9-1-2015

# Improving the Material Point Method

Ming Gong

Ming Gong

*Candidate*

Mathematics and Statistics

*Department*

This dissertation is approved, and it is acceptable in quality and form for publication:

*Approved by the Dissertation Committee:*

Prof. Deborah Sulsky, Chair

Prof. Howard L. Schreyer, Member

Prof. Stephen Lau, Member

Prof. Daniel Appelo, Member

Prof. Paul J. Atzberger, Member

# Improving the Material Point Method

by

**Ming Gong**

B.S., Applied Math, Dalian Jiaotong University, 2009
B.S., Software Engineering, Dalian Jiaotong University, 2009
M.S., Applied Math, University of New Mexico, 2012

DISSERTATION

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy
Mathematics

The University of New Mexico

Albuquerque, New Mexico

July, 2015

# Dedication

*For Xuanhan, Sumei, and Zhongbao.*

# Acknowledgments

I would like to thank my advisor, Professor Deborah Sulsky, for her support and guidance during my study at UNM. Not only have I learned the deep knowledge in numerical methods in PDEs from her, but I have also learned to do research in general. I would also like to thank the other members of my committee. I thank Howard Schreyer for teaching me continuum equations and constitutive modeling. Thanks to Stephen Lau for his teaching in numerical analysis and his patience in helping me set up a solid foundation in numerical analysis. Thanks to Daniel Appelo for his instruction in the Finite Element course and his fruitful suggestions to problems in my research. Thanks to Paul J. Atzberger for being my committee member and his interest in my research.

This dissertation would never have been completed without the support of my family. My mother, Sumei Shen, and my father, Zhongbao Gong, were always supportive and encouraging of the things I wanted to do, and have helped me take care of my daughter. My two year old daughter, Xuanhan Gong, provided me the joy and strength necessary to complete the final part of my study.

# Improving the Material Point Method

by

**Ming Gong**

B.S., Applied Math, Dalian Jiaotong University, 2009

B.S., Software Engineering, Dalian Jiaotong University, 2009

M.S., Applied Math, University of New Mexico, 2012

PH.D, Mathematics, University of New Mexico, 2015

## Abstract

The material point method (MPM) was designed to solve problems in solid mechanics, and it has been used widely in research and industry. In MPM, the equations of motion are solved on a background grid and Lagrangian material points represent the geometry of the body, carrying history-dependent material properties. The focus of this work is on the convergence properties of MPM in terms of order of accuracy and stability. It has been shown numerically that MPM loses convergence and suffers cell crossing errors for large deformation problems. Two remedies that have been proposed are the the generalized interpolation material point method (GIMP) and the convected particle domain interpolation method (CPDI). Both GIMP and CPDI try to improve MPM by altering the geometry of the evolved material-point domain. Such changes lead to improvement of the quadrature part of the MPM algorithm. In this work, we give a different approach to improving MPM by combining ideas from meshfree particle methods and finite element methods. Such an approach provides a general framework for improving MPM. Not only has the framework produced the

improved material point method (IMPM), but it can also be used to improve existing variations of MPM, such as CPDI. In addition, a Neumann stability analysis of MPM on the linearized equations of motion is provided. However, this analysis did not provide insight into the observed behavior of MPM. Limitations of the analysis are given that perhaps account for the lack of correlation between the analysis and observations.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

This dissertation seeks to solve problems in solid mechanics in which a body undergoes large deformation. The underlying differential equations are nonlinear and the solutions are numerical. Numerical methods in computational mechanics include the classical finite element methods and, more recently, meshfree particle methods. The numerical method I am interested in is called the material-point method.

In solid mechanics, equations of motion can be expressed using Lagrangian coordinates, or Eulerian coordinates. In Lagrangian coordinates, the equations of motion are written in the initial configuration. In Eulerian coordinates, the equations of motion are written in the current configuration. The corresponding discretized equations of motion can similarly be expressed using Lagrangian coordinates or Eulerian coordinates. There are two general types of numerical methods which solve the equations of motion: mesh dependent methods, such as Finite Element Methods, and mesh independent methods, such as meshfree particle methods.

For mesh-based methods in the Eulerian frame, one has to deal with the advection term, which is nonlinear. In addition, it is hard to apply history dependent constitutive models. In the Lagrangian frame, one has to deal with mesh distortion

when the body undergoes large deformation, but the history dependent constitutive models can be easily applied. A third option, Arbitrary Lagrangian Eulerian Methods (ALE) introduce a reference domain, and the reference domain moves arbitrarily with respect to the movement of the materials. Because of the freedom of moving the mesh, the ALE approach potentially possesses the best properties of both the Lagrangian and Eulerian approaches.

In meshfree particle methods, one can also solve the equations of motion using an Eulerian approach, or using a Lagrangian approach. Such meshfree methods include Smooth Particle Hydrodynamics, Element Free Galerkin Methods, Reproducing Kernel Particle Methods and Partition of Unity Methods. The goal of meshfree particle methods is to solve the equations of motion without explicitly constructing a mesh which precludes problems with mesh distortion. However, some issues remain such as accurate domain integration for methods developed based on the Galerkin weak form, stable nodal integration for collocation methods, and accurate application of boundary conditions.

The material point method (MPM) was invented by Sulsky and her coworkers [29, 30, 32]. MPM is an extension of the particle in cell (PIC) method of Harlow [11]. It is also a particular type of ALE method, in which the equations of motion are solved on a background grid and the Lagrangian material points represent the geometry of the body, carrying the history-dependent material properties. MPM has been used widely in research and industry. It has been used to model sea ice dynamics [31], rock fractures [33], snow simulations [28] and cutting processes [3]. Recently, it has been used for movie special effects (Disney's *Frozen*) [28].

One of my research goals is to improve MPM in terms of convergence and order of accuracy for large deformation problems. Although MPM is able to handle a solid body that undergoes large deformation, it suffers low order convergence or no convergence for some large deformation problems. There are two main reasons for

the inaccuracy. First is the mapping from the material-point information to the grid information, which is first order. Second is the quadrature to approximate nodal forces or nodal masses. In my research, we constructed a new framework for improving the material-point method, combining ideas from meshfree methods and the classical finite element methods. Not only has our framework produced the improved material point method (IMPM), but it can also be used to improve existing variations of MPM, such as the Convected Particle Domain Interpolation Method (CPDI) [2].

The other goal in my research is to study the stability properties of MPM. It has been seen that all variations of MPM, including MPM, GIMP, CPDI and IMPM, appear to become unstable on fine meshes. Such instability is still under investigation. One possible reason for this instability is called the finite grid instability, which is a well known phenomenon in the particle in cell method [15, 27]. This instability is due to the fact that high frequency modes existing on the material points are mapped to the low frequency modes on the grid. The instability is analyzed by linearizing the equations of motion. In the original MPM, it is also shown by a nonlinear analysis that energy is bounded. Thus, the nonlinear terms have the potential to control the instability. It has also been demonstrated, [15, 27], that an implicit solver controls the instability. Another remedy that has been proposed is jiggling the grid position [7].

The general layout of this work is as follows. Chapter 2 introduces basic concepts and conservation equations in continuum mechanics. Chapter 3 introduces the background on numerical methods for solving the continuum equations. Such methods include finite element methods, and meshfree particle methods. The idea of function reconstruction from scattered data, which is the foundation for the meshfree particle methods, is discussed in detail in the multi-dimensional case with numerical illustrations in $1D$. Chapter 4 reviews the main structure of MPM. The mapping and the

quadrature parts of the MPM algorithm are discussed in detail. The convergence of MPM is studied numerically both for small deformation problems and large deformation problems for $1D$ and $2D$ cases, utilizing the idea of manufactured solutions. Chapter 5 introduces the improved MPM algorithm, in which the improved mapping and quadrature parts of the MPM algorithm are studied, utilizing the ideas of function reconstruction from scattered data introduced in Chapter 3. The improvements of the convergence of the MPM algorithm are shown numerically with the same examples as in the convergence study of the MPM algorithms in Chapter 4, which shows that the development identified as IMPM increases the convergence of MPM to second order. Chapter 6 reviews other improvements of the MPM algorithms from the literature, including GIMP and CPDI. Both GIMP and CPDI try to improve MPM by changing the geometry of the evolved material-point domain. Such changes lead to improvement of the quadrature part of the MPM algorithm; see [1, 2, 26, 24]. Based on the general framework of the IMPM algorithm, the CPDI algorithm is further improved by alterations to the mapping part. Convergence of CPDI, IMPM and improved CPDI are studied numerically in $1D$ with the same examples as in Chapter 4, in which the IMPM and improved CPDI give the best results. Chapter 7 provides a Neumann stability analysis of the original MPM algorithm by linearizing the equations of motion and discretizing the linearized equations analogously to MPM.

A general framework for improving MPM has been proposed by combining ideas of meshfree particle methods and finite element methods. Within the framework, the IMPM and improved CPDI are produced and they increase the order of convergence of the original MPM algorithm to second order for large deformation problems. Besides the improvements to the order of accuracy of the MPM algorithm, a Neumann stability analysis of MPM on the linearized equations of motion is provided.

# Chapter 2

# Summary of Continuum Equations

## 2.1 Introduction

In this chapter, the basic concepts of motion, configuration, and mass conservation are introduced, and the momentum balance equations are derived. Since thermodynamics is not taken into consideration for the time being, the conservation of energy is not discussed. When indicial notation is used, we also invoke the Einstein summation convention of summing over repeated indices.

## 2.2 Motion and Configuration

Consider a body $\Omega_0 \subset R^3$, at the time $t_0$ as the initial configuration, and let the current configuration of the body be $\Omega \subset R^3$, at the time $t$. Assume that an initial point $X \in \Omega_0$ moves to the current position $x \in \Omega$. The motion is represented by a function

$$x = x(X, t). \tag{2.1}$$

Note that the evolution in time of $x(X,t)$ of a specific point $X$ gives the trajectory of a material point. The velocity of the material point is calculated by

$$V(X,t) = \frac{\partial x}{\partial t}(X,t). \tag{2.2}$$

Since any two material points in the free body can never occupy the same position, (2.1) has a unique inverse

$$X = X(x,t), \tag{2.3}$$

which gives the initial position of the material point currently at $x$ at time $t$. The velocity in expression (2.2) also can be represented as a function of the current configuration and time

$$v(x,t) = V(X(x,t),t). \tag{2.4}$$

Based on the above definitions, a more general definition can be given. If a function $\Phi$ is represented in the initial coordinates $(X,t)$, it is referred as the Lagrangian description; and if a function $\phi$ is represented in the current coordinates $(x,t)$, it is referred as the Eulerian description. Although the representations are physically and mathematically equivalent, in practice, one representation might have an advantage over the other depending on the situation. For example, the Lagrangian description is commonly used in solid mechanics because it allows the labeling of the material points in a solid body, and each material point can be traced from the initial state to the current state for the purpose of applying history-dependent constitutive models. By contrast, the Eulerian description is commonly used in fluid mechanics because there is usually no history dependence in the constitutive models, and large deformations make the Lagrangian description inconvenient.

Let $\Phi(X,t)$ be a Lagrangian quantity (scalar, vector or tensor). Since $X$ is not dependent on time, the time differentiation of $\Phi$ is given by $\frac{\partial \Phi}{\partial t}$. On the other hand, if we represent $\Phi(X,t) = \Phi(X(x,t),t) = \phi(x,t)$ in Eulerian coordinates, the total

differential of $\phi(x,t)$ is

$$d\phi(x,t) = \frac{\partial \phi}{\partial t}dt + \frac{\partial \phi}{\partial x_i}dx_i. \tag{2.5}$$

Since $v_i(x,t) = V_i(X(x,t),t) = \frac{\partial x_i(X,t)}{\partial t}$, $i$ = 1,2,3, we can write the material time derivative in different ways

$$\frac{\partial \Phi}{\partial t} = \frac{\partial \phi}{\partial t} + \frac{\partial \phi}{\partial x_i}\frac{\partial x_i}{\partial t} = \frac{\partial \phi}{\partial t} + v \cdot \nabla \phi = \frac{d\phi}{dt}. \tag{2.6}$$

Here $\nabla$ is the gradient with respect to $x$. Remark: the gradient operator $\nabla$ or the divergence operator $\nabla\cdot$ operates on functions defined in the current configuration, and the gradient operator $\nabla_0$ or divergence operator $\nabla_0\cdot$ operates on functions defined in the initial configuration. The quantity $\dot{\phi} = \frac{d\phi}{dt}$ in (2.6) is defined as the material time derivative in Eulerian coordinates. Since the above functions $\phi$ and $\Phi$ can be any representative functions, the above result can be applied to the velocity to obtain expressions for the acceleration.

$$A(X,t) = \frac{\partial V}{\partial t} = \frac{\partial v}{\partial t} + v \cdot \nabla v = a(x,t). \tag{2.7}$$

## 2.3 Mass Conservation Law

Let $R(0)$ be an arbitrary material subregion in $\Omega_0$ with its subsequent image $R(t) \subset \Omega$. The total mass $M$ in the undeformed subregion $R(0)$ is conserved in the deformed subregion $R(t)$. Therefore

$$M = \int_{R(0)} \rho_0 \, dX = \int_{R(t)} \rho dx, \tag{2.8}$$

where $\rho_0 = \rho_0(X,t)$ and $\rho = \rho(x,t)$ are the densities in the Lagrangian and Eulerian coordinates respectively. Since $x = x(X,t)$, we have

$$\begin{aligned} dx_i &= \frac{\partial x_i}{\partial X_j}dX_j, \\ &= F_{ij}dX_j, \end{aligned} \tag{2.9}$$

where $F$ is the deformation gradient tensor and its components are $F_{ij} = \frac{\partial x_i}{\partial X_j}, i, j = 1, 2, 3$. Let $J$ be the Jacobian of $F$ (denoted by $J = |F|$). From relation (2.9), one can derive the following relation between the infinitesimal change of volume between Eulerian coordinates and Lagrangian coordinates

$$dx = J(X,t)dX. \tag{2.10}$$

Substitute (2.10) into (2.8) to derive

$$M = \int_{R(0)} \rho_0 \, dX = \int_{R(t)} \rho(x,t)dx = \int_{R(0)} \rho(x(X,t))J(X,t)dX.$$

From the last equality, we get

$$\int_{R(0)} (\rho_0 - \rho J)dX = 0.$$

Since $R(0)$ is an arbitrary material subregion of $\Omega_0$, and under the assumption that the integrand is continuous, we get

$$\begin{aligned} \rho_0 - \rho J &= 0, \\ J &= \frac{\rho_0}{\rho}. \end{aligned} \tag{2.11}$$

Now, define $R(X,t) = \rho(x(X,t),t)$, and take the time derivative of equation (2.8) to get

$$\begin{aligned} \frac{dM}{dt} &= \frac{d}{dt}\int_{R(t)} \rho(x)dx, \\ &= \frac{d}{dt}\int_{R(0)} \rho(x(X,t))J(X,t)dX, \\ &= \frac{d}{dt}\int_{R(0)} R(X,t)J(X,t)dX, \\ &= \int_{R(0)} \frac{\partial}{\partial t}(R(X,t)J(X,t))dX, \\ &= \int_{R(0)} \frac{\partial R(X,t)}{\partial t}J(X,t)dX + \int_{R(0)} R(X,t)\nabla \cdot v(x(X,t))J(X,t)dX, \\ &= \int_{R(t)} \frac{d\rho(x,t)}{dt}dx + \int_{R(t)} \rho(x,t)\nabla \cdot v(x,t)dx, \\ &= 0. \end{aligned} \tag{2.12}$$

The above uses the identity

$$\begin{aligned}\frac{\partial}{\partial t}J(X,t) &= \frac{\partial}{\partial t}(|\frac{\partial x}{\partial X}|),\\ &= (\nabla\cdot v)|\frac{\partial x}{\partial X}|,\\ &= \nabla\cdot v(x(X,t),t)J(X,t).\end{aligned}$$

A proof of the above identity is

$$\begin{aligned}\frac{\partial}{\partial t}J(X,t) &= \frac{\partial}{\partial t}(|\frac{\partial x}{\partial X}|),\\ &= \frac{\partial}{\partial t}(\epsilon_{ijk}\frac{\partial x_1}{\partial X_i}\frac{\partial x_2}{\partial X_j}\frac{\partial x_3}{\partial X_k}),\\ &= \epsilon_{ijk}(\frac{\partial v_1}{\partial x_1}\frac{\partial x_1}{\partial X_i}\frac{\partial x_2}{\partial X_j}\frac{\partial x_3}{\partial X_k}+\frac{\partial v_2}{\partial x_2}\frac{\partial x_1}{\partial X_i}\frac{\partial x_2}{\partial X_j}\frac{\partial x_3}{\partial X_k}+\frac{\partial v_3}{\partial x_3}\frac{\partial x_1}{\partial X_i}\frac{\partial x_2}{\partial X_j}\frac{\partial x_3}{\partial X_k}),\\ &= (\nabla\cdot v)J,\end{aligned}$$

where the third order tensor $\epsilon_{ijk}$ is defined as $\epsilon_{123}=\epsilon_{312}=\epsilon_{231}=1$ and $\epsilon_{321}=\epsilon_{132}=\epsilon_{213}=-1$.

If the integrand in equation (2.12) is continuous, since $R(t)$ is arbitrary, then in local form, we have derived the following mass conservation law in the Eulerian frame

$$\frac{d\rho}{dt}+\rho\nabla\cdot v=\frac{\partial\rho}{\partial t}+\nabla\cdot(\rho v)=0. \tag{2.13}$$

Note that if the material is incompressible, which means the material derivative of the density is zero (i.e $\frac{d\rho}{dt}=0$), the mass conservation law (2.13) can be written as

$$\nabla\cdot v(x,t)=0. \tag{2.14}$$

Note that the above mass conservation law is derived in the Eulerian coordinates,

while the mass conservation law in the Lagrangian coordinates is derived as follows

$$\begin{aligned}\frac{dM}{dt} &= \frac{d}{dt}\int_{R(0)} \rho_0(X,t)dX, \\ &= \int_{R(0)} \frac{\partial \rho_0(X,t)}{\partial t} dX, \\ &= 0. \end{aligned} \tag{2.15}$$

Again, since $R(0)$ is an arbitrary material subregion of $\Omega_0$ and under the assumption that the integrand is continuous, we get the local form as follows

$$\frac{\partial \rho_0(X,t)}{\partial t} = 0. \tag{2.16}$$

We can therefore write $\rho_0 = \rho_0(X)$. Utilizing the mass conservation law (2.13), for any function $\phi$, we have the following

$$\frac{d}{dt}\int_{R(t)} \rho\phi dx = \int_{R(t)} \rho \frac{d\phi}{dt} dx, \tag{2.17}$$

which is called the Reynold's transport theorem. The proof of this theorem is as follows

$$\begin{aligned}\frac{d}{dt}\int_{R(t)} \rho(x,t)\phi(x,t)dx &= \frac{d}{dt}\int_{R(0)} R(X,t)\Phi(X,t)J(X,t)dX, \\ &= \int_{R(0)} [\frac{\partial R}{\partial t}(\Phi J) + \frac{\partial \Phi}{\partial t}(RJ) + \frac{\partial J}{\partial t}(R\Phi)]dX, \\ &= \int_{R(0)} [\frac{\partial R}{\partial t}(\Phi J) + \frac{\partial \Phi}{\partial t}(RJ) + (\nabla \cdot v)J(R\Phi)]dX, \\ &= \int_{R(t)} [\frac{d\rho}{dt}\phi + \rho\frac{d\phi}{dt} + (\nabla \cdot v)\rho\phi]dx, \\ &= \int_{R(t)} [(\frac{d\rho}{dt} + (\nabla \cdot v)\rho)\phi + \rho\frac{d\phi}{dt}]dx, \\ &= \int_{R(t)} [0\phi + \rho\frac{d\phi}{dt}]dx, \\ &= \int_{R(t)} \rho\frac{d\phi}{dt}dx. \end{aligned}$$

## 2.4 Law of Conservation of Linear Momentum

**Conservation of linear momentum in Eulerian coordinates.** The total linear momentum in an arbitrary subregion $R(t)$ of $\Omega$ is given by

$$L = \int_{R(t)} \rho v dx. \tag{2.18}$$

If a force per unit area, called the traction $t$, acts on the boundary of the subregion denoted as $\partial R(t)$, with the body force per unit mass $b$ acting in the subregion $R(t)$, the total force is

$$\int_{\partial R(t)} t ds + \int_{R(t)} \rho b dx, \tag{2.19}$$

where $ds$ is the infinitesimal surface area on the boundary of the subregion $\partial R(t)$. By Newton's second law (i.e. the rate of change of linear momentum equals the net force),

$$\frac{d}{dt}\int_{R(t)} \rho v dx = \int_{\partial R(t)} t ds + \int_{R(t)} \rho b dx. \tag{2.20}$$

Here the fact that $t = \sigma n$ is stated without proof, where $n$ the unit outward normal to $\partial R(t)$ and $\sigma$ is the Cauchy stress tensor. Use the divergence theorem to write

$$\int_{\partial R(t)} t ds = \int_{\partial R(t)} \sigma n ds = \int_{R(t)} \nabla \cdot \sigma dx. \tag{2.21}$$

Utilizing the Reynold's transport theorem, equation (2.17) and (2.21) are combined in equation (2.20) to get

$$\int_{R(t)} [\rho \frac{dv}{dt} - \nabla \cdot \sigma - \rho b] dx = 0. \tag{2.22}$$

Since $R(t)$ is an arbitrary subdomain of $\Omega$ and the assumption is made that the integrand in (2.22) is continuous, we get the local form of the momentum equation

$$\nabla \cdot \sigma + \rho b = \rho a. \tag{2.23}$$

By conservation of angular momentum, we can get that the stress is symmetric (i.e: $\sigma = \sigma^T$). Here the fact is stated without proving the symmetry of stress.

**Conservation of linear momentum in Lagrangian coordinates.** Linear momentum of an arbitrary subregion $R(0)$ in $\Omega_0$ is given by

$$L = \int_{R(0)} \rho_0 V dX. \tag{2.24}$$

If a force per unit area, called the traction $T$, acts on the boundary of the subregion denoted as $\partial R(0)$, with the body force per unit mass $B$ acting in the subregion $R(0)$, the total force is

$$\int_{\partial R(0)} T dS + \int_{R(0)} \rho_0 B dX, \tag{2.25}$$

where $dS$ is the infinitesimal surface area on the boundary of the subregion $\partial R(0)$. By Newton's second law,

$$\frac{d}{dt}\int_{R(0)} \rho_0 V dX = \int_{\partial R(0)} T dS + \int_{R(0)} \rho_0 B dX. \tag{2.26}$$

Here the fact is stated without proof that $T = PN$, $N$ is the unit outward normal to $\partial R(0)$ and $P$ is the first Piola-Kirchhoff stress tensor. Use the divergence theorem to get

$$\int_{\partial R(0)} T dS = \int_{\partial R(0)} PN dS = \int_{R(0)} \nabla_0 \cdot P dX, \tag{2.27}$$

Since the subregion $R(0)$ is independent of time, the order of differentiation and integration can be interchanged, and with (2.27) substituted in (2.26), we obtain

$$\int_{R(0)} [\rho_0 \frac{\partial V}{\partial t} - \nabla_0 \cdot P - \rho_0 B] dX = 0. \tag{2.28}$$

Since $R(0)$ is an arbitrary subdomain of $\Omega_0$ and the assumption is made that the integrand in (2.28) is continuous, we obtain

$$\nabla_0 \cdot P + \rho_0 B = \rho_0 A. \tag{2.29}$$

If the first Piola-Kirchhoff stress tensor is used, the stresses in the initial configuration and current configuration are related as follows

$$PF^T = J\sigma,$$

where $F^T$ is the transpose of $F$.

