpoint integration error from integrating across a discontinuity in a piecewise-quadratic function is shown in detail in the Appendix. As can be seen by (A.3), the maximum error due to integrating over a discontinuity is given by $E_{jump} = C_3[[\phi'''(0)]]\Delta x^3$, where $[[\cdot]]$ denotes the jump, Δx is the particle width, and C_3 is a constant depending on the polynomial. With the cubic B-spline basis functions used in MPM, the coefficient C_3 for integrating $\nabla\phi$ is $1/24$.

The total maximum error from integrating across discontinuities is then given by:

$$E_{total} = \sigma\left[\frac{1}{24}\left(\frac{1}{2h^3} + \frac{2}{h^3} + \frac{3}{h^3} + \frac{2}{h^3} + \frac{1}{2h^3}\right)\Delta x^3\right] = \sigma\frac{\Delta x^3}{3h^3}. \tag{4.11}$$

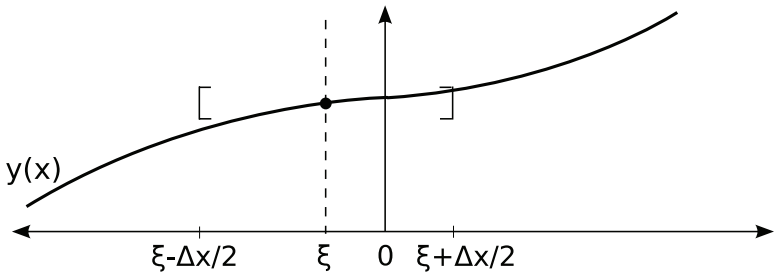


Figure 4.5. Piecewise-quadratic function with a midpoint region spanning a discontinuity in y'' at $x = 0$.

However, $\nabla\phi_i$ is quadratic between the discontinuities and the midpoint rule can not exactly integrate quadratic functions. As a review, if $f \in C^2[a,b]$, then for some μ in (a,b), the composite midpoint error with sub intervals of size Δx is given by [39]

$$E = \frac{(b-a)}{24}\Delta x^2 f''(\mu). \tag{4.12}$$

For our piecewise-cubic ϕ_i, the value of midpoint error for integrating $\nabla\phi_i$ in the four separate regions (see Equation 3.65) is given by

$$\begin{aligned} E &= \frac{1}{24}\Delta x^2 \left[h_1\phi'''(\mu_1) + h_2\phi'''(\mu_2) + h_3\phi'''(\mu_3) + h_4\phi'''(\mu_4)\right] \\ &= \frac{1}{24}\Delta x^2 \left[h_1\frac{1}{h^3} - h_2\frac{3}{h^3} + h_3\frac{3}{h^3} - h_4\frac{1}{h^3}\right], \end{aligned} \tag{4.13}$$

where h_1, h_2, etc. are the size of the integration intervals in the different regions after sub intervals crossing the discontinuities have been removed. There exist many arrangements of particles such that $h_1 = h_2 = h_3 = h_4$ in which case the bracketed term in (4.13) goes to zero, leaving (4.11) as the only source of error. In general, however, the total error from integrating $\nabla\phi_i$ is the sum of (4.11) and (4.13) and since h_1, h_2, h_3, and h_4 are $\mathcal{O}(h)$ the total error is $\mathcal{O}(\sigma\Delta x^2/h^2)$.

We see the internal force error improves when using quadratic B-spline basis functions instead of standard piecewise-linear basis functions. The quadratic B-splines show an error which is $\mathcal{O}(\Delta x^2/h^2)$, while the piecewise-linear basis functions have an error of $\mathcal{O}(\Delta x/h)$. Using even higher order splines, such as cubic B-splines, may give further improvement for some special particle configurations, but in general the composite midpoint integration error limits the error to the same order as with quadratic B-splines.

Brackbill[14] makes a similar observation concerning PIC methods when, building upon theoretical results presented by Vshivkov [52], he states the error of the PIC method is bounded by:

$$\varepsilon \leq C_1\left(\frac{\delta}{h}\right)^2 + C_2 h^2, \tag{4.14}$$

where δ is the particle spacing, h is the mesh spacing, and C_1 and C_2 are constants depending only on the smoothness of the data. Here, the quantity δ/h is a measure of the inverse of the number of particles-per-cell (PPC).

This result follows from Vshivkov's earlier analysis [52] where he calculates the error, δ_k, in the charge density at node k as calculated with PIC. One would expect this error to be analogous to measuring the error in the projection of particle information to the grid in MPM (such as the projection of mass) since the piecewise-linear mesh kernel functions Vshivkov assumes in his analysis are the same as the grid basis functions used in standard MPM. His result states that the error is bounded by:

$$\delta_k \leq \left(\frac{3\rho_{\text{av}}^2}{2\rho_{\text{min}}} + h\frac{\rho_{\text{av}}^2 \rho_{\text{max}}}{6\rho_{\text{min}}^3} \left| \frac{\partial \rho}{\partial x} \right|_{\text{max}} \right) \frac{1}{N^2} + \frac{h^2}{12} \left| \frac{\partial^2 \rho}{\partial x^2} \right|_{\text{max},} \tag{4.15}$$

where N is the average number of particles in a cell. The first term on the right-hand-side relates to the "quadrature" error as a consequence of number of particles and grid spacing and the second term relates to the "approximation" error as a consequence of grid spacing (and tacitly the choice of basis functions).

These results for PIC demonstrate the interplay between approximation error (based upon the choice of the basis functions) and the quadrature error – results that are consistent with and indeed motivated the current work.

4.3 Results

In this section we now attempt to use the perspective provided in Section 4.1 and analysis provide in Section 4.2 to explain common test cases presented in the MPM literature.

4.3.1 Uniformly Stressed Body

We first begin by revisiting the uniformly stressed example mentioned in the previous section. First, we must present a algorithmic way of setting up the problem. Consider the diagram given in Figure 4.6. To describe any arrangement of uniformly spaced particles surrounding a grid cell, we start by selecting a particle spacing Δx less than the cell width h, with $b = \Delta x/h$. Here, b is a fractional

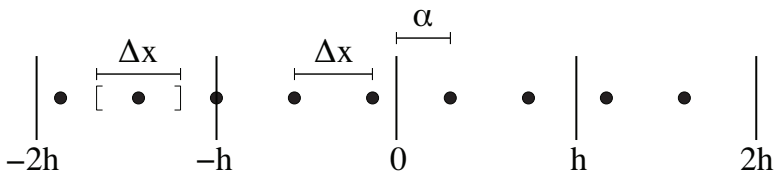


Figure 4.6. Example grid and particle arrangements used for the uniformly stressed body test.

measure of the inverse of the number of particles-per-cell (PPC). Next, place a particle at a location of $\alpha \in [0, \Delta x]$. Define $a = \alpha/\Delta x$ (a percentage shift). Next, fill the region $[-2h, 2h]$ with particles, maintaining the particle spacing Δx. Let the grid span the region $[-2h, 2h]$ with five grid nodes, thus the grid locations will be $x_i = -2h, -h, 0, h, 2h$. Now, give all the particles constant stress, $\sigma_p = \sigma$, and volume $V_p = \Delta x$. We will consider calculation of f_i^{int} on grid node $i = 2$ (the center node). Since stress and volume are constant, we would expect $f_2^{int} = 0$.

The internal force in MPM is calculated as follows:

$$f_i^{int} = \sum_p \sigma_p \cdot \nabla \phi_i(x_p) V_p. \tag{4.16}$$

For piecewise-linear basis functions, this becomes:

$$f_2^{int} = \sum_{p \in [-h,0]} \sigma \frac{-1}{h} \Delta x + \sum_{p \in [0,h]} \sigma \frac{1}{h} \Delta x \tag{4.17}$$

$$= N_2 \sigma \frac{1}{h} \Delta x - N_1 \sigma \frac{1}{h} \Delta x, \tag{4.18}$$

where N_1 is the number of particles in the region $[-h, 0]$, and N_2 is the number of particles in the region $[0, h]$. With this problem setup, there will either be equal number of particles on both sides of the grid nodes, or $N_2 - N_1 = \pm 1$. Thus, f_2^{int} takes on one of three values: $\sigma \frac{1}{h} \Delta x$, 0, or $-\sigma \frac{1}{h} \Delta x$, depending on particle arrangement. This can be seen in the contour plot shown in Figure 4.7 (top). Note that the error of $\sigma \frac{1}{h} \Delta x$ is half of the maximum analytical error shown in (4.7) which only considered the magnitudes of errors when particles overlapped discontinuities in $\nabla \phi_i$. If signs of errors were taken into account the maximum error would be $\sigma \frac{1}{h} \Delta x$.

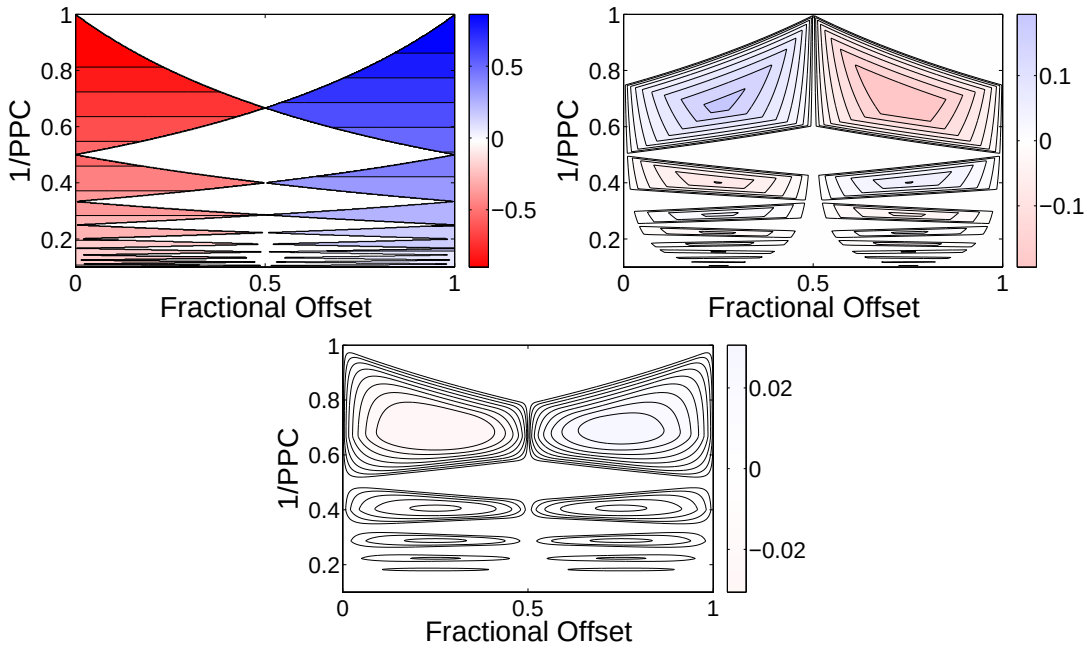


Figure 4.7. Plots of internal force error $|f_2^{int}|$ for a uniformly stressed body discretized with evenly spaced particles. Various particle spacings, b, and offsets, a, are shown using standard piecewise-linear (top), quadratic B-spline (middle) and cubic B-spline (bottom) basis functions. The figures show the maximum error decreases and the convergence rate of the error improves as the continuity of the basis functions is increased. All plots use a consistent color scale.

Several things can be observed in Figure 4.7. First, note that there are combinations of fractional offset a and inverse particles-per-cell b which yield zero error. These combinations consist of two things: (1) when voxel boundaries line up with element boundaries such that there is no jump error term (like in the case of fractional offset $a = 0.5$ where there is a line of zero error for all choices of inverse particles-per-cell b) and (2) when symmetries in the particle positions cause cancellations in the error due to signs of the jumps. The second observation that can be seen in Figure 4.7 is that when quadratic B-spline basis functions are used, the magnitude of the maximum error (on the order of .25) is much less than with piecewise-constant basis functions (on the order of 1.0), and the error approaches zero much faster as the measure of particle spacing b decreases and the number of particles-per-cell (PPC) increases. When cubic B-spline basis

functions are used, the magnitude of the maximum error is again much lower (on the order of .04) and the error approaches zero faster than with quadratic B-splines or piecewise-linear basis functions. This suggests that the maximum error decreases and the convergence rate of the error improves as the continuity of the basis functions is increased.

To better understand the convergence of f_2^{int}, for each b value in Figure 4.7 the maximum error over the fractional offset a was tabulated and plotted in Figure 4.8 on log-log graphs. The top plot shows the error for evenly-spaced particles while the bottom plot shows particles in the same configuration, but randomly perturbed up to 40% of the measure of particle spacing b to the left or right from their nominal positions. Errors which are $\mathcal{O}(\Delta x)$ for piecewise-linear and $\mathcal{O}(\Delta x^2)$ for quadratic B-splines agree with the analysis from the previous section. The cubic B-splines demonstrates errors of $\mathcal{O}(\Delta x^3)$.

The $\mathcal{O}(\Delta x^3)$ behavior of the cubic B-splines are due to the globally uniform spacing of particles in the test, even when the particles are perturbed. This uniform spacing of particles exploits the symmetry in ϕ and the $\mathcal{O}(\Delta x^2)$ error from (4.13) cancels out. Another random test was run where particles were randomized in a global sense (the left half of the domain might have more particles than the right half of the domain) and the results are shown in Figure 4.9. Here, for larger values of b (that is, fewer particles-per-cell (PPC)), the internal force has $\mathcal{O}(x^2)$ behavior. As b decreases, however, there exist more particles contributing to the integration and the $\mathcal{O}(x^3)$ behavior returns.

4.3.2 Test Problem With Dynamic Traction Boundary Conditions

Previous analysis of the spatial convergence properties of MPM has been performed using quasi-static computations and comparisons with analytical solutions [8]. Since MPM is often used for dynamics problems, a 1-D test case with an analytic transient solution was developed.

Given a bar of length l, fixed at $x = 0$, free at $x = l$, with a Young's modulus of E, density of $\rho = E$ (wave-speed of 1.0), driven by a forcing function of

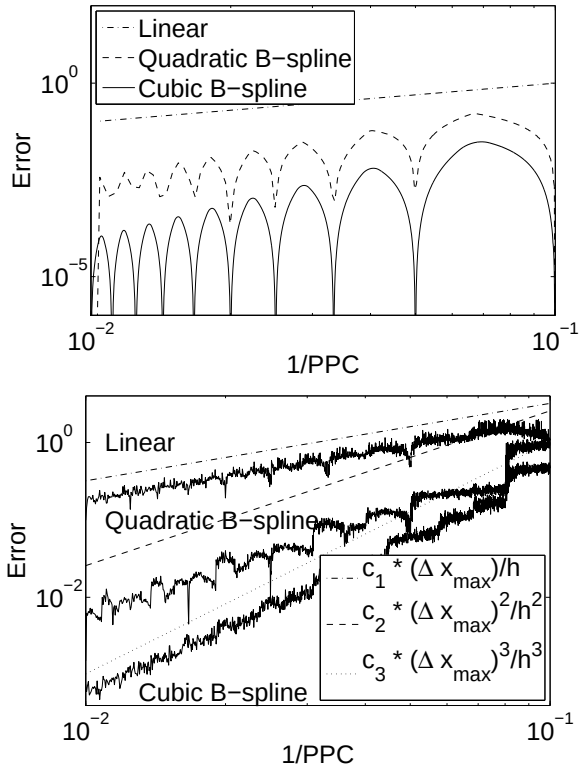


Figure 4.8. Errors in internal force vs. particle spacing b (or the inverse of the number of particles-per-cell) for constant $\sigma = 1$ and grid spacing $h = 0.1$. The top plot uses evenly spaced particles where each sample point is the maximum error over various offsets a. The bottom plot uses randomly spaced particles. The constants on the error bounds are $c_1 = 2$, $c_2 = 1$, and $c_3 = 1/4$. A tighter error bound may be possible for the evenly spaced particles due to symmetries. The Δx_{max} in the bottom plot is 1.6 times the Δx in the top plot due to the random spacing of the particles. This leads to higher error bounds for the randomly spaced particles than the evenly-spaced particles.

$$q(x,t) = \delta(x-l)H(t)\tau\sin(xt/l), \tag{4.19}$$

where $H(t)$ is the Heaviside step function, and initial conditions of $u(x,0) = 0$, $v(x,0) = 0$, has an analytical displacement function derived by wave propagation of the form:

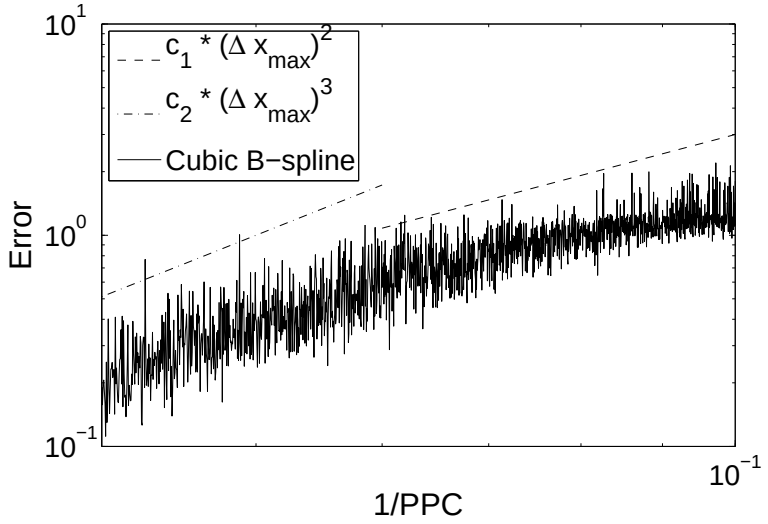


Figure 4.9. Errors in internal force vs. particle spacing for constant $\sigma = 1$ and grid spacing $h = 0.1$. Here particles were globally randomly spaced (the left half of the domain can have more particles than the right half of the domain). This is opposed to the previous figure where particles were only locally randomly spaced and overall particle density remained constant throughout the domain. Instead of perturbing particles from a nominal even spacing, the globally random spacing is accomplished by filling the domain from one side to the other with randomly sized particles, while still using the same overall number of particles as the locally random case.

$$u(x,t) = \begin{cases} 0 & : \quad t \in [0, l-x) \\ \alpha[1+\cos(\omega(t+x))] & : \quad t \in [l-x, l+x) \\ \alpha[\cos(\omega(t+x)) - \cos(\omega(t-x))] & : \quad t \in [l+x, 3l-x) \\ \alpha[-1-\cos(\omega(t-x))] & : \quad t \in [3l-x, 3l+x) \\ 0 & : \quad t \in [3l+x, 4l] \end{cases} \tag{4.20}$$

on $x \in [0, l]$ and $t \geq 0$, where $\alpha = l\tau/(\rho\pi)$ and $\omega = \pi/l$. The stress is given by:

$$\sigma(x,t) = \begin{cases} 0 & : \quad t \in [0, l-x) \\ \tau\sin(\omega(t+x)) & : \quad t \in [l-x, l+x) \\ \tau[\sin(\omega(t+x)) + \sin(\omega(t-x))] & : \quad t \in [l+x, 3l-x) \\ \tau\sin(\omega(t-x)) & : \quad t \in [3l-x, 3l+x) \\ 0 & : \quad t \in [3l+x, 4l]. \end{cases} \tag{4.21}$$

As was stated before, the traction for this problem occurs on the free end of the bar, however the bar end position is time dependent. The analytic end bar

position at any time t can be found by calculating $u(l,t)$ from (4.20). Interpolating the traction force, the external grid forces can then be calculated as

$$f_i^{ext} = \phi_i(l + u(l,t))q(l,t). \tag{4.22}$$

Since there is only one traction force, a maximum of two grid nodes will have non-zero external forces.

The parameters used were, $\rho = E = 100$, $\tau = 1$, and $l = 1$. The uniform MPM grid spanned the region $[0, 1.15]$. The number of grid cells was varied to understand the spatial convergence of the methods. The bar was discretized using $n_p = 3n_g$ number of particles, where n_g is the number of MPM grid nodes. This ends up being slightly more than three particles-per-cell (PPC) since the bar is only of length 1. The problem setup is illustrated in Figure 4.10.

The maximum extension of the bar occurs first at time $T = 1$ which results in an end-bar displacement of $u(l,1) = \pi/50 \approx .06283$. The simulations were thus run to a time of 1 and the RMS errors in displacement were calculated as

$$e^{RMS} = \sqrt{\frac{1}{n_p}\sum_p (u(x_p,1) - u_p)^2}, \tag{4.23}$$

with $u_p = x_p - x_p^0$, the difference between the current and original position of particle p.

A number of important questions can be asked regarding the convergence properties of MPM. Recent studies by Wallstedt and Guilkey [53] looked at convergence

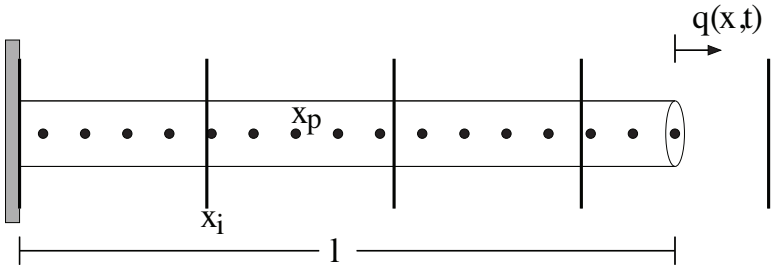


Figure 4.10. One-dimensional bar with traction and corresponding MPM discretization.

with respect to the number of particles-per-cell (PPC). Tran et al. [51] provided an analysis framework for a modified version of MPM used for gas dynamics and showed first order convergence of the method. Bardenhagen et al. [8] performed grid resolution studies with MPM and GIMP in the context of a quasi-static compression problem. For classic MPM, these tests showed convergence for a few data points corresponding to very low grid resolutions (between roughly 5 and 20 grid cells). However, as resolutions increased, the errors started to increase, showing a lack of convergence. In our studies, we fix the number of particles-per-cell (PPC) at approximately 3.5 and focus our attention on the convergence properties with respect to grid resolution for a full dynamic test of the expansion of our fixed-free elastic bar. The results for the simulation with various basis functions are shown in Figure 4.11.

Similar to the Bardenhagen tests [8], standard MPM using piecewise-linear basis functions shows a lack of convergence for 20 grid cells and higher. Simply substituting smoother basis functions for the standard piecewise-linear basis functions traditionally used in MPM drastically improves spatial convergence, with conver-

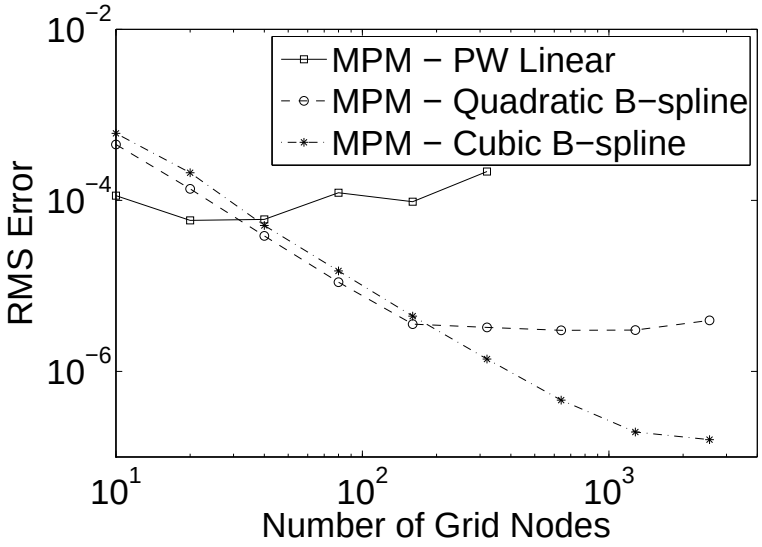


Figure 4.11. Convergence test for a 1-D fixed free elastic bar with sinusoidal traction. The RMS error in displacements are calculated at time $T = 1$, when bar is at maximum extension.

gence rates nearing 2 for both quadratic B-splines and cubic B-splines. Significant integration errors will always exist when using nodal integration such as in MPM, especially when particles are free to move through the domain. Error plateaus can be seen in the simulation when B-spline basis functions are used, however positive convergence results can be obtained out to many thousands of grid cells.

