RINGING INSTABILITIES IN PARTICLE METHODS

by

Christopher E. Gritton

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

Master of Science

in

Computational Engineering and Science

School of Computing

The University of Utah

August 2014

The University of Utah Graduate School

STATEMENT OF THESIS APPROVAL

The thesis of **Christopher E. Gritton**

has been approved by the following supervisory committee members:

Martin Berzins	, Co-Chair	**3/10/14** Date Approved
Mike Kirby	, Co-Chair	**3/10/14** Date Approved
Aaron Fogelson	, Member	**3/16/14** Date Approved

and by **Martin Berzins** , Chair/Dean of

the Department/College/School of **Computational Engineering and Science**

and by David B. Kieda, Dean of The Graduate School.

ABSTRACT

Particle methods have been used in fields ranging from fluid dynamics to plasma physics. The Particle-In-Cell method and the family of methods that are an extension of it are a combination of both Lagrangian and Eularian methods. In this thesis, we present a brief survey of some of the methods and their key components. We show the different methods by which spatial derviates are computed. We propose a method of showing how the so-called "ringing instabilies" associated with particle methods arise and a means to remove them. We also propose that the underlying nodal scheme plays a key role in the stability of the method. Lastly, different particle methods are explored through numerical simulations and compared against an analytic solution.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

First and foremost, I would like to thank my co-advisor Dr. Martin Berzins for his guidance in this work and for all his time spent on my behalf. I have learned so much about what it takes to do research and how to go after the hard questions. Special thanks also to my co-advisor Dr. Mike Kirby for his ideas and insight and the time he put in to read over different parts and iterations of this work. This thesis has truly been a collaborative effort and I am just glad to be along for the ride.

Thanks to Dr. Aaron Fogelson for his classes that have laid the foundation for much of what I know in applied math and for his time serving on my committee.

Thanks to all my friends for their support.

Last I would like to thank my family for all they do for me, the support they give, and for keeping my sane along the way.

Research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-12-2-0023. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

CHAPTER 1

INTRODUCTION

Since the late 1940s, particle methods [29] have been an evolving and growing topic. Particle methods have been used to solve problems in fields ranging from plasma physics to material and fluid mechanics to astrophysics [14, 20, 4, 32, 9]. Along the way, numerous methods such as Particle-In-Cell, Smoothed-Particle-Hydrodynamics, and Fluid Implicit Particles [19, 26, 7] have come out of these fields. The Material Point Method, MPM [32], is a particle method that is currently finding success in computational mechanics. Another area in which MPM is being used is the field of multiscale/multiphysics simulations [16, 11, 10, 24].

Given that MPM's usage is growing in a range of subjects in materials modeling [34, 28, 36, 21] and given it is a particle method, MPM is a good choice for coupling atomistic and continuum models [24, 16, 11]. Despite these positives, little is known about the stability of MPM. If MPM is to continue to be used in multiscale modeling more needs to be done to understand the stability characteristics of the method. The purpose of this thesis is to seek an understanding of the stability issues associated with particle methods in general and in particular apply this understanding to MPM.

The contributions of this thesis are concerned with a the study of the "ringing instability" [5], and consist of two methods for removing the "ringing instability", and a method for understanding the underlying nodal scheme.

The rest of the thesis is as follows. First, there will be an overview of the historic development of particle methods. The overview will cover those methods that are close derivatives of the original Particle-In-Cell method. Other particle methods such as Smoothed-Particle-Hydrodynamics will not be covered. Second is an examination of the different methods of computing gradients and the stability issues, namely the "ringing instability", that are associated with the different methods. Third is an analysis of the underlying nodal schemes of particle methods and how they affect stability. Finally, there is a comparison of the different methods.

CHAPTER 2

THE DEVELOPMENT OF PARTICLE METHODS

This chapter will cover the historical development of particle methods. This will not be an all-inclusive survey but will cover some of the key points and methods that have helped to push forward the state-of-the-art in particle methods.

2.1 Harlow's Particle-in-Cell

The Particle-In-Cell (PIC) method was originally designed to solve fluid dynamics problems that involve large slips and distortions [17]. Purely Lagrangian methods have been used to successfully model fluid dynamics problems but have problems with large distortions and slippages [19]. Purely Eulerian methods do not have problems with distortions and slippages, but they lack the ability to easily handle boundaries between materials [19]. The PIC method combines properties from both classes of methods in an attempt to combine the positives of both the purely Lagrangian and purely Eulerian methods.

In order to better understand how the PIC method works, we will use the following set of equations that arise from gas dynamics,

$$\frac{\partial \rho}{\partial t} + \rho \frac{\partial u}{\partial x} + u \frac{\partial \rho}{\partial x} = 0, \tag{2.1}$$

$$\rho \left(\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} \right) + \frac{\partial p}{\partial x} = 0, \tag{2.2}$$

$$p = c^2 \rho, \tag{2.3}$$

where Equations 2.1 and 2.2 are the equations for the conservation of mass and momentum, respectively, and Equation 2.3 is the equation of state. The symbols ρ, u, and p represent density, velocity, and pressure, respectively, and c is the wave speed. Most PIC codes would include an equation for the conservation of energy, but for simplification purposes, we focus on the conservation of mass and momentum. It is helpful to rewrite these conservation equations in terms of their material derivatives.

$$\frac{d\rho}{dt} = -\rho\frac{\partial u}{\partial x}, \tag{2.4}$$

$$\rho\frac{du}{dt} = -c^2\frac{\partial \rho}{\partial x}. \tag{2.5}$$

PIC discretizes the material domain into mesh cells and represents the solution as particles on this mesh. In PIC material, properties such as temperature, energy, and velocity are stored at the cells and mass is carried with the particles. For our example, we are only concerned with mass and velocity. Other properties such as density and pressure will be considered as needed.

The first phase in the method is to calculate the cell densities and update intermediate material values for each cell [17, 18, 13]. The density is calculated by summing up the particle masses and dividing by the cell volume. For our model problem, we are only dealing with one material type, thus the density calculation is simply,

$$\rho_i = \frac{N_i m}{h} \tag{2.6}$$

where N_i are the number of particles in the cell and h is the cell width. The index i will be used for cell indexing and p will be used for a particle index. With cell densities calculated, we can now use Equation 2.5 to calculate accelerations for each cell. This is done by using a centered finite difference scheme at the cell centers:

$$\rho_i\frac{du_i}{dt} = -c^2\frac{\rho_{i+1} - \rho_{i-1}}{2h}. \tag{2.7}$$

Alternatively, we can rearrange terms to get:

$$\begin{aligned} \frac{du_i}{dt} &= -\frac{c^2}{\rho_i}\frac{\rho_{i+1} - \rho_{i-1}}{2h} && (2.8)\\ &= -\frac{c^2 h}{N_i m}\frac{N_{i+1}m - N_{i-1}m}{2h^2} && (2.9)\\ &= -c^2\frac{N_{i+1} - N_{i-1}}{2hN_i}. && (2.10) \end{aligned}$$

Once the accelerations are calculated, we can now update the cell velocity as follows,

$$\overline{u}_i = u_i^n + \frac{du_i}{dt}dt, \tag{2.11}$$

where dt is the time step, and where the velocity $\overline{u}_i$ is an intermediate value for cell velocity.

The second phase of the method [13] is to move the particles. Particle velocities are calculated via linear interpolation between cell centers using the intermediate cell velocity given by Equation 2.11. If a particle is located between cell center $i+1$ and cell center i, the particle velocity is,

$$u_p = \phi_{i+1,p}\overline{u}_{i+1} + \phi_{i,p}\overline{u}_i, \tag{2.12}$$

where ϕ is the linear basis function defined as,

$$\phi_i(x) = \begin{cases} 1 - \frac{x-x_i}{h} & \text{if } x_i - h \leq x \leq x_i, \\ 1 + \frac{x-x_i}{h} & \text{if } x_i < x \leq x_i + h, \end{cases} \tag{2.13}$$

and $\phi_{i,p}$ is ϕ evaluated using the position of particle p and the location of the center of cell i, $\phi_{i,p} = \phi(x_p, x_i)$. The updated position for the particle is now

$$x_p^{n+1} = x_p^n + u_p dt. \tag{2.14}$$

The third phase is to account for cell crossings [13]. If no particles cross into a cell, then $u_i^{n+1} = \overline{u}_i$, but if a cell does cross over, then we need to account for a change in momentum. Take, for example, when a particle crosses over from cell $i-1$ to cell i, then that particle carries with it momentum $m\overline{u}_{i-1}$. We update the new cell velocity by taking into account the change in momentum caused by the particle crossing as follows,

$$u_i^{n+1} = \frac{N_i^n \overline{u}_i + \overline{u}_{i-1}}{N_i^n + 1}. \tag{2.15}$$

This same of idea of particle "bookkeeping" needs to be applied to cases where particles may enter or leave a cell. It is this "averaging out" of a material property over a cell that leads to the numerical diffusion problems associated with PIC [7]. When we look at the energy, this averaging out process will always lead to a negative change in kinetic energy [17].

These three phases compose one computation cycle of the PIC method. The PIC method developed by Harlow presented a novel method for combining both Lagrangian and Eulerian methods that could resolve problems associated with material boundaries, slippages, and distortions. Its drawback is the numerical diffusion of material properties [7].

2.2 Full Particle PIC

In the original PIC method, mass is conserved because it is a material property that is carried with the particle. As has been discussed above, properties such as energy and velocity are cell properties that are subject to numerical diffusion [7]. A solution to this problem is to have the particles carry the material properties. By having velocity and energy carried with the particles, quantities such as momentum and energy do not suffer from the numerical diffusion found in the original PIC method [7]. Brackbill makes the distinction between the different PIC methods by calling those where only mass and position are carried with the particles as “classical” PIC and those where mass, position, velocity, and energy are carried with the particles as “full particle” PIC. This notation will be used here to distinguish between the two methods.

The idea of carrying material properties with particles comes from the plasma simulation community [5]. In plasma physics, the simulation of individual particles is next to impossible given the density of the plasma particles, $10^{18} cm^{-3}$ for laboratory plasma and $10^{18} km^{-3}$ for space plasmas [12]. In numerical simulations, a large number of plasma particles are represented by a single particle. Each particle is then given the properties of charge, velocity, and position. The background grid does not carry any information and is used only for numerical calculations [5].

Such methods from the plasma simulation community led to the “full particle” PIC codes for fluids. GAP (grid and particle) is one such method [25] that was developed in the mid-1970s that used the ideas of placing material properties with the particles instead of at the cell. A prominent “full particle” PIC method is FLIP (Fluid-Implicit-Particle) [7]. FLIP is an answer to the numerical diffusion problems that are associated with the “classical” PIC methods. FLIP has been used in simulations ranging from the animation of sand for computer graphics [37] to magnetohydrodynamic flow [6]. FLIP is the predecessor to the Material Point Method [32].

A couple of the key points of “full particle” methods will be covered here along with some terminology that will be used throughout this thesis. The term node will be used to indicate the point of intersection at cell corners. Basis functions are the centerpiece of “full particle” methods. The piecewise linear basis function has already been introduced, but basis functions do not need to be linear; quadratic and higher order functions are often used [7, 26, 4, 2]. For the purposes of this thesis, the piecewise linear basis function will be used. The gradient of the basis function is also a key component to particle methods [8, 32]. The gradient of the piecewise linear basis function is,

$$\nabla\phi(x, x_i) = \begin{cases} -\frac{1}{h} & \text{if } x_i - h \leq x \leq x_i \\ \frac{1}{h} & \text{if } x_i < x \leq x_i + h, \end{cases} \tag{2.16}$$

and $\nabla\phi_{ip} = \nabla\phi(x_p, x_i)$.

One of the first steps in “full particle” methods is to map the velocity and mass to the nodes as follows [7]:

$$m_i = \sum_p \phi_{ip} m_p, \tag{2.17}$$

$$u_i = \sum_p \frac{\phi_{ip} m_p u_p}{m_i}, \tag{2.18}$$

where m_p and m_i represent mass at the particle and node and u_p and u_i represent the velocity at particle and node, respectively. This mapping is both mass and momentum conserving [7],

$$\sum_i m_i = \sum_p m_p, \tag{2.19}$$

$$\sum_i m_i u_i = \sum_p m_p u_p. \tag{2.20}$$

In any full particle method, a gradient will be calculated. There are multiple ways of approaching this. For example, we define functions for the velocity and velocity gradient using the values at the nodes as follows,

$$u(x) = \sum_i \phi_i(x) u_i \tag{2.21}$$

$$\nabla u(x) = \sum_i \nabla \phi_i(x) u_i. \tag{2.22}$$

Because the values, u_i, are constants at the nodes, the gradient of the function $u(x)$ ends up being a sum over the gradients of the linear interpolation functions [8]. This is not the only method for computing gradients. The next chapter will explore other methods for computing gradients.

2.3 MPM

The Material Point Method (MPM) is an extension of FLIP that was originally developed to handle elastic bodies that are in contact with a fluid [32]. Like other “full particle” methods, properties such as mass, position, and velocity are stored with the particle. For elastic bodies, strain is another material property that is carried with the particle [32].

The governing equation for elastic materials is defined by the Cauchy momentum equation,

$$\rho \frac{Dv}{Dt} = \nabla \cdot \sigma + b \tag{2.23}$$

where v is velocity, σ is stress, and b is the body force. In the one-dimensional form, this becomes,

$$\rho \frac{Dv}{Dt} = \frac{\partial \sigma}{\partial x} + b. \tag{2.24}$$

As in a finite element method, one of the first steps in the derivation of MPM is to put the governing equation into its weak formulation [32],

$$\int_{\Omega} w\rho \frac{Dv}{Dt}\, d\Omega = \int_{\Omega} w\frac{\partial \sigma}{\partial x}\, d\Omega + \int_{\Omega} wb\, d\Omega, \tag{2.25}$$

$$= -\int_{\Omega} \frac{\partial w}{\partial x}\sigma\, d\Omega + \int_{\Omega} wb\, d\Omega + w\sigma|_{\partial\Omega}, \tag{2.26}$$

where w is the test function.

In the original MPM, the density function is defined at the particles as [32],

$$\rho(x) = \sum_p m_p \delta(x - x_p), \tag{2.27}$$

where the Dirac delta function is used as the particle basis function. Letting w be the standard piecewise linear basis function, $w_i = \phi$, and $\frac{Dv_j}{Dt} = a_j$, the left-hand side of the weak formulation about any node i is given by,

$$\int_{\Omega_i} \phi_i \rho a\, d\Omega_i \approx \int_{\Omega_i} \sum_j \phi_j a_j \phi_i \rho d\Omega_i, \tag{2.28}$$

$$\approx \int_{\Omega_i} \sum_j a_j \sum_p \phi_i \phi_j \delta(x - x_p) m_p d\Omega_i, \tag{2.29}$$

$$= \sum_j a_j \sum_p \phi_{ip}\phi_{jp} m_p. \tag{2.30}$$

The set of terms, $\sum_p \phi_{ip}\phi_{jp}m_p$, form what is called the consistent mass matrix, $\mathrm{M}_{i,j}$ [33]. Mass lumping can be applied [31] to get the following,

$$a_i m_i = a_i \sum_p \phi_{ip} m_p. \tag{2.31}$$

The first term on the right-hand side of Equation 2.26 is the internal force. Let $\sigma(x)$ be defined as

$$\sigma(x) = \sum_p \delta(x - x_p)\sigma_p, \tag{2.32}$$

then the internal force at node i is defined as follows,

$$f_i = -\int_{\Omega_i} \nabla\phi_i \sigma \tag{2.33}$$

$$\approx -\int_{\Omega_i} \nabla\phi_i \sum_p \delta(x - x_p)\sigma_p, \tag{2.34}$$

$$= -\sum_p \sigma_p \int_{\Omega_i} \nabla\phi_i \delta(x - x_p), \tag{2.35}$$

$$= -\sum_p \nabla\phi_{ip}\sigma_p, \tag{2.36}$$

$$\tag{2.37}$$

where σ_p is the assigned particle stress. The second and third terms together constitute the external force and for simplicity will be combined into one term, $b_i = \int_\Omega wb + w\sigma|_{\partial\Omega}$.

As stated above, in MPM, the particle basis function is the Dirac delta function. The Generalized Interpolation Material Point Method (GIMP) extends MPM by generalizing the particle basis function [2]. In GIMP, the particle basis function is defined by $\chi_p(x)$. GIMP uses the particle basis function defined as [2],

$$\chi_p(x) = \begin{cases} 1 & \text{if } x \in \Omega_p, \\ 0 & \text{otherwise,} \end{cases} \tag{2.38}$$

as one possible choice, where Ω_p is the domain of the particle. The new mapping function $\overline{\phi}_{ip}$ is defined as,

$$\overline{\phi}_{ip} = \frac{1}{V_p} \int_{\Omega_p} \chi_p(x)\phi_i(x)\, d\Omega_p, \tag{2.39}$$

where V_p is the particle volume. Note that if $\chi_p(x) = \delta(x - x_p)V_p$ is used as the particle basis function, then we have the original MPM method [2]. For the rest of this thesis, the Dirac delta function will be used as the particle basis function.