In short, the momentum equation can either be expressed in the initial configuration or the current configuration. Conservation of mass and linear momentum in Eulerian coordinates are

$$\frac{d\rho}{dt} + \rho\nabla \cdot v = \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho v) = 0,$$
$$\rho a = \nabla \cdot \sigma + \rho b. \tag{2.30}$$

Conservation of mass is given by the algebraic relation $\rho = \frac{\rho_0}{J}$ in the Lagrangian coordinates and conservation of linear momentum in Lagrangian coordinates is

$$\rho_0 A = \nabla_0 \cdot P + \rho_0 B. \tag{2.31}$$

**Generalization of conservation equations to arbitrary coordinates** One can also derive the conservation equations defined on an arbitrary reference coordinate $\chi$. These equations are briefly summarized as follows without proof. Let $\chi$ be the reference coordinates, $X$ the coordinates of initial configuration and $x$ the coordinates of current configuration. Conservation of mass and momentum in the Arbitrary Lagrangian Eulerian (ALE) representation are

$$\frac{\partial \rho}{\partial t}|_\chi + c \cdot \nabla \rho = -\rho \nabla \cdot v, \tag{2.32}$$
$$\rho(\frac{\partial v}{\partial t}|_\chi + (c \cdot \nabla)v) = \nabla \cdot \sigma + \rho b, \tag{2.33}$$

where $c = v(x,t) - \hat{v}(\chi,t)$, $\hat{v}(\chi,t) = \frac{\partial x}{\partial t}|_\chi$ and $\frac{\partial \phi}{\partial t}|_\chi = \frac{\partial \phi(x(\chi,t),t)}{\partial t}$ for any function $\phi(x,t)$. The gradient operator $\nabla$ and divergence operator $\nabla\cdot$ are taken with respect to the current configuration. Note that if the reference configuration is the initial configuration (i.e $\chi = X$), the conservation equations reduce to the Lagrangian form, and if the reference configuration is the current configuration, the conservation equations reduce to the Eulerian form. See [23].

**Conservation equations in weak form** Soppose equations (2.30) and (2.31) are multiplied by the test functions $\delta u$ and $\delta U$ respectively, integration by parts is performed and the divergence theorem is utilized, the weak form of the conservation equation in the Eulerian coordinates is

$$\int_{\Omega} [\rho \delta u \cdot a + \sigma : \nabla \delta u] dx = \int_{\Omega} \rho \delta u \cdot b dx + \int_{\partial \Omega^{\tau}} \delta u \cdot \overline{\tau} ds. \tag{2.34}$$

The weak form of the conservation equation in the Lagrangian coordinates is

$$\int_{\Omega_0} [\rho_0 \delta U \cdot A + P : \nabla \delta U] dX = \int_{\Omega_0} \rho_0 \delta U \cdot B dX + \int_{\partial \Omega_0^T} \delta U \cdot \overline{T} dS. \tag{2.35}$$

In the above weak forms of the conservation equations, $\overline{\tau}$ and $\overline{T}$ are applied tractions. $\partial\Omega$ is decomposed as $\partial\Omega = \partial\Omega^v \cup \partial\Omega^{\tau}$ with $\partial\Omega^{\tau} \cap \partial\Omega^v = \emptyset$. Here $\partial\Omega^v$ is the part of the boundary where the velocity boundary condition is prescribed and $\partial\Omega^{\tau}$ is the part of the boundary where the traction boundary condition is prescribed. The momentum equation alone is not sufficient to describe a well-posed problem, we also need a constitutive model, which relates the stress and strain, or stress and deformation gradient to describe a complete system of differential equations. Such a relation will be given in the following sections. Furthermore, appropriate initial conditions and boundary conditions are also needed. These will be given for specific problems in later sections.

# Chapter 3

# Background on Numerical Methods

## 3.1 Introduction

In this chapter, some popular numerical methods for solving the conservation equations are reviewed. Such numerical methods can be classified into two categories. One is mesh dependent methods, such as finite element methods, the other one is mesh independent methods, for example, meshfree particle methods. The main difference between finite element methods and meshfree methods is in finite element methods, the equations are solved on a mesh, where the nodes are connected, while in meshfree methods, the equations are solved on the nodes with no implied connectivity between the nodes. When developed using the Galerkin weak form, both the finite element method and meshfree particle methods rely on representing the numerical solution in terms of a finite set of basis functions. A main ingredient of meshfree methods is construction of the basis functions. The basis function construction involves function reconstruction from scattered data, which not only constitutes

the basis for meshfree methods, but is also critical for understanding and improving MPM. In this chapter, finite element methods and meshfree methods are summarized, and the details of how to construct functions from scattered data are discussed, using the ideas of meshfree particle methods.

## 3.2 Finite Element Method

A popular method for solving the equations of motion numerically is the finite element method (FEM). In FEM, an approximate solution is constructed from a basis in a finite dimensional Sobolev space, with a mesh used to discretize the body. For example, assume $x_i$ is the location of ith node on the mesh, $i = 1, 2, \cdots, n$, where $n$ is the total number of nodes. If the displacement $u$ is the unknown variable of interest, and $P_i(x), i = 1, 2, \cdots, n$ is a piecewise polynomial basis in the corresponding finite dimensional Sobolev space, the approximate function $u^h$ has the following form

$$u^h(x) = \sum_{i=1}^{n} P_i(x) u_i,$$

where $u_i$ are the coefficients of the basis functions defined on the node $x_i$. The above approximate function is inserted into the weak form of the equations of motion to solve for the values $u_i$.

The equations of motion can either be solved based on a Lagrangian frame or an Eulerian frame using FEM. For the Eulerian approach, the equations of motion can usually be solved on a uniform grid and grid generation becomes relatively easy. However the Eulerian approach has defects. Because the material flows through the fixed grid, information about the material boundaries is lost, especially for large deformations. Additionally, the Eulerian approach introduces difficulty in keeping track of the material deformation history and dealing with the nonlinear advection term. By contrast, the Lagrangian approach allows the user to keep track of the

histories of the deformation of the material but the defect is that mesh distortion becomes severe for large deformation problems. The presence of small or distorted elements also puts restrictions on the size of the time step in explicit time-stepping algorithms.

Due to the above issues of both the Lagrangian approach and the Eulerian approach, the Arbitrary Lagrangian Eulerian method (ALE) has been developed. This method utilizes the advantages of both the Lagrangian and Eulerian approaches. ALE introduces a reference domain which moves arbitrarily with respect to movement of the material. Accordingly, the conservation equations for mass and momentum must be described in the reference configuration. In practice, one of the possible approaches in ALE is the ability to solve the equations of motion in a Lagrangian frame with an additional rezoning step. The rezoning step requires one to create a new mesh based on the boundaries of the old mesh, and one has to map the information from the old mesh to the new mesh. As a result of this rezoning step, the mesh entanglement is avoided and the constraint on the time step is relaxed correspondingly, so that a small time step is avoided. The drawbacks of ALE are the new mesh generation and numerical dissipation for the rezoning step. See [9, 12].

## 3.3 Overview of Meshfree Methods

The general goal of meshfree methods is to solve the equations of motion without explicitly constructing a mesh. The philosophy of meshfree methods is similar to the finite element method, in which the generic variables or functions are projected and the equations of motion are solved in a finite dimensional space. For example, if the displacement $u$ is the unknown variable of interest and $\phi_i, i = 1, 2, \cdots, n$ are the basis in the corresponding Sobolev space, then the approximate function $u^h$ has

the following form,

$$u^h(x) = \sum_{i=1}^{n} \phi_i(x) u_i,$$

where $u_i$ are the coefficients of the basis functions and $n$ is the maximal number of the basis functions in the Sobolev space. The above approximate function is inserted into the weak form of the equations of motion to solve for the values $u_i$, or a collocation method is used to solve the strong form of the equations at selected points. Many meshfree methods have been developed. Such meshfree methods include Smooth Particle Hydrodynamics (SPH) [17, 25], Element Free Galerkin Methods (EFG) [4, 5], Reproducing Kernel Particle Methods (RKPM) [19, 20] and Partition of Unity Methods (PU) [14].

In SPH, the approximate function is $u^h(x) = \int_\Omega C_h w(x-y,h) u(y) d\Omega_y$, where $w(x-y,h)$ is a kernel or a weight function. In practice, $w$ is given by splines. The parameter $h$ measures the support of the weight function, and $C_h$ is a normalization constant such that $\int_\Omega C_h w(x-y)dy = 1$. Note that this is the current way in which SPH is formulated (see [13]), but the original form did not contain the normalization which restricted accuracy. In discrete form, $u^h(x) = \sum_i \phi_i(x) u_i = \sum_i C_h w(x - x_i) u_i V_i$, where $\phi_i(x) = C_h w(x-x_i)V_i$, $u_i$ is the coefficient, $V_i$ is the volume associated with node $i$, $i = 1, 2, \cdots, n$, and $n$ is the total number of nodes. It turns out that the shape functions constructed above can reproduce polynomials of arbitrary order by choosing the right normalization. The definition of reproducing conditions will be given in the following sections. Equations for the unknowns $u_i$ are generally determined using a collocation method.

In RKPM, one constructs an approximation of a function $u(x)$ through the use of an integral kernel, $k(x,s)$, such that $u^h(x) = \int_\Omega k(x,s)u(s)ds$. The integral is then approximated by choosing a set of nodes, $x_i \subset \Omega$, so that $u^h(x) = \sum_i k(x,x_i)u_i V_i = \sum_i \phi_i(x) u_i$, where $\phi_i(x) = k(x,x_i)V_i$. The discrete kernel has the form $k(x,x_i) = \alpha(x)^T P(x) w(x-x_i)$. Here $u_i$ is the coefficient, $V_i$ is the volume associated with node

$i$, $i = 1, 2, \cdots, n$, and $n$ is the total number of nodes. The shape functions above satisfy the reproducing conditions to arbitrary order if arbitrary order of polynomials $P(x)$ are used. In this representation, let $\alpha(x) = [\alpha_1(x), \alpha_2(x), \cdots, \alpha_m(x)]$, where $m$ is the number of terms in the polynomial basis and $\alpha(x)^T$ is the transpose of $\alpha(x)$. By enforcing the reproducing conditions on $\phi_i$, one can solve for the coefficients $\alpha(x)$. Note that such shape functions have compact support since the weight function has compact support. The details of the construction of such shape functions will be given in the following sections. Note that the equations for the unknowns $u_i$ are usually determined using Galerkin methods.

In EFG, the assumption is made that the approximate function has the form $u^h(x, \bar{x}) = \alpha(x)^T P(\bar{x})$, where $\alpha(x)$ and $P(\bar{x})$ are defined as above. The shape function $\phi_i(x)$ is constructed by minimizing the following functional

$$\begin{aligned} I(x) &= \sum_{i=1}^{n} w(x - x_i)(u^h - u_i)^2, \\ &= \sum_{i=1}^{n} w(x - x_i)(\alpha(x)P(x_i) - u_i)^2, \end{aligned}$$

where $w(x - x_i)$ is the same as above. $\alpha(x)$ is solved for by minimizing the above functional, and inserting the expression for $\alpha(x)$ back into $u^h(x) = \alpha(x)^T P(\bar{x})$ to get the shape function $\phi_i(x)$. It turns out that RKPM and EFG are equivalent. In EFG, the discrete equations of motion are typically developed by the Galerkin conditions as in RKPM.

In the meshfree methods described above, the equations of motion can be solved either in the Lagrangian form or the Eulerian form, using a Galerkin method or a collocation method. In the Galerkin method, the equations of motion are derived in the weak form, and are projected into a finite dimensional space. In the collocation method, a set of points is chosen so that the equations of motion in strong form are satisfied at these points. SPH usually formulates the equations of motion in Eulerian form using collocation, and EFG or RKPM are formulated either in Lagrangian

form or Eulerian form using a Galerkin method. In the Lagrangian formulation of the equations of motion, the weight function $w$ and the basis function $\phi_i$ described above are defined in the initial configuration, $w = w(X - X_i)$. Correspondingly, they are defined once initially, and the support of the weight function $w(X - X_i)$ changes with the dynamics. In the Eulerian formulation of the equations of motion, the weight function $w$ and the basis function $\phi_i$ are defined in the current configuration, $w = w(x - x_i)$. Correspondingly, they need to be constructed at each time step, and usually the support of the weight function is constant for all time. Note that if the internal forces are approximated by nodal quadrature in EFG or RKPM, then EFG or RKPM reduce to SPH with collocation.

There are three main stability issues in the above meshfree methods. The first is the material instability in continua, which is related to the ill-possedness of the PDE. This type of instability is independent of the numerical methods chosen. The second stability issue is the tensile instability, which happens only in the Eulerian formulation of equations of motion in any of SPH, EFG and RKPM. The tensile instability is mainly due to the positive stress field and the negative value of the second derivative of the basis function. The third type of instability results from the rank deficiency of the discrete divergence operator on the stress field. The rank deficiency occurs in both Lagrangian kernels and Eulerian kernels. Such rank deficiency can be eliminated by using a different quadrature rule to approximate the nodal internal forces or the divergence of stress field, but the tensile instability cannot be eliminated in the Eulerian kernel. Therefore the most stable meshfree methods are based on the Lagrangian formulation of the equations of motion. See [6] for more information about the stability of meshfree methods.

There are some drawbacks of the meshfree methods. Although the goal of meshfree methods is to solve the equations of motion without explicitly generating a mesh, these methods do need a mesh for quadrature in practice. Additionally, the

construction of the shape functions and their derivatives are numerically expensive to implement. For example, the inversion of a matrix is required to formulate the shape functions. For a second order accurate approximation, a 2 by 2 matrix has to be inverted in 1D, a 3 by 3 matrix has to be inverted in 2D, and a 4 by 4 matrix has to be inverted in 3D at each node. Moreover, the derivatives of the corresponding shape functions are computationally expensive to construct because of the matrix inversion process. In the Lagrangian formulation of equations of motion, the shape functions and derivatives of the shape functions can be constructed once initially, and a background mesh for quadrature is also defined once initially in order to exclude the rank deficiency. By contrast, the Eulerian formulation requires construction of the shape functions, the derivatives of the shape functions and a background mesh for quadrature at each time step.

Although the meshfree methods in the Lagrangian approach are demonstrated to be the most stable methods, they suffer substantial distortion of the shape functions for large deformation problems, which puts a restrictive constraint on the time step for an explicit algorithm. Whether the large distortion of the shape function leads to stability issues or convergence issues is not clear. The meshfree methods in the Eulerian approach sometimes are advantageous; for example, they do not suffer large distortion of shape functions because the kernels and shape functions are defined in the current configuration. Additionally, the Eulerian formulation puts less restrictive constraints on the time step for an explicit algorithm.

## 3.4 Function Reconstruction from Scattered Data

In order to illustrate the process of constructing the above basis functions in the multi-dimensional case, a multi-index notation is introduced. The notation $\mathbb{P}_r = \mathbb{P}_r(\Omega)$ denotes the space of polynomials of a degree less than or equal to $r$ on $\Omega$. The

dimension of $\mathbb{P}_r$ is $N_r$,

$$N_r = \binom{r+d}{d} = \frac{(r+d)!}{r!d!}.$$

For any point $x \in R^d$, the monomial in $R^d$ is $x^\alpha = (\tilde{x}_1)^{\alpha_1}(\tilde{x}_2)^{\alpha_2}\cdots(\tilde{x}_d)^{\alpha_d}$, where $\alpha = (\alpha_1, \alpha_2, \cdots, \alpha_d)$ and $\tilde{x}_i$ is the $ith$ coordinate in $R^d$. The length of $\alpha$ is $|\alpha| = \sum_{i=1}^{d} \alpha_i$ with $|\alpha| \leq r$. Given data, $u_p$, $p = 1, 2, \cdots, npt$ at points $x_p$, Assume the reconstructed function has the following form:

$$u^h(x) = \sum_{p=1}^{npt} \Phi_p^r(x) u_p, \tag{3.1}$$

where the basis functions $\Phi_p^r$ satisfy the following polynomial reproducing conditions

$$x^\alpha = \sum_{p=1}^{npt} \Phi_p^r(x) x_p^\alpha. \tag{3.2}$$

These are the reproducing conditions mentioned in the previous section used in deriving basis functions for SPH and RKPM. In the meshfree particle methods, the basis functions $\Phi_p^r(x)$, which satisfy the above polynomial reproducing conditions, are assumed to have the following form:

$$\Phi_p^r(x) = w(x - x_p) \sum_{\alpha \leq r} (x - x_p)^\alpha a_\alpha(x), \quad p = 1, 2, \cdots, npt, \tag{3.3}$$

where $w(x - x_p)$ is a weight function with compact support. Based on the reproducing conditions (3.2), there are $N_r$ equations that need to be solved for the coefficients $a_\alpha$.

$$\sum_{|\alpha| \leq r} m_{\alpha+\beta}(x) a_\alpha(x) = \delta_{|\beta|,0}, \quad |\beta| \leq r, \tag{3.4}$$

where the moment functions $m_\alpha$ are defined as

$$m_\alpha = \sum_{p=1}^{npt} w(x - x_p)(x - x_p)^\alpha. \tag{3.5}$$

The above system of equations can be written in the matrix form as

$$M(x)a(x) = b(0), \tag{3.6}$$

where the moment matrix is

$$M(x) = \sum_{p=1}^{npt} w(x - x_p)b(x - x_p)b(x - x_p)^T, \tag{3.7}$$

and $b(x) = (x^\alpha)_{|\alpha|\leq r} \in R^{N_r}$. Invert the moment matrix $M$ to obtain the coefficients $a$ from (3.6), and substitute the expression for the coefficients $a$ back into equation (3.3). The final form of the basis functions $\Phi_p^r(x)$ are derived as

$$\Phi_p^r(x) = w(x - x_p)b^T(x - x_p)M^{-1}(x)b(x - x_p). \tag{3.8}$$

## 3.5 Function Reconstruction Using Moving Least Squares

In this section, the function reconstruction from scattered data is presented using the idea of Moving Least Squares (MLS) with an example in $1D$ used as an illustration. The reproducing condition of MLS basis functions is proved using the $1D$ example. The idea of MLS is to construct basis functions from polynomials with variable coefficients. Assume a function has the following form

$$u^h(x, \bar{x}) = a_0(\bar{x}) + a_1(\bar{x})x + \cdots + a_r(\bar{x})x^r,$$

or

$$u^h(x, \bar{x}) = P^T(x)a(\bar{x}), \tag{3.9}$$

where $P(x)$ is a vector, $\mathbb{P}_r$ is a space of polynomials and the approximation function $u^h(x, \bar{x})$ has two parameters, $x$ and $\bar{x}$. In $1D$, $P(x)$ and $a(x)$ are defined as

$$P^T(x) = [1, x, x^2, \cdots, x^r],$$
$$a^T(x) = [a_0(x), a_1(x), \cdots, a_r(x)].$$

Here capital $P$ is used as a vector for the basis of polynomials and $p$ as the subindex for indicating point values. The general idea of MLS is to find $a(\bar{x})$, which depends on a local parameter $\bar{x}$ so that the approximation function $u^h(x, \bar{x})$ minimizes the following functional with given data $u_p$. The position $x_p$ of the given data $u_p$ is locally close to the point of interest $\bar{x}$. The functional is

$$\begin{aligned} I(a) &= \sum_{p=1}^{npt} w(\bar{x} - x_p)[u_p^h(x_p, \bar{x}) - u_p]^2, \\ &= \sum_{p=1}^{npt} w(\bar{x} - x_p)[P^T(x_p)a(\bar{x}) - u_p]^2. \end{aligned} \tag{3.10}$$

Since the point of interest $\bar{x}$ is arbitrary, we take $\bar{x} = x$ so that

$$\begin{aligned} I(a) &= \sum_{p=1}^{npt} w(x - x_p)[u_p^h(x_p, x) - u_p]^2, \\ &= \sum_{p=1}^{npt} w(x - x_p)[P^T(x_p)a(x) - u_p]^2. \end{aligned} \tag{3.11}$$

Take the derivative of $I$ with respect to each component of $a$ and set it to zero to get

$$\frac{\delta I(a)}{\delta a} = 2\sum_{p=1}^{npy} w(x - x_p)[P^T(x_p)a(x) - u_p]P(x_p) = 0.$$

Further we get

$$M(x)a(x) = B(x)U, \tag{3.12}$$

where $M$, $B$ and $U$ are defined as follows:

$$M = \sum_{p=1}^{npt} w(x - x_p)P(x_p)P^T(x_p), \tag{3.13}$$

$$B(x) = \sum_{p=1}^{npt} w(x - x_p)P(x_p), \tag{3.14}$$

$$U^T = [u_1, u_2, \cdots, u_{npt}]. \tag{3.15}$$

Invert $M$ to get

$$a(x) = M^{-1}(x)B(x)U.$$

Put the above expression for $a$ back into equation (3.9) to get

$$\begin{aligned} u^h(x) &= P^T(x)M^{-1}(x)B(x)U, \\ &= \sum_{p=1}^{npt} \Phi_p(x)u_p. \end{aligned} \tag{3.16}$$

where $\Phi_p(x) = P^T(x)M^{-1}(x)B_p(x)$ and $B_p(x) = w(x - x_p)P(x_p)$. If linear polynomials are used as an example, $P^T(x) = [1, \tilde{x}]$ in $1D$, and the form for $M(x)$ and $B(x)$ would therefore be

$$\begin{aligned} M(x) &= \sum_{p=1}^{npt} w(x - x_p)P(x_p)P^T(x_p), \\ &= w(\tilde{x} - \tilde{x}_1) \begin{bmatrix} 1 & \tilde{x}_1 \\ \tilde{x}_1 & \tilde{x}_1^2 \end{bmatrix} + \cdots + w(\tilde{x} - \tilde{x}_{npt}) \begin{bmatrix} 1 & \tilde{x}_{npt} \\ \tilde{x}_{npt} & \tilde{x}_{npt}^2 \end{bmatrix}, \end{aligned} \tag{3.17}$$

$$\begin{aligned} B(x)U &= \sum_{p=1}^{npt} w(x - x_p)P(x_p) \cdot U, \\ &= \begin{bmatrix} B_1 & B_2 & \cdots & B_{npt} \end{bmatrix} \cdot U, \\ &= \begin{bmatrix} w(\tilde{x} - \tilde{x}_1) \begin{bmatrix} 1 \\ \tilde{x}_1 \end{bmatrix}, & \cdots & , w(\tilde{x} - \tilde{x}_{npt}) \begin{bmatrix} 1 \\ \tilde{x}_{npt} \end{bmatrix} \end{bmatrix} \cdot \begin{bmatrix} u_1 \\ u_2 \\ \cdots \\ u_{npt} \end{bmatrix}. \end{aligned} \tag{3.18}$$

In $2D$, let $P^T(x) = [1, \tilde{x}^1, \tilde{x}^2]$, the form for $M(x)$ and $B(x)$ would therefore be

$$
\begin{aligned}
M(x) &= \sum_{p=1}^{npt} w(x - x_p) P(x_p) P^T(x_p), \qquad (3.19)\\
&= w(\tilde{x}^1 - \tilde{x}_1^1) \cdot w(\tilde{x}^2 - \tilde{x}_1^2) \begin{bmatrix} 1 & \tilde{x}_1^1 & \tilde{x}_1^2 \\ \tilde{x}_1^2 & (\tilde{x}_1^1)^2 & \tilde{x}_1^1 \tilde{x}_1^2 \\ \tilde{x}_1^2 & \tilde{x}_1^1 \tilde{x}_1^2 & (\tilde{x}_1^2)^2 \end{bmatrix} + \cdots \\
&+ w(\tilde{x}^1 - \tilde{x}_{npt}^1) \cdot w(\tilde{x}^2 - \tilde{x}_{npt}^2) \begin{bmatrix} 1 & \tilde{x}_{npt}^1 & \tilde{x}_{npt}^2 \\ \tilde{x}_{npt}^1 & (\tilde{x}_{npt}^1)^2 & \tilde{x}_{npt}^1 \tilde{x}_{npt}^2 \\ \tilde{x}_{npt}^2 & \tilde{x}_{npt}^1 \tilde{x}_{npt}^2 & (\tilde{x}_{npt}^2)^2 \end{bmatrix},
\end{aligned}
$$

$$
\begin{aligned}
B(x)U &= \sum_{p=1}^{npt} w(x - x_p) P(x_p) \cdot U, \qquad (3.20)\\
&= \begin{bmatrix} B_1 & B_2 & \cdots & B_{npt} \end{bmatrix}, \\
&= \begin{bmatrix} w(x - x_1) \begin{bmatrix} 1 \\ \tilde{x}_1^1 \\ \tilde{x}_1^2 \end{bmatrix}, \cdots, w(x - x_{npt}) \begin{bmatrix} 1 \\ \tilde{x}_{npt}^1 \\ \tilde{x}_{npt}^2 \end{bmatrix} \end{bmatrix} \cdot \begin{bmatrix} u_1 \\ u_2 \\ \cdots \\ u_{npt} \end{bmatrix},
\end{aligned}
$$

where $w(x - x_i) = w(\tilde{x}^1 - \tilde{x}_i^1) \cdot w(\tilde{x}^2 - \tilde{x}_i^2)$, $i = 1, 2, \cdots, npt$.