The level of improvement due to increasing the basis function regularity is greater than might be anticipated based solely on the quadrature results seen in the previous section. The benefit of increased regularity goes beyond just improving quadrature as seen in Figure 4.8. In future chapters, we will investigate the other parts of the MPM algorithm which might benefit from the increased level of smoothness of the grid basis functions.

4.4 Summary and Conclusions

In this chapter we have considered the impact of the Material Point Method's choice of maintaining only particle position and volume information when approximating integrals of particle voxels. The nodal integration of equations within MPM was shown to be similar to using midpoint approximations to the integrals. However, the underlying equations being integrated have discontinuities and the partitioning of the domain specified by the particle configuration does not respect these discontinuities, leading to quadrature errors containing information about the jumps in the integrand. Errors in using the midpoint rule across discontinuities were analyzed and applied to the MPM force calculation, the results showing that simply using smoother basis functions such as quadratic and cubic B-splines drastically reduce integration errors.

Grid resolution tests with quadratic B-spline basis functions showed positive spatial convergence up to about 120 grid cells, allowing much lower errors than MPM with piecewise-linear basis functions while only increasing the basis function span by one extra grid cell width. After 120 grid cells, the error plateaus and remains effectively the same up 2560 grid cells. The same test was run with cubic B-splines where positive spatial convergence was demonstrated out to 2560 grid

cells, although the convergence rate starts to drop dramatically after 1280 grid cells. These results are in stark contrast to those obtained using standard (piecewise-linear) MPM in which a lack of convergence is observed.

The analysis and corresponding convergence studies suggest that basis functions smoother than piecewise-linear should be used for moderate grid resolutions. B-spline basis functions are simple to construct and are easily extendable to multiple dimensions. We have implemented these multidimensional B-spline basis functions in Uintah, a massively parallel problem solving environment from the University of Utah [18], which provides a framework for large scale MPM simulations.

CHAPTER 5

EXAMINATION AND ANALYSIS OF IMPLEMENTATION CHOICES WITHIN THE MATERIAL POINT METHOD

MPM, and later, GIMP, were chosen as the solid mechanics component for fluid-structure interaction simulations within the Center for the Simulation of Accidental Fires and Explosions (C-SAFE). The goal of C-SAFE has been the development of a capability to simulate the response of a metal container filled with explosives to a large hydrocarbon pool fire, including heat up, ignition and rupture of the container. The pioneering work of Kashiwa and co-workers [28] inspired this choice as they had demonstrated many of the capabilities that would be required for such simulations, including material failure and solid-to-gas phase transition. To achieve the required level of parallelization and to provide a platform for Adaptive Mesh Refinement, C-SAFE investigators created the Uintah Computational Framework (UCF) [37]. It is within this software environment that the implementations of MPM and GIMP under consideration here exist, along with components for fire simulation, compressible reacting flow, and fluid-structure interaction.

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

choices.

This chapter is organized as follows. Section 5.1 provides an analysis and interpretation of some of the spatial errors present in MPM and GIMP, building on the work in Chapter 4. In the process, we investigate the relationship between GIMP as implemented in the UCF and MPM using B-spline basis functions. Section 5.2 overviews the process for developing interesting problems with analytical solutions which can be used to test our methods and measure errors in our solutions. In Section 5.3 we present numerical results and discuss the differences that result from the aforementioned choices. Lastly, Section 5.4 is a summary of our findings and our conclusions.

5.1 Analysis and Interpretation

In the previous chapter, we performed an analysis on some of the spatial integration errors present within MPM. Other recent work has also focused on formal analysis of errors in the method. Bardenhagen [6], who looked at energy conservation errors in MPM, focusing on the effects of the choice of two time-stepping algorithms. Recently, Wallstedt and Guilkey [55] expanded on the analysis of those time-stepping algorithms. Love and Sulsky [32, 33] analyze an energy consistent implementation of MPM, the second of these showing an implicit implementation to be unconditionally stable and energy-momentum consistent. Wallstedt and Gulikey [53] focus on velocity projection errors and present a scheme which helps ameliorate these errors.

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

5.1.1 The Relationship Between GIMP and B-Splines

Taking a closer look at the weighting function (3.73), we see that the construction is essentially a convolution of the grid basis functions ϕ_i and the particle basis function χ_p. Since a standard piecewise-linear ϕ_i can also be represented as the convolution of piecewise-constant basis functions, we can rewrite the GIMP

weighting function as:

$$\overline{\phi} = \chi_g * \chi_g * \chi_p / (|\chi_g||\chi_p|), \tag{5.1}$$

where the width of χ_g is h (the grid spacing), and the width of χ_p is l_p, as described in the GIMP methods. The equivalent GIMP basis function would then come from evaluating (5.1):

$$\overline{\phi}_i(x) = \overline{\phi}(x - x_i), \tag{5.2}$$

with the GIMP weighting function equivalent to evaluating (5.2) at the particle position, x_p. The reason for rewriting the GIMP basis functions in this way is to demonstrate the similarities between the construction of GIMP basis functions and the construction of B-spline basis functions as in (5.1). Both basis functions are constructed by convolving piecewise-constant basis functions with themselves; however all of the χ in the B-spline basis are of width h while one of the χ functions used in the GIMP method has width l_p.

In cpGIMP, the particle characteristic length l_p (of which there may be different lengths for different directions) is updated in time, meaning the cpGIMP weighting function (3.76) is time dependent, and is different for each particle p. The ideal case would be that the updating of l_p in time will cause the set of particle characteristic functions χ_p to perfectly tile space, but due to the rectilinear constraints of χ_p, this is not possible in general multi-D simulations. Because of this inability to tile space, and the recognition that the major benefit of GIMP is the smoother equivalent basis functions, a simplified standard GIMP is used in which $l_p = l_p^0$ for all time. Furthermore, if $l_p = l$ is constant for all particles p in a simulation (uGIMP), the effect is truly equivalent to using standard MPM with a smoother set of basis functions. In fact, if one were to disassociate the smoothing length, l, from the particles in a uGIMP formulation and instead leave l as a free parameter, the effect is to create quadratic B-spline-like basis functions, with l determining the maximum extent of the functions. Choosing $l = l_p^0$ would give standard GIMP. Choosing $l = h$ would give quadratic B-spline basis functions. Choosing $l = 0$ would lead to the degenerate case of $\chi_p = \delta(x - x_p)$, leaving us with the standard piecewise-linear basis functions.

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

5.1.2 Smoothing Length Dependent Integration Errors

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

$$\int_{\Omega} f(x)\, d\Omega \approx \sum_{p} f(x_p) V_p. \tag{5.3}$$

We performed an analysis of errors in the above approximation within the MPM framework in the previous chapter. There, the nodal integration approximations in MPM were equated to non-uniformly-spaced midpoint integration of functions with discontinuities in various derivatives. In particular, the analysis focused on the errors when calculating the internal force (3.21), which involves the following approximation:

$$f_i^{int} = -\int_{\Omega} \sigma(x) \cdot \nabla \phi_i(x)\, d\Omega \approx -\sum_{p} \sigma_p \cdot \nabla \phi_{ip} V_p. \tag{5.4}$$

The main result from that analysis showed that if the particle arrangements did not respect the discontinuities which arise from the basis functions (i.e., a particle's voxel overhangs node boundaries), an extra integration, or "jump" error can arise in the above approximation which is of the order $C[[f^{(p+1)}]]\Delta x^{p+2}$, where the function, f, being integrated is C_p continuous (with $p = -1$ for discontinuous functions). Here, $[[\cdot]]$ represents the jump in the $p+1$ derivative of f at the discontinuity and Δx is the particle spacing. Note that the function $\nabla \phi_i$ in (5.4) is discontinuous, thus a jump error of $\mathcal{O}(\Delta x)$ can arise in MPM when using standard piecewise-linear basis functions, depending on particle spacing. Numerical examples of this error were shown in Chapter 4.

The jump error for a single particle is calculated as $E_{jump} = \int_{\Omega_p} f(x)\, dx - f(x_p)\Delta x$, where Ω_p spans a discontinuity, or jump, and Δx is the width of the particle. This consists of measuring the midpoint integration error for the single

interval spanning the jumps. Integration approximations, such as in (5.3), involve integration over the whole domain, using multiple intervals, leading to a composite midpoint rule integration error which is $\mathcal{O}(\Delta x^2)$. These two errors are additive, giving a total error of the form

$$E_{total} = \int_{\Omega} f(x)\, d\Omega - \sum_{p} f(x_p)V_p = E_{MP} + E_{jump}, \tag{5.5}$$

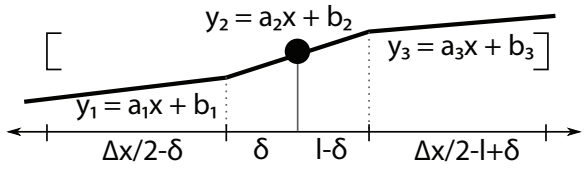
where E_{MP} is the composite midpoint error and E_{jump} is any errors arising from integrating across jumps. Note that if we assume particles are nonoverlapping and fill space, this equation can also be written as

$$E_{total} = \sum_{p} [\int_{\Omega_p} f(x)\, d\Omega - f(x_p)V_p]. \tag{5.6}$$

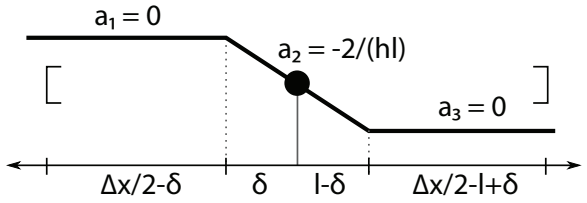
Since the errors are additive and since E_{MP} is always $\mathcal{O}(\Delta x^2)$, C_0 and higher continuous functions exhibit an overall integration error which is $\mathcal{O}(\Delta x^2)$, while discontinuous functions have an error which is $\mathcal{O}(\Delta x)$. Again, this is important because the nodal integration for the internal force calculation in (3.21) involves the gradients of the basis functions, which are discontinuous at grid cell boundaries when the standard piecewise-linear basis functions are used.

In uGIMP, we have the choice of a smoothing parameter l (the width of our general particle characteristic function χ), independent of the individual particle sizes, which ensures us C_1 continuous basis functions but leads to a situation which was not analyzed in Chapter 4 – the case where the width of the particle is greater than the smoothing length. In such cases (as illustrated in Figure 5.1), a particle can span two jumps in the continuity of the basis functions.

Consider a general case from (5.6) where a single particle, or Ω_p, spans three regions of a piecewise linear function. The first region (R_1) is defined by the equation $y_1 = a_1x + b_1$, the second (R_2) will be $y_2 = a_2x + b_2$, and the third (R_3) is $y_3 = a_3x + b_3$ with the particle located a distance δ (here, δ is a distance, not the same as the Dirac delta function presented with respect to basis functions



(a) General Two-Jump Function



(b) GIMP Specific Two-Jump Situation

Figure 5.1. Example cases of a particle's volume (the area between the square brackets) spanning two jumps in a piecewise-linear function: (a) shows a particle spanning two jumps in a general piecewise-linear function with a_i and b_i, $i = 1, 2, 3$ the parameters describing the linear segments, while (b) shows the specific piecewise-linear uGIMP gradient function $\nabla\phi$ and a situation where the particle size Δx is greater than the smoothing length l.

in GIMP) inside the second region (see Figure 5.1). For this case, the integration error is given by:

$$E_{jump} = \int_{\Omega_p} f(x)\,dx - f(x_p)V_p \tag{5.7}$$

$$\begin{aligned} &= \int_{\Omega_p \cap R_1} y_1\,dx + \int_{\Omega_p \cap R_2} y_2\,dx + \\ &\quad \int_{\Omega_p \cap R_3} y_3\,dx - y_2(x_p)\Delta x \end{aligned} \tag{5.8}$$

$$\begin{aligned} &= \frac{1}{2}(a_3 - a_2)l^2 + \frac{1}{2}(a_2 - a_3)l\Delta x + \\ &\quad (a_2 - a_3)l\delta + \frac{1}{2}(a_3 - a_1)\delta^2 + \\ &\quad \frac{1}{2}(a_1 - 2a_2 + a_3)\delta\Delta x + \frac{1}{8}(a_3 - a_1)\Delta x^2, \end{aligned} \tag{5.9}$$

where l is the width of the center region and δ is the particle offset into the center region. Since l and δ are both less than Δx, this whole expression appears to be $\mathcal{O}(\Delta x^2)$. However, when we consider the specific case of measuring the integration error in internal force (5.4) with $\sigma = 1$) when a particle spans the center region

of $\nabla\phi$ as in uGIMP (see Figure 3.6(b) and Figure 5.1(b)), the slopes of the left and right regions in Figure 5.1(b) are zero, while the slope of the center region is dependent on the smoothing length l and the grid spacing h. Specifically, for these regions, $a_1 = 0$, $a_3 = 0$ and $a_2 = -2/(hl)$. When we substitute these parameters into (5.9), we are left with

$$E_{jump} = \frac{-1}{h}\left[\Delta x - l + 2\left(1 - \frac{\Delta x}{l}\right)\delta\right]. \tag{5.10}$$

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

$$E_{jump} = \frac{-1}{hl}[\delta^2 - \delta\Delta x + \frac{1}{4}\Delta x^2]. \tag{5.11}$$

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

To test this analysis, we calculate the force on a single node for a set of particles with constant stress σ. This is the same test performed in Chapter 4 when looking at a particle spanning a single jump instead of the two-jumps analyzed above. In this case, the internal force on a node is calculated as

$$f_i^{int} = -\sum_p \nabla\phi_{ip} \cdot \sigma_p V_p = -\sigma \sum_p \nabla\phi_i(x_p) V_p. \tag{5.12}$$

For a constant stress, internal force should be zero, so any errors are from integrating $\nabla\phi_i$. Figure 5.2 shows the errors for various particle spacings when the smoothing length $l = 1/10$. As expected, when $\Delta x < l$ the error converges as $\mathcal{O}(\Delta x^2)$. When Δx becomes greater than l, the error tends towards $\mathcal{O}(\Delta x)$.

Here, we have shown errors in the internal force which are either $\mathcal{O}(\Delta x)$ or $\mathcal{O}(\Delta x^2)$, depending on the relationship between the smoothing length l and the particle widths Δx. For stability, in addition to the typical CFL constraints one needs when smooth forces exist, we need to consider further time-step restrictions when force kicks arise from these integration errors. These time-step restrictions would be similar to those required, as shown by Tran et al. [51], due to force kicks arising from grid crossing errors. While a time-step of Δt_1 may be sufficient when we are in the $\mathcal{O}(\Delta x^2)$ error region, a smaller Δt_2 may be required to control stability when we are in the $\mathcal{O}(\Delta x)$ error region.

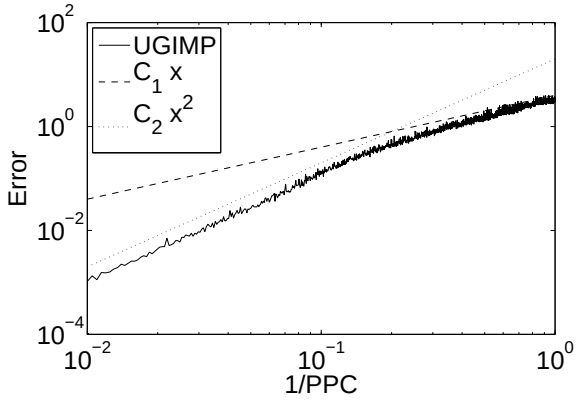


Figure 5.2. Errors in internal force vs. particle spacing for constant $\sigma = 1$, grid spacing $h = 1.0$, and smoothing length $l = 1/10$. The particles have a global uniform density, however they have a locally non-uniform spacing. Otherwise, super-convergent results are observed. Note that when particle spacing is less than the smoothing length l, the error converges as second-order. As the particle spacing becomes greater than the smoothing length, the error tends towards first-order.

5.1.3 Impact of Boundary Treatments

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

In standard MPM with piecewise-linear basis functions, the active grid cells are those which contain a particle. One could argue that a grid cell which contains no particles but still overlaps with a particle voxel (from a particle in a neighboring cell) should also be active, but is not considered so in the current MPM framework. In either case, when considering active cells on the grid, there may be a geometric error of up to h in each direction. When moving to uGIMP, or B-splines, this geometric error can become worse since the support of these basis functions are larger. Cubic B-splines, quadratic B-splines, and uGIMP can experience geometric

errors of up to $2h$, $3h/2$ and $h + l/2$, respectively. All of these errors are $\mathcal{O}(h)$; however it is important to note that these geometric errors are not only a function of how well the object of interest is aligned with grid cells, but they are also a function of basis function choice.

Some work has been performed on MPM background grids which more closely represent the material of interest. For example, [54] has worked on an MLS representation of a material boundary and incorporates this boundary into the MPM integration routines. Here, we sidestep part of the issue by developing test problems in Section 5.2 whose boundaries are perfectly aligned with the grid boundaries (such as a fixed-fixed elastic bar). Even with these aligned test problems, geometric errors can still exist since information is projected to extra nodes outside the domain, as is shown in Figure 3.4(b) and Figure 3.4(c); information which is still used in standard kinematic boundary treatments.

To illustrate this geometric error, Figure 5.3 shows an example of the density field resulting from projecting particles with constant mass (a discretization of a constant density field) to the grid. The density field is calculated as $\rho(x) = \sum_i \rho_i \phi_i(x)$ with $\rho_i = m_i/(\sum_p \phi_{ip} V_p)$. In this example, the constant density field spans the region $[0, 1]$ which is embedded in a grid covering the region $[-0.2, 1.2]$. Since the deformed configuration of the material with respect to the grid is effectively the support of the fields of interest, we can see from Figure 5.3 how implementing boundary conditions and modeling contact can present a challenge when wider basis functions are used.

We postulate that, in general, all of the methods here can suffer from $\mathcal{O}(h)$ geometric errors. In the special case of boundary aligned problems, methods where information is projected to extra nodes, such as uGIMP and standard B-splines, will still be affected by this $\mathcal{O}(h)$ geometric error when Neumann or Dirichlet boundary conditions are applied. These methods should not be affected by this error when periodic boundary conditions are used. Methods which do not require the use of extra nodes, such as standard MPM with piecewise-linear basis functions (Figure 3.4(a)), modified boundary B-splines (Figure 3.5), and cpGIMP will not be affected

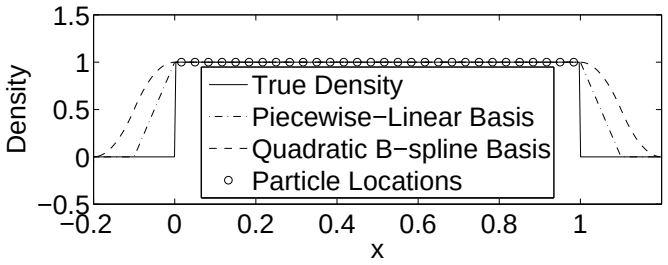


Figure 5.3. Density fields resulting from projecting particle mass to the background grid. The true density field is shown, along with density fields calculated with piecewise-linear and quadratic B-spline basis functions. Here we observe that the geometric extent of information projected to the grid not only depends on which grid cells contain material, but also on basis function choice.

by this geometric error.

It is worth noting that while cpGIMP is implemented in the UCF using extra boundary nodes, information is not projected to these extra nodes in well-behaved boundary aligned simulations. This is because the particle p that is closest to the boundary has width l_p, and is located at a position of $l_p/2$ to the inside of the boundary, and the closest extra node is at a distance of $h + l_p/2$, which is the exact location where the extra node's basis function goes to zero (see Figure 3.6).

5.2 Test Problem Development

Code verification has gained renewed importance in recent decades as costly projects rely more heavily on computer simulations. Full time-dependent test problems with analytical solutions are desired so that simulation errors can be assessed. The Method of Manufactured solutions [42, 29, 5] begins with an assumed solution to the model equations and analytically determines the external force required to achieve that solution. This allows the user to verify the accuracy of numerical implementations, understand the effects of parameter choices within the code, and to find where bugs may exist or improvements can be made. The critical advantage afforded by MMS is the ability to test codes with boundaries or nonlinearities for which exact solutions will never be known. It is argued [29] that MMS is sufficient to verify a code, not merely necessary.

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

5.2.1 Method of Manufactured Solutions Overview

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

Finite element method texts often present total Lagrange and updated Lagrange forms of the equations of motion. The total Lagrange form is written in terms of the reference configuration of the material whereas the updated Lagrange form is written in terms of the current configuration. Either form can be used successfully in a FEM algorithm, and solutions from updated and total Lagrange formulations are equivalent [10].

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

The equations of motion are presented in total and updated Lagrangian forms, respectively:

$$\nabla \mathbf{P} + \rho_0 \mathbf{b} = \rho_0 \mathbf{a}, \tag{5.13}$$

$$\nabla \boldsymbol{\sigma} + \rho \mathbf{b} = \rho \mathbf{a}, \tag{5.14}$$

where $\mathbf{P}$ is the first Piola-Kirchoff Stress, $\boldsymbol{\sigma}$ is Cauchy Stress, ρ is density, $\mathbf{b}$ is acceleration due to body forces, and $\mathbf{a}$ is acceleration.

Many complicated constitutive models are used successfully with GIMP, but for our purposes the simple neo-Hookean is sufficient to test the nonlinear capabilities

of the algorithm. The stress is related in total and updated Lagrangian forms, respectively:

$$\mathbf{P} = \lambda \ln J \mathbf{F}^{-1} + \mu \mathbf{F}^{-1} \left(\mathbf{F}\mathbf{F}^T - \mathbf{I} \right), \tag{5.15}$$

$$\boldsymbol{\sigma} = \frac{\lambda \ln J}{J} \mathbf{I} + \frac{\mu}{J} \left(\mathbf{F}\mathbf{F}^T - \mathbf{I} \right), \tag{5.16}$$

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

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

5.2.2 One-Dimensional Periodic Bar

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

$$u(X, t) = A \sin(2\pi X) \cos(C\pi t), \tag{5.17}$$

$$F(X, t) = 1 + 2A\pi \cos(2\pi X) \cos(C\pi t), \tag{5.18}$$

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

$$b(X, t) = C^2 \pi^2 u(X, t)(2F(X, t)^{-2} + 1). \tag{5.19}$$

The functions u and F are included in (5.19) only to simplify notation. The constitutive model is drawn from (5.16) in 1-D with zero Poisson's ratio:

$$\sigma = \frac{E}{2} \left(F - \frac{1}{F} \right). \tag{5.20}$$

This constitutive model, when combined with the body force given by (5.19) will lead to the analytical displacement solution in (5.17).