With the above formulations, the steps to the MPM method can now be laid out as follows: Firstly, particle masses and velocities are mapped to the nodes,

$$m_i = \sum_p \phi_{ip} m_p, \tag{2.40}$$

$$v_i^n = \frac{\sum_p \phi_{ip} v_p^n m_p}{m_i}, \tag{2.41}$$

where n is the nth time step. Secondly, internal forces at the nodes are calculated,

$$f_i = -\sum_p \nabla\phi_{ip}\sigma_p. \tag{2.42}$$

Thirdly, the accelerations at the nodes are calculated,

$$a_i = \frac{f_i + b_i}{m_i}. \tag{2.43}$$

Fourthly, the velocities at the nodes are updated,

$$v_i^{n+1} = v_i^n + a_i dt. \tag{2.44}$$

Steps five and six update the velocity and displacement of the particles,

$$v_p^{n+1} = v_p^n + dt \sum_i \phi_{ip} a_i, \tag{2.45}$$

$$u_p^{n+1} = u_p^n + dt \sum_i \phi_{ip} v_i^{n+1}. \tag{2.46}$$

Step seven is to calculate the velocity gradients at the particles,

$$v_{xp} = \sum_i \nabla \phi_{ip} v_i^{n+1}, \tag{2.47}$$

where v_{xp} is the gradient of v for particle p. The final step of the MPM method is to update stress. This step will depend on the stress model. For this thesis, a linear elastic model will be used.

CHAPTER 3

COMPUTING GRADIENTS

One of the key components in particle methods is the means of computing gradients and transferring their computed gradient values to particles. This chapter explores the different means by which gradients are calculated. It should be noted that in each of the methods, there may be other steps that follow after the gradient has been calculated at the nodes and before updated values are interpolated to the particles. For example, accelerations at the nodes may be calculated by taking the gradient of stress or pressure. The calculated acceleration may then be used to calculate an updated velocity at a node that is then interpolated to the particle to update a position. For the purposes of this chapter, interpolation of the computed values to the particle will directly follow the gradient calculation. The purpose here is to focus the attention on how gradients are calculated and the possible problems that may arise with these methods for doing so.

3.1 The Test Problems

In order to explore the different means of computing gradients, two example functions on which we can test the different methods will be used. They are,

$$f(x) = \sin 2\pi x \tag{3.1}$$

$$g(x) = e^{-60(x-0.5)^2}. \tag{3.2}$$

These two functions work well for demonstration purposes because, firstly, they are periodic or can be made to be periodic over the domain, $[0, 1]$, which allows us to focus on the computational methods and to not worry about boundary conditions. Secondly, there exists an analytic solution for the gradient in each case,

$$\frac{\partial f(x)}{\partial x} = 2\pi \cos 2\pi x \tag{3.3}$$

$$\frac{\partial g(x)}{\partial x} = -120(x-5)e^{-60(x-0.5)^2}. \tag{3.4}$$

Figures 3.1 and 3.2 show the plots for both the above-mentioned functions and their computed gradients. A particle distribution of two particles evenly distributed between each pair of nodes will be used for the calculation in this chapter.

3.2 Mapping to the Nodes

In most particle methods, one of the first steps that needs to be taken is to map material values from the particles to the nodes. Mass is often the first quantity to be mapped because the nodal mass is used in computing other nodal values [8, 32, 2]. If we let m_p be the particle mass, then we get the following equation for the mass, m_i, at the nodes,

$$m_i = \sum_p m_p \phi_{ip}, \tag{3.5}$$

where ϕ_{ip} is the linear basis function that was defined in the previous chapter, section 2.13. If the value of m_p is set to one, then Equation 3.5 simplifies to,

$$m_i = \sum_p \phi_{ip}. \tag{3.6}$$

The mapping of values g_p, where $g_p = g(x_p)$, to the nodes can be written as,

$$\begin{aligned} g_i &= \frac{\sum_p m_p g_p \phi_{ip}}{m_i}, & (3.7)\\ &= \frac{\sum_p g_p \phi_{ip}}{\sum_p \phi_{ip}}, & (3.8)\\ &= \sum_p S_{ip} g_p. & (3.9)\end{aligned}$$

This mapping from particles to node can also be expressed in terms of a system-wide matrix

$$\mathbf{g}_i = \mathbf{S}_{ip}\mathbf{g}_p, \tag{3.10}$$

where $\mathbf{g}_i$ contains the mapped values at the nodes and $\mathbf{g}_p$ are the values at the particles. $\mathbf{S}_{ip}$ is the mapping matrix defined by,

$$\mathbf{S}_{ip} = \begin{bmatrix} S_{1,1} & S_{1,2} & 0 & 0 & 0 & 0 & 0 & S_{1,n-1} & S_{1,n} \\ S_{2,1} & S_{2,2} & S_{2,3} & S_{2,4} & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & S_{m-1,n-3} & S_{m-1,n-2} & S_{m-1,n-1} & S_{m-1,n} \\ S_{m,1} & S_{m,2} & 0 & 0 & 0 & 0 & 0 & S_{m,n-1} & S_{m,n} \end{bmatrix}. \tag{3.11}$$

Figure 3.3 and Figure 3.4 shows the mapping of particle values to the nodes.

For the purposes of this chapter, piecewise linear basis function continue to be used,

$$\phi_i(x) = \begin{cases} 1 - \frac{x-x_i}{h} & \text{if } x_i - h \le x < x_i \\ 1 + \frac{x-x_i}{h} & \text{if } x_i \le x < x_i + h, \end{cases} \tag{3.12}$$

but other higher order basis functions could be used.

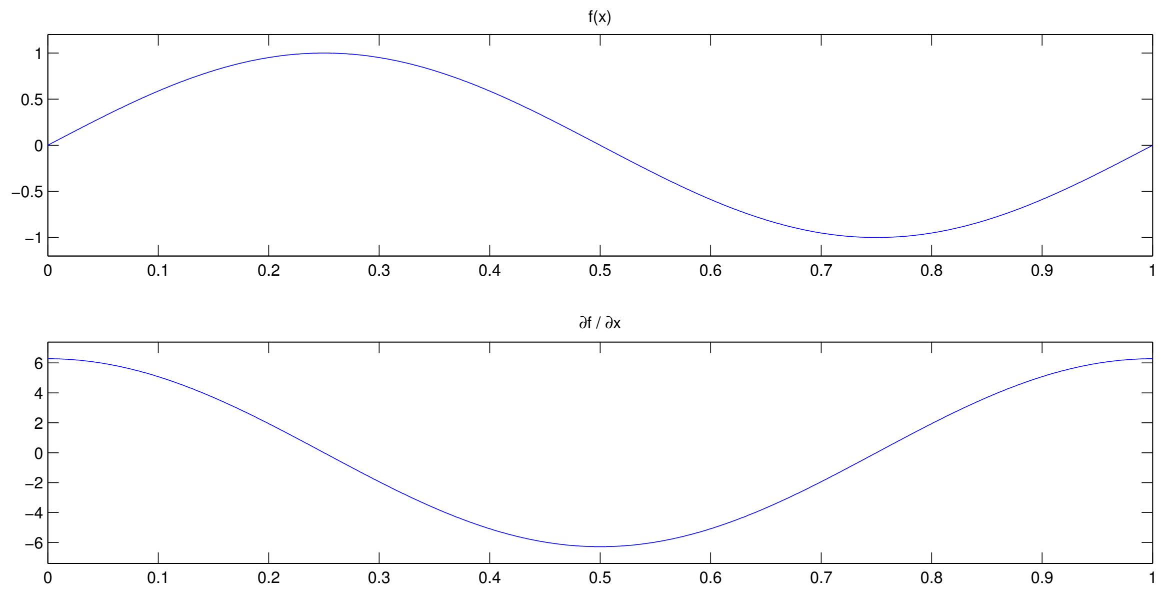

Figure 3.1. $f(x) = \sin 2\pi x$ and $\frac{\partial f(x)}{\partial x} = 2\pi \cos 2\pi x$

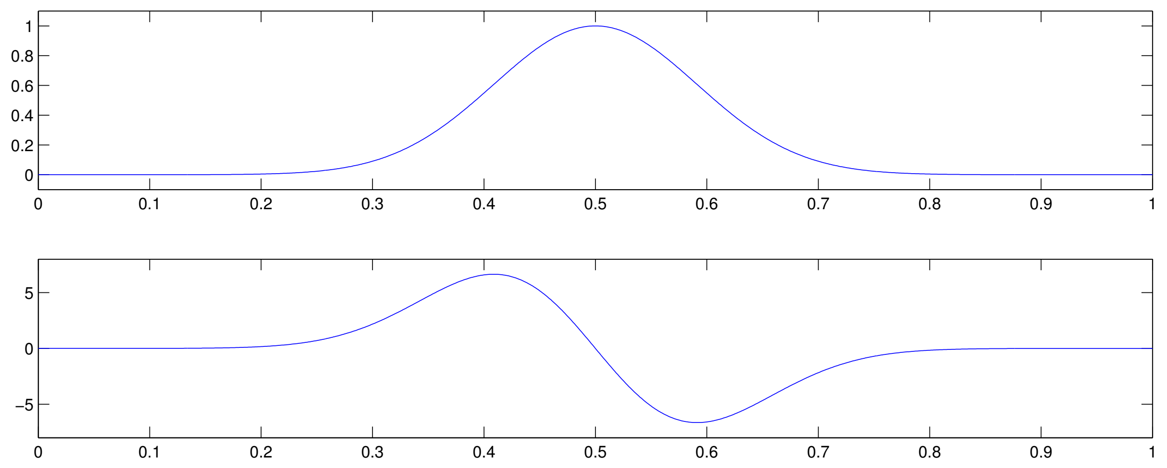

Figure 3.2. $g(x) = e^{-60(x-0.5)^2}$ and $\frac{\partial g(x)}{\partial x} = -120(x-5)e^{-60(x-0.5)^2}$

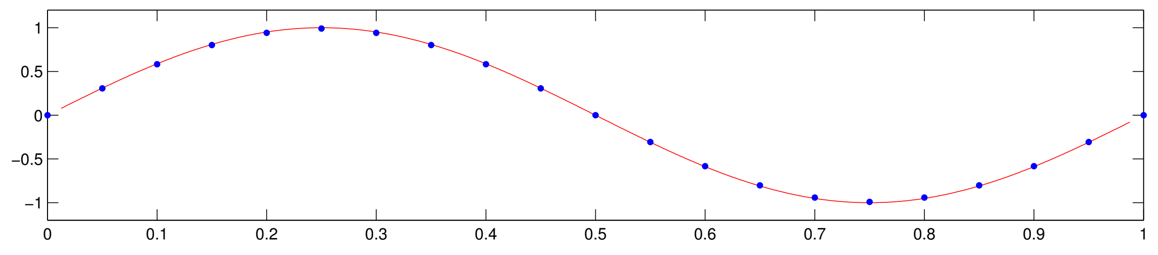

Figure 3.3. $f(x)$ mapped to the nodes

3.3 Classical PIC

As noted above, PIC is different from other particle methods in that material properties are stored at the cell level, and only mass and position are carried by the particles. In order to compute the gradient, a centered finite difference scheme at the cell centers is used,

$$\frac{\partial f_i}{\partial t} = \frac{f_{i+1} - f_{i-1}}{2h}. \tag{3.13}$$

After the gradient is calculated at the cell center, linear interpolation is used to calculate the values at the particles.

$$f_p = \phi_i f_i + \phi_{i+1} f_{i+1}, \quad i < p \leq i+1 \tag{3.14}$$

Figures 3.5 and 3.6 show the plots for the computed gradients.

3.4 Full Particle PIC Method 1

In “full particle” PIC methods, material properties are carried with the particles and consequently, there are multiple ways of computing the gradients. The first of the methods has three steps. The first step is to map the material properties to the nodes,

$$g_i = \frac{\sum_p g_p \phi_{ip}}{\sum_p \phi_{ip}}. \tag{3.15}$$

The second step is to compute the gradients at the nodes. There are a couple of ways to do this. For the purposes of this section, a centered difference finite difference method is used,

$$dg_i = \frac{g_{i+1} - g_{i-1}}{2h}. \tag{3.16}$$

The last step is to map the gradient values at the nodes back to the particles using interpolation.

$$dg_p = \sum_i dg_i \phi_{ip}. \tag{3.17}$$

3.5 Full Particle PIC Method 2

Another approach to “full particle” methods follows the same first step as Method 1 by mapping particle values to the nodes. The second step differs in that instead of calculating gradients at the nodes and then mapping those values back to the nodes, the gradient is calculated by taking the gradient of the interpolating function. Using the piecewise linear

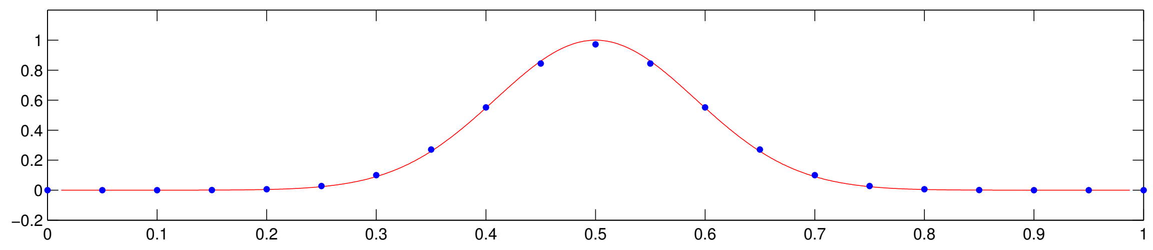

Figure 3.4. $g(x)$ mapped to the nodes

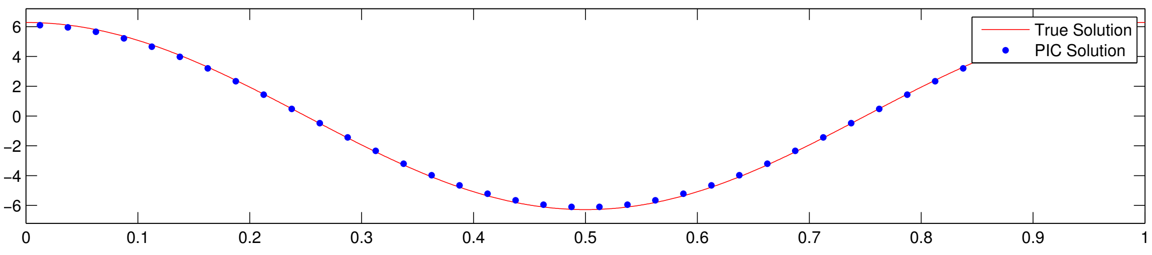

Figure 3.5. Computed values for $\frac{\partial f(x)}{\partial x}$

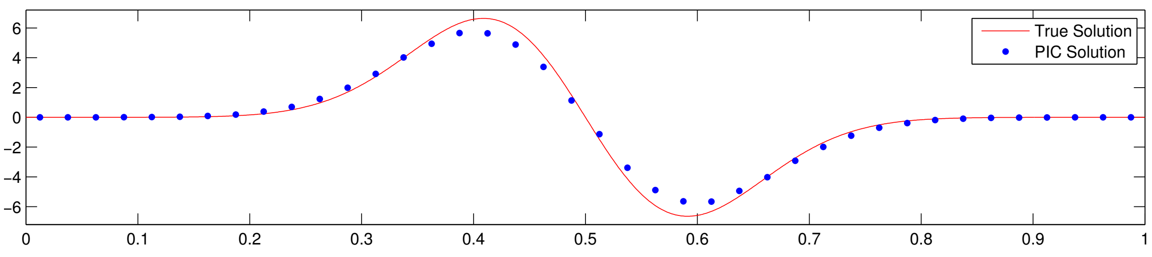

Figure 3.6. Computed values for $\frac{\partial g(x)}{\partial x}$

basis functions, $\phi_i(x)$, the data values at the nodes the function $g(x)$ can be defined as follows,

$$g(x) = \sum_i g_i \phi_i(x). \tag{3.18}$$

The gradient of the function $g(x)$ can be defined as,

$$\nabla g(x) = \sum_i g_i \nabla \phi_i(x). \tag{3.19}$$

For a piecewise linear function, the gradient of $\phi_i(x)$ is defined as,

$$\nabla \phi_i(x) = \begin{cases} -\frac{1}{h} & \text{if } x_{i-1} \leq x < x_i \\ \frac{1}{h} & \text{if } x_i \leq x < x_{i+h}. \end{cases} \tag{3.20}$$

When using a piecewise linear basis function, the gradient is piecewise constant across each interval. Figure 3.7 and Figure 3.8 show the comparison between the computed gradients at the particles and the true solution. As can be seen by Figures 3.7 and 3.8, the resolution at which we can compute gradients is no smaller then the nodal spacing. However, the particle spacing is at a finer resolution still than that of nodal spacing.

