Study of Tsunami-Induced Fluid and Debris Load on Bridges using the Material Point Method

Wen-Chia Yang

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2016

Reading Committee:

Pedro Arduino, Chair

Peter Mackenzie-Helnwein, Chair

Gregory R. Miller

Program Authorized to Offer Degree:
Civil and Environmental Engineering

University of Washington

Abstract

Study of Tsunami-Induced Fluid and Debris Load on Bridges using the Material Point Method

Wen-Chia Yang

Co-Chairs of the Supervisory Committee:
Professor Pedro Arduino
Civil and Environmental Engineering

Research Associate Professor Peter Mackenzie-Helnwein
Civil and Environmental Engineering

This research focuses on the development of techniques for the material point method (MPM) modeling the fluid-solid interaction, and gives preliminary results for researchers and engineers trying to understand the demands on bridge superstructures by tsunami-driven debris. First, a numerical flux smoothing algorithm is proposed to stabilize (nearly) incompressible fluid simulations with a control mechanism (i.e. numerical flux). Secondly, an enhanced boundary modeling technique is presented to decouple the geometries of grids and boundaries, by introducing boundary specific force fields for applying boundary conditions. Both of the newly developed algorithms are validated with numerical examples and test cases, ranging from fundamental hydrostatic problems to tsunami flows interacting with bridges. In the end, with the new algorithms, debris-induced loads, impacts and loads caused by damming effect, on bridges during a tsunami are studied. Results show (1) tsunami-driven debris can significantly increase demands on bridges compared with no-debris cases; (2) when flow is partially blocked by debris, peak horizontal forces are proportional in the order of about 3.6 to net cross-section area of the water flow at initial conditions; (3) impact forces are related to contact areas and propagation path of stress waves; and (4) contribution of water flow to the debris-induced impact forces can vary from 25 to 35 percent of those in the corresponding in-air cases.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

Throughout history, strong earthquakes have struck countries all over the world and caused major damage. Most of this damage has been due to ground shaking, but earthquakes have also triggered tsunamis, introducing even more damage in the coastal areas. As coastal population and the associated infrastructure continue to increase around the world, understanding and managing tsunami effects on infrastructure becomes increasingly important.

The objective of this research is to provide a starting point using the material point method (Sulsky, Zhou, & Schreyer, 1995) to study the fluid-solid interaction, and give preliminary results for researchers and engineers trying to understand the demands on bridge superstructures by tsunami-driven debris. Bridges represent a key part of infrastructure, playing a critical role in emergency response and post-event reconstruction, providing transportation support within the disaster areas and with the outside world. Therefore, it is important for bridges to be able to survive both ground shaking and the effects of tsunamis. This work focuses on the development of techniques for numerically modeling bridge loading due to tsunamis. Many prior numerical and experimental studies considering tsunami induced loads on bridges have been done, especially after the Great East Japan Earthquake (Nakao, Zhang, Sumimura, & Hoshikuma, 2013; Nakao, Itonaga, Nozaka, Izuno, & Kobayashi, 2013; Bricker & Nakayama, 2014; Azadbakht & Yim, 2015). However, few of these earlier studies have examined the influence of debris carried by the tsunami. The studies that have been completed have demonstrated debris can cause strong impact forces on columns and walls (Hiraishi, Haruo, & Saitoh, 2010; Ko, Cox, Riggs, & Naito, 2014;

Piran Aghl, Naito, & Riggs, 2014). Lessons from these studies should be straightforward to apply to bridges' vertical structures (i.e. bridge piers), but need some adjustment when applied to bridge superstructures since in this case flows carrying debris have different characteristics compared with flows considered in the previous studies. In addition, debris can also affect the fluid motion around the bridge and introduce additional forces with longer duration compared to impact. This phenomenon includes damming effects. These classes of problems are not well studied yet in the literature and involve complex contact interactions between solids and fluids. These effects are not easily accommodated with typical fluid-oriented or solid-oriented numerical frameworks. In this research the material point method (MPM) is used to model these complex fluid/solid (moving and stationary) interactions.

Originating from a particle-in-cell (fluid-oriented) method, FLIP (Brackbill, Brackbill, & Repel, 1986), MPM has been applied to a number of applications involving large deformations and complex interaction among bodies. However, because it has been optimized for solid mechanics, MPM is less commonly used to model fluid. Given the potential for modeling fluid-solid interactions and hence handling tsunami-driven-debris-induced loads on bridge superstructures, this research tries to enhance the capability of MPM in fluid modeling. The major issue of this application are integration errors, which arise from the particle form (an approximation of volume integrals) used in MPM and that can cause destabilization in the problem. Consequently, the node-based algorithm (Mast, Mackenzie-Helnwein, Arduino, Miller, & Shin, 2012) is the only solution in the MPM literature that is found (in this study) to be able to stabilize these analyses by smoothing the stress states and dissipating strain energy. However, the algorithm does not consider, and is unable to control, the nonphysical energy dissipation and hence may cause evident spurious behaviors by over-smoothing the stress states. Based on these findings, a numerical flux smoothing algorithm is proposed to stabilize the effects arising from integration errors, using a controlled strain energy dissipation mechanism. In addition, to increase flexibility in

grid geometry generation, an enhanced boundary modeling technique is presented. To demonstrate the effectiveness of the algorithms and modeling strategies developed in this dissertation, a number of numerical examples and test cases are presented ranging from fundamental hydrostatic problems to tsunami flows interacting with bridges including the effects of floating debris. However, limited by the performance of the single-threaded MPM program, mesh and particle refinements used in the validation examples and the debris-induced load study are relatively coarse compared to general applications found in the literature. A program allowing for parallel computing is necessary for further study in the future.

In this dissertation, the MPM and some of its branches are described in Chapter 2. The numerical flux smoothing algorithm proposed to improve the capability of MPM for modeling fluids is introduced in Chapter 3. A new boundary modeling method for the MPM is then described in Chapter 4. The performance in modeling water using MPM and the proposed algorithms are examined in Chapter 5, followed by a study of debris-induced impacts and damming effects discussed in Chapter 6. Conclusions of the current work and possible future extensions and improvements are summarized in Chapter 7.

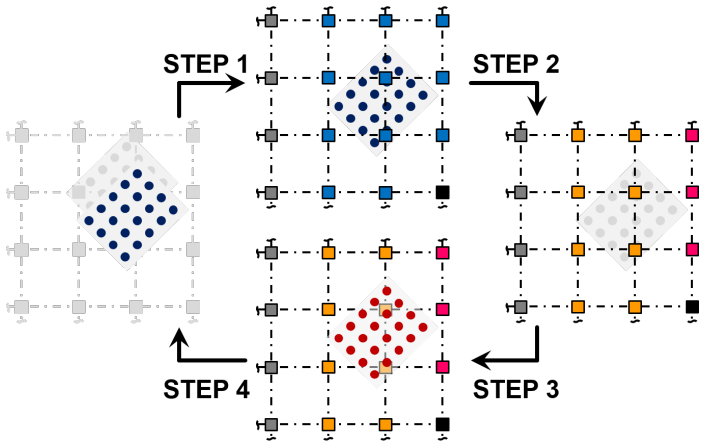
Chapter 2

VARIANTS OF THE MATERIAL POINT METHOD

The Material Point Method (MPM) is an extension of particle-in-cell methods to solid mechanics applications and was introduced by Sulsky et al. (1995). The two major components of the method, *material points* and *computational grid*, not only allow material states to be tracked and recorded for an arbitrary motion, and non-linear stress-strain relations to be accommodated in a direct manner, but also allow handling of large displacement and/or large deformation problems, contact problems and interaction between multiple bodies. Because of these features, MPM has been used in a number of applications during the past decades. Several extensions of MPM have been developed to mitigate some inherent problems including cell-crossing noise, an error of volume integration that will be discussed in Section 2.2. Such extensions include the Generalized Integration Material Point Method (GIMP) (Bardenhagen & Kober, 2004), the Convected Partial Domain Integration method (CPDI) (Sadeghirad, Brannon, & Burghardt, 2011) and the Dual Domain Material Point method (DDMP) (Zhang, Ma, & Giguere, 2011). Each of these extensions share the same philosophy and skeleton with the original method but use different implementation details. In this chapter, an overview of the methods will be given.

2.1 Material Point Method Families

All the variants of the Material Point Method (MPM), or material point methods, follow an updated Lagrangian numerical approach, which solves problems in a Lagrangian point of view, and updates, or pushes forward, the configuration at time t_n to its current (deformed) state at time t_{n+1}. The configuration at the end of each time



STEP 1 mapping material point data to grid nodes; **STEP 2** computing nodal variable evolution; **STEP 3** updating material point state variables via mapping from nodes; **STEP 4** updating material point locations based on current velocity and acceleration.

Figure 2.1: Computational cycle of material point method families at each time step.

step is then reset as a reference for the next step. Similar to finite element methods, finite continuum material domains in material point methods are discretized with and mapped to Lagrangian meshes (grids), which are used to evaluate time evolution of field variables. However, instead of recording the time-dependent values at nodes, the methods encapsulate deformation, velocity and all other variables at a finite number of *material points* (particles) as part of the continuum domain. Since no data are recorded on the grid, meshes in material point methods are often called (background) *computational grids* in the literature. These grids are predefined and are considered unchanged (in most of implementations) at the beginning of each time step, but are deformable during each time step. In other words, material point methods remesh the particle-represented continuum at each time step with the same predefined grid pattern. Although it does not mesh the particle domain perfectly during most of the simulation interval and hence generally causes error at edge cells and nodes, this predefined grid pattern allows avoiding cell distortion in large deformation problems,

saves the cost of remeshing, and simplifies algorithms related to contact problems. As the price for using a predefined grid pattern, mapping back and forth between variable fields of the particle-represented and grid-represented continuum domains (shortened as "the particle domains" and "the grid domains" in this dissertation, respectively) is necessary during each computational cycle.

A complete computational cycle consists of four major steps (Figure 2.1),

1. computing grid values via mapping from material points
2. computing field variable evolutions on the grid
3. updating particle state variables via mapping an interpolating from grid nodes
4. updating material point locations

Since mapping between the two domains is necessary for each cycle, only variable increments are updated from grids to particles to reduce diffusion of variable fields. Acceleration $\boldsymbol{a}^G$ and velocity $\boldsymbol{v}^G$ on the grid are used to update velocity $\boldsymbol{v}^P$ and motion-related variables, e.g. displacement $\boldsymbol{u}^P$ and strain $\boldsymbol{\varepsilon}^P$, of particles, respectively, in which the superscripts P and G indicate variable fields on particle and grid domains.

Like FEM, all variants of MPM interpolate variables on grids as

$$(\bullet)^G(\boldsymbol{x},t) = \sum_i N_i^G(\boldsymbol{x})\,(\bullet)_i^G(t) \tag{2.1}$$

in which $N_i^G(\boldsymbol{x})$ is a nodal shape function satisfying $\sum_i N_i^G(\boldsymbol{x}) = 1$ at any location $\boldsymbol{x}$, $N_i^G(\boldsymbol{x}_i) = 1$ at the i^{th} node (with position $\boldsymbol{x}_i$), and $N_i^G(\boldsymbol{x}_j) = \delta_{ij}$ (the Kronecker delta). To construct $\boldsymbol{a}^G$ and $\boldsymbol{v}^G$, a lumped nodal mass m_i^G,

$$m_i^G = \int_\Omega N_i^G \rho^P \, dV \tag{2.2}$$

momentum $\boldsymbol{p}_i^G$

$$\boldsymbol{p}_i^G = \int_\Omega N_i^G \rho^P \boldsymbol{v}^P \; dV \tag{2.3}$$

and force $\boldsymbol{f}_i^G$

$$\boldsymbol{f}_i^G = \int_\Omega N_i^G \rho^P \boldsymbol{a}^P \; dV \tag{2.4}$$

are determined first from particles, where ρ is mass density and Ω is a computational domain. The right-hand-side of Equation (2.4) has to be further processed because $\boldsymbol{a}^P$ is an unknown that needs to be determined. Assuming a smooth stress distribution in the particle domain, conservation of momentum can be written in differential form as

$$\rho \boldsymbol{a} = \operatorname{div} \boldsymbol{\sigma} + \boldsymbol{b} \tag{2.5}$$

where $\boldsymbol{\sigma}$ is the Cauchy stress tensor and $\boldsymbol{b}$ is a force density vector. Using a Galerkin weak form for Equation (2.5) with test function N_i^G and substituting back in Equation (2.4), the nodal force $\boldsymbol{f}_i^G$ can be expressed as

$$\boldsymbol{f}_i^G = \int_\Omega \rho^P \left(\operatorname{grad} N_i^G \cdot \bar{\boldsymbol{\sigma}}^P + N_i^G \bar{\boldsymbol{b}}^P \right) \; dV - \int_{\partial\Omega} N_i^G \boldsymbol{\tau}^P \; dS \tag{2.6}$$

where $\boldsymbol{\tau}$ is a (prescribed) surface traction, $\bar{\boldsymbol{\sigma}} = \boldsymbol{\sigma}/\rho$ is a specific stress tensor and $\bar{\boldsymbol{b}} = \boldsymbol{b}/\rho$ is a specific force (gravity) vector. Finally, $\boldsymbol{a}_i^G = \boldsymbol{f}_i^G / m_i^G$ and $\boldsymbol{v}_i^G = \boldsymbol{p}_i^G / m_i^G$.

The last missing piece is how to describe and record the particle domain. In the literature, there are two different ways to look at it. The first one, restricting the definition of material points, only tracks and records variable field values at particle location (Sulsky et al., 1995; Zhang et al., 2011). In other words, the selected particles are sample points of objects in the analysis. The other one tries to record particle variable fields with interpolating polynomials (Bardenhagen & Kober, 2004; Sadeghirad et al., 2011). In this latter scheme particles represent certain support regions with predefined characteristic (shape) functions. More details about these implementation strategies of material point methods are introduced in the following

sections, respectively.

2.2 The Material Point Method (MPM)

The Material Point Method (MPM) tracks states of objects of interest using material points. Assuming the same time evolution of variable fields on the grid and particle domains, the states of material points can hence be updated with variable increments in the grid domains. For instance,

$$\Delta \boldsymbol{v}_p^{(n)} = \Delta \boldsymbol{v}^G(\boldsymbol{x}_p, t_n) = \Delta t \sum_i N_i^G(\boldsymbol{x}_p) \boldsymbol{a}_i^G(t_n) \tag{2.7}$$

where a superscript (n) means a value at time t_n and a subscript p identifies variable values of a specific particle, p, located at $\boldsymbol{x}_p$, i.e. $(\bullet)_p(t) \equiv (\bullet)^P(\boldsymbol{x}_p, t)$. This requires special treatment of volume integrals where variables are recorded point-wise. In MPM, an approximation for volume integrals, generally called a particle form, is derived using Taylor series. Equation (2.3), a volume integral including variable fields in the particle domain, is arbitrarily taken to illustrate the idea in the following paragraphs.

To find a particle form of the right-hand side of Equation (2.3), $\boldsymbol{v}^P(\boldsymbol{x}, t)$ is first expanded as a Taylor series around particle position $\boldsymbol{x}_p$ in p^{th} particle domain, Ω_p,

$$\boldsymbol{v}^P(\boldsymbol{x}, t) = \boldsymbol{v}_p(t) + \sum_{k=1}^{\infty} \frac{1}{k!} T_p^k \left(\boldsymbol{v}^P(\boldsymbol{x}, t) \right) \tag{2.8}$$

where

$$T_p^k(\bullet) := \sum_{\alpha_1=1}^{3} \sum_{\alpha_2=1}^{3} \cdots \sum_{\alpha_k=1}^{3} \left[\left. \frac{\partial^k (\bullet)}{\partial x_{\alpha_1} \cdots \partial x_{\alpha_k}} \right|_{\boldsymbol{x}=\boldsymbol{x}_p} (\boldsymbol{e}_{\alpha_1} \cdot \boldsymbol{r}_p) \cdots (\boldsymbol{e}_{\alpha_k} \cdot \boldsymbol{r}_p) \right] \tag{2.9}$$

and

$$\boldsymbol{r}_p \equiv (\boldsymbol{x} - \boldsymbol{x}_p) \ \forall \boldsymbol{x} \in \Omega_p \tag{2.10}$$

x_1 to x_3 are three scalar coordinate variables of a three-dimensional Cartesian coordinate system, and $\boldsymbol{e}_1$ to $\boldsymbol{e}_3$ are the respective base vectors. The function $\boldsymbol{v}^P(\boldsymbol{x},t)$ then can be expressed as a direct sum, notated as $\bigoplus$, of the local Taylor series

$$\boldsymbol{v}^P(\boldsymbol{x},t) = \bigoplus_p \left[\boldsymbol{v}_p(t) + \sum_{k=1}^{\infty} \frac{1}{k!} T_p^k \left(\boldsymbol{v}^P(\boldsymbol{x},t) \right) \right]_{\boldsymbol{x}\in\Omega_p} \tag{2.11}$$

Similarly, N_i^G can be expanded around each particle as

$$N_i^G(\boldsymbol{x}) = N_i^G(\boldsymbol{x}_p) + \sum_{k=1}^{\infty} \frac{1}{k!} T_p^k \left(N_i^G(\boldsymbol{x}) \right) \tag{2.12}$$

Substituting the series representations of both $\boldsymbol{v}^P(\boldsymbol{x},t)$ and N_i^G into Equation (2.3) yields

$$\begin{aligned} \int_\Omega \rho^P N_i^G \boldsymbol{v}^P \, dV = & \sum_p m_p N_i^G(\boldsymbol{x}_p) \boldsymbol{v}_p \\ & + \sum_p \left[\boldsymbol{v}_p \otimes \operatorname{grad} N_i^G(\boldsymbol{x}_p) + N_i^G(\boldsymbol{x}_p) \operatorname{grad} \boldsymbol{v}_p \right] \cdot \int_{\Omega_p} \rho^P \boldsymbol{r}_p \, dV \\ & + O(\ell_p^5) \end{aligned} \tag{2.13}$$

where $m_p = \int_{\Omega_p} \rho^P \, dV$ and ℓ_p is a characteristic particle dimension. Generally, only the first term is taken

$$\int_\Omega \rho^P N_i^G \boldsymbol{v}^P \, dV \approx \sum_p m_p N_i^G(\boldsymbol{x}_p) \boldsymbol{v}_p \tag{2.14}$$

and hence the error in this approximation would be $O(\ell_p^4)$. If particles are located at mass centers of each particle domain, the second term on the right hand side of Equation (2.13) goes to zero automatically, $\int_{\Omega_p} \rho^P \boldsymbol{r}_p \, dV = \boldsymbol{0}$, and hence error of the particle form can reduce to $O(\ell_p^5)$.

Using this approximation, Equations (2.2) to (2.4) can be rewritten as

$$m_i^G \approx \sum_p m_p N_i^G(\boldsymbol{x}_p) \tag{2.15}$$

$$\boldsymbol{p}_i^G \approx \sum_p m_p N_i^G(\boldsymbol{x}_p)\boldsymbol{v}_p \tag{2.16}$$

$$\boldsymbol{f}_i^G \approx \sum_p m_p \left(\operatorname{grad} N_i^G(\boldsymbol{x}_p)\cdot\bar{\boldsymbol{\sigma}}_p + N_i^G(\boldsymbol{x}_p)\bar{\boldsymbol{b}}_p\right) - \int_{\partial\Omega} N_i^G \boldsymbol{\tau}^P \, dS \tag{2.17}$$

These particle forms basically give good approximation and converge quickly as numbers of particles increases and cell sizes decreases with any smooth functions and can be used while values at $\boldsymbol{x}_p$ are known. However, when looking at Equation (2.17) $\operatorname{grad} N_i^G$ is discontinuous if linear shape functions are used in the grid domain. This implies that extra errors may occur when using this approximation if the p^{th} particle domain is near a cell boundary, i.e. $\Omega_p \cap \Omega_{cell} \neq \varnothing$ and $\Omega_p \cup \Omega_{cell} \neq \Omega_{cell}$. This error jumps from one extreme value to the opposite one when the particle crosses a boundary from one cell to another, and can cause destabilizing noise. This cell-crossing noise is a weakness of MPM. Mitigating this effect leads to variants of MPM as described in the following sections.

2.3 The Dual Domain Material Point Method (DDMP)

The Dual Domain Material Point Method, DDMP (Zhang et al., 2011) tries to mitigate cell-crossing noise and improve the performance of MPM by introducing two additional variable tensors on grids: the velocity gradient $\tilde{\boldsymbol{L}}^G(\boldsymbol{x},t)$, and the stress $\tilde{\boldsymbol{\sigma}}^G(\boldsymbol{x},t)$ to adjust the corresponding problematic terms, $\operatorname{grad}\boldsymbol{v}^G = \sum_i \boldsymbol{v}_i^G \otimes \operatorname{grad} N_i^G$ and $\int_\Omega \boldsymbol{\sigma}^P \cdot \operatorname{grad} N_i^G \, dV$, both of which include the discontinuous function $\operatorname{grad} N_i^G$ if N_i^G is only C^0-continuous along cell boundaries. These two tensor variables,

$$\tilde{\boldsymbol{L}}^G(\boldsymbol{x},t) = \sum_i N_i^G(\boldsymbol{x})\,\tilde{\boldsymbol{L}}_i^G(t) \tag{2.18}$$

and

$$\tilde{\boldsymbol{\sigma}}^G(\boldsymbol{x},t) = \sum_i N_i^G(\boldsymbol{x})\,\tilde{\boldsymbol{\sigma}}_i^G(t) \tag{2.19}$$

utilize the same interpolation introduced in Equation (2.1), and are evaluated with weighted arithmetic averaging as

$$\tilde{\boldsymbol{L}}_i^{\,G} = \frac{\int_\Omega N_i^G \mathrm{grad}\,\boldsymbol{v}^G\,dV}{\int_\Omega N_i^G\,dV} = \sum_j \boldsymbol{v}_j^G \otimes \frac{\int_\Omega N_i^G \mathrm{grad}\,N_j^G\,dV}{\int_\Omega N_i^G\,dV} \tag{2.20}$$

and

$$\tilde{\boldsymbol{\sigma}}_i^G = \frac{\int_\Omega N_i^G \boldsymbol{\sigma}^P\,dV}{\int_\Omega N_i^G\,dV} \tag{2.21}$$

respectively. The interpolated velocity gradient $\tilde{\boldsymbol{L}}^G$ is substituted for $\mathrm{grad}\,\boldsymbol{v}^G$, calculated from the velocity field, and is used to update the strain tensor. Nonetheless, for the volume integral $\int_\Omega \boldsymbol{\sigma}^P \cdot \mathrm{grad}\,N_i^G\,dV$, DDMP only partially replaces $\boldsymbol{\sigma}^P$ with $\tilde{\boldsymbol{\sigma}}^G$ for stability considerations and hence introduces a dual domain (G and P) approximation for the stress field;

$$\int_\Omega \boldsymbol{\sigma}^P \cdot \mathrm{grad}\,N_i^G\,dV = \int_\Omega \left(\alpha(\boldsymbol{x})\boldsymbol{\sigma}^P + (1-\alpha(\boldsymbol{x}))\boldsymbol{\sigma}^P\right) \cdot \mathrm{grad}\,N_i^G\,dV \tag{2.22}$$

$$= \int_\Omega \alpha(\boldsymbol{x})\boldsymbol{\sigma}^P \cdot \mathrm{grad}\,N_i^G\,dV + \int_\Omega (1-\alpha(\boldsymbol{x}))\tilde{\boldsymbol{\sigma}}^G \cdot \mathrm{grad}\,N_i^G\,dV \tag{2.23}$$

in which $0 \leq \alpha \leq 1$ is a coefficient function, and weak equality is applied to connect $\boldsymbol{\sigma}^P$ and $\tilde{\boldsymbol{\sigma}}^G$ using $\mathrm{grad}\,N_i^G$ as a test function:

$$\int_\Omega \left(\tilde{\boldsymbol{\sigma}}^G - \boldsymbol{\sigma}^P\right) \cdot \mathrm{grad}\,N_i^G\,dV = \mathbf{0} \tag{2.24}$$

More details about the stability issue of using $\tilde{\boldsymbol{\sigma}}^G$ and the performance of different coefficient functions $\alpha(\boldsymbol{x})$ can be found in Zhang et al. (2011).

2.4 The Generalized Integration Material Point Method (GIMP)

The Generalized Integration Material Point Method (GIMP) interpolates variable fields of particles, $(\mathring{\bullet})^P(\boldsymbol{x},t)$, with

$$(\mathring{\bullet})^P(\boldsymbol{x},t) \approxeq (\bullet)^P(\boldsymbol{x},t) = \sum_j \chi_j^P(\boldsymbol{x})(\bullet)_j^P(t) \tag{2.25}$$

where $\chi_j^P(\boldsymbol{x})$ is a non-negative characteristic function with value no larger than 1 in its supported region Ω_j and 0 when $\boldsymbol{x} \notin \Omega_j$. Furthermore, $\sum_j \chi_j^P(\boldsymbol{x}) = 1$ for all $\boldsymbol{x}$ where particle-represented objects exist. Interpolation of the particle domain is the major difference between MPM and GIMP, and the key to eliminate cell-crossing noise, because volume integrals can now be evaluated exactly. More discussion about the interpolated particle domain is addressed in the literature (Bardenhagen & Kober, 2004).

Weighted arithmetic averaging is used in GIMP to build the interpolated variable fields in the particle domain. Therefore, initial conditions of the fields can be determined with

$$(\bullet)_j^P(0) = \frac{1}{V_{jo}^P}\int_{\Omega_{jo}} \chi_j^P(\mathring{\bullet})^P(\boldsymbol{x},0)\ dV \tag{2.26}$$

where

$$V_{jo}^P = \int_{\Omega_{jo}} \chi_j^P\ dV \tag{2.27}$$

and Ω_{jo} represents the j^{th} supported region at $t = 0$. Similarly, incremental values at particles can be calculated with

$$(\bullet)_j^P = \frac{1}{V_j^P}\int_{\Omega_j} \chi_j^P(\bullet)^G\ dV \tag{2.28}$$

where

$$V_j^P = \int_{\Omega_j} \chi_j^P\ dV \tag{2.29}$$

Considering Equation (2.17), a particle-form expression can be derived by substituting variable fields of particles with interpolated ones,

$$\boldsymbol{f}_i^G = \sum_j \left(\boldsymbol{\sigma}_j^P \cdot \int_{\Omega_j} \operatorname{grad} N_i^G \chi_j^P \, dV + (\rho \bar{\boldsymbol{b}})_j^P \int_{\Omega_j} N_i^G \chi_j^P \, dV \right) - \int_{\partial\Omega} N_i^G \boldsymbol{\tau}^P \, dS \quad (2.30)$$

Here

$$(\rho \bar{\boldsymbol{b}})_j^P = \frac{\int_{\Omega_j} \chi_j^P \mathring{\rho}^P \mathring{\bar{\boldsymbol{b}}}^P \, dV}{\int_{\Omega_j} \chi_j^P \, dV} = \frac{\boldsymbol{f}^{\boldsymbol{b}P}_j}{V_j^P} = \frac{m_j^P \bar{\boldsymbol{b}}_j^P}{V_j^P} \quad (2.31)$$

and

$$\bar{\boldsymbol{b}}_j^P = \frac{\int_{\Omega_j} \chi_j^P \mathring{\bar{\boldsymbol{b}}}^P \, dV}{V_j^P} \quad (2.32)$$

For conservation of mass, m_j^P must be independent of time, i.e.

$$m_j^P = \int_{\Omega_j} \chi_j^P \mathring{\rho}^P \, dV = \int_{\Omega_{jo}} \chi_j^P \mathring{\rho}_o^P \, dV \quad (2.33)$$

where $\mathring{\rho}_o^P$ means mass density at the initial time. Using Equation (2.31), Equation (2.30) becomes

$$\boldsymbol{f}_i^G = \sum_j \left(\boldsymbol{\sigma}_j^P \cdot \boldsymbol{\nabla} \boldsymbol{S}_{\boldsymbol{ij}} + m_j^P \bar{\boldsymbol{b}}_j^P S_{ij} \right) - \int_{\partial\Omega} N_i^G \boldsymbol{\tau}^P \, dS \quad (2.34)$$

with

$$S_{ij} = \frac{1}{V_j^P} \int_{\Omega_j} N_i^G \chi_j^P \, dV \quad (2.35)$$

$$\boldsymbol{\nabla} \boldsymbol{S}_{\boldsymbol{ij}} = \int_{\Omega_j} \chi_j^P \operatorname{grad} N_i^G \, dV \quad (2.36)$$

Following a similar procedure, Equations (2.2) and (2.3) can also be rephrased as

$$m_i^G = \sum_j m_j^P S_{ij} \quad (2.37)$$

and

$$\boldsymbol{p}_i^G = \sum_j m_j^P \boldsymbol{v}_j^P S_{ij} \tag{2.38}$$

In the end, with m_j^P, $\boldsymbol{p}_i^G$ and $\boldsymbol{f}_i^G$, the rate of change of particle variable fields can be determined using weighted arithmetic averages. For instance,

$$\boldsymbol{a}_j^P = \sum_i S_{ij} \boldsymbol{a}_i^G \tag{2.39}$$

Interpolated particle domains do eliminate cell-crossing noise but also reduce computational efficiency tracking shapes of Ω_j and calculating volume integrals S_{ij} and $\boldsymbol{\nabla S_{ij}}$. To simplify the procedure and speed up the algorithm, stretch-only discontinuous rectangular pieces, represented by their own centroids (which are the material points) are generally used to approximate the shapes of particle-supported regions in Bardenhagen and Kober (2004). However, because only stretches are recorded, rotation and shape distortion caused by shear cannot be captured by this implementation and hence introduces error in the particle forms.

2.5 The Convected Partial Domain Integration Method (CPDI)

To improve GIMP, the Convected Partial Domain Integration Method (CPDI) introduces an approximation scheme that simplifies and speeds up the procedure of calculating S_{ij} and $\boldsymbol{\nabla S_{ij}}$. Consequently, more complicated geometry can be used for particle support regions, like parallelepipeds (Sadeghirad et al., 2011) and hexahedra (Sadeghirad, Brannon, & Guilkey, 2013) in three-dimensional problems. In CPDI a constant-valued approximation (presence) function is generally used,

$$\chi_j^P(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \boldsymbol{x} \in \Omega_j \\ 0 & \text{if } \boldsymbol{x} \notin \Omega_j \ . \end{cases} \tag{2.40}$$

Particle support regions Ω_j are non-overlapping, and shape functions N_i^G in each particle domain Ω_j are smoothed with an interpolation,

$$N_i^G(\boldsymbol{x}) \approx N_i^{G.app}(\boldsymbol{x}) = \sum_{m=1}^{N_{corner}} N_i^G(\boldsymbol{x}_m) N_m(\boldsymbol{x}) \tag{2.41}$$

for $\boldsymbol{x} \in \Omega_j$, where $\boldsymbol{x}_m$ means the coordinate of m^{th} corner of Ω_j and N_m is a trilinear shape function with nodal expression. Substituting Equation (2.40) and Equation (2.41) back to Equation (2.35) and Equation (2.36),

$$S_{ij} \approx \sum_{m=1}^{N_{corner}} \frac{N_i^G(\boldsymbol{x}_m)}{V_j^P} \int_{\Omega_j} N_m \, dV \tag{2.42}$$

$$\boldsymbol{\nabla S_{ij}} \approx \sum_{m=1}^{N_{corner}} N_i^G(\boldsymbol{x}_m) \int_{\Omega_j} \operatorname{grad} N_m \, dV \tag{2.43}$$

which saves time integration and avoids finding intersections between grid cells and particle-supported regions. (Sadeghirad et al., 2011)

2.6 Method Used in Present Research

So far in this chapter, MPM and its branches, DDMP, GIMP and CPDI, have been introduced. Although GIMP and CPDI can reduce the cell-crossing error of volume integrals, the interpolated particle domain increases computing cost and also causes integration errors when the particles can not describe the deformed continuum correctly. Also, adapting multi-field formulations to MPM (Chapter 3) introduces similar smoothing effect as DDMP. Therefore, standard MPM is used in this research.