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

5.2.3 Axis-Aligned Displacement in a Unit Cube

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

The displacement field is chosen to be:

$$\mathbf{u} = \begin{pmatrix} A\sin(2\pi X)\sin(C\pi t) \\ A\sin(2\pi Y)\sin(\frac{2}{3}\pi + C\pi t) \\ A\sin(2\pi Z)\sin(\frac{4}{3}\pi + C\pi t) \end{pmatrix} \tag{5.21}$$

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

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

$$\begin{aligned} \mathbf{F}_{XX} &= 1 + 2A\pi\cos(2\pi X)\sin(C\pi t) \\ \mathbf{F}_{YY} &= 1 + 2A\pi\cos(2\pi Y)\sin(\tfrac{2}{3}\pi + C\pi t) \\ \mathbf{F}_{ZZ} &= 1 + 2A\pi\cos(2\pi Z)\sin(\tfrac{4}{3}\pi + C\pi t) \end{aligned} \tag{5.22}$$

Acceleration is found by twice differentiating displacement (5.21) in time. Then substituting stress $\mathbf{P}$ into (5.13) and solving for the body force $\mathbf{b}$ (used as the MMS source term) it is found that:

$$\mathbf{b} = \pi^2 \begin{pmatrix} u_X\left(\frac{4\mu}{\rho_0} - C^2 - 4\frac{\lambda(K-1)-\mu}{\rho_0 F_{XX}^2}\right) \\ u_Y\left(\frac{4\mu}{\rho_0} - C^2 - 4\frac{\lambda(K-1)-\mu}{\rho_0 F_{YY}^2}\right) \\ u_Z\left(\frac{4\mu}{\rho_0} - C^2 - 4\frac{\lambda(K-1)-\mu}{\rho_0 F_{ZZ}^2}\right) \end{pmatrix}, \tag{5.23}$$

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

5.3 Results

5.3.1 One-Dimensional Smoothing Length Experiments

We simulated the 1-D periodic bar developed in Section 5.2.2 on the domain $[0, 1]$ to understand the effect of smoothing length on errors within MPM.

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

$$\text{Error} = |x_p^T - X_p^0 - u(X_p^0, T)|. \tag{5.24}$$

This simulation was run with the parameters $A = 0.02$, $E = 10^4$, and $\rho_0 = 1.0$ to a final time $T = 2/C$ (one full period of oscillation) using uGIMP with various numbers of particles-per-cell (PPC) and various smoothing lengths. Figure 5.4 shows how errors depend on smoothing length for different numbers of particles-per-cell (PPC) when we run at a relatively large time-step corresponding to a Courant-Friedrichs-Levy (CFL) number of 0.8. We see that the simulations go unstable when the smoothing length is close to, or less than the initial width of the particles. The two particles-per-cell (PPC) simulation goes unstable for $L < h/2$, the three particles-per-cell (PPC) simulation goes unstable for $L < h/3$, etc.

This observation of a region of instability when using uGIMP helps explain anomalies observed when implementing the quasi-static 1-D bar under gravity from Bardenhagen and Kober's original paper on GIMP [8]. In practice, results match the paper when the bar is under compression but go unstable when under extension. This is understandable in light of our results since the bar under compression will have particles smaller than the smoothing length L at finial time T, while the bar under extension will have particles larger than the smoothing length L at final time T. This later scenario is a good demonstration of the region of instability. Furthermore, since the simulation is a quasi-static problem, the solution

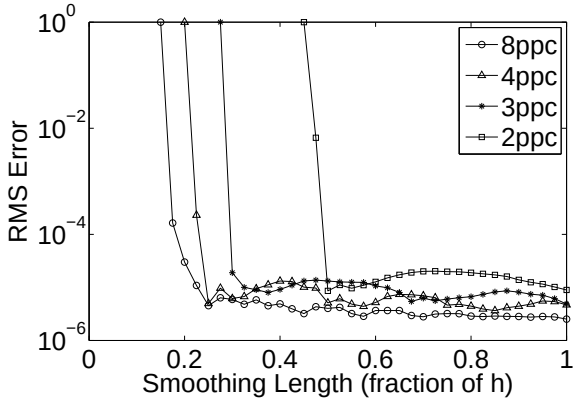


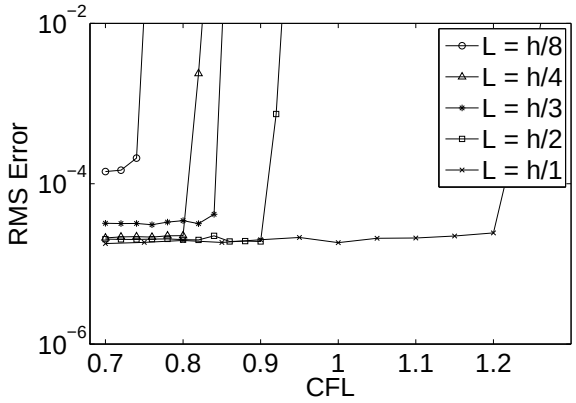
Figure 5.4. Stability analysis for uGIMP showing errors vs. uGIMP smoothing length. The simulations were run with a fixed time-step of $\Delta t = 0.02$, corresponding to a CFL number of 0.8.

is dominated by internal forces (instead of inertial forces) and we would expect to see the same ringing instabilities described by Brackbill [13] which have been observed in PIC codes with low-speed flow calculations.

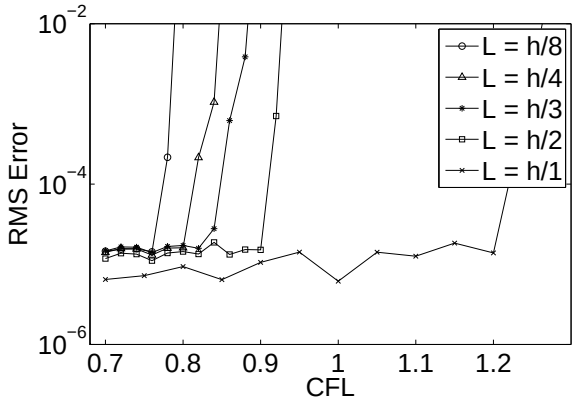
Figure 5.5 shows the effect of smoothing length on the time-step stability restrictions. Figure 5.5(a) shows larger smoothing lengths are stable for a wider range of CFL values when the problem is discretized with four particles-per-cell (PPC). While the simulation using a smoothing length of $h/8$ goes unstable at a CFL number of approximately 0.75, the same simulation with a smoothing length of h (equivalent to quadratic B-spline basis functions) is stable up to a CFL number of approximately 1.2. Figure 5.5(b) shows similar behaviors for eight particles-per-cell (PPC). Furthermore, the time-step stability restrictions do not change significantly between the four particles-per-cell (PPC) and eight particles-per-cell (PPC) simulations, suggesting that the stability is more dependent on the smoothing length than the number of particles-per-cell (PPC).

5.3.2 One-Dimensional Spatial Convergence Results

To investigate the spatial convergence properties of the various MPM methods, we start by simulating the same 1-D periodic bar from Section 5.2.2, now focusing on the behavior of the error with respect to grid resolution and how that error



(a) Four Particles-Per-Cell (PPC)



(b) Eight Particles-Per-Cell (PPC)

Figure 5.5. Examination of stability for uGIMP showing errors versus CFL number for various choices of smoothing length.

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

Figure 5.6 shows results from simulations with both the standard MPM piecewise linear basis functions and quadratic B-splines. With an initial discretization of four particles-per-cell (PPC), we see the standard MPM piecewise linear basis functions showing no significant convergence beyond a modest 16 grid cells. Quadratic

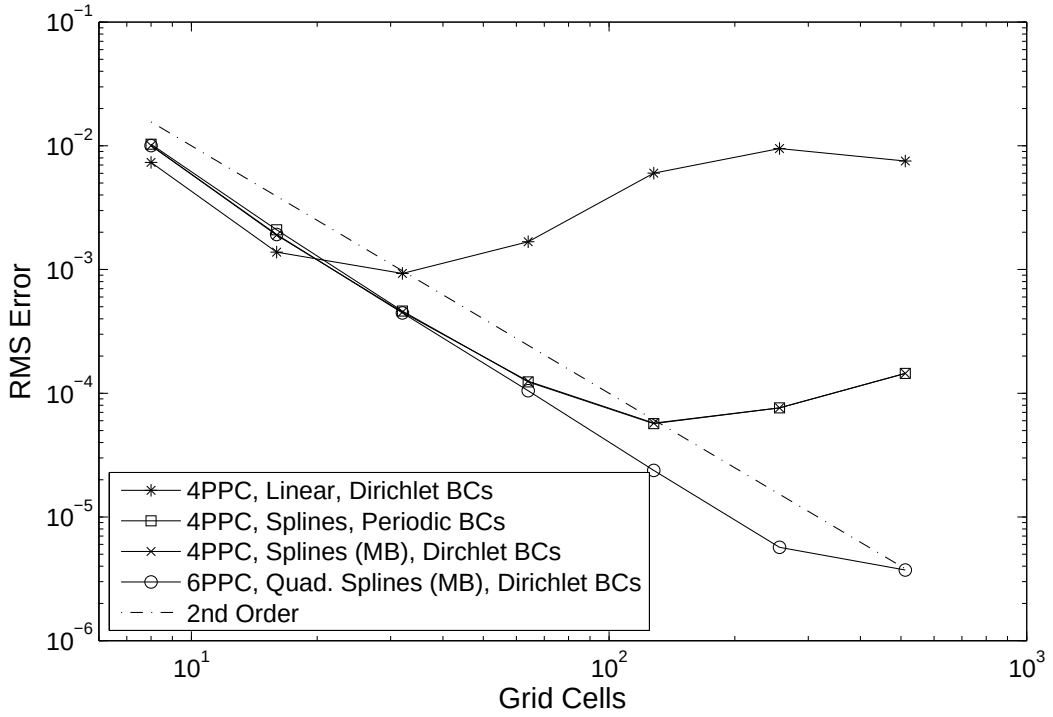


Figure 5.6. Spatial convergence on a 1-D bar manufactured solution problem.

B-splines show a significant improvement, demonstrating $\mathcal{O}(h^2)$ convergence, with an error plateau occurring past 128 grid cells. The four particles-per-cell (PPC) quadratic B-spline simulation was run with both periodic boundary conditions using standard splines (see Figure 3.4(b)) and Dirichlet boundary conditions using the modified boundary splines (Figure 3.5(a)). The results from the two boundary treatments are nearly identical. As was shown previously, using smoother basis functions greatly improves numerical quadrature errors and stability issues, however nodal integration will always give some quadrature error which can explain the error plateaus starting at 128 grid cells. The last set of simulations in Figure 5.6 shows the same quadratic B-spline simulation with Dirichlet boundary conditions, except this time the problem has been discretized using six particles-per-cell (PPC). The extra particles helps lower the quadrature error and lower the error plateau.

To further illustrate errors stemming from boundary treatments, Figure 5.7 shows errors for the same problem simulated with B-splines, using three distinct

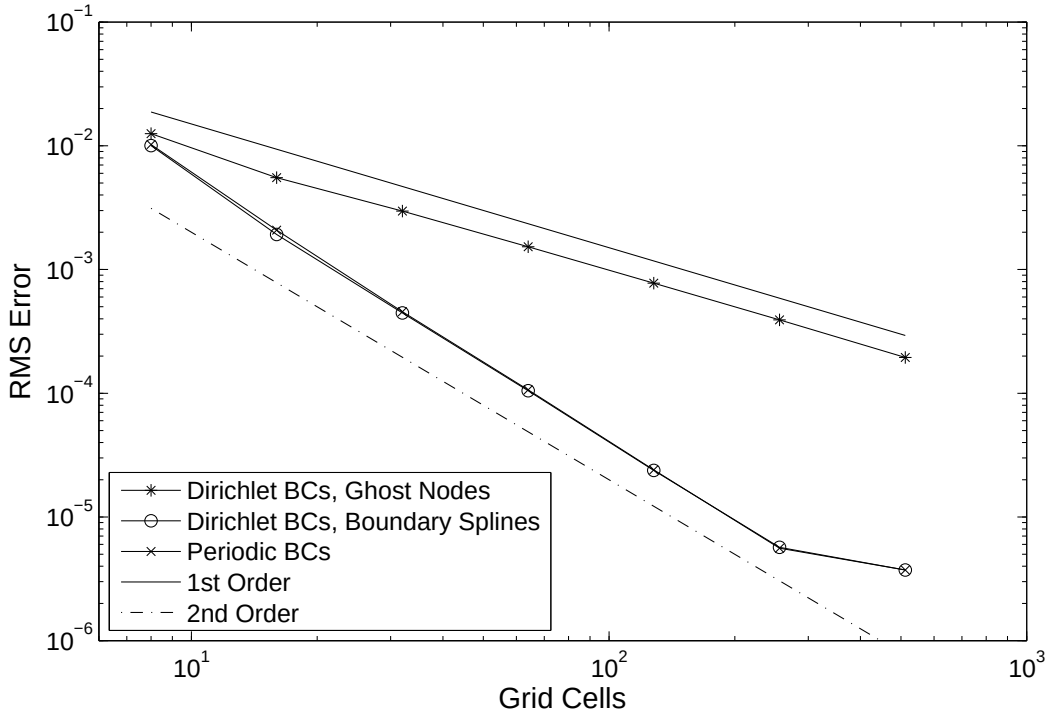


Figure 5.7. Spatial convergence on a 1-D bar manufactured solution using quadratic B-splines with various boundary treatments.

boundary treatments. Similar to Figure 5.6, the Dirichlet boundary conditions with modified boundary B-splines and the periodic boundary conditions with standard B-splines show nearly identical results and demonstrate $\mathcal{O}(h^2)$ convergence. Also shown are results for standard B-splines with Dirichlet boundary conditions. This technique requires handling of the extra or "ghost" boundary nodes as explained in Section 3.2.5. As was discussed previously, this can lead to a geometric error on the grid which is $\mathcal{O}(h)$ and the results show that the error convergence is in fact reduced to $\mathcal{O}(h)$.

5.3.3 Verification with the Method of Manufactured Solutions in Multi-D

The full 3-D axis-aligned problem from Section 5.2.3 was implemented in the UCF to both demonstrate the validity of the multi-D manufactured solution and show that many of the 1-D convergence results from the previous section are also

valid in 3-D. The simulation was run with the parameters $A = 0.05$ (5% maximum displacement), $\rho_0 = 1.0$, $E = 10^4$ and a Poisson's ratio of 0.3. The problem was discretized using four particles-per-cell (PPC) in each dimension (64 total particles-per-cell). Both B-spline basis functions (with symmetric and periodic boundary conditions) and cpGIMP were used in the simulations. The study consisted of grid resolutions from $8 \times 8 \times 8$ cells (32768 particles) up to $64 \times 64 \times 64$ cells (16.8 million particles).

The results in Figure 5.8 clearly show $\mathcal{O}(h^2)$ convergence with cpGIMP for all grid resolutions in the study. Using B-spline basis functions with periodic boundary conditions did nearly as well, with convergence rates trailing off at higher grid resolutions. Similar to the 1-D results, errors when using standard B-splines with the extra, or "ghost" boundary nodes (this time with symmetric boundary conditions) demonstrate the $\mathcal{O}(h)$ convergence we expect due to the geometric errors on the grid.

It is not surprising that cpGIMP outperforms other methods, as this problem is well suited for cpGIMP since particles remain axis-aligned and their voxels area a true partition of the domain. Figure 5.9 is a visualization of a representative 2-D slice of the actual solution, showing the axis-aligned particle voxels and how they partition the domain. B-splines when using extra boundary nodes performed as expected, demonstrating the same $\mathcal{O}(h)$ error as the 1-D results. There is an obvious benefit to using periodic boundary conditions over symmetric boundary conditions for this problem since the errors are significantly lower. It is still unclear, however, why the convergence rate for the periodic boundary conditions trails off from the $\mathcal{O}(h^2)$ behavior we would expect from the 1-D results. There are a number of possibilities, including a more complicated quadrature error behavior in multi-D, or the buildup of grid crossing errors (similar to those analyzed by Tran et al. [51]), which may be more significant in multi-D since many more grid crossing events occur than in 1-D simulations using similar resolutions.

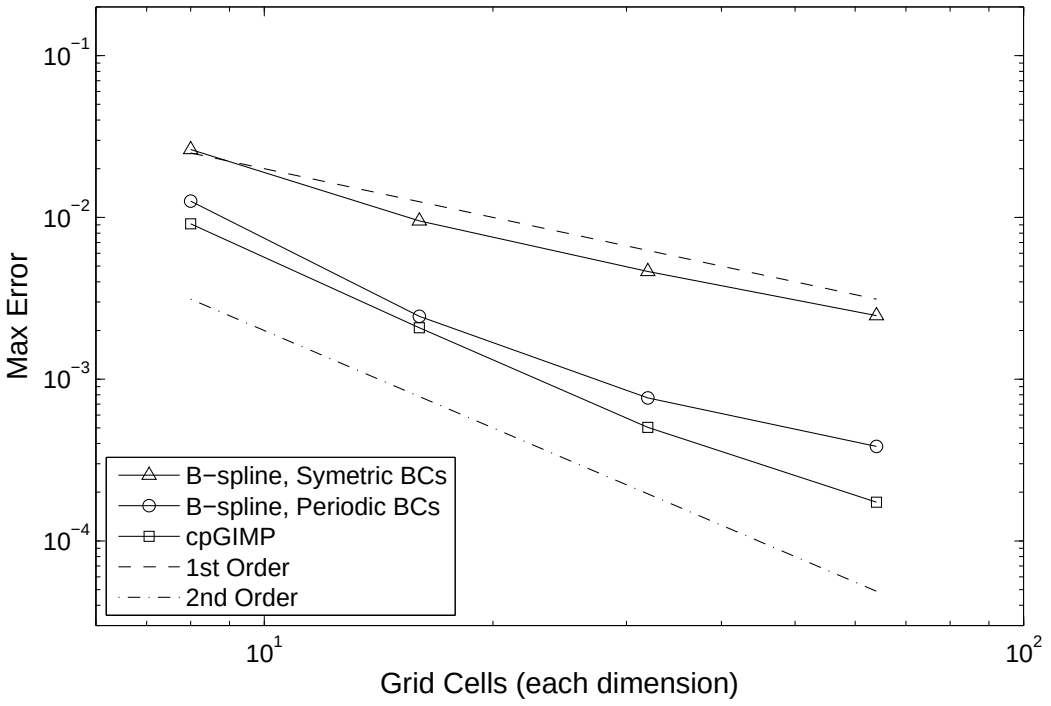


Figure 5.8. Spatial convergence on the 3-D axis-aligned manufactured solution problem.

5.4 Summary and Conclusions

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

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

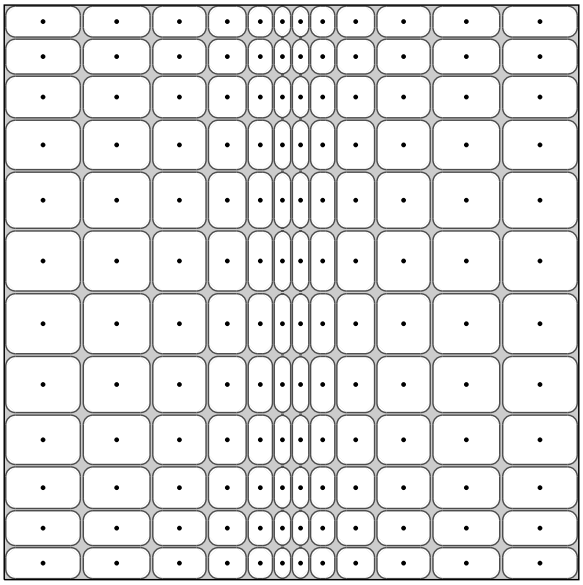


Figure 5.9. Visualization of a representative 2-D slice of the exact solution (5.21) at time $t = 0.005$, showing deformed particle positions (black dots) and a conceptualization of the axis aligned particle voxels. The voxels have been rounded and their sizes slightly reduced for visual clarity.

smoothing length is smaller than the particle widths, this integration error becomes first order. The effects of this relationship between particle widths and smoothing lengths were demonstrated in simulations where instabilities occurred when the smoothing length was set smaller than the particle widths.

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

CHAPTER 6

DECOUPLING AND BALANCING SPACE AND TIME ERRORS IN THE MATERIAL POINT METHOD

Chapters 4 and 5 looked at various errors in MPM, however most of the focus has thus far been on spatial errors. Time-stepping algorithms and their associated errors 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 chapter is to reconcile through analysis and numerical experimentation statements that the time-stepping method used in MPM is "formally second-order" [49] with the recent and detailed convergence tests showing "zero-order" temporal convergence [55].

In this chapter 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 spatial and temporal errors of the method. Moving-mesh MPM is a fully Lagrangian method which helps control some of the more complicated sources of errors within MPM – quadrature and grid crossing errors – thereby allowing us to construct computational experiments which help ferret out the mathematical and algorithmic choices within MPM which violate the mathematical assumptions upon which time-stepping algorithms are based. A simplified nonphysical mathematical problem with similar error characteristics to MPM helps us to both analyze and demonstrate 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 which balance space and time errors. In particular, we help the practitioner understand the trade-offs between increasing spatial resolution through increasing grid spacing and number of particles and the corresponding impact on temporal errors. In the case in which explicit time-stepping algorithms are used (as are often the case in the MPM community and as are analyzed in this paper), the practitioner can also further appreciate the trade-offs between temporal accuracy and stability as dictated by their time-step choice.

This chapter is organized as follows: Section 6.1 provides background and reviews previous temporal error analysis performed in MPM. Section 6.2 outlines an explanation of the coupling of spatial and temporal errors within MPM.

Section 6.3 provides three studies of various error behaviors for both a simplified nonphysical problem with MPM type characteristics and a single step standard and moving-mesh MPM. Section 6.4 shows a demonstration of the errors analyzed in Section 6.3, this time in the full MPM framework. Section 6.5 provides some guidelines to the practitioner on how algorithm parameters affect various errors. Lastly, Section 6.6 is a summary of our findings, our conclusions, and future work.

6.1 Background

One main theme in previous chapters is 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. In Chapter 4 we 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 continuity of the integrand. This analysis helped 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 represented field quantities on the Eulerian mesh within MPM. The simple adaptation to quadratic B-spline basis functions (also detailed

in Chapter 5) allowed the demonstration of second-order spatial convergence of full MPM simulations. Midpoint integration errors are also second-order and therefore, while higher-order basis functions may improve the overall error further, spatial convergence rates will not improve with 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 previous chapters 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 particles overlap 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 at 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 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 [22] and in studying texture evolution in polycrystalline nickel [56].

This chapter seeks to use and extend the perspective on spatial errors gained in previous chapters to understand the lack of temporal convergence demonstrated in [55]. The inspiration for connecting the spatial error characteristics with the temporal error characteristics lies outside the MPM literature. Lawson et al. [30] 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,

$$v^{k+1} = v^k + (a^k + c_1 h^p)\Delta t + c_2 \Delta t^q \tag{6.1}$$

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 allow 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.

6.2 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, Chapter 4 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 grid cells. It is this overhanging of particle voxels with 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 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 voxel overlaps with grid cell boundaries 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 grid cells, and thus influencing future quadrature errors. This is to say, quadrature errors have a compounding effect in

MPM.

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

$$\dot{\mathbf{v}}(\mathbf{x},t) = \mathbf{a}(\mathbf{x},t) \tag{6.2}$$

$$\dot{\mathbf{u}}(\mathbf{x},t) = \mathbf{v}(\mathbf{x},t), \tag{6.3}$$

is the centered difference time integration method:

$$\mathbf{v}^{k+1/2} = \mathbf{v}^{k-1/2} + \mathbf{a}^k \Delta t \tag{6.4}$$

$$\mathbf{u}^{k+1} = \mathbf{u}^k + \mathbf{v}^{k+1/2} \Delta t. \tag{6.5}$$

As has been pointed out in the MPM literature [49], this method is "formally" second-order in time. However, this formal analysis carries with it assumptions regarding smoothness and accuracy of $\mathbf{a}$, assumptions which 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 [51], both of which make second-order temporal convergence unrealizable to the MPM practitioner. Figure 6.1 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 when solving the 1-D bar introduced in Section 5.2.2. The jump in 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 3.2.2. As can be seen in Figure 3.2, particles will remain fixed with respect to the grid for all time. Since 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 since no particle-voxel and grid cell boundary overlap occurs.