3.5.1 Ringing Instability

This mismatch between the mesh and particle resolutions is responsible for the aliasing error known as the "ringing instability" [5]. Aliasing occurs when data of higher frequency is indistinguishable from sampled data of a lower frequency. In Figure 3.9, it can be seen that the the higher frequency data in red and the lower frequency data in blue are indistinguishable at the sample points in black. In particle methods, this aliasing happens when data at a higher degree of freedom at the particles are mapped to a lower degree of freedom at the nodes and then mapped back to the higher degree of freedom particles. Brackbill explained it this way [5],

> Since all modulations of the particle density which have the same amplitudes at the grid points will produce the same interactions, two different modulations of the particle density with wavelengths that differ only by harmonics of the the grid wave number, $k_g = \frac{\pi}{\Delta x}$, are indistinguishable on the grid. They are called aliases. The aliases introduce resonances in the dispersion relation, which may cause instability through a nonlinear interaction.

Figure 3.10 shows how this occurs when gradients are calculated using Method 2. When we compute the gradients at each particle, the "ringing instability" is introduced into the nullspace of the particle to node mapping matrix [3, 22]. To gain a better understanding of how the "ringing instability" is introduced into a calculation and how the nullspace and "ringing instability" are connected, we start by looking at the how particle data are mapped to the nodes.

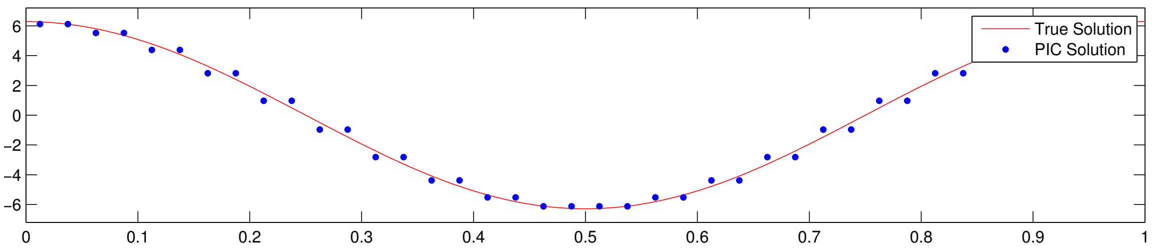

Figure 3.7. Computed versus true solution for $\frac{\partial f(x)}{\partial x}$

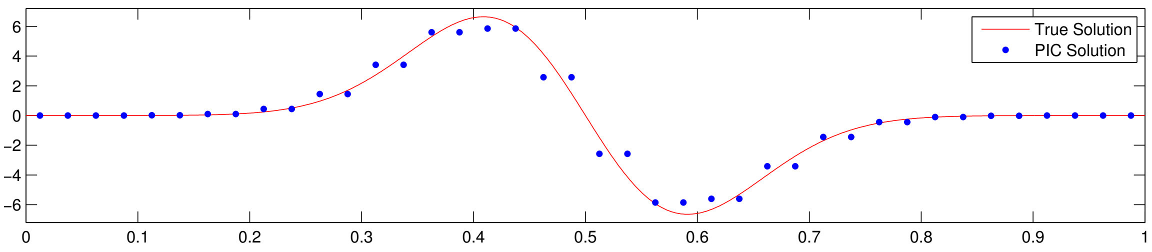

Figure 3.8. Computed versus true solution for $\frac{\partial g(x)}{\partial x}$

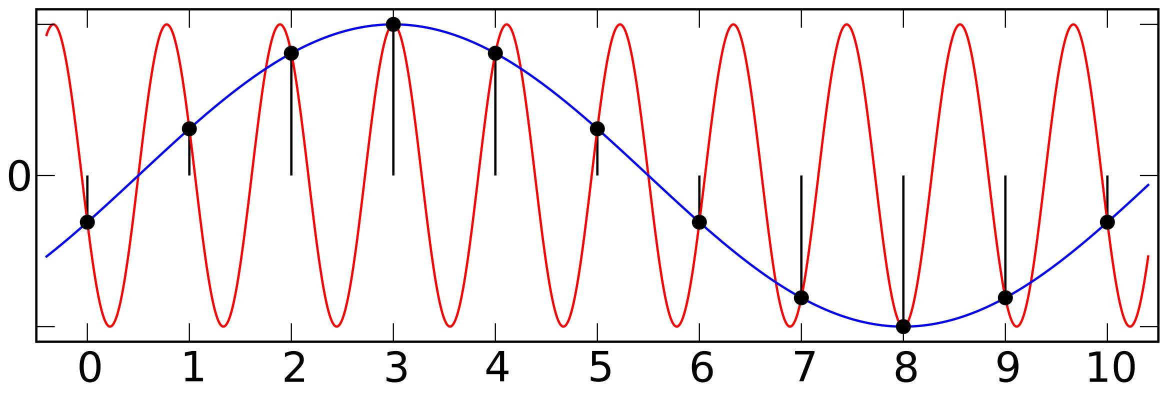

Figure 3.9. Aliasing [27]

3.5.2 The Mapping Matrix $\mathbf{S}_{ip}$

Using Equation 3.10, we can define the mapping from particles to nodes as a matrix $\mathbf{S}_{ip}$. If there are m nodes and n particles, then the matrix $\mathbf{S}_{ip}$ is m by n with $m < n$. The matrix $\mathbf{S}_{ip}$ is rectangular and it has a nontrivial nullspace. For example, let $\mathbf{a}$ be a vector in $\mathbb{R}^n$. We can decompose this vector into two vectors such that $\mathbf{a} = \mathbf{b} + \mathbf{c}$. Applying the matrix $\mathbf{S}_{ip}$ to $\mathbf{a}$ we get,

$$\mathbf{S}_{ip}\mathbf{a} = \mathbf{S}_{ip}\mathbf{b} + \mathbf{S}_{ip}\mathbf{c}. \tag{3.21}$$

If $\mathbf{S}_{ip}\mathbf{c} = \mathbf{0}$, then we say that $\mathbf{c}$ is in the nullspace of $\mathbf{S}_{ip}$. In this next section, it will be shown how we can decompose a vector in $\mathbb{R}^n$ into a nullspace and non-nullspace component.

3.5.3 Defining the Nullspace

We can define the nullspace of $\mathbf{S}_{ip}$ by making use of its singular value decomposition, SVD [22]. Taking the SVD of $\mathbf{S}_{ip}$ gives the following decomposition,

$$\mathbf{S}_{ip} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T, \tag{3.22}$$

where $\mathbf{U}$ has dimension m by m, $\mathbf{\Sigma}$ is m by n, and $\mathbf{V}$ is n by n. The matrices $\mathbf{U}$ and $\mathbf{V}$ are unitary, meaning that the columns are orthonormal [35]. In other words, if $\mathbf{u}_i$ and $\mathbf{u}_j$ are columns of the matrix U, then,

$$\mathbf{u}_i^T\mathbf{u}_j = 0 \tag{3.23}$$

and

$$\mathbf{u}_i^T\mathbf{u}_i = 1, \tag{3.24}$$

where the superscript T is the transpose of the vector.

Given that the columns of $\mathbf{U}$ and $\mathbf{V}$ are orthogonal, they are linearly independent and therefore span the entire spaces $\mathbb{R}^m$ and $\mathbb{R}^n$, respectively [35]. This means, for example, that any vector $\mathbf{a} \in \mathbb{R}^m$ can be expressed as a linear combination of the columns of $\mathbf{U}$.

$$\mathbf{a} = c_1 \begin{bmatrix} \\ u_1 \\ \\ \end{bmatrix} + \cdots + c_m \begin{bmatrix} \\ u_m \\ \\ \end{bmatrix} \tag{3.25}$$

where $c_1, \ldots, c_m$ are constants.

The matrix $\mathbf{\Sigma}$ is an m by n diagonal matrix of the form

$$\mathbf{\Sigma} = \begin{bmatrix} \sigma_1 & 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & 0 \\ 0 & 0 & \dots & \sigma_r & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \dots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 \end{bmatrix} \tag{3.26}$$

where the columns 1 to r contain the nonzero singular values $\sigma_1, \dots, \sigma_r$ and the columns $r+1$ to m are columns of zeros.

When we take the matrix product of $\mathbf{\Sigma}$ and $\mathbf{V}^T$, we get the following,

$$\mathbf{\Sigma V}^T = \left(\mathbf{V\Sigma}^T\right)^T = (\mathbf{V\Sigma})^T = \left[\begin{array}{c|c|c|c|c|c} \\ \sigma_1\mathbf{v}_1 & \dots & \sigma_r\mathbf{v}_r & 0 * \mathbf{v}_{r+1} & \dots & 0 * \mathbf{v}_m \\ \\ \end{array}\right]^T. \tag{3.27}$$

From this, we can see that the column vectors $\mathbf{v}_{r+1}$ to $\mathbf{v}_m$ span the nullspace of $\mathbf{\Sigma}$, which in turn means that they span the nullspace of $\mathbf{S_{ip}}$.

As mentioned earlier, in Equation 3.25, any vector can be decomposed into its orthogonal components [35]. Since the columns of $\mathbf{V}$ are orthogonal, they form a basis for $\mathbb{R}^n$, which means that any vector $b \in \mathbb{R}^m$ can be expressed as a linear combination of the columns of $\mathbf{V}$,

$$\mathbf{b} = c_1 \begin{bmatrix} \\ \mathbf{v}_1 \\ \\ \end{bmatrix} + \dots + c_r \begin{bmatrix} \\ \mathbf{v}_r \\ \\ \end{bmatrix} + \underbrace{c_{r+1} \begin{bmatrix} \\ \mathbf{v}_{r+1} \\ \\ \end{bmatrix} + \dots + c_m \begin{bmatrix} \\ \mathbf{v}_m \\ \\ \end{bmatrix}}_{null(S_{ip})}. \tag{3.28}$$

From this it can be seen that a portion of $\mathbf{b}$ is in the nullspace of $\mathbf{S}_{ip}$.

3.5.4 Finding the Residual Vector

Now that we have a basis for the nullspace of $\mathbf{S}_{ip}$, we can find the components of the particle vectors $\mathbf{u}_p$ and $\boldsymbol{\rho}_p$ that are in the nullspace. In order to do this, we use the inner product [35]. The geometric definition of the inner product of two vectors is,

$$\mathbf{x}^T\mathbf{y} = \|\mathbf{x}\|\|\mathbf{y}\|cos\theta. \tag{3.29}$$

If we substitute $\mathbf{u}_p$ and the i^{th} column vector, $\mathbf{v}_i$, of $\mathbf{V}$ into the equation, we get,

$$\mathbf{u}_p^T\mathbf{v}_i = \|\mathbf{u}_p\|\|\mathbf{v}_i\|cos\theta. \tag{3.30}$$

$$\mathbf{u}_p^T \mathbf{v}_i = \|\mathbf{u}_p\| cos\theta, \tag{3.31}$$

which is the length of orthogonal projection of $\mathbf{u}_p$ onto $\mathbf{v}_i$, or the amount that $\mathbf{u}_p$ goes in the direction of $\mathbf{v}_i$. If we define the vector $\mathbf{r}_i$ as,

$$\mathbf{r}_i = \mathbf{u}_p - (\mathbf{v}_i^T \mathbf{u}_p)\mathbf{v}_i, \tag{3.32}$$

then what is left is a vector, $\mathbf{r}_i$, that has no component in the direction of $\mathbf{v}_i$. The vector $\mathbf{u}_p$ can now be expressed as linear combination of two vectors,

$$\mathbf{u}_p = (\mathbf{v}_i^T \mathbf{u}_p) \begin{bmatrix} \mathbf{v}_i \end{bmatrix} + \begin{bmatrix} \mathbf{r}_i \end{bmatrix}. \tag{3.33}$$

If we repeat this process using the column vectors 1 to r of V, and define the vector $\mathbf{r}$ by,

$$\mathbf{r} = \mathbf{u}_p - \sum_{i=1}^{r} (\mathbf{v}_i^T \mathbf{u}_p)\mathbf{v}_i, \tag{3.34}$$

then the vector, $\mathbf{r}$, is the components of $\mathbf{u}_p$ that lie entirely in the nullspace of $\mathbf{S}_{ip}$.

3.5.5 The getRes() Function

We have described the steps for finding what portion of a vector lies in the nullspace of $\mathbf{S}_{ip}$. Here is an enumeration of the steps that are to be taken to find the residual vector. We can place these steps inside a function called *getRes*() that takes as its parameters an n dimensional vector $\mathbf{a}$ and the matrix $\mathbf{S}_{ip}$. We will also use a function called *svd*(), which decomposes a matrix into $\mathbf{U}$, $\mathbf{\Sigma}$, and $\mathbf{V}$ and a function called *rank*(), which returns the number of singular values in a matrix. Using a ®Matlab style of syntax, here is the function.

```
function getRes(a, Sip)
    U, S, V = svd(Sip)    \\ S = singular values
    k = rank(S)
    for i = 1 to k
        r = a - (a' * V(:,i))*V(:,i)
    end
    return r
```

3.5.6 Removing the Nullspace

Equation 3.18 maps particle values to the nodes. In matrix form, this is,

$$\mathbf{g}_i = \mathbf{S}_{ip}\mathbf{g}_p. \tag{3.35}$$

At this point, the nullspace component of $\mathbf{g}_p$ has been removed by the nature of the mapping. It is at the next step in the computation, Equation 3.19, that a nullspace component can be re-introduced. If we express the computed gradients at the particles, ∇g_p, in the vector form $\mathbf{dg}_p$ then we can decompose the vector into a nullspace and a non-nullspace component. Figure 3.11 and Figure 3.12 show what the nullspace components look like.

Now that we have the nullspace component, we can remove it from the computed gradient,

$$\mathbf{dgn} = \mathbf{dg} - \mathbf{rdg}, \tag{3.36}$$

to get a smoothed version of $\mathbf{dg}$. Figure 3.13 and Figure 3.14 show the results of $\mathbf{df}$ and $\mathbf{dg}$ with the nullspace component removed.

3.5.7 Nodalwise Noise Removal

Using singular value decomposition for the removal of nullspace noise works well for small one-dimensional problems, but it does not scale well when running a multidimensional simulation across multiple cores. A second issue is that the computational complexity of generating the matrix $\mathbf{V}$ with a singular value decomposition is $O(m^2n + n^3)$ [15]. In order to have a method for removing the nullspace noise that scales well across hundreds of cores, we need a new approach that works locally across just a few nodes and not the entire set of nodes.

3.5.7.1 Local Method

This method takes a different approach then the SVD method to removing the nullspace components. The key idea in the local method is to use the already mentioned fact that any vector, $\mathbf{a} \in \mathbb{R}^n$, can be decomposed into a nullspace and non-nullspace component, $\mathbf{a} = \mathbf{b} + \mathbf{c}$. When the matrix $\mathbf{S}_{ip}$ is applied to the vector $\mathbf{a}$, the nullspace portion of $\mathbf{a}$ is removed,

$$\mathbf{S}_{ip}\mathbf{a} = \mathbf{S}_{ip}\mathbf{b} + \mathbf{S}_{ip}\mathbf{c} \tag{3.37}$$

$$= \mathbf{S}_{ip}\mathbf{b}, \tag{3.38}$$

where $\mathbf{c}$ is the nullspace component, $\mathbf{S}_{ip}\mathbf{c} = \mathbf{0}$, of the vector $\mathbf{a}$.