Like in many general applications in the literature, a regular cuboid grid pattern with linear shape functions is used for efficiency. However, this approach also introduces locking and causes errors matching boundary conditions. A numerical flux smoothing algorithm is proposed here to improve existing anti-locking algorithms

(Chapter 3). Moreover, a new technique is developed to mitigate the errors at boundaries (Chapter 4). In order to examine and implement the new proposed algorithms, a material point method program is developed with the C++ programming language for this research: Continuum Analysis Program (CAP). Some details and philosophy of the program implementation can be found in Appendix A.

Chapter 3

MODELING FLUID WITH THE MPM

In many MPM applications, regular orthogonal grids with tri-linear shape functions are used to simplify implementation and maintain computational efficiency. However, linear shape functions also introduce volumetric locking for incompressible and nearly incompressible materials, hence causing problems when modeling liquids. To eliminate locking, hybrid formulations are generally used in finite element methods and were adapted to MPM by Mast et al. (2012). In this previous work two classes of anti-locking algorithms were developed for fluid modeling, one cell-based and the other node-based. These anti-locking algorithms not only filter out unwanted strain and stress fields responsible for locking, but also can reduce the impact of integration errors and consequently help stabilize MPM calculations. Although very useful, these methods are not perfect. In the case of hydrodynamic modeling the cell-based algorithm does not fully smooth out error-induced destabilization, and this can result in high frequency oscillations, noisy stress fields, and, even worse, instabilities. On the other hand, the node-based algorithm can smooth out the strain fields excessively, leading to inaccurate response in hydrostatic problems, even though it gives a relatively smooth stress field in complex hydrodynamic problems. In the present study, the term "anti-locking algorithm" follows the usage in FEM but not in the work of Mast et al. (2012), in which this term includes both the anti-locking effect and the smoothing effect. The goal of the smoothing is to stabilize high frequency oscillations caused by integration errors. In this chapter a new numerical flux smoothing algorithm is proposed to give stable analysis in complex hydrodynamic problems similar to the earlier node-based approach, and to be able to solve hydrostatic problems as

in the case of the cell-based algorithm.

3.1 Anti-Locking Algorithm

3.1.1 Hu-Washizu Weak Forms

The Hu-Washizu variational principle (H.-C. Hu, 1954; Washizu, 1982) includes three independent *interpolated* variable fields, the velocity $\boldsymbol{v}(\boldsymbol{x},t)$, the stress $\tilde{\boldsymbol{\sigma}}(\boldsymbol{x},t)$ and the rate of deformation (or the strain rate) $\tilde{\boldsymbol{d}}(\boldsymbol{x},t)$. The variables $\tilde{\boldsymbol{\sigma}}$ and $\tilde{\boldsymbol{d}}$ are generally called assumed stress and assumed strain rate to distinguish them from the stress $\boldsymbol{\sigma}(\tilde{\boldsymbol{d}},\text{etc.})$ evaluated directly with constitutive equations and the rate of deformation

$$\boldsymbol{d} = \frac{1}{2}(\text{grad}\,\boldsymbol{v} + \text{grad}^{\,T}\boldsymbol{v}) \tag{3.1}$$

which is the symmetric part of the velocity gradient. Separating the boundary $\partial\Omega$ of a continuum domain Ω into two regions, $\partial\Omega^{\tau}$ and $\partial\Omega^{v}$, on which tractions and velocity fields are described with $\boldsymbol{\tau}^*$ and $\boldsymbol{v}^*$, respectively, the Hu-Washizu principle is expressed as the weak form

$$\begin{aligned} 0 &= \delta\Pi_{HW}(\boldsymbol{v},\tilde{\boldsymbol{\sigma}},\tilde{\boldsymbol{d}};\delta\boldsymbol{v},\delta\tilde{\boldsymbol{\sigma}},\delta\tilde{\boldsymbol{d}}) \\ &= \int_{\Omega}\left\{\delta\tilde{\boldsymbol{d}}:\boldsymbol{\sigma}+\delta\left[\tilde{\boldsymbol{\sigma}}:\left(\boldsymbol{d}-\tilde{\boldsymbol{d}}\right)\right]+\delta\boldsymbol{v}\cdot\rho\left(\boldsymbol{a}-\bar{\boldsymbol{b}}\right)\right\}\,dV-\int_{\partial\Omega^{\tau}}\boldsymbol{\tau}^*\cdot\delta\boldsymbol{v}\,dS \end{aligned} \tag{3.2}$$

in which ρ is mass density, $\boldsymbol{a}$ is the acceleration, and $\bar{\boldsymbol{b}}$ is the specific (body) force. Because $\delta\boldsymbol{v}$, $\delta\tilde{\boldsymbol{\sigma}}$ and $\delta\tilde{\boldsymbol{d}}$ are independent and arbitrary functions, the original weak form leads to three independent ones

$$\int_{\Omega}\delta\tilde{\boldsymbol{d}}:(\boldsymbol{\sigma}-\tilde{\boldsymbol{\sigma}})\;dV=0 \tag{3.3}$$

$$\int_{\Omega}\delta\tilde{\boldsymbol{\sigma}}:\left(\boldsymbol{d}-\tilde{\boldsymbol{d}}\right)\;dV=0 \tag{3.4}$$

and

$$\begin{aligned}\int_\Omega \delta\boldsymbol{v}\cdot\rho\boldsymbol{a}\;dV &= -\int_\Omega \delta\boldsymbol{d}:\tilde{\boldsymbol{\sigma}}\;dV+\int_\Omega \delta\boldsymbol{v}\cdot\rho\bar{\boldsymbol{b}}\;dV+\int_{\partial\Omega^\tau}\boldsymbol{\tau}^*\cdot\delta\boldsymbol{v}\;dS\\ &= -\int_\Omega \operatorname{grad}\delta\boldsymbol{v}:\tilde{\boldsymbol{\sigma}}\;dV+\int_\Omega \delta\boldsymbol{v}\cdot\rho\bar{\boldsymbol{b}}\;dV+\int_{\partial\Omega^\tau}\boldsymbol{\tau}^*\cdot\delta\boldsymbol{v}\;dS \quad (3.5)\end{aligned}$$

where

$$\begin{aligned}\delta\boldsymbol{d}:\tilde{\boldsymbol{\sigma}} &= \delta\underbrace{\left[\frac{1}{2}(\operatorname{grad}\boldsymbol{v}+\operatorname{grad}^T\boldsymbol{v})\right]}_{\text{Equation (3.1)}}:\tilde{\boldsymbol{\sigma}}\\ &= \frac{1}{2}(\operatorname{grad}\delta\boldsymbol{v}+\operatorname{grad}^T\delta\boldsymbol{v}):\tilde{\boldsymbol{\sigma}}\\ &= \frac{1}{2}(\operatorname{grad}\delta\boldsymbol{v}+\operatorname{grad}^T\delta\boldsymbol{v}):\tilde{\boldsymbol{\sigma}}+\underbrace{\frac{1}{2}(\operatorname{grad}\delta\boldsymbol{v}-\operatorname{grad}^T\delta\boldsymbol{v}):\tilde{\boldsymbol{\sigma}}}_{=0,\text{ because }\tilde{\boldsymbol{\sigma}}\text{ is symmetric}}\\ &= \operatorname{grad}\delta\boldsymbol{v}:\tilde{\boldsymbol{\sigma}} \quad (3.6)\end{aligned}$$

Equations (3.3) and (3.4) are used to evaluate the assumed fields $\tilde{\boldsymbol{\sigma}}$ and $\tilde{\boldsymbol{d}}$. Equation (3.5) is similar to the conservation weak equation in displacement finite elements, but has the assumed stress in the internal force (second) term. (Belytschko, Liu, & Moran, 2000)

3.1.2 Volumetric-Deviatoric Decomposition

When volumetric locking is the major concern, e.g. handling nearly incompressible materials, it is sufficient to only approximate the volumetric part of both the stress and the strain rate fields. In this case,

$$\begin{aligned}0 &= \delta\Pi_{HW}(\boldsymbol{v},\tilde{\sigma}_v,\tilde{d}_v;\delta\boldsymbol{v},\delta\tilde{\sigma}_v,\delta\tilde{d}_v)\\ &= \int_\Omega \delta\boldsymbol{v}\cdot\rho\left(\boldsymbol{a}-\bar{\boldsymbol{b}}\right)\;dV-\int_{\partial\Omega^\tau}\boldsymbol{\tau}^*\cdot\delta\boldsymbol{v}\;dS+\delta P^{int} \quad (3.7)\end{aligned}$$

in which

$$\begin{aligned}
\delta P^{int} &= \int_\Omega \left\{ \delta \left[\tilde{d}_v \mathbb{1} + \boldsymbol{d}^{dev} \right] : \boldsymbol{\sigma} + \delta \left[\left(\tilde{\sigma}_v \mathbb{1} + \boldsymbol{\sigma}^{dev} \right) : \left(\boldsymbol{d}^{vol} - \tilde{d}_v \mathbb{1} \right) \right] \right\} \, dV \\
&= \int_\Omega \left[\delta \tilde{d}_v \mathbb{1} : \boldsymbol{\sigma}^{vol} + \delta \boldsymbol{d}^{dev} : \boldsymbol{\sigma}^{dev} \right] \, dV + \\
&\quad \int_\Omega \left[\delta \tilde{\sigma}_v \mathbb{1} : \left(\boldsymbol{d}^{vol} - \tilde{d}_v \mathbb{1} \right) + \tilde{\sigma}_v \mathbb{1} : \left(\delta \boldsymbol{d}^{vol} - \delta \tilde{d}_v \mathbb{1} \right) \right] \, dV \\
&= \int_\Omega \left[\delta \boldsymbol{d} : \left(\tilde{\sigma}_v \mathbb{1} + \boldsymbol{\sigma}^{dev} \right) \right] \, dV + \\
&\quad \int_\Omega \left[\delta \tilde{d}_v \mathbb{1} : \left(\boldsymbol{\sigma}^{vol} - \tilde{\sigma}_v \mathbb{1} \right) + \delta \tilde{\sigma}_v \mathbb{1} : \left(\boldsymbol{d}^{vol} - \tilde{d}_v \mathbb{1} \right) \right] \, dV \\
&= \int_\Omega \left[\operatorname{grad} \delta \boldsymbol{v} : \left(\tilde{\sigma}_v \mathbb{1} + \boldsymbol{\sigma}^{dev} \right) \right] \, dV + \\
&\quad \int_\Omega \left[\delta \tilde{d}_v \left(\mathbb{1} : \boldsymbol{\sigma} - 3 \tilde{\sigma}_v \right) + \delta \tilde{\sigma}_v \left(\mathbb{1} : \boldsymbol{d} - 3 \tilde{d}_v \right) \right] \, dV
\end{aligned} \tag{3.8}$$

$\mathbb{1}$ is a second order identity tensor; superscripts *vol* and *dev* indicate a volumetric and deviatoric part of a second order tensor, respectively. Using the arbitrariness and independence of $\delta \boldsymbol{v}$, $\delta \tilde{\sigma}_v$ and $\delta \tilde{d}_v$, the weak form is separated into

$$\int_\Omega \delta \tilde{d}_v \left(\frac{\operatorname{tr} (\boldsymbol{\sigma})}{3} - \tilde{\sigma}_v \right) \, dV = 0 \tag{3.9}$$

$$\int_\Omega \delta \tilde{\sigma}_v \left(\frac{\operatorname{tr} (\boldsymbol{d})}{3} - \tilde{d}_v \right) \, dV = 0 \tag{3.10}$$

and

$$\int_\Omega \delta \boldsymbol{v} \cdot \rho \boldsymbol{a} \, dV = - \int_\Omega \operatorname{grad} \delta \boldsymbol{v} : \left(\tilde{\sigma}_v \mathbb{1} + \boldsymbol{\sigma}^{dev} \right) \, dV + \int_\Omega \delta \boldsymbol{v} \cdot \rho \bar{\boldsymbol{b}} \, dV + \int_{\partial \Omega^\tau} \boldsymbol{\tau}^* \cdot \delta \boldsymbol{v} \, dS \tag{3.11}$$

Similarly, the multi-field weak form equations can be written for assumed fields with only deviatoric parts:

$$\int_\Omega \delta \tilde{\boldsymbol{d}}_{\boldsymbol{d}} : \left(\boldsymbol{\sigma}^{dev} - \tilde{\boldsymbol{\sigma}}_{\boldsymbol{d}} \right) \, dV = 0 \tag{3.12}$$

$$\int_\Omega \delta\tilde{\boldsymbol{\sigma}}_{\boldsymbol{d}} : \left(\boldsymbol{d}^{dev} - \tilde{\boldsymbol{d}}_{\boldsymbol{d}}\right)\ dV = 0 \tag{3.13}$$

and

$$\int_\Omega \delta\boldsymbol{v}\cdot\rho\boldsymbol{a}\ dV = -\int_\Omega \operatorname{grad}\delta\boldsymbol{v} : \left(\boldsymbol{\sigma}^{vol} + \tilde{\boldsymbol{\sigma}}_{\boldsymbol{d}}\right)\ dV + \int_\Omega \delta\boldsymbol{v}\cdot\rho\bar{\boldsymbol{b}}\ dV + \int_{\partial\Omega^\tau} \boldsymbol{\tau}^*\cdot\delta\boldsymbol{v}\ dS \tag{3.14}$$

Equations (3.12)-(3.14) are targeted on curing shear locking problems and are not commonly explicitly used in finite element methods because the number of assumed variables in these equations is the same as the number used in Hu-Washizu weak formulas. In contrast, using Equations (3.9)-(3.11) reduces the additional variables, compared to displacement finite elements, down to two and hence increases computing efficiency. In any case, these weak form equations offer a remedy for locking problems and are extended to MPM by Mast et al. (2012). This will be described in more detail in the next section.

3.1.3 The MPM Implementation of Hu-Washizu Principle

To adapt the multi-field weak forms to the MPM, $\boldsymbol{\sigma}$ and $\tilde{\boldsymbol{\sigma}}$ are substituted with $\rho\bar{\boldsymbol{\sigma}}$ and $\rho\tilde{\bar{\boldsymbol{\sigma}}}$, respectively (Mast et al., 2012). Consequently, the weak form equations given by the Hu-Washizu variational principle expressed in Equations (3.3)-(3.5) can be rephrased as

$$\int_\Omega \delta\tilde{\boldsymbol{d}} : \rho\left(\bar{\boldsymbol{\sigma}} - \tilde{\bar{\boldsymbol{\sigma}}}\right)\ dV = 0 \tag{3.15}$$

$$\int_\Omega \rho\delta\tilde{\bar{\boldsymbol{\sigma}}} : \left(\boldsymbol{d} - \tilde{\boldsymbol{d}}\right)\ dV = 0 \tag{3.16}$$

and

$$\int_\Omega \delta\boldsymbol{v}\cdot\rho\boldsymbol{a}\ dV = -\int_\Omega \operatorname{grad}\delta\boldsymbol{v} : \rho\tilde{\bar{\boldsymbol{\sigma}}}\ dV + \int_\Omega \delta\boldsymbol{v}\cdot\rho\bar{\boldsymbol{b}}\ dV + \int_{\partial\Omega^\tau} \boldsymbol{\tau}^*\cdot\delta\boldsymbol{v}\ dS \tag{3.17}$$

Letting the particle and grid domains (Section 2.1) have the same assumed fields:

$$\tilde{\bar{\boldsymbol{\sigma}}}^P(\boldsymbol{x},t) = \tilde{\bar{\boldsymbol{\sigma}}}^G(\boldsymbol{x},t) = \sum_i \chi_i^G(\boldsymbol{x})\, \tilde{\bar{\boldsymbol{\sigma}}}_i^G(t) \tag{3.18}$$

and

$$\tilde{\boldsymbol{d}}^P(\boldsymbol{x},t) = \tilde{\boldsymbol{d}}^G(\boldsymbol{x},t) = \sum_i \chi_i^G(\boldsymbol{x})\, \tilde{\boldsymbol{d}}_i^G(t) \tag{3.19}$$

with superscripts P and G referring to variable fields on the particle and grid domains. Equations (3.15)-(3.17) can be derived into MPM-styled equations as

$$\sum_j \left(\int_\Omega \rho^P \chi_i^G \chi_j^G \; dV \right) \tilde{\bar{\boldsymbol{\sigma}}}_j^G = \int_\Omega \rho^P \chi_i^G \bar{\boldsymbol{\sigma}}^P \; dV \tag{3.20}$$

$$\sum_j \left(\int_\Omega \rho^P \chi_i^G \chi_j^G \; dV \right) \tilde{\boldsymbol{d}}_j^G = \int_\Omega \rho^P \chi_i^G \boldsymbol{d}^P \; dV \tag{3.21}$$

and

$$\int_\Omega N_i^G \rho^P \boldsymbol{a}^P \; dV = -\int_\Omega \rho^P \mathrm{grad}\, N_i^G \cdot \tilde{\bar{\boldsymbol{\sigma}}}^P \; dV + \int_\Omega N_i^G \rho^P \bar{\boldsymbol{b}}^P \; dV + \int_{\partial\Omega^\tau} \boldsymbol{\tau}^{*P} N_i^G \; dS \tag{3.22}$$

where

$$\boldsymbol{d}^P = \boldsymbol{d}^G = \frac{1}{2}\left(\mathrm{grad}\, \boldsymbol{v}^G + \mathrm{grad}^T \boldsymbol{v}^G\right) \tag{3.23}$$

The interpolant χ_i^G of the assumed fields is then something that can be modified within this algorithm as described in the following.

3.1.4 One-Point Quadrature Element

In FEM, the assumed fields $\tilde{\bar{\boldsymbol{\sigma}}}$ and $\tilde{\boldsymbol{d}}$ are generally interpolated on an element level to avoid global interpolation and hence to reduce computational cost. The simplest approach in many of the hybrid finite elements addressing locking is the one-point quadrature element, which is equivalent to using piece-wise constant interpolation for

both assumed stress and strain rate, or

$$\tilde{\boldsymbol{\sigma}}^c = \frac{1}{V^c}\int_{\Omega^c} \boldsymbol{\sigma}\ dV \tag{3.24}$$

and

$$\tilde{\boldsymbol{d}}^c = \frac{1}{V^c}\int_{\Omega^c} \boldsymbol{d}\ dV \tag{3.25}$$

where superscript c indicates the cell under consideration, Ω^c is the domain of the cell, and V^c means the volume of Ω^c. Although it is rank deficient and requires additional stabilization, the one-point quadrature element still plays a main role in finite element communities because of its speed of calculation. Adapting the procedure described in Section 3.1.3 gives a set of cell-based equations for determining assumed fields in MPM:

$$\tilde{\bar{\boldsymbol{\sigma}}}^c = \frac{\int_{\Omega^c} \rho^P \bar{\boldsymbol{\sigma}}^P\ dV}{\int_{\Omega^c} \rho^P\ dV} \tag{3.26}$$

and

$$\tilde{\boldsymbol{d}}^c = \frac{\int_{\Omega^c} \rho^P \boldsymbol{d}^P\ dV}{\int_{\Omega^c} \rho^P\ dV} \tag{3.27}$$

3.1.5 Node-Based Assumed Fields

If interpolants of the assumed stress and strain rate are selected as nodal shape functions, i.e. $\chi_i^G = N_i^G$, and the consistent mass,

$$m_{ij}^G := \int_\Omega \rho^P \chi_i^G \chi_j^G\ dV = \int_\Omega \rho^P N_i^G N_j^G\ dV \tag{3.28}$$

on the left-hand-side of Equations (3.20) and (3.21) are approximated with a lumped mass, then

$$\tilde{\bar{\boldsymbol{\sigma}}}_i^G \approx \frac{1}{m_i^G}\int_\Omega \rho^P N_i^G \bar{\boldsymbol{\sigma}}^P\ dV \tag{3.29}$$

and

$$\tilde{\boldsymbol{d}}_i^G \approx \frac{1}{m_i^G}\int_\Omega \rho^P N_i^G \boldsymbol{d}^P\ dV \tag{3.30}$$

here m_i^G is the lumped mass on node i defined in Equation (2.2). The node-based linear interpolation gives higher resolution of assumed stress and strain than the standard MPM and the one-point quadrature element, but also introduces zero-energy modes and instability, which are briefly discussed in Sections 3.5 and 3.6.

3.1.6 Alternative Hu-Washizu Weak Forms for Small Deformation

When small deformation is considered, the strain $\boldsymbol{\varepsilon}$ can be expressed as a function of the displacement $\boldsymbol{u}$ as follows

$$\boldsymbol{\varepsilon} = \frac{1}{2}\left(\operatorname{grad}\boldsymbol{u} + \operatorname{grad}^T\boldsymbol{u}\right) \tag{3.31}$$

Consequently, two of the three assumed fields in Section 3.1.3, $\boldsymbol{v}$ and $\tilde{\boldsymbol{d}}$ can be reselected as $\boldsymbol{u}$ and the assumed strain $\tilde{\boldsymbol{\varepsilon}}$, and the Cauchy stress $\boldsymbol{\sigma}$ can be expressed as a function of the assumed strain:

$$\boldsymbol{\sigma} = \frac{\partial W(\tilde{\boldsymbol{\varepsilon}})}{\partial \tilde{\boldsymbol{\varepsilon}}} \tag{3.32}$$

where $W(\tilde{\boldsymbol{\varepsilon}})$ is a work function and is usually equivalent to strain energy density. The Hu-Washizu variational principle in this case minimizes the potential energy and gives a variational weak form

$$\begin{aligned} 0 &= \delta\Pi_{HW}(\boldsymbol{u}, \tilde{\boldsymbol{\sigma}}, \tilde{\boldsymbol{\varepsilon}}, \delta\boldsymbol{u}, \delta\tilde{\boldsymbol{\sigma}}, \delta\tilde{\boldsymbol{\varepsilon}}) \\ &= \int_\Omega \delta\tilde{\boldsymbol{\varepsilon}} : \boldsymbol{\sigma} + \delta\left[\tilde{\boldsymbol{\sigma}} : (\boldsymbol{\varepsilon} - \tilde{\boldsymbol{\varepsilon}})\right] + \delta\boldsymbol{u} \cdot \rho\left(\boldsymbol{a} - \boldsymbol{b}\right)\, dV - \int_{\partial\Omega^\tau} \boldsymbol{\tau}^* \cdot \delta\boldsymbol{u}\, dS \end{aligned} \tag{3.33}$$

Using the arbitrariness of $\delta\boldsymbol{u}$, $\delta\tilde{\boldsymbol{\sigma}}$ and $\delta\tilde{\boldsymbol{\varepsilon}}$, the virtual energy equation falls apart into

$$\int_\Omega \delta\tilde{\boldsymbol{\varepsilon}} : (\boldsymbol{\sigma} - \tilde{\boldsymbol{\sigma}})\, dV = 0 \tag{3.34}$$

$$\int_\Omega \delta\tilde{\boldsymbol{\sigma}} : (\boldsymbol{\varepsilon} - \tilde{\boldsymbol{\varepsilon}})\, dV = 0 \tag{3.35}$$

and

$$\int_{\Omega} \delta \boldsymbol{u} \cdot \rho \boldsymbol{a} \; dV = -\int_{\Omega} \operatorname{grad} \delta \boldsymbol{u} : \tilde{\boldsymbol{\sigma}} \; dV + \int_{\Omega} \delta \boldsymbol{u} \cdot \rho \boldsymbol{b} \; dV + \int_{\partial \Omega^{\tau}} \boldsymbol{\tau}^{*} \cdot \delta \boldsymbol{u} \; dS \tag{3.36}$$

Finally, using a similar procedure as described in Section 3.1.3, Equations (3.34) and (3.36) can be derived to the same MPM-styled equations (3.20) and (3.22), respectively; Equation (3.35) yields to

$$\sum_{j} \left(\int_{\Omega} \rho^{P} \chi_{i}^{G} \chi_{j}^{G} \; dV \right) \tilde{\boldsymbol{\varepsilon}}_{j}^{G} = \int_{\Omega} \rho^{P} \chi_{i}^{G} \boldsymbol{\varepsilon}^{P} \; dV \tag{3.37}$$

3.2 Introduction of Smoothing Algorithms

In MPM, only variable increments on the grid are used to update states at material points to avoid diffusion (Chapter 2). However, this can cause problems to the grid's ability to reproduce variable fields on the particle domain when particles start to move across the grid, especially material states, which are updated with the C^{-1} continuous rate of deformation (strain rate) tensor field when linear shape functions are used. Besides, integration errors in MPM result in nonphysical high frequency oscillation and also destabilization. In this context, in this research, material states are smoothed to allow computational grids to reproduce the material state fields, and most importantly to provide stabilization mechanisms by dissipating strain energy.

The idea of using strain energy dissipation as a stabilization mechanism for MPM in fluid modeling is presented in the work of Mast et al. (2012) implicitly. In their work the anti-locking algorithm is based on the Hu-Washizu principle described in Section 3.1.6 (i.e. using assumed strain $\tilde{\boldsymbol{\varepsilon}}$ instead of assumed strain rate $\tilde{\boldsymbol{d}}$). However, the implementation implicitly follows Section 3.1.3 plus an additional smoothing procedure applied to the strain. To illustrate this idea, a linear elastic problem is con-

sidered with a constitutive relation

$$\bar{\boldsymbol{\sigma}} = \bar{\mathbb{C}} : \boldsymbol{\varepsilon} \tag{3.38}$$

in which $\bar{\mathbb{C}}$ is a fourth order specific elasticity tensor. The elastic strain and stress at particle p updated at the end of the n^{th} time step in the standard MPM (Sulsky et al., 1995) are

$$\boldsymbol{\varepsilon}_p^{(n+1)} = \boldsymbol{\varepsilon}_p^{(n)} + \Delta t^{(n)} \left[\boldsymbol{d}^{G(n)} \right]_{\boldsymbol{x}_p^{(n+1)}} \tag{3.39}$$

and

$$\bar{\boldsymbol{\sigma}}_p^{(n+1)} = \bar{\mathbb{C}}_p : \boldsymbol{\varepsilon}_p^{(n+1)} \tag{3.40}$$

If the three-field weak form (Section 3.1.3) is applied, the assumed strain rate is used for the strain (material state) update and hence the strain evolution equation becomes

$$\boldsymbol{\varepsilon}_p^{(n+1)} = \boldsymbol{\varepsilon}_p^{(n)} + \Delta t^{(n)} \left[\tilde{\boldsymbol{d}}^{G(n)} \right]_{\boldsymbol{x}_p^{(n+1)}} \tag{3.41}$$

This research follows the previous work of Mast et al. (2012) smoothing the particle strain field at each time step and hence gives a modified strain update:

$$\boldsymbol{\varepsilon}_p^{(n+1)} = \left[\hat{\boldsymbol{\varepsilon}}^{G(n)} + \Delta t^{(n)} \tilde{\boldsymbol{d}}^{G(n)} \right]_{\boldsymbol{x}_p^{(n+1)}} \tag{3.42}$$

in which $\hat{\boldsymbol{\varepsilon}}^{G(n)}$ is a smoothed strain field calculated with smoothing algorithms introduced in the following sections. These strain and stress evolution equations, (3.42) and (3.40), used in the previous work (Mast et al., 2012) can be easily confused with the alternative Hu-Washizu weak form equations for the small deformation case described in Section 3.1.6, that updates the strain with Equation (3.39) and the stress with

$$\bar{\boldsymbol{\sigma}}_p^{(n+1)} = \bar{\mathbb{C}}_p : \left[\tilde{\boldsymbol{\varepsilon}}^{G(n+1)} \right]_{\boldsymbol{x}_p^{(n+1)}} \tag{3.43}$$

This is because the assumed strain $\tilde{\boldsymbol{\varepsilon}}^{G(n+1)}$ is identical to the strain used in Equation (3.42) if the assumed fields $\tilde{\boldsymbol{d}}^{G(n)}$ and $\tilde{\boldsymbol{\varepsilon}}^{G(n+1)}$ are interpolated with the same functions χ_i^G (which is time invariant during any time steps in MPM framework) and the smoothed strain $\hat{\boldsymbol{\varepsilon}}^{G(n)}$ is selected as an interpolated field also using the interpolant χ_i^G:

$$\hat{\boldsymbol{\varepsilon}}^{G(n)} = \sum_i \chi_i^G \hat{\boldsymbol{\varepsilon}}_i^{G(n)} \tag{3.44}$$

and

$$\begin{aligned} &\sum_j \left(\int_{\Omega^{(n+1)}} \rho^{P(n+1)} \chi_i^G \chi_j^G \, dV \right) \hat{\boldsymbol{\varepsilon}}_j^{G(n)} \\ &\quad = \sum_j \left(\int_{\Omega^{(n)}} \rho^{P(n)} \chi_i^G \chi_j^G \, dV \right) \hat{\boldsymbol{\varepsilon}}_j^{G(n)} = \int_{\Omega^{(n)}} \rho^{P(n)} \chi_i^G \boldsymbol{\varepsilon}^{P(n)} \, dV \end{aligned} \tag{3.45}$$

In this case,

$$\begin{aligned} &\int_{\Omega^{(n+1)}} \rho^{P(n+1)} \chi_i^G \tilde{\boldsymbol{\varepsilon}}^{G(n+1)} \, dV \\ &= \int_{\Omega^{(n+1)}} \rho^{P(n+1)} \chi_i^G \left(\boldsymbol{\varepsilon}^{P(n)} + \Delta t^{(n)} \boldsymbol{d}^{G(n)} \right) \, dV \\ &= \int_{\Omega^{(n)}} \rho^{P(n)} \chi_i^G \boldsymbol{\varepsilon}^{P(n)} \, dV + \Delta t^{(n)} \int_{\Omega^{(n+1)}} \rho^{P(n+1)} \chi_i^G \boldsymbol{d}^{G(n)} \, dV \\ &= \int_{\Omega^{(n+1)}} \rho^{P(n+1)} \chi_i^G \left(\hat{\boldsymbol{\varepsilon}}^{G(n)} + \Delta t^{(n)} \tilde{\boldsymbol{d}}^{G(n)} \right) \, dV \end{aligned} \tag{3.46}$$

which yields to

$$\tilde{\boldsymbol{\varepsilon}}_i^{G(n+1)} = \hat{\boldsymbol{\varepsilon}}_i^{G(n)} + \Delta t^{(n)} \tilde{\boldsymbol{d}}_i^{G(n)} \tag{3.47}$$

corresponding to any interpolant χ_i^G, or simply

$$\tilde{\boldsymbol{\varepsilon}}^{G(n+1)} = \hat{\boldsymbol{\varepsilon}}^{G(n)} + \Delta t^{(n)} \tilde{\boldsymbol{d}}^{G(n)} \tag{3.48}$$

Although they may use the same strain values to update particle stress, the work done

by Mast et al. (2012) and the alternative Hu-Washizu weak form equations for small deformation are different because they have different particle strain fields, that can be easily understood by comparing the strain evolution equations (3.42) and (3.39) which are used in these two algorithms, respectively.