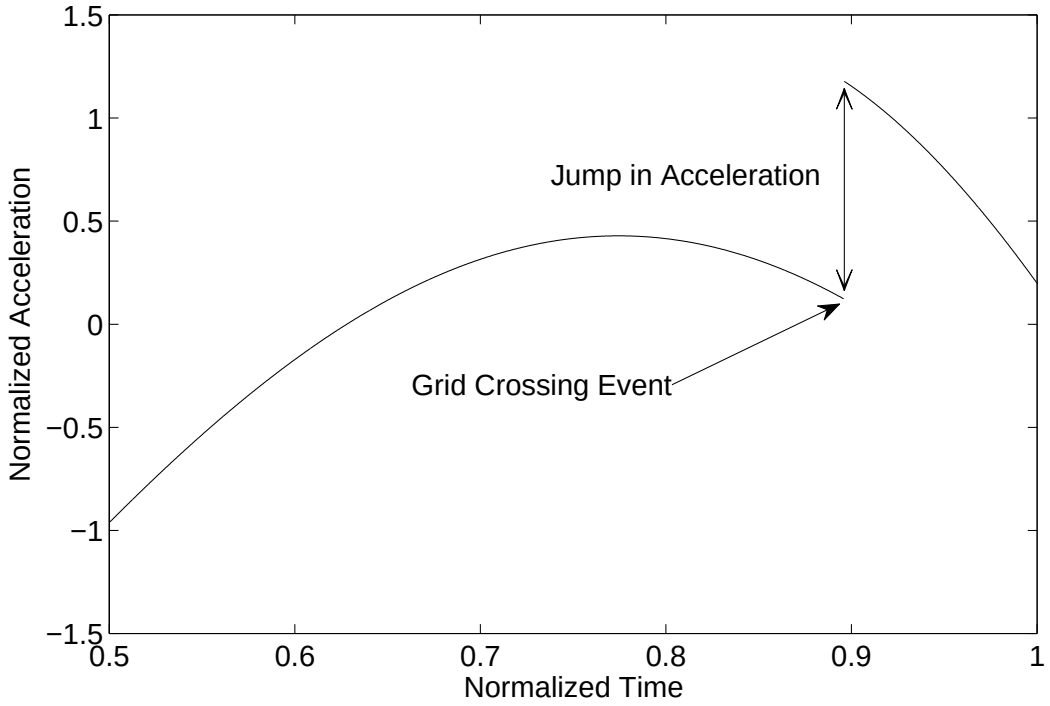


Figure 6.1. Grid acceleration field over time sampled by following the displacement of one particle. Standard MPM and piecewise-linear basis functions were used. The jump in acceleration occurs when a particle crosses a grid cell.

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 following sections.

6.3 Studies of Simplified Decoupled Problems

The entire MPM algorithm, whether we consider standard MPM or moving mesh MPM as outlined in Equations (3.48)-(3.59), involves many steps and many approximations. Furthermore, as argued in Section 6.2, 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, which allows us to study and analyze simpler problems which 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.

6.3.1 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:

$$\frac{dx_p}{dt} = v(x_p(t)). \tag{6.6}$$

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 [24]. 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 on the particles. The errors arising in this situation will be illustrated in Section 6.3.2 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:

$$\frac{dv_p}{dt} = a(x_p(t)) \tag{6.7}$$

$$a_p = a(x_p) = \sum_i a_i \phi_i(x_p) \tag{6.8}$$

$$a_i = f_i/m_i = \frac{1}{m_i}\int_\Omega \nabla\phi_i(x)\sigma(x)\,d\Omega \approx \frac{1}{m_i}\sum_p \nabla\phi_{ip}\sigma(x_p)V_p. \tag{6.9}$$

Our next decoupling strategy which will allow us to look at the impact of the spatial quadrature errors in (6.9) on time-stepping is to specify a discontinuous field $g(x)$ (since $\nabla\phi_i(x)\sigma(x)$ is discontinuous when piecewise-linear basis functions are used) and define acceleration as $a = \int_\Omega g(x)\,d\Omega$. 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 calculated in standard MPM, however the integration will be over a similarly discontinuous function, and thus we will see similar error behaviors. To help 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 for this problem follow in Section 6.3.3.

And finally, in Section 6.3.4 we will consider all errors in the problem. With 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.

6.3.2 Impact of Spatial Discontinuities on Time-Stepping

Recent work by Tran et al. [51] 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 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 continuity of the velocity field. This error is present in the standard MPM algorithm and we will consider it here.

6.3.2.1 Simplified problem. Before we proceed with an analysis, we wish to devise a simplified nonphysical problem which exhibits many of the same math-

ematical 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 expected error behavior in MPM.

The main mathematical features we wish to preserve from the full MPM algorithm is 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 6.2(a). This velocity field is not time-dependent and is defined by:

$$v(x) = \begin{cases} 3x + 1 & : \quad x \in [0, 1/3] \\ 6x & : \quad x \in [1/3, 2/3] \\ 12x - 4 & : \quad x \in [2/3, 5/6] \\ 18x - 9 & : \quad x \in [5/6, 1]. \end{cases} \tag{6.10}$$

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

$$\frac{\partial x}{\partial t} = v(x(t)). \tag{6.11}$$

Since 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

$$x(t) = \frac{b + ax_0}{ae^{at_0}} e^{at} - \frac{b}{a}. \tag{6.12}$$

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

$$x(t) = \frac{1}{3} e^{3t} - \frac{1}{3}. \tag{6.13}$$

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

$$t_1^{cross} = \frac{1}{a} \ln \left[\frac{x_1^{cross} + b/a}{b + ax_0} ae^{at_0} \right]. \tag{6.14}$$

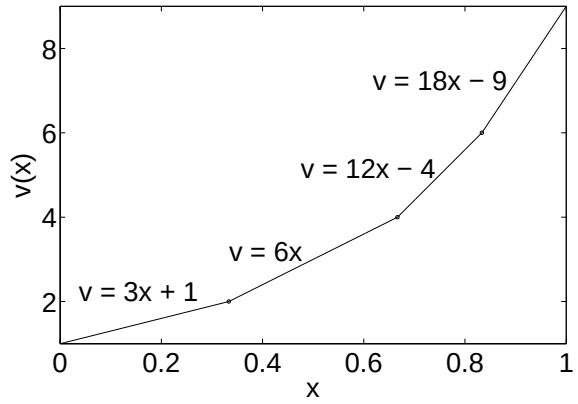
Therefore, (6.13) is valid for $t = [0, t_1^{cross}]$. The second segment, valid for $x \in [1/3, 2/3]$, is calculated in a similar manner, with $a = 6$, $b = 0$, $t_0 = t_1^{cross}$, and

$x_0 = 1/3$. The second crossing time t_2^{cross} is calculated in a similar manner to (6.14), with $x = x_2^{cross} = 2/3$. The resulting piecewise-exponential position function $x(t)$ is shown in Figure 6.2(b).

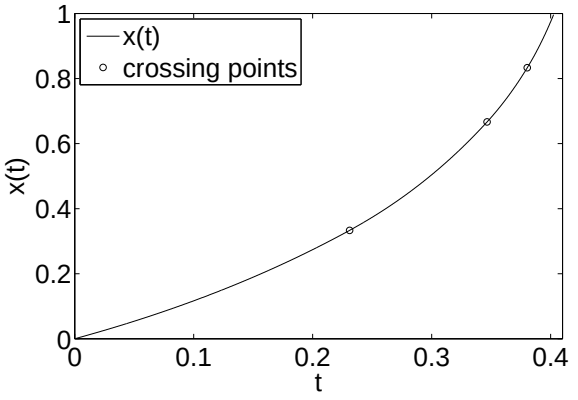
We can see the error behavior of this system by performing the following Forward-Euler time-integration strategy:

$$x_p^{k+1} = x_p^k + v(x_p^k)\Delta t. \tag{6.15}$$

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



(a) Velocity Field



(b) Particle Position

Figure 6.2. Fixed piecewise-linear velocity field and resulting $x(t)$ for our simplified problem.

6.3.2.2 Analysis. Figure 6.3 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., $x_p^k < x_i < x_p^{k+1}$. Since the velocity field $v(x)$ has a jump in continuity at x_i, standard ODE error bounds do not necessarily apply.

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 [51]. They showed this difference to be:

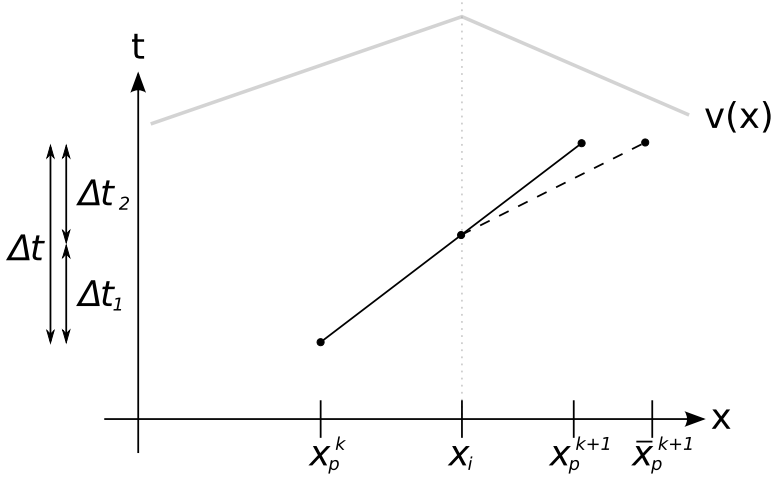


Figure 6.3. One-step versus two-step method for crossing a discontinuity in a velocity field.

$$\bar{x}_p^{n+1} - x_p^{n+1} = (v_i^{n+1} - v_{i-1}^{n+1}) \left[\frac{x_i - x_p^n}{x_i - x_{i-1}} \right] \Delta t_2 + \left[a_{i-1}^n + \frac{x_p^n - x_{i-1}}{x_i - x_{i-1}} (a_i^n - a_{i-1}^n) \right] \Delta t_1 \Delta t_2. \tag{6.16}$$

Here, we continue to expand on the analysis in [51] to help understand the relationship between decreasing Δt and the expected behavior of the difference $\bar{x}_p^{n+1} - x_p^{n+1}$ in (6.16).

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:

$$\bar{x}_p^{n+1} - x_p^{n+1} = (v_i^{n+1} - v_{i-1}^{n+1}) \left[\frac{x_i - x_p^n}{x_i - x_{i-1}} \right] \Delta t_2 + a_p^n \Delta t_1 \Delta t_2. \tag{6.17}$$

Rearranging the first term gives:

$$\frac{v_i^{n+1} - v_{i-1}^{n+1}}{x_i - x_{i-1}} (x_i - x_p^n) \Delta t_2 \approx \frac{\partial v}{\partial x} (x_i - x_p^n) \Delta t_2. \tag{6.18}$$

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 the first term looks like:

$$\frac{\partial v}{\partial x} v_p \Delta t_1 \Delta t_2. \tag{6.19}$$

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

$$\bar{x}_p^{n+1} - x_p^{n+1} = \left[\frac{\partial v}{\partial x} v_p + a_p^n \right] \Delta t_1 \Delta t_2. \tag{6.20}$$

Since $\Delta t_2 = \Delta t - \Delta t_1$ 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

$$\Delta t_1 \Delta t_2 = \alpha (1 - \alpha) \Delta t^2. \tag{6.21}$$

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

$$\bar{x}_p^{n+1} - x_p^{n+1} \leq \frac{1}{4} \left[\frac{\partial v}{\partial x} v_p + a_p^n \right] \Delta t^2. \tag{6.22}$$

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

6.3.2.3 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

$$x_i = x_p + \Delta t_1 v(x_p) \tag{6.23}$$

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.:

$$x_p^{n+1} = x_p^n + v_p \Delta t_1 + v_i \Delta t_2 = x_i + \frac{\Delta t}{2} v_i. \tag{6.24}$$

The difference $\bar{x}_p^{n+1} - x_p^{n+1}$ is calculated and plotted in Figure 6.4. Here, we can see the $\mathcal{O}(\Delta t^2)$ convergence we expect.

Our estimate for the jump error in Section 6.3.2.2 was

$$\varepsilon_{jump} = \frac{1}{4}\left[\frac{\partial v}{\partial x_-} v_p + a_p^n\right] \Delta t^2. \tag{6.25}$$

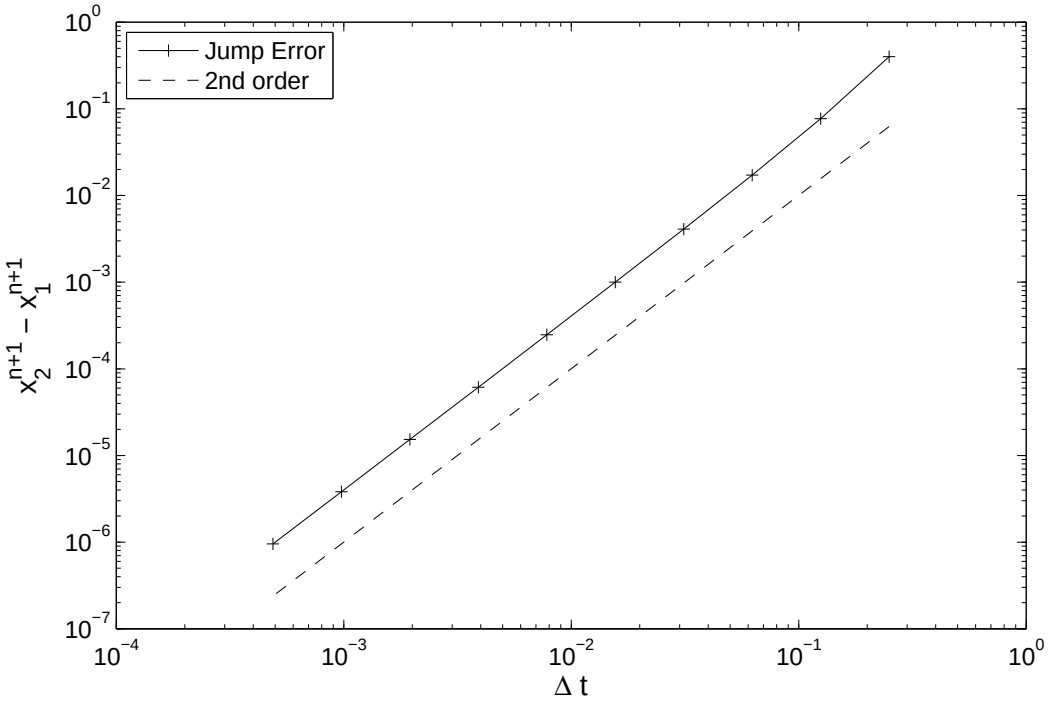
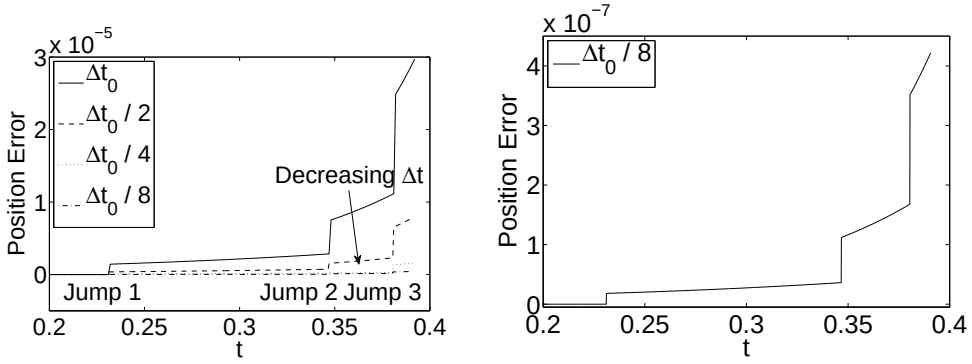


Figure 6.4. Convergence of the jump error ($\bar{x}_p^{n+1} - x_p^{n+1}$) when x_i is half the distance between x_p^n and x_p^{n+1}.

The terms $\partial v/\partial x$, v_p, and a_p are easily calculated for the simplified problem in Section 6.3.2.1. 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 (6.25) 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 $\bar{x}_p$:

$$\varepsilon^k_{jump} = x^k_p - \bar{x}^k_p. \tag{6.26}$$

Figures 6.5(a) and 6.5(b) show the calculated jump errors for various time-step selections. Table 6.1 shows the estimated and calculated jump errors for a particular time-step, demonstrating the error bounds are tight.



(a) Multiple Time-Step Sizes (b) Single Time-Step Size

Figure 6.5. Measured jump errors for simplified problem.

Table 6.1. 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}

6.3.3 Impact of Spatial Quadrature Errors on Time-Stepping

Chapter 4 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.

6.3.3.1 Simplified problem. We will use the same prescribed velocity field v as developed in Section (6.3.2.1), 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 6.6, is not time-dependent and is defined as:

$$g(x) = \begin{cases} -J & : \quad x \in [0, 1/3] \\ 0 & : \quad x \in [1/3, 2/3] \\ J & : \quad x \in [2/3, 5/6] \\ 3J/2 & : \quad x \in [5/6, 1]. \end{cases} \tag{6.27}$$

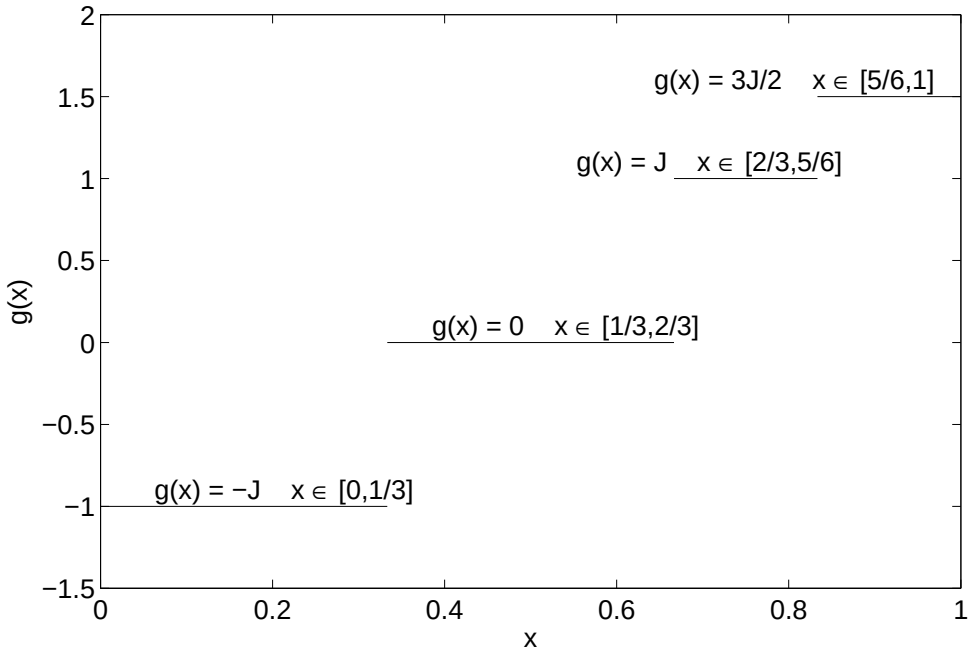


Figure 6.6. Function $g(x)$ used in calculating external acceleration for a simplified problem.

The function $g(x)$ was chosen such that $\int_\Omega g(x)\,dx = 0$. 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

$$a_e(t) = \int_\Omega g(x)\,dx \approx \int_{\Omega - \Omega_p(t)} g(x)\,dx + g(x_p)\Delta x. \tag{6.28}$$

Again, the inclusion of a_e is not meant to be physical, but to include a forcing term which exhibits quadrature errors similar to those which occur when calculating acceleration in MPM. This is accomplished since the integrands in the calculation of our simplified external acceleration a_e and the full MPM acceleration (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 6.2(b). Any error in integration will result in nonzero external accelerations. The errors in 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:

$$a_e^k = \int_{\Omega - \Omega_p} g(x)\,dx + g(x_p^k)\Delta x \tag{6.29}$$

$$v_e^{k+1} = v_e^k + a_e^k \Delta t \tag{6.30}$$

$$v^{k+1} = v(x_p^k) + v_e^{k+1} \tag{6.31}$$

$$x_p^{k+1} = x_p^k + v^{k+1}\Delta t. \tag{6.32}$$

The MPM algorithm exhibits similar behavior as seen in this simplified problem due to quadrature errors in calculating internal forces (3.21). This complicated interplay between spatial and temporal errors is one reason why analysis of MPM is not straightforward.

6.3.3.2 Analysis. Analysis of quadrature errors in Chapter 4 calculated errors in internal force when evenly spaced particles sample a material with constant stress:

$$E_f = \int_\Omega \sigma(x) \cdot \nabla \phi_i \, d\Omega - \sum_p \sigma_p \cdot \nabla \phi_{ip} V_p = \sigma \cdot \left[\int_\Omega \nabla \phi_i \, d\Omega - \Delta x \sum_p \nabla \phi_{ip} \right], \quad (6.33)$$

where Δx is the particle spacing, or volume. Here, $\nabla \phi_i$ is either piecewise-constant or piecewise-linear depending on if 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 continuity of the integrand; however, in general this is not the case with MPM. Figure 6.7 shows an example of a particle spanning breaks in continuity.