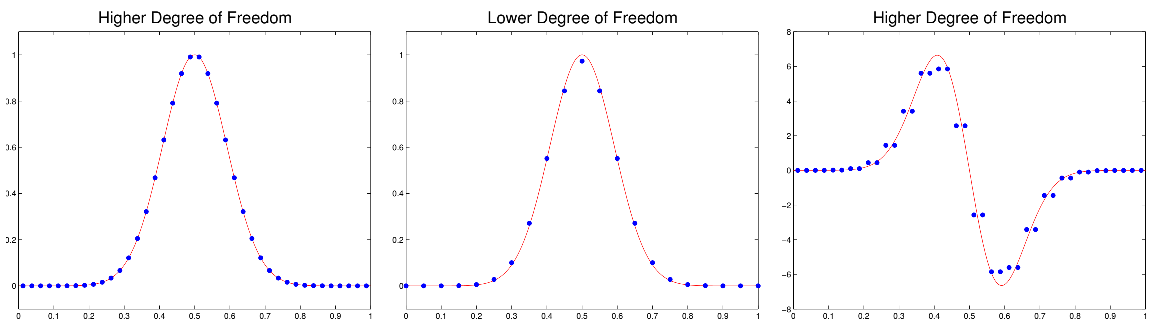

Figure 3.10. How ringing instability is introduced

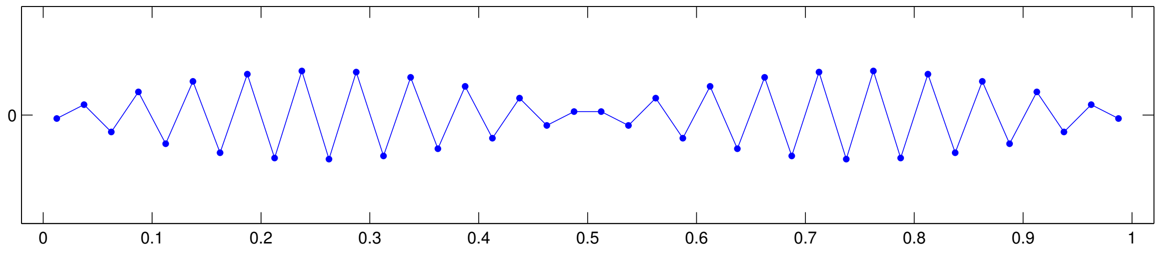

Figure 3.11. Nullspace of computed $\frac{\partial f(x)}{\partial x}$

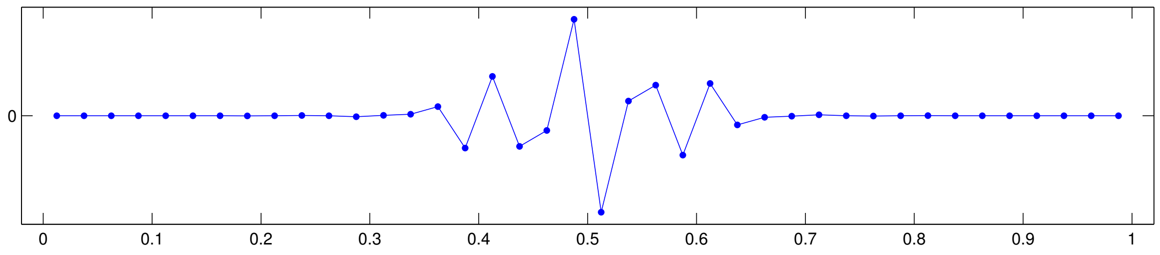

Figure 3.12. Nullspace of computed $\frac{\partial g(x)}{\partial x}$

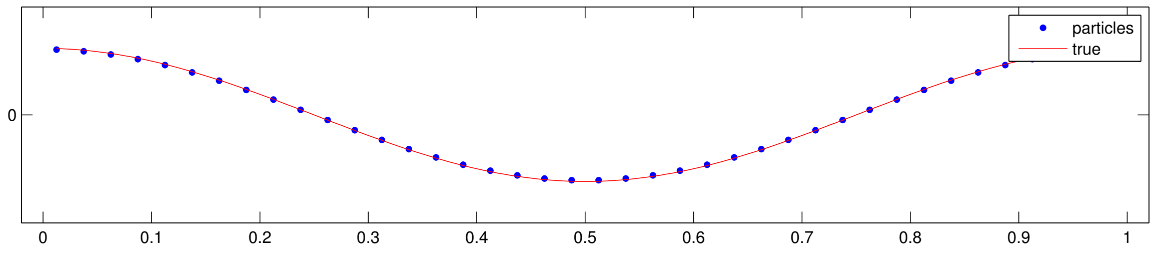

Figure 3.13. The results of $\frac{\partial f(x)}{\partial x}$ with the nullspace component removed

For Method 2, the gradient at the particles is computed by first mapping particle values to the nodes,

$$g_i = \frac{\sum_p \phi_{ip} g_p}{\sum_p \phi_{ip}} \tag{3.39}$$

and then computing the gradients at the particles by using the gradient of the the interpolating function that interpolates values from nodes to particles,

$$\nabla g_p = \sum_i \nabla \phi_{ip} g_i. \tag{3.40}$$

When this happens, a nullspace component is introduced by this gradient calculation.

Taking advantage of the observation made earlier, if the newly computed gradient values are mapped back to the nodes,

$$\nabla g_i = \frac{\sum_p \phi_{ip} \nabla g_p}{\sum_p \phi_{ip}}, \tag{3.41}$$

or in matrix form,

$$\nabla \mathbf{g}_i = \mathbf{S}_{ip} \nabla \mathbf{g}_p, \tag{3.42}$$

then its nullspace component is removed. Let the vector representing calculated gradients at the particle values be decomposed into its non-nullspace, $\nabla \mathbf{b}_p$, and nullspace, $\nabla \mathbf{c}_p$, components,

$$\nabla \mathbf{g}_p = \nabla \mathbf{b}_p + \nabla \mathbf{c}_p. \tag{3.43}$$

Then mapping gradient values at the particles back to the nodes we get,

$$\nabla \mathbf{g}_i = \mathbf{S}_{ip} \nabla \mathbf{g}_p \tag{3.44}$$

$$= \mathbf{S}_{ip} \nabla \mathbf{b} + \nabla \mathbf{S}_{ip} \mathbf{c} \tag{3.45}$$

$$= \mathbf{S}_{ip} \nabla \mathbf{b}, \tag{3.46}$$

which removes the null component that was introduced by the gradient calculation. With gradient values mapped to the nodes, all that needs to be done now is to interpolate the values back to the particles,

$$\nabla g_p = \sum_i \nabla g_i \, \phi_{ip}. \tag{3.47}$$

While there is no nullspace component at the nodes, there is nothing to prevent nullspace noise from being introduced at the particle via the interpolation process. Furthermore, there is no longer a need to explicitly calculate the nullspace in order to smooth out the particle gradients. Figures 3.15 and 3.16 show a comparison between the true gradient and the calculated solution at the particles.

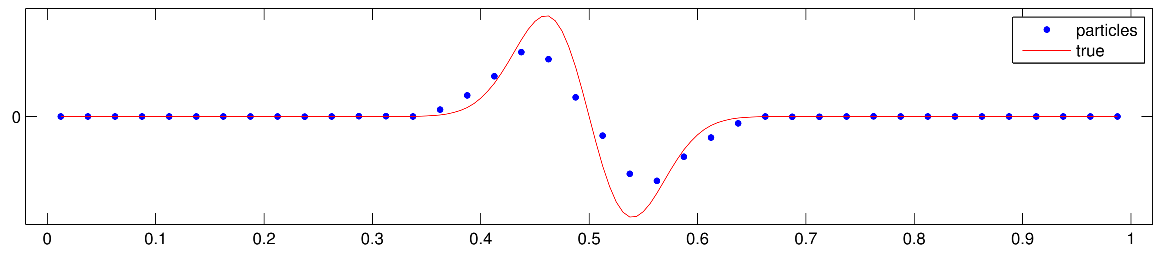

Figure 3.14. The results of $\frac{\partial g(x)}{\partial x}$ with the nullspace component removed

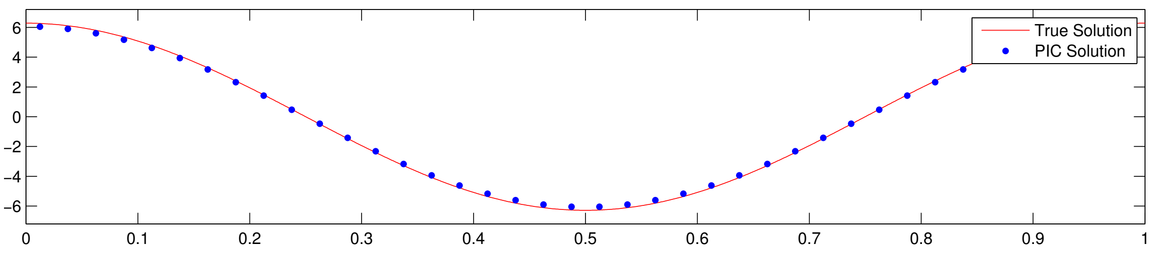

Figure 3.15. Comparison between the true solution and the computed solution using the local method as a filter for $\frac{\partial f}{\partial x}$

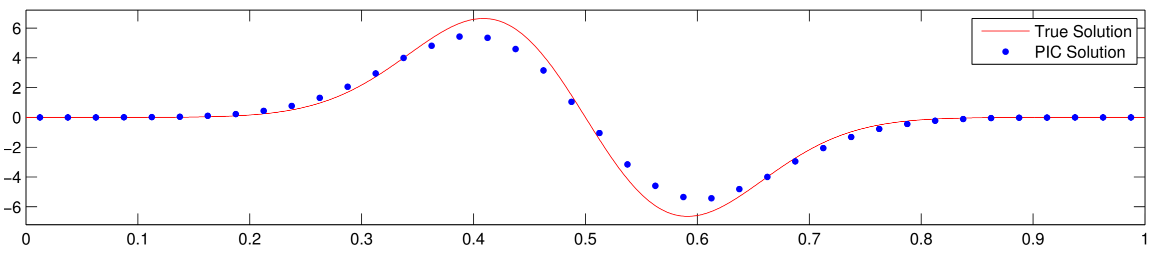

Figure 3.16. Comparison between the true solution and the computed solution using the local method as a filter for $\frac{\partial g}{\partial x}$

3.6 MPM

MPM differs from the Full Particle methods described in this section in that it computes the gradient by taking the gradient of the mapping function. The first step in MPM is to compute the mass at the nodes in the same way as the other "full particle" methods. The second step is to map to the nodes using the gradient of the mapping function and dividing by the nodal mass,

$$dg_i = \frac{\sum_p g_p \nabla \phi_{ip}}{\sum_p \phi_{ip}}. \tag{3.48}$$

The next step is to map the gradient values back to the particles by using linear interpolation,

$$dg_p = \sum_i dg_i \phi_{ip}. \tag{3.49}$$

Figure 3.17 and Figure 3.18 show a comparison between the true gradient and the calculated solution at the particles using the MPM version on the functions $f(x)$ and $g(x)$ defined by Equations 3.1 and 3.2.

3.7 Comparison of Methods

Now that we have described all the different methods for computing gradients, we can do a side-by-side comparison of the different methods. In the first test case, the gradients are computed using a uniform distribution of particles over a grid with two particles per grid. Since the true solution is known, the error in the L_2 norm can be computed.

A second test case is presented that is more representative of what happens when computing gradients using particle methods. Normally in a simulation, particles will not remain evenly distributed over the grid. To represent this, a set of particles are evenly distributed between nodes like in the first test case. Then each particle is randomly shifted a varying amount to the left or right of its original position. The shifting of particles is limited so that no particle will cross over a node or another particle. Gradients for the different methods are then calculated and an error in the L_2 norm is calculated. This process is then repeated a given number of times and then an average is calculated.

In Table 3.1, the results of the two test cases are given. For both test cases, the nodes are evenly distributed from 0 to 1 with the spacing between nodes being $h = 0.025$. The randomized test case is performed 100 times to compute an average. Calculated errors are relative.

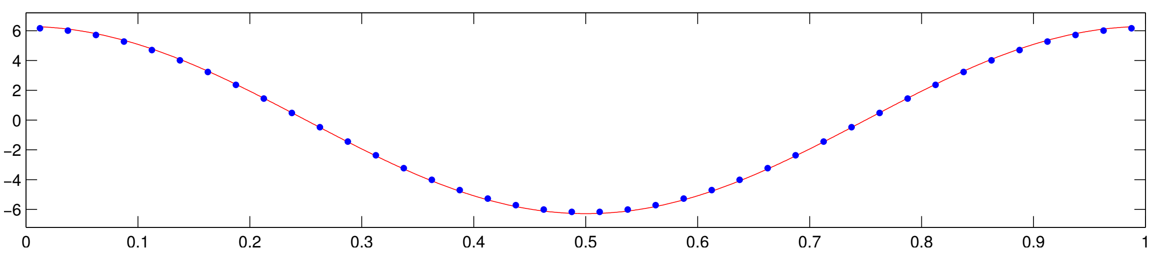

Figure 3.17. The MPM version results for computing gradients for $\frac{\partial f(x)}{\partial x}$

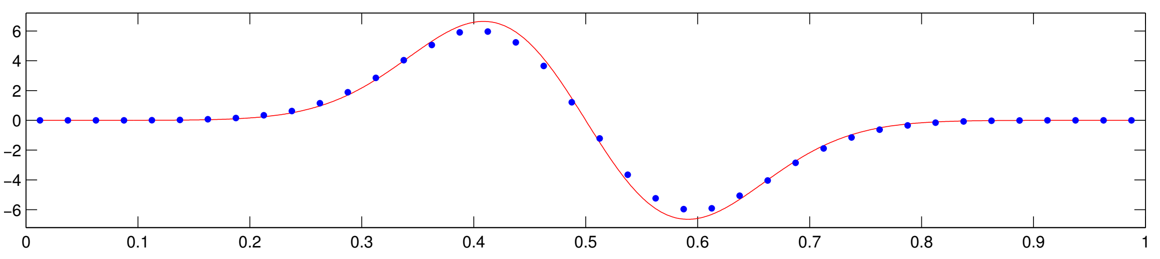

Figure 3.18. The MPM version results for computing gradients for $\frac{\partial g(x)}{\partial x}$

	$f(x)$		$g(x)$	
Method	Uniform	Randomized	Uniform	Randomized
Classical PIC:	0.006409	0.006358	0.036819	0.036681
Full Particle 1:	0.008705	0.020588	0.049420	0.053552
Full Particle 2:	0.004108	0.038973	0.023921	0.053081
MPM:	0.004105	0.096314	0.023774	0.095828

Table 3.1. Comparison of results for computed gradients

CHAPTER 4

STABILITY OF PIC METHODS

In the previous chapter, we described the different ways in which a gradient can be calculated in particle methods, but computing gradients is not the only issue when it comes to stability. While the methods for computing gradients, mapping from particles to nodes, and the methods for interpolating back to particles all contribute to the stability of the numerical code, they are not the only potential causes of instabilities. The stability of the underlying method at the nodes plays a crucial role in the overall stability of the solution. The first part of this chapter will examine the stability issues associated with the gas dynamics problem. The second half of the chapter will examine the underlying stability of the Cauchy momentum problem.

4.1 Gas Dynamics Problem

Starting with the gas dynamics problem that we have already examined, we will use the same simplifying assumptions that Brackbill makes [5]. The original gas dynamics Equations 2.1 and 2.2 are linearized in [5] to get,

$$\frac{\partial \rho_1}{\partial t} + u_0 \frac{\partial \rho_1}{\partial x} + \rho_0 \frac{\partial u_1}{\partial x} = 0 \tag{4.1}$$

$$\rho_0 \left[\frac{\partial u_1}{\partial t} + u_0 \frac{\partial u_1}{\partial x} \right] + \frac{\partial p_1}{\partial x} = 0, \tag{4.2}$$

where u_0 and ρ_0 are constants representing the constant flow and the initial uniform density, respectively, and u_1 and ρ_1 are the values subject to change. Substituting $p_1 = c^2 \rho_1$ and rewriting Equations 4.1 and 4.2 in terms of their material derivative,

$$\frac{D\rho_1}{Dt} = \frac{\partial \rho_1}{\partial t} + u_0 \frac{\partial \rho_1}{\partial x}, \tag{4.3}$$

$$\frac{Du_1}{Dt} = \frac{\partial u_1}{\partial t} + u_0 \frac{\partial u_1}{\partial x}, \tag{4.4}$$

we get the following set of coupled equations,

$$\frac{D\rho_1}{Dt} + \frac{\partial u_1}{\partial x} = 0, \tag{4.5}$$

$$\frac{Du_1}{Dt} + c^2\frac{\partial \rho_1}{\partial x} = 0. \tag{4.6}$$

Without a loss of generality, we set $\rho_0 = 1$ and note that Equations 4.5 and 4.6 can be rewritten in an Eulerian form,

$$\frac{\partial \rho_1}{\partial t} = -\frac{\partial u_1}{\partial x}, \tag{4.7}$$

$$\frac{\partial u_1}{\partial t} = -a^2\frac{\partial \rho_1}{\partial x}, \tag{4.8}$$

on a grid that is moving at velocity u_0. The coupled set of equations can then be written in the form of the wave equation on a grid moving with the velocity u_0 as,

$$\frac{\partial^2 \rho_1}{\partial t^2} = c^2\frac{\partial^2 \rho_1}{\partial x^2}. \tag{4.9}$$

If our initial conditions are $\rho_1(x, 0) = f(x)$ and $\frac{\partial \rho_1(x,0)}{\partial t} = 0$ and the boundary conditions are $-\infty < x < \infty$ and $t > 0$, then the solution to the wave equation is d'Alembert's formula [23] on a moving grid,

$$\rho_1(x, t) = \frac{1}{2}(f(x - ct) + f(x + ct)). \tag{4.10}$$

If you want to take into account the moving frame of reference, then the solution in the fixed frame is the modified d'Alembert's formula,

$$\rho_1(x, t) = \frac{1}{2}(f(x - (c - u_0)t) + f(x + (c - u_0)t)). \tag{4.11}$$

Throughout this paper, all problems will be solved on the boundary conditions $-\infty < x < \infty$ and $t > 0$. When performing numerical calculations, the spatial boundary conditions will be periodic.