Remark: MLS reproduces functions up to order $r$ if rth order polynomials are chosen as the basis. The proof of reproducing conditions of MLS basis functions is illustrated in $1D$ as follows.

Definition: a set of basis functions satisfies polynomial reproducing conditions of

order $r$ if and only if the following are true

$$\begin{aligned}
&\sum_{p=1}^{npt} \Phi_p(x) \cdot 1 = 1, \\
&\sum_{p=1}^{npt} \Phi_p(x) \cdot x_p = x, \\
&\sum_{p=1}^{npt} \Phi_p(x) \cdot x_p^2 = x^2, \\
&\qquad\qquad \vdots \\
&\sum_{p=1}^{npt} \Phi_p(x) \cdot x_p^r = x^r.
\end{aligned} \tag{3.21}$$

Now let us study the reproducing conditions of the basis functions constructed from MLS. Let

$$\begin{aligned}
P^T(x) &= \left[1, x, x^2, \cdots, x^r\right], \\
a^T(x) &= \left[a_0(x), a_1(x), a_2(x), \cdots, a_r(x)\right].
\end{aligned}$$

Since $\Phi_p(x) = P^T(x)M^{-1}(x)B_p(x)$, where $M$ and $B$ are given by (3.17) and (3.18), we derive the following

$$\begin{aligned}
\sum_{p=1}^{npt} \Phi_p \left[1, x_p, x_p^2, \cdots, x_p^r\right] &= \sum_{p=1}^{npt} P^T(x)M^{-1}(x)B_p(x) \left[1, x_p, x_p^2, \cdots, x_p^r\right], \\
&= \sum_{p=1}^{npt} P^T(x)M^{-1}(x)B_p(x)P^T(x_p), \\
&= \sum_{p=1}^{npt} P^T(x)M^{-1}(x)w(x - x_p)P(x_p)P^T(x_p), \\
&= P^T(x)M^{-1}(x) \sum_{p=1}^{npt} w(x - x_p)P(x_p)P^T(x_p), \\
&= P^T(x)M^{-1}(x)M(x), \\
&= P^T(x).
\end{aligned}$$

In the above proof, the definition for $B_p = w(x - x_p)P(x_p)$ and the equation of (3.17) for $M$ were used. Note that the factor $P^T(x)M^{-1}(x)$ can be factored out of the sum because it is a quantity independent of the parameter $x_p$. The reproducing condition (i.e equation(3.21)) has been obtained. It can be observed from the above calculation that the reproducing property does not depend on the choice of the weight functions, but the smoothness of the shape function does depend on the smoothness of the weight function, see [16]. In other words, the shape function $\Phi_p$ inherits the smoothness of the weight function $w$.

# Chapter 4

# The Original Material-Point Method

## 4.1 Introduction

The material point method (MPM) was invented by Sulsky, Chen and Schreyer (1994). It is an extension of the particle in cell (PIC) method of Harlow [8]. It is also a particular type of ALE method, in which Lagrangian material points represent the geometry of the body and carry the history-dependent material properties with a background grid used to solve the equations of motion.

Compared to the above meshfree methods, MPM has the advantage of efficiently using finite element shape functions to discretize the momentum equations, with the material points used to represent the geometry of the body. More specifically, MPM can use an arbitrary grid and classical finite element shape functions to discretize the equations of motion while the material points follow trajectories. The equations of motion are solved in an updated Lagrangian frame, and the material derivative becomes the total derivative so that the convection term does not appear in the

formulation. The disadvantage of MPM is that it needs a special algorithm to handle the interplay between the grid information and the material-point information, which requires slightly more work in this aspect compared with classical finite element methods. For details of a comparison study of MPM and SPH, see [22]. Remark: the main difference between MPM and meshfree particle methods (SPH, EFG and RKPM) is that in MPM, the equations of motion are solved on the background grid and material points are used as the quadrature points to approximate nodal forces and nodal masses, while in meshfree methods the equations of motion are solved on the material points and a background mesh is constructed for quadrature purposes to approximate the nodal forces.

The issue of MPM is that it has low order convergence or even no convergence for some large deformation problems. This issue has necessitated the development of an improved material point method (IMPM) for increasing the order of accuracy of original MPM for large deformation problems. In MPM, the low order accuracy is mainly due to two reasons. The first of these is the mapping from the material-point field to the grid field. The mapping formula in MPM is first order for large deformation problems, but almost second order for small deformation problems. For a small deformation problem, the material points do not move excessively. If the material points are initially equally spaced, they are almost equally spaced throughout the computation, then the mapping formula is almost second order. See Figure 4.1, 4.2 and 4.3 for illustration in the following sections. However for a large deformation problem, the material points can have large displacements, so the corresponding mapping formula in MPM reduces to first order. See Figure 4.4, 4.5 and 4.6 in the following sections for illustration.

The other source of error is the integration scheme which we use to approximate the nodal forces and nodal masses. In MPM, material points are used as the integration points. Since the material points are moving, their positions can be arbitrary in

the computational cell, and then the accuracy of corresponding nodal forces is low order.

The next sections review the original MPM formulation and its convergence properties.

## 4.2 Equations Solved by MPM

An example of the equations solved by MPM (i.e. the momentum equation plus the constitutive equation and strain relation) is

$$\nabla \cdot \sigma + \rho b = \rho \frac{dv}{dt}, \tag{4.1}$$

$$\frac{dF}{dt} = \nabla v \cdot F, \tag{4.2}$$

$$\rho = \frac{\rho_0}{J}, \tag{4.3}$$

$$\sigma = \sigma(F), \tag{4.4}$$

where $F$ is the deformation gradient tensor and $J$ is the Jacobian of $F$. For example, for the Neo-Hookean model, the stress is given as

$$\sigma = \frac{1}{J}[\lambda(\ln J)I + \mu(FF^T - I)], \tag{4.5}$$

where $\lambda$ and $\mu$ are material constants. Other constitutive models also can be treated. These equations are solved in the domain $\Omega$ with initial conditions

$$v(x, 0) = v_0, x \in \Omega_0$$
$$\sigma(x, 0) = \sigma_0(x).$$

Boundary conditions must also be specified and are given on the velocity or traction.

The philosophy of MPM is to solve the equations of motion on a background grid and to keep track of trajectories of a set of material points, where the material points

represent the geometry of the body $\Omega$ during deformation. The process is accomplished using the mapping between the information on the grid and the information on the material points. The details of this process are given in the following sections.

## 4.3 Main Steps of MPM

A body $\Omega_0$ is discretized into a finite set of $npt$ material points with positions $x_p, p = 1, 2, \cdots, npt$. Each of these points represents a material volume $\Omega_p$ with a mass $M_p$. Mass conservation requires that $M_p$ be constant. The initial discretization determines $x_p(0)$ and $\Omega_p(0)$, whereas $v_p(0)$ and $\sigma_p(0)$ come from the initial conditions. The aim is to determine the velocity, $v_p$, stress, $\sigma_p$ and displacement $u_p$ at later times. The MPM algorithm for each time step consists of four steps:

(1) Map information from material points to the background grid (i.e reconstruct functions from scattered data).

(2) Solve the momentum equation on the background grid.

(3) Update information on the material points.

(4) Generate a new grid.

### 4.3.1 Reconstruct a function from scattered data

In the MPM algorithm, mapping the information from the material points to the grid is an important step. It involves reconstructing a function from scattered data on the material points and evaluating the reconstructed function on the background grid. The reconstructed function in the MPM algorithm is actually a particular case of Shepard function interpolation. The Shepard function interpolation constructs

$u^h(x)$ as

$$u^h(x) = \sum_{p=1}^{npt} \phi_p(x) u_p, \tag{4.6}$$

where the shape function, $\phi_p(x)$ is

$$\phi_p(x) = \frac{w(x - x_p)}{\sum_{p=1}^{npt} w(x - x_p)},$$

and $u_p$, $p = 1, 2, \cdots, npt$ is data on a set of scattered points. In MPM, the weight function $w$ is defined as

$$w(x - x_p) = M_p s(x - x_p), \tag{4.7}$$

where in 1D, with $r_x = \frac{|x - x_p|}{h}$,

$$s(x - x_p) = \begin{cases} 1 - r_x & r_x \leq 1 \\ 0 & \text{otherwise} \end{cases}, \tag{4.8}$$

where $h$ is the support of $s$ and is also the mesh spacing. In 2D, $s(x - x_p, y - y_p) = s(x - x_p) \cdot s(y - y_p)$. Note that the Shepard interpolation reproduces only constant functions. In other words, if $u_p = C$, where $C$ is a constant, we would have $u^h(x) = C$. The proof is as follows:

$$\begin{aligned} u^h(x) &= \sum_{p=1}^{npt} \phi_p(x) C, \\ &= \frac{\sum_{p=1}^{npt} w(x - x_p) C}{\sum_{p=1}^{npt} w(x - x_p)}, \\ &= C. \end{aligned}$$

For equally distributed sampled data, the Shepard interpolation almost reproduces linear functions, except on the boundary of the domain. See the following $1D$ examples for an illustration. Here the domain is $\Omega_0 = [-1, 1]$ and $h = \frac{1}{10}$. In each element, there are four equally spaced material points, which are the sampled points. For a

given function $u(x)$, sample the function at points $x_p$ to get $u_p = u(x_p)$, use formula (4.6) to obtain $u^h(x)$ with $M_p = 1$, and $s(x)$ is given by (4.8), and $p = 1, 2, \cdots, 40$. The reconstructed function $u^h(x)$ is computed at the grid nodes $x_i = -1 + (i-1)h$, where $i = 1, 2, \cdots, 11$. The exact function values are given by $u(x_i)$ and the reconstructed values are given by $u^h(x_i)$ at $i = 1, 2, \cdots, 11$. The difference $u(x_i) - u^h(x_i)$ at the 11 nodes is computed for $u(x) = 1$, $u(x) = x$ and $u(x) = x^2$. Table 4.1 shows the error at each node for three cases, $u(x) = 1$, $u(x) = x$ and $u(x) = x^2$. From Table 4.1, we see that for equally spaced points, the error, $u(x_i) - u^h(x_i)$, is identically zero for $u(x) = 1$. The error is also identically zero for interior nodes even for the linear function $u(x) = x$, but is nonzero throughout the interval for the example $u(x) = x^2$. The error for $u(x) = x^2$ is fairly uniform in the interior of the interval with larger values at the endpoints. Therefore, although Shepard functions are only guaranteed to reproduce constant functions, the error in reproducing a linear function with equally spaced points is generally nonzero only at the boundary. In Figures 4.1, 4.2 and 4.3, the blue circles are the computed nodal values of the reconstructed function $u^h(x)$, and the blue line is the graph of $u(x)$. If the material points are equally spaced, one can see that in Figure 4.1, the Shepard interpolation reproduces constant functions. In Figure 4.2, the Shepard interpolation reproduces a linear function in the interior of the domain, but not on the boundary. In Figure 4.3, the Shepard interpolation does not reproduce the quadratic exactly, and the errors are larger at the boundary than in the interior of the interval.

Table 4.1: Pointwise errors for Shepard Interpolation with equally spaced points.

| $i$ | $1 - u^h(x_i)$ | $x_i - u^h(x_i)$ | $x_i^2 - u^h(x_i)$ |
|---|---|---|---|
| 1 | 0 | -0.0688 | 0.1306 |
| 2 | 0 | 0 | -0.0069 |
| 3 | 0 | 0 | -0.0069 |
| 4 | 0 | 0 | -0.0069 |
| 5 | 0 | 0 | -0.0069 |
| 6 | 0 | 0 | -0.0069 |
| 7 | 0 | 0 | -0.0069 |
| 8 | 0 | 0 | -0.0069 |
| 9 | 0 | 0 | -0.0069 |
| 10 | 0 | 0 | -0.0069 |
| 11 | 0 | 0.0687 | 0.1306 |

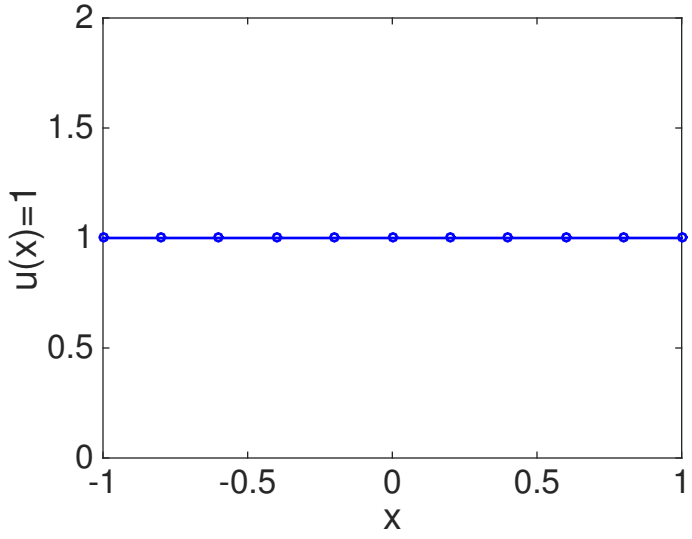


Figure 4.1: Shepard interpolation for the constant function $u(x) = 1$ with equally spaced points.

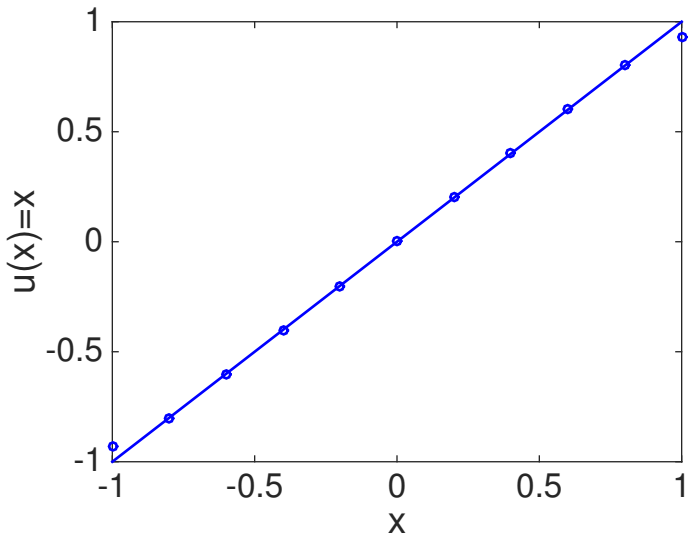


Figure 4.2: Shepard interpolation for the linear function $u(x) = x$ with equally spaced points.

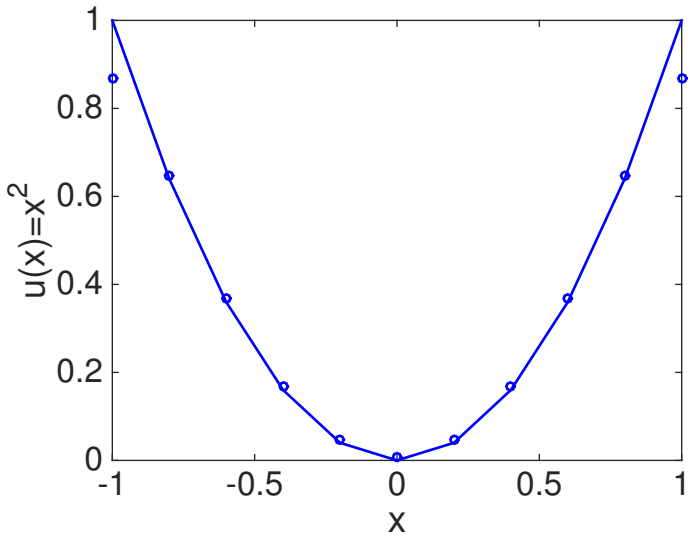


Figure 4.3: Shepard interpolation for the quadratic function $u(x) = x^2$ with equally spaced points.

For arbitrarily placed sampling points $x_p$ in the interval [-1,1] generated by the rand command in Matlab, the Shepard interpolation only reproduces constant functions. See the following table and 1D plots for an illustration. From Table 4.2, one can see that the error is identically zero for $u(x) = 1$. The errors become large throughout the domain for the linear function $u(x) = x$ and the quadratic function $u(x) = x^2$. In Figures 4.4, 4.5 and 4.6, the blue circles are nodal values of the reconstructed function $u^h(x)$, and the blue lines are graphs of the function $u(x)$. For the constant function $u(x) = 1$ in Figure 4.4, the Shepard interpolation reproduces constant functions exactly. For the linear function $u(x) = x$ and the quadratic function $u(x) = x^2$, Shepard interpolation has errors throughout the interval, with larger errors at the boundary. This observation explains why the mapping from the material points to the grid in the MPM algorithm drops down to first order when the positions of the material points become unequally distributed, which is one of the main reasons for low order accuracy of the overall MPM algorithm.

Table 4.2: Pointwise errors for Shepard Interpolation with unequally spaced points.

| $i$ | $1 - u^h(x_i)$ | $x_i - u^h(x_i)$ | $x_i^2 - u^h(x_i)$ |
|---|---|---|---|
| 1 | 0 | -0.1370 | 0.1963 |
| 2 | 0 | 0.0105 | 0.0114 |
| 3 | 0 | -0.0046 | -0.0273 |
| 4 | 0 | -0.0330 | -0.0250 |
| 5 | 0 | 0.0403 | 0.0104 |
| 6 | 0 | -0.0129 | -0.0090 |
| 7 | 0 | -0.0275 | -0.0187 |
| 8 | 0 | -0.0134 | -0.0187 |
| 9 | 0 | 0.0390 | 0.0472 |
| 10 | 0 | -0.0119 | -0.1511 |
| 11 | 0 | 0.0797 | 0.1751 |

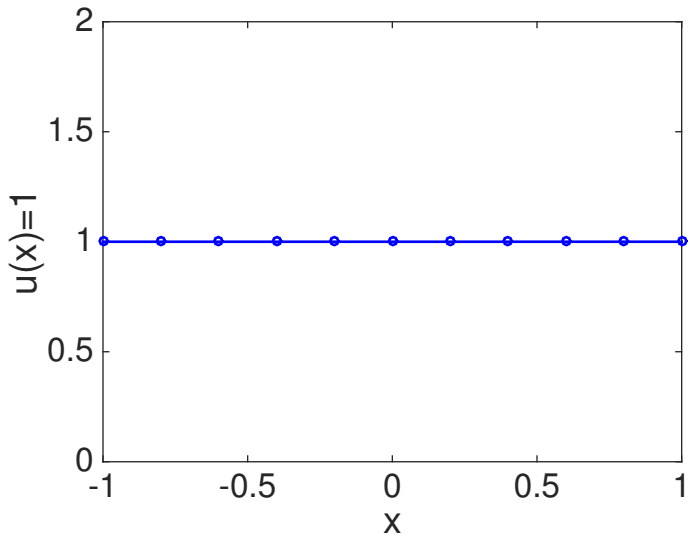


Figure 4.4: Shepard interpolation for the constant function $u(x) = 1$ with unequally spaced points.

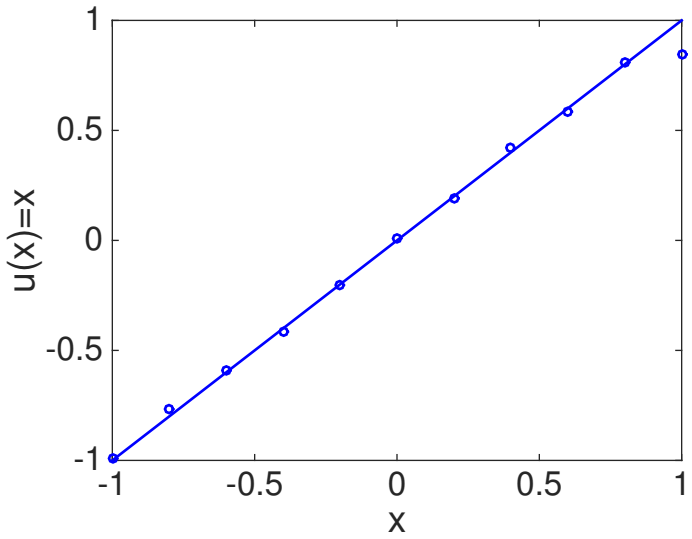


Figure 4.5: Shepard interpolation for the linear function $u(x) = x$ with unequally spaced points.

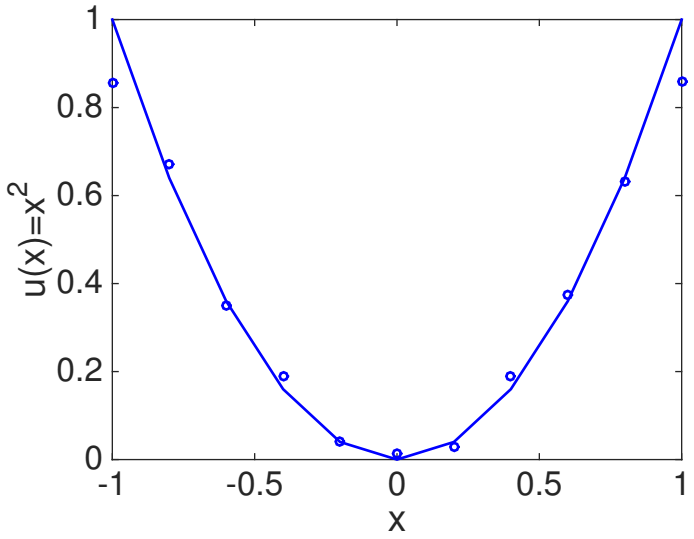


Figure 4.6: Shepard interpolation for the quadratic function $u(x) = x^2$ with unequally spaced points.