In the work by Mast et al. (2012), the material states (strain) are smoothed by interpolation using Equations (3.44) and (3.45). If χ_i^G are selected as piece-wise continuous functions, Equation (3.45) gives a cell-based smoothing procedure (Section 3.3); If $\chi_i^G = N_i^G$ are the nodal shape functions, solving Equation (3.45) can become time consuming because the mass matrix $\left[m_{ij}^G\right]$ (with m_{ij}^G defined in Equation (3.28)) at the left hand side of Equation (3.45) is global, even though it is banded. In this case, lumped mass is generally used for efficiency and consequently introduces strong diffusion and pollutes the strain field. Problems arising from the node-based smoothing algorithm will be discussed in Sections 3.5 and 3.6, after the derivation of a new proposed smoothing algorithm, the numerical flux smoothing algorithm.

3.3 Cell-Based Smoothing Algorithm

This section introduces the cell-based smoothing algorithm from a more mathematical point of view. The results will then be connected to the specific physical quantity, the (total) strain, and explain why this smoothing procedure introduces strain energy dissipation in the applications considered in this study.

A smooth bounded spatial function $q(\boldsymbol{x})$ with $\boldsymbol{x} = (x, y, z)$ can be expressed as a combination of an infinite polynomial series,

$$q(\boldsymbol{x}) = \sum_{i=0}^{\infty} q_i \phi_i(\boldsymbol{x}) \tag{3.49}$$

Let $\phi_i(\boldsymbol{x})$, $i = 0, 1, \ldots, \infty$, be a set of polynomials with leading coefficient[1] 1, such

[1]The coefficient of the term of highest degree in a polynomial.

that

$$\deg(\phi_{n-1}(\boldsymbol{x})) \leq \deg(\phi_n(\boldsymbol{x})) \ \forall \ n \in \mathbb{N}^+ \tag{3.50}$$

in which $\deg(\bullet)$ gives the degree of a polynomial, and $\mathbb{N}^+$ is a set of positive integers. Moreover, in a considered domain Ω with a selected positive weight function $w(\boldsymbol{x})$ the polynomials can be chosen to be orthogonal so that

$$\int_\Omega w(\boldsymbol{x})\phi_i(\boldsymbol{x})\phi_j(\boldsymbol{x}) \ dV = 0 \tag{3.51}$$

for any i and j when $i \neq j$ and

$$\int_\Omega w(\boldsymbol{x})\phi_i^2(\boldsymbol{x}) \ dV > 0 \tag{3.52}$$

for any i. Consequently,

$$\begin{aligned} \int_\Omega w(\boldsymbol{x})\phi_i(\boldsymbol{x})q(\boldsymbol{x}) \ dV &= \int_\Omega \left[w(\boldsymbol{x})\phi_i(\boldsymbol{x}) \sum_{i=0}^{\infty} q_j\phi_j(\boldsymbol{x}) \right] \ dV & (3.53) \\ &= q_i \int_\Omega w(\boldsymbol{x})\phi_i(\boldsymbol{x})^2 \ dV & (3.54) \end{aligned}$$

or

$$q_i = \frac{\int_\Omega w(\boldsymbol{x})\phi_i(\boldsymbol{x})q(\boldsymbol{x}) \ dV}{\int_\Omega w(\boldsymbol{x})\phi_i^2(\boldsymbol{x}) \ dV} \tag{3.55}$$

When only the first finite number n of polynomials is used, this leads to an approximation (smoothed function), i.e.

$$q(\boldsymbol{x}) \approxeq \hat{q}(\boldsymbol{x}) = \sum_{i=0}^{n} q_i\phi_i(\boldsymbol{x}) \tag{3.56}$$

If a certain energy $\mathbb{E}$ is defined as

$$\mathbb{E} = \int_\Omega w(\boldsymbol{x})q^2(\boldsymbol{x}) \ dV \tag{3.57}$$

energy change $\Delta\mathbb{E}$ caused by the cell-based smoothing procedure can be calculated as

$$\begin{aligned}
\Delta\mathbb{E} &= \int_\Omega w(\boldsymbol{x}) \left(\hat{q}^2(\boldsymbol{x}) - q^2(\boldsymbol{x})\right)\ dV \\
&= \int_\Omega w(\boldsymbol{x}) \left[\left(\sum_{i=0}^{n} q_i\phi_i(\boldsymbol{x})\right)^2 - \left(\sum_{i=0}^{\infty} q_i\phi_i(\boldsymbol{x})\right)^2\right]\ dV \\
&= \int_\Omega w(\boldsymbol{x}) \left[\left(\sum_{i=0}^{n} q_i^2\phi_i^2(\boldsymbol{x})\right) - \left(\sum_{i=0}^{\infty} q_i^2\phi_i^2(\boldsymbol{x})\right)\right]\ dV \\
&= \int_\Omega w(\boldsymbol{x}) \left[-\sum_{i=n+1}^{\infty} q_i^2\phi_i^2(\boldsymbol{x})\right]\ dV \\
&= -\sum_{i=n+1}^{\infty} \left[q_i^2 \int_\Omega w(\boldsymbol{x})\phi_i^2(\boldsymbol{x})\ dV\right] \leq 0 \qquad (3.58)
\end{aligned}$$

which means the cell-based smoothing algorithm dissipates energy and hence offers a stabilization mechanism in the analysis.

In practice, it can be complicated to define approximation functions $\phi_i(\boldsymbol{x})$ for an arbitrary three-dimensional domain Ω. Therefore, if only degree one terms are considered, a simplified approximation of $q(\boldsymbol{x})$ can be written as

$$\begin{aligned}
\hat{q}(\boldsymbol{x}) &= \sum_{i=0}^{3} q_i\phi_i(\boldsymbol{x}) \qquad (3.59) \\
&= q_0 + q_1\,(x - x_o) + q_2\,(y - y_o) + q_3\,(z - z_o) \qquad (3.60)
\end{aligned}$$

where x_o, y_o and z_o are the coordinates for a weighted mean center and for any α in $\{x, y, z\}$ are obtained as

$$\alpha_o = \frac{\int_\Omega w(\boldsymbol{x})\alpha\ dV}{\int_\Omega w(\boldsymbol{x})\ dV} \qquad (3.61)$$

Then q_o can be calculated with Equation (3.55) and q_1 to q_3 are determined by solving

a three by three linear system. For any α, β in $\{x, y, z\}$ and $\alpha \neq \beta$

$$\int_\Omega w(\boldsymbol{x})\alpha\beta \; dV = \alpha_o\beta_o \int_\Omega w(\boldsymbol{x}) \; dV \tag{3.62}$$

which yields

$$\int_\Omega w(\boldsymbol{x}) \left(\alpha - \alpha_o\right)\left(\beta - \beta_o\right) \; dV = 0 \tag{3.63}$$

Here $(x - x_o)$, $(y - y_o)$ and $(z - z_o)$ are $\phi_1(\boldsymbol{x})$ to $\phi_3(\boldsymbol{x})$, respectively, and hence q_o to q_3 can all be determined with Equation (3.55) in this special case.

In the work of Mast et al. (2012), the (total) strain in most cases is smoothed with a constant function for particles in each cell, or a cell particle domain, Ω_c^P,

$$\hat{\boldsymbol{\varepsilon}}(\boldsymbol{x}) = \boldsymbol{\varepsilon}_0 = \frac{\int_{\Omega_c^P} \rho(\boldsymbol{x})\boldsymbol{\varepsilon}(\boldsymbol{x}) \; dV}{\int_{\Omega_c^P} \rho(\boldsymbol{x}) \; dV} = \frac{1}{m_c^P}\sum_p m_p\boldsymbol{\varepsilon}_p \tag{3.64}$$

where m_c^P is the total mass in a cell particle domain,

$$m_c^P = \sum_p m_p \tag{3.65}$$

If the strain can be separated into elastic and plastic parts:

$$\boldsymbol{\varepsilon}(\boldsymbol{x}) = \boldsymbol{\varepsilon}^{\boldsymbol{e}}(\boldsymbol{x}) + \boldsymbol{\varepsilon}^{\boldsymbol{p}}(\boldsymbol{x}) \tag{3.66}$$

and the stress states are the strain $\boldsymbol{\varepsilon}(\boldsymbol{x})$, the plastic strain $\boldsymbol{\varepsilon}^{\boldsymbol{p}}(\boldsymbol{x})$ and some other internal variables, the approach proposed by Mast et al. (2012) implies the smoothing procedure is applied on the elastic strain, i.e.

$$\hat{\boldsymbol{\varepsilon}^{\boldsymbol{e}}}(\boldsymbol{x}) = \hat{\boldsymbol{\varepsilon}}(\boldsymbol{x}) - \boldsymbol{\varepsilon}^{\boldsymbol{p}}(\boldsymbol{x}) \tag{3.67}$$

and

$$\Delta\boldsymbol{\varepsilon}^{\boldsymbol{e}}(\boldsymbol{x}) = \hat{\boldsymbol{\varepsilon}^{\boldsymbol{e}}}(\boldsymbol{x}) - \boldsymbol{\varepsilon}^{\boldsymbol{e}}(\boldsymbol{x}) = \hat{\boldsymbol{\varepsilon}}(\boldsymbol{x}) - \boldsymbol{\varepsilon}(\boldsymbol{x}) = -\sum_{i=1}^{\infty} \boldsymbol{\varepsilon}_i \phi_i(\boldsymbol{x}) \tag{3.68}$$

Consequently, the free (elastic strain) energy change, $\Delta\mathbb{E}$, caused by the smoothing can be evaluated numerically using

$$\Delta\mathbb{E} = \int_{\Omega} \left\{ \rho(\boldsymbol{x}) \Delta\boldsymbol{\varepsilon}^{\boldsymbol{e}}(\boldsymbol{x}) : \bar{\mathbb{C}}_s : [\boldsymbol{\varepsilon}^{\boldsymbol{e}}(\boldsymbol{x}) + \omega \Delta\boldsymbol{\varepsilon}(\boldsymbol{x}^e)] \right\} \, dV \tag{3.69}$$

in which $\bar{\mathbb{C}}_s$ is a specific secant modulus, ω is a real number between 0 to 1, and

$$\boldsymbol{\varepsilon}^{\boldsymbol{e}}(\boldsymbol{x}) = \underbrace{\sum_{i=0}^{\infty} \boldsymbol{\varepsilon}_i \phi_i(\boldsymbol{x})}_{\boldsymbol{\varepsilon}(\boldsymbol{x})} - \underbrace{\sum_{i=0}^{\infty} \boldsymbol{\varepsilon^p}_i \phi_i(\boldsymbol{x})}_{\boldsymbol{\varepsilon^p}(\boldsymbol{x})} \tag{3.70}$$

Substituting Equations (3.70) and (3.68) back into Equation (3.69) gives

$$\Delta\mathbb{E} = -(1-\omega) \sum_{i=1}^{\infty} \left[\boldsymbol{\varepsilon}_i : \bar{\mathbb{C}}_s : \boldsymbol{\varepsilon}_i \int_{\Omega} \rho(\boldsymbol{x}) \phi_i^2(\boldsymbol{x}) \, dV \right] \leq 0 \tag{3.71}$$

which can be expressed as Equation (3.58) with $n = 0$, $w(\boldsymbol{x}) = \rho(\boldsymbol{x})$ and

$$q_i = \sqrt{(1-\omega) \left(\boldsymbol{\varepsilon}_i : \bar{\mathbb{C}}_s : \boldsymbol{\varepsilon}_i \right)} \tag{3.72}$$

Equation (3.71) shows the cell-based smoothing algorithm contributes to system stabilization during the smoothing procedure because of strain energy dissipation. However, in some cases this is still not enough to balance the instability arising from integration errors. Therefore, numerical fluxes are introduced in this thesis to offer additional stabilization.

3.4 Numerical Flux Smoothing Algorithm

In the new proposed algorithm, smoothing is thought of as a conserved quantity transported within cells (control volumes). When this quantity is related to energy, the fundamental idea of the algorithm is to control energy increases to avoid instability, and at the same time control energy dissipation so variable fields will not be over-smoothed.

In the following subsections, to give a big picture of the new proposed algorithm, the procedure of the numerical flux smoothing algorithm is briefly explained. The background theory of the numerical flux is introduced in Section 3.4.2. A localization scheme, the essential technique for developing the presented algorithm is discussed in Section 3.4.3. Constant flux (Section 3.4.6), limited constant flux (Section 3.4.7) and linear flux (Section 3.4.8) smoothing algorithms are derived following the general form and stability condition described in Sections 3.4.4 and 3.4.5, respectively.

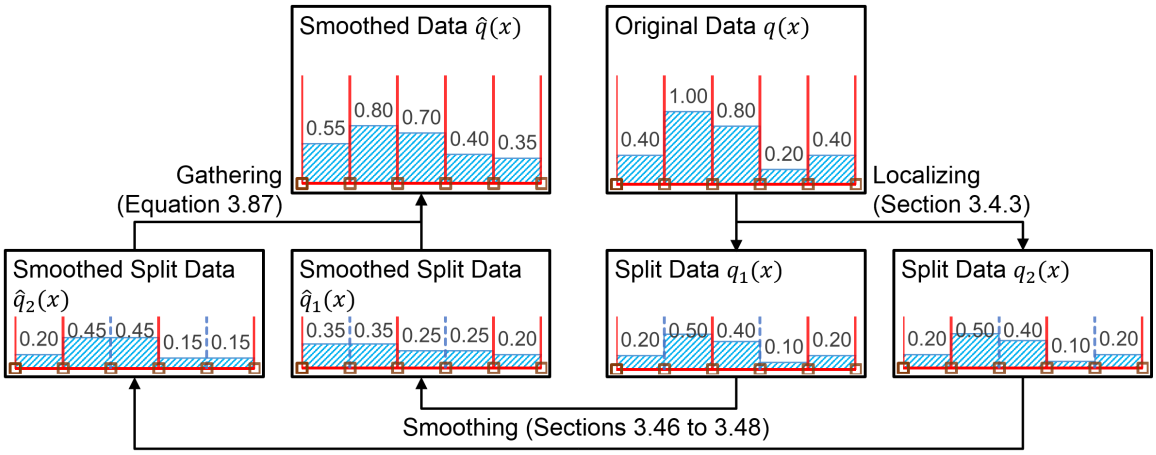


Figure 3.1: Flow diagram of the numerical flux smoothing algorithm

3.4.1 Preview of the Numerical Flux Smoothing Algorithm

A brief flow diagram of the numerical flux smoothing algorithm procedure is shown in Figure 3.1. A one-dimensional example with a piece-wise constant variable field (described with function $q(x)$) is also illustrated in this figure. The first (and also the critical) step in the algorithm is to separate a three-dimensional smoothing problem into multiple one-dimensional ones, each of which occurs between two adjacent cells, e.g. cells share interfaces highlighted with blue dash lines in the one-dimensional example shown in Figure 3.1. Next, one of the smoothing algorithms described in Sections 3.4.6 to 3.4.8 is used for the localized one-dimensional smoothing procedure. Constant flux smoothing algorithm (Section 3.4.6) is applied in the sample example. Finally, a smoothed field $\hat{q}(x)$ is evaluated by summing up the smoothed split functions, e.g. $\hat{q}_1(x)$ and $\hat{q}_2(x)$ in the example. More details about the procedure can be found in the following sections after an introduction of the background theory.

3.4.2 Discontinuous Galerkin Method and Numerical Flux

A discontinuous Galerkin method (DGM) is the background theory motivating the new smoothing algorithm. DGM can be considered a hybrid method between the finite element method (FEM) and the control volume method (CVM). DGM uses smooth shape functions within cells and allows discontinuity at cell boundaries or surfaces. In each cell, DGM, just like FEM, uses Galerkin weak form equations to evaluate rates of change of variables. However it introduces numerical traces or fluxes from CVM to solve interface conditions between cells. The numerical flux is the key for getting stable results in the method. In the following, weak form equations of a one-dimensional partial differential equation are derived based on the framework of DGM, not only as an example to illustrate the concept of the discontinuous Galerkin method, but also as a preliminary work for Section 3.4.4, in which a general form of the necessary equations for the new smoothing algorithm is presented. More details about

DGM and numerical fluxes can be found in Cockburn and Shu (2001), Hesthaven and Warburton (2008), and LeVeque (1992, 2002).

In the numerical flux smoothing algorithm, a one-dimensional wave equation is considered:

$$\frac{\partial}{\partial t}[\rho_x(x)\boldsymbol{q}(x,t)] + \frac{\partial}{\partial x}\boldsymbol{f}(\rho_x(x)\boldsymbol{q}(x,t)) = \mathbf{0} \tag{3.73}$$

where $\rho_x(x)$ is a linear mass density, $\boldsymbol{q}(x,t)$ is a variable vector function,

$$\boldsymbol{f} = \rho_x(x)\boldsymbol{\Lambda}_x \cdot \boldsymbol{q}(x,t) \tag{3.74}$$

is a flux vector, and $\boldsymbol{\Lambda}_x$ is designed as a constant diagonal second-order coefficient tensor. With the special designed $\boldsymbol{f}$, Equation (3.73) represents a series of independent transport equations:

$$\frac{\partial}{\partial t}[\rho_x(x)q_i(x,t)] + \lambda_i \frac{\partial}{\partial x}\left[\rho_x(x)q_i(x,t)\right] = 0 \tag{3.75}$$

in which $q_i(x,t)$ is a component of $\boldsymbol{q}(x,t)$; and λ_i not only is a diagonal component (an eigenvalue) of $\boldsymbol{\Lambda}_x$, but also means the transport speed in the equation.

The weak form equation of one cell corresponding to Equation (3.73) used in DGM is similar to the one used in FEM:

$$\int_{x_1^c}^{x_2^c} \boldsymbol{\psi} \cdot \rho_x \frac{\partial}{\partial t}\boldsymbol{q}\ dx = \int_{x_1^c}^{x_2^c} \frac{\partial \boldsymbol{\psi}}{\partial x} \cdot \boldsymbol{f}\ dx - [\boldsymbol{\psi} \cdot \boldsymbol{f}]_{x_1^c}^{x_2^c} \tag{3.76}$$

in which x_2^c and x_1^c are two x-directional boundaries of the cell such that $x_2^c > x_1^c$; and $\boldsymbol{\psi}(x)$ is a test vector function. However, unlike FEM (which assumes both $\boldsymbol{\psi}$ and $\boldsymbol{f}$ are globally continuous so that the boundary values shown at the right hand side of Equation (3.76) are canceled out at cell interfaces), DGM considers $\boldsymbol{\psi}$ and $\boldsymbol{f}$ are only continuous in cell level and explicitly allows discontinuities to occur at cell interfaces. To define a unique value at each cell interface, DGM introduces the numerical flux,

$\boldsymbol{f}^*$, and hence has the weak form,

$$\int_{x_1^c}^{x_2^c} \boldsymbol{\psi} \cdot \rho_x \frac{\partial}{\partial t} \boldsymbol{q} \, dx = \int_{x_1^c}^{x_2^c} \frac{\partial \boldsymbol{\psi}}{\partial x} \cdot \boldsymbol{f} \, dx - [\boldsymbol{\psi} \cdot \boldsymbol{f}^*]_{x_1^c}^{x_2^c} \tag{3.77}$$

For hyperbolic (wave) equations, $\boldsymbol{f}^*$ can be considered as a resultant flux at each cell interface. In the special case described in Equation (3.73), components of the numerical flux are fluxes of a series of independent transport equations (Equation (3.75)) at a cell interface, respectively, and can be evaluated with an up-winding scheme (Cockburn & Shu, 2001; Hesthaven & Warburton, 2008; LeVeque, 1992, 2002):

$$f_i^*(x,t) = \frac{\lambda_i}{2}\left[\rho_x^+(x) q_i^+(x,t) + \rho_x^-(x) q_i^-(x,t)\right] + \frac{|\lambda_i|}{2}\left[\rho_x^-(x) q_i^-(x,t) - \rho_x^+(x) q_i^+(x,t)\right] \tag{3.78}$$

Here superscripts $\pm$ indicate cells at positive and negative sides of x, respectively. It's worth noting here that the use of superscripts $\pm$ is different with general usage in the literature, e.g. Cockburn and Shu (2001), in which $\pm$ indicate limits approaching from positive and negative sides, respectively. Therefore, m^+ and m^- give the total mass of cells at two sides of a given interface.

Finally, let $\boldsymbol{q}(x,t)$ be a piece-wise linear function (the highest degree of a polynomial considered in the present work): in each cell c,

$$\boldsymbol{q}(x,t) = \boldsymbol{q}_0^c(t) + \boldsymbol{q}_1^c(t)(x - x_o^c) \tag{3.79}$$

and $\boldsymbol{\psi}(x,t)$ has the similar expression:

$$\boldsymbol{\psi}(x,t) = \boldsymbol{\psi}_0^c + (x - x_o^c)\boldsymbol{\psi}_1^c \tag{3.80}$$

in which $\boldsymbol{\psi}_0^c$ and $\boldsymbol{\psi}_1^c$ are two arbitrary and independent coefficient vectors. Substituting Equations (3.79) and (3.80) into (3.77) and using the arbitrariness of $\boldsymbol{\psi}_0^c$ and $\boldsymbol{\psi}_1^c$,

Equation (3.77) yields

$$m^c \frac{\partial}{\partial t} \boldsymbol{q}_0^c(t) = -\boldsymbol{f}^*(x_2^c, t) + \boldsymbol{f}^*(x_1^c, t) \tag{3.81}$$

and

$$I_{ox}^c \frac{\partial}{\partial t} \boldsymbol{q}_1^c(t) = m^c \boldsymbol{\Lambda}_x \cdot \boldsymbol{q}_0^c(t) - (x_2^c - x_o^c)\boldsymbol{f}^*(x_2^c, t) + (x_1^c - x_o^c)\boldsymbol{f}^*(x_1^c, t) \tag{3.82}$$

where m^c is the total mass in the domain Ω^c

$$m^c = \int_{x_1^c}^{x_2^c} \rho_x(x)\ dx = \int_{x_1^c}^{x_2^c} \left(\int_{\partial\Omega^c(x)} \rho(\boldsymbol{x})\ dS \right)\ dx = \int_{\Omega^c} \rho(\boldsymbol{x})\ dV \tag{3.83}$$

x_o^c is the x coordinate of center of mass such that $x_1^c < x_o^c < x_2^c$

$$x_o^c = \frac{1}{m^c} \int_{\Omega^c} \rho(\boldsymbol{x}) x\ dV \tag{3.84}$$

and I_{ox}^c is the x component of the moment of inertia

$$I_{ox}^c = \int_{x_1^c}^{x_2^c} \rho_x(x)(x - x_o^c)^2\ dx = \int_{\Omega^c} \rho(\boldsymbol{x})(x - x_o^c)^2\ dV \tag{3.85}$$

3.4.3 Localization Scheme

Localizing a smoothing procedure is an essential part of the proposed algorithm. It simplifies the smoothing procedure and maximizes the efficiency and flexibility of the algorithm. In the proposed algorithm, Cauchy's inequality is used to split one single multi-dimensional smoothing problem into multiple partitions, each of which represents an independent multi-dimensional smoothing problem; the concept of a discontinuous Galerkin method is used to localize the split smoothing problems within two cells as a series of one-dimensional cases.

Theory of Splitting

The purpose of splitting a smoothing problem into multiple partitions is to allow each cell to smooth with all its neighbors one at a time. To do so, the (minimum) number of partitions is equal to the (maximum) number of surfaces, edges and ends of a three-, two- and one-dimensional cell, respectively. For instance, as shown in Figure 3.1, two partitions are used in a one-dimensional problem because these cells have two ends. The following shows if each of the partitions has no energy increase after smoothing, a smoothed field has equal or less energy than the original one. Consequently, the smoothing algorithm is stable.

Consider a variable field $q(\boldsymbol{x})$, with a positive function $w(\boldsymbol{x})$ so that $\mathbb{E}$

$$\mathbb{E} = \int_{\Omega} w(\boldsymbol{x}) q^2(\boldsymbol{x}) \, dV \tag{3.86}$$

represents a certain energy. If $q(\boldsymbol{x})$ is uniformly split into n partitions,

$$q(\boldsymbol{x}) = \sum_{m=1}^{n} q_m(\boldsymbol{x}) \tag{3.87}$$

in which

$$q_m(\boldsymbol{x}) = \frac{q(\boldsymbol{x})}{n} \tag{3.88}$$

and $\hat{q}(\boldsymbol{x})$ is the union of smoothed variable fields $\hat{q}_m(\boldsymbol{x})$ of those partitions,

$$\hat{q}(\boldsymbol{x}) = \sum_{m=1}^{n} \hat{q}_m(\boldsymbol{x}), \tag{3.89}$$

then the energy $\hat{\mathbb{E}}$ related to the smoothed field $\hat{q}(\boldsymbol{x})$,

$$\hat{\mathbb{E}} = \int_{\Omega} w(\boldsymbol{x}) \hat{q}^2(\boldsymbol{x}) \, dV \tag{3.90}$$

has to be smaller or equal to $\mathbb{E}$ to ensure this smoothing algorithm is stable. Substi-

tuting Equation (3.87) into Equation (3.86), $\mathbb{E}$ can be related to the energy of each partition,

$$\mathbb{E} = n \sum_{m=1}^{n} \left(\int_{\Omega} w(\boldsymbol{x}) q_m^2(\boldsymbol{x}) \, dV \right) = n \sum_{m=1}^{n} \mathbb{E}_m \tag{3.91}$$

Moreover, according to Cauchy's inequality

$$\hat{q}^2(\boldsymbol{x}) = \left(\sum_{m=1}^{n} \hat{q}_m(\boldsymbol{x}) \right)^2 \leq n \sum_{m=1}^{n} \hat{q}_m^2(\boldsymbol{x}) \tag{3.92}$$

which yields

$$\hat{\mathbb{E}} \leq n \sum_{m=1}^{n} \hat{\mathbb{E}}_m \tag{3.93}$$

with

$$\hat{\mathbb{E}}_m = \int_{\Omega} w(\boldsymbol{x}) \hat{q}_m^2(\boldsymbol{x}) \, dV \tag{3.94}$$

Therefore, if no energy increases in any partition after smoothing, i.e.

$$\hat{\mathbb{E}}_m \leq \mathbb{E}_m \; \forall m \in \mathbb{N}^+, \; m \leq n \tag{3.95}$$

then

$$\hat{\mathbb{E}} \leq n \sum_{m=1}^{n} \hat{\mathbb{E}}_m \leq n \sum_{m=1}^{n} \mathbb{E}_m = \mathbb{E} \tag{3.96}$$

Localizing Technique

The techniques used to localize a smoothing procedure as a series of one-dimensional problems within two adjacent cells, respectively, is to manually block the numerical flux at most of the cell interfaces. In other words, interfaces between any two cells are only opened for the numerical flux when the two cells are designed to smooth to each other (in a split partition). For instance, the interfaces highlighted with blue dash lines shown in Figure 3.1 are the only ones where the numerical fluxes are allowed, that limits the smoothing to occur between two adjacent cells in the one-dimensional

sample example. Although the implementation derived in the following is based on structured grid geometry, the localization technique can be easily extended to non-structured grids (and hence the smoothing algorithm).

Let $q(\boldsymbol{x}, s)$ be a map which links an original field $q(\boldsymbol{x})$ to a smooqthed one $\hat{q}(\boldsymbol{x})$, with s representing pseudo-time in a smoothing procedure such that $q(\boldsymbol{x}, 0) = q(\boldsymbol{x})$ and $q(\boldsymbol{x}, 1) = \hat{q}(\boldsymbol{x})$. Consider $q(\boldsymbol{x})$ is a composition of cell-wise continuous variable fields of a three-dimensional structured grid

$$q(\boldsymbol{x}) = \bigoplus_{c_x=1}^{N_{cx}} \bigoplus_{c_y=1}^{N_{cy}} \bigoplus_{c_z=1}^{N_{cz}} q^{c_x c_y c_z}(\boldsymbol{x}) \tag{3.97}$$

where $\bigoplus$ is a direct sum operator; N_{cx}, N_{cy} and N_{cz} are numbers of cells in x, y and z directions, respectively; and

$$q^c(\boldsymbol{x}) = q_0^c + (x - x_o)q_1^c + (y - y_o)q_2^c + (z - z_o)q_3^c \tag{3.98}$$

in which $c \equiv c_x c_y c_z$ is the cell index. $q^c(\boldsymbol{x}, s)$ is expressed the same way as

$$q^c(\boldsymbol{x}, s) = q_0^c(s) + (x - x_o)q_1^c(s) + (y - y_o)q_2^c(s) + (z - z_o)q_3^c(s) \tag{3.99}$$

By blocking fluxes across cell interfaces in y and z directions and forcing

$$\frac{\partial}{\partial s}\left[\frac{\partial}{\partial y} q^{c_x c_y c_z}(\boldsymbol{x}, s)\right] = \frac{\partial}{\partial s} q_2^{c_x c_y c_z}(s) = 0 \tag{3.100}$$

and

$$\frac{\partial}{\partial s}\left[\frac{\partial}{\partial z} q^{c_x c_y c_z}(\boldsymbol{x}, s)\right] = \frac{\partial}{\partial s} q_3^{c_x c_y c_z}(s) = 0 \tag{3.101}$$

for all $\alpha \in \{x, y, z\}, c_\alpha \in \mathbb{N}^+, c_\alpha \leq N_{c\alpha}$, the problem can be reduced to one-dimensional cases. This step shreds (decomposes) a three-dimensional smoothing procedure into multiple one-dimensional cases, or shreds. If numerical fluxes are

blocked at interfaces between $c_x = 2n$ and $2n+1$ for any $n \in \mathbb{N}^+$ and $2n+1 \leq N_{cx}$ for each x-directional shred, then smoothing procedures can be localized within two cells, $c_x = 2n-1$ and $2n$. This is the case shown in the partition 1 of the one-dimensional sample example (Figure 3.1). Similarly, by blocking fluxes between $c_x = 2n-1$ and $2n$ for any $n \in \mathbb{N}^+$ and $2n \leq N_{cx}$, cells can separately smooth the other neighbors in the x direction (e.g. the partition 2 in Figure 3.1). By applying the described scheme in the y and z directions, respectively, smoothing can be applied between any cell and one of its six neighbors in a three-dimensional structured grid, if the neighbor exists.