4.2 A Finite Difference Approach

In order to better understand how different PIC formulations can affect the stability of a given solution, we consider the following pair of coupled equations,

$$\frac{\partial \rho}{\partial t} = c\frac{\partial u}{\partial x} \tag{4.12}$$

$$\frac{\partial u}{\partial t} = c\frac{\partial \rho}{\partial x}. \tag{4.13}$$

using finite difference methods.

At first glance, a common approach might be to discretize the equations in the following manner,

$$\rho_i^{t+1} = \rho_i^t + k\frac{1}{2}(u_{i+1}^t - u_{i-1}^t) \tag{4.14}$$

$$u_i^{t+1} = u_i^t + k\frac{1}{2}(\rho_{i+1}^t - \rho_{i-1}^t), \tag{4.15}$$

where $k = c(dt/h)$. The solution is first order accurate in time and second order accurate in space. The method can be solved explicitly. The problem is that this solution is unstable [30] for $c < 0$. By contrast, a slight variation on the above equations can lead to the following conditionally stable form [30],

$$\rho_i^{t+1} = \rho_i^t + k\frac{1}{2}(u_{i+1}^t - u_i^t) \tag{4.16}$$

$$u_i^{t+1} = u_i^t + k\frac{1}{2}(\rho_i^{t+1} - \rho_{i-1}^{t+1}). \tag{4.17}$$

Both formulations are solved explicitly, but in this case, by using the updated values for ρ in the second calculation and a slight shift in spatial differences, an unstable form can become stable. This small example using a finite difference approach will be helpful when looking at the stability of the “full particle” PIC method.

4.2.1 PIC Formulations for Compressible Flow

The stability of PIC methods is more difficult to analyze than simple finite difference methods because of the movement of information between particles and nodes.

A naive first approach to the problem might be to map the particle values for both ρ and u to the nodes, to compute the gradients of both ρ and u and update both ρ and u, and then to move the particles. A second approach might be to map only the particle values for ρ to the nodes nodes, to compute the gradient of ρ, and update the particle values of u. The updated particle values for u are then mapped to the nodes. The gradients of u are then calculated and are used to update the particle values of ρ. Finally, the particles are then moved.

Both approaches are possible solutions to the problem, but the first one is unstable while the second contains a stable nodal method; this comes as a result of using the updated u values in a similar way to that found in Equations 4.16 and 4.17. Here are the two different formulations written out in steps.

Formulation 1	Formulation 2
1. $u_i^t = \sum_{p=1}^{np} S_{ip} u_p^t$	1. $\rho_i^t = \sum_{p=1}^{np} S_{ip} \rho_p^t$
2. $\rho_i^t = \sum_{p=1}^{np} S_{ip} \rho_p^t$	2. $u_p^{t+1} = u_p^t + c\frac{dt}{h}(\rho_{i+1}^t - \rho_i^t)$
3. $u_p^{t+1} = u_p^t + c\frac{dt}{h}(\rho_{i+1}^t - \rho_i^t)$	3. $u_i^{t+1} = \sum_{p=1}^{np} S_{ip} u_p^{t+1}$
4. $\rho_p^{t+1} = \rho_p^t + c\frac{dt}{h}(u_{i+1}^t - u_i^t)$	4. $\rho_p^{t+1} = \rho_p^t + c\frac{dt}{h}(u_{i+1}^{t+1} - u_i^{t+1})$
5. $x_p^{t+1} = x_p^t + vdt$	5. $x_p^{t+1} = x_p^t + vdt$

Note that to avoid confusion in notation, the variable v is used instead the variable u_0 that is found in Equations 4.1 and 4.2 to represent the particle velocity.

4.3 Stability Analysis of PIC Formulations

The starting point for answering the stability questions about the above formulations is to look at what happens at the nodes [3].

4.3.1 Analysis of Formulation 1

The value at the node for u_i^{t+1} is given by.

$$u_i^{t+1} = \sum_{p=1}^{np} S_{ip}^{t+1} u_p^{t+1}. \tag{4.18}$$

From the first formulation, we substitute in for u_p^{t+1} [3] to get,

$$\begin{aligned} u_i^{t+1} &= \sum_{p=1}^{np} S_{ip}^{t+1} u_p^{t+1}, \\ &= \sum_{p=1}^{np} S_{ip}^{t+1} [u_p^t + c\frac{dt}{h}(\rho_{i+1}^{t+1} - \rho_i^t)], \\ &= \sum_{p=1}^{np} S_{ip}^{t+1} u_p^t + c\frac{dt}{h} \sum_{p=1}^{np} S_{ip}^{t+1} (\rho_{i+1}^t - \rho_i^t), \\ &= u_i^t + \sum_{p=1}^{np} (S_{ip}^{t+1} - S_{ip}^t) u_p^t + c\frac{dt}{h} \sum_{p=1}^{np} S_{ip}^{t+1} (\rho_{i+1}^t - \rho_i^t). \end{aligned}$$

The same steps can be followed to arrive at the values for ρ_i^{t+1}. For simplicity, we set $k = c\frac{dt}{h}$. We now have the following pair of equations representing what happens at the nodes [3].

$$u_i^{t+1} = u_i^t + \sum_{p=1}^{np}(S_{ip}^{t+1} - S_{ip}^t)u_p^t + k\sum_{p=1}^{np} S_{ip}^{t+1}(\rho_{i+1}^t - \rho_i^t), \tag{4.19}$$

$$\rho_i^{t+1} = \rho_i^t + \sum_{p=1}^{np}(S_{ip}^{t+1} - S_{ip}^t)\rho_p^t + k\sum_{p=1}^{np} S_{ip}^{t+1}(u_{i+1}^t - u_i^t). \tag{4.20}$$

Further insight can be gained by looking at the individual terms for Equations 4.19 and 4.20 [3]. If we are using piecewise linear basis functions for our kernel functions and if x_p is in between nodes $i-1$ and i, then

$$\begin{aligned} S_{ip}^{t+1} - S_{ip}^t &= w_i\left[\left(1 + \frac{x_p^{t+1} - x_i}{h})\right) - \left(1 + \frac{x_p^t - x_i}{h}\right)\right], \\ &= w_i\frac{x_p^{t+1} - x_p^t}{h}, \end{aligned}$$

where $w_i = \frac{1}{\sum_p \phi_{ip}}$. If x_p is in between nodes i and $i+1$, then we get

$$\begin{aligned} S_{ip}^{t+1} - S_{ip}^t &= w_i\left[\left(1 - \frac{x_p^{t+1} - x_i}{h})\right) - \left(1 - \frac{x_p^t - x_i}{h}\right)\right], \\ &= -w_i\frac{x_p^{t+1} - x_p^t}{h}. \end{aligned}$$

We can now use the fact that $x_p^{t+1} = x_p^t + vdt$ to write [3],

$$S_{ip}^{t+1} - S_{ip}^t = \pm w_i v\frac{dt}{h}. \tag{4.21}$$

From the description of the problem, we know that v is constant across all particles. Hence, if we start out with a uniform distribution of particles, w_i is constant for all the nodes.

$$S_{ip}^{t+1} - S_{ip}^t = \pm w_i u_0\frac{dt}{h} = \pm C_1 \tag{4.22}$$

Further insight can also be gained by looking at the third term of the equation for u_i^{t+1}. The term

$$\sum_{p=1}^{np} S_{ip}^{t+1}(\rho_{i+1}^t - \rho_i^t) \tag{4.23}$$

is a summation of all the gradients affecting the particles within the support of node i; therefore, Equation 4.23 ends up being a weighted combination of the nodal gradients across the spans $i-1$ to i and i to $i+1$. Equation 4.23 ends up being [3],

$$\sum_{p=1}^{np} S_{ip}^{t+1}(\rho_{i+1}^t - \rho_i^t) = C_2(\rho_i^t - \rho_{i-1}^t) + C_3(\rho_{i+1}^t - \rho_i^t) \tag{4.24}$$

where $C_2 + C_3 = 1$. The same steps [3] that we used for Equation 4.23 can also be applied to the third term in Equation 4.20 to get,

$$\sum_{p=1}^{np} S_{ip}^{t+1}(u_{i+1}^t - u_i^t) = C_2(u_i^t - u_{i-1}^t) + C_3(u_{i+1}^t - u_i^t). \tag{4.25}$$

Applying Equations 4.22, 4.24, and 4.25 to Equations 4.19 and 4.20, we get the following updated set of equations,

$$u_i^{t+1} = u_i^t + C_1 \sum_{p=1}^{np} \pm u_p^t + k\left(C_2(\rho_i^t - \rho_{i-1}^t) + C_3(\rho_{i+1}^t - \rho_i^t)\right), \tag{4.26}$$

$$\rho_i^{t+1} = \rho_i^t + C_1 \sum_{p=1}^{np} \pm \rho_p^t + k\left(C_2(u_i^t - u_{i-1}^t) + C_3(u_{i+1}^t - u_i^t)\right). \tag{4.27}$$

4.3.1.1 What happens if $v = 0$

Motivated by Brackbill's [5] comments that the "ringing instability" occurs in cases for which the particles move at low velocities, we consider the case when each particle velocity is zero. If $v = 0$, then $C_2 = C_3 = 0.5$ and $C_1 = 0$, and we get the following equations,

$$u_i^{t+1} = u_i^t + k(\rho_{i+1}^t - \rho_{i-1}^t), \tag{4.28}$$

$$\rho_i^{t+1} = \rho_i^t + k(u_{i+1}^t - u_{i-1}^t). \tag{4.29}$$

Using Von Neumann analysis, we can gain some insight into the stability of the underlying schemes. Let ξ and η represent the errors at the nodes for u and ρ, respectively.

$$\xi = aG^t e^{i\beta j}, \tag{4.30}$$

$$\eta = bG^t e^{i\beta j}. \tag{4.31}$$

Substituting the error terms ξ and η into Equations 4.28 and 4.29 we get an expression for the error growth at the nodes,

$$aG^{t+1}e^{i\beta j} = aG^t e^{i\beta j} + k(bG^t e^{i\beta j+1} - bG^t e^{i\beta j-1}), \tag{4.32}$$

$$bG^{t+1}e^{i\beta j} = bG^t e^{i\beta j} + k(aG^t e^{i\beta j+1} - aG^t e^{i\beta j-1}). \tag{4.33}$$

We then collect and rearrange terms to form the following system of equations.

$$\begin{bmatrix} (1-G) & k(2i\sin\beta) \\ k(2i\sin\beta) & (1-G) \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{4.34}$$

Using the fact that a solution, a and b, exists if and only if the determinant of the the system is zero [30], and setting $\gamma = k(2i\sin\beta)$, we get the following characteristic polynomial.

$$(1-G)^2 - \gamma^2 = G^2 - 2G + 1 - \gamma^2 = 0. \tag{4.35}$$

Solving for G we get,

$$G = \frac{2 \pm \sqrt{4 - 4(1-\gamma^2)}}{2},$$
$$= 1 \pm \gamma.$$

As $|G| > 1$, this leads to an unstable scheme.

4.3.2 Analysis of Formulation 2

If we follow the same steps as in the previous section, we end up with a very similar set of equations.

$$u_i^{t+1} = u_i^t + C_1 \sum_{p=1}^{np} \pm u_p^t + k\left(C_2(\rho_i^t - \rho_{i-1}^t) + C_3(\rho_{i+1}^t - \rho_i^t)\right), \tag{4.36}$$

$$\rho_i^{t+1} = \rho_i^t + C_1 \sum_{p=1}^{np} \pm \rho_p^t + k\left(C_2(u_i^{t+1} - u_{i-1}^{t+1}) + C_3(u_{i+1}^{t+1} - u_i^{t+1})\right). \tag{4.37}$$

The difference between the two sets of equations is the use of the updated values of u at time, $t+1$, at the nodes.

4.3.2.1 What happens if $v = 0$

As we did in the previous section, we set each particle velocity to zero. This means that $C_2 = C_3 = 0.5$ and $C_1 = 0$.

$$u_i^{t+1} = u_i^t + k(\rho_{i+1}^t - \rho_{i-1}^t), \tag{4.38}$$

$$\rho_i^{t+1} = \rho_i^t + k(u_{i+1}^{t+1} - u_{i-1}^{t+1}). \tag{4.39}$$

Let ξ and η represent the errors at the nodes for u and ρ, respectively. ξ and η are defined as follows,

$$\xi = aG^t e^{i\beta j}, \tag{4.40}$$

$$\eta = bG^t e^{i\beta j}. \tag{4.41}$$

Substituting the error terms ξ and η into Equations 4.38 and 4.39 gives us an expression for the error growth at the nodes.

$$aG^{t+1}e^{i\beta j} = aG^t e^{i\beta j} + k(bG^t e^{i\beta j+1} - bG^t e^{i\beta j-1}), \tag{4.42}$$

$$bG^{t+1}e^{i\beta j} = bG^t e^{i\beta j} + k(aG^{t+1} e^{i\beta j+1} - aG^{t+1} e^{i\beta j-1}). \tag{4.43}$$

We then collect and rearrange terms to form the following system of equations.

$$\begin{bmatrix} (1-G) & k(2i\sin\beta) \\ kG(2i\sin\beta) & (1-G) \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{4.44}$$

As stated previously, we know that for there to be a solution to the system of equations, 4.44, the determinant of the system must be zero and for simplification purposes, let $\gamma = k(2i\sin\beta)$); it then follows that,

$$(1-G)^2 - G\gamma^2 = G^2 - G(\gamma^2+2) + 1 = 0. \tag{4.45}$$

Solving for G, we get,

$$\begin{aligned} G &= \frac{(\gamma^2+2) \pm \sqrt{(\gamma^2+2)^2-4}}{2}, \\ &= 1 + \frac{\gamma^2 \pm \sqrt{\gamma^4+2\gamma^2}}{2}. \end{aligned}$$

Given that $\gamma = k(2i\sin\beta)$, G then has the form,

$$\begin{aligned} G &= 1 - 2k^2\sin^2\beta \pm \frac{\sqrt{16k^2\sin^4\beta - 8k^2\sin^2\beta}}{2}, \\ &= 1 - 2k^2\sin^2\beta \pm k\sin\beta\sqrt{4k^2\sin^2\beta - 2}. \end{aligned}$$

If $|k| < \frac{1}{\sqrt{2}}$, then G can be re written as follows,

$$G = 1 - 2k^2\sin^2\beta \pm k\sin\beta i\sqrt{2-4k^2\sin^2\beta}. \tag{4.46}$$

We can now compute the norm of G,

$$\begin{aligned} |G| &= \sqrt{\left(1-2k^2\sin^2\beta\right)^2 + \left(k\sin\beta\sqrt{2-4k^2\sin^2\beta}\right)^2}, \qquad (4.47) \\ &= \sqrt{1-2k^2\sin^2\beta}. \qquad (4.48) \end{aligned}$$

Because $|k| < \frac{1}{\sqrt{2}}$, the term $2k^2\sin^2\beta$ is always positive and less then one. This means that formulation 2 at velocity $u_0 = 0$ shows a sufficient condition for stability.

4.4 Results Using the Two Formulations

Figures 4.1, 4.2, and 4.3 are snapshots from a simulation that was run comparing Formulation 1 and Formulation 2. As can be seen by the figures at time step 530, Formulation 1 is already showing the results of the numerical instability while Formulation 2 is still stable. As can be seen by Figure 4.3, even Formulation 2 starts to becomes unstable at time step 700. Given that $v = 0$, it has been shown that Formulation 2 shows a sufficient condition for numerical stability. The question then arises what happens when $v \neq 0$. For example, is it the movement of particles that causes the nonlinear "ringing instability" [5] to arise? In the next section, it will be shown that the null space filter covered in the previous chapter can be used to remove what appears to be the "ringing instability".