### 4.3.2 Solve the momentum equation on grid

Define grid coordinates $x_i, i = 1, 2, \cdots, n$, where $n$ is the total number of grid points. Multiply equation (4.1) with a test function $\delta u$ on both sides of the equation and perform integration by parts to get the following alternative form:

$$\int_\Omega \rho[\delta u \cdot a + \sigma^s : \nabla \delta u]dx = \int_\Omega \rho \delta u \cdot b dx + \int_{\partial\Omega^\tau} \delta u \cdot \overline{\tau} ds, \tag{4.9}$$

where $\rho(x,t)\sigma^s(x,t) = \sigma(x,t)$ and $\partial\Omega = \partial\Omega^v \cup \partial\Omega^\tau$ with $\partial\Omega^\tau \cap \partial\Omega^v = \emptyset$. Here $\partial\Omega^v$ is part of the boundary where the velocity boundary condition is prescribed and $\partial\Omega^\tau$ is part of the boundary where the traction boundary condition is prescribed. Now, project functions to a finite dimensional space and use the conventional finite element representation for the continuous variables to write the following:

$$\delta u = \sum_{i=1}^{n} \delta u_i N_i(x), \tag{4.10}$$

$$v = \sum_{i=1}^{n} v_i N_i(x), \tag{4.11}$$

$$a = \sum_{i=1}^{n} a_i N_i(x), \tag{4.12}$$

where $N_i(x)$ are standard FEM nodal basis functions. Nodal values of $\delta u$, $v$ and $a$ are denoted by $\delta u_i$, $v_i$ and $a_i$. In $1D$, the basis function is defined by $N_i(x_j) = \delta_{ij}$, where $\delta_{ij}$ is the Kronecker delta function. In $2D$, the basis function is the tensor product of the $1D$ basis functions. Now, substitute (4.10), (4.11) and (4.12) into each term of equation (4.9) and use one-point quadrature over material point domains to get

the following expansions:

$$\int_\Omega \rho\delta u \cdot a dx = \sum_{i,j=1}^{n} \delta u_i m_{ij} \cdot a_j, \tag{4.13}$$

$$m_{ij} = \int_\Omega \rho N_i(x) N_j(x) dx = \sum_{p=1}^{N} M_p N_i(x_p) N_j(x_p),$$

$$\int_\Omega \rho\sigma^s : \nabla\delta u dx = -\sum_{i=1}^{n} \delta u_i \cdot f_i^{int}, \tag{4.14}$$

$$f_i^{\text{int}} = -\int_\Omega \rho\sigma^s \cdot \nabla N_i(x) dx = -\sum_{p=1}^{N} \nabla N_i(x_p) M_p \cdot \sigma_p^s.$$

If the lumped mass is used, then the nodal mass has the following simplified form:

$$\begin{aligned} m_i &= \sum_{j=1}^{n} \int_\Omega \rho N_i(x) N_j(x) dx, \\ &= \int_\Omega \rho N_i(x) \sum_{j=1}^{n} N_j(x) dx, \\ &= \int_\Omega \rho N_i(x) dx, \\ &= \sum_{p=1}^{N} M_p N_i(x_p). \end{aligned} \tag{4.15}$$

For the right-hand side of equation (4.9), similar expansions can be found:

$$\int_\Omega \rho\delta u \cdot b dx = \sum_{i=1}^{n} \delta u_i \cdot b_i, \tag{4.16}$$

$$b_i = \int_\Omega N_i(x) \rho b dx = \sum_{p=1}^{N} N_i(x_p) M_p b_p, \tag{4.17}$$

$$\int_\Omega \delta u \cdot \tau ds = \sum_{i=1}^{n} \delta u_i \cdot \hat{\tau}_i, \tag{4.18}$$

$$\hat{\tau}_i = \int_\Omega N_i \tau ds. \tag{4.19}$$

Here $b_p = b(x_p, t)$ is defined as the body force evaluated at the material points; $[m_{ij}]$ is the mass matrix associated with the background computation grid; and $f_i^{\text{int}}$ is

defined as the internal force associated with node $i$. In addition, $f_i^{\text{ext}}$ is defined as the external nodal force in the following manner:

$$f_i^{\text{ext}} = b_i + \hat{\tau}_i. \tag{4.20}$$

Combining (4.13) (4.14) and (4.20), the discrete form of the equation of motion has been derived as follows:

$$\begin{aligned}
&\sum_{i,j=1}^{n} \delta u_i m_{ij} \cdot a_j = -\sum_{i=1}^{n} \delta u_i \cdot f_i^{\text{int}} + \sum_{i=1}^{n} \delta u_i \cdot b_i + \sum_{i=1}^{n} \delta u_i \cdot \hat{\tau}_i, \\
&\sum_{i,j=1}^{n} \delta u_i m_{ij} \cdot a_j = -\sum_{i=1}^{n} \delta u_i \cdot f_i^{\text{int}} + \sum_{i=1}^{n} \delta u_i f_i^{\text{ext}}.
\end{aligned} \tag{4.21}$$

Since $\delta u_i$ is arbitrary, if terms with $\delta u_i$ as the coefficient are collected, the following system of equations has been achieved:

$$\sum_{j=1}^{n} m_{ij} a_j = f_i^{\text{int}} + f_i^{\text{ext}}, \quad i = 1, 2, \cdots, n. \tag{4.22}$$

If the lumped mass is used, the equation of motion has the following simplified form:

$$m_i a_i = f_i^{\text{int}} + f_i^{\text{ext}}, \quad i = 1, 2, \cdots, n, \tag{4.23}$$

where $m_i$ is given by (4.15).

**Solving the momentum equation on the grid and updating the grid velocity.** The above procedure presents the discretized momentum equation, and in practice, we need to indicate the time step. Let a superscript $s$ denote the time step. The grid velocity is updated as follows:

$$\begin{aligned}
&m_i^s a_i^s = (f_i^{int})^s + (f_i^{ext})^s, \quad i = 1, 2, \cdots, n, \\
&v_i^{s+\frac{1}{2}} = v_i^{s-\frac{1}{2}} + a_i^s \Delta t,
\end{aligned} \tag{4.24}$$

where $\Delta t$ is the time step. Note the initialization of velocity at a half time step $v_i^{\frac{1}{2}}$ is required. The initial half time step velocity can be approximated using $v_i^{\frac{1}{2}} =$

$\frac{1}{2}(v_i^1 + v_i^0)$, where $v_i^0$ is given by the initial condition and $v_i^1$ can be obtained by solving the equations of motion on the background grid (i.e. $v_i^1 = v_i^0 + a_i^0 \Delta t$). Or the velocity at half time step can be approximated by another method as desired.

### 4.3.3 Update the material-point information

Once the equations of motion on the grid are solved, the material-point information must be updated. The steps are presented as follows:

$$v_p^{s+\frac{1}{2}} = v_p^{s-\frac{1}{2}} + \sum_{i=1}^{n} N_i(x_p^s)(v_i^{s+\frac{1}{2}} - v_i^{s-\frac{1}{2}}), \tag{4.25}$$

$$x_p^{s+1} = x_p^s + \sum_{i=1}^{n} N_i(x_p^s)(v_i^{s+\frac{1}{2}})\Delta t, \tag{4.26}$$

$$u_p^{s+1} = x_p^{s+1} - x_p^0. \tag{4.27}$$

The deformation gradient, density and stress are updated as

$$F_p^{s+1} = (\sum_{i=1}^{n} \nabla N_i(x_p^{s+1}) v_i^{s+\frac{1}{2}} \Delta t + I) F_p^s, \tag{4.28}$$

$$\rho_p^{s+1} = \frac{\rho_0}{|F_p^{s+1}|}, \tag{4.29}$$

$$\sigma_p^{s+1} = \sigma(F_p^{s+1}), \tag{4.30}$$

where $\nabla N_i$ is the gradient of the basis function.

### 4.3.4 Generate a new grid

After the material-point information is updated, a new grid must be generated for the next time step. Usually in MPM implementations, the new grid is simply moved back to the location of the old grid so that grid generation becomes trivial. During

the Lagrangian step, the grid moves to

$$x_i^{s+1} = x_i^s + v_i^{s+\frac{1}{2}}\Delta t.$$

During the re-grid, the grid is set back to the original grid

$$x_i^{s+1} = x_i^s = x_i. \tag{4.31}$$

Note, however, one can generate any convenient grid instead.

## 4.4 Summary of the algorithm of MPM

There are four main steps in the MPM algorithm.

(1) Map information from material points to grid.

$$\begin{aligned}
(mv)_i^{s-\frac{1}{2}} &= \sum_{p=1}^{npt} M_p v_p^{s-\frac{1}{2}} N_i(x_p^s), \quad i = 1, 2, \cdots, n, \\
m_i^s &= \sum_{p=1}^{npt} M_p N_i(x_p^s), \\
v_i^{s-\frac{1}{2}} &= \frac{(mv)_i^{s-\frac{1}{2}}}{m_i^s}, \\
f_i^{\text{int}} &= -\sum_{p=1}^{N} \nabla N_i(x_p) M_p \cdot \sigma_p^s, \\
f_i^{\text{ext}} &= b_i + \hat{\tau}_i.
\end{aligned}$$

$b_i$ and $\hat{\tau}_i$ are given by equations (4.17) and (4.19).

(2) Solve the equation of motion on the grid and update the grid information.

$$\begin{aligned}
m_i^s a_i^s &= (f_i^{\text{int}})^s + (f_i^{\text{ext}})^s, \quad i = 1, 2, \cdots, n, \\
v_i^{s+\frac{1}{2}} &= v_i^{s-\frac{1}{2}} + a_i^s \Delta t.
\end{aligned}$$

(3) Update the information on the material points.

$$
\begin{aligned}
v_p^{s+\frac{1}{2}} &= v_p^{s-\frac{1}{2}} + \Delta t \sum_{i=1}^{n} N_i(x_p^s) a_i^s, \\
x_p^{s+1} &= x_p^s + \Delta t \sum_{i=1}^{n} N_i(x_p^s) v_i^{s+\frac{1}{2}}, \\
u_p^{s+1} &= x_p^{s+1} - x_p^0, \\
F_p^{s+1} &= (\sum_{i=1}^{n} \nabla N_i(x_p^{s+1}) v_i^{s+\frac{1}{2}} \Delta t + I) F_p^s, \\
\rho_p^{s+1} &= \frac{\rho_0}{|F_p^{s+1}|}, \\
\sigma_p^{s+1} &= \sigma(F_p^{s+1}).
\end{aligned}
$$

(4) Generate a new grid.

## 4.5 Method of Manufactured Solutions in $1D$

The idea of a manufactured solution is that one assumes a solution has a certain form and puts the solution back into the governing equation to solve for appropriate data such as the body force or boundary conditions. See [1, 24] for the example used in this section. First consider the equation of motion in Lagrangian form,

$$\nabla_0 \cdot P + \rho_0 B = \rho_0 A, \tag{4.32}$$

where $P$ is the first Piola-Kirchoff stress. The Neo-Hookean model is used, which is given as

$$P = [\lambda(\ln J)I + \mu(F^T F - I)]F^{-T}. \tag{4.33}$$

where $\lambda$ and $\mu$ are generalized Lam*é* constants, $F$ is the deformation gradient, which is defined as $F = I + \frac{\partial u}{\partial X}$ and $J = |F|$. The stress is related in Eulerian and

Lagrangian forms as

$$J\sigma = PF^T, \tag{4.34}$$

where $\sigma$ is the Cauchy stress.

Consider a bar in the domain $0 \leq X \leq 1$. Assume the following analytical solution for displacement.

$$U(X,t) = G\sin(\pi X)\sin(c\pi t), \tag{4.35}$$

where $c = \sqrt{\frac{E}{\rho_0}}$ and $E$ is Young's modulus. Then, from $U$, find $F$, $V$ and $A$:

$$F(X,t) = 1 + G\pi\cos(\pi X)\sin(c\pi t), \tag{4.36}$$

$$V(X,t) = c\pi G\sin(\pi X)\cos(c\pi t), \tag{4.37}$$

$$A(X,t) = -Gc^2\pi^2\sin(c\pi t)\sin(\pi X). \tag{4.38}$$

$P$ is given by expression (4.33). The boundary conditions are given by

$$V(X,t)|_{X=0} = 0,$$
$$V(X,t)|_{X=1} = 0.$$

The corresponding initial conditions are given as

$$V(X,0) = cG\pi\sin(\pi X),$$
$$P(X,0) = 0,$$
$$F(X,0) = 1.$$

In order that equation (4.35) is a solution to equation (4.32) with the Neo-Hooken model, the corresponding equation for body force is

$$B(X,t) = \frac{\pi^2 U(X,t)}{\rho_0}\left(\frac{\lambda}{F^2}(1-\ln F) + \mu\left(1+\frac{1}{F^2}\right) - E\right). \tag{4.39}$$

In this expression, $E$ is the Young's modulus which is related to $\lambda$ and $\mu$ by $E = \frac{\lambda(1+\mu)(1-2\mu)}{\mu}$.

## 4.6 Convergence of MPM in $1D$

This section considers the solution to a initial-boundary-value problem in $1D$ using the original version of MPM as summarized in the previous sections. The problem is given as follows:

$$\begin{aligned}
\rho a &= \nabla \cdot \sigma + \rho b, \\
\frac{dF}{dt} &= F v_x, \\
\sigma &= \frac{1}{J}[\lambda(\ln J)I + \mu(FF^T - I)], \\
\sigma(x,0) &= 0, \\
F(x,0) &= 1, \\
v_0(x) &= c\pi G \sin(\pi x), \\
v(x,t)|_{x=x(X=0,t=0)} &= 0, \\
v(x,t)|_{x=x(X=1,t=0)} &= 0.
\end{aligned}$$

in what follows, we study convergence of the original MPM algorithm to the solution of this problem. The analytical solution is (4.35), (4.36), (4.37) and (4.38). Here the grid is defined as $x_i = (i-1)h$, $i = 1, 2, \cdots, N_e + 1$ on the interval $[0, 1]$, and the meshsize is $h = \frac{1}{Ne}$, where $Ne$ is the total number of elements. Note that the mesh sizes used for the following convergence plot are $h = \frac{1}{2^i}, i = 3, 4, 5, 6$. For the following convergence plot, the axes are log base 10 of the $L_2$ norm of the error vs. the log base 10 of the mesh size. The slope of the line that approximates the data reveals the convergence rate. The $L_2$ error is defined as

$$E_{l2} = (\int_\Omega (u_n^{\text{num}}(x) - u_n^{\text{ex}}(x))^2 d\Omega)^{\frac{1}{2}}, \tag{4.40}$$

where $u_n^{\text{num}}$ is the numerical solution and $u_n^{\text{ex}}$ is the analytical solution at time $t^n$. The error is computed using one point quadrature over the material-point domain for the stress, density, velocity and displacement, and over the element domain for

the velocity. Since the analytical solution for the velocity is only known on the material points, second order MLS can be used to map the analytical velocity on the material points to the grid so that the analytical solution of velocity on the grid can be approximated and the error of velocity on the grid can be computed.

Figures 4.7 and 4.8 show computed convergence rates for the original MPM algorithm for this manufactured solution. The Young's modulus is $E = 10^7 Pa$ and the Poisson's ratio is $\nu = 0.3$, the initial density is $\rho_0 = 10^3 kg/m^3$ and the final time is $T = 0.02s$, which corresponds to the solution after one period. The time step chosen for the simulation is given by $\Delta t = 0.4\frac{h}{c}$, where $c = \sqrt{\frac{E}{\rho_0}}$. If we use strain as a measure of how much deformation the elastic body undergos, in Figure 4.7, the maximum strain is $G \max \frac{\partial U}{\partial X} = \pi G = 0.000314$, which is a small deformation. In Figure 4.8, the maximum strain is $\pi G = 0.314$, which is a large deformation. In the figures, the top line is the error in the stress on the material points, the second line is the error in the density on the material points, the third line is the error in the velocity on the material points and on the grid, and the fourth line is the error in the displacement on the material points.

As one can see in the Figures 4.7 and 4.8, for small deformation problems, MPM appears to be second order accurate, but for large deformation problems, it does not converge. The improved material point method is designed to improve MPM in this situation.

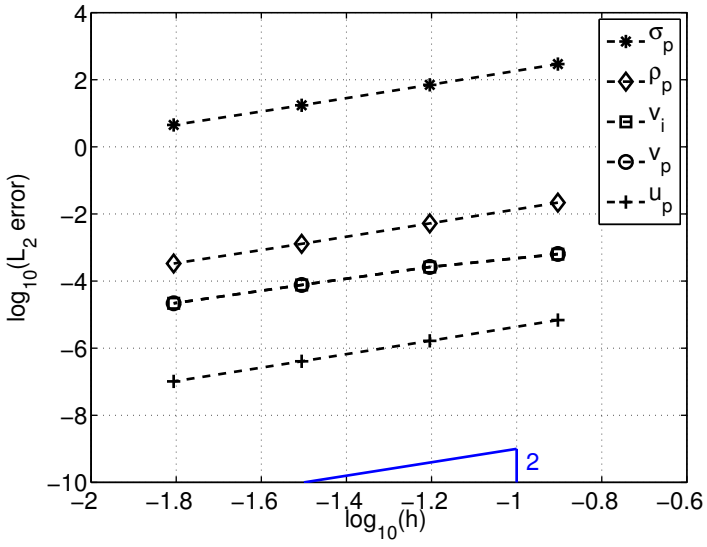


Figure 4.7: Convergence of the original MPM for a small deformation problem in $1D$, G=0.0001.

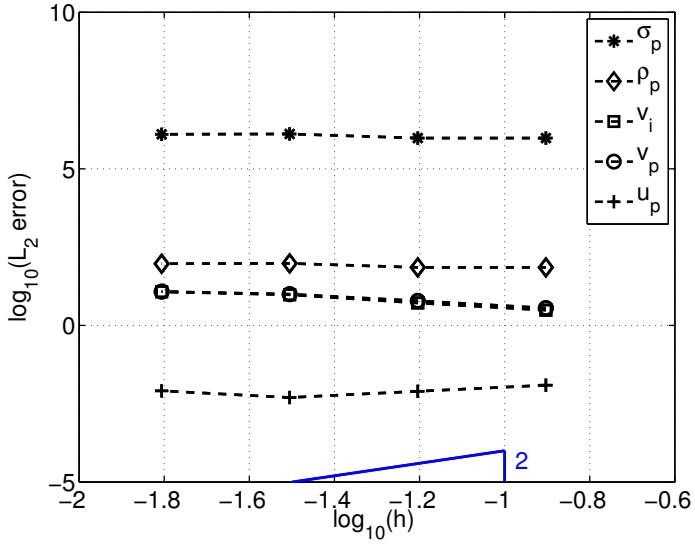


Figure 4.8: Convergence of the original MPM for a large deformation problem in $1D$, G=0.1.

## 4.7 Convergence of MPM in $2D$

In this section, a manufactured solution in two space dimensions is discussed as originally presented in [1, 24]. For equations (4.1) to (4.5), the displacement is assumed to have the following form:

$$U(X,Y,t) = \begin{bmatrix} G\sin(\pi X)\sin(c\pi t) \\ G\sin(\pi Y)\sin(c\pi t) \\ 0 \end{bmatrix}. \tag{4.41}$$

From $U$, we can compute velocity and acceleration as follows:

$$V(X,Y,t) = \begin{bmatrix} c\pi G\sin(\pi X)\cos(c\pi t) \\ c\pi G\sin(\pi Y)\cos(c\pi t) \\ 0 \end{bmatrix}, \tag{4.42}$$

$$A(X,Y,t) = \begin{bmatrix} -c^2\pi^2 G\sin(\pi X)\sin(c\pi t) \\ -c^2\pi^2 G\sin(\pi Y)\sin(c\pi t) \\ 0 \end{bmatrix}. \tag{4.43}$$

The deformation gradient is

$$F(X,Y,t) = \begin{bmatrix} 1+\pi G\cos(\pi X)\sin(c\pi t) & 0 & 0 \\ 0 & 1+\pi G\cos(\pi Y)\sin(c\pi t) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{4.44}$$

The first Piola-Kirchoff stress $P$ is found using equation (4.33) to be

$$P(X,Y,t) = \begin{bmatrix} P_{11} & 0 & 0 \\ 0 & P_{22} & 0 \\ 0 & 0 & P_{33} \end{bmatrix}, \tag{4.45}$$

where $P_{11} = \frac{\lambda}{F_{11}} \ln(F_{11}F_{22}) + \mu(F_{11} - \frac{1}{F_{11}})$, $P_{22} = \frac{\lambda}{F_{22}} \ln(F_{11}F_{22}) + \mu(F_{22} - \frac{1}{F_{22}})$ and $P_{33} = \lambda \ln(F_{11}F_{22}))$. The corresponding body force is

$$B(X,Y,t) = \begin{bmatrix} \frac{\pi^2 U_1(X,Y,t)}{\rho_0}(\frac{\lambda}{F_{11}^2}(1 - \ln(F_{11}F_{22})) + \mu(1 + \frac{1}{F_{11}^2}) - E) \\ \frac{\pi^2 U_2(X,Y,t)}{\rho_0}\frac{\lambda}{F_{22}^2}(1 - \ln(F_{11}F_{22})) + \mu(1 + \frac{1}{F_{22}^2}) - E) \\ 0 \end{bmatrix}. \tag{4.46}$$

$U_1$ is the first component of the displacement vector, and $U_2$ is the second component of the displacement vector. $F_{11}$ and $F_{22}$ are the diagonal components of the deformation gradient tensor.

The initial and boundary conditions are :

$$\begin{aligned} \sigma(x,y,0) &= 0, \\ F(x,y,0) &= I, \\ v_0(x,y) &= [c\pi G \sin(\pi x), c\pi G \sin(\pi y)], \\ v^1(x,y,t)|_{x=x(X=0,Y,t=0)} &= 0, \\ v^1(x,y,t)|_{x=x(X=1,Y,t=0)} &= 0, \\ v^2(x,y,t)|_{y=y(X=0,Y,t=0)} &= 0, \\ v^2(x,y,t)|_{y=y(X=1,Y,t=0)} &= 0, \\ \tau(x,y,t)|_{x=x(X,Y=0,t=0)} &= 0, \\ \tau(x,y,t)|_{x=x(X,Y=1,t=0)} &= 0, \end{aligned}$$

where $v^1$ and $v^2$ are respectively the first and second components of the velocity. In the following plots, the Young's modulus is $E = 10^7 Pa$ and $\nu = 0.3$, the initial density is $\rho_0 = 10^3 kg/m^3$ and the final time $T = 0.02s$, which corresponds to the solution after one period. The time step chosen for the simulation is given by $\Delta t = 0.4\frac{h}{c}$, where $c = \sqrt{\frac{E}{\rho_0}}$. The error in the density is computed using equation (4.40). The $L_2$ error of displacement is computed to be

$$E_{l2} = (\int_\Omega ((u_n^1)^{\text{num}} - (u_n^1)^{\text{ex}})^2 + ((u_n^2)^{\text{num}} - (u_n^2)^{\text{ex}})^2 dx)^{\frac{1}{2}}, \tag{4.47}$$

where $(u_n^1)^{\text{num}}$ and $(u_n^2)^{\text{num}}$ are the numerical values of the first and second components of the displacement, and $(u_n^1)^{\text{ex}}$ and $(u_n^2)^{\text{ex}}$ are the exact values of the first and second components of the displacement at time $t^n$. The $L_2$ error in the velocity is computed similarly. The error in stress is computed to be

$$E_{l2} = (\int_\Omega ((\sigma_n^{11})^{\text{num}} - (\sigma_n^{11})^{\text{ex}})^2 + ((\sigma_n^{12})^{\text{num}} - (\sigma_n^{12})^{\text{ex}})^2 \\ +((\sigma_n^{21})^{\text{num}} - (\sigma_n^{21})^{\text{ex}})^2 + ((\sigma_n^{22})^{\text{num}} - (\sigma_n^{22})^{\text{ex}})^2 dx)^{\frac{1}{2}}, \tag{4.48}$$

where $(\sigma_n^{11})^{\text{num}}$, $(\sigma_n^{12})^{\text{num}}$, $(\sigma_n^{21})^{\text{num}}$, and $(\sigma_n^{22})^{\text{num}}$ are the 11, 12, 21 and 22 components of the numerically computed stress tensor, and $(\sigma_n^{11})^{\text{ex}}$, $(\sigma_n^{12})^{\text{ex}}$, $(\sigma_n^{21})^{\text{ex}}$, and $(\sigma_n^{22})^{\text{ex}}$ are the 11, 12, 21 and 22 components of the analytical solution for the stress tensor at time $t^n$. The above errors are computed using one point quadrature over the material-point domain for stress, density, velocity and displacement, and over the element domain for the velocity. Since we only know the analytical solution of velocity on the material points, we use second order MLS to map the analytical velocity on the material points to the grid so that we can approximate the analytical solution of velocity on the grid and compute the error of velocity on the grid.

In Figures 4.9 and 4.10, the top line shows errors in the stress tensor on the material points, the second line is the error in the density on the material points, the third line is the error in the velocity on the material points and on the grid, and the fourth line is the error in the displacement on the material points.

For the small deformation problem in Figure 4.9, where $G = 0.0001$, the displacement of MPM appears to be second order accurate, but other quantities have a convergence rate between order 1 and order 2. For the large deformation problem in Figure 4.10, where $G = 0.05$, MPM does not converge.

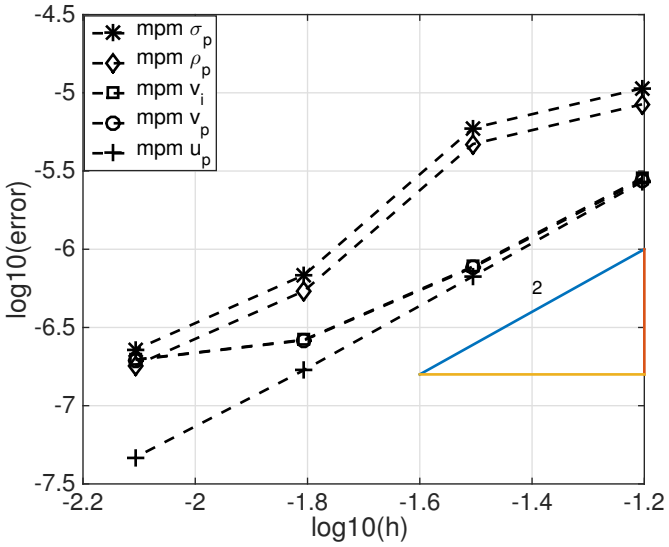


Figure 4.9: Convergence of the original MPM for a small deformation problem in $2D$, G=0.0001.