The maximum internal force error from (6.33) when using piecewise-linear basis functions is due to integrating over breaks in continuity of the piecewise-constant function $\nabla \phi$, as can be seen in Figure 6.7(a). This error looks like $E_{jump} = C_1[[\phi'(0)]]\Delta x$, where $[[\cdot]]$ denotes the jump condition, and C_1 is a constant depending on the integrand. For the case of piecewise-linear basis functions, $C_1 = 1/2$. Evaluating the entire integral in (6.33), taking into account each continuity jump, the upper bound on the total force error E_f (denoted as E_{total}) is

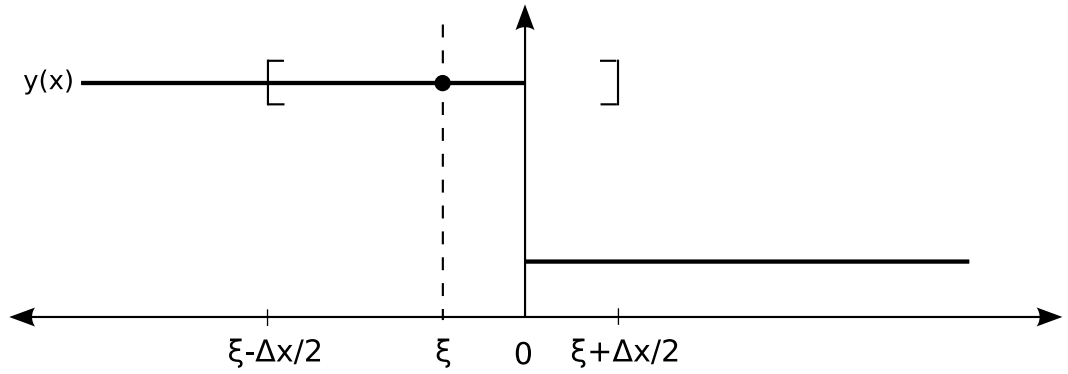
$$E_f \leq E_{total} = 2\sigma \frac{\Delta x}{h}. \quad (6.34)$$

Performing the same analysis when using quadratic B-splines leads to a jump error of the form $E_{jump} = C_2[[\phi''[0]]]\Delta x^2$. For our quadratic B-splines, C_2 when integrating $\nabla \phi$ is 1/8. This leads to an upper bound on the total force error E_f of

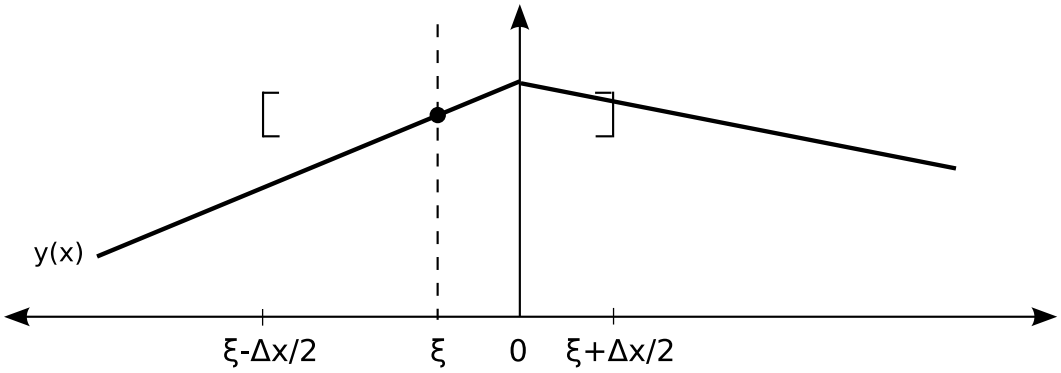
$$E_f \leq E_{total} = \sigma \frac{\Delta x^2}{h^2}. \quad (6.35)$$

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

$$\varepsilon = C\alpha\gamma(t), \quad (6.36)$$



(a) Piecewise-constant integrand



(b) Piecewise-linear integrand

Figure 6.7. 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.

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

$$v^{k+1} = v^k + (a^k + C\alpha\gamma(t^k))\Delta t, \tag{6.37}$$

$$\begin{aligned} x^{k+1} &= x^k + v^{k+1}\Delta t & (6.38)\\ &= x^k + v^k\Delta t + a^k\Delta t^2 + C\alpha\gamma(t^k)\Delta t^2, & (6.39) \end{aligned}$$

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

$$v^{k+2} = v^{k+1} + (a^{k+1} + C\alpha\gamma(t^{k+1}))\Delta t \tag{6.40}$$
$$= v^k + a^k\Delta t + C\alpha\gamma(t^k)\Delta t + a^{k+1}\Delta t + C\alpha\gamma(t^{k+1})\Delta t. \tag{6.41}$$

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

$$v^{k+2} = v^k + a^k\Delta t + a^{k+1}\Delta t + 2C\alpha\Delta t, \tag{6.42}$$

$$x^{k+2} = x^{k+1} + v^{k+2}\Delta t \tag{6.43}$$
$$= x^k + v^k\Delta t + a^k\Delta t^2 + C\alpha\Delta t^2 + v^k\Delta t + a^k\Delta t^2 + a^{k+1}\Delta t^2 + 2C\alpha\Delta t^2 \tag{6.44}$$
$$= x^k + 2v^k\Delta t + 2a^k\Delta t^2 + 3C\alpha\Delta t^2. \tag{6.45}$$

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

$$v^N = v^0 + \sum_{i=1}^{N} a^i\Delta t + NC\alpha\Delta t, \tag{6.46}$$

$$x^N = x^0 + \sum_{j=1}^{N} v^j\Delta t \tag{6.47}$$
$$= x^0 + Nv_0\Delta t + \sum_{j=1}^{N}\left[(\sum_{i=1}^{j} a^i\Delta t) + jC\alpha\Delta t\right]\Delta t \tag{6.48}$$
$$= x^0 + Tv_0 + \sum_{j=1}^{N}\sum_{i=1}^{j} a^i\Delta t^2 + \sum_{j=1}^{N} jC\alpha\Delta t^2 \tag{6.49}$$
$$= x^0 + Tv_0 + \sum_{i=1}^{N}(N-i+1)a^i\Delta t^2 + \frac{N(N+1)}{2}C\alpha\Delta t^2 \tag{6.50}$$
$$= x^0 + Tv_0 + \sum_{i=1}^{N}(N-i+1)a^i\Delta t^2 + \frac{1}{2}TC\alpha\Delta t + \frac{1}{2}T^2C\alpha, \tag{6.51}$$

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

$$E_q = \frac{1}{2}TC\alpha\Delta t + \frac{1}{2}T^2C\alpha. \tag{6.52}$$

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

$$E_q = \frac{1}{2}TC\alpha^2\Delta t + \frac{1}{2}T^2C\alpha^2. \tag{6.53}$$

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 show a demonstration of these errors in the simplified problem.

6.3.3.3 Results. In Section 6.3.3, 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:

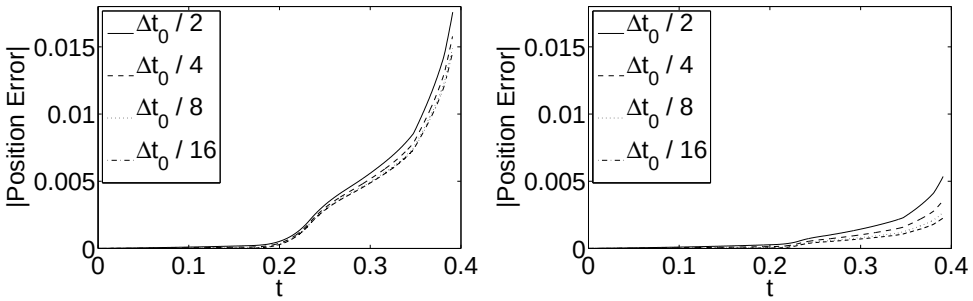
$$x^N = x^0 + Tv_0 + \sum_{i=1}^{N}(N - i + 1)a^i\Delta t^2 + \frac{1}{2}TC\Delta x\Delta t + \frac{1}{2}T^2C\Delta x. \tag{6.54}$$

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:

$$\varepsilon = \frac{1}{2}TC\Delta x\Delta t + \frac{1}{2}T^2C\Delta x. \tag{6.55}$$

From this estimate, we would expect global errors to decrease with decreasing time-step Δt, but to be limited by the last term, which has no dependence on time-step. We also expect global quadrature errors to increase quadratically with final time T and to decrease with decreasing particle spacing Δx as seen in Figure 6.8.

While the error estimate in (6.55) 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 6.8 exhibit this unbounded error, it is worth noting that the global error in many simulations oscillate around the true solution.



(a) $\Delta x = 1/4$ (b) $\Delta x = 1/8$

Figure 6.8. Global errors for two values of Δx and numerous values of Δt plotted on the same axes.

6.3.4 Balancing Space and Time Errors

Until now, we have focused on analysis of errors in simplified problems which 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.

6.3.4.1 Moving-mesh MPM. The velocity update equation in Equation (6.4) 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:

$$v^{k+1/2} = v^k + \dot{v}^k(\Delta t/2) + \frac{1}{2}\ddot{v}^k(\Delta t/2)^2 + \frac{1}{6}\dddot{v}^k(\Delta t/2)^3 + \mathcal{O}(\Delta t)^4, \tag{6.56}$$

$$v^{k-1/2} = v^k - \dot{v}^k(\Delta t/2) + \frac{1}{2}\ddot{v}^k(\Delta t/2)^2 - \frac{1}{6}\dddot{v}^k(\Delta t/2)^3 + \mathcal{O}(\Delta t)^4. \tag{6.57}$$

Subtracting (6.57) from (6.56) yields:

$$v^{k+1/2} - v^{k-1/2} = \Delta t\dot{v}^k + \frac{1}{24}\Delta t^3\dddot{v}^k + \cdots. \tag{6.58}$$

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

$$a^k = \dot{v}^k = \frac{v^{k+1/2} - v^{k-1/2}}{\Delta t} + \mathcal{O}(\Delta t^2). \tag{6.59}$$

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

$$v^{k+1/2} = v^{k-1/2} + \Delta t a^k + \mathcal{O}(\Delta t^3). \tag{6.60}$$

Again, these are standard ODE theory results and assume $\mathbf{a}$ is known and sufficiently smooth [24]. However, as was shown above, significant spatial errors

can exist in MPM. In fact, assuming second-order spatial errors, acceleration will take the form

$$a^k = \tilde{a}^k + c_1 h^2, \tag{6.61}$$

where $\tilde{a}$ is our calculated acceleration and c_1 is a constant not dependent on h. Substituting (6.61) into (6.60) gives us our MPM time-update equation for the centered difference velocity update scheme:

$$v^{k+1/2} = v^{k-1/2} + \Delta t(\tilde{a}^k + c_1 h^2) + c_2 \Delta t^3. \tag{6.62}$$

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

$$c_1 h^2 = c_2 \Delta t^2, \tag{6.63}$$

which occurs when

$$\Delta t = Ch, \tag{6.64}$$

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

6.3.4.2 Standard MPM. When using standard MPM with piecewise-linear basis functions, we expect first-order spatial errors. Therefore, instead of (6.61), acceleration will now be:

$$a^k = \tilde{a}^k + c_1 h, \tag{6.65}$$

where $\tilde{a}$ is the calculated acceleration. Substituting (6.65) into (6.60) gives us our MPM time-update equation for the centered difference velocity update scheme within the standard MPM framework:

$$v^{k+1/2} = v^{k-1/2} + \Delta t(\tilde{a}^k + c_1 h) + c_2 \Delta t^3. \tag{6.66}$$

This differs from (6.62) 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

$$c_1 h = c_2 \Delta t^2, \tag{6.67}$$

which occurs when

$$\Delta t = C\sqrt{h}, \tag{6.68}$$

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 (6.68), the transition point for standard MPM with B-spline basis functions should still occur when $\Delta t = Ch$, as in (6.64).

Demonstrations of these transition, or balancing points will be shown in Section 6.4.3 for both moving-mesh and standard MPM.

6.4 Results for Full MPM Simulations

In Section 6.3, 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 when simulating the 1-D periodic bar in Section 5.2.2 and show similar behaviors as in the simplified problems.

6.4.1 Impact of Spatial Discontinuities on Time-Stepping

In an attempt to reduce spatial quadrature errors to a point where the jump error from Section 6.3.2 can be seen, the 1-D periodic bar was solved using 64 grid cells and 100 particles-per-cell (PPC). The parameters used were $A = 0.05$ (five percent maximum displacement), $E = 10^4$, $\rho_0 = 1.0$, and a time-step which 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 particle positions x_p^{k+1}, but on each step, the two-step method for handling grid crossings (outlined in Section 6.3.2.2) was used to calculate the two-step particle position $\bar{x}_p^{k+1}$. The single-step jump error was then calculated as $\varepsilon_{jump}^k = x_p^k - \bar{x}_p^k$. These jump errors were then accumulated to obtain the global jump error $E_{jump}^k = \sum_k \varepsilon_{jump}^k$.

Figure 6.9 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 particles-per-cell (PPC).

6.4.2 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 particles-per-cell (PPC). Figure 6.10 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 particles-per-cell (PPC). The particle acceleration a_p is clearly discontinuous for piecewise-linear basis functions and continuous for B-spline basis functions. Close inspection

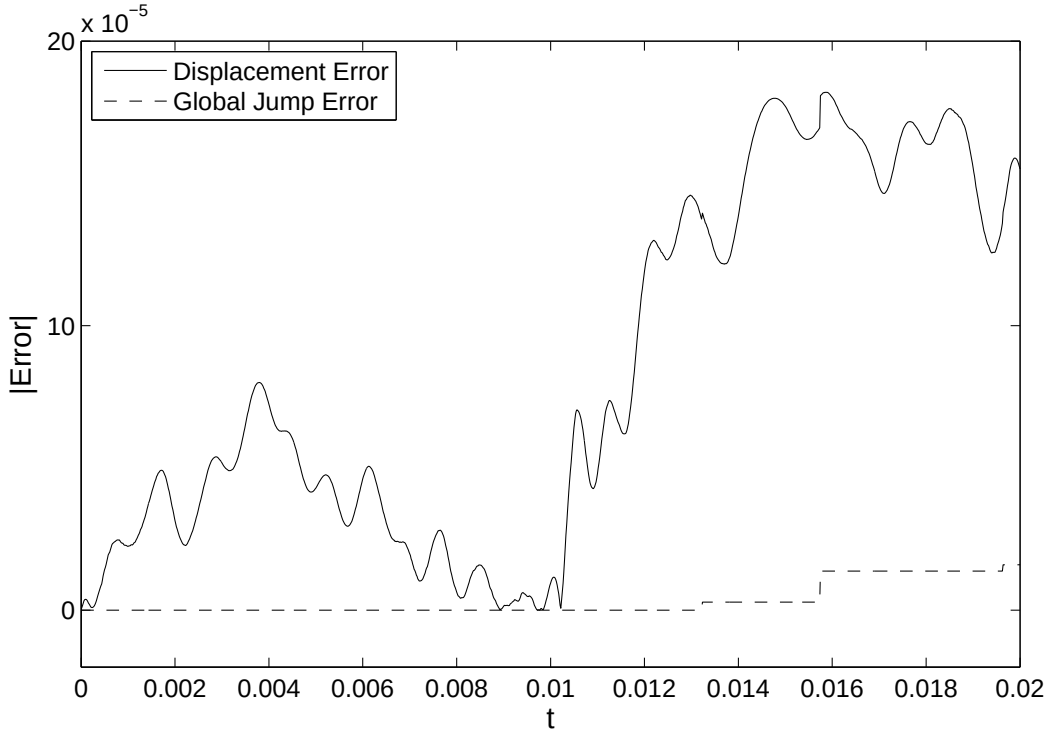


Figure 6.9. Displacement error and cumulative jump error (calculated as the sum of the difference between single-step and two-step time integration) for a single typical particle in a simulation with 64 grid cells and 100 particles-per-cell (PPC).

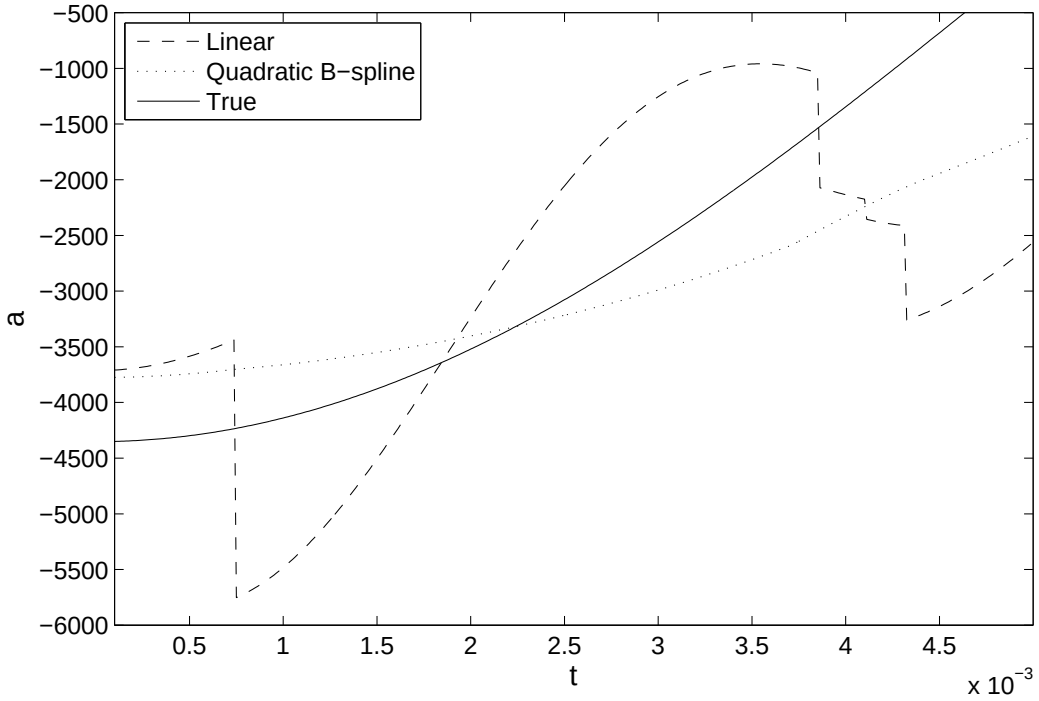
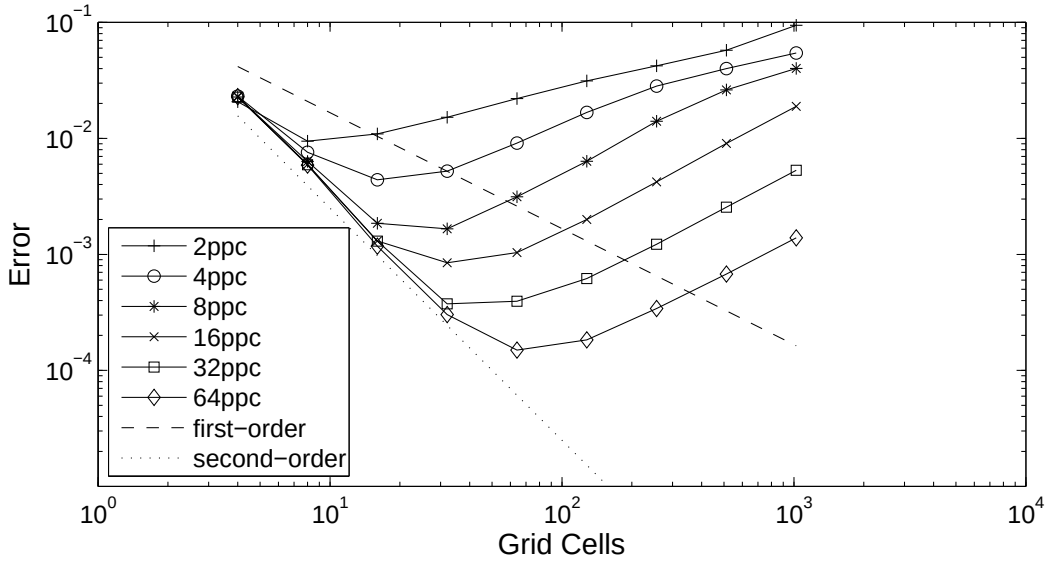


Figure 6.10. Acceleration felt by a particle for both piecewise-linear and B-spline basis functions in standard MPM. Discontinuities in accelerations occur at grid crossings with piecewise-linear basis functions. With B-spline basis functions, acceleration remains continuous when particles cross grid nodes.

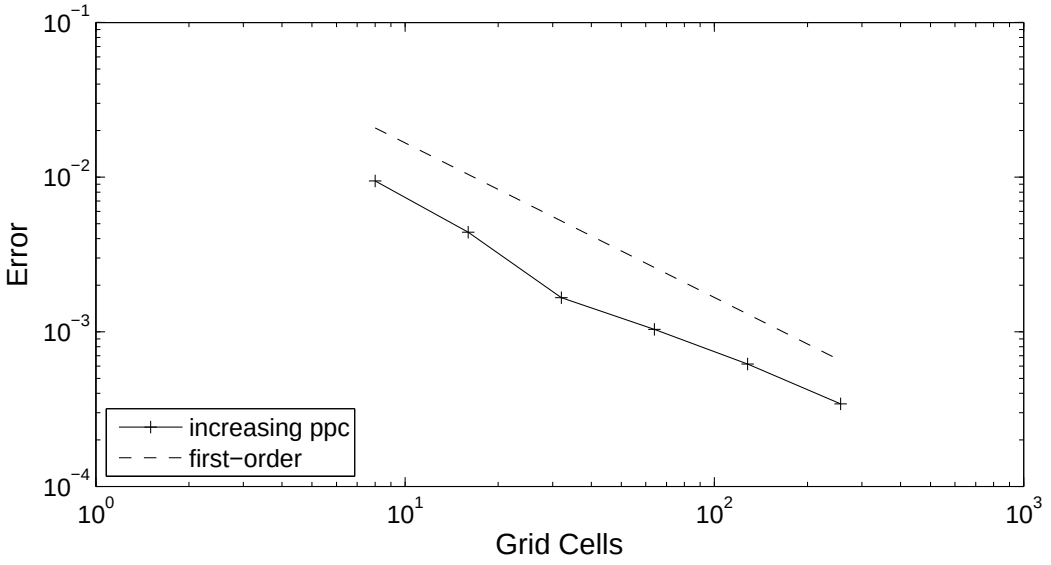
shows that a_i is only C_0 continuous for quadratic B-spline basis functions. This behavior is mainly due to the quadrature errors generated as particles cross grid nodes. This jump in acceleration is unaffected by time-step selection.

Spatial convergence studies were then performed on the 1-D bar with various numbers of particles-per-cell (PPC), and the RMS displacement error was calculated after one full period of oscillation. When the number of particles-per-cell (PPC) is held constant, standard MPM initially converges at as $\mathcal{O}(h^2)$, as would be expected in standard finite elements with piecewise-linear basis functions, but soon reaches a point where quadrature error starts to dominate, and convergence is lost. Increasing the number of particles-per-cell (PPC) lowers the point at which quadrature errors start to dominate. Figure 6.11(a) shows these results.

Figure 6.11(b) shows the results of increasing the number of particles-per-cell



(a) Various PPC



(b) Increasing PPC

Figure 6.11. 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 (PPC) held constant in each convergence test. Figure (b) shows a convergence test where the number of particles-per-cell (PPC) is increased along with the number of grid cells. In this case, the number of particles-per-cell (PPC) was set to one-fourth the number of grid cells, resulting in a total of $N^2/4$ particles.

(PPC) at the same rate as the number of grid cells. In this case, the number of particles-per-cell (PPC) was set to one-quarter the number of grid cells. This ever increasing number of particles-per-cell (PPC) results in a seemingly consistent first-order method. This, however, may be prohibitive in practice since 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.