3.4.4 General Form of Equations of the Numerical Flux Smoothing Algorithm

In this subsection, general form of equations used in the numerical flux algorithm are derived. Consider a structured grid, a variable field $q(\boldsymbol{x})$ can be expressed as a composition of cell-wise continuous variable fields as shown in Equation (3.97):

$$q(\boldsymbol{x}) = \bigoplus_{c_x=1}^{N_{cx}} \bigoplus_{c_y=1}^{N_{cy}} \bigoplus_{c_z=1}^{N_{cz}} q^{c_x c_y c_z}(\boldsymbol{x})$$

To localize the smoothing procedure with the technique introduced in Section 3.4.3, the field $q(\boldsymbol{x})$ is first split into six (i.e. the number of surfaces of a hexahedral cell used in this research) partitions:

$$q(\boldsymbol{x}) = \bigoplus_{c_x=1}^{N_{cx}} \bigoplus_{c_y=1}^{N_{cy}} \bigoplus_{c_z=1}^{N_{cz}} \sum_{m=1}^{6} \frac{1}{6} q^{c_x c_y c_z}(\boldsymbol{x}) \tag{3.102}$$

The smoothed field of cell $c \equiv c_x c_y c_z$ can then be expressed as

$$\hat{q}^c(\boldsymbol{x}) = q^c(\boldsymbol{x}) + \frac{1}{6} \sum_{\alpha=x}^{z} [\Delta q^c_{\alpha^-}(\boldsymbol{x}) + \Delta q^c_{\alpha^+}(\boldsymbol{x})] \tag{3.103}$$

in which subscripts $x^{\pm}$, $y^{\pm}$ and $z^{\pm}$ indicate at which side of the cell c is located in a local one-dimensional smoothing procedure. For instance, y^{-} indicates cell c adjoins the other cell in the negative y direction in a localized smoothing procedure. The increment values $\Delta q^c_{\alpha^{\pm}}$ for any $\alpha \in \{x, y, z\}$ caused by smoothing can be approximated with an explicit time integration scheme as:

$$\begin{aligned} \Delta q^c_{\alpha^{\pm}}(\boldsymbol{x}) &= \hat{q}^c_{\alpha^{\pm}}(\boldsymbol{x}) - q^c(\boldsymbol{x}) \\ &= q^c_{\alpha^{\pm}}(\boldsymbol{x}, 1) - q^c_{\alpha^{\pm}}(\boldsymbol{x}, 0) \\ &\approx \left[\frac{\partial}{\partial s} q^c_{\alpha^{\pm}}(\boldsymbol{x}, s)\right]_{s=0} \end{aligned} \tag{3.104}$$

Here, as defined in Section 3.4.3, $q^c_{\alpha^{\pm}}(\boldsymbol{x}, s)$ is a map which links an original field $q^c(\boldsymbol{x})$ to a smoothed one $\hat{q}^c_{\alpha^{\pm}}(\boldsymbol{x})$, with s representing pseudo-time in a smoothing procedure such that $q^c_{\alpha^{\pm}}(\boldsymbol{x}, 0) = q^c(\boldsymbol{x})$ and $q^c_{\alpha^{\pm}}(\boldsymbol{x}, 1) = \hat{q}^c_{\alpha^{\pm}}(\boldsymbol{x})$.

To determine the rate of change $\frac{\partial}{\partial s} q^c_{\alpha^{\pm}}(\boldsymbol{x}, s)$ for any cell c and $\alpha \in \{x, y, z\}$, first consider a x-directional shred (Section 3.4.3) and define

$$\boldsymbol{q}^c_x(\boldsymbol{x}, s) = \boldsymbol{a}^c_x q^c_x(\boldsymbol{x}, s) \tag{3.105}$$

as quantities flowing separately in three different velocities $c_{\leftarrow}$, c_* and $c_{\rightarrow}$. Here $c_{\leftarrow}$ indicates velocity of flow heading toward the negative x direction and hence $c_{\leftarrow} < 0$. In contrast, $c_{\rightarrow}$ indicates velocity of flow heading toward the positive x direction; thus $c_{\rightarrow} > 0$. $c_* = 0$ means the corresponding quantity is stationary. In other words, $q^c_x(\boldsymbol{x}, s)$ indicates a total quantity of $\boldsymbol{q}^c_x$ at any pseudo-time s and location $\boldsymbol{x}$ in a cell c, i.e.

$$q^c_x(\boldsymbol{x}, s) = \mathbf{1} \cdot \boldsymbol{q}^c_x(\boldsymbol{x}, s) = \mathbf{1} \cdot \boldsymbol{a}^c_x q^c_x(\boldsymbol{x}, s) \tag{3.106}$$

where $\mathbf{1}$ is a vector of all ones. Hence, $\boldsymbol{a}^c_x$ is a vector of non-negative constants that maps $q^c_x(\boldsymbol{x}, s)$ to $\boldsymbol{q}^c_x(\boldsymbol{x}, s)$ and

$$\mathbf{1} \cdot \boldsymbol{a}^c_x = 1 \tag{3.107}$$

Consider

$$q_x^c(\boldsymbol{x}, s) = q_0^c(s) + (x - x_o)q_1^c(s) + (y - y_o)q_2^c + (z - z_o)q_3^c \tag{3.108}$$

such that

$$\frac{\partial}{\partial s}q_x^c(\boldsymbol{x}, s) = \frac{\partial}{\partial s}q_0^c(s) + (x - x_o)\frac{\partial}{\partial s}q_1^c(s) \tag{3.109}$$

Then $\boldsymbol{q}_x^c(\boldsymbol{x}, s)$ defined in Equation (3.105) can be expressed as

$$\boldsymbol{q}_x^c(\boldsymbol{x}, s) = \boldsymbol{q}_0^c(s) + (x - x_o)\boldsymbol{q}_1^c(s) + \left[(y - y_o)q_2^c + (z - z_o)q_3^c\right]\boldsymbol{a}_x^c \tag{3.110}$$

with $\boldsymbol{q}_0^c(s) = \boldsymbol{a}_x^c q_0^c(s)$ and $\boldsymbol{q}_1^c(s) = \boldsymbol{a}_x^c q_1^c(s)$. Finally, the rate of change $\frac{\partial}{\partial s}q_x^c(\boldsymbol{x}, s)$ can be evaluated as

$$\begin{aligned}
\frac{\partial}{\partial s}q_x^c(\boldsymbol{x}, s) &= \mathbf{1} \cdot \frac{\partial}{\partial s}\boldsymbol{q}_x^c(\boldsymbol{x}, s) && (3.111)\\
&= \mathbf{1} \cdot \frac{\partial}{\partial s}\boldsymbol{q}_0^c(s) + (x - x_o)\mathbf{1} \cdot \frac{\partial}{\partial s}\boldsymbol{q}_1^c(s) && (3.112)
\end{aligned}$$

in which, using Equations (3.81) and (3.82),

$$\begin{aligned}
\mathbf{1} \cdot \frac{\partial}{\partial s}\boldsymbol{q}_0^c(s) &= \frac{1}{m^c}\mathbf{1} \cdot \left(m^c \frac{\partial}{\partial s}\boldsymbol{q}_0^c(s)\right)\\
&= \frac{1}{m^c}\mathbf{1} \cdot \left[-\boldsymbol{f}^*(x_2^c, s) + \boldsymbol{f}^*(x_1^c, s)\right] && (3.113)
\end{aligned}$$

$$\begin{aligned}
\mathbf{1} \cdot \frac{\partial}{\partial s}\boldsymbol{q}_1^c(s) &= \frac{1}{I_{ox}^c}\mathbf{1} \cdot \left(I_{ox}^c \frac{\partial}{\partial s}\boldsymbol{q}_1^c(s)\right)\\
&= \frac{1}{I_{ox}^c}\mathbf{1} \cdot \left[m^c \boldsymbol{\Lambda}_x \cdot \boldsymbol{a}_x^c q_0^c - (x_2^c - x_o^c)\boldsymbol{f}^*(x_2^c, s) + (x_1^c - x_o^c)\boldsymbol{f}^*(x_1^c, s)\right] && (3.114)
\end{aligned}$$

where components of the diagonal tensor $\boldsymbol{\Lambda}_x$ are equal to the flow velocities (i.e. $\lambda_1 = c_{\leftarrow}$, $\lambda_2 = 0$ and $\lambda_3 = c_{\rightarrow}$). Following the localization scheme (Section 3.4.3), if cell c is smoothed with a cell in the positive x direction, the numerical flux at other

surfaces of cell c are blocked, i.e. $\boldsymbol{f}^*(x_1^c, s) = \mathbf{0}$; if cell c is smoothed with a cell in the negative x direction, $\boldsymbol{f}^*(x_2^c, s) = \mathbf{0}$. Similar procedure can be applied to y and z directions. The rate of change $\frac{\partial}{\partial s} q^c_{\alpha^\pm}(\boldsymbol{x}, s)$ for any cell c and $\alpha \in \{x, y, z\}$ can be calculated as

$$\frac{\partial}{\partial s} q^c_{\alpha^+}(\boldsymbol{x}, s) = \frac{m^c q_0^c}{I^c_{o\alpha}} \mathbf{1} \cdot \mathbf{\Lambda}_{\alpha^+} \cdot \boldsymbol{a}^c_{\alpha^+} + \left[\frac{1}{m^c} + \frac{(\alpha_1^c - \alpha_o^c)}{I^c_{o\alpha}} (\alpha - \alpha_o^c) \right] \mathbf{1} \cdot \boldsymbol{f}^*(\alpha_1^c, s) \quad (3.115)$$

$$\frac{\partial}{\partial s} q^c_{\alpha^-}(\boldsymbol{x}, s) = \frac{m^c q_0^c}{I^c_{o\alpha}} \mathbf{1} \cdot \mathbf{\Lambda}_{\alpha^-} \cdot \boldsymbol{a}^c_{\alpha^-} - \left[\frac{1}{m^c} + \frac{(\alpha_2^c - \alpha_o^c)}{I^c_{o\alpha}} (\alpha - \alpha_o^c) \right] \mathbf{1} \cdot \boldsymbol{f}^*(\alpha_2^c, s) \quad (3.116)$$

Here, $\pm$ are added into the subscripts of $\mathbf{\Lambda}$ and $\boldsymbol{a}^c$ to indicate that $\mathbf{\Lambda}$ and $\boldsymbol{a}^c$ evaluated independently with adjacent cells in each localized smoothing procedure. The diagonal tensor $\mathbf{\Lambda}$ and the vector $\boldsymbol{a}^c$ (or more precisely the vector $\mathbf{\Lambda} \cdot \boldsymbol{a}^c$) are determined in Sections 3.4.6 to 3.4.8, respectively, along with the numerical flux $\boldsymbol{f}^*$ (or $\mathbf{1} \cdot \boldsymbol{f}^*$).

In the end, substituting Equations (3.115) and (3.116) back into (3.104), the smoothing-induced value change $\Delta q^c_{\alpha^\pm}$ for any $\alpha \in \{x, y, z\}$ can be evaluated as:

$$\Delta q^c_{\alpha^+}(\boldsymbol{x}) = \frac{m^c q_0^c}{I^c_{o\alpha}} \mathbf{1} \cdot \mathbf{\Lambda}_{\alpha^+} \cdot \boldsymbol{a}^c_{\alpha^+} + \left[\frac{1}{m^c} + \frac{(\alpha_1^c - \alpha_o^c)}{I^c_{o\alpha}} (\alpha - \alpha_o^c) \right] \mathbf{1} \cdot \boldsymbol{f}^*(\alpha_1^c, 0) \quad (3.117)$$

$$\Delta q^c_{\alpha^-}(\boldsymbol{x}) = \frac{m^c q_0^c}{I^c_{o\alpha}} \mathbf{1} \cdot \mathbf{\Lambda}_{\alpha^-} \cdot \boldsymbol{a}^c_{\alpha^-} - \left[\frac{1}{m^c} + \frac{(\alpha_2^c - \alpha_o^c)}{I^c_{o\alpha}} (\alpha - \alpha_o^c) \right] \mathbf{1} \cdot \boldsymbol{f}^*(\alpha_2^c, 0) \quad (3.118)$$

or with piece-wise constant interpolation Equations (3.117) and (3.118) can be simplified into

$$\Delta q^c_{\alpha^+}(\boldsymbol{x}) = \frac{1}{m^c} \mathbf{1} \cdot \boldsymbol{f}^*(\alpha_1^c, 0) \quad (3.119)$$

$$\Delta q^c_{\alpha^-}(\boldsymbol{x}) = \frac{-1}{m^c} \mathbf{1} \cdot \boldsymbol{f}^*(\alpha_2^c, 0) \quad (3.120)$$

3.4.5 Stability Condition

Before starting to implement the numerical flux smoothing algorithm in more detail, it is important to understand the stability condition first. In the proposed algorithm, the

smoothing procedure is applied locally between two adjacent cells as a one-dimensional problem. If each of these pairs has no energy increase, then the whole system is stable. Consider a one-dimensional case such that

$$q^{\pm}(x) = q_0^{\pm} + q_1^{\pm}(x - x_o^{\pm}) \tag{3.121}$$

are variable fields of two adjacent cells. Superscripts + and − indicate cells on positive and negative sides of the interface, respectively. Let $w(x) = \rho_x(x)$ be linear mass density, then energy of this unit system is

$$\mathbb{E} = m^- \left(q_0^-\right)^2 + I_{ox}^- \left(q_1^-\right)^2 + m^+ \left(q_0^+\right)^2 + I_{ox}^+ \left(q_1^+\right)^2 \tag{3.122}$$

If, after smoothing, the fields become

$$\hat{q}^{\pm}(x) = \left(q_0^{\pm} + \Delta q_0^{\pm}\right) + \left(q_1^{\pm} + \Delta q_1^{\pm}\right)(x - x_o^{\pm}) \tag{3.123}$$

the smoothing procedure is stable when energy change is less than or equal to zero,

$$\Delta\mathbb{E} = \Delta\mathbb{E}_0 + \Delta\mathbb{E}_1 \leq 0 \tag{3.124}$$

in which

$$\Delta\mathbb{E}_0 = m^- \Delta q_0^- \left(2q_0^- + \Delta q_0^-\right) + m^+ \Delta q_0^+ \left(2q_0^+ + \Delta q_0^+\right) \tag{3.125}$$

and

$$\Delta\mathbb{E}_1 = I_{ox}^- \Delta q_1^- \left(2q_1^- + \Delta q_1^-\right) + I_{ox}^+ \Delta q_0^+ \left(2q_1^+ + \Delta q_1^+\right) \tag{3.126}$$

3.4.6 Constant Flux Smoothing Algorithm

In this section, the generalized equations of the numerical flux algorithm described in Section 3.4.4 is implemented for piece-wise constant variable approximation specifi-

cally. Consider two adjacent cells, in which variable fields $q(x)$ are both constant;

$$q^{\pm}(x) = q_0^{\pm} \tag{3.127}$$

Using Equation (3.78)

$$\mathbf{1} \cdot \boldsymbol{f}^* = b^+ q_0^+ + b^- q_0^- \tag{3.128}$$

in which $b^+ = c_{\leftarrow} a_{\leftarrow}^+ \rho_x^+(x_m) < 0$, $b^- = c_{\rightarrow} a_{\rightarrow}^- \rho_x^-(x_m) > 0$ and x_m is the coordinate of the interface, then

$$\Delta q_0^+ = \frac{1}{m^+} \left(b^+ q_0^+ + b^- q_0^- \right) \tag{3.129}$$

and

$$\Delta q_0^- = \frac{-1}{m^-} \left(b^+ q_0^+ + b^- q_0^- \right) \tag{3.130}$$

or

$$m^+ \Delta q_0^+ = -m^- \Delta q_0^- \tag{3.131}$$

Then

$$\begin{aligned} \Delta \mathbb{E}(b^+, b^-) &= m^- \Delta q_0^- \left(2q_0^- + \Delta q_0^- \right) + m^+ \Delta q_0^+ \left(2q_0^+ + \Delta q_0^+ \right) && (3.132) \\ &= 2 \left(q_0^+ - q_0^- \right) \left(b^+ q_0^+ + b^- q_0^- \right) + \frac{m^- + m^+}{m^- m^+} \left(b^+ q_0^+ + b^- q_0^- \right)^2 && (3.133) \end{aligned}$$

To determine $b^{\pm}$ that give an extreme value of $\Delta \mathbb{E}(b^+, b^-)$, a set of equations has to be satisfied:

$$\begin{aligned} 0 &= \frac{\partial}{\partial b^+} \Delta \mathbb{E} && (3.134) \\ &= 2 q_0^+ \left[\left(q_0^+ - q_0^- \right) + \left(b^+ q_0^+ + b^- q_0^- \right) \frac{m^- + m^+}{m^- m^+} \right] && (3.135) \end{aligned}$$

and

$$0 = \frac{\partial}{\partial b^-}\Delta\mathbb{E} \tag{3.136}$$

$$= 2q_0^- \left[\left(q_0^+ - q_0^-\right) + \left(b^+ q_0^+ + b^- q_0^-\right) \frac{m^- + m^+}{m^- m^+} \right] \tag{3.137}$$

For arbitrary q_0^+ and q_0^-,

$$b^+ = -\frac{m^- m^+}{m^- + m^+} \qquad < 0 \tag{3.138}$$

and

$$b^- = \frac{m^- m^+}{m^- + m^+} \qquad > 0 \tag{3.139}$$

Substituting $b^\pm$ back into Equations (3.129), (3.130) and (3.132) yields to

$$\Delta q_0^+ = \frac{m^-}{m^- + m^+} \left(q_0^- - q_0^+\right) \tag{3.140}$$

$$\Delta q_0^- = \frac{m^+}{m^- + m^+} \left(q_0^+ - q_0^-\right) \tag{3.141}$$

and

$$\Delta\mathbb{E} = \frac{-m^- m^+}{m^- + m^+} \left(q_0^+ - q_0^-\right)^2 \leq 0 \tag{3.142}$$

3.4.7 Limited Constant Flux Smoothing Algorithm

Equation (3.142) shows the constant flux smoothing algorithm continues dissipating energy until the variable field goes down to a global constant function, i.e. $q_0^+ = q_0^-$ at any interfaces. This characteristic can easily ruin a simulation when a global constant function is not enough to describe the desired variable field. For instance, a pressure field in a hydrostatic problem. In this algorithm, the concept of a flux limiter, also called a slope limiter (LeVeque, 2002), is borrowed to tune down or turn off the fluxes as desired.

Tuning Coefficient of Constant Flux

Taking $b^{\pm}$ from Section 3.4.6 as

$$b^- = -b^+ = \gamma \frac{m^- m^+}{m^- + m^+} \tag{3.143}$$

in which γ is the tuning coefficient and $0 \leq \gamma \leq 1$, then

$$\Delta q_0^{\pm} = \pm \frac{m^{\mp}\gamma}{m^- + m^+} \left(q_0^- - q_0^+ \right) \tag{3.144}$$

and

$$\Delta \mathbb{E} = \gamma(2-\gamma) \frac{-m^- m^+}{m^- + m^+} \left(q_0^+ - q_0^- \right)^2 \leq 0 \tag{3.145}$$

$\gamma = 1$ gives the extreme energy dissipation; $\gamma = 0$ filters out all fluxes between cells and hence leads to a cell-based smoothing algorithm as described in Section 3.3.

Linear Limiter

The constant flux smoothing algorithm uses piece-wise constant interpolation in each cell, and hence introduces errors predicting variable values at cell edges. In the algorithm, linear terms of the variable fields for each cell are estimated using the piece-wise constant interpolation to determine allowable errors at cell boundaries. Consequently, the constant fluxes are adjusted based on the correction terms.

For each control volume c, and its neighbors l and r located at the negative and positive side of c, q_1^c can be estimated as

$$q_1^c = \begin{cases} \left(q_0^r - q_0^l\right) / \left(x_o^r - x_o^l\right) & \text{, if both } l \text{ and } r \text{ exist} \\ \left(q_0^c - q_0^l\right) / \left(x_o^c - x_o^l\right) & \text{, if only } l \text{ exists} \\ \left(q_0^r - q_0^c\right) / \left(x_o^r - x_o^c\right) & \text{, if only } r \text{ exists} \end{cases} \tag{3.146}$$

Then the allowable error e_a^c for element c at interface x_m is

$$e_a^c = q_1^c(x_m - x_o^c) \tag{3.147}$$

If x_m is approximated with

$$x_m = \frac{m^- x_o^+ + m^+ x_o^-}{m^- + m^+} \tag{3.148}$$

the tuning coefficient can be evaluated as

$$\gamma = \min\left\{ \max\left\{ \frac{\Delta q_0 + (e_a^- - e_a^+)}{\Delta q_0}, 0 \right\} 1 \right\} \tag{3.149}$$

where $\Delta q_0 = \left(q_0^- - q_0^+\right)$. In an implementation, to prevent $\Delta q_0 = 0$ and save computational cost $\Delta q_0^M = \gamma \Delta q_0$ can be calculated directly with

$$\Delta q_0^M = \text{sign}\left(\Delta q_0\right) \min\left\{ \max\left\{ \text{sign}\left(\Delta q_0\right) \left[\Delta q_0 + \left(e_a^- - e_a^+\right)\right], 0 \right\} \left|\Delta q_0\right| \right\} \tag{3.150}$$

Here sign($\bullet$) is a function returning -1, 0 and 1 when the input value is negative, zero and positive, respectively.

Quadratic Limiter

The quadratic limiter shares the idea of the linear limiter, but includes quadratic terms determining allowable errors at cell boundaries. For each control volume c and its neighbors l and r existing,

$$q_2^c = \frac{S^r - S^l}{\left(x_o^r + L_I^r\right) - \left(x_o^l + L_I^l\right)} \tag{3.151}$$

and

$$q_1^c = \frac{1}{2}\left(S^r + S^l\right) + q_2^c \left[\left(x_o^c - \frac{x_o^r + x_o^l}{2} \right) - \frac{1}{2}\left(L_I^r + L_I^l\right) \right] \tag{3.152}$$

where

$$S^r = \frac{q_0^r - q_0^c}{x_o^r - x_o^c} \tag{3.153}$$

$$S^l = \frac{q_0^c - q_0^l}{x_o^c - x_o^l} \tag{3.154}$$

$$L_I^r = \left(\frac{I_o^r}{m^r} - \frac{I_o^c}{m^c}\right)(x_o^r - x_o^c)^{-1} \tag{3.155}$$

$$L_I^l = \left(\frac{I_o^l}{m^l} - \frac{I_o^c}{m^c}\right)\left(x_o^l - x_o^c\right)^{-1} \tag{3.156}$$

If only r exists, by assuming $q_2^c = q_2^r$ Equations (3.151) and (3.152) give

$$q_1^c = S^r - q_2^c\left[x_o^r - x_o^c + L_I^r\right] \tag{3.157}$$

Similarly, if only l exists, $q_2^c = q_2^l$ and

$$q_1^c = S^l - q_2^c\left[x_o^l - x_o^c + L_I^l\right] \tag{3.158}$$

The allowable error e_a^c at a given interface x_m then can be evaluated with

$$e_a^c = q_1^c(x_m - x_o^c) + q_2^c\left[(x_m - x_o^c)^2 - \frac{I_o^c}{m^c}\right] \tag{3.159}$$

Finally, an adjusted constant flux can be calculated using Equation (3.150).

Oscillation Limiter

This limiter blocks all fluxes between cells unless local high-frequency oscillation occurs, which is treated as nonphysical or spurious behavior in the analysis. The condition indicating when a control volume is undergoing high-frequency oscillation is given by

$$\left(q_0^r - q_0^c\right)\left(q_0^c - q_0^l\right) \leq 0 \tag{3.160}$$

which means the slopes S^r and S^l defined in Equations (3.153) and (3.154) have different orientations. If and only if this condition is satisfied, the tuning coefficients γ for fluxes between c and its neighbors are set as 1; otherwise, $\gamma = 0$.

3.4.8 Linear Flux Smoothing Algorithm

In this section, piece-wise linear variable fields are considered for the implementation of Equations (3.117) and (3.118). Consider two adjacent cells, in which variable fields $q(x)$ are both linear;

$$q^{\pm}(x) = q_0^{\pm} + q_1^{\pm}(x - x_o^{\pm}) \tag{3.161}$$

Using Equation (3.78):

$$\mathbf{1} \cdot \boldsymbol{f}^* = f_m = b^+ \left(q_0^+ - q_1^+ L^+\right) + b^- \left(q_0^- + q_1^- L^-\right) \tag{3.162}$$

in which $L^{\pm}$ are the distance from $x_o^{\pm}$ to the cell interface; $b^+ = c_{\leftarrow} a_{\leftarrow}^+ \rho_{xe}^+ < 0$; $b^- = c_{\rightarrow} a_{\rightarrow}^- \rho_{xe}^- > 0$; and $\rho_{xe}^{\pm} = \rho_x^{\pm}(x_o^{\pm} \mp L^{\pm})$ are values of the mass density at the interface, then

$$m^+ \Delta q_0^+ = -m^- \Delta q_0^- = f_m \tag{3.163}$$

$$I_{ox}^+ \Delta q_1^+ = m^+ q_0^+ (c_{\leftarrow} a_{\leftarrow}^+ + c_{\rightarrow} a_{\rightarrow}^+) - L^+ f_m \tag{3.164}$$

and

$$I_{ox}^- \Delta q_1^- = m^- q_0^- (c_{\leftarrow} a_{\leftarrow}^- + c_{\rightarrow} a_{\rightarrow}^-) - L^- f_m \tag{3.165}$$

Let $c_{\leftarrow} a_{\leftarrow}^+ = -c_{\rightarrow} a_{\rightarrow}^+$ and $c_{\leftarrow} a_{\leftarrow}^- = -c_{\rightarrow} a_{\rightarrow}^-$, Equation (3.164) and Equation (3.165) can be simplified as

$$\frac{-I_{ox}^+ \Delta q_1^+}{L^+} = \frac{-I_{ox}^- \Delta q_1^-}{L^-} = f_m \tag{3.166}$$

Then the energy change caused by constant and linear modes can be respectively calculated as

$$\Delta\mathbb{E}_0 = f_m \left[2\left(q_0^+ - q_0^-\right) + f_m \frac{m^- + m^+}{m^- m^+} \right] \tag{3.167}$$

and

$$\Delta\mathbb{E}_1 = -L^+ f_m \left(2q_1^+ - f_m \frac{L^+}{I_{ox}^+} \right) - L^- f_m \left(2q_1^- - f_m \frac{L^-}{I_{ox}^-} \right) \tag{3.168}$$

After collecting terms, the total energy change is

$$\Delta\mathbb{E} = f_m \left\{ 2\left[\left(q_0^+ - L^+ q_1^+\right) - \left(q_0^- + L^- q_1^-\right) \right] + f_m \left(\frac{(L^+)^2}{I_{ox}^+} + \frac{(L^-)^2}{I_{ox}^-} + \frac{m^- + m^+}{m^- m^+} \right) \right\} \tag{3.169}$$

To determine $b^\pm$ that give extreme values of $\Delta\mathbb{E}(b^+, b^-)$, a set of equations, $\partial\Delta\mathbb{E}/\partial b^\pm = 0$, must be satisfied, both of that yields

$$\begin{aligned} &\left[\left(q_0^+ - L^+ q_1^+\right) - \left(q_0^- + L^- q_1^-\right) \right] + \\ &\quad \left[b^+ \left(q_0^+ - q_1^+ L^+\right) + b^- \left(q_0^- + q_1^- L^-\right) \right] \left(\frac{(L^+)^2}{I_{ox}^+} + \frac{(L^-)^2}{I_{ox}^-} + \frac{m^- + m^+}{m^- m^+} \right) = 0 \end{aligned} \tag{3.170}$$

For arbitrary $q_0^\pm$ and $q_1^\pm$,

$$b^- = -b^+ = \left(\frac{(L^+)^2}{I_{ox}^+} + \frac{(L^-)^2}{I_{ox}^-} + \frac{m^- + m^+}{m^- m^+} \right)^{-1} > 0 \tag{3.171}$$

However, $c_{\leftarrow} a_{\leftarrow}^- = -c_{\rightarrow} a_{\rightarrow}^- = -b^-/\rho_{xe}^-$, $c_{\rightarrow} a_{\rightarrow}^+ = -c_{\leftarrow} a_{\leftarrow}^+ = -b^+/\rho_{xe}^+$ and $a_{\leftarrow}^\pm + a_{\rightarrow}^\pm \leq 1$. To satisfy these conditions, let

$$c_{\rightarrow} = -c_{\leftarrow} = \left(\frac{(L^+)^2}{I_{ox}^+} + \frac{(L^-)^2}{I_{ox}^-} + \frac{m^- + m^+}{m^- m^+} \right)^{-1} \frac{1}{\min\left\{\rho_{xe}^+, \rho_{xe}^-\right\}} \tag{3.172}$$

and

$$a_{\leftarrow}^\pm = a_{\rightarrow}^\pm = \frac{\min\left\{\rho_{xe}^+, \rho_{xe}^-\right\}}{2\rho_{xe}^\pm} \tag{3.173}$$

Consequently,

$$b^- = -b^+ = \frac{1}{2}\left(\frac{(L^+)^2}{I_{ox}^+} + \frac{(L^-)^2}{I_{ox}^-} + \frac{m^- + m^+}{m^- m^+}\right)^{-1} \tag{3.174}$$

Substituting $b^{\pm}$ back to Equations (3.162) and (3.169) leads to

$$m^+ \Delta q_0^+ = -m^- \Delta q_0^- = \frac{-I_{ox}^+ \Delta q_1^+}{L^+} = \frac{-I_{ox}^- \Delta q_1^-}{L^-} = f_m \tag{3.175}$$

with

$$f_m = \frac{1}{2}\left(\frac{(L^+)^2}{I_{ox}^+} + \frac{(L^-)^2}{I_{ox}^-} + \frac{m^- + m^+}{m^- m^+}\right)^{-1} \left[\left(q_0^- + L^- q_1^-\right) - \left(q_0^+ - L^+ q_1^+\right)\right] \tag{3.176}$$

and

$$\Delta\mathbb{E} = \frac{-3}{4}\left(\frac{(L^+)^2}{I_{ox}^+} + \frac{(L^-)^2}{I_{ox}^-} + \frac{m^- + m^+}{m^- m^+}\right)^{-1} \left[\left(q_0^- + L^- q_1^-\right) - \left(q_0^+ - L^+ q_1^+\right)\right]^2 \leq 0 \tag{3.177}$$

If $L^{\pm}$ is selected as

$$L^{\pm} = \sqrt{\frac{I_{ox}^{\pm}}{m^{\pm}}} \tag{3.178}$$

then

$$f_m = \frac{1}{4}\frac{m^- m^+}{m^- + m^+}\left[\left(q_0^- + L^- q_1^-\right) - \left(q_0^+ - L^+ q_1^+\right)\right] \tag{3.179}$$

and

$$\Delta\mathbb{E} = \frac{-3}{8}\frac{m^- m^+}{m^- + m^+}\left[\left(q_0^- + L^- q_1^-\right) - \left(q_0^+ - L^+ q_1^+\right)\right]^2 \leq 0 \tag{3.180}$$

3.5 Characteristics of The Smoothing Algorithms

Before applying the smoothing algorithms in practice, it's worth having some simple tests to understand their characteristics first. With a given strain function $\varepsilon(x, s = 0)$ in a linear elastic bar, the tests are going to show how the strain field looks and how the strain energy is dissipated after 1, 10, 100, and 10000 smoothing iterations in a

single time step. To make the results more general and material free, a normalized energy index $I_{\mathbb{E}}$ is used instead of showing the linear strain energy directly:

$$\mathbb{E}(s) = C \int \varepsilon^2(x,s) dx \tag{3.181}$$

where C is a material modulus such that $\sigma(x,s) = C\varepsilon(x,s)$.