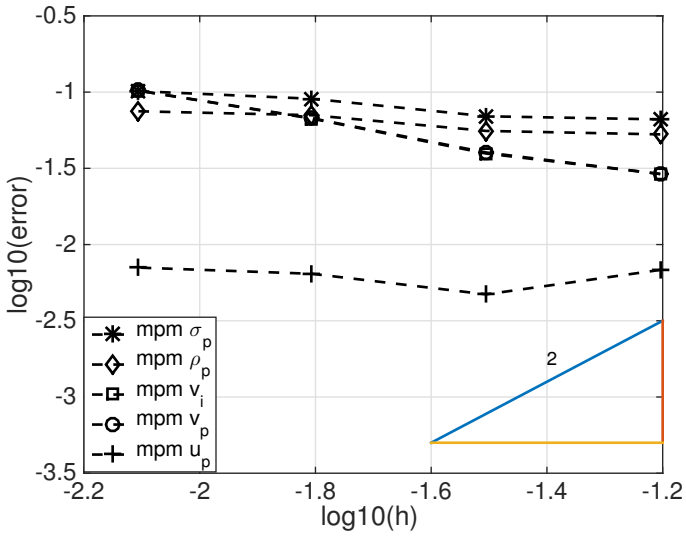


Figure 4.10: Convergence of the original MPM for a large deformation problem in $2D$, G=0.05.

# Chapter 5

# Improved Material-Point Method

## 5.1 Introduction

The main steps of IMPM are the same as the main steps of MPM. Two main issues account for the low order behavior of the original MPM algorithm. The first issue corresponds to mapping the velocity field from the material-point information to the grid, which is first order for large deformation problems. The second issue corresponds to the quadrature used to approximate the nodal forces or nodal masses. In MPM, material points are used as the integration points. Since the material points move, their positions can be arbitrary in the computational cell, and then the accuracy of corresponding nodal forces and nodal masses is low order for large deformation problems. The general purpose of IMPM is to improve the algorithm in each piece so that overall, the whole algorithm is improved. Here we only try to improve MPM to second order (i.e., velocity has second order convergence).

The above two reasons naturally lead to the consideration of a better method of constructing the mapping and the quadrature points. Here in IMPM, the MLS techniques discussed in section (3.4) are used to reconstruct the functions from the

information given on the material points. The function is then evaluated at any desired points. For example, the reconstructed velocity can be evaluated on the grid and the reconstructed stress tensors or density can be evaluated on any quadrature points. More generally, by using MLS, the mapping can be done to an arbitrary order from material-point information to information on any other desired points. Here in IMPM, functions are only reconstructed up to second order, which will be sufficient to increase the order of accuracy of original MPM in space up to second order.

In this chapter, the improved mapping and improved quadrature will be studied in detail and the main steps of IMPM will be discussed. The convergence of IMPM in $1D$ and $2D$ simulations will be presented.

## 5.2 Convergence Study of First Order and Second Order MLS

In this section, the function reconstruction technique illustrated in Section 3.4 is used to study the convergence properties of first and second order MLS numerically for a one-dimensional problem.

The domain is the interval [0,1], and $h = \frac{1}{10}$. In Figures, 5.1 and 5.2, the x-axis is the position and the y-axis is the function values for both the original function $u(x)$ and the reconstructed function $u^h(x)$ given in equation (3.16). Two cases are studied; $u(x) = 1.5$ and $u(x) = x + 1$. we sample the function $u(x)$ at points $x_p$ to get $u_p = u(x_p)$, with arbitrarily placed sample points $x_p$. The black squares in the figures are the positions of the sampled material points $x_p$. Once the information on the sampled points is obtained, the task is to reconstruct the function $u^h(x)$ from the sampled points. The blue stars are the values of the reconstructed $u^h(x)$ at the grid

points, where the grid points are the equally spaced points, $x_i = -1 + ih$. The blue line is the graph of the function $u(x)$. As we saw in Chapter 4, the first order MLS reproduces constant functions exactly, but no longer reproduces linear functions for unequally distributed points. By contrast, in Figures 5.1 and 5.2, second order MLS reproduces constant functions and linear functions exactly.

In Figures 5.3 and 5.4, the x-axis is $\log_{10}(h)$ where $h$ is the spacing, and the y-axis is $\log_{10}(\text{error})$ where the error is computed using the standard $l_2$ norm as follows:

$$\text{error} = \sqrt{\sum_{i=1}^{n} h(u(x_i) - u^h(x_i))^2},$$

where $n$ is the total number of grid points and $h = \frac{1}{n-1}$. The function used in this example is $u(x) = \sin(\pi x)$. First order MLS is second order convergent for equally spaced points, but reduces to first order convergence if the material points are unequally distributed as in Figure 5.3. Second order MLS, even though the material points are unequally distributed, is second order accurate as in Figure 5.4. This situation explains how in order to improve MPM for large deformation problems, where the material points become arbitrarily distributed, the mapping part of the algorithm must be improved. In IMPM, second order MLS is used instead of Shepard interpolation (first order MLS) to map the information from material-point fields to information on the background grid.

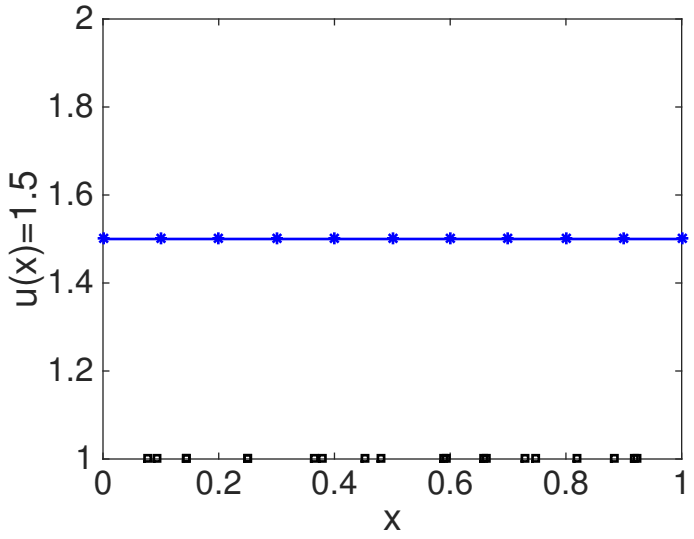


Figure 5.1: 2nd order MLS for a constant function with unequally spaced points.

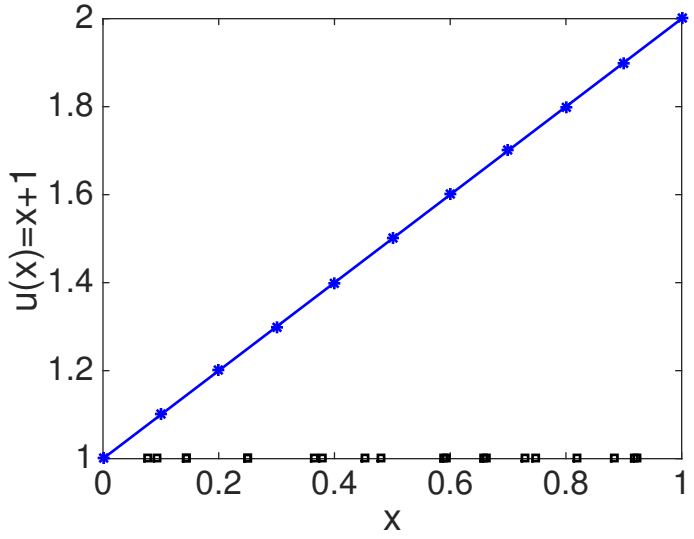


Figure 5.2: 2nd order MLS for a linear function with unequally spaced points.

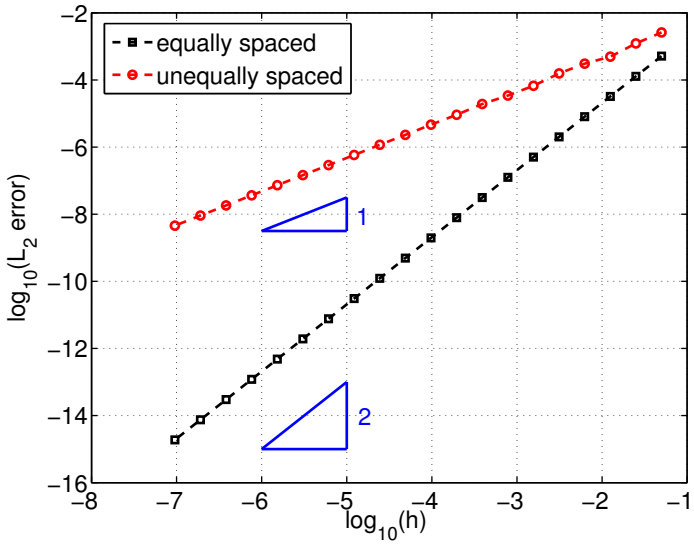


Figure 5.3: Convergence of the 1st order MLS.

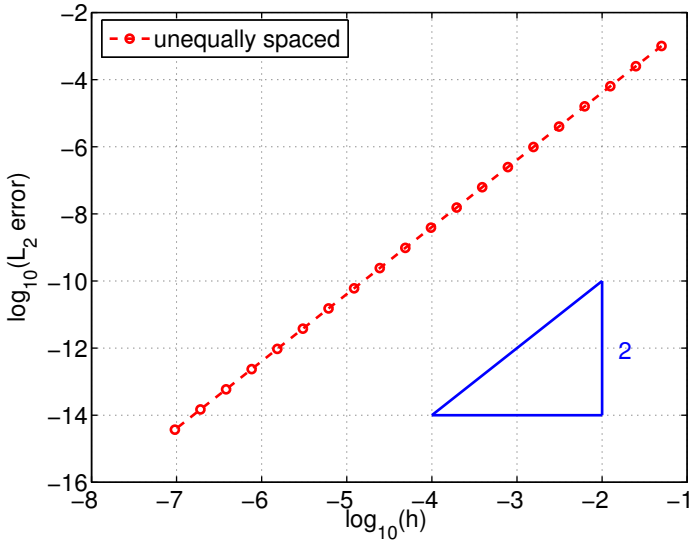


Figure 5.4: Convergence of the 2nd order MLS.

## 5.3 Improvements of Mapping and Quadrature

This section utilizes the techniques of function reconstruction, using MLS to improve the mapping and quadrature. First, the velocity field data is reconstructed from the material points using second order MLS, and the reconstructed velocity is evaluated on the background grid. This reconstruction and evaluation improves the mapping of velocity from material-point information to the information on the background grid. Next, the density and stress tensor data are reconstructed from the material points, and the reconstructed density and stress tensor are evaluated on the quadrature points. Once we obtain the information on the quadrature points, we can use them to improve the accuracy of nodal forces and nodal masses.

At each time step, the density, velocity, and stress are known on the material points, and by utilizing MLS, density, velocity and stress can be reconstructed from the material points. Here linear polynomials are used as the basis for constructing the function (i.e. $P(x) = [1, \tilde{x}^1]$ in $1D$, and $P(x) = [1, \tilde{x}^1, \tilde{x}^2]$ in $2D$). To reconstruct the velocity field, a cubic spline is used as the weight function. In $1D$, the cubic spline is

$$w(x - x_p) = \begin{cases} \frac{2}{3} - 4r_x^2 + 4r_x^3 & r_x \leq \frac{1}{2} \\ \frac{4}{3} - 4r_x + 4r_x^2 - \frac{4}{3}r_x^3 & \frac{1}{2} \leq r_x \leq 1 \\ 0 & r_x \geq 1 \end{cases}, \tag{5.1}$$

where $r_x = \frac{1}{2h}|x - x_p|$ and $h$ is the grid spacing. To reconstruct the density and stress field, the quadratic spline is used as the weight function. In $1D$, the quadratic spline is

$$w(x - x_p) = \begin{cases} \frac{3}{4} - r_x^2 & r_x \leq \frac{1}{2} \\ \frac{1}{2}(\frac{3}{2} - r_x)^2 & \frac{1}{2} \leq r_x \leq \frac{3}{2} \\ 0 & r_x \geq \frac{3}{2} \end{cases}, \tag{5.2}$$

where $r_x = \frac{1}{h}|x - x_p|$. In higher dimensions, tensor products of weight functions are used, eg. in 2D

$$w(r) = w(r_x) \otimes w(r_y).$$

By utilizing the process in Section 3.4, velocity, density and stress can be constructed. The following expressions show the final expressions:

$$v^h(x) = \sum_{p=1}^{npt} \Phi_p(x) v_p. \tag{5.3}$$

$$\rho^h(x) = \sum_{p=1}^{npt} Q_p(x) \rho_p. \tag{5.4}$$

$$\sigma^h(x) = \sum_{p=1}^{npt} Q_p(x) \sigma_p. \tag{5.5}$$

Here $\Phi_p(x)$ is the basis function constructed from cubic splines, and $Q_p(x)$ is the basis function constructed from quadratic splines. From this construction, the nodal velocity, cell centered stress and cell centered density can be derived as

$$v_i = v^h(x_i) = \sum_{p=1}^{npt} \Phi_p(x_i) v_p, \tag{5.6}$$

$$\rho_c = \rho^h(x_c) = \sum_{p=1}^{npt} Q_p(x_c) \rho_p, \tag{5.7}$$

$$\sigma_c = \sigma^h(x_c) = \sum_{p=1}^{npt} Q_p(x_c) \sigma_p, \tag{5.8}$$

where in 1D, $x_c = \frac{1}{2}(x_i + x_{i+1})$, with $c = i + \frac{1}{2}$, and in 2D, $x_c$ is the center of the 4-noded element.

**Calculating the Nodal forces and Nodal Masses Using One Point Quadrature.** By the definition of the nodal forces and nodal masses (i.e. equation (4.14)), nodal forces and masses can be approximated using a one point quadrature rule as follows:

$$f_i^{\text{int}} = -\int_\Omega \sigma \cdot \nabla N_i(x) dx = -\sum_{c=1}^{n_c} \nabla N_i(x_c) \cdot \sigma_c \Omega_c, \tag{5.9}$$

$$b_i = \int_\Omega N_i(x) \rho b dx = \sum_{c=1}^{n_c} N_i(x_c) \rho_c b(x_c) \Omega_c, \tag{5.10}$$

$$\hat{\tau}_i = \int_\Omega N_i \tau ds, \tag{5.11}$$

$$f_i^{\text{ext}} = b_i + \hat{\tau}_i, \tag{5.12}$$

$$m_{ij} = \int_\Omega \rho N_i(x) N_j(x) dx = \sum_{c=1}^{n_c} \rho_c^h N_i(x_c) N_j(x_c) \Omega_c, \tag{5.13}$$

where $\Omega_c$ is the volume of the element and $n_c$ is the total number of elements. The subscript $c$ means that the reconstructed function is evaluated at the element center. If the lumped mass is used, the lumped nodal mass is

$$\begin{aligned} m_i &= \int_\Omega \rho N_i(x) dx, \\ &= \sum_{c=1}^{n_c} \rho_c N_i(x_c) \Omega_c. \end{aligned} \tag{5.14}$$

## 5.4 Summary of the IMPM Algorithm

The main steps of the IMPM algorithm is summarized as follows.

(1) Map the information from material points to the grid and element center.

$$v_i^{s-\frac{1}{2}} = (v^h)^{s-\frac{1}{2}}(x_i) = \sum_{p=1}^{npt} \Phi_p^s(x_i) v_p^{s-\frac{1}{2}}, \tag{5.15}$$

$$\rho_c^s = (\rho^h)^s(x_c) = \sum_{p=1}^{npt} Q_p^s(x_c)\rho_p^s, \tag{5.16}$$

$$\sigma_c^s = (\sigma^h)^s(x_c) = \sum_{p=1}^{npt} Q_p^s(x_c)\sigma_p^s. \tag{5.17}$$

(2) Solve the equation of motion on the grid and update the grid information.

$$m_i^s a_i^s = (f_i^{\text{int}})^s + (f_i^{\text{ext}})^s, i = 1, 2, \cdots, n, \tag{5.18}$$

$$v_i^{s+\frac{1}{2}} = v_i^{s-\frac{1}{2}} + a_i^s \Delta t, \tag{5.19}$$

where $(f_i^{\text{int}})^s$, $(f_i^{\text{ext}})^s$ and $m_i^s$ are given by equations (5.9), (5.12) and (5.14) respectively.

(3) Update the information on the material points.

$$v_p^{s+\frac{1}{2}} = v_p^{s-\frac{1}{2}} + \Delta t \sum_{i=1}^{n} N_i(x_p^s) a_i^s, \tag{5.20}$$

$$x_p^{s+1} = x_p^s + \Delta t \sum_{i=1}^{n} N_i(x_p^s) v_i^{s+\frac{1}{2}}, \tag{5.21}$$

$$u_p^{s+1} = x_p^{s+1} - x_p^0, \tag{5.22}$$

$$F_p^{s+1} = (\sum_{i=1}^{n} \nabla N_i(x_p^{s+1}) v_i^{s+\frac{1}{2}} \Delta t + I) F_p^s, \tag{5.23}$$

$$\rho_p^{s+1} = \frac{\rho_0}{|F_p^{s+1}|}, \tag{5.24}$$

$$\sigma_p^{s+1} = \sigma(F_p^{s+1}). \tag{5.25}$$

(4) Generate a new grid.

Specifically, the steps of the algorithm are implemented as follows in $1D$. First, we need to initialize the grid. The algorithm is presented for the manufactured solution

in Section 4.5. The computational domain is given by the interval $[0, 1]$. The total number of elements is denoted by $N_e$. For equally spaced nodes, the grid spacing is $h = \frac{1}{Ne}$. Let $x$ be a vector of length $N_e + 1$ for the grid-point positions and $x_c$ be a vector of length $N_e$ for the element-center positions. The grid points are computed as $x_i = (i-1)h$, $i = 1, 2, \cdots, N_e+1$. The cell-center points are computed as $x_c(i) = \frac{1}{2}(x_i + x_{i+1})$, $i = 1, 2, \cdots, N_e$. Let *npcell* be the number of material points per cell, then $dpt = \frac{h}{npcell}$ is the initial material-point volume. In $1D$, the quantity *dpt* is also the spacing between the material points. The material points are generated initially by looping over the elements and for each element, $[x_i, x_{i+1}]$, creating *npcell* equally spaced points at positions $x_i + (j-1)dpt$, $j = 1, 2, \cdots, npcell$, $i = 1, 2, \cdots, N_e$. The material-point positions are kept as a list in a vector $x_p$ of length $np = npcell \cdot Ne$.

Let $x0$ be a vector of length $np$ for the initial positions of the material points, $v_p$ be a vector of length $np$ for the velocity of the material points, *stress* be a vector of length $np$ for the stress of the material points, *density* be a vector of length $np$ for the density of the material points, *mass* be a vector of length $np$ for the mass of the material points, *strain* be a vector of length $np$ for the strain of the material points, $u_p$ be a vector of length $np$ for the displacement of the material points, $FP$ be a vector of length $np$ for the deformation gradient of the material points, and *body* be a vector of length $np$ for the body force on the material points. These arrays are initialized using the initial conditions for the problem.

For convenience, at the beginning of each time step, we store the element number in which each material point resides and its natural coordinate. The computation loops over material points, $p = 1, 2, \cdots, np$. Within this loop set $xpt = x_p(p)$ to the position of material point $p$. The element number of the element containing this point is $nepl(p) = fix(\frac{xpt}{h}) + 1$ and the natural coordinate of the material point is $\xi_p = (\frac{xpt - x_{nepl(p)}}{h})$. Each of the vectors *nepl* and $\xi$ have length $np$.

Now the time step begins by mapping information from the material points to grid points. The implementation of equation (5.15) is accomplished by first constructing the shape functions $\Phi_p(x_i)$ evaluated at each grid point $x_i$ and then by evaluating the sum. Since $\Phi_p(x)$ is given by equations (3.16), (3.17) and (3.18) with the weight function given by equation (5.1), it is first necessary to form a moment matrix for each grid point, $M(x_i)$, and the vector $B(x_i)$ for each grid point. These arrays are assembled by looping over material points and determining the contribution each material point makes to the arrays. Next we describe the assembly process.

Cubic splines (5.1) have compact support with support size $2h$. Thus, given a material point $x_p$, there are four grid points for which $w(x_i - x_p)$ is nonzero in the sum (3.17) for the moment matrix $M(x_i)$. Similarly, four components of $B(x_i)$ have contributions from $x_p$. The four grid points are determined as follows. Given a material point $x_p$, it resides in element $i = nepl(p)$, the four grid points are $ni(1) = i - 1$, $ni(2) = i$, $ni(3) = i + 1$ and $ni(4) = i + 2$. The radius $r_x = |\frac{x-x_p}{2h}|$ used in the weight function (5.1) at the four grid points $x = x_{ni(k)}$, $k = 1, 2, 3, 4$ for a given material point $p$ is computed as $r_x(1) = \frac{1}{2}(\xi_p + 1)$, $r_x(2) = \frac{\xi_p}{2}$, $r_x(3) = \frac{1}{2}(1 - \xi_p)$ and $r_x(4) = 1 - \frac{\xi_p}{2}$. Then the weight function for the four grid points is computed by evaluating equation (5.1) at $r_x(1)$, $r_x(2)$, $r_x(3)$ and $r_x(4)$ and the outputs are saved as $w_3(1) = \frac{(1-\xi_p)^3}{6}$, $w_3(2) = \frac{3}{2} + \xi_p^2(-1 + 0.5\xi_p)$, $w_3(3) = \frac{1}{6} + \xi_p(0.5 + \xi_p(0.5 - 0.5\xi_p))$ and $w_3(4) = \frac{\xi_p^3}{6}$. The polynomial basis $P^T(x) = [1, x - x_p]$ has to be computed in order to construct the moment matrix $M(x_i)$ and the vector $B(x_i)$. Since the first component of the polynomial basis is 1, it can be computed trivially. The second component of the polynomial basis at the four grid points $x_{ni(k)}$, $k = 1, 2, 3, 4$ is computed as $aijx(1) = x_{ni(1)} - x_p$, $aijx(2) = x_{ni(2)} - x_p$, $aijx(3) = x_{ni(3)} - x_p$ and $aijx(4) = x_{ni(4)} - x_p$. Initialize $M = zeros(3, N_e+1)$ and $B = zeros(2, N_e+1)$, where $M(1, i)$ is the 11 component of the moment matrix, $M(2, i)$ is the 12 component of the moment matrix, $M(3, i)$ is the 12 and 21 component of the moment matrix, $B(1, i)$ is the first component of the vector, and $B(2, i)$ is the second component of

the vector, $i = 1, 2, \cdots, N_e + 1$. The assembly of the moment matrix $M$ and the vector $B$ at each grid point is given as

**Algorithm 1** Assembling of moment matrix $M$ and $B$ on the grid points

1: *loop* $p = 1, np$
2: Get the element in which the material point $x_p$ is located from *nepl*.
3: Compute the weight function $w_3$ for the 4 grid points to which the material point contributes.
4: Compute the 4 grid-point numbers saved in $ni$ to which the material point contributes.
5: Assemble the moment matrix $M$ and $B$ at each grid point.
6: *loop* $j = 1, 4$
$aijx(j) = x(ni(j)) - x_p,$
$M(1, ni(j)) = M(1, ni(j)) + w3(j),$
$M(2, ni(j)) = M(2, ni(j)) + w3(j)aijx(j),$
$M(3, ni(j)) = M(3, ni(j)) + w3(j)aijx(j)^2,$
$B(1, ni(j)) = B(1, ni(j)) + w3(j),$
$B(2, ni(j)) = B(2, ni(j)) + w3(j)ajix(j).$
7: *end loop* $j$
8: *end loop* $p$.

After we have assembled the moment matrix $M$ and the vector $B$, we need to solve a linear system $M(x)a(x) = B(x)U$, where $U^T = [v_p(1), v_p(2), \cdots, v_p(np)]$, at each grid point $x_i$, $i = 1, 2, \cdots, N_e + 1$. Since $M$ is a $2 \times 2$ matrix (in $1D$) at each grid point, use gaussian elimination to solve for the coefficient $a(x)$, where $a(x)$ is a vector of length two at each grid points (i.e. $a$ has dimensions $(2, N_e + 1)$).