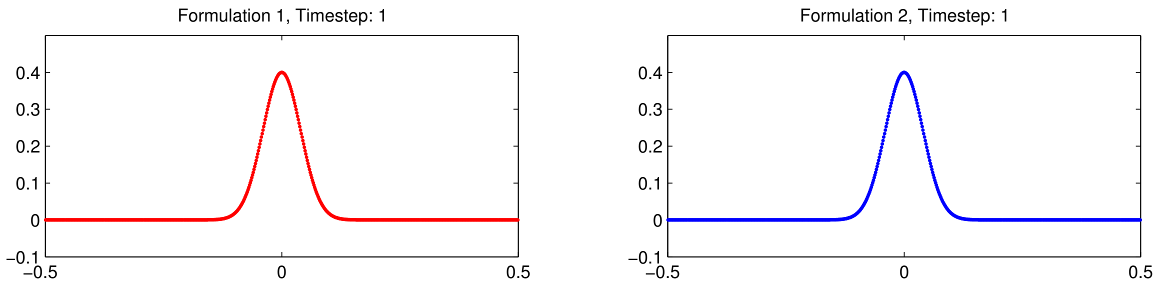

Figure 4.1. Initial time step

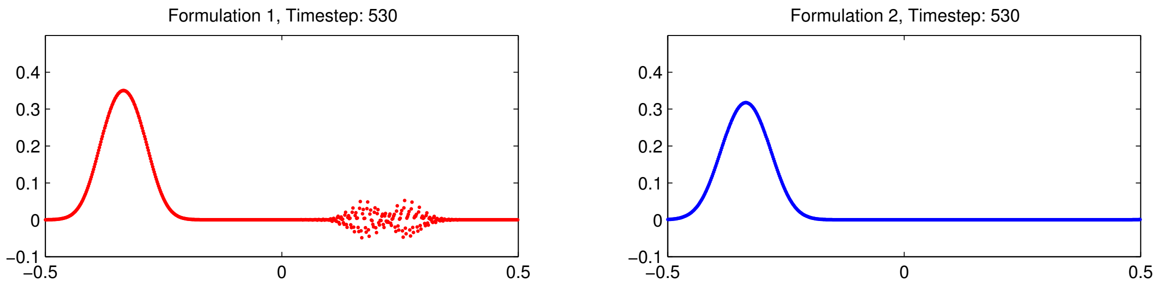

Figure 4.2. Time step 530

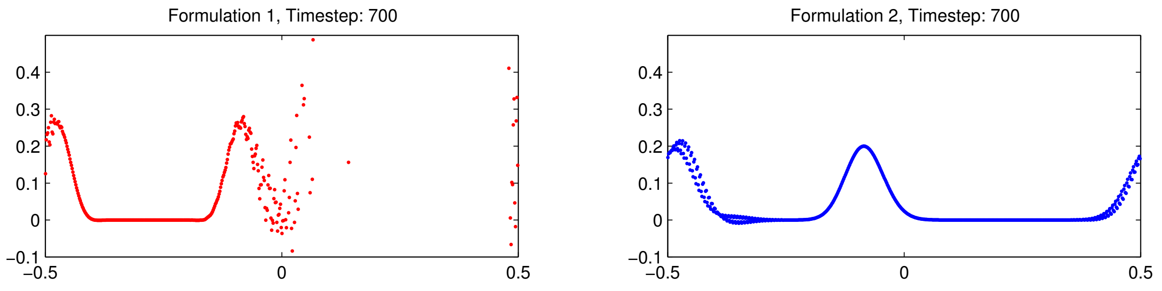

Figure 4.3. Time step 700

4.5 Applying the Nullspace Filter

In order to help in the explanation of how the nullspace filter is applied to our PIC solution, the method will be rewritten in terms of vectors and matrices.

4.5.1 Vector-Based Form of Formulation 2

Let $\mathbf{u}_p^t$ and $\boldsymbol{\rho}_p^t$ be the vectors containing the particle values for u and ρ at time step t and let $\mathbf{u}_i^t$ and $\boldsymbol{\rho}_i^t$ be the analogous vectors for the nodal values. The first step in the formulation is to map particle values to the nodes.

1. $\rho_i^t = \sum_{p=1}^n S_{ip}\rho_p^t$

This step is easily written in terms of a mapping matrix, but unlike the individual particle version, we first need to construct the matrix $\mathbf{S}_{ip}$. It is important to remember that this needs to be done after each iteration because the mapping is dependent on particle position. The building of the matrix will be handled by the function $buildSip(\mathbf{x}_p, \mathbf{x}_i)$ that takes as its inputs a vector containing the particle positions and vector with node positions. Now the first two steps of the new formulation look like this,

1. $\mathbf{S}_{ip} = buildSip(\mathbf{particles}^t, \mathbf{nodes})$

2. $\boldsymbol{\rho}_i^t = \mathbf{S}_{ip}\boldsymbol{\rho}_p^t$

Step two of the formulation involves taking the gradient of the nodal basis functions and a time integration step.

2. $u_p^{t+1} = u_p^t + c\frac{dt}{h}(\rho_{i+1}^t - \rho_i^t)$

We can rewrite the gradient portion of this equation as a matrix, $\mathbf{D}$, with dimension n by m, which calculates the gradient across the nodes and maps it to the particles. For example if we define the gradient as,

$$\frac{1}{h}(\rho_{i+1}^t - \rho_i^t), \tag{4.49}$$

then the matrix form can be written as

$$\mathbf{d}\rho_p^t = \mathbf{D}\boldsymbol{\rho}_i^t. \tag{4.50}$$

The updated matrix form for the steps are then written as follows:

3. $\mathbf{du}_p^t = \mathbf{D}\mathbf{u}_i^t$

4. $\mathbf{u}_p^{t+1} = \mathbf{u}_p^t + cdt\mathbf{d}\boldsymbol{\rho}_p^t$

The same idea is applied to next three steps of the formulation. The final step in the formulation is to advance the particles. This is simply rewritten in vector form where $\mathbf{v}$ is a vector with all the same entries of u_0.

5. $\mathbf{x}_p^{t+1} = \mathbf{x}_p^t + \mathbf{v}dt.$

For completeness, here is the rewritten formulation in terms of vectors and matrices,

1. $\mathbf{S}_{ip} = buildSip(\mathbf{u}_p^t, \mathbf{nodes})$
2. $\mathbf{u}_i^t = S_{ip}\mathbf{u}_p^t$
3. $\mathbf{du}_p^t = \mathbf{Du}_i^t$
4. $\boldsymbol{\rho}_p^{t+1} = \boldsymbol{\rho}_p^t + cdt\mathbf{d}\boldsymbol{u}_p^t$
5. $\boldsymbol{\rho}_i^{t+1} = \mathbf{S}_{ip}\boldsymbol{\rho}_p^{t+1}$
6. $\mathbf{d}\boldsymbol{\rho}_p^{t+1} = \mathbf{D}\boldsymbol{\rho}_i^{t+1}$
7. $\mathbf{u}_p^{t+1} = \mathbf{u}_p^t + cdt\mathbf{d}\boldsymbol{\rho}_p^{t+1}$
8. $\mathbf{x}_p^{t+1} = \mathbf{x}_p^t + \mathbf{v}dt.$

4.5.2 Updated Formulation

With Formulation 2 written in vector form, we can now take a look at how the nullspace noise can be filtered out during each computation cycle. As was seen in the previous chapter, nullspace noise can be injected any time a value goes from node to particle. When the interpolation is linear, the noise tends to be minimal, but in the cases when the piecewise constant gradient calculated using nodal values is mapped to a particle, then there tends to be more nullspace noise. There are two points in Formulation 2 when this occurs. The first is when the gradient of u_i is calculated, $\mathbf{du}_p^t = D\mathbf{u}_i^t$, and the second is when the gradient at ρ_i is calculated, $\mathbf{d}\boldsymbol{\rho}_p^{t+1} = D\boldsymbol{\rho}_i^{t+1}$. If after applying the $\mathbf{D}$ operator, the vectors $\mathbf{du}_i$ and $\mathbf{d}\boldsymbol{\rho}_i$ contain a component that is in the null space of $\mathbf{S}_{ip}$, then that contribution is a result of the $\mathbf{D}$ operator.

Here are the updated formulations with the calculated residual vectors. The vectors $\mathbf{ru1}$ and $\mathbf{r}\boldsymbol{\rho}\mathbf{1}$ are the null space components $\mathbf{du}_p^t$ and $\mathbf{d}\boldsymbol{\rho}_p^t$.

1. $\mathbf{S}_{ip} = buildSip(\mathbf{x}_p^t, \mathbf{nodes})$
2. $\mathbf{u}_i^t = \mathbf{S}_{ip}\mathbf{u}_p^t$

3. $\mathbf{du}_p^t = \mathbf{D}\mathbf{u}_i^t$
4. $\mathbf{ru1} = getRes(\mathbf{du}_p^t, \mathbf{S}_{ip})$
5. $\boldsymbol{\rho}_p^{t+1} = \boldsymbol{\rho}_p^t + cdt(\mathbf{d}\boldsymbol{u}_p^t - \mathbf{ru1})$
6. $\boldsymbol{\rho}_i^{t+1} = \mathbf{S}_{ip}\boldsymbol{\rho}_p^{t+1}$
7. $\mathbf{d}\boldsymbol{\rho}_p^{t+1} = \mathbf{D}\boldsymbol{\rho}_i^{t+1}$
8. $\mathbf{r}\boldsymbol{\rho}\mathbf{1} = getRes(\boldsymbol{\rho}_p^{t+1}, \mathbf{S}_{ip})$
9. $\mathbf{u}_p^{t+1} = \mathbf{u}_p^t + cdt(\mathbf{d}\boldsymbol{\rho}_p^{t+1} - \mathbf{r}\boldsymbol{\rho}\mathbf{1})$
10. $\mathbf{x}_p^{t+1} = \mathbf{x}_p^t + \mathbf{v}dt$

4.5.3 Comparison of Filtered and Nonfiltered Formulations

Figures 4.4, 4.5, and 4.6 are snapshots from a simulation that was run comparing Formulation 2 without the nullspace filter and Formulation 2 with the nullspace filter. As can be seen by Figure 4.5 at time step 700, the method without the nullspace filter is showing the results of the numerical nullspace noise. On the other hand, it is shown in Figure 4.6 that the method with the nullspace filter is still stable.

4.6 Cauchy Momentum Equation

This section will consider solving the Cauchy Momentum Equation,

$$\rho\frac{Dv}{Dt} = \nabla \cdot \sigma + b, \tag{4.51}$$

In what follows, body forces, b, will be ignored and as in the previous cases, only one space dimension will be considered. Equation 4.51 now becomes,

$$\rho\frac{Dv}{Dt} = \frac{\partial \sigma}{\partial x}. \tag{4.52}$$

Stress will be defined as,

$$\sigma = E\frac{\partial u}{\partial x}, \tag{4.53}$$

where E is Young's modulus and is a constant.

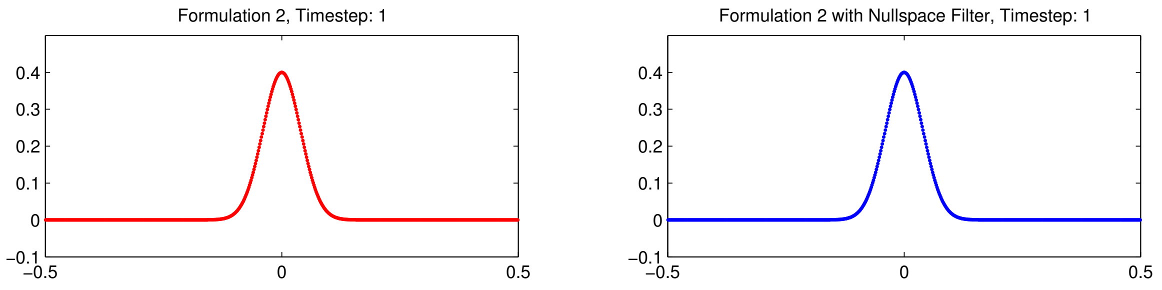

Figure 4.4. Initial time step

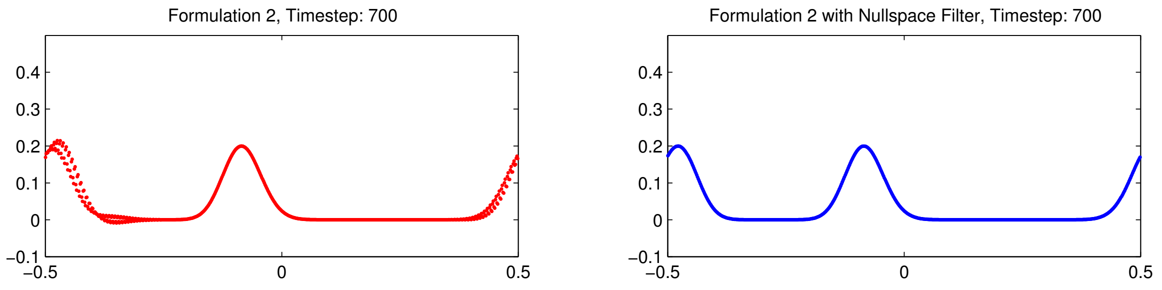

Figure 4.5. Time step 700

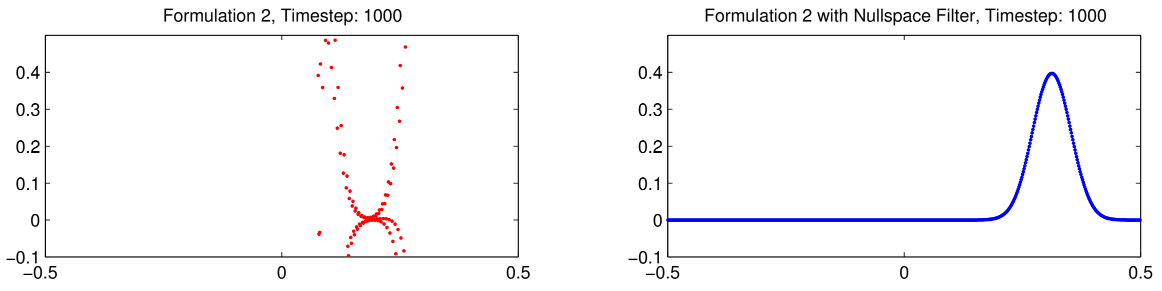

Figure 4.6. Time step 1000

4.6.1 Full Particle PIC

Let us explore one possible "full particle" PIC method. Substituting Equation 4.53 into Equation 4.52 gives,

$$\rho \frac{Dv}{Dt} = \frac{\partial}{\partial x}\left(E\frac{\partial u}{\partial x}\right). \tag{4.54}$$

Also, ρ will be constant throughout the domain. After rearranging of terms and setting $c = \sqrt{E/\rho}$, we get

$$\frac{Dv}{Dt} = c^2\frac{\partial^2 u}{\partial x^2}. \tag{4.55}$$

By noting, $\frac{Du}{Dt} = v$, we can form the following set of coupled equations that gives us the standard 1d wave equation in Lagrangian form,

$$\left.\begin{aligned} \frac{Du}{Dt} &= v \\ \frac{Dv}{Dt} &= c^2\frac{\partial^2 u}{\partial x^2} \end{aligned}\right\} \Longrightarrow \frac{D^2u}{Dt^2} = c^2\frac{\partial^2 u}{\partial x^2} \tag{4.56}$$

If $u(x,0) = f(x)$ and $v(x,0) = 0$, then we have d'Alembert's formula as the analytic solution to the problem.

The formulation of the "full particle" PIC method will follow the basic steps of mapping particle displacement and velocity to the nodes, computing acceleration at the nodes using a second order finite difference stencil, updating the velocity at the nodes using the acceleration, and updating velocity and displacement at the particles. In a step-by-step form, we get,

1. $u_i^t = \sum_{p=1}^{np} S_{ip}u_p^t$
2. $v_i^t = \sum_{p=1}^{np} S_{ip}v_p^t$
3. $a_i^t = \frac{c^2}{h^2}(u_{i+1}^t - 2u_i^t + u_{i-1}^t)$
4. $v_i^{t+1} = v_i^t + dt * a_i^t$
5. $v_p^{t+1} = v_p^t + dt\sum_{p=1}^{np} S_{ip}a_i^t$
6. $u_p^{t+1} = u_p^t + dt\sum_{p=1}^{np} S_{ip}v_i^{t+1}$.