6.4.3 Balancing Space and Time Errors

Temporal convergence studies thus far have not demonstrated second order convergence as we would expect [6, 55]. The assumption has been that spatial errors are dominating in the regimes being tested. One limitation on previous studies has been 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 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. Since 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 (6.61) assumes a second-order spatial error when calculating acceleration. To understand the constant associated with this term, we measure the L^2 error of the calculated acceleration on the grid:

$$\varepsilon_a^2 = \int_\Omega (\tilde{a}(x) - a(x))^2 \, dx, \tag{6.69}$$

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

$$\tilde{a}(x) = \sum_i a_i \phi_i(x). \tag{6.70}$$

Figure 6.12 shows the spatial convergence of these acceleration errors with both piecewise-linear and quadratic B-spline basis functions at a time corresponding to

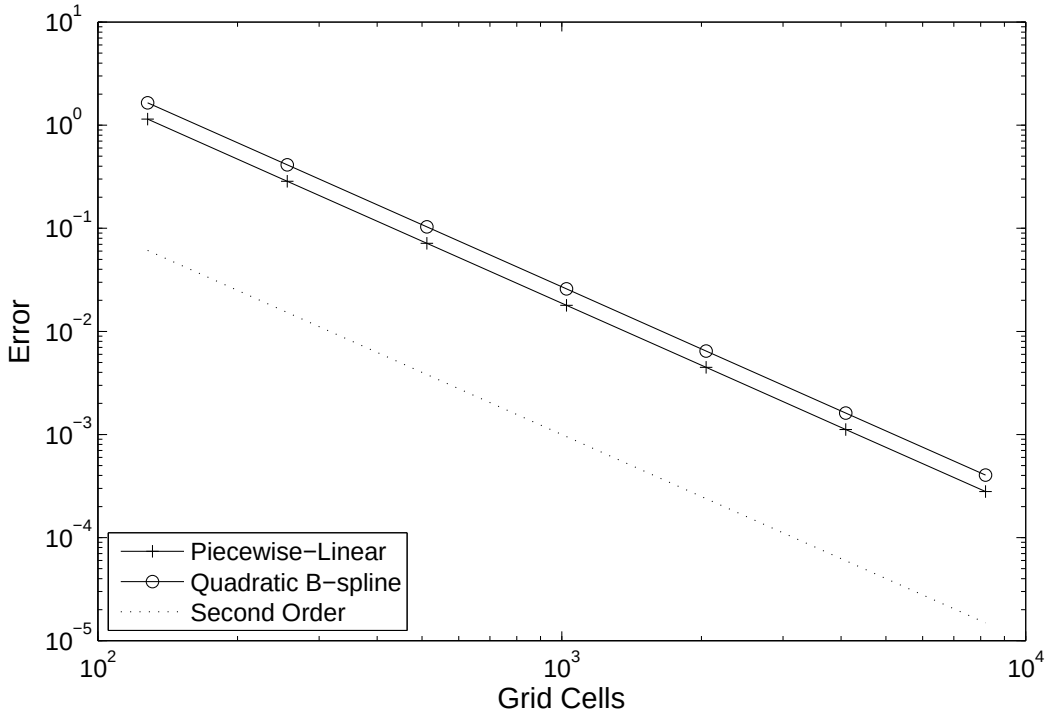


Figure 6.12. Spatial convergence of grid acceleration L^2 errors at time $t = 0.2122 \cdot C$, with moving-mesh MPM, four particles-per-cell (PPC), and periodic boundary conditions for the 1-D axis aligned problem.

approximately 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 since the data in Figure 6.12 is 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 (since our domain is of length $L = 1$) is

$$c_1 = \frac{\varepsilon_a}{h^2} = 1.8764 \times 10^4. \tag{6.71}$$

Next, to measure the constant c_2, we measured the L^2 errors in grid velocity after a single time-step. The data in Figure 6.13 show third-order temporal convergence for the local truncation error as expected from (6.62). Knowing that the local velocity

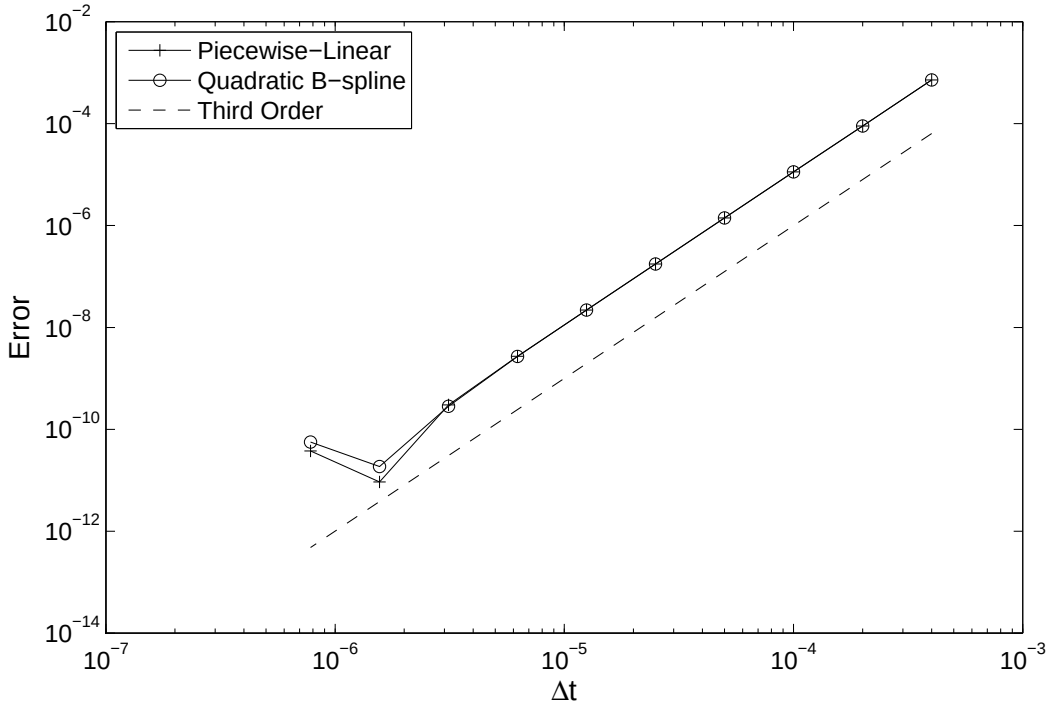


Figure 6.13. Temporal convergence of local truncation errors in grid velocity with moving-mesh MPM, 2^{20} grid cells, four particles-per-cell (PPC), and periodic boundary conditions for the 1-D axis aligned problem.

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

$$c_2 = \frac{\varepsilon_v}{\Delta t^3} = 1.1277 \times 10^7. \tag{6.72}$$

Running a single time-step with 2048 grid cells, ($h = 4.88 \times 10^{-4}$), we would expect the transition to occur from (6.64) at $\Delta t = 1.9917 \times 10^{-5}$. Figure 6.14 shows this experiment and the transition occurs precisely where we expect. Using the same techniques above with the data in Figures 6.12 and 6.13, 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 since the errors for piecewise-linear and quadratic B-splines are comparable.

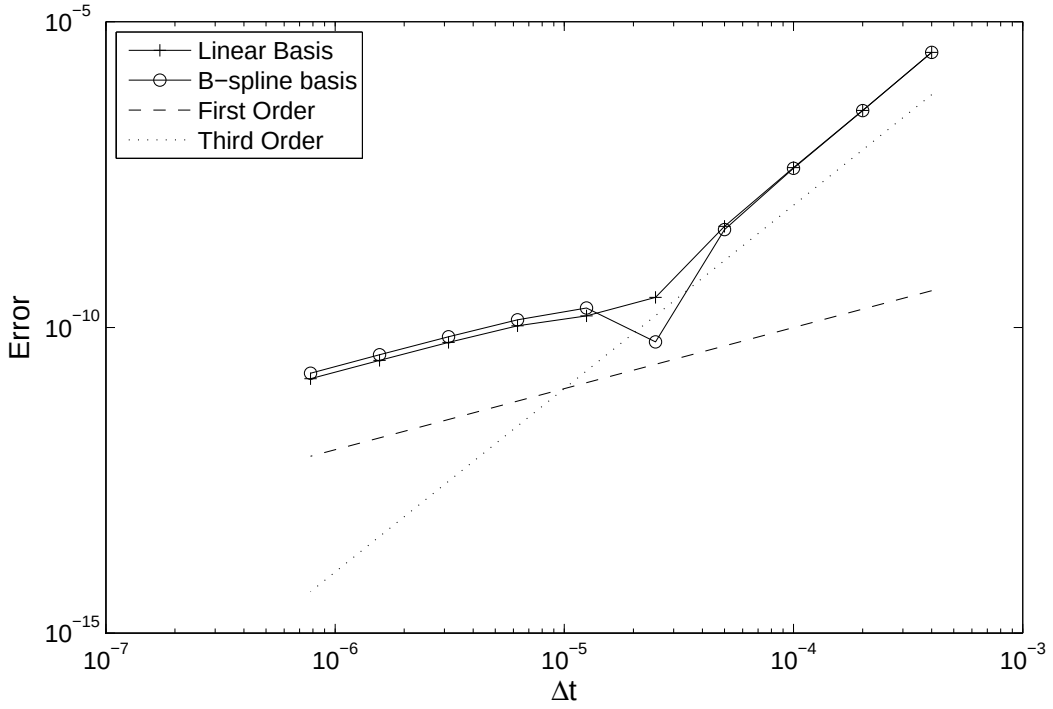


Figure 6.14. Temporal convergence for local truncation errors in grid velocity with moving-mesh MPM, 2048 grid cells, four particles-per-cell (PPC), and periodic boundary conditions for the 1-D axis aligned problem.

6.4.3.1 Standard MPM. Figure 6.15 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.

Since the acceleration error for piecewise-linear basis functions now takes the form $\varepsilon_a = c_1 h$ in the asymptotic region, and since the data in Figure 6.15 demonstrates 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

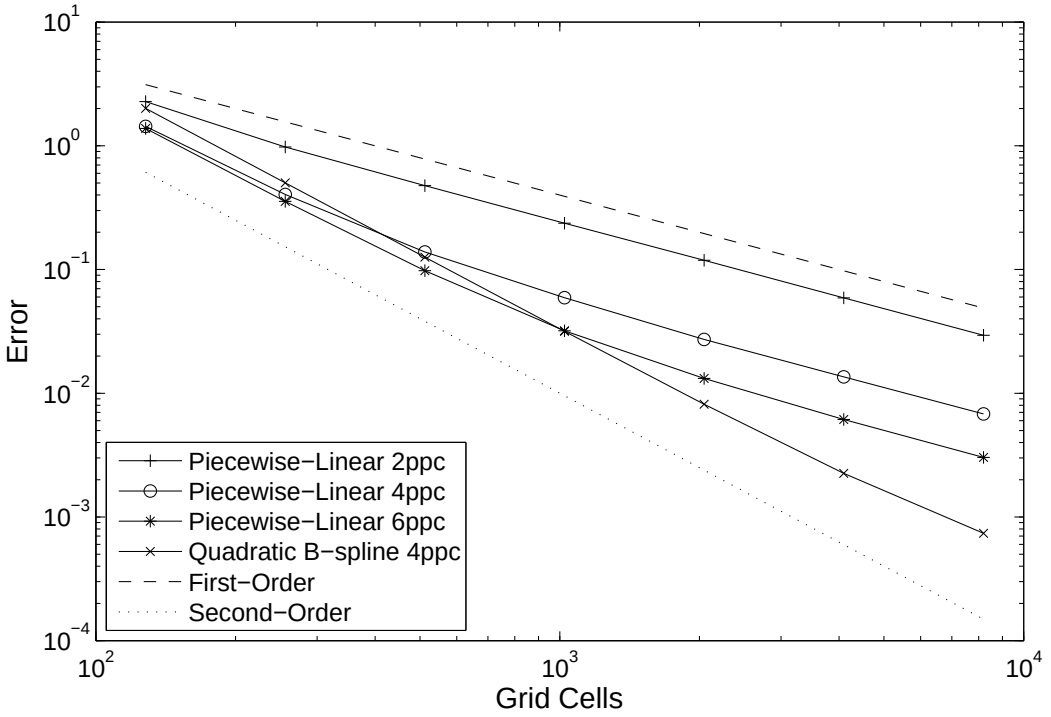


Figure 6.15. Spatial convergence of grid acceleration L^2 errors with standard MPM and periodic boundary conditions for the 1-D axis aligned problem. Piecewise-linear basis functions with various numbers of particles-per-cell (PPC), and quadratic B-spline basis functions with four particles-per-cell (PPC) are shown.

$$c_1 = \frac{\varepsilon_a}{h} = 6.0538 \times 10^1. \tag{6.73}$$

Next, we again measure the constant c_2 using the errors in grid velocity after a single time-step. The data in Figure 6.16 show strong third-order local truncation error convergence with larger Δt, as expected from (6.66). With smaller Δt, a convergence plateau is reached, where the larger spatial errors in standard MPM are starting to dominate. Since 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

$$c_2 = \frac{\varepsilon_v}{\Delta t^3} = 1.1278 \times 10^7. \tag{6.74}$$

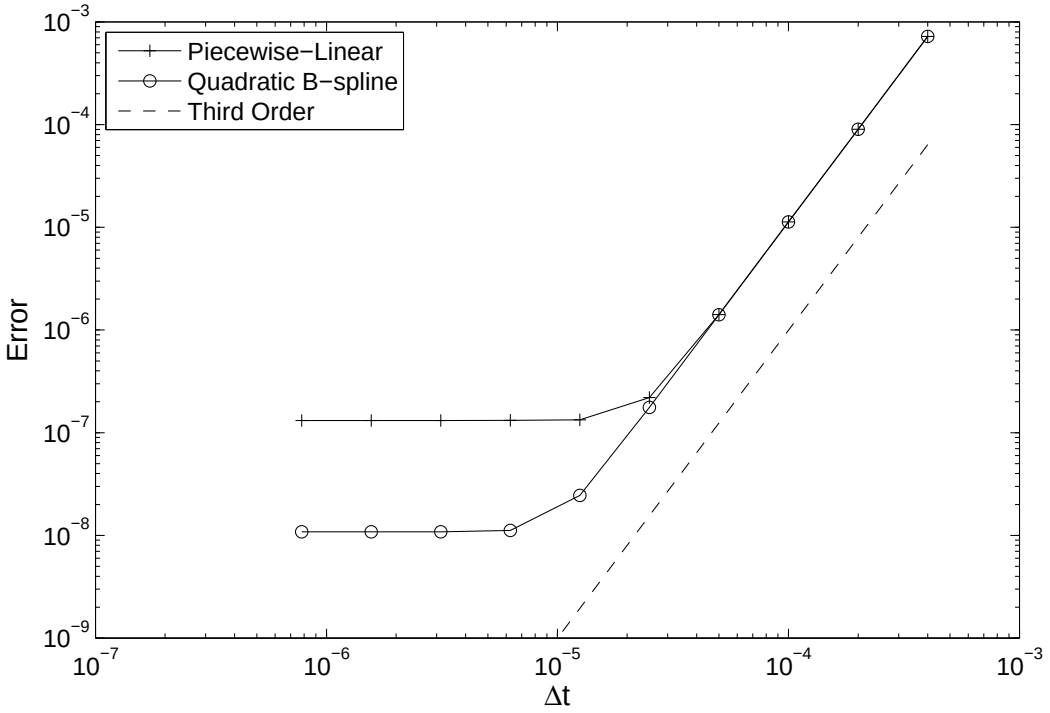


Figure 6.16. Temporal convergence of local truncation errors in grid velocity with standard MPM, 2^{20} grid cells, four particles-per-cell (PPC), and periodic boundary conditions for the 1-D axis aligned problem.

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 (6.68) when piecewise-linear basis functions are used (6.68) is at $\Delta t = 5.1196 \times 10^{-5}$. Figure 6.17 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 6.15 and 6.16, along with (6.64), the transition point for standard MPM using quadratic B-splines is calculated as $\Delta t = 2.6507 \times 10^{-5}$.

6.5 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

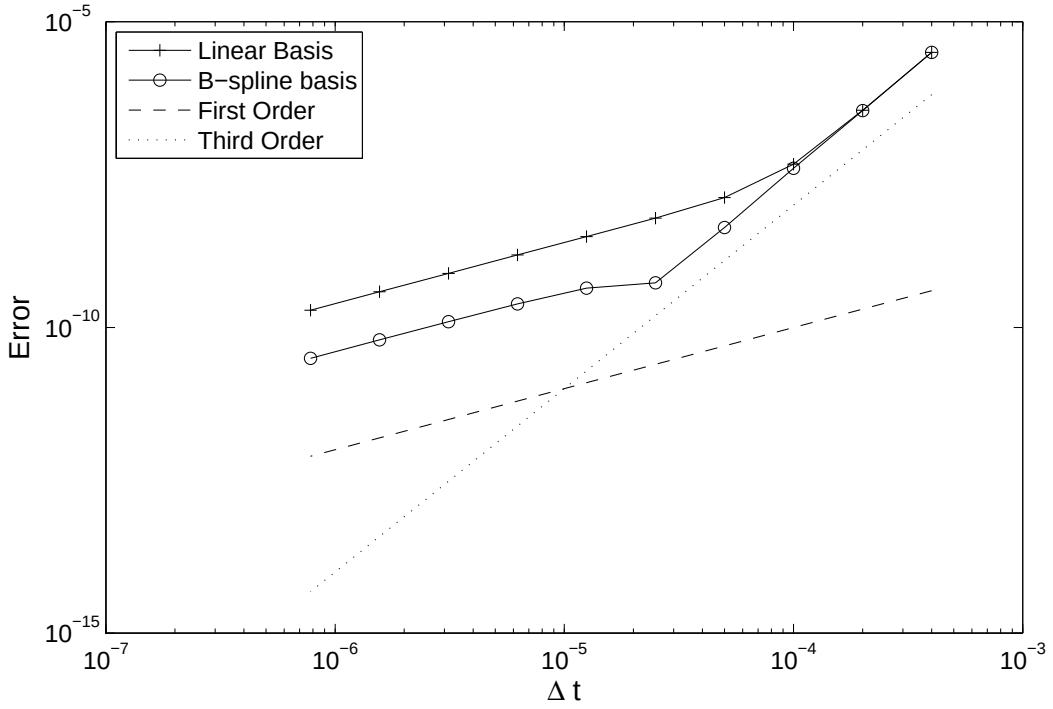


Figure 6.17. Temporal convergence for local truncation errors in grid velocity with standard MPM, 2048 grid cells, four particles-per-cell (PPC), and periodic boundary conditions for the 1-D axis aligned problem.

integration jump errors. There are also a number of parameters that a practitioner must chose when discretizing a problem with MPM, and many 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 $\mathcal{V}$ 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 chapter. 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 [27]. 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 – ideas 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 6.3.2, 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 since smoother basis functions result in smoother velocity fields that particles travel through.

In Section 6.3.3 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 chapter) increases k from 1 to 2, and since α is typically much less than 1, this significantly reduces the overall effect of 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 particles-per-cell (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 particles-per-cell (PPC) must increase as the number of grid cells is increased. A demonstration of this is shown in Section 6.4.2.

The local truncation error results in Section 6.4.3 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 problem. This explains why previous researchers [55] 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 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 temporal errors further.

To demonstrate the effects of following these guidelines while solving an engineering problem, the 1-D periodic bar from Section 5.2.2 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 particles-per-cell (PPC), and a relatively small time-step corresponding to a CFL of 0.1. A second simulation used more expensive quadratic B-spline basis functions, 64 grid cells, the same four particles-per-cell (PPC), and a larger time-step corresponding to a CFL of 0.9. The RMS error of particle positions and corresponding run-times are shown in Table 6.2. Here, we see that a selection of parameters based upon the guidelines provided above results in a much lower error with less computational effort.

6.6 Summary and Conclusions

In this chapter we performed three studies on various errors present in the Material Point Method. These studies, which included analysis and demonstrations,

Table 6.2. 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 (seconds)
Simulation 1	2.21×10^{-2}	9.62
Simulation 2	8.39×10^{-5}	0.50

were performed on simplified problems which 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 which helps control some of the complexities of 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 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 work in previous chapters, we were able to develop an estimate for global errors arising from the compounding effects of the quadrature errors. Lastly, 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, while 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 reduce the quadrature errors and therefore reduce global errors in the method, and that time-steps near the stability limit are sufficient in most cases since time-steps near the stability limit already lead to solutions which are dominated by spatial errors. This helps more fully explain results showing zero-order global temporal convergence demonstrated by previous researchers [55]. Further guidelines were given, helping the practitioner understand which algorithm parameters can be expected to affect the various errors in the method.

The analysis in Chapter 4 showed that the nodal integration and the implicit mass lumping in MPM currently restricts the method to second-order in space. To further reduce errors in the method, more advanced error control techniques beyond simple grid refinement may be required. Therefore, a detailed understanding of the

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

CHAPTER 7

IMPROVING SPATIAL ERRORS IN THE MATERIAL POINT METHOD USING EXTRAPOLATION TECHNIQUES

Thus far our attempts to improve the order of accuracy of the Material Point Method have, for the most part, stayed within the standard MPM framework. The most successful of these have included the use of smoother basis functions (Chapter 4), careful use of the centered difference time-stepping scheme (Chapters 5 and 6) and careful implementation of boundary conditions (Chapter 5).

We anticipate that careful use of this previous knowledge will allow us to move beyond simple time and space refinement strategies for error improvement, and instead use more advanced error control techniques such as Richardson extrapolation to improve the error convergence properties of the method.

7.1 Richardson Extrapolation

Richardson extrapolation [41] is a technique originally developed to increase the order of accuracy in finite difference solutions. The basic technique relies on applying Taylor's theorem to analyze the error of a finite difference formula, calculating two solution approximations at different spatial resolutions, and combining those approximations in a way which cancels leading error terms in the finite difference approximation, resulting in a higher-order solution. These techniques typically fail for non-smooth problems, such as shock problems and simulations of cracks, but are often useful for smooth problems. In this section, we will show how a Richardson extrapolation formula can be developed for a centered difference approximation of a first derivative operator. We will also show how Richardson extrapolation techniques can be used in numerical integration to increase order of accuracy.

7.1.1 Standard Extrapolation

As an example, we will develop a Richardson extrapolation formula which can be applied to a centered finite difference approximation to $f'(x)$:

$$f'_k \approx \frac{f_{k+1} - f_{k-1}}{2h}, \tag{7.1}$$

where $f_k = f(x_k)$ and $x_k = x_0 + kh$. To begin developing our Richardson extrapolation scheme, we start by performing Taylor series analysis of the centered difference scheme. The Taylor series of f in the neighborhood of a is given by:

$$f(x) = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f^{(3)}(a)}{3!}(x-a)^3 + \cdots . \tag{7.2}$$

Expanding this about x_k and evaluating at x_{k+1} gives

$$f_{k+1} = f_k + f'_k h + \frac{1}{2}f''_k h^2 + \frac{1}{6}f^{(3)}_k h^3 + \frac{1}{24}f^{(4)}_k h^4 + \mathcal{O}(h^5). \tag{7.3}$$

Performing the same expansion, but evaluating at x_{k-1} gives

$$f_{k-1} = f_k - f'_k h + \frac{1}{2}f''_k h^2 - \frac{1}{6}f^{(3)}_k h^3 + \frac{1}{24}f^{(4)}_k h^4 - \mathcal{O}(h^5). \tag{7.4}$$

Subtracting (7.4) from (7.3) gives

$$f_{k+1} - f_{k-1} = 2f'_k h + \frac{1}{3}f^{(3)}_k h^3 + \mathcal{O}(h^5). \tag{7.5}$$

Rearranging terms gives us our centered difference formula and error term:

$$A(x_k, h) = \frac{f_{k+1} - f_{k-1}}{2h} = f'_k + \frac{1}{6}f^{(3)}_k h^2 + \mathcal{O}(h^4). \tag{7.6}$$

The Richardson extrapolated solution comes from applying (7.6) with two different resolutions, h and $h/2$, giving the following for $h/2$:

$$A(x_k, h/2) = f'_k + \frac{1}{6}f^{(3)}_k \frac{h^2}{4} + \mathcal{O}(h^4/16). \tag{7.7}$$

Our Richardson extrapolation formula then comes from combining (7.6) and (7.7) to eliminate the h^2 term:

$$\begin{aligned} \frac{4A(x_k, h/2) - A(x_k, h)}{3} &= \frac{4f'_k + \frac{1}{6}f^{(3)}_k h^2 - f'_k - \frac{1}{6}f^{(3)}_k h^2}{3} + \mathcal{O}(h^4) & (7.8) \\ &= f'_k + \mathcal{O}(h^4). & (7.9) \end{aligned}$$

In general, if our approximation $A(h)$ is of order p, then the Richardson extrapolation formula is given by [24]:

$$A = \frac{m^p A(h/m) - A(h)}{m^p - 1} + \mathcal{O}(h^q) \tag{7.10}$$

where $q > p$.

To demonstrate the above Richardson extrapolation formula, Figure 7.1 shows results of performing the extrapolation techniques in (7.9) when estimating $f'(x)$ at $x = \pi/4$ when $f(x) = \sin(x)$. Errors between the estimated $f'(x)$ and the actual solution are plotted for various values of h. The standard centered difference estimates show second-order convergence rates, as expected by error analysis of the method. The Richardson extrapolation solution demonstrates fourth-order convergence, also as expected.