Recall that the shape function $\Phi_p(x_i)$ that appears in equation (5.15) is given by $\Phi_p(x_i) = P^T(x_i)a(x_i)$. The sum in (5.15) is evaluated by looping over the material points. The assembly process is given as follows:

**Algorithm 2** Assembling of the basis function $\Phi_p(x_i)$ on the grid points

1: *loop* $p = 1, np$
2: Get the element number in which the material point $x_p$ is located from *nepl*.
3: Compute the 4 grid-point numbers saved in $ni$ to which the material point contributes.
4: *loop* $j = 1, 4$
$aijx(j) = x(ni(j)) - x_p,$
$v(ni(j)) = v(ni(j)) + a(1, ni(j)) + aijx(j)a(2, ni(j)),$
5: *end loop* $j$
6: *end loop* $p$.

At the end, boundary conditions have to be applied to $v(1)$ and $v(N_e + 1)$, which are zero for the manufactured solution.

The support of the quadratic spline given in equation (5.2) is $\frac{3}{2h}$. Thus, in evaluating (5.16) and (5.17), the shape function $Q_p(x_c)$ is assembled analogously to $\Phi_p(x_i)$ with each material point contributing to three element centers.

Let *densityc* be a vector of length $N_e$ for the density at the element centers. It is computed using equation (5.16). Let *stressc* be a vector of length $N_e$ for the stress at the element centers. It is computed using equation (5.17). Let *bodyc* be a vector of length $N_e$ for the body force at the element centers. It is computed using either equation (5.16) or equation (5.17) by replacing either $\rho_p$ or $\sigma_p$ with *body*$(p)$. Let $m$, $fint$, $fext$ and $f$ be vectors of length $N_e + 1$. The nodal masses $m(i)$, internal forces $fint(i)$ and external forces $fext(i)$, $i = 1, 2, \cdots, N_e + 1$ can be computed as follows. Loop over element centers, $i = 1, 2, \cdots, N_e$. At each center point $x_{i+\frac{1}{2}}$, compute the linear hat basis functions defined on the two nearby grid points at $N_i(x_{i+\frac{1}{2}})$ and $N_{i+1}(x_{i+\frac{1}{2}})$, which equal $\frac{1}{2}$. Then compute the gradient of $\nabla N_i(x_{i+\frac{1}{2}})$ and $\nabla N_{i+1}(x_{i+\frac{1}{2}})$, which are given as $\nabla N_i(x_{i+\frac{1}{2}}) = -h$ and $\nabla N_{i+1}(x_{i+\frac{1}{2}}) = h$. The nodal masses, internal forces and external forces are assembled by $m(i) = m(i) +$

$\frac{1}{2}densityc(i)$, $m(i+1) = m(i+1) + \frac{1}{2}densityc(i)$, $fint(i) = fint(i) + stressc(i)$, $fint(i+1) = fint(i+1) - stressc(i)$, $fext(i) = fext(i) + \frac{1}{2}bodyc(i)densityc(i)$ and $fext(i+1) = fext(i+1) + \frac{1}{2}bodyc(i)densityc(i)$. The total forces are computed as the sum of the internal forces and external forces, $f(i) = fint(i) + fext(i)$, $i = 2, \cdots, N_e$.

Equations (5.18) and (5.19) are implemented as follows. Let *accel* be a vector of length $N_e+1$ for the acceleration on the grid points and v be a vector of length $N_e+1$ for the velocity on the grid points. Loop over the grid points, $i = 1, 2, \cdots, N_e + 1$, if $m(i) = 0$, set $accel(i) = 0$ and $v(i) = 0$, otherwise $accel(i) = \frac{f(i)}{m(i)}$. Then use $accel(i)$ to update the velocity vector $v(i)$ according to equation (5.19). Apply boundary conditions to $v(1)$ and $v(N_e + 1)$, which are given as $v(1) = 0$ and $v(N_e + 1) = 0$.

Equations (5.20), (5.21) and (5.22) are implemented as follows. Loop over the material points, $p = 1, 2, \cdots, np$. Get the natural coordinate of each material point, $xip = xi(p)$. Get the element number in which the material point is located, $k = nepl(p)$. Update the position of the material point to $x_p(p) + \Delta t(v(k)(1 - xip) + v(k + 1)xip)$. Update the velocity of the material point to $v_p(p) + \Delta t(accel(k)(1 - xip) + accel(k + 1)xip)$. Update the displacement of the material point, $u_p(p) = x_p(p) - x0(p)$.

Equations (5.23), (5.24) and (5.25) are implemented as follows. Loop over the material points, $p = 1, 2, \cdots, np$. Get the element number in which each material point is located, $k = nepl(i)$. The deformation gradient of the material points is updated to $(1 + \frac{\Delta t}{h}(v(k + 1) - v(k)))FP(p)$. The density of the material points is updated as $density(p) = \frac{\rho_0}{|FP(p)|}$, where $\rho_0$ is the initial density and is a constant. The stress on the material points is updated as $stress(p) = \sigma(FP(p))$.

## 5.5 Convergence of IMPM in $1D$

With the same manufactured solutions and parameters as in Section 4.6, the convergence of IMPM is given in the following graphs. The axes are log base 10 of the $L_2$ norm of the error vs. the log base 10 of the mesh size. The slope of the line that approximates the data reveals the convergence rate. The $L_2$ error is defined as

$$E_{l2} = (\int_{\Omega} (u_n^{\text{num}}(x) - u_n^{\text{ex}}(x))^2 d\Omega)^{\frac{1}{2}}, \tag{5.26}$$

where $u_n^{\text{num}}$ is the numerical solution and $u_n^{\text{ex}}$ is the analytical solution at time $t^n$. The error is computed using one point quadrature over the material-point domain for the stress, density, velocity and displacement, and over the element domain for the velocity. Since the analytical solution for the velocity is only known on the material points, second order MLS can be used to map the analytical velocity on the material points to the grid so that the analytical solution of velocity on the grid can be approximated and the error of velocity on the grid can be computed.

In Figures 5.5 and 5.6, the top line is the error in the stress on the material points, the second line is the error in the density on the material points, the third line is the error in the velocity on the material points and on the grid, and the fourth line is the error in the displacement on the material points.

For a small deformation problem, where $G = 0.0001$, IMPM has a similar convergence rate to original MPM. For a large deformation problem, where $G = 0.1$, IMPM has significant improvements over MPM for stress, density, velocity and displacement. As one can see in Figures 5.5 and 5.6, IMPM has second order convergence both for small deformation and large deformation problems.

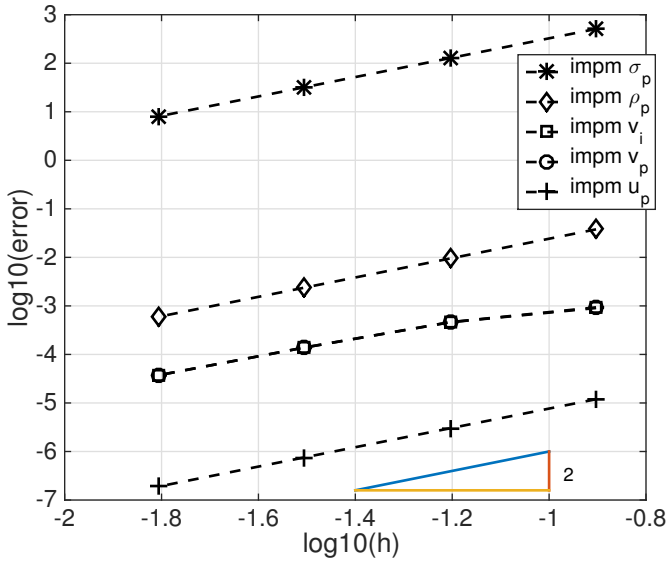


Figure 5.5: Convergence of IMPM for the small deformation problem in $1D$, G=0.0001.

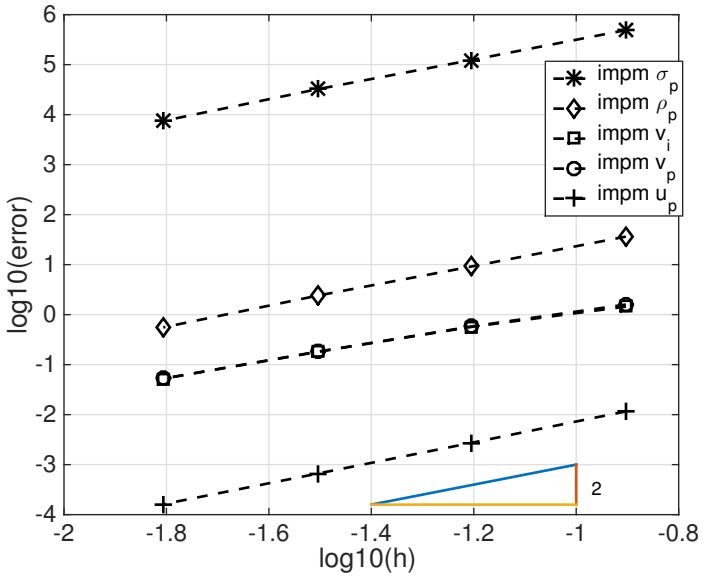


Figure 5.6: Convergence of IMPM for the large deformation problem in $1D$, G=0.1.

## 5.6 Convergence of IMPM in $2D$

With the same manufactured solutions and parameters as those in Section 4.7, the convergence of IMPM in the $2D$ is given in the following graphs. The error of the density is computed using equation (5.26). The $L_2$ error of displacement is computed to be

$$E_{l2} = (\int_\Omega ((u_n^1)^{\text{num}} - (u_n^1)^{\text{ex}})^2 + ((u_n^2)^{\text{num}} - (u_n^2)^{\text{ex}})^2 dx)^{\frac{1}{2}}, \tag{5.27}$$

where $(u_n^1)^{\text{num}}$ and $(u_n^2)^{\text{num}}$ are the numerical values of the first and second components of the displacement, and $(u_n^1)^{\text{ex}}$ and $(u_n^2)^{\text{ex}}$ are the exact values of the first and second components of the displacement at time $t^n$. The $L_2$ error in the velocity is computed similarly. The error in stress is computed to be

$$\begin{aligned} E_{l2} = (\int_\Omega &((\sigma_n^{11})^{\text{num}} - (\sigma_n^{11})^{\text{ex}})^2 + ((\sigma_n^{12})^{\text{num}} - (\sigma_n^{12})^{\text{ex}})^2 \\ &+((\sigma_n^{21})^{\text{num}} - (\sigma_n^{21})^{\text{ex}})^2 + ((\sigma_n^{22})^{\text{num}} - (\sigma_n^{22})^{\text{ex}})^2 dx)^{\frac{1}{2}}, \end{aligned} \tag{5.28}$$

where $(\sigma_n^{11})^{\text{num}}$, $(\sigma_n^{12})^{\text{num}}$, $(\sigma_n^{21})^{\text{num}}$, and $(\sigma_n^{22})^{\text{num}}$ are the 11, 12, 21 and 22 components of the numerically computed stress tensor, and $(\sigma_n^{11})^{\text{ex}}$, $(\sigma_n^{12})^{\text{ex}}$, $(\sigma_n^{21})^{\text{ex}}$, and $(\sigma_n^{22})^{\text{ex}}$ are the 11, 12, 21 and 22 components of the analytical solution for the stress tensor at time $t^n$. The above errors are computed using one point quadrature over the material-point domain and grid domain respectively. Since we only know the analytical solution of velocity on the material points, we use second order MLS to map the analytical velocity on the material points to the grid so that we can approximate the analytical solution of velocity on the grid and compute the error of velocity on the grid.

In Figures 5.7 and 5.8, the top line shows errors in the stress tensor on the material points, the second line is the error in the density on the material points, the third line is the error in the velocity on the material points and on the grid, and the fourth line is the error in the displacement on the material points.

For both the small deformation problem and the large deformation problem as in Figure 5.7 and Figure 5.8, where $G = 0.0001$ and $G = 0.1$ respectively, IMPM has dramatic improvements in terms of convergence over MPM in $2D$ case. In IMPM not only the displacement has second order convergence, but all other quantities have close to second order convergence.

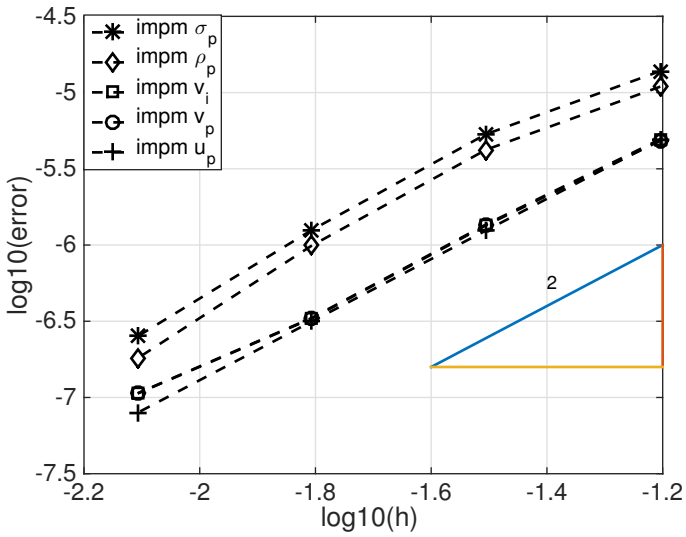


Figure 5.7: Convergence of IMPM for the small deformation problem in $2D$, G=0.0001.

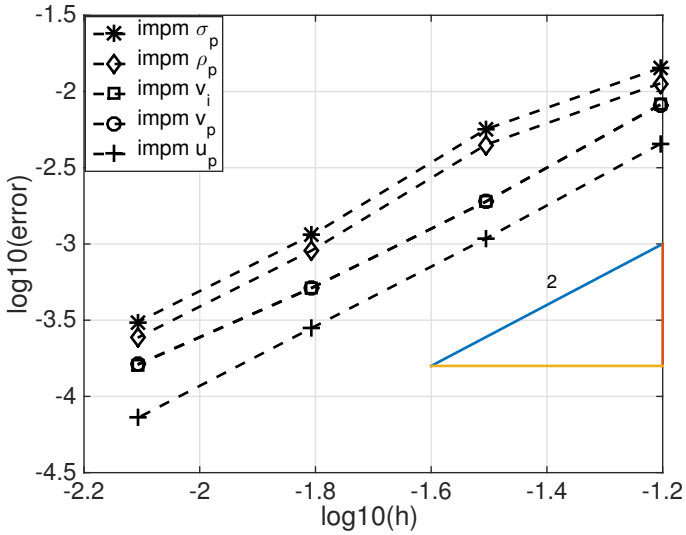


Figure 5.8: Convergence of IMPM for the large deformation problem in $2D$, G=0.1.

# Chapter 6

# Variations of MPM and Suggested Improvements

## 6.1 Introduction

Some improvements of MPM have been proposed in the last ten years including various versions of the Generalized Interpolation Material Point Method (GIMP) [24, 26] and various versions of the Convected Particle Domain Interpolation Method (CPDI) [1, 2]. The philosophy of these methods is to improve MPM by improving the accuracy of the approximation of nodal masses and nodal forces. Such a procedure is accomplished by introducing the characteristic function $\chi_p$ over the material point domain centered at position $x_p$ and by more accurately keeping track of the geometry of the material volume centered at $x_p$. Such characteristic functions form a partition of unity:

$$\sum_{p=1}^{N_p} \chi_p(x,t) = 1. \tag{6.1}$$

Based on the above identity, the entire material body can be decomposed by

$$\int_{\Omega(t)} dx = \sum_{p=1}^{N_p} \int_{\Omega_p(t)} \chi_p(x,t) dx, \tag{6.2}$$

where $\Omega_p$ is the material volume centered at current position $x_p$. In order to compute the right-hand side of the above integral, two approximations need to be made. Firstly, the characteristic function $\chi_p(x,t)$ has to be approximated properly; secondly, the material volume $\Omega_p(x,t)$ has to be approximated more accurately.

In the original MPM algorithm, $\chi_p(x,t) = \delta_p(x - x_p)$ is chosen, and the material volume is computed as $\Omega_p(x,t) = m_p/\rho_p(x,t)$. Such approximation results in low order approximation to nodal forces and nodal masses.

By contrast, GIMP and CPDI choose the following characteristic function instead

$$\chi_p(x,t) = \begin{cases} 1, & x \in \Omega_p(t) \\ 0, & \text{otherwise} \end{cases} . \tag{6.3}$$

The main difference between GIMP and CPDI is the method of approximating the evolving geometry of the material volume $\Omega_p(x,t)$ in time. The evolving geometry is ignored in MPM and only the evolving volume of the material-point domain is computed.

In GIMP, there are two methods of approximating the geometry of the material volume $\Omega_p(x,t)$. One method is called unchanged GIMP (uGIMP), where $\Omega_p(x,t) = \Omega_p(x,0)$ is kept fixed throughout the entire computation. This approximation is inexpensive computationally, but obviously inaccurate. The second method is called contiguous particle GIMP (cpGIMP), in which the geometry of the material domain is updated using the diagonal part of the deformation gradient tensor $F$. Such approximation takes into account the changes of the geometry of the material domain in the axis aligned directions. In other words, it changes height and width. This approximation does not accurately model material geometries which undergo shear motion.

In CPDI, there are two approximations of the evolving geometry of the material domain, which result in CPDI1 and CPDI2. In CPDI1, the evolution of the geometry of the material domain is approximated through the local tangent affine deformation mapping to map initially parallelogram-shaped material domains into new parallelograms. Such approximation takes into account the shear motion of the material domain but restricts itself to the geometry of parallelograms. By contrast, in CPDI2, the evolution of the geometry of the material domain is further improved by updating the geometry of the material domain as $4-$node quadrilateral elements in two dimensions and $8-$node hexahedral elements in three dimensions.

In this chapter, the general framework and formulations of GIMP and CPDI are reviewed and the improved CPDI is proposed. A comparison of order of convergence among CPDI, IMPM and improved CPDI is studied in an $1D$ example.

## 6.2 General framework of GIMP and CPDI

There are four main steps of GIMP and CPDI, which are the same as the four main steps of MPM. They are listed as follows.

(1) Map information from material points to the grid.

(2) Solve the momentum equation on the background grid.

(3) Update material-point information.

(4) Generate a new grid.

Since the fourth step in GIMP and CPDI is exactly the same as MPM, only the details of the first three steps will be illustrated.

An effective shape function $\bar{S}_{ip}$ is defined as the convolution of the characteristic function $\chi_p$ and nodal basis function $N_i(x)$ (i.e. the same nodal basis functions as in MPM) as follows:

$$\bar{S}_{ip} = \frac{1}{\Omega_p(t)} \int_{\Omega_p \cap \Omega} \chi_p(x) N_i(x) dx, \tag{6.4}$$

where $\Omega$ is the support of the whole material volume, $\Omega_p$ is the support of the material point volume, and $\Omega_p(t)$ is the material volume. The gradient of the effective basis function is defined as

$$\nabla \bar{S}_{ip} = \frac{1}{\Omega_p(t)} \int_{\Omega_p \cap \Omega} \chi_p(x) \nabla N_i(x) dx. \tag{6.5}$$

(1). **Map information from material points to the background grid.**

$$m_i = \sum_{p=1}^{N_p} \bar{S}_{ip} M_p, \tag{6.6}$$

$$(mv)_i = \sum_{p=1}^{N_p} \bar{S}_{ip} M_p v_p, \tag{6.7}$$

$$v_i = \frac{(mv)_i}{m_i}. \tag{6.8}$$

(2). **Solve the discretized weak form of the equations of motion on the background grid.**

$$m_i a_i = f_i^{\text{int}} + f_i^{\text{ext}}, \tag{6.9}$$

$$v_i^{s+\frac{1}{2}} = v_i^{s-\frac{1}{2}} + a_i \Delta t. \tag{6.10}$$

In the above equations, the leapfrog scheme is used to update the grid velocity instead of the Forward Euler scheme as in the original GIMP and CPDI methods.

The internal nodal forces $f_i^{\text{int}}$ and external nodal forces are computed as follows:

$$f_i^{\text{int}} = -\sum_{p=1}^{N_p} \nabla \bar{S}_{ip} \sigma_p \Omega_p, \tag{6.11}$$

$$f_i^{\text{ext}} = \sum_{p=1}^{N_p} \bar{S}_{ip} b_p M_p. \tag{6.12}$$

(3). **Update material-point information.** The material-point positions and velocities are updated by

$$x_p^{s+1} = x_p^s + \sum_{i=1}^{n} \bar{S}_{ip} v_i^{s+\frac{1}{2}} \Delta t, \tag{6.13}$$

$$v_p^{s+\frac{1}{2}} = v_p^{s-\frac{1}{2}} + \sum_{i=1}^{n} \bar{S}_{ip} (v_i^{s+\frac{1}{2}} - v_i^{s-\frac{1}{2}}). \tag{6.14}$$

The velocity gradients at the material points are computed as

$$\nabla v_p^{s+\frac{1}{2}} = \sum_{i=1}^{n} \nabla \bar{S}_{ip} v_i^{s+\frac{1}{2}}. \tag{6.15}$$

The deformation gradient and stress tensors on the material points are updated as

$$F_p^{s+1} = (1 + \nabla v_p^{s+\frac{1}{2}} \Delta t) F_p^s, \tag{6.16}$$

$$\sigma_p^{s+1} = \sigma(F_p^{s+1}), \tag{6.17}$$

$$\Omega_p^{s+1} = \Omega_p^0 |F_p^{s+1}|. \tag{6.18}$$

## 6.3 GIMP formulation

In GIMP, the effective basis function $\bar{S}_{ip}$ in $1D$ is constructed as follows:

$$\bar{S}_{ip} = \bar{S}_i(x_p) = \frac{1}{\Omega_p(t)} \int_{\Omega_p \cap \Omega} \chi_p(x) N_i(x) dx, \tag{6.19}$$

where the basis functions $N_i(x)$ in $1D$ are defined as follows:

$$N_i(x) = \begin{cases} 1 + \frac{x - x_i}{h}, & -h < x - x_i \leq 0 \\ 1 - \frac{x - x_i}{h}, & 0 < x - x_i \leq h \\ 0, & \text{otherwise.} \end{cases} \tag{6.20}$$

The support of the material volume $\Omega_p$, which characterizes the local geometry of each material point, is defined as $|x - x_p| \leq l_p$, and the characteristic function $\chi_p(x)$ equals 1 when its domain lies in the support of the material volume $\Omega_p$ and is zero otherwise. By expanding the above effective basis functions, the following can be derived:

$$\bar{S}_{ip} = \bar{S}_i(x_p) = \begin{cases} \frac{(h + l_p + x_p - x_i)^2}{4hl_p}, & -h - l_p < x_p - x_i \leq -h + l_p \\ 1 + \frac{x_p - x_i}{h}, & -h + l_p < x_p - x_i \leq -l_p \\ 1 - \frac{(x_p - x_i)^2 + l_p^2}{2hl_p}, & -l_p < x_p - x_i \leq l_p \\ 1 - \frac{x_p - x_i}{h}, & l_p < x_p - x_i \leq h - l_p \\ \frac{(h + l_p - (x_p - x_i))^2}{4hl_p}, & h - l_p < x_p - x_i \leq h + l_p \\ 0, & \text{otherwise.} \end{cases}$$

Its gradient is computed as

$$\bar{S}_{ip} = \bar{S}_i(x_p) = \begin{cases} \frac{h + l_p + x_p - x_i}{2hl_p}, & -h - l_p < x_p - x_i \leq -h + l_p \\ \frac{1}{h}, & -h + l_p < x_p - x_i \leq -l_p \\ -\frac{x_p - x_i}{hl_p}, & -l_p < x_p - x_i \leq l_p \\ -\frac{1}{h}, & l_p < x_p - x_i \leq h - l_p \\ \frac{h + l_p - (x_p - x_i)}{2hl_p}, & h - l_p < x_p - x_i \leq h + l_p \\ 0, & \text{otherwise.} \end{cases}$$

In the above equations, $h$ is the same as the grid spacing in MPM. The above effective basis functions are one of the ingredients of GIMP; the second ingredient of GIMP is the approximation of the geometry of the material volume, which defines the support size of the characteristic function $\chi_p$. In uGIMP, $\Omega_p(t) = \Omega_p(0)$ is kept fixed, so it is computationally cheap. By contrast, in cpGIMP, the geometry of each

material point domain is updated from a rectangular shape to rectangular shape. The updated material point domain is therefore an axis-aligned rectangle in $2D$ without shear motions taken into account; such approximations can model materials which undergo axis-aligned motion, but can't handle materials which undergo rotations and shear motions.