3.5.1 Normalized Energy Index

As shown in Section 3.3, the function $\varepsilon(x,s)$ can be expressed as

$$\varepsilon(x,s) = \varepsilon_0(s) + \varepsilon_1(s)\phi_1(x) + O(x^2) \tag{3.182}$$

The strain energy contributed by second and higher order terms can then be written as

$$\Delta\mathbb{E}_0(s) = \mathbb{E}(s) - \mathbb{E}_0(s) = C \int \left[\varepsilon(x,s) - \varepsilon_0(s)\right]^2 dx \tag{3.183}$$

Because all smoothing algorithms discussed here conserve quantities, $\varepsilon_0(s) = \varepsilon_0(0)$ is a constant and

$$\Delta\mathbb{E}_0(s) = C \int \left[\varepsilon(x,s) - \varepsilon_0(0)\right]^2 dx \tag{3.184}$$

Finally, a normalized energy index is defined as

$$I_{\mathbb{E}}(s) := \frac{\Delta\mathbb{E}_0(s) - \Delta\mathbb{E}_0(0)}{\Delta\mathbb{E}_0(0)} = \frac{\int \left[\varepsilon(x,s) - \varepsilon_0(0)\right]^2 dx}{\int \left[\varepsilon(x,0) - \varepsilon_0(0)\right]^2 dx} - 1 \tag{3.185}$$

for $\Delta\mathbb{E}_0(0) \neq 0$, i.e. $\varepsilon(x, s=0)$ is not constant in space. $-1 \leq I_{\mathbb{E}} \leq 0$ if the smoothing procedure is stable; $I_{\mathbb{E}} = 0$ if no energy is dissipated; and $I_{\mathbb{E}} = -1$ if only the constant term of the strain function is left after smoothing.

Table 3.1: Abbreviations used for the smoothing algorithms

Abbreviation	Smoothing Algorithm	Order of Continuity	Order of Cell Interpolant	Limiters
NB	node-based	C^0	linear	
C0	cell-based	C^{-1}	constant	
C1	cell-based	C^{-1}	linear	
F0	numerical flux	C^{-1}	constant	
F0L1	numerical flux	C^{-1}	constant	linear
F0L2	numerical flux	C^{-1}	constant	quadratic
F0A	numerical flux	C^{-1}	constant	oscillation
F0AL1	numerical flux	C^{-1}	constant	oscillation; linear
F0AL2	numerical flux	C^{-1}	constant	oscillation; quadratic
F1	numerical flux	C^{-1}	linear	

3.5.2 Results of The Smoothing Tests

Consider a bar with unit length ($L = 1$) uniformly discretized by ten cells ($L_c = 0.1$) and thirty particles ($L_p = 0.333$).

Figures (3.2)(a)-(j) display smoothed linear strain functions at different smoothing time steps, s, using the algorithms described in this chapter, and Figure 3.2(k) plots the normalized energy indices at the end of the tests. As shown in Table 3.1, the abbreviation "NB" indicates a node-based smoothing algorithm. "C" and "F" refer to cell-based and numerical flux smoothing algorithms, respectively. Numbers behind the first letters give the order of the corresponding algorithms. "A" indicates the use of the oscillation limiter for anti-oscillation. "L1" and "L2" imply that linear and quadratic limiters are used. Figure 3.2 shows that all the algorithms maintain the linear strain function to the end of the tests after the first smoothing, except the node-based Figure 3.2(a) and constant numerical flux Figure 3.2(d) algorithms, which converge to a constant function globally and hence have $I_{\mathbb{E}}(10^4) = -1$.

Figures 3.3 and 3.4 show similar results of smoothing tests for quadratic and cubic strain functions, respectively. Like in the linear function test, node-based and

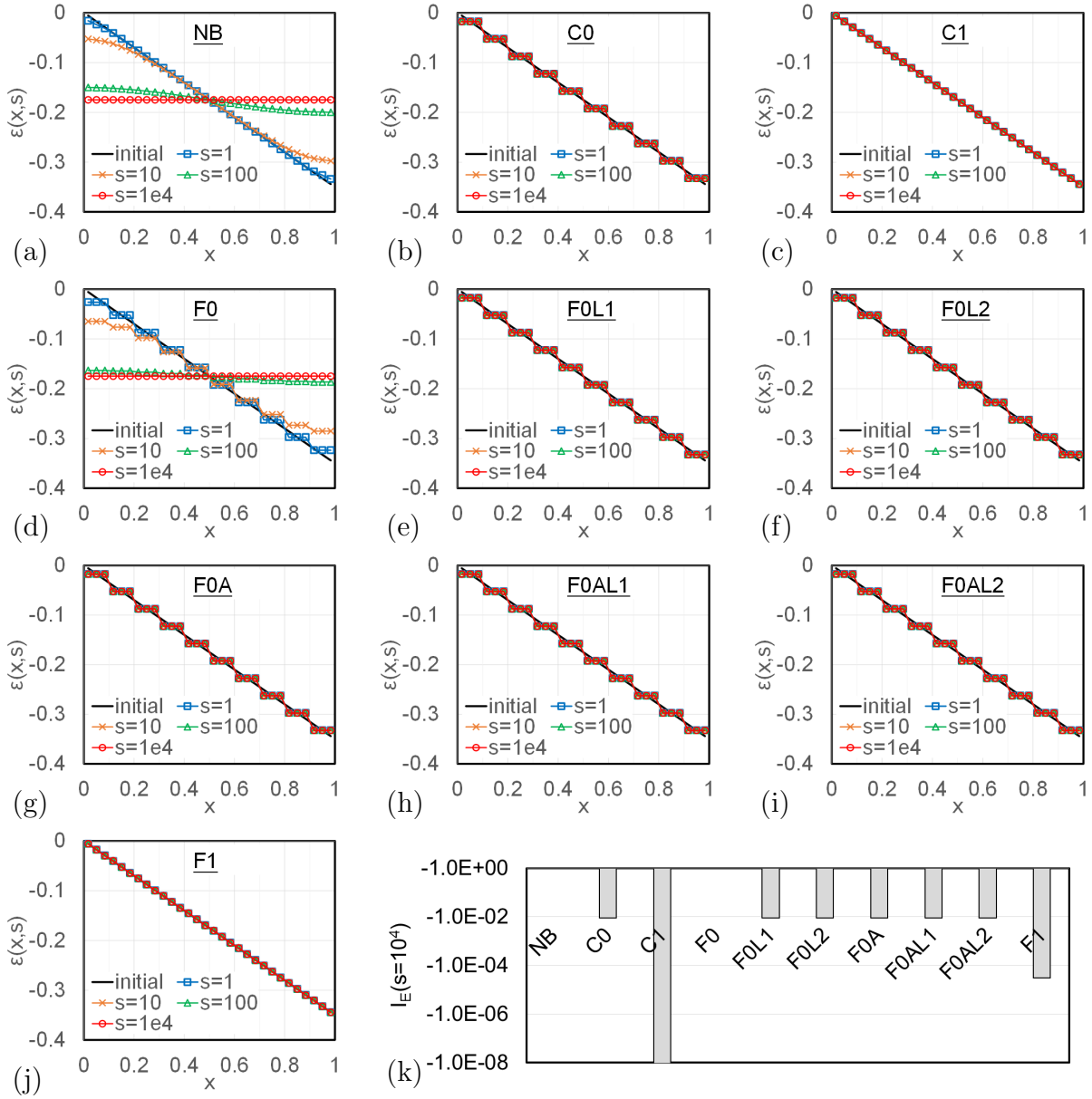


Figure 3.2: Smoothing test results for a linear strain function

constant flux algorithms cause the original functions to become global constant ones. In contrast, cell-based and linear flux algorithms conserve the shape of functions to the end of both tests after the first smoothing. Linear and quadratic limiters start to allow constant fluxes to flow across cell interfaces when the strain functions have

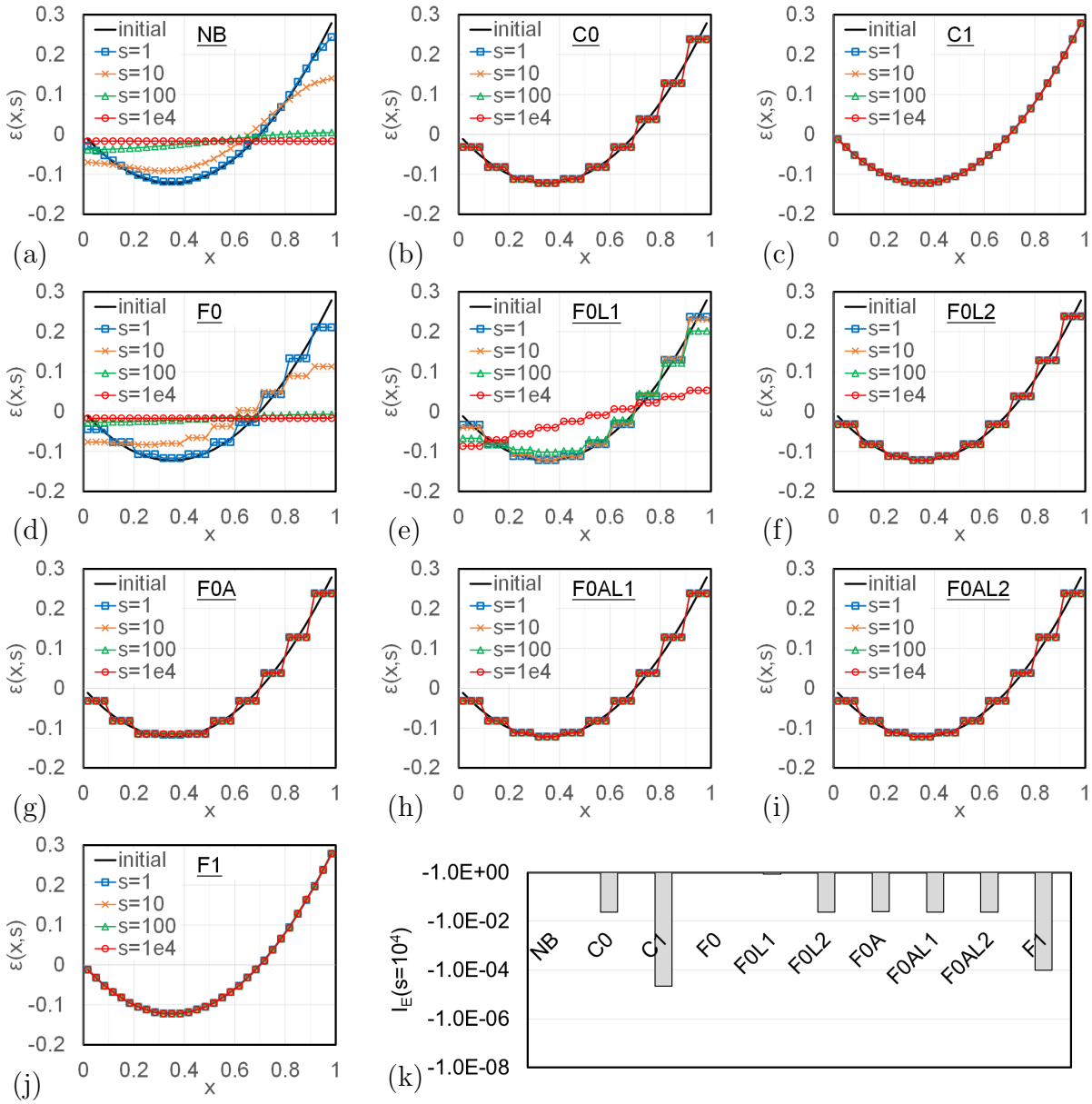


Figure 3.3: Smoothing test results for a quadratic strain function

higher orders than the limiters', respectively. However, when the limiters separately work with the oscillation limiter, the fluxes are well-controlled again. This implies that the special treatment of the edge cells for linear and quadratic limiters can only maintain functions up to the same orders, respectively. From another point of view,

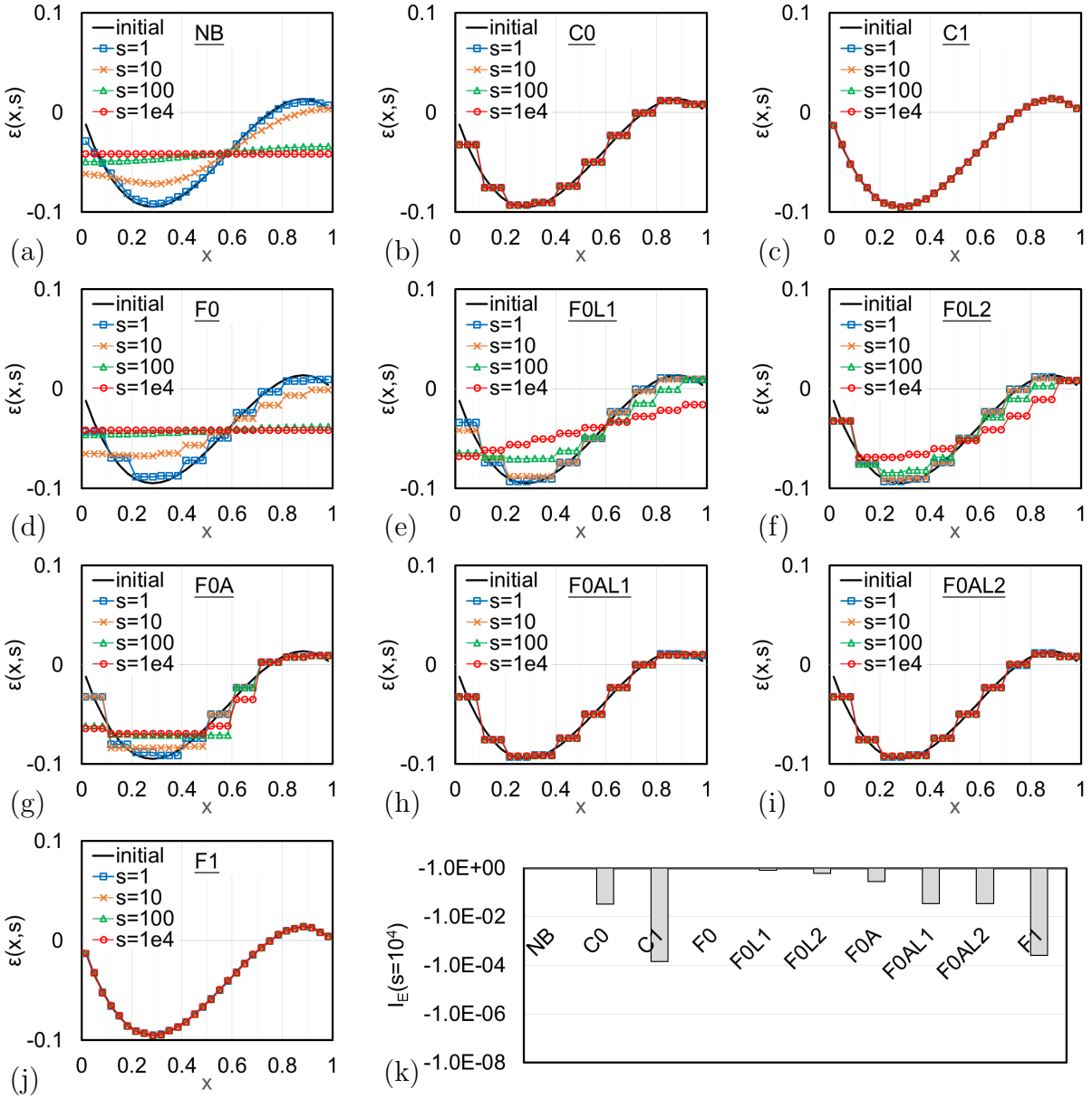


Figure 3.4: Smoothing test results for a cubic strain function

both linear and quadratic limiters enhance the performance of the oscillation limiter, which can introduce partial over-smoothing as observed in Figure 3.4(g).

Figures 3.5 illustrate how node-based and numerical flux smoothing algorithms control checkerboard instability, which can be a serious issue for cell-based algorithms.

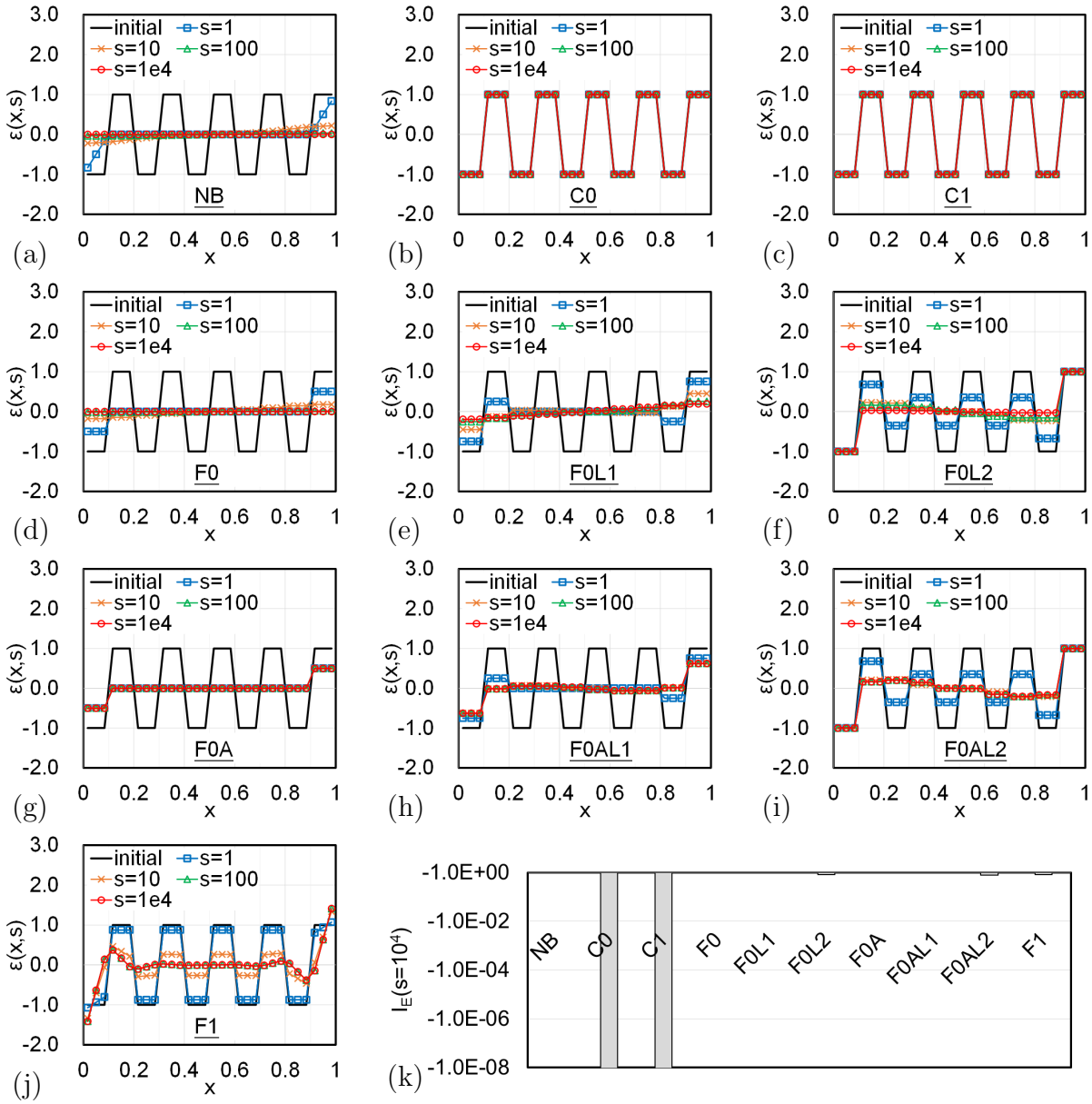


Figure 3.5: Smoothing test results for strain in checkerboard pattern

Figures 3.5(k) shows all the algorithms, other than the cell-based ones, can significantly dissipate out the energy of strain with a checkerboard pattern. However, like the node-based algorithm, the numerical flux algorithms smooth out the checkerboard pattern so fast that some additional stabilization, e.g. artificial viscosity, might be

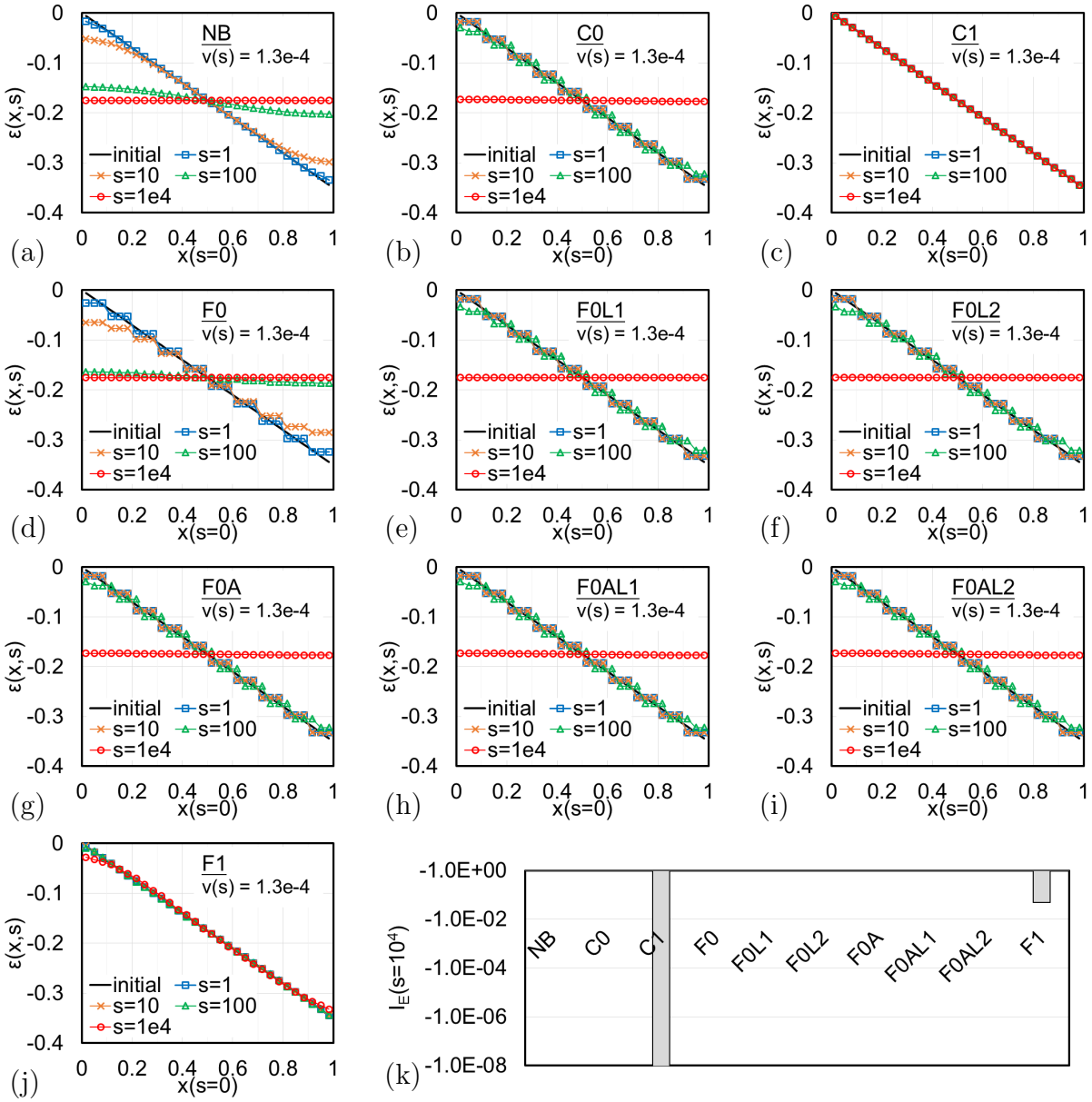


Figure 3.6: Smoothing test results for a linear strain function with relative velocity $v(s) = 1.3 \times 10^{-4}$

necessary.

Finally, Figures 3.6 indicate an additional energy dissipating mechanism using piece-wise constant interpolation. For this case a linear strain function is used like in

the first test (Figure 3.2), but now particles have velocity $v(s) = 1.3 \times 10^{-4}$ relative to the grids moving toward the left hand side. In this case, quantities (strain) physically transport within cells and hence the related energy is dissipated. However, in cases considered in this research particle motion is significantly slower than the acoustic speed in materials. Consequently, it takes many time steps for particles to move from one cell to another. Hence the energy dissipating mechanism caused by cell-crossing motion plays a minor role in system stabilization.

3.6 One-Dimensional Bar Examples

In this section one-dimensional bars which have a length of 1.6 m, 1 m^2 cross-sectional area, elastic modulus 1.2 MPa and mass density 3000 $^{kg}/_{m^3}$ are considered. Three particles per cell is used and cells with lengths L_c varying from 0.4 m to 0.025 m discretize the bars uniformly at $t = 0$ sec. 2% damping ratio (Section A.3) is added for stabilization.

3.6.1 Bars In Equilibrium

Figure 3.7 shows the final states ($t = 5$ sec) of the stress in the bars, which are fixed at $x = 0$ m with uniform distributed load -10 $^{N}/_{m}$ acting to the left and a corresponding initial linear stress field. As expected all the bars keep equilibrium to the end of the analysis except those smoothed with node-based (NB) and constant flux (F0) algorithms, respectively. Although the errors of stress around the fixed ends seem to decrease with mesh refinement, the low rate of convergence suggests that these two algorithms may have limitations in practical applications. On the other hand, if the strong smoothing characteristics are desired in particular applications, F0 can be an alternative algorithm for NB to reduce computational cost, since F0 only has one shape function per cell compared to the two, four, and eight shape functions in NB for one-, two- and three-dimensional elements, respectively.

In Figure 3.7(b), the particle around $x = 1.6$ m ($x/L_{bar} = 0.9$) undergoes a large

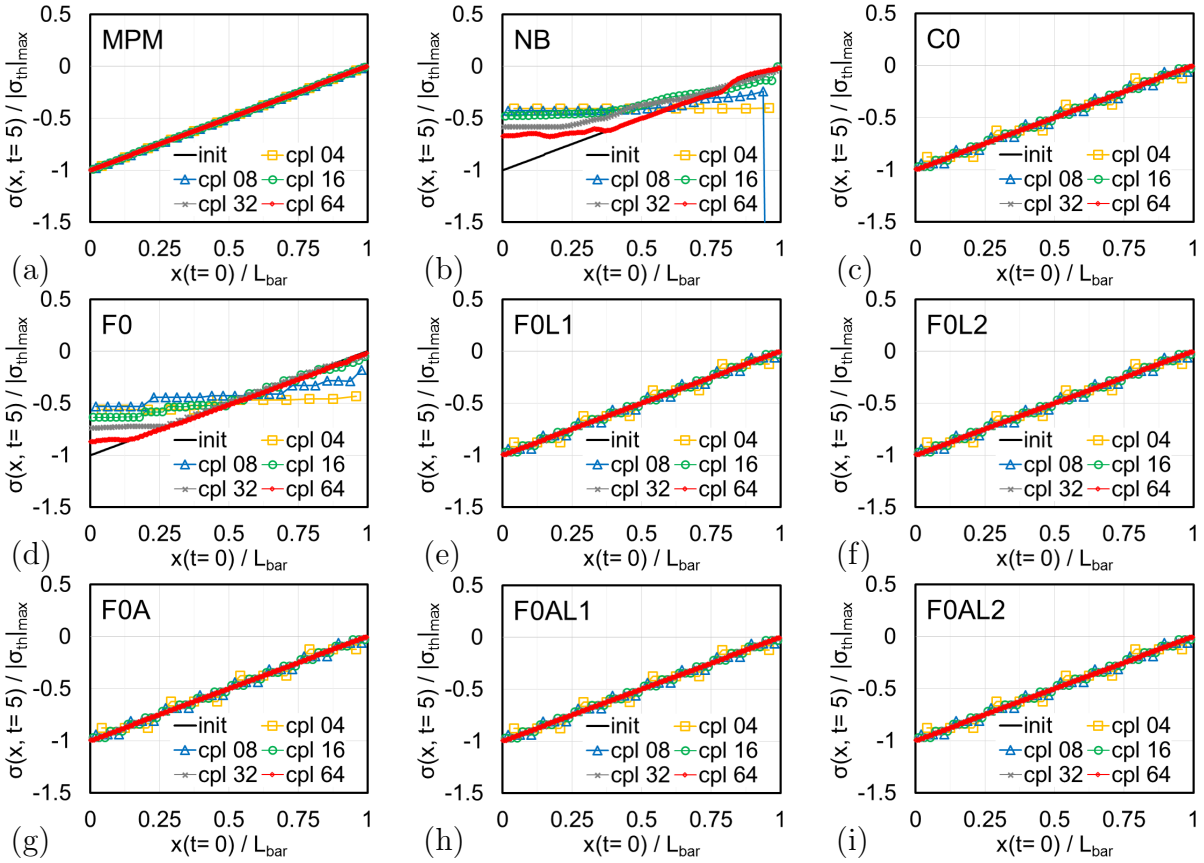


Figure 3.7: Stress at $t = 5$ sec of a fixed-end 1D bar in equilibrium under uniform distributed loads

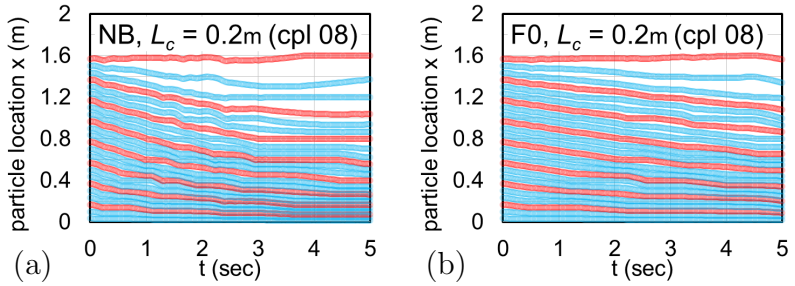


Figure 3.8: Particle trace of a fixed-end 1D bar in equilibrium under uniform distributed loads

compression force and the stress is out of the range of the plot. This is triggered by one of the major destabilization phenomena in MPM, often observed in particles of

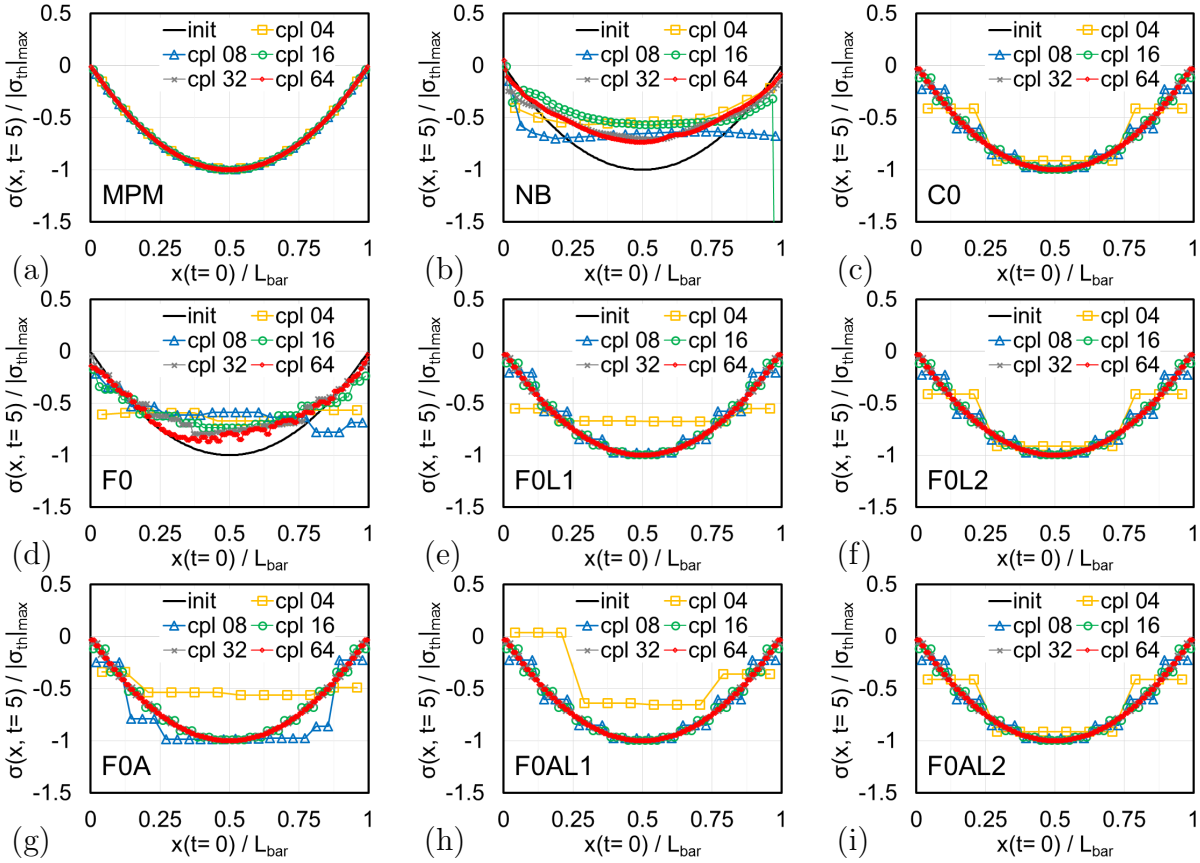


Figure 3.9: Stress at $t = 5\,\text{sec}$ of a 1D bar in equilibrium under linear distributed loads

edge cells (as the case shown in Figure 3.9(b)). Also, the linear interpolants used in the NB algorithm have the amplified stress untouched, compared to the constant interpolants used in the F0 algorithm which averages the stress of particles in the same cell. Therefore, even with similar particle traces (Figure 3.8), stress fields handled by the F0 algorithm are always smoother. Another interesting observation is that the strain energy dissipated during smoothing procedures can be implicitly transfered into kinetic energy and hence can be recovered. This phenomenon helps the F0L1 algorithm survive the next example, in which the stress field is a quadratic function.