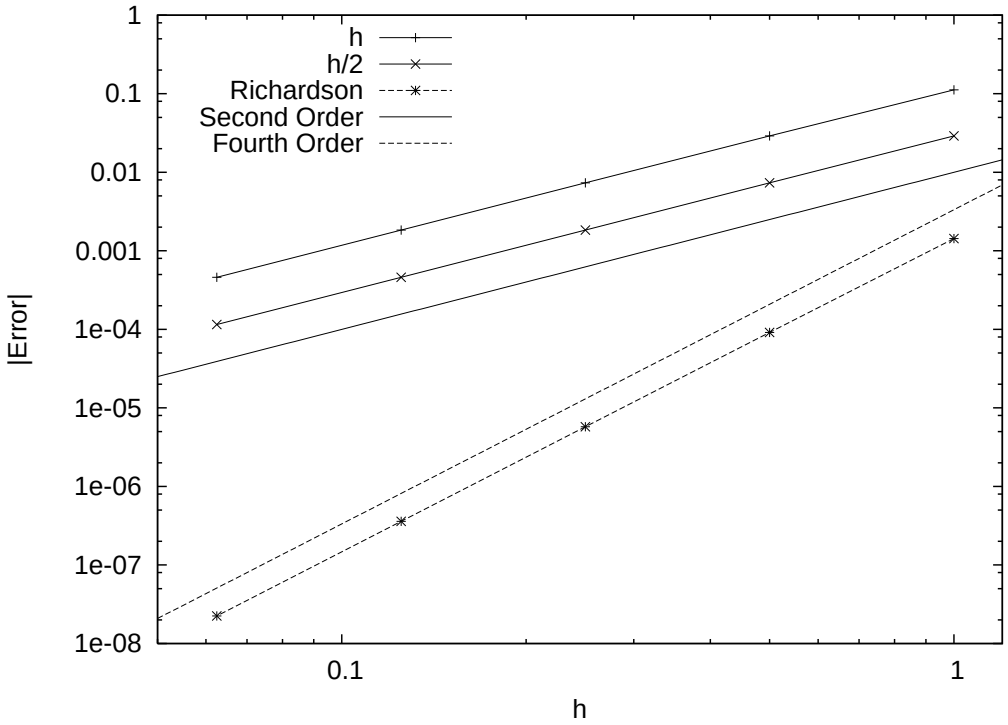


Figure 7.1. Example of Richardson extrapolation when estimating $f'(x)$ when $f(x) = \sin(x)$. The standard centered difference estimates demonstrate second-order convergence rates while the Richardson extrapolation solution demonstrates fourth-order behavior.

The true power of Richardson extrapolation is apparent when discretizing and solving a problem domain with $N = L/h$ points, where L is the size of the domain, and h is the spacing of the discretization points. To achieve a particular accuracy with a standard finite-difference method, one might need to use many more discretization samples, and require much longer run-times, than if one were to use Richardson extrapolation techniques.

7.1.2 Extrapolation Techniques in Numerical Quadrature

While the above technique shows how to use Richardson extrapolation when numerically estimating derivatives, similar ideas can be implemented for numerous numerical schemes, as long as the Taylor expansion of the scheme's error is well behaved. As was discussed in Chapter 4, if $f \in C^2[a, b]$, then for some μ in (a, b), the composite midpoint error with sub intervals of size Δx is given by

$$E = \frac{(b-a)}{24}\Delta x^2 f''(\mu). \tag{7.11}$$

While the constant $f''(\mu)$ is not guaranteed to be the same for different choices of Δx, in practice it is generally well behaved. With an expected second-order behavior, we can perform a similar extrapolation technique as in the previous section, that is our new extrapolated solution M is given by:

$$M = \frac{4M(f, \Delta x/2) - M(f, \Delta x)}{3}, \tag{7.12}$$

where $M(f, h)$ is the solution of the composite midpoint rule applied to the function f with a spacing of size h:

$$M(f, h) = \sum_{i=1}^{N} f(x_i)h, \tag{7.13}$$

with $h = (b - a)/N$ and N being the number of sub-intervals.

To demonstrate this extrapolation technique, Figure 7.2 shows results of performing the extrapolation scheme in (7.12) on the function $f(x) = \sin(x)$ on the interval $[0, \pi/2]$. Errors between the approximated integral and the actual solution are plotted for various numbers of midpoint samples. The composite midpoint

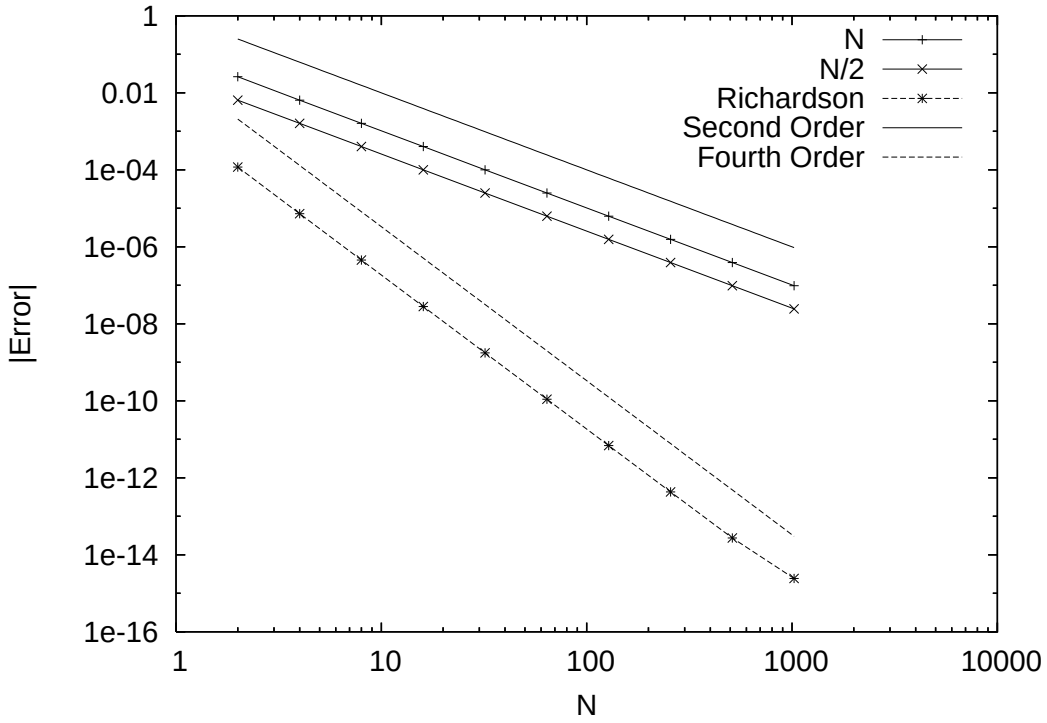


Figure 7.2. Example of Richardson extrapolation in numerical quadrature when integrating $f(x) = \sin(x)$ over the interval $[0, \pi/2]$. Standard composite midpoint integration is shown to exhibit second-order convergence while a Richardson extrapolation based solution is shown to exhibit fourth-order behavior.

rule shows second-order convergence rates, as expected by the above error formula. The Richardson extrapolation scheme does indeed improve convergence rates, with demonstrated rates being fourth-order in this case.

It is in cases like this, where numerous samples are required in a numerical scheme, that extrapolation techniques can provide tremendous advantages. For example, if one wanted to estimate the above integral with an error less than 10^{-6}, the first of our data points which satisfy that requirement is with 512 samples, for an error of 3.921×10^{-7}. The extrapolated solution with 8 samples (using a second calculation with 16 samples) provides an error of 4.522×10^{-7} with a total of 24 function evaluations. In other words, to achieve our required accuracy, standard composite midpoint integration requires 24 times the number of samples as does the Richardson extrapolation solution. This factor only increases as our

error requirements become more stringent.

7.1.3 Extrapolation Techniques in Finite Element Methods

There exists a large volume of literature where Richardson extrapolation techniques are applied to finite element methods. Early work where these techniques were derived and applied can be found in Blum, Lin, and Rannacher [12]. A survey of various techniques can be found in Rannacher [40]. Work in this area continues with recent techniques developed by Asadzadeh, Schatz and Wendland [1]. Most of these techniques are based on so-called h-refinement of a triangle mesh to achieve extrapolated super convergent results. There are two main difficulties in applying these techniques to the Material Point Method. For one, the success of convergence improving extrapolation techniques when applied to finite element type methods depends on the presence of asymptotic error expansions of the type

$$u_h(x) = u(x) + \sum_{k=1}^{n} h^{2k} e^{(k)}(x) + \mathcal{O}(h^{2n}), \tag{7.14}$$

where $u_h(x)$ is the approximated solution at point x, $u(x)$ is the true solution, and $e^{(k)}(x)$ are coefficients independent of the mesh size parameter h [12]. Determining the coefficients $e^{(h)}(x)$ is required when developing the extrapolation techniques. These coefficients are nontrivial to obtain for even fairly simple problems. We are unaware of results where extrapolation techniques have been employed when solving mechanics problems, such as the equation of motion (3.1).

The second difficulty is that most of the extrapolation techniques in the above literature only exhibit super convergent results in the infinity norm at specific points–most often the shared grid nodes between the coarse and refined meshes. A difficulty then arises when applying these techniques to MPM since the method requires evaluating field variables at particle positions at various times in the solution procedure, which does not result in super convergent results.

As has been discussed in previous chapters, quadrature errors seems to be the dominant error in MPM, therefore the remainder of this chapter will focus on using Richardson extrapolation techniques applied to the numerical quadrature inherent in the method to improve results, and not necessarily improve convergence rates.

7.2 Extrapolation Techniques in MPM

Previous chapters have focused on quadrature errors as being one of the dominant errors in MPM. For this reason, we choose to apply extrapolation techniques as demonstrated in Section (7.1.2) to improve numerical integration in MPM. As a reminder, MPM discretizes integrals within procedure by performing sums over particles, multiplying samples of the function being integrated at particles positions by particle volumes. For example, if the function being integrated is $f(x)$, the integration of $f(x)$ over the domain is approximated by:

$$\int_\Omega f(\mathbf{x})\, d\Omega \approx \sum_p f(\mathbf{x}_p) V_p = \sum_p f_p V_p. \tag{7.15}$$

There are multiple points within the MPM algorithm where this type of integration is performed. When projecting particle mass to the grid, the following approximations are made (the remainder of this section will be presented in 1-D for simplification):

$$m_i = \int_\Omega \phi_i(x)\rho(x)\, dx \approx \sum_p \phi_i(x_p)\rho(x_p)V_p = \sum_p \phi_{ip} m_p, \tag{7.16}$$

where ϕ_i is the basis function associated with grid node i, ρ is the density field, and $\rho_p V_p = m_p$. Particle momentum is projected to the grid in a similar way:

$$p_i = \int_\Omega \phi_i(x)\rho(x)v(x)\, dx \approx \sum_p \phi_i(x_p)\rho(x_p)v(x_p)V_p = \sum_p \phi_{ip} m_p v_p, \tag{7.17}$$

where $v(x)$ is the velocity field and v_p is particle velocity. And lastly the internal force calculation on the grid involves another integration:

$$f_i = -\int_\Omega \sigma(x)\cdot\nabla\phi_i(x)\, dx \approx -\sum_p \sigma(x_p)\cdot\nabla\phi_i(x_p)V_p = -\sum_p \sigma_p\cdot\nabla\phi_{ip}V_p. \tag{7.18}$$

As was demonstrated and analyzed in Chapter 4, the above approximations are very much akin to the midpoint integration rule in (7.13), where (for example) $\phi_{ip}\rho_p$ is the sample of our function $\phi_i(x)\rho(x)$ and V_p is our weight. The major difference between this an (7.13) is that our weights are sample dependant, *i.e.* V_p may not be (and is typically not) constant for all p in standard MPM. And furthermore, as

is more fully described in Chapter 4, the particle voxels do not respect breaks in continuity of the integrand, leading to quadrature errors. These quadrature errors are particle-position dependant, and thus the error function is not representable as

$$\varepsilon = \sum_{i=p}^{\infty} h^i e^{(i)}, \tag{7.19}$$

where p is the largest order of the error, h is some measure of resolution, and $e^{(i)}$ is some constant, independent of the spatial discretization, as would be required to implement a simple extrapolation technique. Instead, the constants $e^{(i)}$ are dependent on spatial discretization (or particle positions). This can be seen in our previous work in (6.36) where the error in acceleration looks like $\varepsilon = C\alpha g(t)$, where $g(t)$ is a time-dependant function dependant on particle positions. A first step toward a workable extrapolation solution is to simplify the problem and use moving-mesh MPM as described in Section (3.2.2). In moving mesh, the above three integrals become:

$$m_i = \int_{\Omega^0} \Phi_i(X)\rho(X)\, dX \approx \sum_p \Phi_i(X_p)\rho(X_p)V_p^0 = \sum_p \Phi_{ip} m_p \tag{7.20}$$

$$p_i = \int_{\Omega^0} \Phi_i(X)\rho(X)v(X)\, dX \approx \sum_p \Phi_i(X_p)\rho(X_p)v(X_p)V_p^0 = \sum_p \Phi_{ip} m_p v_p \tag{7.21}$$

$$f_i = -\int_{\Omega^0} P(X)\cdot\nabla\Phi_i(X)\, dX \approx -\sum_p P(X_p)\cdot\nabla\Phi_i(X_p)V_p^0 = -\sum_p P_p\cdot\nabla\Phi_{ip}V_p^0 \tag{7.22}$$

where Ω^0 is the reference domain, X is position in the reference domain, and V_p^0 is the particle width in the reference domain. Further simplifying, if we assume the domain is initially discretized with particles of the same width, *i.e.* $V_p^0 = \Delta x$ for all p, then the above equations reduce to:

$$m_i = \sum_p \Phi_i(X_p)\rho(X_p)\Delta x = \sum_p \Phi_{ip} m_p \tag{7.23}$$

$$p_i = \sum_p \Phi_i(X_p)\rho(X_p)v(X_p)\Delta x = \sum_p \Phi_{ip} m_p v_p \tag{7.24}$$

$$f_i = -\sum_p P(X_p)\cdot\nabla\Phi_i(X_p)\Delta x = -\sum_p P_p\cdot\nabla\Phi_{ip}\Delta x \tag{7.25}$$

at which point the direct relation to the midpoint rule (7.13) should be apparent. Since the above assumptions do leave us with midpoint integration, application of simple Richardson extrapolation techniques to improve any errors in this integration as in (7.12) is possible.

We start our extrapolation technique by discretizing our domain into two separate sets of particles. We will refer to the two sets with subscripts p, and q. The domain is discretized such that Δx_p, or the particle spacing for the p-set of particles is given by $\Delta x_p = h/P$, where h is the grid spacing and P is an integer number of particles-per-cell (PPC). The particles are placed in the domain such that particle boundaries align with grid boundaries, with the particle positions x_p defined as the center of each particle voxel. The second set of particles are initialized in the same way, however $\Delta x_q = h/Q = \frac{1}{2}\Delta x_p$, and $Q = 2P$. In other words, there are twice as many particles in the set of particles $\{x_q\}$ as in the set $\{x_p\}$.

The main difference between moving-mesh MPM and the Richardson extrapolated version is the integration steps above. Separate grid masses are calculated:

$$m_{i,p} = \sum_p \Phi_{ip} m_p, \qquad m_{i,q} = \sum_q \Phi_{iq} m_q. \tag{7.26}$$

Since midpoint integration is second-order with respect to particle spacing, Δx, the extrapolated grid mass is then calculated as

$$m_i = \frac{4m_{i,q} - m_{i,p}}{3} = \frac{4\sum_q \Phi_{iq} m_q - \sum_p \Phi_{ip} m_p}{3}. \tag{7.27}$$

Similarly, extrapolated grid momentum and internal force are calculated as

$$p_i = \frac{4p_{i,q} - p_{i,p}}{3} = \frac{4\sum_q \Phi_{iq} m_q v_q - \sum_p \Phi_{ip} m_p v_p}{3}, \tag{7.28}$$

$$f_i = \frac{4f_{i,q} - f_{i,p}}{3} = \frac{\sum_p P_p \cdot \nabla\Phi_{ip}\Delta x - 4\sum_q P_q \cdot \nabla\Phi_{iq}\Delta x}{3}. \tag{7.29}$$

And lastly, there is usually an initialization of grid displacements at the beginning of the simulation. The initialization step and discretization are given by:

$$u_i = (\int_{\Omega^0} \Phi_i(X) u(X)\, dX)(\int_{\Omega^0} \Phi_i(X)\, dX), \tag{7.30}$$

$$u_i = (4\sum_q \Phi_{iq} u_q - \sum_p \Phi_{ip} u_p)/(4\sum_q \Phi_{iq} - \sum_p \Phi_{ip}). \tag{7.31}$$

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

Solve for mass at grid with p $$m_{i,p} = \sum_p m_p \Phi_{ip} \tag{7.32}$$

Solve for mass at grid with q $$m_{i,q} = \sum_q m_q \Phi_{iq} \tag{7.33}$$

Solve for extrapolated grid mass $$m_i = (4m_{i,q} - m_{i,p})/3 \tag{7.34}$$

Solve for grid momentum with p $$\mathbf{p}_{i,p}^k = \sum_p m_p \mathbf{v}_p^k \Phi_{ip} \tag{7.35}$$

Solve for grid momentum with q $$\mathbf{p}_{i,q}^k = \sum_q m_q \mathbf{v}_q^k \Phi_{iq} \tag{7.36}$$

Solve for extrapolated grid momentum $$\mathbf{p}_i^k = (4p_{i,q} - p_{i,q})/3 \tag{7.37}$$

Solve for grid velocity $$\mathbf{v}_i^k = p_i/m_i \tag{7.38}$$

Solve for external forces with p $$\mathbf{f}_{i,p}^{ext} = \sum_p m_p \mathbf{b}_p \Phi_{ip} \tag{7.39}$$

Solve for external forces with q $$\mathbf{f}_{i,q}^{ext} = \sum_q m_q \mathbf{b}_q \Phi_{iq} \tag{7.40}$$

Solve for extrapolated external forces $$\mathbf{f}_i^{ext} = (4f_{i,q}^{ext} - f_{i,p}^{ext})/3 \tag{7.41}$$

Solve for internal forces with p $$\mathbf{f}_{i,p}^{int} = -\sum_p \mathbf{P}_p^k \cdot \nabla \Phi_{ip} \Delta x \tag{7.42}$$

Solve for internal forces with q $$\mathbf{f}_{i,q}^{int} = -\sum_q \mathbf{P}_q^k \cdot \nabla \Phi_{iq} \Delta x \tag{7.43}$$

Solve for extrapolated internal forces $$\mathbf{f}_i^{int} = (4f_{i,q}^{int} - f_{i,p}^{int})/3 \tag{7.44}$$

Solve for grid acceleration $$\mathbf{a}_i^k = (\mathbf{f}_i^{int} + \mathbf{f}_i^{ext})/m_i \tag{7.45}$$

Time advance grid velocity $$\mathbf{v}_i^{k+1} = \mathbf{v}_i^k + \mathbf{a}_i^k \Delta t \tag{7.46}$$

Time advance grid displacements $$\mathbf{u}_i^{k+1} = \mathbf{u}_i^k + \mathbf{v}_i^{k+1} \Delta t \tag{7.47}$$

Advance particle deformation gradient $$\mathbf{F}_{p,q}^{k+1} = 1 + \sum_i \mathbf{u}_i^{k+1} \cdot \nabla \Phi_{ip,q} \tag{7.48}$$

Solve constitutive model $$\mathbf{P}_{p,q}^{k+1} = \mathbf{P}(\mathbf{F}_{p,q}^{k+1}) \tag{7.49}$$

Time advance particle velocities $$\mathbf{v}_{p,q}^{k+1} = \mathbf{v}_{p,q}^k + \Delta t \sum_i \mathbf{a}_i^k \Phi_{ip,q} \tag{7.50}$$

Time advance particle displacements $$\mathbf{u}_{p,q}^{k+1} = \mathbf{u}_{p,q}^k + \Delta t \sum_i \mathbf{v}_i^{k+1} \Phi_{ip,q}. \tag{7.51}$$

7.3 Results

To test the above method, we simulate the same 1-D periodic bar described in Section (5.2.2) with the parameters $A = 0.05$ (five percent maximum displacement), $E = 10^4$, $\rho_0 = 1.0$, and time-steps which correspond to a CFL of 0.2. The problem was solved with periodic, moving-mesh MPM using standard piecewise-linear basis functions. The simulations were run with standard moving-mesh MPM as outlined in Section (3.2.2) with both two and six particles-per-cell (PPC). Another simulation was run using our extrapolation techniques, with $P = 2$, $Q = 4$, thereby giving us a total of $6N$ particles (with N being the number of grid cells), the same number of particles as the standard moving-mesh MPM simulation with six particles-per-cell (PPC). Convergence results for the three tests are shown in Figure 7.3. Here, we can see the effects of the quadrature errors do not show themselves until 8,192 grid cells, when errors in the two particles-per-cell (PPC) solution (using standard moving-mesh MPM) start to level off. At 16,384 grid cells, errors in the the six particles-per-cell (PPC) solution begin to level off. For both this, and the next grid resolution, the extrapolated solution using two and four particles-per-cell (PPC) outperforms the six particles-per-cell (PPC) standard solution.

7.4 Summary and Conclusions

In this chapter, we reviewed the mathematics behind Richardson extrapolation techniques and demonstrated how the technique can be applied in situations other than standard h-refinement of finite difference solvers where Richardson extrapolation is commonly employed. We showed how Richardson extrapolation can be used in numerical integration methods such as the midpoint rule, where fourth-order convergence can be obtained by combining results where two different numbers of sample points have been used. We developed a modification to moving-mesh MPM which employs these extrapolation techniques to improve numerical quadrature in the method. We showed that these extrapolation techniques outperformed standard moving-mesh MPM when similar numbers of particles were used.

Richardson extrapolation applied to improve quadrature errors in moving-mesh

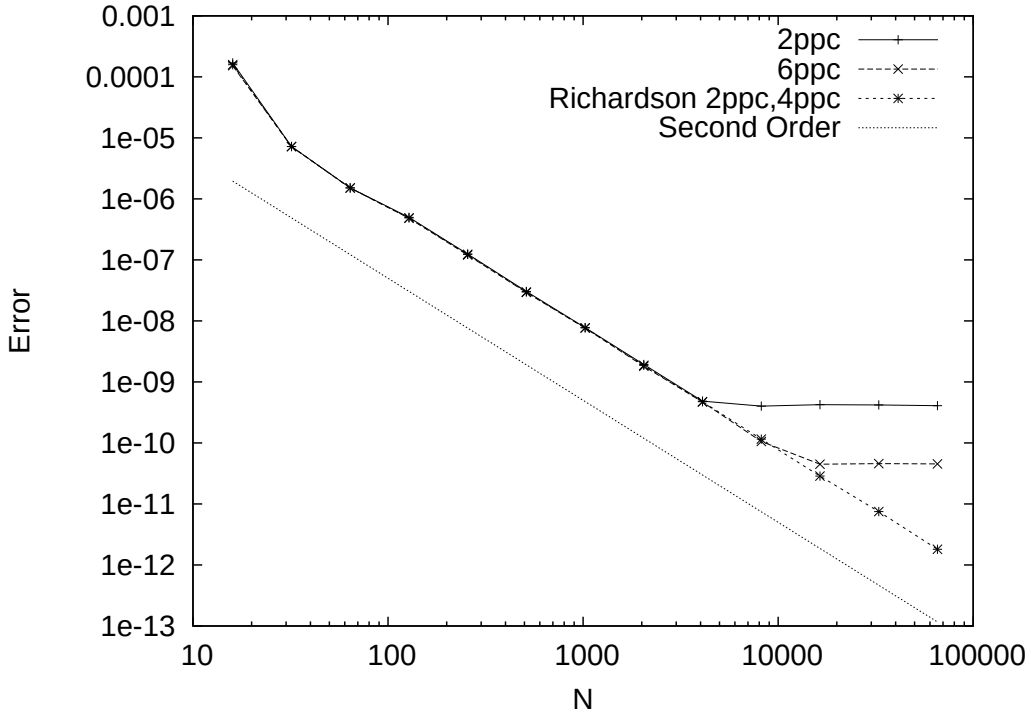


Figure 7.3. Results for Richardson extrapolation techniques applied to moving-mesh MPM. Results for standard moving-mesh MPM with two particles-per-cell (PPC) and six particles-per-cell (PPC) are compared to the Richardson extrapolated solution when the two-particles-per-cell (PPC) and four particles-per-cell (PPC) results are combined. Here, the Richardson extrapolated solution out-performs the six particles-per-cell (PPC) solution, even though the same number of particles are used.