## 6.4 CPDI1 formulation

In CPDI1, the initial particle domain is discretized as a parallelogram described by two vectors $(r_1^0, r_2^0)$ with the centroid as the material point $x_p$. At each step $s$, the deformed particle domain is approximated by using the fully updated deformation gradient tensor as follows:

$$\begin{aligned} r_1^s &= F_p^s r_1^0, \\ r_2^s &= F_p^s r_2^0. \end{aligned} \tag{6.21}$$

The corresponding effective basis functions and their gradients are defined as

$$\bar{S}_{ip} = \frac{1}{\Omega_p(t)} \int_{\Omega_p} S_i^p(x) dx, \tag{6.22}$$

$$\nabla \bar{S}_{ip} = \frac{1}{\Omega_p(t)} \int_{\Omega_p} \nabla S_i^p(x) dx, \tag{6.23}$$

where

$$S_i^p(x) = \begin{cases} \sum_{\alpha=1}^4 N_\alpha^p(x) N_i(x_\alpha^p), & x \in \Omega_p \\ 0, & \text{otherwise} \end{cases}, \tag{6.24}$$

where $N_i(x)$ uses the same FEM nodal basis functions as MPM, and $N_\alpha^p(x)$ is the linear finite element nodal basis functions defined on the particle domain as a four-node element. For both CPDI1 and CPDI2, $N_\alpha^p(x)$ are defined on the $\alpha^{th}$ corner of the particle domain with $x_\alpha^p$ being the position of this corner, having the following

property:

$$N_\alpha^p(x_\beta^p) = \begin{cases} 1, & \alpha = \beta \\ 0, & \text{otherwise} \end{cases} . \tag{6.25}$$

Based on the above property of the finite element basis function $N_\alpha^p$, the effective basis function can be computed as

$$\begin{aligned} \bar{S}_{ip} &= \frac{1}{\Omega_p(t)} \int_{\Omega_p} S_i^p(x) dx \\ &= \frac{1}{\Omega_p(t)} \sum_{\alpha=1}^{4} N_i(x_\alpha^p) \int_{\Omega_p} N_\alpha^p(x) dx \\ &= \frac{1}{4}(N_i(x_1^p) + N_i(x_2^p) + N_i(x_3^p) + N_i(x_4^p)). \end{aligned} \tag{6.26}$$

Its gradient can be computed as

$$\begin{aligned} \nabla \bar{S}_{ip} &= \frac{1}{\Omega_p(t)} \int_{\Omega_p} \nabla S_i^p(x) dx, \\ &= \frac{1}{\Omega_p(t)} \sum_{\alpha=1}^{4} N_i(x_\alpha^p) \int_{\Omega_p} \nabla N_\alpha^p(x) dx, \\ &= \frac{1}{2\Omega_p(t)} \left[ (N_i(x_1^p) - N_i(x_3^p)) \begin{bmatrix} r_{1y} - r_{2y} \\ r_{2x} - r_{1x} \end{bmatrix} + (N_i(x_2^p) - N_i(x_4^p)) \begin{bmatrix} r_{1y} + r_{2y} \\ -r_{1x} - r_{2x} \end{bmatrix} \right], \end{aligned} \tag{6.27}$$

where $(r_{1x}, r_{1y})$ and $(r_{2x}, r_{2y})$ are respectively the components of vectors $r_1$ and $r_2$.

## 6.5 CPDI2 formulation

In CPDI2, the material domains are initialized as quadrilaterals, and they evolve as quadrilaterals to quadrilaterals. Such a process is accomplished by generating a quadrilateral mesh initially, in which the location of each material point is the centroid of each element defining the material-point domain, and the evolving material geometry is computed by tracking the positions of the corners of the quadrilateral

material-point domain. Instead of updating the material-point positions as in the CPDI1, GIMP, and MPM algorithms, an update of the positions of the material-point corners is needed at each time step. These are given as

$$x_\alpha^{s+1} = x_\alpha^s + \sum_{i=1}^{n} N_i(x_\alpha) v_i^{s+\frac{1}{2}} \Delta t, \tag{6.28}$$

where $N_i(x)$ is the original MPM nodal basis functions on the background grid, and $x_\alpha$ is the position of the corner node $\alpha$. Here the letter $p$ is still referred to a material point, and $\alpha$ is referred to the corner of the material point domain. If the positions of the material points are needed, the material-point position can be obtained by averaging the material-point corners' positions. For example,

$$\begin{aligned} x_p &= \frac{1}{\Omega_p(t)} \int_{\Omega_p} x dx, \\ &= \frac{1}{\Omega_p(t)} \sum_{\alpha=1}^{4} x_\alpha^p \int_{\Omega_p} N_\alpha^p(x) dx, , \\ &= \frac{1}{24\Omega_p(t)} ((1-a-b)x_1^p + (1-a+b)x_2^p + (1+a+b)x_3^p + (1+a-b)x_4^p), \end{aligned} \tag{6.29}$$

where $a = (x_4^p - x_1^p)(y_2^p - y_3^p) - (x_2^p - x_3^p)(y_4^p - y_1^p)$, $b = (x_3^p - x_4^p)(y_1^p - y_2^p) - (x_1^p - x_2^p)(y_3^p - y_4^p)$ and $x_i^p$, $i = 1$, 2, 3, and 4 are the positions of the four corners of the material-point domain $p$.

Since in CPDI2 the material-point domains are computed as quadrilaterals by keeping track of the positions of the corners of the material-point domain, the effective basis functions become

$$\begin{aligned} \bar{S}_{ip} &= \frac{1}{\Omega_p(t)} \int_{\Omega_p} S_i^p dx, \\ &= \frac{1}{\Omega_p(t)} \sum_{\alpha=1}^{4} N_i(x_\alpha^p) \int_{\Omega_p} N_\alpha^p(x) dx, \\ &= \frac{1}{24\Omega_p(t)} ((1-a-b)N_i(x_1^p) + (1-a+b)N_i(x_2^p) + \\ &(1+a+b)N_i(x_3^p) + (1+a-b)N_i(x_4^p)). \end{aligned} \tag{6.30}$$

Gradients of the shape functions become

$$\nabla \bar{S}_{ip} = \frac{1}{\Omega_p(t)} \int_{\Omega_p} \nabla S_i^p(x) dx, \tag{6.31}$$

$$= \frac{1}{\Omega_p(t)} \sum_{\alpha=1}^{4} N_i(x_\alpha^p) \int_{\Omega_p} \nabla N_\alpha^p(x) dx,$$

$$= \frac{1}{2\Omega_p(t)} \left[ \; N_i(x_1^p)\bar{A} + N_i(x_2^p)\bar{B} + N_i(x_3^p)\bar{C} + N_i(x_4^p)\bar{D} \; \right], \tag{6.32}$$

where $\bar{A} = \begin{bmatrix} y_2^p - y_4^p \\ x_4^p - x_2^p \end{bmatrix}$, $\bar{B} = \begin{bmatrix} y_3^p - y_1^p \\ x_1^p - x_3^p \end{bmatrix}$, $\bar{C} = \begin{bmatrix} y_4^p - y_2^p \\ x_2^p - x_4^p \end{bmatrix}$ and $\bar{D} = \begin{bmatrix} y_1^p - y_3^p \\ x_3^p - x_1^p \end{bmatrix}$.

As seen in the above equations, the CPDI2 formulation gives the best approximation of the evolving geometry of material-point domains and results in the best accuracy among MPM, CPDI, GIMP, CPDI1, and CPDI2. Both GIMP and CPDI formulations aim toward improving the accuracy of the geometry of the evolved material-point domain. Such improvements essentially lead to the improvements of the accuracy of nodal forces and nodal masses. However based on the general framework of the IMPM algorithm, simply improving the accuracy of the nodal forces and nodal masses, is not enough to improve the whole algorithm to be second order in space. The mapping of velocity from the material point field to the background grid field must also be improved. In the next section, the improved CPDI algorithm is presented using the techniques of Moving Least Squares.

## 6.6 Improve the mapping in CPDI using MLS

In this section, the illustration of the algorithm of CPDI and improved CPDI will be presented in $1D$. In $1D$, we don't distinguish the differences between CPDI1 and CPDI2, since they are equivalent. The effective basis functions of CPDI in $1D$ are

given as follows:

$$\begin{aligned}\bar{S}_{ip} &= \frac{1}{\Omega_p(t)}\int_{\Omega_p} S_i^p dx \\ &= \frac{1}{\Omega_p(t)}\sum_{\alpha=1}^{2} N_i(x_\alpha^p)\int_{\Omega_p} N_\alpha^p(x)dx \\ &= \frac{1}{2(x_2^p - x_1^p)}(N_i(x_1^p) + N_i(x_2^p)),\end{aligned} \tag{6.33}$$

where $N_i$ is given by equation (6.20). $\Omega_p(t) = x_2^p - x_1^p$ is the length of the material interval at time $t$, $x_2^p$ is the right corner of the material-point domain containing $x_p$, and $x_1^p$ is the left corner of the material-point domain containing $x_p$. Here $S_i$ is the same function as in equation (6.20). Correspondingly, the gradient of the effective basis function is given as

$$\begin{aligned}\nabla\bar{S}_{ip} &= \frac{1}{\Omega_p(t)}\int_{\nabla\Omega_p} S_i^p dx \\ &= \frac{1}{\Omega_p(t)}\sum_{\alpha=1}^{2} N_i(x_\alpha^p)\int_{\Omega_p} \nabla N_\alpha^p(x)dx \\ &= \frac{1}{(x_2^p - x_1^p)}(-N_i(x_1^p) + N_i(x_2^p)).\end{aligned} \tag{6.34}$$

With the above given effective basis functions and its gradients, the algorithm of CPDI is summarized as follows:

(1) The mapping from material points to grid.

$$\begin{aligned} m_i^s &= \sum_{p=1}^{N_p} \bar{S}_{ip} M_p, i = 1, 2, \cdots, n, \\ (mv)_i^{s-\frac{1}{2}} &= \sum_{p=1}^{N_p} \bar{S}_{ip} M_p v_p^{s-\frac{1}{2}}, \\ v_i^{s-\frac{1}{2}} &= \frac{(mv)_i^{s-\frac{1}{2}}}{m_i^s}. \end{aligned}$$

(2) Solve the equation of motion on the grid and update the grid information.

$$\sum_{i=1}^{n} m_i^s a_i^s = (f_i^{\text{int}})^s + (f_i^{\text{ext}})^s, i = 1, 2, \cdots, n,$$
$$v_i^{s+\frac{1}{2}} = v_i^{s-\frac{1}{2}} + a_i^s \Delta t,$$

where $(f_i^{\text{int}})^s$ and $(f_i^{\text{ext}})^s$ are given by equation (6.11) and (6.12) respectively.

(3) Update the information on the material points.

$$(x_1^p)^{s+1} = (x_1^p)^s + \sum_{i=1}^{n} N_i((x_1^p)^s) v_i^{s+\frac{1}{2}} \Delta t,$$
$$(x_2^p)^{s+1} = (x_2^p)^s + \sum_{i=1}^{n} N_i((x_2^p)^s) v_i^{s+\frac{1}{2}} \Delta t,$$
$$x_p^{s+1} = x_p^s + \sum_{i=1}^{n} \bar{S}_{ip} v_i^{s+\frac{1}{2}} \Delta t,$$
$$v_p^{s+\frac{1}{2}} = v_p^{s-\frac{1}{2}} + \sum_{i=1}^{n} \bar{S}_{ip} (v_i^{s+\frac{1}{2}} - v_i^{s-\frac{1}{2}}),$$
$$\nabla v_p^{s+\frac{1}{2}} = \sum_{i=1}^{n} \nabla \bar{S}_{ip} v_i^{s+\frac{1}{2}},$$
$$F_p^{s+1} = (1 + \nabla v_p^{s+\frac{1}{2}} \Delta t) F_p^s,$$
$$\sigma_p^{s+1} = \sigma(F_p^{s+1}),$$
$$\Omega_p^{s+1} = (x_2^p)^{s+1} - (x_1^p)^{s+1}.$$

(4) Generate a new grid.

In the first step of the above algorithm, one notices that the mapping from the material-point data to the background grid is essentially first order MLS; in the improved CPDI, we choose to use the same second order MLS mapping defined in the IMPM algorithm. The improved CPDI algorithm is summarized as follows:

(1) The mapping from material points to grid.

$$m_i^s = \sum_{p=1}^{N_p} \bar{S}_{ip} M_p, i = 1, 2, \cdots, n,$$
$$v_i^{s-\frac{1}{2}} = \sum_{p=1}^{npt} \Phi_p^s(x_i) v_p^{s-\frac{1}{2}}.$$

Here $\Phi_p(x)$ is defined the same as in equation (5.6).

(2) Solve the equation of motion on the grid and update the grid information.

$$\sum_{i=1}^{n} m_i^s a_i^s = (f_i^{\text{int}})^s + (f_i^{\text{ext}})^s, i = 1, 2, \cdots, n,$$
$$v_i^{s+\frac{1}{2}} = v_i^{s-\frac{1}{2}} + a_i^s \Delta t,$$

where $(f_i^{\text{int}})^s$ and $(f_i^{\text{ext}})^s$ are given by equations (6.11) and (6.12) respectively.

(3) Update the information on the material points.

$$(x_1^p)^{s+1} = (x_1^p)^s + \sum_{i=1}^{n} N_i((x_1^p)^s) v_i^{s+\frac{1}{2}} \Delta t,$$
$$(x_2^p)^{s+1} = (x_2^p)^s + \sum_{i=1}^{n} N_i((x_2^p)^s) v_i^{s+\frac{1}{2}} \Delta t,$$
$$x_p^{s+1} = x_p^s + \sum_{i=1}^{n} \bar{S}_{ip} v_i^{s+\frac{1}{2}} \Delta t,$$
$$v_p^{s+\frac{1}{2}} = v_p^{s-\frac{1}{2}} + \sum_{i=1}^{n} \bar{S}_{ip} (v_i^{s+\frac{1}{2}} - v_i^{s-\frac{1}{2}}),$$
$$\nabla v_p^{s+\frac{1}{2}} = \sum_{i=1}^{n} \nabla \bar{S}_{ip} v_i^{s+\frac{1}{2}}$$
$$F_p^{s+1} = (1 + \nabla v_p^{s+\frac{1}{2}} \Delta t) F_p^s,$$
$$\sigma_p^{s+1} = \sigma(F_p^{s+1}),$$
$$\Omega_p^{s+1} = (x_2^p)^{s+1} - (x_1^p)^{s+1}.$$

(4) Generate a new grid.

## 6.7 Convergence of CPDI and improved CPDI

With the same manufactured solutions and parameters as in Section 4.6, the convergence of CPDI, IMPM and improved CPDI is given in the following graphs. The axes are log base 10 of the $L_2$ norm of the error vs. the log base 10 of the mesh size. The slope of the line that approximates the data reveals the convergence rate. The $L_2$ error is defined as

$$E_{l2} = (\int_{\Omega} (u_n^{\text{num}}(x) - u_n^{\text{ex}}(x))^2 d\Omega)^{\frac{1}{2}}, \tag{6.35}$$

where $u_n^{\text{num}}$ is the numerical solution and $u_n^{\text{ex}}$ is the analytical solution at time $t^n$. The error is computed using one-point quadrature over the material-point domain for the stress, density, velocity and displacement, and over the element domain for the velocity. Since the analytical solution for the velocity is only known on the material points, second order MLS can be used to map the analytical velocity on the material points to the grid so that the analytical solution of velocity on the grid can be approximated and the error of velocity on the grid can be computed.

In Figures 6.1, 6.2, 6.3 and 6.4, the top line is the error in the stress on the material points, the second line is the error in the density on the material points, the third line is the error in the velocity on the material points and on the grid, and the fourth line is the error in the displacement on the material points.

As one can see in Figure 6.1, CPDI appears second order for all the field variables for course meshes. But if one refines the mesh size as in Figure 6.2, the convergence rate of CPDI drops, and when the mesh size is very small, it stops converging. In Figure 6.3, the black curves show errors computed using IMPM, and the curves in red show errors for CPDI. As one can see, IMPM has better convergence than CPDI in terms of finer meshes but it also eventually becomes unstable on fine meshes. In Figure 6.4, the red curves show errors obtained using original CPDI, and the curves in black show errors computed using improved CPDI using second order MLS for

the mapping. As one can see, the rate of convergence of CPDI is improved for finer meshes when the mapping algorithm is improved. When the mesh size is in the range of $\frac{1}{10^4}$, CPDI, improved CPDI and IMPM all have stability issues. Such instability is still under investigation.

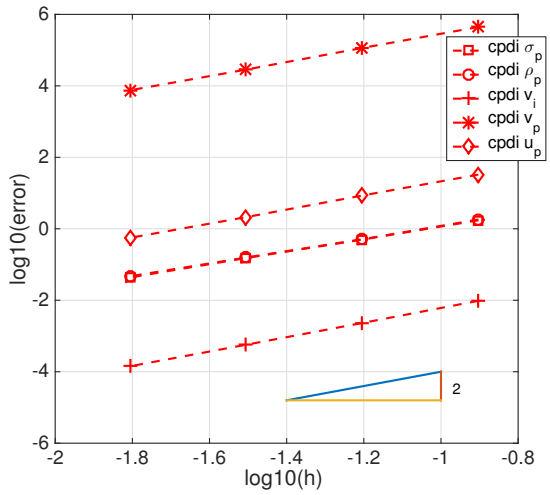


Figure 6.1: Convergence of CPDI for a large deformation problem with course mesh sizes in $1D$, G=0.1.

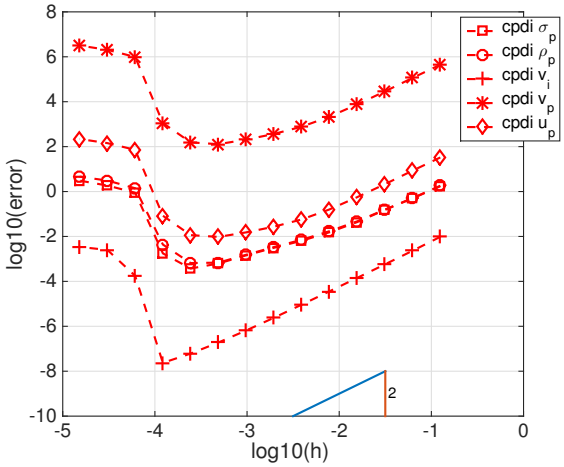


Figure 6.2: Convergence of CPDI for a large deformation problem with finer mesh sizes in $1D$, G=0.1.

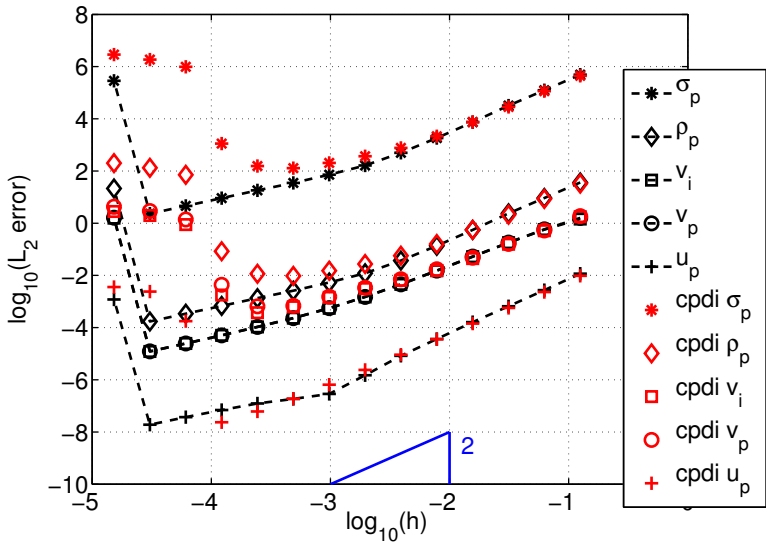


Figure 6.3: Convergence of IMPM and CPDI for a large deformation problem in $1D$, G=0.1.

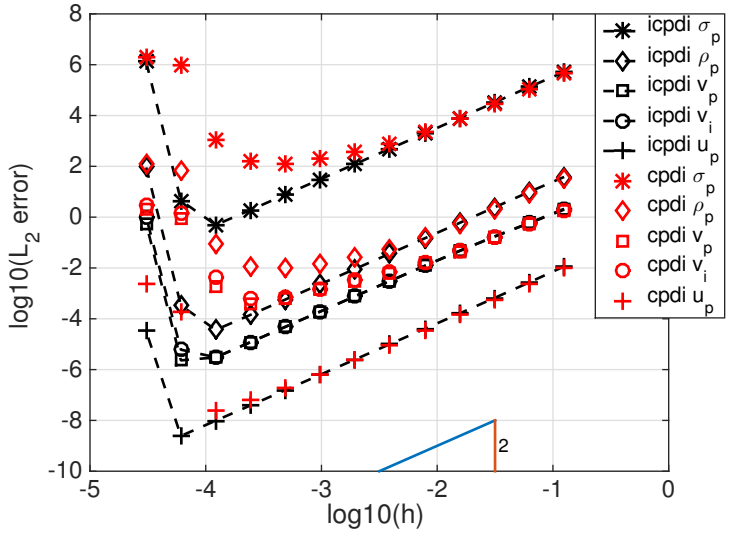


Figure 6.4: Convergence of CPDI and improved CPDI for a large deformation problem in $1D$, G=0.1.

# Chapter 7

# Stability Analysis of MPM

## 7.1 Introduction

There has been some confusion about what should be the correct $CFL$ condition in the MPM literature. We have also seen that all variations of MPM, including MPM, GIMP, CPDI and IMPM, appear to become unstable on fine meshes. Such instability is still under investigation. One possible reason for this instability is called the finite grid instability, which is a well known phenomenon in the particle-in-cell method [15, 27]. This instability is due to the fact that high frequency modes existing on the material points are mapped to the low frequency modes on the grid. The instability is analyzed by linearizing the equations of motion. In original MPM, it is also shown by a nonlinear analysis that energy is bounded. Thus, the nonlinear terms have the potential to control the instability. It has been demonstrated, [15, 27], that an implicit solver controls the instability. Another remedy that has been proposed is jiggling the grid position [7].

In this chapter, the linearization of the MPM equations will be presented and the corresponding linearized stability analysis will be studied. As usual, the solution

to the linearized equations is studied by Fourier transform. The analysis in this thesis differs from the reference [15, 27] by providing a more careful derivation of the transformed equations which leads to a slightly different dispersion relation.

## 7.2 Equations to Solve

Consider the following system of equations for MPM to solve in $1D$,

$$\rho \frac{dv}{dt} = \frac{\partial \sigma}{\partial x}, \tag{7.1}$$

$$\frac{dF}{dt} = F\frac{\partial v}{\partial x}, \tag{7.2}$$

$$\rho = \frac{\rho_0}{J}, \tag{7.3}$$

$$\sigma = E(F-1), \tag{7.4}$$

where $\sigma$ is the Cauchy stress, $\rho$ is the density in current configuration, $\rho_0$ is the density in the initial configuration, $E$ is Young's modulus which is a constant, $F$ is the deformation gradient and $J$ is the Jacobian of $F$ (i.e $J = |F|$). Other constitutive models rather than equation (7.4) can be treated similarly. Since the equations are written in $1D$, we have $J = F$. By substituting equations (7.3) and (7.4) into equation (7.1), we get

$$\frac{\rho_0}{|F|}\frac{dv}{dt} = \frac{\partial (E(F-1))}{\partial x},$$

$$\frac{dF}{dt} = F\frac{\partial v}{\partial x}.$$

Further simplify to get

$$\frac{dv}{dt} = C^2|F|\frac{\partial F}{\partial x}, \tag{7.5}$$

$$\frac{dF}{dt} = F\frac{\partial v}{\partial x}, \tag{7.6}$$

where $C^2 = \frac{E}{\rho_0}$. We need to linearize the above system of equations written in Eulerian coordinates in order to study the dispersion relation.