Consider next a bar with a linear distributed load varying from $10\,\text{N/m}$ at the

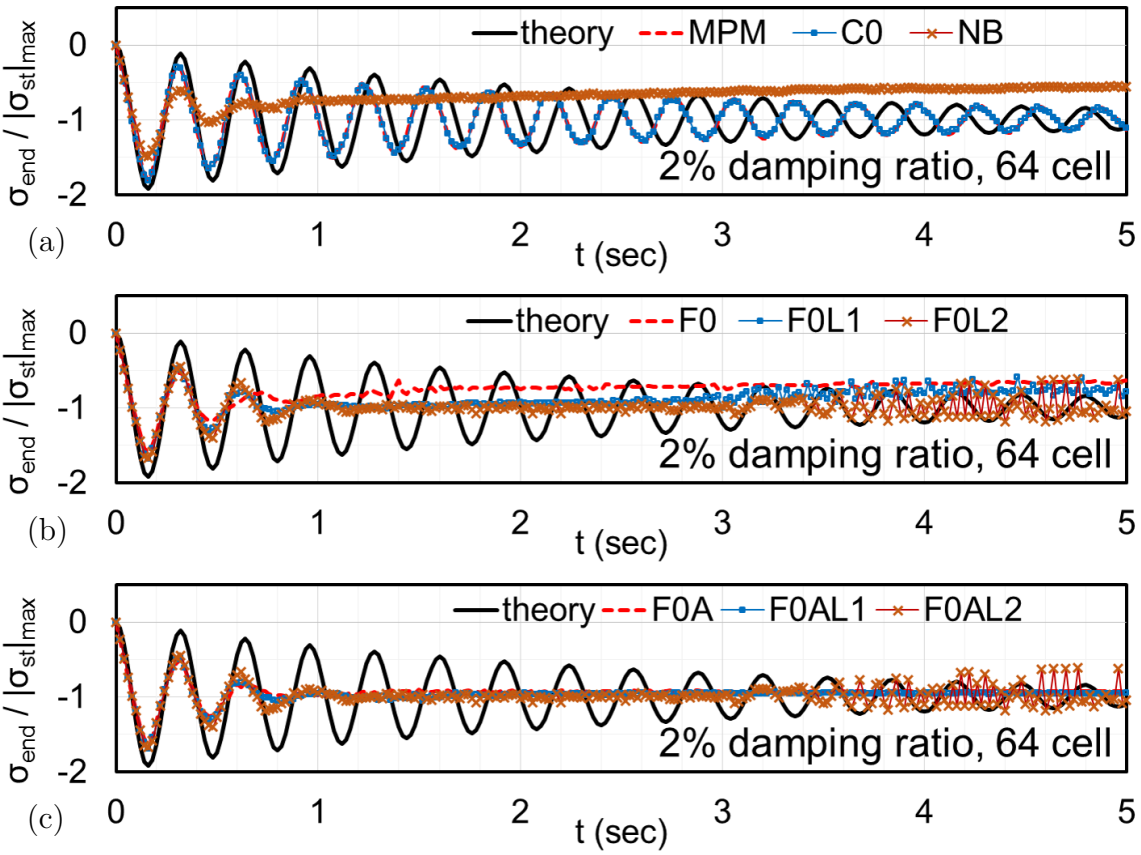


Figure 3.10: Time history of averaged particle stress at the fixed end of a vibrating bar under uniform distributed loads

left hand side to $-10\,^{\mathrm{N}}/_{\mathrm{m}}$ at the other end, in which an initial stress is assigned according to standard equilibrium equations. Figure 3.9 shows the final state of the bar under different smoothing algorithms. The NB and F0 algorithms can not maintain equilibrium, nor can F0A and F0A1 with coarse meshes. F0L1 in this case takes advantage of the recovery mechanism described at the end of the previous paragraph and hence maintains equilibrium at $t = 5\,\mathrm{sec}$. It appears there is an upper bound for the system to implicitly transfer the dissipated strain energy caused by smoothing into kinetic energy. More study will be necessary to explore this conjecture.

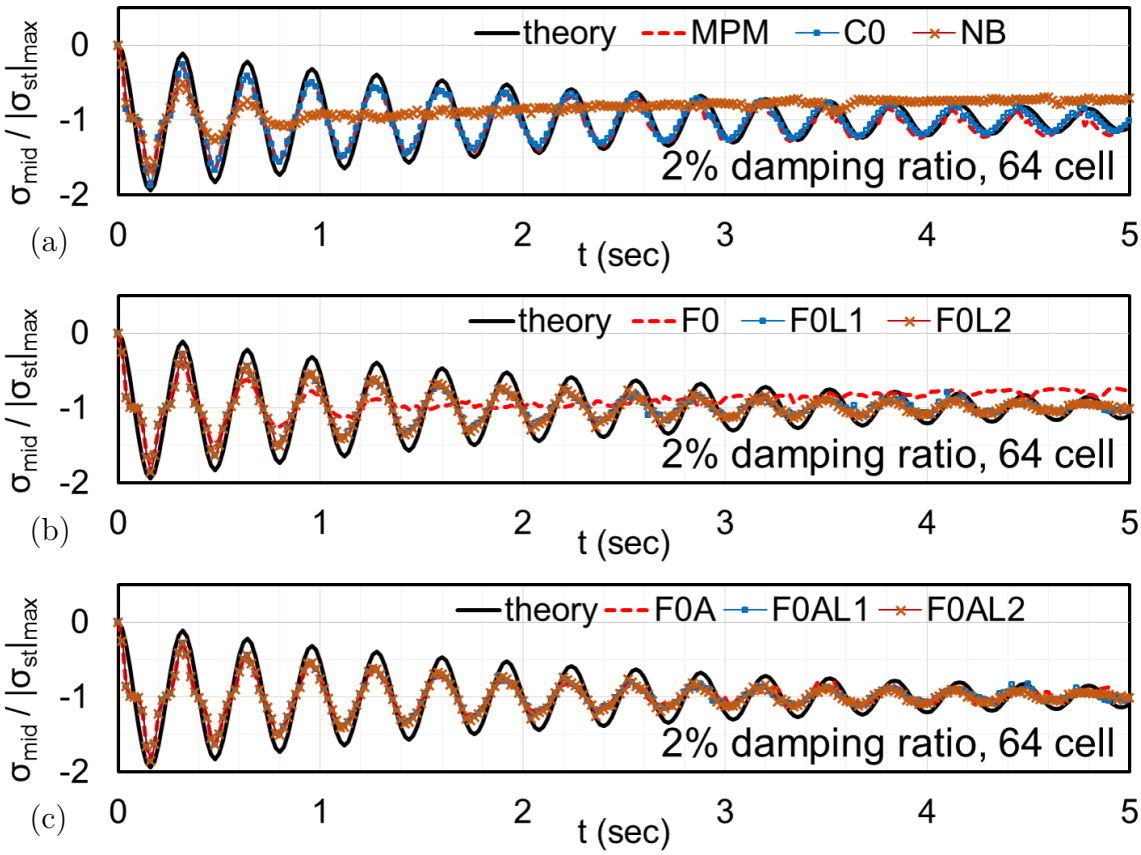


Figure 3.11: Time history of averaged particle stress at the middle of a vibrating bar under linearly varying loads

3.6.2 Bars In Vibration

In this section consideration is given to two dynamic one-dimensional problems which have the same loading (uniform and linearly varying) and boundary conditions as those described in previous section but zero initial stress.

Figures 3.10 and 3.11 show the time history of the average stress of particles initially in the cells ($L_c = 0.025\,\mathrm{m}$) near the fixed end and middle (symmetric) point of the bars in the two cases, respectively. Both Figures 3.10(a) and 3.11(a) indicate the averaged stress derived from MPM with or without the cell-based smoothing procedure are nearly identical to each other. This is an interesting observation, because,

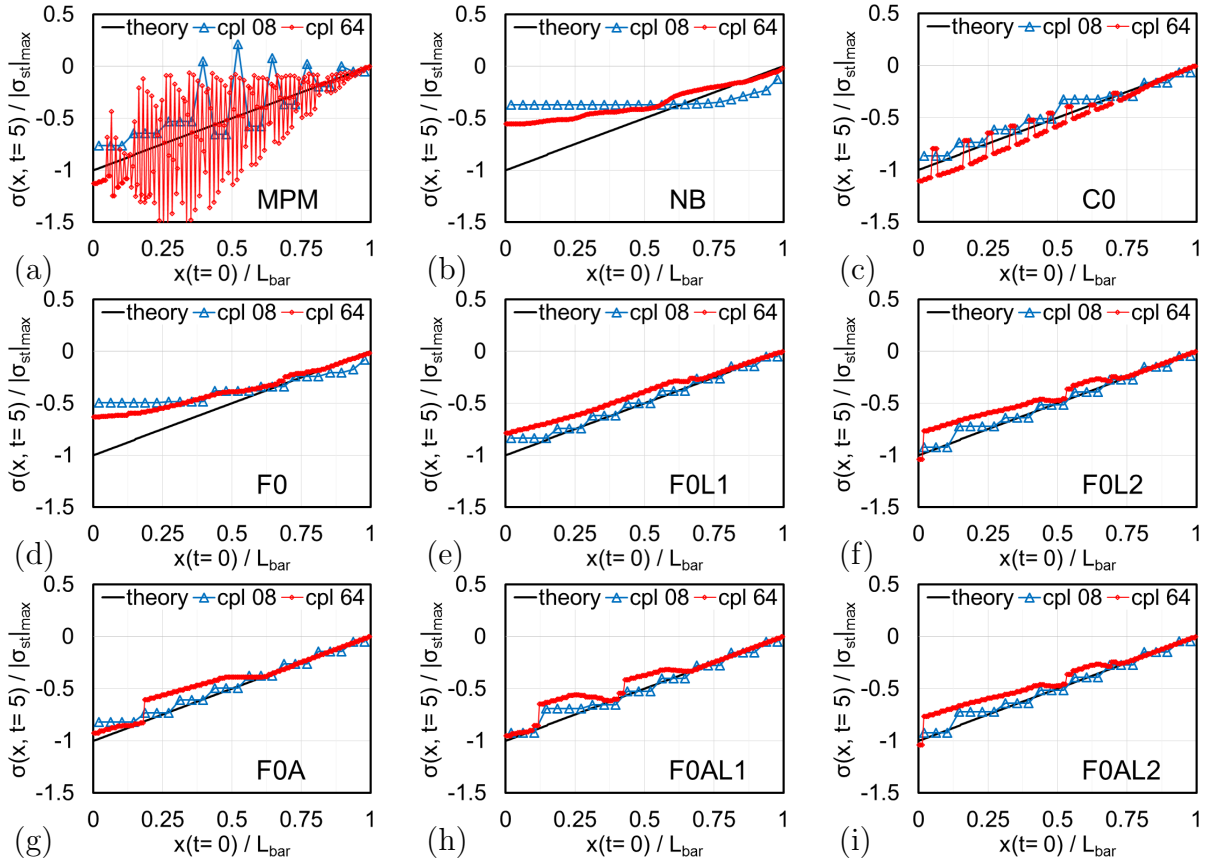


Figure 3.12: Stress at $t = 5$ sec of a vibrating fixed-end bar under uniform distributed loads

as shown in Figures 3.12 and 3.13, the stress fields at $t = 5$ sec are significantly more noisy when using the standard MPM. This implies the computational grid is able to filter out the noise and hence give reasonable nodal values and variable evolutions at each time step.

Continuing with Figures 3.10 to 3.13, the node-based and constant flux algorithms as expected dissipate energies quickly, stop the vibration in a few cycles, and continue smoothing out the stress fields. The limited constant flux algorithms all stop the vibration in a few cycles and include sudden changes in stress fields at $t = 5$ sec (Figures 3.12(e)-(i)) in the first case; but tend to converge to equilibrium after the vibration

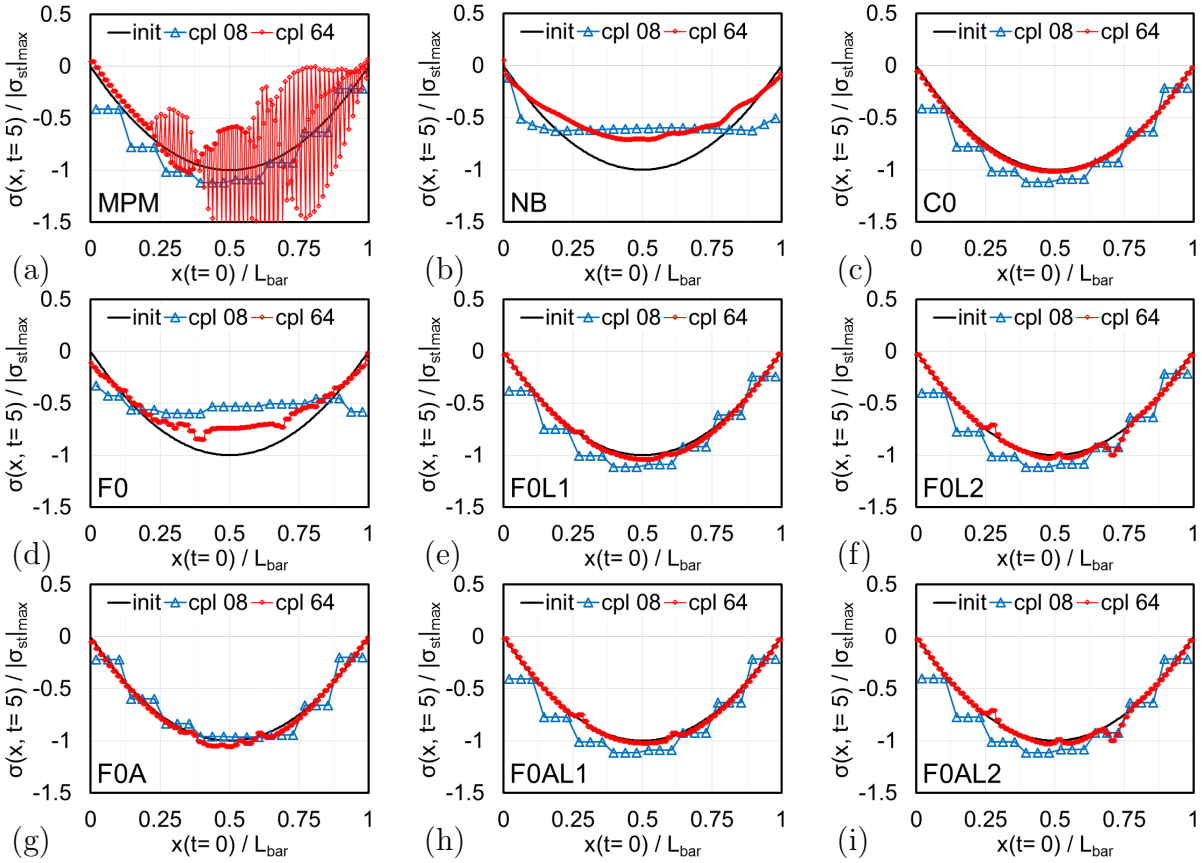


Figure 3.13: Stress at $t = 5\,\text{sec}$ of a vibrating bar under linearly varying loads

has dissipated out in the second case (Figures 3.13(e)-(i)). The nonphysical behavior observed in Figures 3.12(e)-(i) is mainly a derivation of unbalanced internal forces caused by different numbers of particles in the adjacent cells. This results in an integration error which is also the cause of the noisy stress field shown in Figure 3.12(c), and might be also the reason for the changing vibration period (Figure 3.10(a)). This problem needs a treatment for the integration errors and is not covered in this dissertation. In the present work, this noise is smoothed out directly with a controllable rate using the smoothing algorithms.

3.7 Summary and Findings from Chapter 3

In this chapter, anti-locking algorithms and smoothing procedures are first clarified and separated from the previous work (Mast et al., 2012). A numerical flux smoothing algorithm is then proposed to combine the beneficial features of both the cell-based and node-based smoothing algorithms. Consequently, the new proposed algorithm not only can solve hydrostatic problems as well as the cell-based approach, but also is as stable as the node-based algorithm. Based on the one-dimensional bar examples, the standard MPM gives reasonable nodal (averaged particle) values and hence is still appropriate for modeling solids (debris). The constant flux algorithm with both linear and oscillation limiters works the best among all others, and can handle higher order functions and converge to equilibrium after dissipating out the dynamic behavior, at least at boundaries (Figures 3.12(h) and 3.13(h)). Therefore, the F0AL1 algorithm is chosen for fluid modeling in this research. Some validation examples can be found in Chapter 5 after a new proposed boundary modeling algorithm introduced in the next chapter.

Chapter 4

ENHANCED BOUNDARY TREATMENT IN THE MATERIAL POINT METHOD

In the material point method, grid nodes link to particles contained in their own supported regions at the beginning of every time step. The cost of the linking procedure can be reduced to a minimum if a regular cuboid grid pattern is used. However this pattern restricts the flexibility of applying boundary conditions, because edges of the grid pattern may not appropriately represent the boundaries considered in many research applications. Therefore, a boundary modeling scheme which can decouple the grid geometry from the boundary geometry is developed in this research. Moreover, a strategy of modeling inlet and outlet boundaries for laminar flow in MPM is also discussed in this chapter. The flow boundaries can significantly simplify and reduce the scale of simulation models and hence save computing costs.

4.1 Boundary Conditions in MPM

4.1.1 Nodal Boundary Force

For the case of simple boundaries of a computational grid domain Ω^G on which either the traction or the velocity is prescribed (i.e. $\partial\Omega^G = \partial\Omega^{G\tau} + \partial\Omega^{Gv}$), the nodal acceleration $\boldsymbol{a}_i^{G(n)}$ for the n^{th} time step can be evaluated as

$$\boldsymbol{a}_i^{G(n)} = \frac{\boldsymbol{v}^*(\boldsymbol{x}_i, t_{n+1}) - \boldsymbol{v}_i^{G(n)}}{\Delta t} \tag{4.1}$$

if node $i \in \partial\Omega^{Gv}$, or for node $i \in \partial\Omega^{G\tau}$

$$\boldsymbol{a}_i^{G(n)} = \frac{1}{m_i^{G(n)}} \left(\boldsymbol{f}_i^{int(n)} + \boldsymbol{f}_i^{ext(n)} + \boldsymbol{f}_i^{bc(n)} \right) \tag{4.2}$$

in which

$$\boldsymbol{f}_i^{int} = -\int_{\Omega^G} \rho^P \text{grad}\, N_i^G \cdot \bar{\boldsymbol{\sigma}}^P \; dV \tag{4.3}$$

$$\boldsymbol{f}_i^{ext} = \int_{\Omega^G} \rho^P N_i^G \bar{\boldsymbol{b}}^P \; dV \tag{4.4}$$

$$\boldsymbol{f}_i^{bc} = \int_{\partial\Omega^{G\tau}} N_i^G \boldsymbol{\tau}^* \; dS \tag{4.5}$$

and $\boldsymbol{v}^*$ and $\boldsymbol{\tau}^*$ are the prescribed velocity and traction on boundaries $\partial\Omega^{Gv}$ and $\partial\Omega^{G\tau}$, respectively.

If $\boldsymbol{v}^*(\boldsymbol{x}_i, t_{n+1})$ is viewed as a result of some prescribed boundary traction, then the nodal boundary force $\boldsymbol{f}_i^{bc(n)}$ for the velocity boundary condition can be calculated by substituting Equation (4.1) into Equation (4.2):

$$\boldsymbol{f}_i^{bc(n)} = -\boldsymbol{f}_i^{int(n)} - \boldsymbol{f}_i^{ext(n)} + \frac{1}{\Delta t} \Delta \boldsymbol{p}_i^{(n)} \tag{4.6}$$

with $\Delta \boldsymbol{p}_i^{(n)}$ being the momentum change of node i during the n^{th} time step:

$$\Delta \boldsymbol{p}_i^{(n)} = m_i^{G(n)} \boldsymbol{v}^*(\boldsymbol{x}_i, t_{n+1}) - \boldsymbol{p}_i^{G(n)} \tag{4.7}$$

in which $\boldsymbol{p}_i^G$ is the lumped nodal momentum defined in Equation (2.3). Therefore, Equation (4.2) can be treated as a general form for both velocity and traction boundary conditions. In the following subsections, nodal boundary forces $\boldsymbol{f}_i^{bc(n)}$ for some often used boundary conditions are introduced.

4.1.2 Fixed Boundary Condition

If node i is constrained (i.e. $\boldsymbol{v}^* = \mathbf{0}$), the boundary force can be calculated based on Equation (4.6):

$$\boldsymbol{f}_i^{bc(n)} = -\boldsymbol{f}_i^{int(n)} - \boldsymbol{f}_i^{ext(n)} - \frac{1}{\Delta t}\boldsymbol{p}_i^{G(n)} \tag{4.8}$$

4.1.3 Velocity Boundary Condition

If node i is only constrained in a prescribed direction $\boldsymbol{n}_i$ and free in any other directions, the boundary force is

$$\boldsymbol{f}_i^{bc(n)} = \left(-\boldsymbol{f}_i^{int(n)} - \boldsymbol{f}_i^{ext(n)}\right) \cdot \boldsymbol{N}_i + \frac{1}{\Delta t}\Delta\boldsymbol{p}_i^{(n)} \tag{4.9}$$

where

$$\Delta\boldsymbol{p}_i^{(n)} = \left(m_i^{G(n)} v^*(\boldsymbol{x}_i, t_{n+1}) - \boldsymbol{p}_i^{G(n)} \cdot \boldsymbol{n}_i\right) \boldsymbol{n}_i \tag{4.10}$$

$\boldsymbol{N}_i := \boldsymbol{n}_i \otimes \boldsymbol{n}_i$ and v^* is a prescribed velocity magnitude in the direction of $\boldsymbol{n}_i$.

4.1.4 Slip Wall Boundary Condition

For a surface in space with normal $\boldsymbol{n}_i$ at node i, the boundary force, $\boldsymbol{f}_i^{bc(n)}$, then is the normal force,

$$\boldsymbol{f}_i^{bc(n)} = \boldsymbol{f}_{ni}^{(n)} = \left[-\boldsymbol{f}_i^{int(n)} - \boldsymbol{f}_i^{ext(n)} - \frac{1}{\Delta t}\boldsymbol{p}_i^{G(n)}\right] \cdot \boldsymbol{N}_i \tag{4.11}$$

if node i tends to bump against the surface;

$$\left[\left(\boldsymbol{f}_i^{int(n)} + \boldsymbol{f}_i^{ext(n)}\right)\Delta t + \boldsymbol{p}_i^{G(n)}\right] \cdot \boldsymbol{n}_i \leq 0 \tag{4.12}$$

otherwise $\boldsymbol{f}_i^{bc(n)} = \mathbf{0}$.

4.1.5 Wall Boundary Condition

If dry (Coulomb) friction is considered on the surface, the nodal boundary force includes both normal and shear forces

$$\boldsymbol{f}_i^{bc(n)} = \boldsymbol{f}_{ni}^{(n)} + \boldsymbol{f}_{fi}^{(n)} \tag{4.13}$$

when the node is moving toward the surface (Equation (4.12)). In this case the direction of the friction force is parallel to $\boldsymbol{s}_i^{(n)}$:

$$\boldsymbol{s}_i^{(n)} = \begin{cases} \frac{\boldsymbol{p}_{ti}^{(n)}}{\|\boldsymbol{p}_{ti}^{(n)}\|} & \text{, if } \|\boldsymbol{p}_{ti}^{(n)}\| \neq 0 \\ \frac{\boldsymbol{f}_{ti}^{(n)}}{\|\boldsymbol{f}_{ti}^{(n)}\|} & \text{, if } \|\boldsymbol{p}_{ti}^{(n)}\| = 0 \text{ and } \|\boldsymbol{f}_{ti}^{(n)}\| \neq 0 \\ \mathbf{0} & \text{, otherwise} \end{cases} \tag{4.14}$$

in which

$$\boldsymbol{p}_{ti}^{(n)} = \boldsymbol{p}_i^{G(n)} - \boldsymbol{p}_i^{G(n)} \cdot \boldsymbol{N}_i \tag{4.15}$$

and

$$\boldsymbol{f}_{ti}^{(n)} = \left(\boldsymbol{f}_i^{int(n)} + \boldsymbol{f}_i^{ext(n)}\right) - \left(\boldsymbol{f}_i^{int(n)} + \boldsymbol{f}_i^{ext(n)}\right) \cdot \boldsymbol{N}_i \tag{4.16}$$

and the friction force is bounded by the product between the coefficient of friction μ_f and the magnitude of normal force,

$$\boldsymbol{f}_{fi}^{(n)} = \text{sign}\left(f_{si}^{(n)}\right) \min\left\{\mu_f \|\boldsymbol{f}_{ni}^{(n)}\|,\ |f_{si}^{(n)}|\right\} \boldsymbol{s}_i^{(n)} \tag{4.17}$$

with

$$f_{si}^{(n)} = \left(\boldsymbol{f}_i^{int(n)} + \boldsymbol{f}_i^{ext(n)} + \frac{1}{\Delta t}\boldsymbol{p}_i^{G(n)}\right) \cdot \boldsymbol{s}_i^{(n)} \tag{4.18}$$

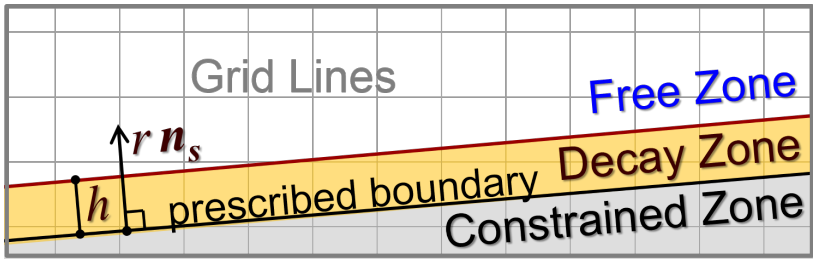


Figure 4.1: Sketch of the influence region of a boundary.

4.2 Boundary Condition for Laminar Flow

In the present work, a velocity boundary condition is selected for a region in which flow is laminar and has the same velocity along each stream line. For this boundary condition, the constrained direction is defined based on flow, and the velocity magnitude is read from upstream velocities. For an inflow boundary condition, upstream velocity is an input value; for an outflow boundary condition, upstream velocities are read from adjacent cell nodes. If node j is the adjacent node at the upstream side of a boundary node i, then

$$v^* = \max\left\{\left(\boldsymbol{v}_j \cdot \boldsymbol{n}_i\right), 0\right\} \tag{4.19}$$

Here v^* has to be a nonnegative value to ensure the flow is going in a correct direction. Moreover, the scheme defining velocities of the outflow boundary condition also implies the grid lines are parallel to the stream lines.

4.3 Arbitrary Boundaries for Regular Grids

If the nodal boundary force $\boldsymbol{f}_i^{bc}$ is thought of as a result of a specific force field $\bar{\boldsymbol{f}}(\boldsymbol{x})$ acting on the lumped mass m_i^G of node i on $\partial\Omega^G$:

$$\boldsymbol{f}_i^{bc} = m_i^G \bar{\boldsymbol{f}}(\boldsymbol{x}_i) \tag{4.20}$$

grid geometries and cell shapes can be independent of constrained (boundary) conditions. Hence a regular cuboid grid pattern can freely be used to minimize the particle linking procedures in MPM. In this study, the specific force field is defined as

$$\bar{\boldsymbol{f}}(\boldsymbol{x}) := \gamma_{SF}(\boldsymbol{x})\bar{\boldsymbol{f}}^{B}(\boldsymbol{x}) \tag{4.21}$$

in which

$$\bar{\boldsymbol{f}}^{B}(\boldsymbol{x}_i) = \frac{\boldsymbol{f}_i^{bc}}{m_i^G} \tag{4.22}$$

and $\gamma_{SF}(\boldsymbol{x})$ is an influence factor of the specific force $\bar{\boldsymbol{f}}^{B}$:

$$\gamma_{SF}(\boldsymbol{x}) = \begin{cases} 1 & \text{, if } \boldsymbol{x} \text{ is in a constrained zone } \Omega^C \\ \gamma_d(r/h) & \text{, if } \boldsymbol{x} \text{ is in a decay zone } \Omega^D \\ 0 & \text{, otherwise } (\boldsymbol{x} \text{ is in a free zone } \Omega^F) \end{cases} \tag{4.23}$$

Here γ_d is a decay coefficient, which is a function of the thickness h of the decay layer (zone), and the distance r between $\boldsymbol{x}$ and the surface of the constrained zone, such that $\gamma_d(0) = 1$, $\gamma_d(1) = 0$ and $0 \leq \gamma_d(r/h) \leq 1$. With Equations (4.21) to (4.23), Equation (4.2) can be rewritten as

$$\boldsymbol{a}_i^{G(n)} = \frac{1}{m_i^{G(n)}}\left(\boldsymbol{f}_i^{int(n)} + \boldsymbol{f}_i^{ext(n)} + \gamma_{SF}(\boldsymbol{x}_i)\boldsymbol{f}_i^{bc(n)}\right) \tag{4.24}$$

Figure 4.1 illustrates the specific force field. The boundary in the standard MPM described in Section 4.1 then is a special case such that $\partial\Omega^G = \Omega^C \cap \Omega^D$ and the thickness of the decay layer is smaller or equal to the dimension of the cells.