4.6.2 Analysis of Full Particle PIC

As in the "full particle" PIC formulations of the gas dynamics problem, the method will be rewritten in terms of the nodal values for u_i. Beginning with

$$u_i^{t+1} = \sum_{p=1}^{np} S_{ip}^{t+1} u_p^{t+1} \tag{4.57}$$

we then substitute in u_p^{t+1} to get,

$$\begin{aligned} u_i^{t+1} &= \sum_{p=1}^{np} S_{ip}^{t+1} u_p^{t+1} \\ &= \sum_{p=1}^{np} S_{ip}^{t+1} (u_p^t + dt \sum_{p=1}^{np} S_{ip}^t v_i^{t+1}) \\ &= \sum_{p=1}^{np} S_{ip}^{t+1} u_p^t + dt \sum_{p=1}^{np} S_{ip}^{t+1} S_{ip}^t v_i^{t+1} \\ &= u_i^t + \sum_{p=1}^{np} (S_{ip}^{t+1} - S_{ip}^t) u_p^t + dt \sum_{p=1}^{np} S_{ip}^{t+1} S_{ip}^t v_i^{t+1}. \end{aligned}$$

If the following condition holds

$$\sum_{p=1}^{np} S_{ip} = 1 \tag{4.58}$$

then we get the following

$$\begin{aligned} u_i^{t+1} &= u_i^t + \sum_{p=1}^{np} (S_{ip}^{t+1} - S_{ip}^t) u_p^t + dt \sum_{p=1}^{np} S_{ip}^{t+1} \sum_{p=1}^{np} S_{ip}^t v_i^{t+1} \\ &= u_i^t + \sum_{p=1}^{np} (S_{ip}^{t+1} - S_{ip}^t) u_p^t + v_i^{t+1} dt. \end{aligned}$$

Similarly, for v_i we get,

$$\begin{aligned} v_i^{t+1} &= \sum_{p=1}^{np} S_{ip}^{t+1} v_p^{t+1} \\ &= \sum_{p=1}^{np} S_{ip}^{t+1} (v_p^t + dt \sum_{p=1}^{np} S_{ip}^t a_i^t) \\ &= v_i^t + \sum_{p=1}^{np} (S_{ip}^{t+1} - S_{ip}^t) v_p^t + dt \sum_{p=1}^{np} S_{ip}^{t+1} \sum_{p=1}^{np} S_{ip}^t a_i^t \\ &= v_i^t + \sum_{p=1}^{np} (S_{ip}^{t+1} - S_{ip}^t) v_p^t + a_i^t dt. \end{aligned}$$

Now substitute in for a_i

$$\begin{aligned} v_i^{t+1} &= v_i^t + \sum_{p=1}^{np} (S_{ip}^{t+1} - S_{ip}^t) v_p^t + a_i^t dt, \\ &= v_i^t + \sum_{p=1}^{np} (S_{ip}^{t+1} - S_{ip}^t) v_p^t + \frac{c^2 dt}{h^2} (u_{i+1}^t - 2u_i^t + u_{i-1}^t). \end{aligned}$$

We can now combine the two equations, u_i^{t+1} and v_i^{t+1},

$$\begin{aligned} u_i^{t+1} &= u_i^t + \sum_{p=1}^{np} (S_{ip}^{t+1} - S_{ip}^t) u_p^t + v_i^{t+1} dt, \\ &= u_i^t + \sum_{p=1}^{np} (S_{ip}^{t+1} - S_{ip}^t) u_p^t + v_i^t dt + \sum_{p=1}^{np} (S_{ip}^{t+1} - S_{ip}^t) v_p^t + a_i^t dt^2. \end{aligned}$$

Knowing that

$$u_i^t = u_i^{t-1} + \sum_{p=1}^{np} (S_{ip}^t - S_{ip}^{t-1}) u_p^{t-1} + v_i^t dt \tag{4.59}$$

we can arrive at $v_i^t dt$,

$$v_i^t dt = u_i^t - u_i^{t-1} - \sum_{p=1}^{np} (S_{ip}^t - S_{ip}^{t-1}) u_p^{t-1}. \tag{4.60}$$

After substituting for v_i^t and rearranging some terms, we get the following finite difference scheme at the nodes,

$$\frac{u_i^{t+1} - 2u_i^t + u_i^{t-1}}{dt^2} = c^2 \frac{u_{i+1}^t - 2u_i^t + u_{i-1}^t}{h^2} + S, \tag{4.61}$$

where

$$S = \frac{1}{dt^2} \sum_{p=1}^{np} (S_{ip}^{t+1} - S_{ip}^t)(u_p^t + v_p^t dt) - (S_{ip}^t - S_{ip}^{t-1}) u_p^{t-1}. \tag{4.62}$$

4.6.2.1 Von Neumann Analysis of Full Particle PIC Formulation

Again, we set the particle velocity to zero and use Von Neumann analysis. Let ξ,

$$\xi = aG^t e^{i\beta j}, \tag{4.63}$$

represent the error at the nodes for u. Substituting in ξ for the u gives us the following,

$$\frac{aG^{t+1}e^{i\beta j} - 2aG^t e^{i\beta j} + aG^{t-1} e^{i\beta j}}{dt^2} = c^2 \frac{aG^t e^{i\beta(j+1)} - 2aG^t e^{i\beta j} + aG^t e^{i\beta(j-1)}}{h^2}, \tag{4.64}$$

Canceling out terms gives the following,

$$\frac{G - 2G^t + G^{t-1}}{dt^2} = c^2 \frac{G^t e^{i\beta(j+1)} - 2G^t e^{i\beta j} + G^t e^{i\beta(j-1)}}{h^2}. \tag{4.65}$$

Letting $k = \frac{cdt}{h}$ gives the following set of equations,

$$G^2 - 2G\gamma + 1 = 0, \qquad \gamma = 1 - 2k^2 \sin^2(\beta/2). \tag{4.66}$$

and solving for the the growth term, G, gives,

$$G = \gamma \pm \sqrt{\gamma^2 - 1}. \tag{4.67}$$

If $k \leq 1$ then $|\gamma| \leq 1$ and $\sqrt{1-\gamma^2}$ is real for all β and then we get,

$$\begin{aligned} G &= \gamma \pm i\sqrt{1-\gamma^2} \\ |G| &= \sqrt{\gamma^2 + (1-\gamma^2)} \\ &= 1. \end{aligned}$$

It can be seen that Von Neumann analysis gives a sufficient condition for stability at the nodes; what is left to be determined is what happens with the term S. If particles do not move, then $S_{ip}^{t+1} = S_{ip}^{t} = S_{ip}^{t-1}$ and $S = 0$; beyond that, it is difficult to determine what instabilities may arise without applying further analysis to the term S found in Equation 4.61.

CHAPTER 5

COMPARISON OF METHODS

Now that we have covered the different approaches used by some particle methods, devised a means for dealing with the “ringing instabilities”, and looked at the underlying stability of the different methods, we can compare the different approaches to see which performs better and under what conditions. We have looked at both the gas dynamics and the Cauchy momentum problems. We have seen that both can be reduced down to the standard wave equation for which we have a known solution in the d’Alembert formula. By imposing periodic boundary conditions and an initial condition that is also periodic at the boundaries, we have a true solution that we can compare against. For this chapter, the model problem, give the initial and boundary conditions, with a defined analytic solution will be used. Three different methods that incorporate the different approaches to the problem will be used and compared. Then the results are shown how each method performed against the analytic solution.

5.1 Cauchy Momentum Equation

For clarity, here is the set of equations that we will be solving,

$$\frac{Du}{Dt} = v, \tag{5.1}$$

$$\rho\frac{Dv}{Dt} = \frac{\partial\sigma}{\partial x}, \tag{5.2}$$

with $\sigma = Eu_x$ and Young’s modulus, E, being constant.

Given that we are dealing with a linear stress model, E being constant, there are two approaches that can be taken to finding the gradient of stress. The first is to use the values of stress in the gradient calculation and the second approach is to rewrite the gradient of stress in terms of displacement,

$$\frac{\partial\sigma}{\partial x} = \frac{\partial}{\partial x}\left(E\frac{\partial u}{\partial x}\right), \tag{5.3}$$

$$= E\frac{\partial^2 u}{\partial x^2}. \tag{5.4}$$

Both approaches will be explored here.

5.1.1 Parameters and Initial Conditions

The initial conditions and parameters for the numerical solutions are as follows. The spatial domain is $-.5 < x < .5$, node spacing is $h = .01$, and the initial particle distribution is two particles evenly spaced between nodes. Particle masses are $m_p = 1$ and initial velocity is $v(x, 0) = 0$, where initial $v_p = v(x_p, 0)$. The initial displacement is $u(x, 0) = .001e^{-60x^2}$ where $u_p = u(x_p, 0)$. The Young's modulus is $E = .0001$ and the initial strain is $u_x(x, 0) = -.12e^{-60x^2}$ where $u_{xp} = u_x(x_p, 0)$.

5.1.2 The Analytic Solution

Given that the initial velocity is zero, the analytic solution to the problem becomes,

$$u(x,t) = .0005\left(e^{-60(x-.01t)^2} + e^{-60(x+.01t)^2}\right). \tag{5.5}$$

At each time step, $t^n = dt * n$, where n is the nth time step, $u_p^n = u(x_p^n, t^n)$. With this linear elastic model, the displacements, u_p, are small in comparison to the node spacing; consequently, few particles cross over nodes.

5.1.3 Method 1 - Full Particle PIC

Method 1 is a "full particle" PIC method where the values for displacement are used instead of the values for stress in the gradient of stress calculation. The first step is to compute the mass at the nodes. Given that $m_p = 1$, we get the following,

$$\begin{aligned} m_i &= \sum_p \phi_{ip} m_p, && (5.6) \\ &= \sum_p \phi_{ip}. && (5.7) \end{aligned}$$

The second and third steps are to map displacements and velocities to the nodes,

$$\begin{aligned} u_i &= \sum_p \phi_{ip} u_p. && (5.8) \\ v_i &= \frac{\sum_p \phi_{ip} m_p v_p}{m_i}. && (5.9) \end{aligned}$$

The fourth and fifth step is to calculate the force and acceleration at the nodes,

$$\begin{aligned} f_i &= E\left(\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2}\right) && (5.10) \\ a_i &= \frac{f_i}{m_i}. && (5.11) \end{aligned}$$

Step six updates the velocity at the nodes,

$$v_i^{n+1} = v_i^n + a_i dt. \tag{5.12}$$

Steps seven, eight, and nine update velocity, displacement, and position at the particles, respectively:

$$v_p^{n+1} = v_p^n + dt \sum_i \phi_{ip} a_i, \tag{5.13}$$

$$u_p^{n+1} = u_p^n + dt \sum_i \phi_{ip} v_i, \tag{5.14}$$

$$x_p^{n+1} = x_p^n + dt \sum_i \phi_{ip} v_i. \tag{5.15}$$

5.1.4 Method 2 - Full Particle PIC

Method 2 differs from Method 1 in that stress is calculated from particle strains. As with Method 1, this is a "full particle" PIC method. The first step in this method is the same as Method 1, to map particle masses to the nodes. The second step is to map the strain values to the nodes using,

$$u_{xi} = \sum_p \phi_{ip} u_{xp}, \tag{5.16}$$

where u_{xi} is the strain at the nodes and u_{xp} is strain at the particles. The third step is to calculate accelerations at the particles,

$$a_p = E \sum_i \frac{\nabla \phi_{ip} u_{xi}}{m_i}. \tag{5.17}$$

The fourth step is to remove the nullspace noise from the gradient of stress calculation as described in sections 3.5.2-3.5.6 of Chapter 3. For brevity, we will simply define the process as $removeNull()$,

$$a_p = removeNull(S_{ip}, a_p). \tag{5.18}$$

The fifth and sixth steps are to update velocity, displacement, and position of the particles.

$$v_p^{n+1} = v_p^n + dt a_p, \tag{5.19}$$

$$u_p^{n+1} = u_p^n + dt v_p, \tag{5.20}$$

$$x_p^{n+1} = x_p^n + dt v_p. \tag{5.21}$$

Step nine is to map the velocities to the nodes,

$$v_i^{n+1} = \frac{\sum_p \phi_{ip} v_p^{n+1} m_p}{m_i}. \tag{5.22}$$

Step ten is calculate the velocity gradients at the particles

$$v_{xp} = \sum_i \nabla \phi_{ip} v_i. \tag{5.23}$$

The final step is to update strain at the particles,

$$u_{xp}^{n+1} = u_{xp}^n + v_{xp} dt. \tag{5.24}$$

5.1.5 Method 3 - Material Point Method

Method 3 is an implementation of the Material Point Method. Like the other two methods, the first step is to map particle masses to the nodes. The second step is to map velocity to the nodes. The third is to calculate force at the nodes,

$$f_i = E \sum_p \nabla \phi_{ip} u_{xp}. \tag{5.25}$$

Step four is to use the calculated force to calculate the acceleration at the nodes,

$$a_i = -\frac{f_i}{m_i}. \tag{5.26}$$

The fifth step is to update the velocity at the node,

$$v_i^{n+1} = v_i^n + a_i dt. \tag{5.27}$$

Steps six, seven, and eight are to update the velocity, displacement, and position at the particles

$$v_p^{n+1} = v_p^n + dt \sum_i S_{ip} a_i, \tag{5.28}$$

$$u_p^{n+1} = u_p^n + dt \sum_i S_{ip} v_i^{n+1}, \tag{5.29}$$

$$x_p^{n+1} = x_p^n + dt \sum_i S_{ip} v_i^{n+1}. \tag{5.30}$$

Step nine is to compute the velocity gradient at the particles,

$$v_{xp} = \sum_i \nabla \phi_{ip} v_i^{n+1}. \tag{5.31}$$

The final step is to update the strain values at the particles,

$$u_{xp}^{n+1} = u_{xp}^n + dt v_{xp}. \tag{5.32}$$

In MPM strain can be updated either at the beginning of the calculation cycle or at the end [1].

5.1.6 Results

Numerical simulations were run using all three methods. All runs used had the same initial and boundary conditions. The calculations were run for a total of 10000 time steps. Figure 5.1 shows a plot of the relative error growth over the 10000 time steps. As can be seen, the MPM version performed best in terms of the relative error growth. What is interesting is that the method where the gradient of strain is calculated implicitly performs the poorest of the three in terms of error growth. The difference being displacement is used in method 1 to calculate acceleration versus methods 2 and 3, which use strain.

5.1.7 Ringing Instability in Strain Calculation

Steps nine and ten of the MPM formulation have the potential for a "ringing instability". Figure 5.2 shows the initial values for both displacement and strain. Figure 5.3 shows time step 2000 where there is visible "ringing instability" in the strain plot. The null space filter can be applied between steps nine and ten. Figures 5.4 and 5.5 show a side-by-side comparison of the unfiltered and filtered version. As can be seen, the nullspace filter smooths out the "ringing instabilities" found in the strain calculations. Figure 5.6 shows the results of the error calculation for both cases. The unfiltered method is more accurate. More needs to done to understand the reasons for this, but one hypothesis is that the smoothing caused by the filter has an averaging affect that reduces the accuracy.