## 7.3 Linearized Equations

Let us assume

$$\overline{v} = v_0 + v,$$
$$\overline{F} = F_0 + F.$$

In the above, $v_0$ and $F_0$ are constant terms, and $v$ and $F$ are perturbations. Substituting these expressions into equation (7.5) and (7.6) for $v$ and $F$. We derive the following by separating the constant and perturbed terms:

$$\frac{dv_0}{dt} = C^2 F_0 \frac{\partial F_0}{\partial x}, \tag{7.7}$$
$$\frac{dF_0}{dt} = F_0 \frac{\partial v_0}{\partial x}, \tag{7.8}$$

$$\frac{\partial v}{\partial t} + v_0 \frac{\partial v}{\partial x} = C^2 F_0 \frac{\partial F}{\partial x}, \tag{7.9}$$
$$\frac{\partial F}{\partial t} + v_0 \frac{\partial F}{\partial x} = F_0 \frac{\partial v}{\partial x}. \tag{7.10}$$

## 7.4 Dispersion Relation of MPM

Consider the case of equally spaced material points, labeled as $x_{p-\frac{1}{2}}$ with spacing $h' = \frac{h}{N_p^e}$, where $N_p^e$ is the number of material points per element. Thus the location of $x_{p-\frac{1}{2}}$ is $(p-\frac{1}{2})h' = (p-\frac{1}{2})\frac{h}{N_p^e}$.

A local numbering of material points is denoted by $x_{p^e,I}$, $p = 1, 2, \cdots, N_p^e$ within element $I$ which is the interval $\Omega^I = [x_{I-1}, x_I]$ of size $h = x_I - x_{I-1}$. The position $x_{p^e,I}$ is

$$x_{p^e,I} = x_{I-1} + \frac{2p^e - 1}{2N_p^e} h, \quad p^e = 1, 2, \cdots, N_p^e, \tag{7.11}$$

or

$$\begin{aligned} x_{p^e,I} &= (I-1+\frac{2p^e-1}{2N_p^e})h, \\ &= ((I-1)N_p^e+p^e-\frac{1}{2})h'. \end{aligned} \tag{7.12}$$

The relation between the local and global numbering is, given $p^e$ and $I$, then $p-\frac{1}{2} = (I-1)N_p^e+p^e-\frac{1}{2}$ or $p=(I-1)N_p^e+p^e$. Now, given the global number $p-\frac{1}{2}$, we can find the element number and the local material-point number as follows: divide $p-\frac{1}{2}$ by $N_p^e$ and take the integer part, rounding to $-\infty$ and set $I-1=\text{floor}(\frac{p-\frac{1}{2}}{N_p^e})$, where floor is the floor function in Matlab, then $p^e=p-(I-1)N_p^e$.

A discretization of (7.9) similar to the MPM discretization is

$$\begin{aligned} \frac{v_I^{s+\frac{1}{2}}-v_I^{s-\frac{1}{2}}}{\Delta t} &= -C^2F_0\frac{1}{N_p^e}\sum_p \frac{\partial N_I}{\partial x}|_{x=x_{p-\frac{1}{2}}}F^s_{p-\frac{1}{2}}, \\ &= -C^2F_0\frac{1}{N_p^e}(\sum_{p^e=1}^{N_p^e}F^n_{p^e,I}\frac{\partial N_I}{\partial x}|_{x=x_{p^e,I}}-\sum_{p^e=1}^{N_p^e}F^s_{p^e,I+1}\frac{\partial N_I}{\partial x}|_{x=x_{p^e,I+1}}), \\ &= C^2F_0\frac{1}{N_p^e}(-\frac{1}{h}\sum_{p^e=1}^{N_p^e}F^s_{p^e,I}+\frac{1}{h}\sum_{p^e=1}^{N_p^e}F^s_{p^e,I+1}), \\ &= C^2F_0\frac{1}{hN_p^e}\sum_{p^e=1}^{N_p^e}(F^s_{p^e,I+1}-F^s_{N_p^e-p^e+1,I}). \end{aligned} \tag{7.13}$$

A discretization of (7.10) similar to the MPM discretization is

$$\frac{F^{s+1}_{p-\frac{1}{2}}-F^s_{p-\frac{1}{2}}}{\Delta t} = F_0(\frac{v_I^{s+\frac{1}{2}}-v_{I-1}^{s+\frac{1}{2}}}{h}). \tag{7.14}$$

The above equation holds for any material point $p^e$ in element $I$. Since $F^s_{p^e,I}$ has global index $p=(I-1)N_p^e+p^e-\frac{1}{2}$, $F^s_{p^e,I+1}$ has global index $p=IN_p^e+p^e-\frac{1}{2}$ and $F^s_{N_p^e-p^e+1,I}$ has global index $p=(I-1)N_p^e+(N_p^e-p^e+1)-\frac{1}{2}$. Then equations (7.13)

and (7.14) become

$$\frac{v_I^{s+\frac{1}{2}} - v_I^{s-\frac{1}{2}}}{\Delta t} = C^2 F_0 \frac{1}{hN_p^e} \sum_{p^e=1}^{N_p^e} (F^s_{IN_p^e+p^e-\frac{1}{2}} - F^s_{(I-1)N_p^e+(N_p^e-p^e+1)-\frac{1}{2}}), \tag{7.15}$$

$$\frac{F^{s+1}_{(I-1)N_p^e+p^e-\frac{1}{2}} - F^s_{(I-1)N_p^e+p^e-\frac{1}{2}}}{\Delta t} = F_0(\frac{v_I^{s+\frac{1}{2}} - v_{I-1}^{s+\frac{1}{2}}}{h}). \tag{7.16}$$

To study the dispersion relation, Let us assume that the solution has the following form, where $\omega$ is the frequency and $k$ is the wave number of a Fourier mode,

$$F^s_{p-\frac{1}{2}} = \hat{F} e^{ikx_{p-\frac{1}{2}}} e^{-i\omega t^s}, \tag{7.17}$$

$$v_I^{s+\frac{1}{2}} = \hat{v} e^{ikx_I} e^{-i\omega t^{s+\frac{1}{2}}}. \tag{7.18}$$

Put the above forms of solution into equation (7.15) to get

$$\frac{\hat{v}e^{ikx_I}e^{-i\omega t^{s+\frac{1}{2}}} - \hat{v}e^{ikx_I}e^{-i\omega t^{s-\frac{1}{2}}}}{\Delta t} = C^2 F_0 \frac{1}{hN_p^e} \sum_{p^e=1}^{N_p^e} (\hat{F} e^{ikx_{IN_p^e+p^e-\frac{1}{2}}} e^{i\omega t^s} - \hat{F} e^{ikx_{(I-1)N_p^e+(N_p^e-p^e+1)-\frac{1}{2}}} e^{i\omega t^s}).$$

Simplify to get

$$\frac{\hat{v}(e^{-i\omega\frac{\Delta t}{2}} - e^{i\omega\frac{\Delta t}{2}})}{\Delta t} = C^2 F_0 \hat{F} \frac{1}{hN_p^e} \sum_{p^e=1}^{N_p^e} (e^{ik(x_{IN_p^e+p^e-\frac{1}{2}} - x_I)} - e^{ik(x_{(I-1)N_p^e+(N_p^e-p^e+1)-\frac{1}{2}} - x_I)}). \tag{7.19}$$

Since

$$\begin{aligned} x_{IN_p^e+p^e-\frac{1}{2}} - x_I &= -(x_{(I-1)N_p^e+(N_p^e-p^e+1)-\frac{1}{2}} - x_I), \\ &= -\frac{h'}{2} + p^e h' = \frac{2p^e-1}{2} h', \end{aligned} \tag{7.20}$$

further simplify (7.19) as

$$-\frac{2i\hat{v}\sin\omega\frac{\Delta t}{2}}{\Delta t} = C^2F_0\hat{F}\frac{1}{hN_p^e}\sum_{p^e=1}^{N_p^e}(e^{ikh'\frac{2p^e-1}{2}} - e^{-ikh'\frac{2p^e-1}{2}}),$$

$$= C^2F_0\hat{F}\frac{1}{hN_p^e}\sum_{p^e=1}^{N_p^e}2i\sin(kh'\frac{2p^e-1}{2}), \tag{7.21}$$

$$= C^2F_0\hat{F}\frac{1}{hN_p^e}2i\frac{\sin^2(\frac{kh}{2})}{\sin(\frac{kh'}{2})}. \tag{7.22}$$

The last equality is derived from the following identity:

$$\sum_{p^e=1}^{N_p^e}\sin(kh'\frac{2p^e-1}{2}) = \sum_{p^e=1}^{N_p^e}\mathrm{Im}(e^{ikh'\frac{2p^e-1}{2}}),$$

$$= \mathrm{Im}(\sum_{p^e=1}^{N_p^e}e^{ikh'\frac{2p^e-1}{2}}),$$

$$= \mathrm{Im}(e^{\frac{k}{2}h'i}\frac{1-e^{N_p^ekh'i}}{1-e^{kh'i}}),$$

$$= \frac{\sin^2(\frac{kh}{2})}{\sin(\frac{kh'}{2})}. \tag{7.23}$$

Put equations (7.17) and (7.18) into equation (7.16) to get

$$\frac{\hat{F}(e^{ikx_{(I-1)N_p^e+p^e-\frac{1}{2}}e^{-i\omega t^{s+1}}} - e^{ikx_{(I-1)N_p^e+p^e-\frac{1}{2}}e^{-i\omega t^s}})}{\Delta t} = F_0\frac{1}{h}\hat{v}(e^{ikx_I}e^{-i\omega t^{s+\frac{1}{2}}} - e^{ikx_{I-1}}e^{-i\omega t^{s+\frac{1}{2}}}).$$

Simplify further to get

$$\frac{\hat{F}(e^{-i\omega\frac{\Delta t}{2}} - e^{i\omega\frac{\Delta t}{2}})}{\Delta t} = F_0\frac{1}{h}\hat{v}(e^{ik(x_I-x_{(I-1)N_p^e+p^e-\frac{1}{2}})} - e^{ik(x_{I-1}-x_{(I-1)N_p^e+p^e-\frac{1}{2}})}).$$

Since $x_I - x_{(I-1)N_p^e+p^e-\frac{1}{2}} = h - h'\frac{2p^e-1}{2}$ and $x_{I-1} - x_{(I-1)N_p^e+p^e-\frac{1}{2}} = -h'\frac{2p^e-1}{2}$, we get

$$-\frac{2i\hat{F}\sin(\omega\frac{\Delta t}{2})}{\Delta t} = F_0\frac{1}{h}\hat{v}(e^{ik(h-h'\frac{2p^e-1}{2})} - e^{-ikh'\frac{2p^e-1}{2}}).$$

Sum over $p^e$ on both sides of the above equation to get

$$-\sum_{p^e=1}^{N_p^e}\frac{2i\hat{F}\sin(\omega\frac{\Delta t}{2})}{\Delta t} = F_0\frac{1}{h}\hat{v}\sum_{p^e=1}^{N_p^e}(e^{ik(h-h'\frac{2p^e-1}{2})} - e^{-ikh'\frac{2p^e-1}{2}}). \tag{7.24}$$

Since we have the following identity for equally spaced material points:

$$\sum_{p^e=1}^{N_p^e} e^{ik(h-h'\frac{2p^e-1}{2})} = \sum_{p^e=1}^{N_p^e} e^{ikh'\frac{2p^e-1}{2}}, \tag{7.25}$$

equation (7.24) simplifies further to

$$\begin{aligned} -N_p^e \frac{2i\hat{F}\sin(\omega\frac{\Delta t}{2})}{\Delta t} &= F_0\hat{v}\frac{1}{h}\sum_{p^e=1}^{N_p^e}(e^{ikh'(\frac{2p^e-1}{2})} - e^{-ikh'(\frac{2p^e-1}{2})}), \\ &= F_0\hat{v}\frac{1}{h}\sum_{p^e=1}^{N_p^e}(2i\sin(kh'\frac{2p^e-1}{2})), \\ &= F_0\hat{v}\frac{1}{h}2i\frac{\sin^2(\frac{kh}{2})}{\sin(\frac{kh'}{2})}. \end{aligned} \tag{7.26}$$

Again, the last equality is derived from identity (7.23). To summarize, the following dispersion relations are derived

$$-\frac{2i\hat{v}\sin(\omega\frac{\Delta t}{2})}{\Delta t} = C^2F_0\hat{F}\frac{1}{hN_p^e}\frac{2i\sin^2(\frac{kh}{2})}{\sin(\frac{kh'}{2})}, \tag{7.27}$$

$$-\frac{2i\hat{F}\sin(\omega\frac{\Delta t}{2})}{\Delta t} = F_0\hat{v}\frac{1}{h}\frac{2i\sin^2(\frac{kh}{2})}{N_p^e\sin(\frac{kh'}{2})}. \tag{7.28}$$

Further, the relation between the frequency $\omega$ and the wave number $k$ is as follows:

$$\omega = \frac{2}{\Delta t}\sin^{-1}(\pm\frac{\Delta t}{h}CF_0\frac{1}{N_p^e}\frac{\sin^2(\frac{kh}{2})}{\sin(\frac{kh'}{2})}). \tag{7.29}$$

Now, let $\theta = kh$, $CFL = CF_0\frac{\Delta t}{h}$, and let $M = \frac{\Delta t}{h}CF_0\frac{1}{N_p^e}\frac{\sin^2(\frac{kh}{2})}{\sin(\frac{kh'}{2})}$; then equation (7.29) becomes

$$\omega = \frac{2}{\Delta t}\sin^{-1}(\pm M). \tag{7.30}$$

If $|M| \leq 1$, we have for the amplification factor $|g| = e^{\mathrm{Im}\,\omega\Delta t}$ that $|g| \leq 1$. Thus, if $|M| \leq 1$, solutions to the linearized system do not grow and the method is stable.

If $|M| > 1$, $|g| > 1$; and the method is unstable. Since $M = \pm CFL\frac{1}{N_p^e}\frac{\sin(\frac{\theta}{2})}{\sin(\frac{\theta}{2N_p^e})}$, in order to have stability (i.e. $|M| \leq 1$), we need to have

$$CFL \leq \min_{0\leq\theta\leq 2\pi}(N_p^e\frac{\sin(\frac{\theta}{2N_p^e})}{\sin(\frac{\theta}{2})}). \tag{7.31}$$

Since $f(\theta) = \frac{\sin(\frac{\theta}{2N_p^e})}{\sin(\frac{\theta}{2})}$ is an increasing function of $\theta$ for a given $N_p^e$, we can derive $\min_{0\leq\theta\leq 2\pi} N_p^e f(\theta) = 1$. A sufficient condition for stability becomes

$$CFL \leq \ 1. \tag{7.32}$$

Consider the dispersion relation in equation (7.29), Let $\theta = kh$, since $h^{'} = \frac{h}{N_p^e}$, we have $kh^{'} = \frac{\theta}{N_p^e}$. For $\theta$ small, we have $\sin(\frac{\theta}{2}) = \frac{\theta}{2} +$ higher order terms, then we can approximate $\omega$

$$\omega = \frac{1}{\Delta t}(\frac{\Delta t}{h}CF_0\theta + O(\theta^3)). \tag{7.33}$$

Since $\theta = kh$, simplify further to get

$$\omega = CF_0k + O(\theta^2). \tag{7.34}$$

Since the analytic dispersion relation for equations (7.9) and (7.10) is $\omega = CF_0k$, we can see that the numerical dispersion relation in equation (7.34) has the leading order term $O(\theta^2)$.

In summary, the linearized stability analysis derived in this chapter has limitations. First of all, it does not take into account the nonlinear advection term. Second of all, it does not predict the dispersive and dissipative properties of the MPM algorithm accurately. Third of all, although it gives a rough bound on the sufficient condition for the $CFL$ number, it fails to give a tight bound.

# Chapter 8

# Conclusions

## 8.1 Summary

A general framework for improving MPM is provided by combining ideas of function reconstruction from scattered data and the finite element method. Within this framework, the IMPM and improved CPDI are produced and they become the most accurate algorithms among the variations of MPM including MPM, GIMP, CPDI, IMPM and improved CPDI. IMPM not only retains good features of the original MPM, but it increases the order of accuracy of MPM from zeroth order to second order for large deformation problems both in space and time. In the improved CPDI, it also retains the good features of the original CPDI and it increases the order of accuracy of CPDI from first order to second order for fine meshes. In addition, a stability analysis on the linearized equations of motion is derived. Although it gives a rough approximation on the sufficient condition for the upper bound of the $CFL$ number, such stability analysis has limitations. It does not predict the dispersive and decaying behavior of the numerical solutions of MPM very well.

## 8.2 Future Work

**Boundary Conditions** Even though IMPM provides a general framework for improving MPM, it has limitations in terms of how to enforce boundary conditions in complicated geometry. One interesting question is how to enforce boundary conditions in the situation where the boundary of the body does not coincide with the background grid. In MPM, IMPM, GIMP, CPDI and improved CPDI, we enforce boundary conditions on the closest exterior grid points to the boundary of the body instead of enforcing boundary conditions on the exact boundary. As a result, errors are introduced.

One of my future research directions is to make further improvements in handling boundary conditions in complicated geometry. A possible approach is to use ideas from the immersed boundary method [21], the immersed interface method [18] or the fictitious domain method [10]. For example, in the immersed boundary method, the body is immersed in the background grid, where the boundary of the body also does not coincide with the background grid. In such a situation, there are two general approaches to enforce the boundary condition in a more accurate way. One approach is to use ghost cells, where one has to do interpolation to get information on the ghost cell or closest exterior grid points from information on the boundary and interior points of the body. The other approach relies on Lagrangian multipliers, where one has to solve a larger system of equations. The most promising approach to further improve IMPM is to use the ghost cell method. In such an approach, we can firstly find the image point of the ghost grid point along the normal to the boundary. By doing simple linear interpolation, or higher order interpolation as one desires, from the nearby interior grid points, we can obtain values (i.e velocity) on the image point. Then we can obtain the values of the ghost grid point by linear interpolation from the values of image point and the boundary point along the normal to the boundary.

**Arbitrary Order** The second direction of my future research is to improve IMPM to arbitrary order. In order to accomplish this, one has to combine ideas from the meshfree particle methods and the Finite Element Methods. In original MPM, the function reconstruction is first order MLS, while in IMPM, the function reconstruction is second order MLS. One can utilize the ideas from the meshfree particle methods to construct an approximate function from scattered data to arbitrary order, so that one can further improve IMPM in the function reconstruction. In the second step of MPM, the equations of motion are solved on the background grid. In MPM and IMPM, the linear nodal basis function is used, which limits the algorithm to second order in this step. For further improvements, one can use higher or arbitrary order nodal basis functions to solve the equations of motion. Furthermore, one can use arbitrary MLS to map data from material points to the quadrature points to improve the quadrature to arbitrary order. Of course, one possible difficulty in generalizing IMPM to arbitrary order is the issue of stability, which is still under investigation and further research.

**Stability Control** The third direction of my future research is to control the stabilities of IMPM for fine meshes. One possible approach is to add artificial numerical dissipation to the IMPM algorithm. The other approach is to use an implicit solver instead of an explicit solver. It has been demonstrated, [15, 27], that an implicit solver controls the instability.

# References


[1] Sadeghirad. A., Brannon. R., and Burghardt. J. Convected particle domain interpolation technique to extend applicability of the material point method for problems involving massive deformations. *International Journal for Numerical Methods in Engineering*, 86:1435–1456, 2011.

[2] Sadeghirad. A., Brannon. R., and Guilkey. J. Second-order convected particle domain interpolation with enrichment for weak discontinuities at material interfaces. *International Journal for Numerical Methods in Engineering*, 95:928–925, 2013.

[3] Ravindra Ambati, Xiaofei Pan, Huang Yuan, and Xiong Zhang. Application of material point methods for cutting process simulations. *Computational Materials Science*, 57:102–110, 2012.

[4] T. Beltschko, Y.Y. Lu, and L. Gu. Element-free galerkin methods. *International Journal of Numerical Methods in Engineering*, 37:229–256, 1994.

[5] T. Belytschko, Y. Krongauz, D. Organ, M. Fleming, and P. Krysl. Meshless methods: An overview and recent developments. *Computer Methods in Applied Mechanics and Engineering*, 139:3–47, 1996.

[6] Ted Belytschko, Yong Guo, Wing Kam Liu, and Shao Ping Xiao. A unified stability analysis of meshless particle methods. *Int.Journl.Numer.Methods.Eng*, 48:1359–1400, 2000.

[7] J.W. Brackbill and Giovanni Lapenta. A method to suppress the finite-grid instability in plasma simulations. *Journal of Computational Physics*, 114:77–84, 1994.

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

[9] M.S. Gadala and J. Wang. ALE formulation and its application in solid mechanics. *Computer Methods in Applied Mechanics and Engineering*, 167:35–55, 1998.

[10] R. Glowinski, T-W. Panand, T.I. Hesla, and D.D. Joseph. A distributed Lagrange multi-plier/fictitious domain method for particulate flows. *Int J Multiphase Flow*, 25:755–794, 1999.

[11] Francis. H. Harlow. The particle-in-cell computing method for fluid dynamics. *Methods Comput. Phys*, 3:319–343, 1964.

[12] C.W. Hirt, A.A. Amsden, and J.L. Cook. An arbitrary Lagrangian-Eulerian computing method for all flow speeds. *Journal of Computational Physics*, 135:203–216, 1997.

[13] Bonet. J. and Kulasegaram. S. Correction and stabilization of smooth particle hydrodynamics methods with applications in metal forming simulations. *Int. J. Numer. Methods Eng.*, 47:1189–1214, 2000.

[14] Melenk. J.M. and Babuska. I. Partition of unity finite element method. *Computer Methods in Applied Mechanics and Engineering*, 139:314–389, 1996.

[15] Brackbill. J.U. The ringing instability of particle-in-cell calculations of low-speed flow. *Journal of Computational Physics*, 75:469–492, 1988.

[16] P. Lancaster and K. Salkauskas. Surfaces generated by moving least squares methods. *Mathematics of Computation*, 37:141–158, 1981.

[17] Lucy. L.B. A numerical approach to the testing of the fission hypothesis. *Astrophysical journal*, 82:1013–1024, 1997.

[18] R.J. Leveque and Z. Li. Immersed interface methods for Stokes flow with elastic boundaries or surface tension. *SIAM J Sci Comput*, 18:709–735, 1997.

[19] W. Liu, S. Jun, S. Li, J. Adee, and T. Beltschko. Reproducing kernel particle methods for structural dynamics. *International Journal of Numerical Methods in Engineering*, 38:1655–1680, 1995.

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

[21] Haoxiang Luo, Rajat Mittal, Xudong Zheng, Steven A. Bielamowicz, Raymond. J. Walsh, and James K Hahn. An immersed-boundary method for flow-structure interaction in biological systems with application to phonation. *Journal of Computational Physics*, 227:9303–9332, 2009.

[22] S. Ma, X. Zhang, and X.M. Qiu. Comparison study of MPM and SPH in modeling hypervelocity impact problems. *International Journal of Impact Engineering*, 36:272–282, 2009.

[23] F. Martinet and P. Chabrand. Application of ALE finite elements method to a lubricated friction model in sheet metal forming. *International Journal of Solids and Structures*, 37:4005–4031, 2000.

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

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

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

[27] Cummins. S.J. and Brackbill. J.U. An implicit particle-in-cell method for granular materials. *Journal of Computational Physics*, 180:506–548, 2002.

[28] Alexey Stomakhin, Craig Schroeder, Lawrence Chai, Joseph Teran, and Andrew Selle. A material point method for snow simulation. *ACM Transactions on Graphics*, 32:102–114, 2013.

[29] D. Sulsky, Z. Chen, and H.L. Schreyer. A particle method for history-dependent materials. *Computer Methods in Applied Mechanics and Engineering*, 5:179–196, 1994.

[30] D. Sulsky and H.L. Schreyer. Axisymmetric form of the material point method with applications to upsetting and Taylor impact problems. *Computer Methods in Applied Mechanics and Engineering*, 139:409–429, 1996.

[31] D. Sulsky, H.L. Schreyer, K. Peterson, R. Kwok, and M. Coon. Using the material-point method to model sea ice dynamics. *Journal of Geophysical Research*, 112:C02S90, 2007.

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

[33] Ling Xu, Howard Schreyer, and Deborah Sulsky. Blast-induced rock fracture near a tunnel. *International Journal for Numerical and Analytical Methods in Geomechanics*, 39:23–50, 2014.