MPM was the most straight-forward application of these techniques to the method. Currently there are numerous second-order errors in standard MPM, even when more advanced basis functions are used. Some of the many errors include the quadrature errors we have previously discussed, grid-crossing, and mass lumping errors, the combination of which makes the application of Richardson extrapolation to achieve an MPM algorithm which is higher than second order (spatially) non-trivial. This is a worthwhile goal, however, and is the subject of future work. Since quadrature errors are much greater in standard (non moving-mesh) MPM and still dominate many of our simulations, a good middle ground may be developing an extrapolation method specifically for standard MPM.

CHAPTER 8

FUTURE WORK

Throughout this dissertation work, we have been conscious to offer analysis and improvements to the Material Point Method, while at the same time not diverging from what we believe to be core attributes giving MPM its unique benefits. One of these attributes is the implicit mass lumping as has been presented throughout this work. While MPM implementations exist which use full mass-matrices [32, 33, 48], it is our experience that the vast majority of practitioners use the implicit mass lumping techniques. Mass lumping allows for trivial solutions to the linear system which do not require a large matrix solve. This, in turn, allows for efficient parallel implementations of the method. The second intrinsic feature of the method which we felt the need to preserve is the nodal quadrature employed in the method. Nodal quadrature makes approximations to integrals extraordinarily easy to compute and requires no neighborhood information or neighbor-finding routines, other than simple calculations to find out which grid cell contains a particular particle. This is a constant-time calculation which makes implementation of the method relatively easy. A number of interesting paths could emerge if we were to relax these requirements and expand upon the material already presented.

While much attention was spent on numerical integration in the method, there is still much that can be improved. In his dissertation, Wallstedt [54] diverged from the nodal-quadrature scheme in MPM, and employed least-squared based integration to better approximate the integrals in the method. While not fully developed, at various times we have played with similar techniques, using Lagrange interpolating polynomials through points, and integrating the interpolant using standard Gaussian quadrature. These early attempts at improved integration

showed promising results, but new difficulties arose. In particular, if we are interpolating and integrating field variables, the domain which we are integrating must be clearly defined. Wallstedt also made progress toward one solution, using boundary particles and performing least squares calculations to estimate boundary geometry. This boundary geometry then defined the exact regions to be integrated. We believe that significant improvements to the method will require attention to be focused in this particular area–improvement of the integration of field variables, and better treatment of the domain and boundary geometries. There are numerous graphics and mesh-free algorithms which may provide inspiration for future improvements.

We believe the extrapolation techniques presented in Chapter 7 just scratch the surface of the benefits they may provide to the method. The work presented here only focuses on moving-mesh MPM, and while useful as a jumping-off point, moving-mesh MPM is only useful in small-deformation problems. Expansion of the techniques to standard MPM will require careful attention to particle voxel and grid cell alignment issues, but if successful, may provide a tremendous improvement in errors.

Along with the reduction of errors, extrapolation techniques can also be used in error estimation and error control. Dynamic error control, such as automatic time-step selection, or spatial refinement strategies, requires a thorough understanding of the errors in a method. Dynamic error control could also be used to drive adaptive *hp*-refinement strategies–often used in the finite element community. The ability to monitor and control these errors in a simulation is key to verification and validation efforts which are becoming ever more important as people are wanting to put trust in numerical simulation results. This body of work has moved us much closer to understanding the many subtle errors in this method, and makes dynamic error estimation a realistic goal.

APPENDIX

A.1 Integrating Piecewise-Constant Functions with the Midpoint Rule

If $y(x)$ is piecewise constant with a discontinuity at $x = 0$, as shown in Figure 4.3, then y can be thought of as

$$y(x) = \begin{cases} y_1(x) = a_1 & : \quad x \leq 0 \\ y_2(x) = a_2 & : \quad x > 0. \end{cases} \tag{A.1}$$

Assuming $0 \in [\xi - \Delta x/2, \xi + \Delta x/2]$, the exact integral of $y(x)$ over the region needs to be evaluated in two parts,

$$\int_{\xi - \frac{\Delta x}{2}}^{0} y_1 \, dx = a_1 x|_{\xi - \frac{\Delta x}{2}}^{0} = -a_1(\xi - \frac{\Delta x}{2}) \tag{A.2}$$

and

$$\int_{0}^{\xi + \frac{\Delta x}{2}} y_2 \, dx = a_2(\xi + \frac{\Delta x}{2}), \tag{A.3}$$

giving the total integral over the region

$$\int_{\xi - \frac{\Delta x}{2}}^{\xi + \frac{\Delta x}{2}} y(x) \, dx = (a_2 - a_1)\xi + \frac{1}{2}(a_2 + a_1)\Delta x. \tag{A.4}$$

Using the midpoint rule over the same region gives $\mathrm{MP}(\xi) = y(\xi)\Delta x$. When $\xi \leq 0$, $\mathrm{MP}(\xi) = a_1 \Delta x$. The integration, or quadrature error in this case is

$$E(\xi) = \int_{\xi - \frac{\Delta x}{2}}^{\xi + \frac{\Delta x}{2}} y(x) \, dx - \mathrm{MP}(\xi) = (a_2 - a_1)[\xi + \frac{1}{2}\Delta x]. \tag{A.5}$$

The only value for ξ which gives zero error in (A.5) is $\xi = -\Delta x/2$ which corresponds to a zero error when integrating the region $[-\Delta x, 0]$. Since $y(x) = y_1$, with no jumps on the open interval $(-\Delta x, 0)$, a zero error when $\xi = -\Delta x/2$ make sense because the midpoint rule can integrate a constant function exactly. When $\xi > 0$, a similar analysis shows $E(\Delta x/2) = 0$. The maximum error magnitude occurs when $\xi = 0$, giving

$$E_{max} = E(0) = \frac{1}{2}|a_2 - a_1|\Delta x. \tag{A.6}$$

A.2 Integrating Piecewise-Linear Functions with the Midpoint Rule

If $y(x)$ is piecewise linear, as shown in Figure 4.4, composed of $y_1(x)$ for $x \leq 0$ and $y_2(x)$ for $x > 0$, with $y_1(0) = y_2(0) = 0$, and $y_1'(0) \neq y_2'(0)$, $y(x)$ can be written as

$$y(x) = \begin{cases} y_1(x) = a_1 x & : \quad x \leq 0 \\ y_2(x) = a_2 x & : \quad x > 0. \end{cases} \tag{A.7}$$

The exact integral of $y(x)$ over the region $[\xi - \Delta x/2, \xi + \Delta x/2]$, needs to be evaluated in two parts,

$$\int_{\xi - \frac{\Delta x}{2}}^{0} y_1 \, dx = a_1 x^2 |_{\xi - \frac{\Delta x}{2}}^{0} = -\frac{1}{2} a_1 (\xi^2 - \xi \Delta x + \frac{1}{4} \Delta x^2) \tag{A.8}$$

and

$$\int_{0}^{\xi + \frac{\Delta x}{2}} y_2 \, dx = a_2 x^2 |_{0}^{\xi + \frac{\Delta x}{2}} = \frac{1}{2} a_2 (\xi^2 + \xi \Delta x + \frac{1}{4} \Delta x^2), \tag{A.9}$$

giving the total integral over the region

$$\int_{\xi - \frac{\Delta x}{2}}^{\xi + \frac{\Delta x}{2}} y(x) \, dx = \frac{1}{2}(a_2 - a_1)\xi^2 + \frac{1}{2}(a_2 + a_1)\xi \Delta x + \frac{1}{8}(a_2 - a_1)\Delta x^2. \tag{A.10}$$

Using the midpoint rule over the same region gives $\mathrm{MP}(\xi) = y(\xi)\Delta x$. When $\xi \leq 0$, $\mathrm{MP}(\xi) = a_1 \xi \Delta x$. The integration, or quadrature error in this case is

$$E_1(\xi) = \int_{\xi - \frac{\Delta x}{2}}^{\xi + \frac{\Delta x}{2}} y(x) \, dx - \mathrm{MP}(\xi) = \frac{1}{2}(a_2 - a_1)\xi^2 + \frac{1}{2}(a_2 - a_1)\xi \Delta x + \frac{1}{8}(a_2 - a_1)\Delta x^2. \tag{A.11}$$

To find values of ξ corresponding to maximum or minimum error, we solve $\frac{d}{d\xi} E_1 = 0$ for ξ:

$$\frac{d}{d\xi} E_1 = (a_2 - a_1)\xi + \frac{1}{2}(a_2 - a_1)\Delta x \tag{A.12}$$

$$= (a_2 - a_1)(\xi + \frac{1}{2}\Delta x). \tag{A.13}$$

Since $a_2 \neq a_1$, the above equation is zero only when $\xi = -\Delta x/2$. A similar analysis for when $\xi > 0$ shows $\frac{d}{d\xi} E_2 = 0$ only when $\xi = \Delta x/2$. To find the maximum magnitude, we first note that $E(\pm \Delta x/2) = 0$ which also says that the midpoint

rule integrates a linear function exactly. Lastly, since $E_1(0) = E_2(0)$, the maximum magnitude error must be when $\xi = 0$ giving

$$E_{max} = E(0) = \frac{1}{8}|a_2 - a_1|\Delta x^2. \tag{A.14}$$

A.3 Integrating Piecewise-Quadratic Functions with the Midpoint Rule

If $y(x)$ is piecewise quadratic, as shown in Figure 4.5, composed of $y_1(x)$ for $x \leq 0$ and $y_2(x)$ for $x > 0$, with $y_1(0) = y_2(0) = 0$ and $y_1'(0) = y_2'(0)$, and $y_1''(0) \neq y_2''(0)$, $y(x)$ can be written as

$$y(x) = \begin{cases} y_1 = a_1x^2 + bx & : \quad x \leq 0 \\ y_2 = a_2x^2 + bx & : \quad x > 0. \end{cases} \tag{A.15}$$

The exact integral of $y(x)$ over the region $[\xi - \Delta x/2, \xi + \Delta x/2]$, needs to be evaluated in two parts,

$$\begin{aligned} \int_{\xi-\frac{\Delta x}{2}}^{0} y_1 &= [\frac{1}{3}a_1x^3 + \frac{1}{2}bx^2]_{\xi-\frac{\Delta x}{2}}^{0} \\ &= -[\frac{1}{3}a_1(\xi - \frac{\Delta x}{2})^3 + \frac{1}{2}b(\xi - \frac{\Delta x}{2})^2] \\ &= -\frac{1}{3}a_1(\xi^3 - \frac{3}{2}\xi^2\Delta x + \frac{3}{4}\xi\Delta x^2 - \frac{1}{8}\Delta x^3) - \frac{1}{2}b(\xi^2 - \xi\Delta x + \frac{1}{4}\Delta x^2) \end{aligned} \tag{A.16}$$

and

$$\begin{aligned} \int_{0}^{\xi+\frac{\Delta x}{2}} y_2 &= [\frac{1}{3}a_2x^3 + \frac{1}{2}bx^2]_{0}^{\xi+\frac{\Delta x}{2}} \\ &= \frac{1}{3}a_2(\xi^3 + \frac{3}{2}\xi^2\Delta x + \frac{3}{4}\xi\Delta x^2 + \frac{1}{8}\Delta x^3) + \frac{1}{2}b(\xi^2 + \xi\Delta x + \frac{1}{4}\Delta x^2) \end{aligned} \tag{A.17}$$

giving the total integral over the region

$$\begin{aligned} \int_{\xi-\frac{\Delta x}{2}}^{\xi+\frac{\Delta x}{2}} y &= \int_{\xi-\frac{\Delta x}{2}}^{0} y_1 + \int_{0}^{\xi+\frac{\Delta x}{2}} y_2 \\ &= \frac{1}{3}(a_2 - a_1)\xi^3 + \frac{1}{2}(a_2 + a_1)\Delta x\xi^2 \\ &\quad + \frac{1}{4}(a_2 - a_1)\Delta x^2\xi + \frac{1}{24}(a_2 + a_1)\Delta x^3 + \xi b\Delta x. \end{aligned} \tag{A.18}$$

Using the midpoint rule over the same region gives $\mathrm{MP}(\xi) = y(\xi)\Delta x$. When $\xi \leq 0$, the midpoint rule gives

$$\mathrm{MP}(\xi) = (a_1\xi^2 + b\xi)\Delta x. \tag{A.19}$$

The integration error across the discontinuity is then

$$\begin{aligned} E_1(\xi) &= \int_{\xi-\frac{\Delta x}{2}}^{\xi+\frac{\Delta x}{2}} y - \mathrm{MP}(\xi) \\ &= \frac{1}{3}(a_2-a_1)\xi^3 + \frac{1}{2}(a_2-a_1)\Delta x\xi^2 + \frac{1}{4}(a_2-a_1)\Delta x^2\xi + \frac{1}{24}(a_2+a_1)\Delta x^3 \end{aligned} \tag{A.20}$$

To find the ξ corresponding to maximum error, we solve for $\frac{d}{d\xi}E_1 = 0$.

$$\begin{aligned} \frac{d}{d\xi}E_1 &= (a_2-a_1)\xi^2 + (a_2-a_1)\Delta x\xi + \frac{1}{4}(a_2-a_1)\Delta x^2 \\ &= (a_2-a_1)(\xi+\frac{1}{2}\Delta x)^2. \end{aligned} \tag{A.21}$$

Since $a_2 \neq a_1$, this is zero only when $\xi = -\frac{1}{2}\Delta x$. A similar analysis for when $\xi > 0$ shows $\frac{d}{d\xi}E_2 = 0$ only when $\xi = \Delta x/2$. Since $E(\pm\frac{1}{2}\Delta x) = 0$, $|E|$ must be a maximum at $\xi = 0$. Therefore, the maximum error is

$$E_{max} = E(0) = \frac{1}{24}|a_1 + a_2|\Delta x^3.$$

REFERENCES

[1] Asadzadeh, M., Schatz, A. H., and Wendland, W. A new approach to richardson extrapolation in the finite element method for second order elliptic problems. *Mathematics of Computation 78*, 268 (2009), 1951–1973.

[2] Atluri, S. N. Meshless Local Petrov-Galerkin (MLPG) mixed collocation method for elasticity problems. *Computer Modeling in Engineering & Sciences 14*, 3 (2006), 141–152.

[3] Atluri, S. N., Liu, H. T., and Han, Z. D. Meshless Local Petrov-Galerkin (MLPG) mixed finite difference method for solid mechanics. *Computer Modeling in Engineering & Sciences 15*, 1 (2006), 1–16.

[4] Atluri, S. N., and Zhu, T. A new Meshless Local Petrov-Galerkin (MLPG) approach in computational mechanics. *Computational Mechanics 22*, 2 (1998), 117–127.

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

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

[7] Bardenhagen, S. G., Brydon, A. D., and Guilkey, J. E. Insight into the physics of foam densification via numerical simulation. *Journal of the Mechanics and Physics of Solids 53*, 3 (2005), 597–617.

[8] Bardenhagen, S. G., and Kober, E. M. The generalized interpolation material point method. *Computer Modeling in Engineering and Science 5*, 6 (2004), 477–495.

[9] Beissel, S., and Belytschko, T. Nodal integration of the element-free Galerkin method. *Computer Methods in Applied Mechanics and Engineering 139*, 1 (1996), 49–74.

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

[11] Belytschko, T., Lu, Y. Y., and Gu, L. Element free Galerkin methods. *International Journal for Numerical Methods in Engineering 37*, 2 (1994), 229–256.

[12] Blum, H., Lin, Q., and Rannacher, R. Asymptotic error expansion and richardson extrapolation for linear finite elements. *Numerische Mathematik 49*, 1 (1986), 11–37.

[13] Brackbill, J. U. The ringing instability in particle-incell calculations of low-speed flow. *Journal of Computational Physics 75* (1988), 469–492.

[14] Brackbill, J. U. Particle methods. *International Journal of Numerical Methods in Fluids 47* (2005), 693–705.

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

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

[17] Chen, J. S., Yoon, S., and Wu, C. T. Non-linear verson of stabilized conforming nodal integration for galerkin mesh-free methods. *International Journal of Numerical Methods in Engineering 53*, 12 (2002), 2587–2615.

[18] de St. Germain, J. D., Parker, S. G., McCorquodale, J., and Johnson, C. R. Uintah: A Massively Parallel Problem Solving Environment. In *HPDC* (2000), pp. 33–42.

[19] Dolbow, J., and Belytschko, T. Numerical integration of the galerkin weak form in meshfree methods. *Computational Mechanics 23*, 3 (1999), 1432–0924.

[20] Grigoryev, Y. N., Vshivkov, V. A., and Fedoruk, M. P. *Numerical "Particle-in-Cell" Methods*. VSP BV, 2002.

[21] Guilkey, J. E., Harman, T. B., and Banerjee, B. An Eulerian-Lagrangian approach for simulating explosions of energetic devices. *Computers and Structures 85*, 11-14 (2007), 660–674.

[22] Guilkey, J. E., Hoying, J. B., and Weiss, J. A. Computational modeling of multicellular constructs with the ma terial point method. *Journal of Biomechanics 39*, 11 (2006), 2074–2086.

[23] Guilkey, J. E., and Weiss, J. A. Implicit time integration for the material point method: Quantitative and algorithmic comparisons with the finite element method. *International Journal for Numerical Methods in Engineering 57*, 9 (2003), 1323–1338.

[24] Hairer, E., Nrsett, S., and Wanner, G. *Solving Ordinary Differential Equations I.* Springer, 1993.

[25] Han, Z. D., Liu, H. T., Rajendran, A. M., and Atluri, S. N. The applications of Meshless Local Petrov-Galerkin (MLPG) approaches in high-speed impact, penetration and perforation problems. *Computer Modeling in Engineering & Sciences 14*, 2 (2006), 119–128.

[26] Han, Z. D., Rajendran, A. M., and Atluri, S. N. Meshless Local Petrov-Galerkin (MLPG) approaches for solving nonlinear problems with large deformations and rotations. *Computer Modeling in Engineering & Sciences 10*, 1 (2005), 1–12.

[27] Hughes, T. J. R. *The finite element method: linear static and dynamic finite element analysis*. Prentice-Hall, 1987.

[28] Kashiwa, B. A., Lewis, M. L., and Wilson, T. Fluid-structure interaction modeling. Tech. Rep. LA-13111-PR, Los Alamos National Laboratory, Los Alamos, 1996.

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

[30] Lawson, J., Berzins, M., and Dew, P. Balancing space and time errors in the method of lines for par abolic equations. *SIAM Journal on Scientific and Statistical Computing 12*, 3 (1991), 573–594.

[31] Li, S., and Liu, W. K. *Meshfree Particle Methods*. Springer, 2004.

[32] Love, E., and Sulsky, D. L. An energy-consistent material-point method for dynamic finite deformation plasticity. *International Journal for Numerical Methods in Engineering 65*, 10 (2006), 1608–1638.

[33] Love, E., and Sulsky, D. L. An unconditionally stable, energy-momentum consistent implementation of the material-point method. *Computer Methods in Applied Mechanics and Engineering 195*, 33-36 (2006), 3903–3925.

[34] Ma, J., Lu, H., and Komanduri, R. Structured mesh refinement in generalized interpolation material point (GIMP) method for simulation of dynamic problems. *Computer Methods in Applied Mechancs and Engineering 12* (2006), 213–227.

[35] Nairn, J. A. Numerical simulations of transverse compression and densification in wood. *Wood and Fiber Science 38*, 4 (2006), 576–591.

[36] Oñate, E., and Idelsohn, S. R. The particle finite element method. an overview. *International Journal of Computational Methods 1*, 2 (2004), 267–307.

[37] Parker, S., Guilkey, J., and Harman, T. A component-based parallel infrastructure for the simulation of fluid-structure interaction. *Engineering With Computers 22*, 3 (2006), 277–292.

[38] Quinlan, N. J., Basa, M., and Lastiwka, M. Truncation error in mesh-free particle methods. *International Journal of Numerical Methods in Engineering 66*, 13 (2006), 2064–2085.

[39] Ralston, A., and Rabinowitz, P. *A First Course in Numerical Analysis*, second ed. Dover, 2001.

[40] Rannacher, R. Extrapolation techniques in the finite element method: A survey. In *Proceedings of the Summer School in Numerical Analysis at Helsinki* (1988), pp. 80–113.

[41] Richardson, L. F. The approximate arithmetical solution by finite differences of physical problems involving differential equations, with an application to the stresses in a masonry dam. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character 210* (1911), 307–357.

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

[43] Spencer, A. J. M. *Continuum Mechanics*. Dover, 2004.

[44] Steffen, M., Kirby, R. M., and Berzins, M. Analysis and reduction of quadrature errors in the material point method (MPM). *International Journal for Numerical Methods in Engineering 76*, 6 (2008), 922–948. DOI: 10.1002/nme.2360.

[45] Steffen, M., Kirby, R. M., and Berzins, M. Decoupling and balancing of space and time errors in the material point method (MPM). *International Journal for Numerical Methods in Engineering* (2009). submitted.

[46] Steffen, M., Wallstedt, P. C., Guilkey, J. E., Kirby, R. M., and Berzins, M. Examination and analysis of implementation choices within the material point method (MPM). *Computer Modeling in Engineering & Sciences 32*, 2 (2008), 107–127.

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

[48] Sulsky, D., and Kaul, A. Implicit dynamics in the material-point method. *Computer Methods in Applied Mechanics and Engineering 193*, 12-14 (2004), 1137–1170.

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

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

[51] Tran, L. T., Kim, J., and 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.

[52] Vshivkov, V. A. The approximation properties of the particles-in-cells method. *Computational Mathematics and Mathematical Physics 36*, 4 (1996), 509–515.

[53] Wallstedt, P., and Guilkey, J. Improved velocity projection for the material point method. *Computer Modeling in Engineering and Science 19*, 3 (2007), 223–232.

[54] Wallstedt, P. C. *On the Order of Accuracy of the Generalized Interpolation Material Point Method*. PhD thesis, University of Utah, 2008.

[55] Wallstedt, P. C., and Guilkey, J. E. An evaluation of explicit time integration schemes for use with the generalized interpolation material point method. *Journal of Computational Physics 227*, 22 (2008), 9628–9642.

[56] 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*. PhD thesis, Colorado School of Mines, 2008.

[57] Zienkiewicz, O. C., and Taylor, R. L. *The Finite Element Method*, fourth ed., vol. 1. McGraw-Hill, 1989.