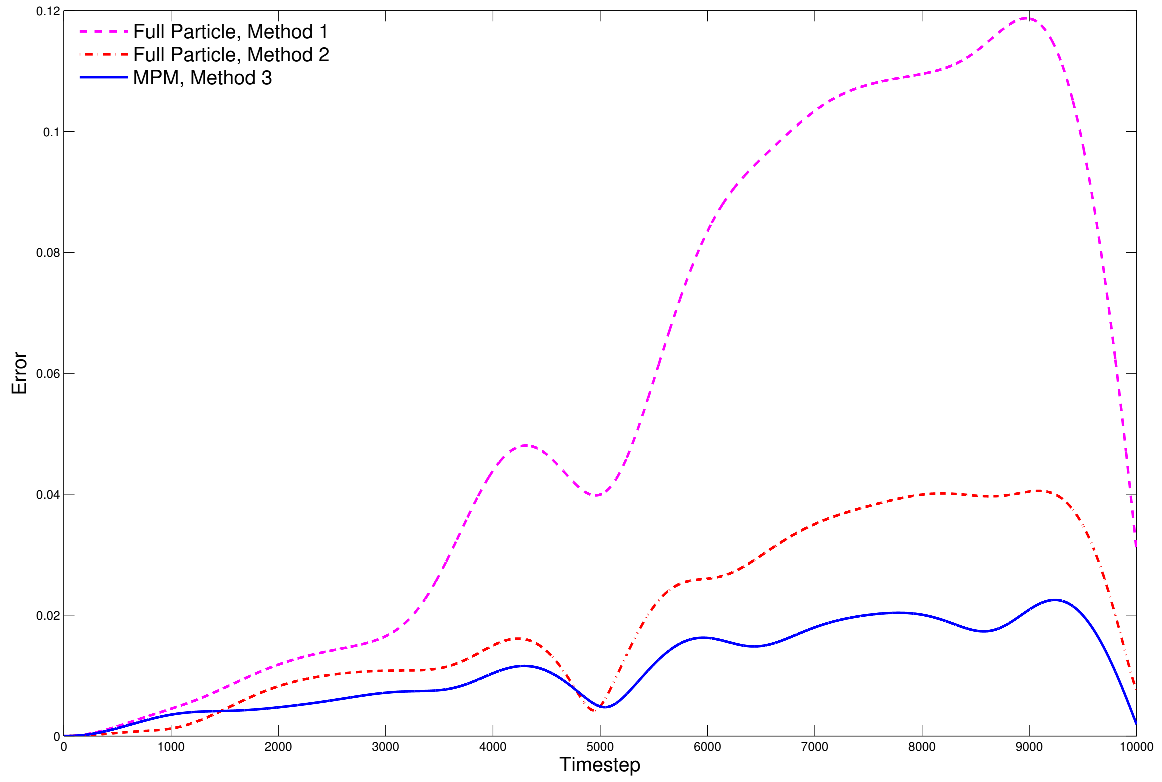

Figure 5.1. Relative error of methods versus time

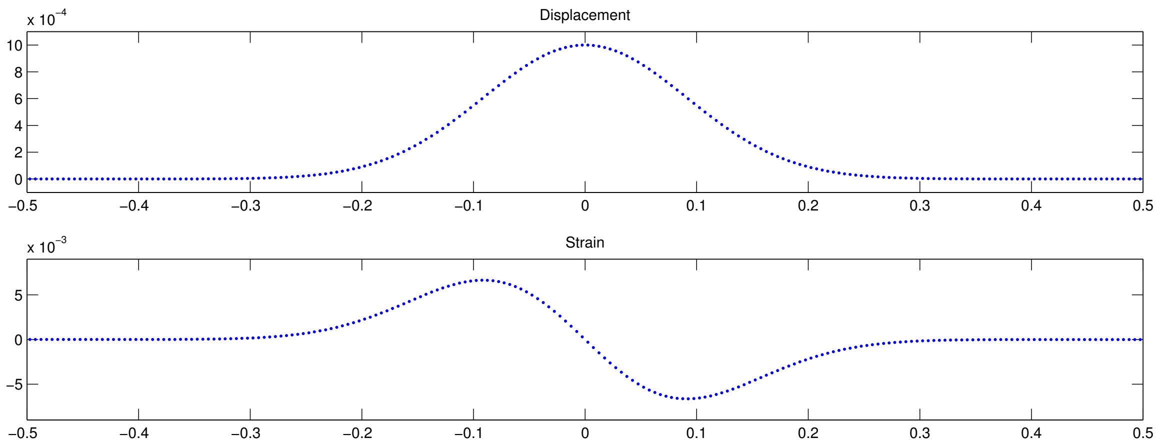

Figure 5.2. Displacement and strain, time step = 1

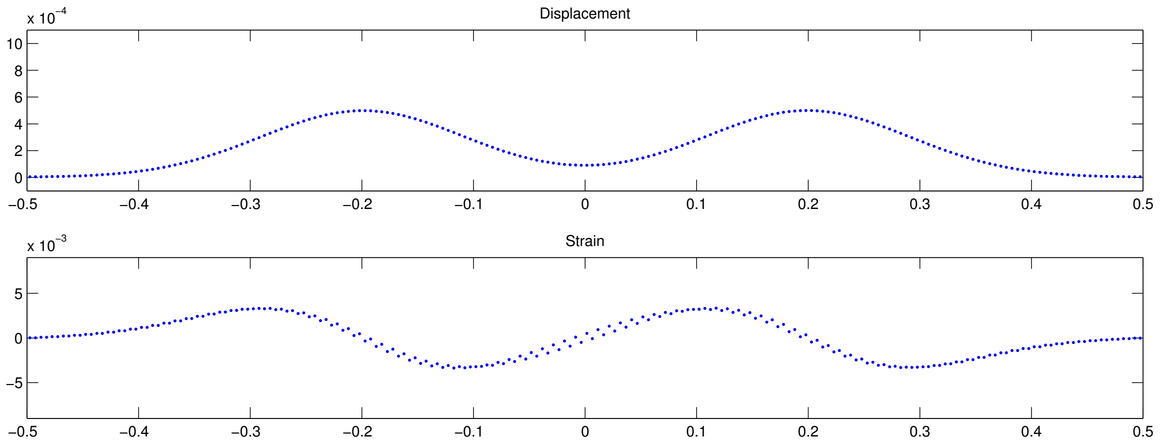

Figure 5.3. Displacement and strain, time step = 2000

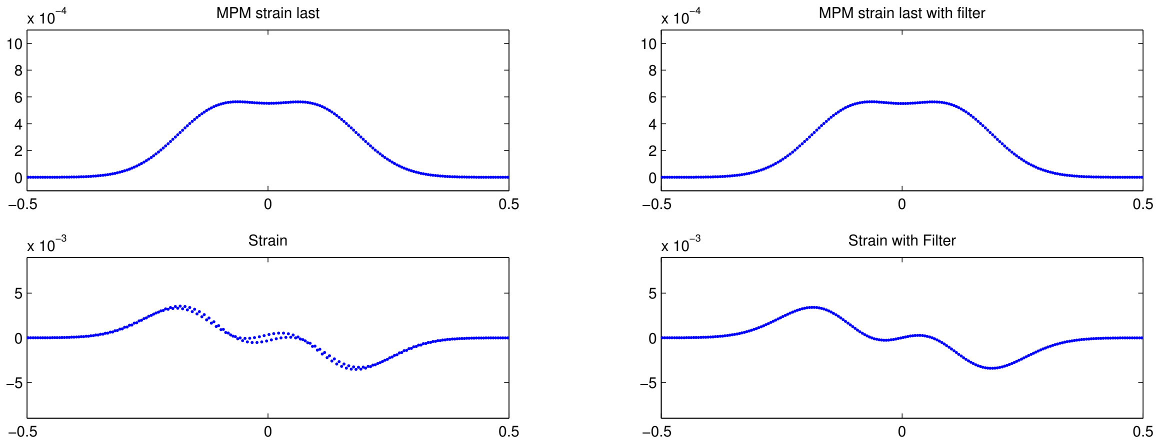

Figure 5.4. Displacement and strain, time step = 1000

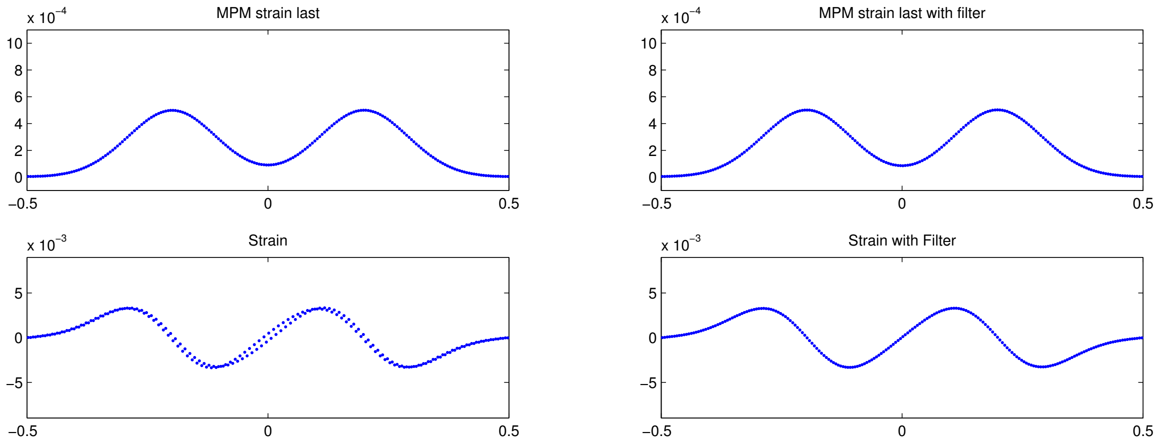

Figure 5.5. Displacement and strain, time step = 2000

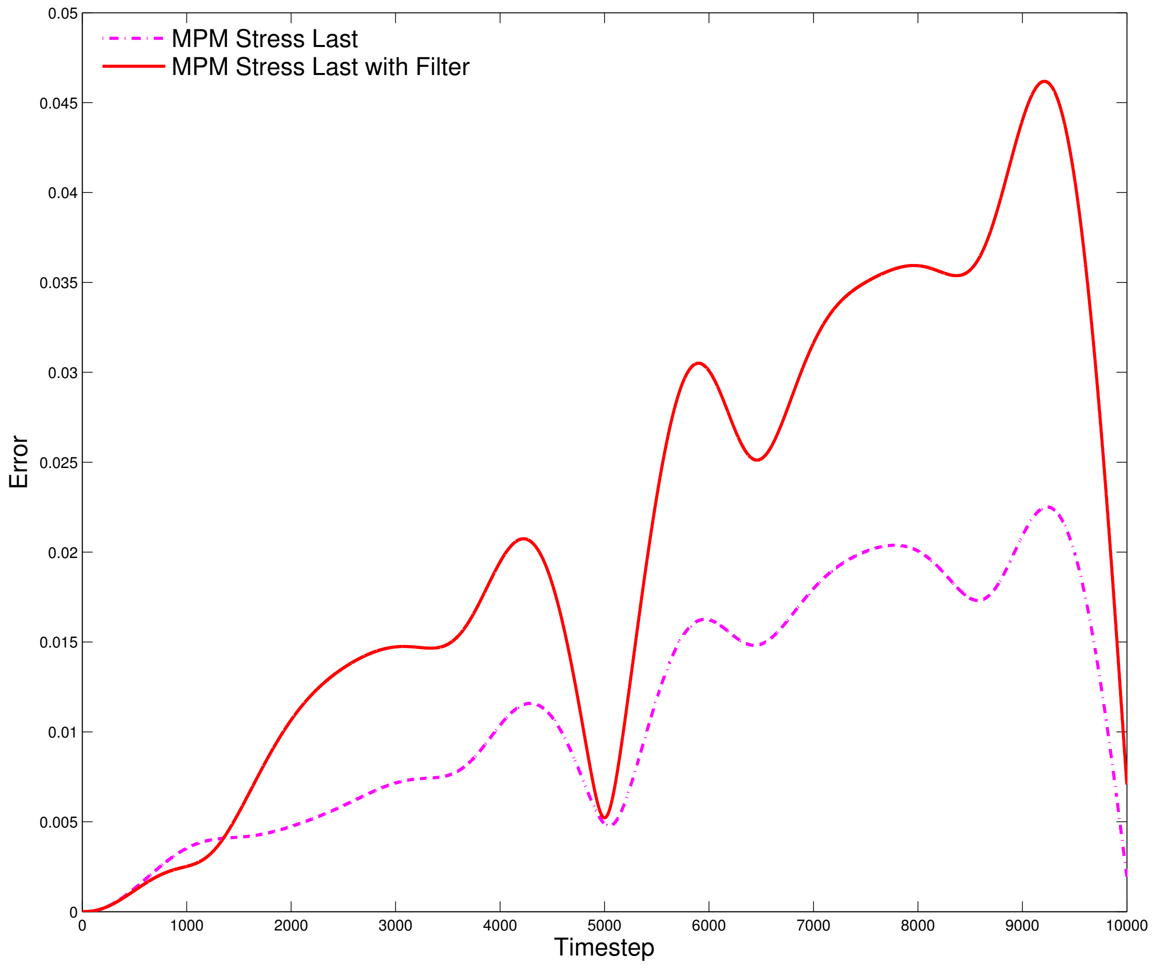

Figure 5.6. Relative error between unfiltered and filtered strain

CHAPTER 6

CONCLUSION

In this thesis, we briefly reviewed some of the key methods and concepts in the field of PIC-like particle methods. We have studied different methods of computing gradients in particle methods. How gradients are computed can play a key role in the stability of particle methods. It has been shown that the "ringing instability" comes about as numerical "noise" is introduced in the nullspace of the mapping from particles to nodes. Two methods were introduced to suppress the "ringing instabilities". The first used the SVD of the mapping matrix to define a basis for a nullspace from which the nullspace component of the particle vector could be removed. While the SVD method works well for small problems, its computational complexity and lack of scalability makes it impractical for large multicore simulations. The second method removes the nullspace "noise" by mapping particle values to the nodes, which by definition removes the nullspace, and then interpolates the values back to the nodes. This method removes the noise locally and scales to multicore systems.

We have also shown that the underlying nodal scheme plays a key role in the stability of the particle method. By writing a particle method in terms of its nodal values, methods of analysis used in finite difference methods can be used to analyze the stability of a particle method. It was shown that a slight change in a nodal scheme can turn a method that is unstable to one that shows sufficient conditions for stability.

Lastly, we used the Cauchy momentum equation as a test model to compare different particle methods. By ignoring the external body force, imposing periodic boundary conditions, and using linear elastic model for stress, we were able to use the d'Alembert formulation as the analytic solution to compare against. The MPM method performed the best in this test problem. It was also shown that the "ringing instability" did occur in the strain updates and the nullspace noise filter was used to smooth out the instabilities. As an open question, it is unclear why the filtered method did not perform as well as the unfiltered method for the MPM method.

REFERENCES

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

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

[3] M. Berzins, *Private communication.*

[4] C. Birdsall and A. Langdon, *Plasma Physics via Computer Simulation*, Taylor and Francis Group, New York, first ed., 2005.

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

[6] ——, *{FLIP} mhd: A particle-in-cell method for magnetohydrodynamics*, Journal of Computational Physics, 96 (1991), pp. 163 – 192.

[7] J. Brackbill, D. Kothe, and H. Ruppel, *Flip: A low-dissipation, particle-in-cell method for fluid flow*, Computer Physics Communications, 48 (1988), pp. 25 – 38.

[8] J. Brackbill and H. Ruppel, *Flip: A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions*, Journal of Computational Physics, 65 (1986), pp. 314 – 343.

[9] R. Bridson, *Fluid Simulation for Computer Graphics*, A K Peters, Wellesley, MA, first ed., 2008.

[10] Z. Chen, Y. Han, S. Jiang, Y. Gan, and T. D. Sewell, *A multiscale material point method for impact simulation*, Theoretical and Applied Mechanics Letters, 2 (2012), pp. –.

[11] N. P. Daphalapurkar, *Multiscale simulation from atomistic to continuum – coupling molecular dynamics(md) with material point method (mpm)*, Master's thesis, Oklahoma State University, 2004.

[12] J. M. Dawson, *Particle simulation of plasmas*, Rev. Mod. Phys., 55 (1983), pp. 403–447.

[13] M. W. Evans and F. H. Harlow, *The particle-in-cell method for hydrodynamic calculations*, Tech. Rep. LA-2139, Los Alamos Scientific Laboratory of University of California, June 1957.

[14] R. A. Gingold and J. J. Monaghan, *Smoothed particle hydrodynamics-theory and application to non-spherical stars*, Monthly Notices of the Royal Astronomical Society, 181 (1977), pp. 375–389.

[15] G. H. Golub and C. F. V. Loan, *Matrix Computations*, The John Hopkins University Press, third ed., 1996.

[16] Z. Guo and W. Yang, *Mpm/md handshaking method for multiscale simulation and its application to high energy cluster impacts*, International Journal of Mechanical Sciences, 48 (2006), pp. 145 – 159. ¡ce:title¿7th Asia-Pacific Symposium on Advances in Engineering Plasticity and its Applications (AEPA 2004)¡/ce:title¿.

[17] F. H. Harlow, *A machine calculation method for hydrodynamic problems*, Tech. Rep. LAMS-1956, Los Alamos Scientific Laboratory of University of California, November 1955.

[18] ——, *The particle-in-cell method for two-dimensional hydrodynamic problems*, Tech. Rep. LAMS-2082, Los Alamos Scientific Laboratory of University of California, August 1956.

[19] ——, *The particle-in-cell method for fluid dynamics*, Methods in Computational Physics, 3 (1964).

[20] R. Hockney and J. Eastwood, *Computer Simulation using Particles*, Taylor and Francis Group, New York, first ed., 1988.

[21] I. Ionescu, J. E. Guilkey, M. Berzins, J. A. Weiss, and R. M. Kirby, *Simulation of soft tissue failure using the material point method*, Journal of Biomechanical Engineering, 128 (2006), pp. 917–924.

[22] M. Kirby, *Private communication.*

[23] J. D. Logan, *Applied Partial Differential Equations*, Springer-Verlag, New York, second ed., 2004.

[24] J. Ma, *Multiscale simulation using the generalized interpolation material point method,discrete dislocations and molecular dynamics*, Master's thesis, Oklahoma State University, 2002.

[25] B. Marder, *Gap-a pic-type fluid code*, Mathermatics of Computation, 29 (1975), pp. 434–446.

[26] J. J. Monaghan, *Smoothed particle hydrodynamics*, Annual Review of Astronomy and Astrophysics, 30 (1992), pp. 543–574.

[27] Moxfyre, *File:aliasingsines.svg.* http://en.wikipedia.org/wiki/File:AliasingSines.svg.

[28] J. A. Nairn, *Numerical Simulations of Transverse Compression and Densification in Wood*, Wood and Fiber Science, 38 (2006), pp. 576–591.

[29] H. Neunzert, A. Klar, and J. Struckmeier, *Particle methods: Theory and applications*, tech. rep., ICIAM 95: proceedings of the Third International Congress on Industrial and Applied Mathematics held in, 1995.

[30] J. Noye, *Finite difference techniques for partial differential equations*, Computational Techniques for Differential Equations, (1984).

[31] M. Steffen, R. M. Kirby, and M. Berzins, *Analysis and reduction of quadrature errors in the material point method (mpm)*, International Journal for Numerical Methods in Engineering, 76 (2008), pp. 922–948.

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

[33] D. Sulsky and A. Kaul, *Implicit dynamics in the material-point method*, Computer Methods in Applied Mechanics and Engineering, 193 (2004), pp. 1137 – 1170. ¡ce:title¿Meshfree Methods: Recent Advances and New Applications¡/ce:title¿.

[34] D. Sulsky, H. Schreyer, K. Peterson, R. Kwok, and M. Coon, *Using the material-point method to model sea ice dynamics*, Journal of Geophysical Research: Oceans, 112 (2007), pp. n/a–n/a.

[35] L. N. Trefethen and I. David Bau, *Numerical Linear Algebra*, SIAM, 1997.

[36] H. Zhang, K. Wang, and Z. Chen, *Material point method for dynamic analysis of saturated porous media under external contact/impact of solid bodies*, Computer Methods in Applied Mechanics and Engineering, 198 (2009), pp. 1456 – 1472.

[37] Y. Zhu and R. Bridson, *Animating sand as a fluid*, ACM Trans. Graph., 24 (2005), pp. 965–972.