4.4 Decay Coefficient

To use Equation (4.23), the thickness of the layer h and the path of the coefficient γ_d have to be defined first. In this research, the layer thickness h is selected based on

the dimension of cuboid cells l_x, l_y and l_z and the surface normal $\boldsymbol{n_s}$ to $\partial\Omega^C$:

$$h = \sqrt{(l_x^2, l_y^2, l_z^2) \cdot \boldsymbol{n_s}} \tag{4.25}$$

And the function γ_d is

$$\gamma_d(r/h) = \left(1 - \frac{r}{h}\right)^2 \tag{4.26}$$

which has the best performance in the test described next.

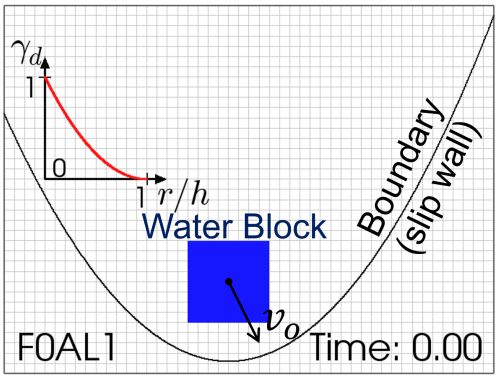


Figure 4.2: Initial setup for testing decay coefficient $\gamma_d(r/h)$

Consider a 0.4 m by 0.4 m water block shown in Figure 4.2 with bulk modulus 2.2 GPa, mass density 1000 $^{kg}/_{m^3}$ and viscosity 1 mPa · s. The block has initial velocity $v_o = (0.4\,^{m}/_{s}, -0.8\,^{m}/_{s})$ and gravity load $-10\,^{m}/_{s^2}$ pointing downward. The cell size is 0.05 m by 0.05 m and hence $h = 0.05$ m. 3 by 3 particles per cell are used for the water block. The boundary in this case is considered as a slip wall defined by a quadratic curve. Water is modeled with the F0AL1 algorithm described in Chapter 3. Figures 4.3 to 4.8 display snapshots of test results using $\gamma_d = 0$, $(1 - {}^{r}/_{h})^2$, $0.5[1 - \cos({}^{\pi r}/_{h})]$, $1 - {}^{r}/_{h}$, $1 - ({}^{r}/_{h})^2$ and 1, respectively. All the tested functions prevent particles from crossing the abstract boundary surface, except $\gamma_d = 0$. Based on Figures 4.3 to 4.8, $\gamma_d = (1 - {}^{r}/_{h})^2$ allows particles to go closest to the boundary without penetration and hence is used in this research.

4.5 Summary and Findings from Chapter 4

In this chapter, resultant nodal accelerations caused by prescribed velocity were first interpreted as equivalent nodal boundary forces. The nodal forces corresponding to often used boundary conditions were also introduced. A modeling scheme of inlet and outlet boundary conditions for laminar flow was then discussed. In the end, a new scheme for handling arbitrary boundary geometrics with regular grid patterns was proposed. The basic idea of this algorithm is to introduce a specific force field, in which a decay zone (layer) connects a constrained zone with a free zone, where nodes are boundary free. In the decay zone, nodal boundary forces are reduced with a decay coefficient to reflect decreasing influence. These coefficients are computed from a decay function with distance between nodes and the boundary surface as input values. Results show $\gamma_d = (1 - r/h)^2$ allows particles to go closest to the expected boundary surface without penetrating, compared to other five functions: $\gamma_d = 0$, $0.5[1 - \cos(\pi r/h)]$, $1 - r/h$, $1 - (r/h)^2$ and 1. More study will be necessary to understand and optimize the performance of the new proposed boundary modeling algorithm.

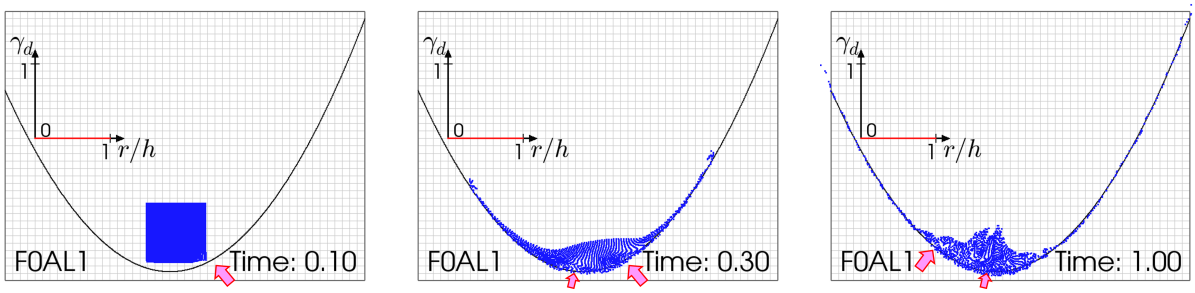


Figure 4.3: Observation focuses (pointed by red arrows) on the results of decay coefficient test with $\gamma_d = 0$

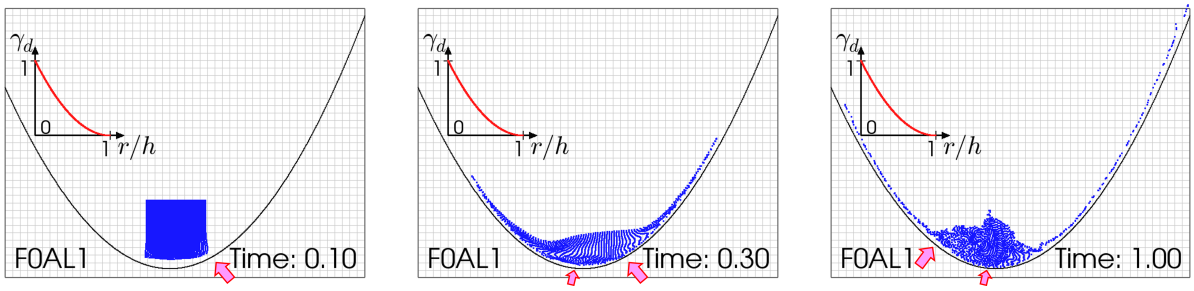


Figure 4.4: Observation focuses (pointed by red arrows) on the results of decay coefficient test with $\gamma_d = (1 - r/h)^2$

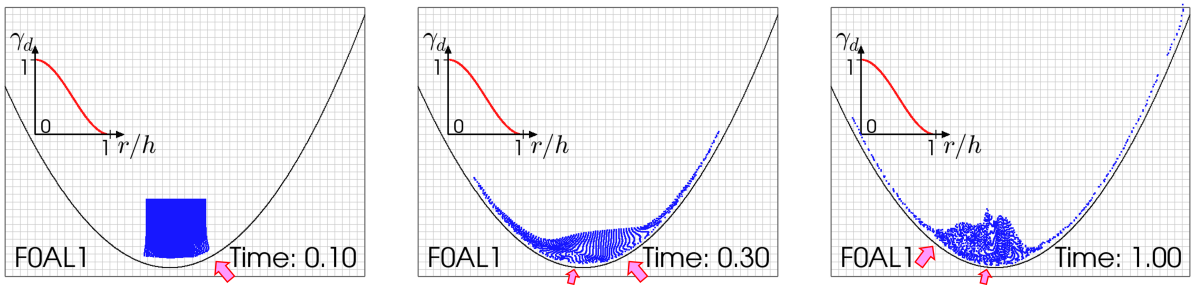


Figure 4.5: Observation focuses (pointed by red arrows) on the results of decay coefficient test with $\gamma_d = 0.5(1 - \cos(\pi r/h))$

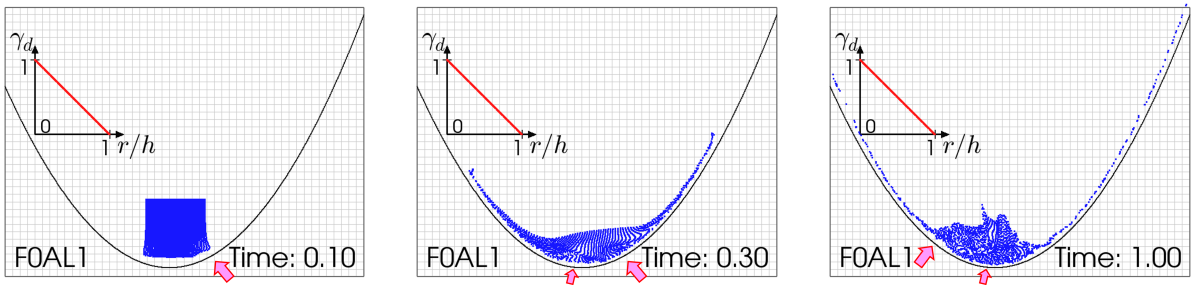


Figure 4.6: Observation focuses (pointed by red arrows) on the results of decay coefficient test with $\gamma_d = 1 - (r/h)$

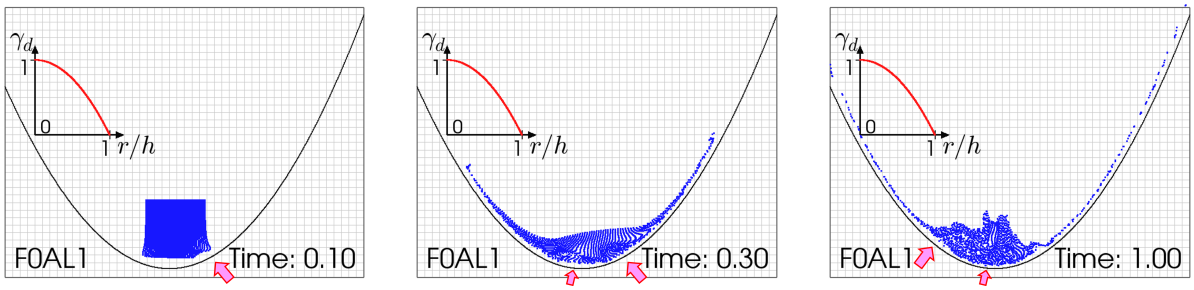


Figure 4.7: Observation focuses (pointed by red arrows) on the results of decay coefficient test with $\gamma_d = 1 - (r/h)^2$

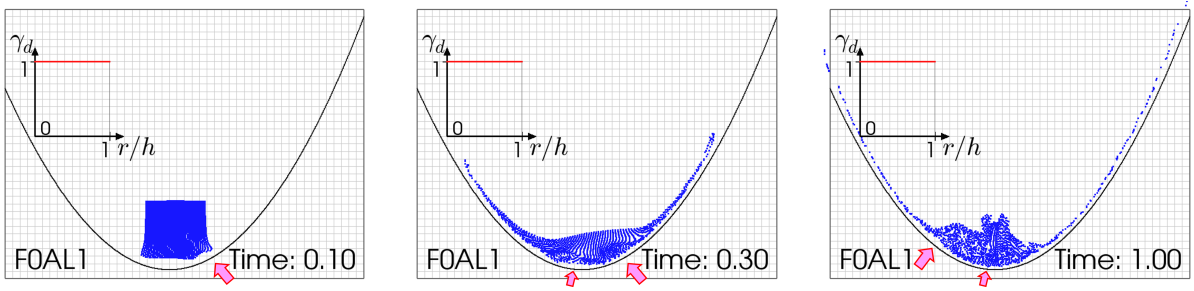


Figure 4.8: Observation focuses (pointed by red arrows) on the results of decay coefficient test with $\gamma_d = 1$

Chapter 5

VALIDATION EXAMPLES

The material point method has been applied to a number of applications involving large deformations and complex interactions among bodies. However, it is not common to use MPM to simulate fluids, even though the method has its roots in fluid mechanics, primarily because the method has been optimized for solid mechanics (i.e., complex constitutive and state tracking with modestly complex motion kinematics). Given the potential for modeling fluid-solid interactions and hence handling tsunami-driven-debris-induced loads on bridge superstructures, this research tries to enhance the capability of modeling fluid with the MPM (Chapter 3) and the flexibility of selecting the grid geometry (Chapter 4). In this chapter, two fluid examples are used to evaluate the material point method and the new proposed algorithms (described in Chapters 3 and 4). The capabilities for modeling fluid/rigid boundary phenomena would then lay the groundwork for extended analysis of more complex solid/fluid and debris phenomena.

The first example is a hydrostatic problem: water in a tank (Figure 5.1). This simple and computationally small problem is a good tool to examine the limits of MPM and the presented algorithms when modeling fluids. The second example is a complicated hydrodynamic problem, modeling tsunami-bridge interaction. This example complements the understanding learned in the first problem and provides more information about the performance of the enhanced MPM. More details can be found in the following sections.

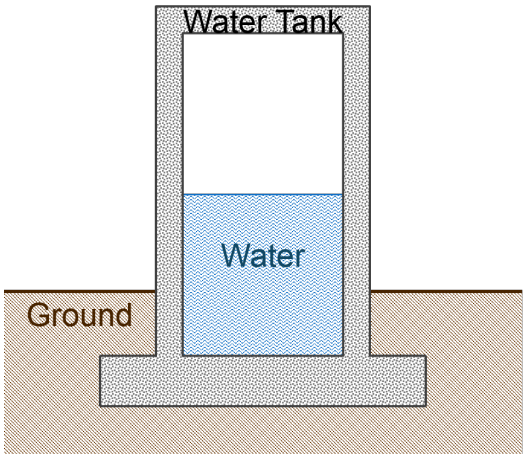


Figure 5.1: Sketch of the water tank in consideration

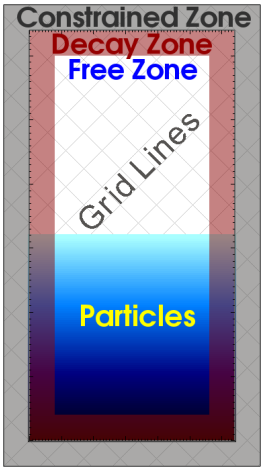


Figure 5.2: Illustration of a water tank model

5.1 Water in a Rectangular Tank

As shown in Figure 5.1, a tank half full of water at rest in stable equilibrium is used to study the performance and limitation of the new proposed smoothing algorithms (Chapter 3) and the enhanced boundary treatment (Chapter 4) comprehensively. In this problem, the water has bulk modulus $K = 2.2\,\text{GPa}$, mass density $\rho = 1000\,{}^{\text{kg}}/_{\text{m}^3}$ and viscosity $\mu = 0.001\,\text{Pa}\cdot\text{s}$; the acceleration due to gravity is $g = 10\,{}^{\text{m}^2}/_{\text{sec}}$; and the tank has 3.2 m width and 6.4 m height. The general setup of the MPM 2D strain models for this water tank is illustrated in Figure 5.2. The boundaries are treated as slip walls (Section 4.1.4). The orientation θ_g and size of the computational grid may vary in each model. Simulation duration is taken as 3 seconds, which allows the water block to have visible motion, no matter what causes this nonphysical behavior. Moreover, $t = 0.1\,\text{s}$ is selected here to observe the initial change of the stress fields due to the smoothing algorithms, the boundary conditions, and/or integration errors in the simulations. The selected time interval (0.1 second) is long enough for the stress waves to travel more than 20 rounds within the water block but too short for the particles to have evident motion, if any, from the initial stillness. For convenience,

quantification of refinement, numbers of particles per cell (ppc) and cells per length (cpl), defined for the models with $\theta_g = 0\,^\circ$ (described in the next paragraph) are also used to describe the refinement level of the models with $\theta_g = 45\,^\circ$, in which grid lines are not parallel to the edges of the square water block as illustrated in Figure 5.2 and hence cells do not necessarily have the same number of particles. In the following paragraphs, simulation results using the different smoothing algorithms C0, F0AL1 and NB (defined in Table 3.1) are first compared to each other. Effects of mesh (particle) refinement are then examined. Performance of different decay coefficient functions are discussed at the end of this section.

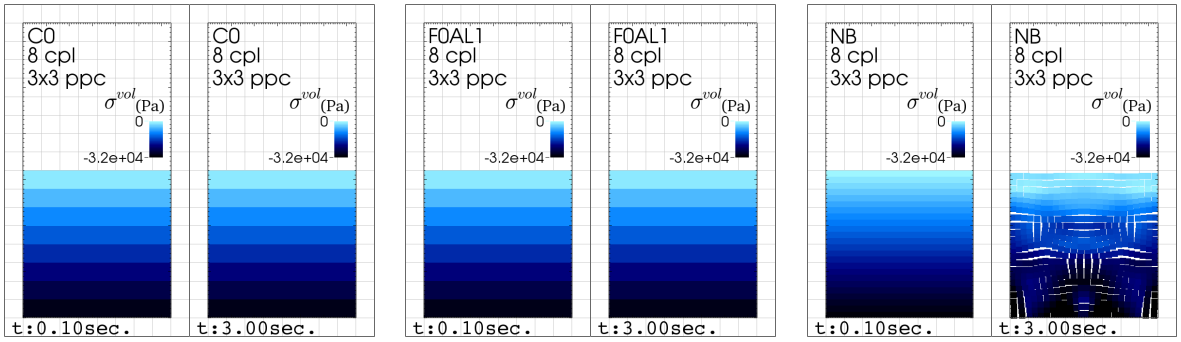


Figure 5.3: Comparison of simulation snapshots between C0, F0AL1 and NB algorithms ($\theta_g = 0\,^\circ$)

The first test is designed to show more details about what has been learned in Chapter 3, that the node-based algorithm (NB) is not suitable for hydrostatic problems. In this test, the grid orientation is selected as $\theta_g = 0\,^\circ$ to fit the geometry of the water tank, and 8 cells per length (of the square water block) and 3×3 particles per cell are used. In other words, initially the water block is uniformly discretized by sixty four cells and each cell has nine, uniformly distributed particles. Figure 5.3 compares the simulation results using C0, F0AL1 and NB at $t = 0.1\,\mathrm{s}$ and $3\,\mathrm{s}$. It shows NB gives higher resolution in the stress field at $t = 0.1\,\mathrm{s}$ compared to C0 and F0AL1. However, C0 and F0AL1 both can keep the same stress field till the end of

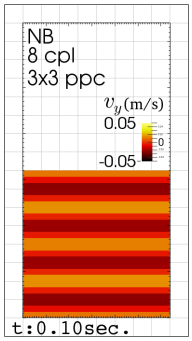


Figure 5.4: Spurious mode introduced by the node-based algorithm

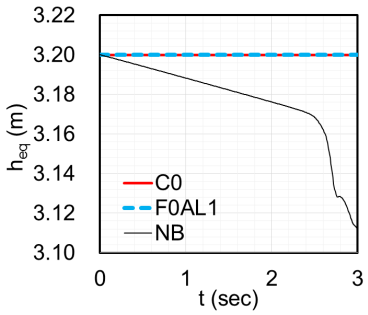


Figure 5.5: Time history of equivalent height h_{eq} (8 cpl, 3^2 ppc, $\theta_g = 0\,^\circ$)

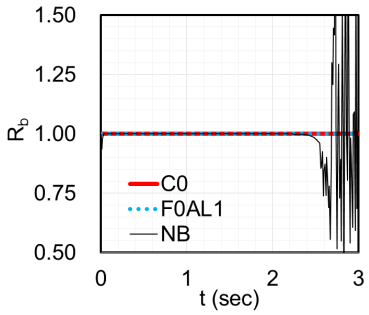


Figure 5.6: Time history of normalized reaction force R_b (8 cpl, 3^2 ppc, $\theta_g = 0\,^\circ$)

the simulations, 3 second, after the first smoothing procedures. In contrast, NB over-smooths the stress fields and changes the equilibrium status. This artificial change of the stress field causes global unbalanced downward forces and hence introduces the nonphysical reduction of volume (height), which has been seen in Figure 3.8(a) and is observed through time histories of the equivalent height h_{eq} (Figure 5.5):

$$h_{eq} = \frac{2}{N_p} \sum_{p=1}^{N_p} h_p \tag{5.1}$$

in which N_p is the numbers of particles and h_p is the distance between particle p and the bottom. Besides, NB also introduces a spurious mode which can be observed through the velocity field as shown in Figure 5.4. This mode exists because NB smooths out efficiently functions with checkerboard pattern as exhibited in Section 3.5, and hence takes away the resisting forces (stresses) corresponding to the displacement of the spurious mode. Consequently, particles move gradually according to the nonphysical velocity fields until some of them move close to and across cell edges, which introduces unbalanced nodal (internal) forces. This destabilizes the simulation and introduces nonphysical accelerations onto the particles. Finally, Figure 5.6 shows the normalized reaction force R_b at the tank bottom when NB is

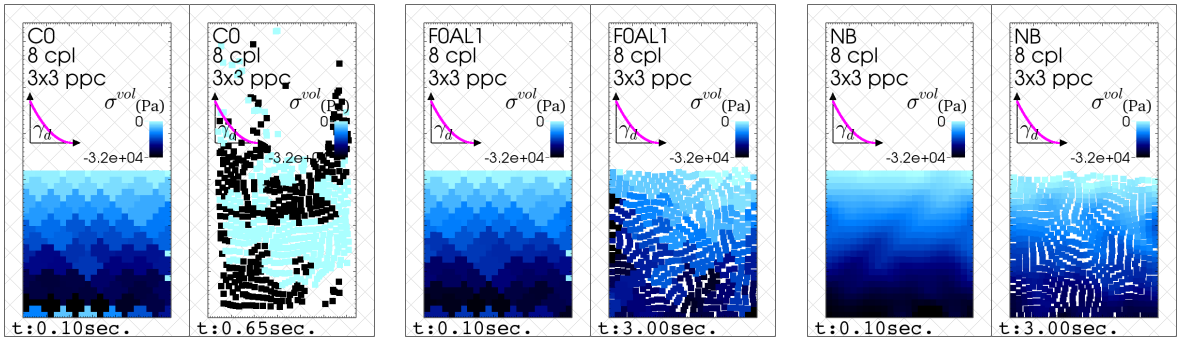


Figure 5.7: Comparison of simulation snapshots between C0, F0AL1 and NB algorithms ($\theta_g = 45\,^\circ$)

used;

$$R_b = \frac{1}{m_w g} \sum_{i=1}^{N_{nd}} \boldsymbol{f}_i^{bc} \cdot \boldsymbol{n}_b \tag{5.2}$$

Here $m_w = 10240\,\text{kg}$ is the total mass of the water; N_{nd} is the numbers of nodes; $\boldsymbol{f}_i^{bc}$ is a nodal boundary force (Section 4.1.1); and $\boldsymbol{n}_b$ is a surface normal to the tank bottom. From Figure 5.6, the normalized reaction force R_b matches with the theory value 1, even though the node-based algorithm tends to smooth out the stress field and reduce the stress near the tank bottom. This is due to the contribution of the velocity gradient in the bottom cells (Figure 5.4) which recovers the over-smoothed stress during the particle updating. R_b starts to decrease rapidly when particles move close to the cell edges, and has high frequency vibration after the particles move across the edges and introduce shock waves in the stress due to unbalanced nodal forces. The spurious (destabilization) forces triggered by integration errors may play an important role in dynamic analyses, in which particles move within cells and are not always uniformly distributed in each of them.

Next, consider the same setup as the first test except the grid is rotated $45\,^\circ$ counterclockwise around the left bottom corner of the water tank as shown in Figure 5.2. This problem targets not only the performance of specific boundary force fields but also the capabilities of the smoothing algorithms in handling destabilizing sources

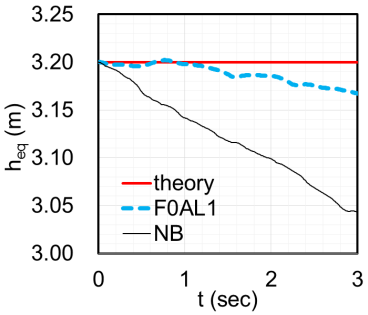


Figure 5.8: Time history of equivalent height h_{eq} (8 cpl, 3^2 ppc, $\theta_g = 45\,^\circ$)

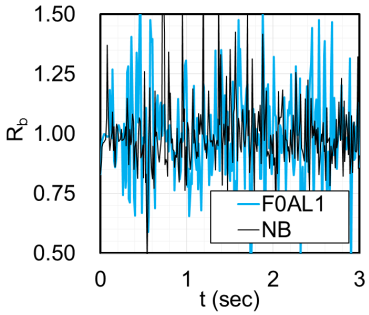


Figure 5.9: Time history of normalized reaction force R_b (8 cpl, 3^2 ppc, $\theta_g = 45\,^\circ$)

derived from integration errors. The unbalanced forces arising from integration errors can be observed in the stress fields at $t = 0.1\,\mathrm{s}$ (Figure 5.7), especially in the results using NB, which dissipates out the vibration of the redistributed stresses. Figure 5.7 also indicates that the proposed boundary treatment is able to keep the particles in the predefined domain if the simulation does not get unstable. The cell-based algorithm (C0) is not able to sufficiently control the destabilization of integration errors and hence the simulation blows up. Consistent with its design, the numerical flux algorithm (F0AL1) starts to dissipate energy like the node-based algorithm (NB). However, with the flux limiter F0AL1 significantly reduces the artificial plastic deformation and hence decreases the nonphysical volume (height) changes as shown in Figure 5.8. Nonetheless, according to Figure 5.9, support forces contributed by the wanter tank are noisy no matter whether NB or F0AL1 is used because water particles accelerated by the nonphysical forces continue slowly flowing across cells and then introduce more destabilizing sources. This error can be controlled by increasing the number of particle per cell (ppc) as discussed in the next paragraph.

In the material point method, refinement can be accomplished in two ways: increasing the number of particle per cell (ppc) or increasing the number of cells (cpl in this section). The first refinement reduces the integration errors and the second improves the accuracy of variable evolution, even though both of them increase the

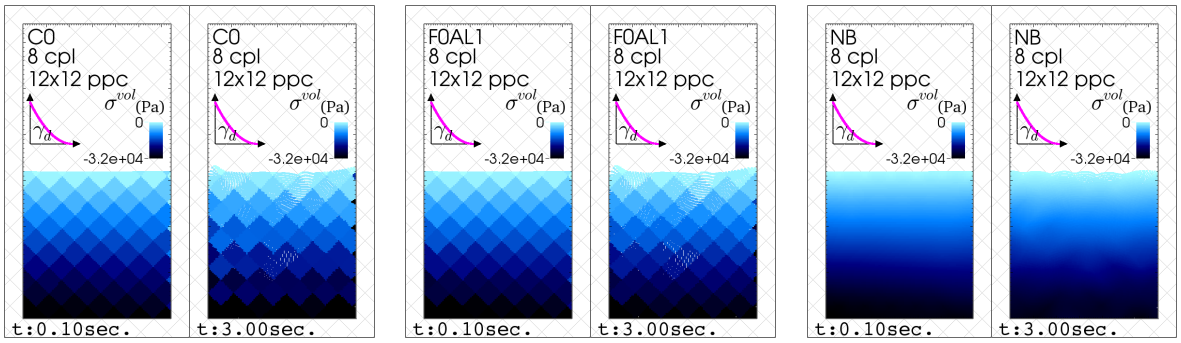


Figure 5.10: Comparison of simulation snapshots between C0, F0AL1 and NB algorithms (12^2 ppc, $\theta_g = 45\,^\circ$)

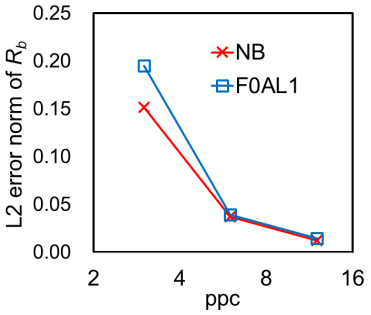


Figure 5.11: Influence of ppc refinement on L2 error norm of R_b (8 cpl, $\theta_g = 45\,^\circ$)

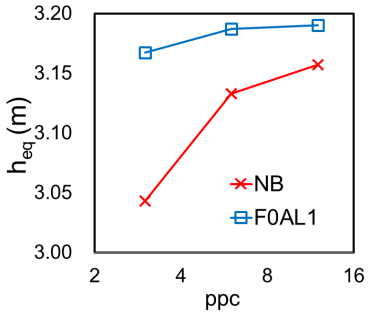


Figure 5.12: Influence of ppc refinement on h_{eq} (8 cpl, $\theta_g = 45\,^\circ$)

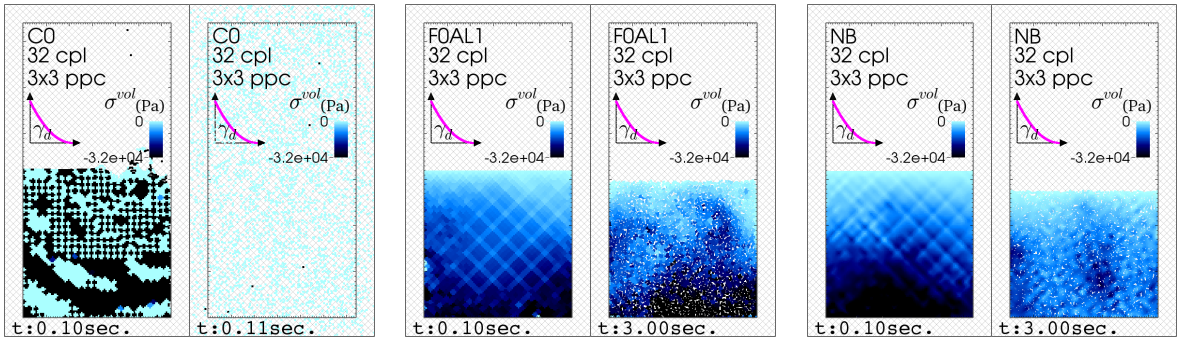


Figure 5.13: Comparison of simulation snapshots between C0, F0AL1 and NB algorithms (32 cpl, $\theta_g = 45\,^\circ$)

number of particles. For the second test discussed in the previous paragraph, increasing ppc brings down the influence of integration errors, reduces the noise in the reaction force, and most importantly stabilizes the simulation. As shown in Figure 5.10, initial integration errors are reduced significantly, and the stress fields at $t = 0.1$ s are

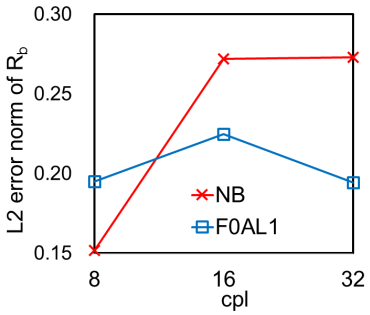


Figure 5.14: Influence of cpl refinement on L2 error norm of R_b (3 ppc, $\theta_g = 45\,^\circ$)

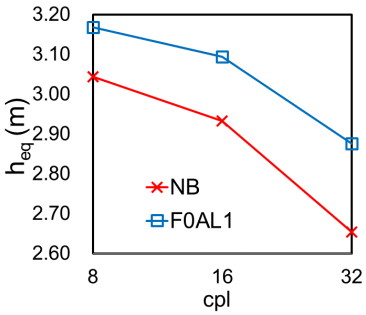


Figure 5.15: Influence of cpl refinement on h_{eq} (3 ppc, $\theta_g = 45\,^\circ$)

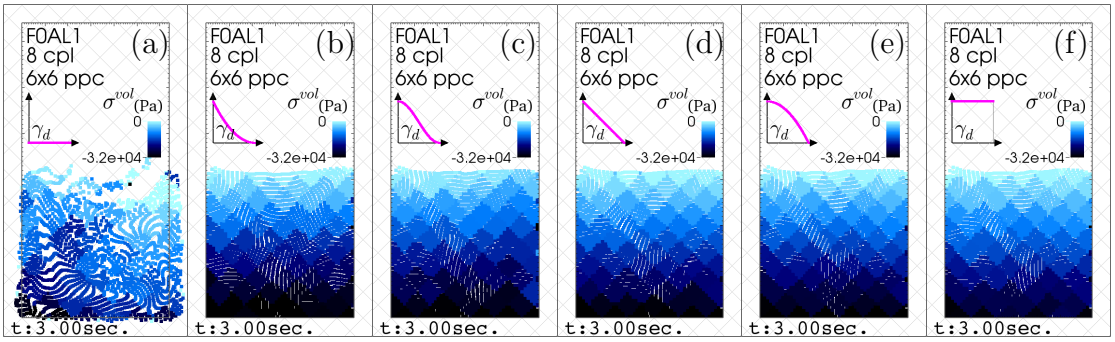


Figure 5.16: Comparison of simulation snapshots at $t = 3$ sec between decay coefficient functions

closer to the equilibrium status than those observed in Figure 5.7. Therefore, the C0 algorithm survives the test with 12 by 12 ppc, sixteen times more particles per cell than the original case. Figures 5.11 and 5.12 also indicate increasing ppc reduces the L2 error norm (noise) of the normalized reaction force and the nonphysical volume (height) changes of the water. On the contrary, Figures 5.13 to 5.15 all demonstrate that refining the grids in this problem increases the sources of destabilization and hence reduces the accuracy of the simulations. Again, both F0AL1 and NB give nonphysical stress fields at $t = 0.1$ s, at which C0 gives unstable results already

Figure 5.16 displays the final snapshots of simulations using different decay coefficient (γ_d) functions. As observed in Section 4.4, all the selected functions are able

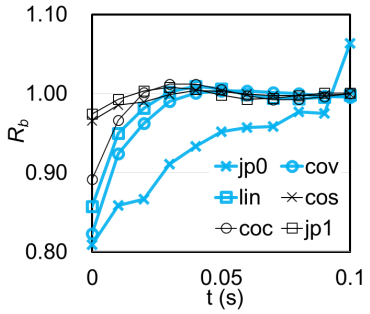


Figure 5.17: Comparison of $R_b(t)$ between decay coefficient functions (8 cpl, 6^2 ppc, $\theta_g = 45\,^\circ$)

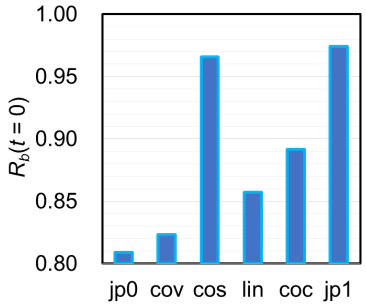


Figure 5.18: Comparison of $R_b(t = 0)$ between decay coefficient functions (8 cpl, 6^2 ppc, $\theta_g = 45\,^\circ$)

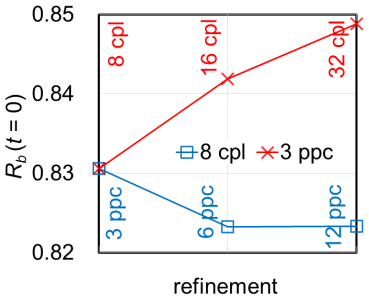


Figure 5.19: Influences of ppc and cpl refinement on $R_b(t = 0)$ ($\theta_g = 45\,^\circ$)

Table 5.1: Abbreviations of decay coefficient γ_g functions

Abbrev.	Description	Function
coc	concave	$1 - (r/h)^2$
cos	cosine	$0.5\,[1 + \cos(\pi r/h)]$
cov	convex	$(1 - r/h)^2$
jp0	jump at $r = 0$	0
jp1	jump at $r = h$	1
lin	linear	$1 - r/h$

to prevent particles from penetrating the prescribed boundaries except $\gamma_d = 0$ (Figure 5.16(a)). Figure 5.17 shows the time history of the normalized reaction forces R_b at the first 0.1 sec and Figure 5.18 compares $R_b(t = 0)$ side by side between different γ_d functions. Abbreviations of the γ_d functions used in these figures are defined in Table 5.1. It is worth noting here that the nodal values at $t = 0$ are based on initial conditions for the particles and hence are independent of the smoothing algorithms. Although Figure 5.18 indicates jp1 ($\gamma_d = 1$) offers better initial support than cov, jp1

can also provide too much constraint such that particles may be kept distant from the prescribed boundary as observed in Section 4.4. Moreover, except jp0, all the tested γ_d functions are able to converge to correct analytical values quickly in 0.05 sec (Figure 5.17). Therefore, the convex function (cov) is used in subsequent simulations. Finally, the error of $R_b(t = 0)$ is related to the nodal values and hence can be reduced by refining the cell size as shown in Figure 5.19.

In summary, the MPM and the new proposed smoothing (stabilizing) algorithms and boundary modeling strategy have been tested with a hydrostatic problem, water in a tank. The results show:

1. unlike C0 and F0AL1, NB includes a spurious mode and cannot keep the equilibrium status of static water even when there's no integration errors in the calculation;

2. integration errors shown in the MPM particle forms in Section 2.2 are sources of destabilization, which caused the C0 simulations to be unstable;

3. both F0AL1 and NB can handle the destabilization caused by integration errors by introducing stain energy dissipation;

4. F0AL1 offers a control mechanism for the energy dissipation that NB does not have;

5. increasing the number of particle per cell controls the integration errors (destabilizing sources);

6. mesh refinement can also introduce destabilization with motion of particles, and hence does not necessarily increase the accuracy; and

7. the approach of using specific force field for applying boundary conditions works as expected, and prevents particles from penetrating the prescribed boundary.

5.2 Tsunami Induced Load on Bridges

In this section four of the wave flume experiments performed by Public Works Research Institute (PWRI) for studying tsunami-bridge interaction (Nakao, Zhang, et al., 2013) are highlighted to test the performance of MPM and the new proposed algorithms in complicated hydrodynamic problems. These experiments are also the benchmark problems of the 2014 UJNR Tsunami Modeling Workshop, in which simulation results of the four cases using various numerical methods, e.g. CFD and SPH, are presented. In the following sections, more details about the experiments and the corresponding numerical models are described. Preliminary discussions about the simulation data and defects of the numerical models are then addressed, followed by a comprehensive discussions about the simulation results.

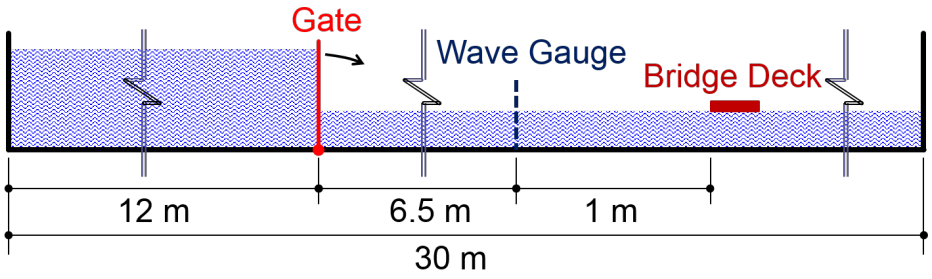
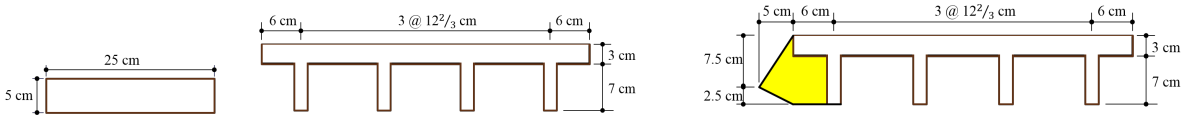


Figure 5.20: Elevation of the 1/20-scale PWRI wave flume experiments (Nakao et al., 2013)



(a) Type A (b) Type B (c) Type C

Figure 5.21: Cross sections of Bridges

Table 5.2: PWRI wave flume experiments in consideration

Case	Bridge Type	Water Depth in Tank Reservoir	Water Depth in Flume
1	A	0.5180 m	0.20 m
2	B	0.6170 m	0.10 m
3	B	0.5180 m	0.15 m
4	C	0.6170 m	0.10 m

5.2.1 PWRI Water Flume Testes

After the 2011 Tohoku tsunami, the Public Works Research Institute started a series of experimental studies targeted at determining tsunami induced loads on bridge superstructures, using a 1/20-scale wave flume (Nakao, Zhang, et al., 2013). Figure 5.20 shows an elevation of the flume experimental setup. The bridge deck was mounted on a 0.2 m high, 0.08 m thick pier standing at the middle of the test flume which has 1 m width. Water depths in the tank reservoir at the left hand (upstream) side of the gate and in the flume for the four experiments (cases) considered in this study are listed in Table 5.2.

The experiments start by releasing the holding force of the pin-ended gate at its top and letting the gate fall freely with the unbalanced forces at its two sides. This setup approximately gives a dam-break initial condition, and hence velocity and height of a shock wave traveling downstream can be determined by solving the Riemann problem for the shallow water equation (LeVeque, 1992, 2002). The resulting velocity and height for the shock wave (tsunami bore) are used as the inflow boundary conditions for the water flume models introduced in the next section.

5.2.2 Water Flume Models

Figure 5.22 shows a two-dimensional (plane strain) reduced domain model, which has initial conditions (Figure 5.23) representing the moment when the tsunami bore

Table 5.3: Reduced domain models of the PWRI wave flume experiments

Model	Bridge	h_o^l(m)	v_o^l(m/s)	h_o^r(m)	v_o^r(m/s)	L_{in}(m)	Mesh Size (mm, mm)
M1	Type A	0.3384	0.8645	0.20	0	0.5	(25.0, 25.0)
M1-L	Type A	0.3384	0.8645	0.20	0	1.0	(25.0, 25.0)
M1-R	Type A	0.3384	0.8645	0.20	0	0.5	(12.5, 12.5)
M2	Type B	0.2903	1.5454	0.10	0	0.5	(20.0, 15.0)
M3	Type C	0.3022	1.0648	0.15	0	0.5	(20.0, 15.0)
M4	Type D	0.2903	1.5454	0.10	0	0.5	(20.0, 15.0)

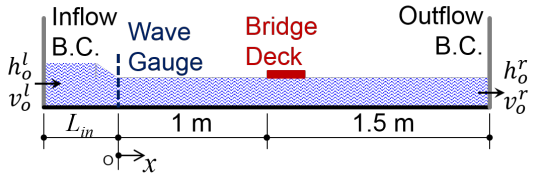


Figure 5.22: Reduced domain models for the wave flume experiments

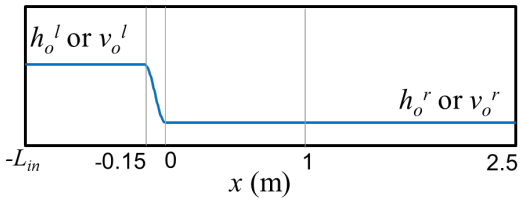


Figure 5.23: The initial velocity and height of the reduced domain model

is going to pass the wave gauge at $x = 0\,\text{m}$. The bridge is modeled as a slip wall, and surfaces of the water flume are treated using velocity boundary conditions with prescribed magnitude $0\,\text{m/s}$ and directions equal to surface normals. Furthermore, inflow and outflow boundary conditions (Section 4.2) enforce flow in the region to be laminar and have the same velocity along each stream line. The height and velocity of the inlet flow are assumed constant in time and hence equal to the initial values h_o^l and v_o^l, respectively. Table 5.3 summarizes the dimensions and initial conditions of all the numerical models considered in this study. The finest mesh sizes are selected based on the efficiency limitation of the single-threaded program CAP (Appendix A), though they are still relatively coarse compared to general applications in the literature. In the model designators, the first number indicates the corresponding experiment case number. Tags L and R represent a longer domain and refined meshes, respectively,

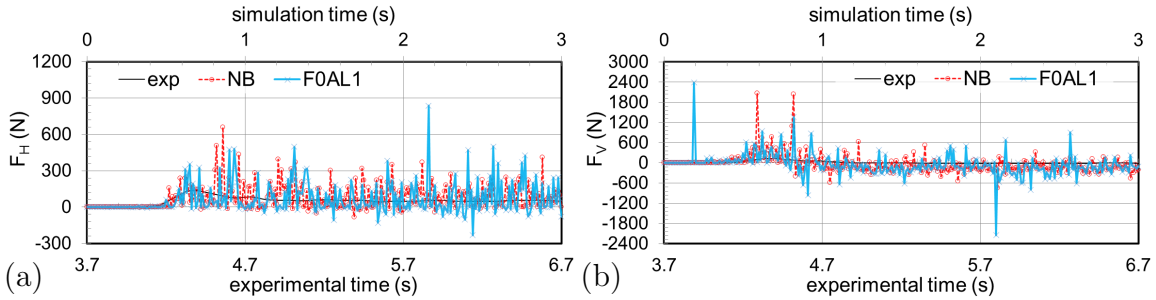


Figure 5.24: Model M1: original output data of (a) total horizontal force and (b) total vertical force on the bridge deck.

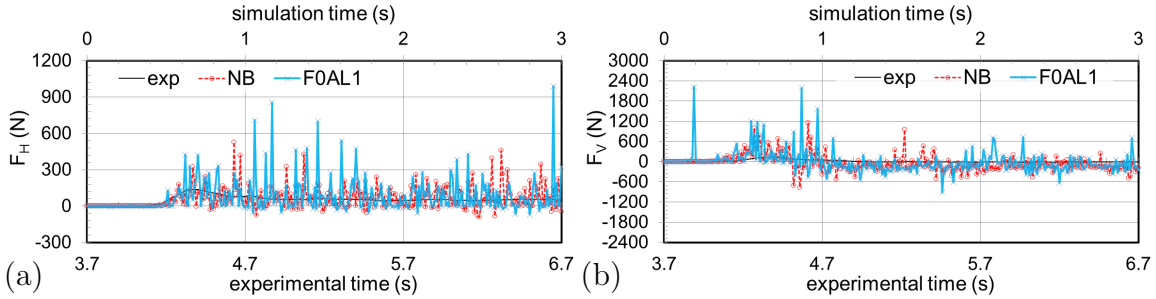


Figure 5.25: Model M1-L: original output data of (a) total horizontal force and (b) total vertical force on the bridge deck.

compared to the base model with no tags in its name. For convenience, additional tags *FL* and *NB* are added behind the model designations to indicate which smoothing algorithm, F0AL1 or NB (Table 3.1), is used in the models. For instance, M1-L-NB is a model for the Case 1 water flume experiment using the node-based smoothing algorithm and has a domain longer than the base Model M1. Before looking into the simulation results for each case, comprehensive discussions are first made about how the output data are processed and what the limitations of the reduced domain model are.

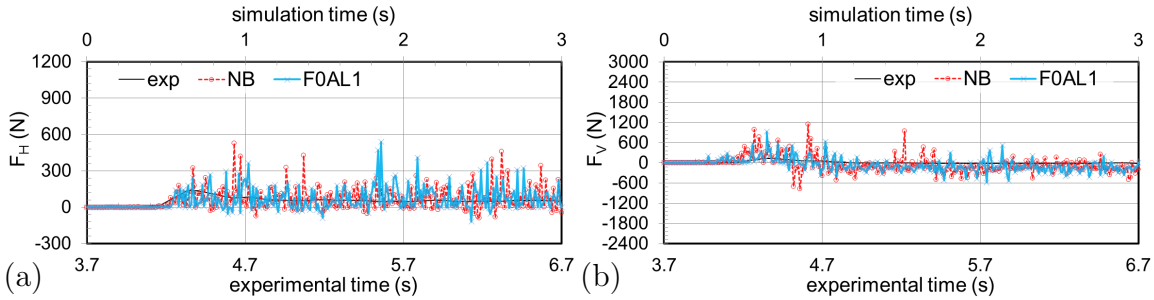


Figure 5.26: Model M1-R: original output data of (a) total horizontal force and (b) total vertical force on the bridge deck.

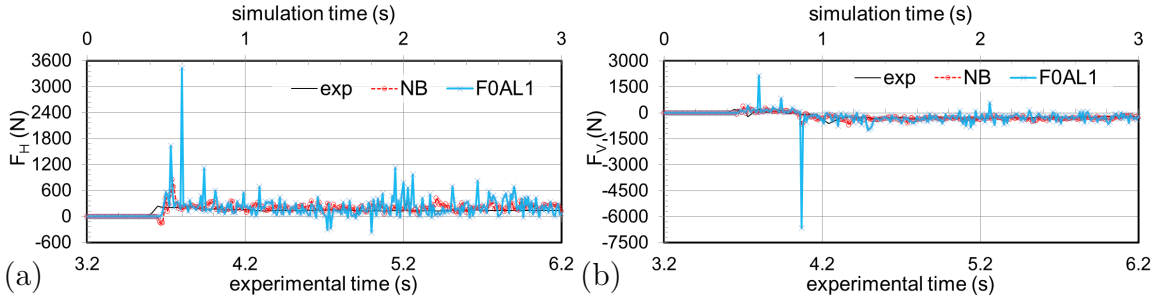


Figure 5.27: Model M2: original output data of (a) total horizontal force and (b) total vertical force on the bridge deck.

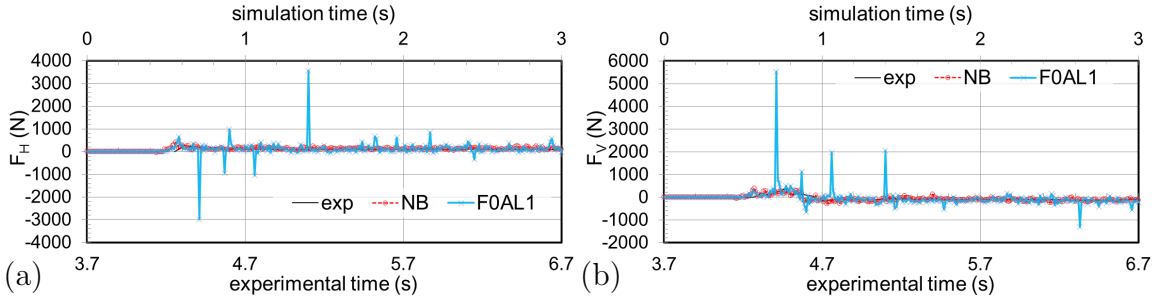


Figure 5.28: Model M3: original output data of (a) total horizontal force and (b) total vertical force on the bridge deck.

5.2.3 Data Processing

In this study, equivalent bore height and velocity at the wave gauge ($x = 0\,\text{m}$ shown in Figure 5.20) are calculated using the weighted average of particle values:

$$\text{bore height} = \frac{2\sum_p h_p w_p}{\sum_p w_p} - h_o^r \tag{5.3}$$

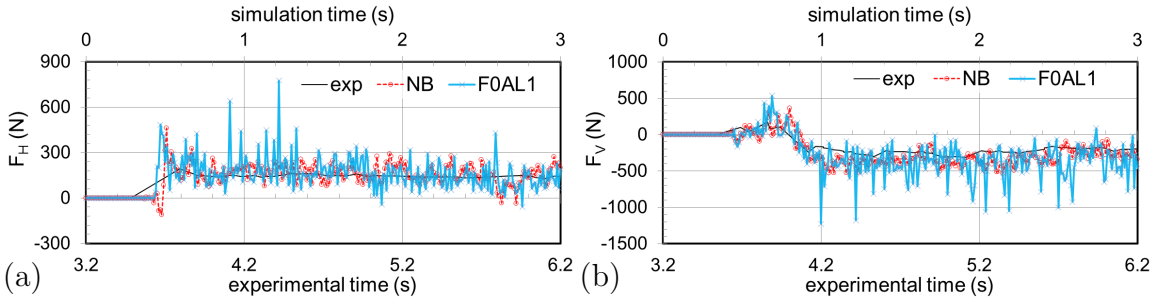


Figure 5.29: Model M4: original output data of (a) total horizontal force and (b) total vertical force on the bridge deck.

and

$$\text{bore velocity} = \frac{\sum_p v_{px} w_p}{\sum_p w_p} \tag{5.4}$$

in which h_p is the distance of particle p to the flume bottom; $w_p = \max\left[0, (1 - x_p/L_{cx})\right]$ is a weight of particle data; h_o^r is the initial water depth in the flume; and v_{px}, x_p and L_{cx} are particle velocity, particle coordinate, and mesh size in the x direction, respectively. Reaction forces are calculated as the sum of all nodal boundary forces (Section 4.1.1) around the bridge sections. However, as shown in Section 5.1 and observed in Figures 5.24 to 5.29 (the raw simulation results), MPM gives noisy nodal boundary forces because of integration errors. Therefore, in this research a simple moving average is used to smooth the output data in time. Each average value is calculated based on nine of eleven time series (with constant interval 0.01 s), by excluding the maximal and minimal values to avoid the nonphysical strong discrete data observed in the figures. Finally, the simulation time is mapped to experimental time by matching the moment when the bore front reaches the wave gauge.

5.2.4 Limitation of The Water Flume Models

Limited by the efficiency of the single-threaded program CAP, a reduced domain model is necessary to bring down the computational time. However, there are some

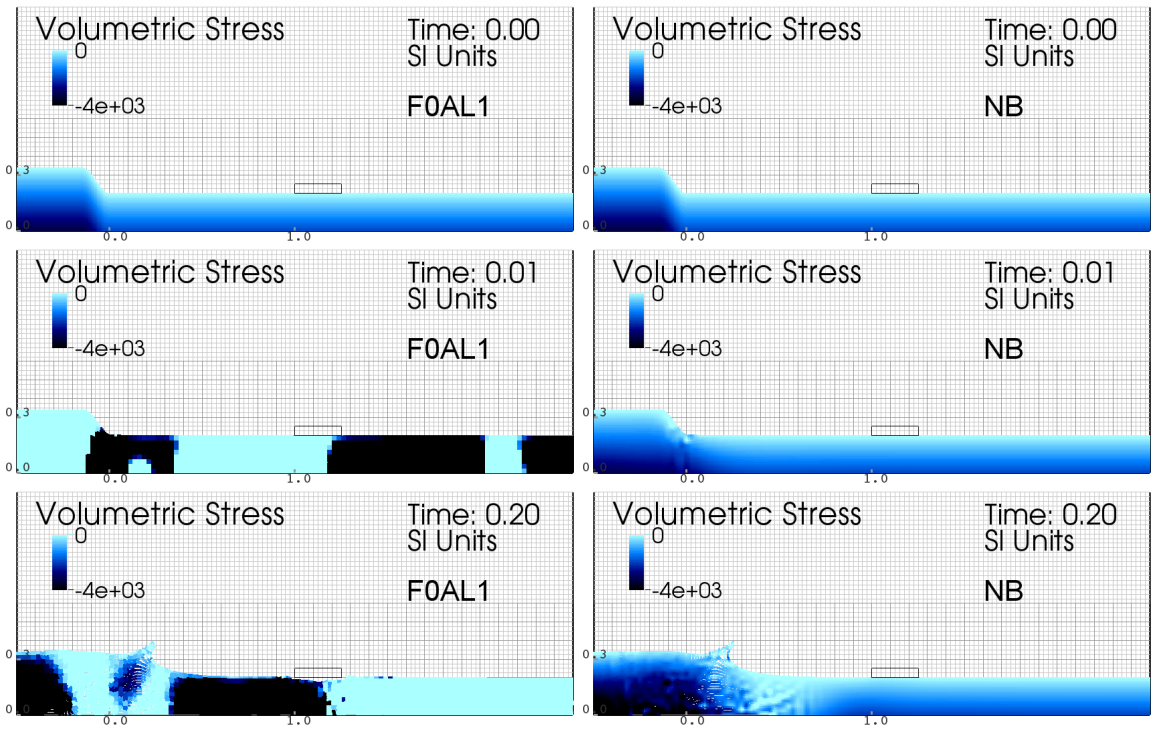


Figure 5.30: Comparison of the stress in Model M1 between different smoothing algorithms, F0AL1 and NB

defects in the numerical models that can be observed in all simulations. A preliminary discussion about these imperfections is made here so it is easier to read the simulation data in the next section.

The first issue of the numerical models is the mesh sizing. In the MPM, boundary conditions are applied on nodes to constrain motion of particles in the cells connected to the nodes. Therefore, the coarser the meshes are, the broader the regions in which the boundary conditions are felt. In the analyses presented here, the finest vertical cell size is 12.5 mm for Bridge Type A (which has depth 5 cm) and 15 mm for Bridges Type B and Type C (which both have depths 10 cm). Consequently, the influence depths of the bridge boundaries in the numerical models are at least 1.3 times larger than the real bridge depth. This can change the impact area felt by particles in the models and hence affect the flow pattern. The effects arising from this coarse

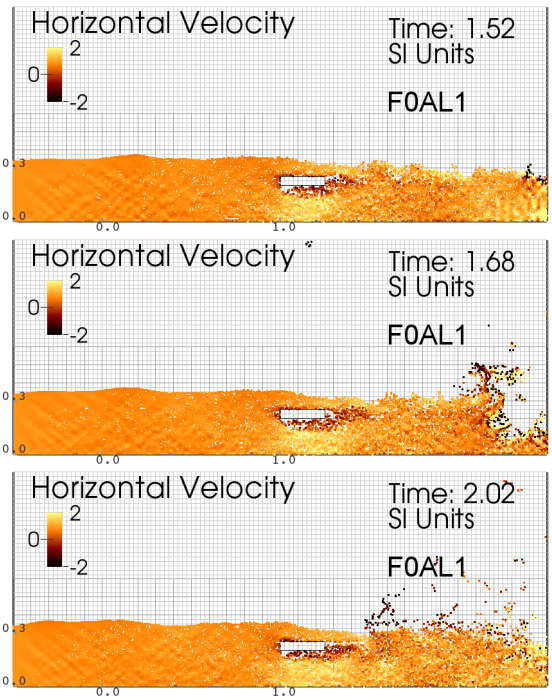


Figure 5.31: Nonphysical explosion observed in simulation results of Model M1 using F0AL1

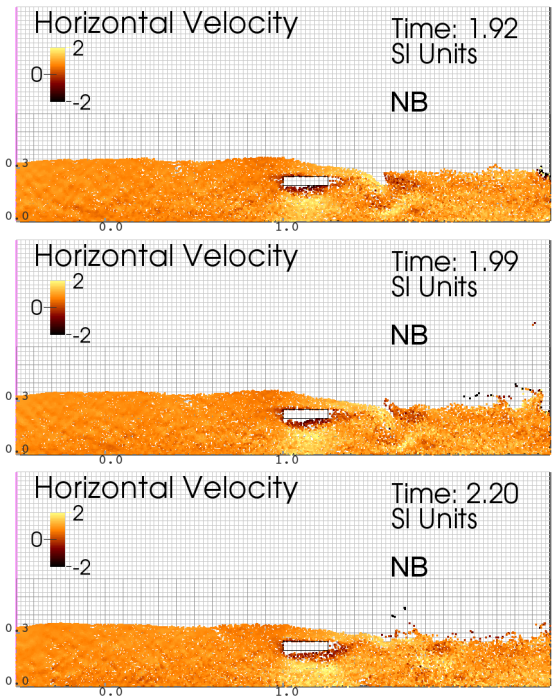


Figure 5.32: Nonphysical explosion observed in simulation results of Model M1-L using NB

discretization in simulations are discovered in Section 5.2.5.

Secondly, as shown in Figure 5.30, using the velocity boundary conditions for the inlet and outlet flow introduces nonphysical reflection of stress waves when F0AL1 (Table 3.1) is used. However, compared to the nonphysical stress waves triggered by integration errors described in Section 5.1, the spurious reflection of stress waves at boundaries seems not a major problem in the MPM models. This can be understood by looking into the simulation results using the node-based smoothing algorithm (NB), which dissipates out the shock wave propagation with its strong smoothing characteristics, and hence keeps no evident changes in the stress at the downstream region until the bore is close (Figure 5.30). In other words, reflected waves, if any, from boundaries are dissipated away within a few time steps and hence only affect

small regions adjacent to the boundaries. Therefore, taking Model M1 as an example, the oscillation shown in the bridge loading time histories (red line in Figure 5.24) is mainly caused by the combined effects of dynamic behavior of the fluid around the bridge and integration errors of MPM. Comparing the results of Model M1 using different smoothing algorithms, the bridge loadings given by M1-FL do not look more noisy than those of M1-NB but include a few more outlier points, which are believed to arise from integration errors at edge cells as observed in Section 3.6.2, but further study will be necessary to confirm this. Also, the integration errors at edge cells are believed responsible for destabilizing the flow models, of which the most severe cases are displayed in Figures 5.31 and 5.32 when F0AL1 and NB are used, respectively. More discussion of this can be found in the following section.

5.2.5 Simulation Results and Comprehensive Discussions

Comparisons of simulation results of six models against experimental observations are presented in Figures 5.33 to 5.38, respectively. In general, all models can capture the flow depth at the wave gauge (Figures 5.33(a) to 5.38(a)) reasonably well, especially for Cases 2 and 4. However, Figures 5.33(b) to 5.38(b) show the initial velocities derived from the shallow water equation overestimate the bore velocities during the first second after the bore front passes the wave gauge location. Models M2 and M4 are able to capture the bore velocities after the first second; Model M3 captures the trend of velocity changes but overestimates the flow speed; Model M1 fits the experimental data during 1 to 2 simulation seconds but then loses accuracy tracking the flow velocity. Since either enlarging the computational domain (M1-L) or refining the mesh sizes (M1-R) does not result in significant improvement in capturing the bore velocity, the inlet boundary conditions assuming constant horizontal velocity in the vertical direction might not fit with the real flow characteristics in the Case 1 experiment. More experimental data would be necessary to study this observation. Noise observed in the time histories of the gauge data shown in Figures 5.33(a) and (b)

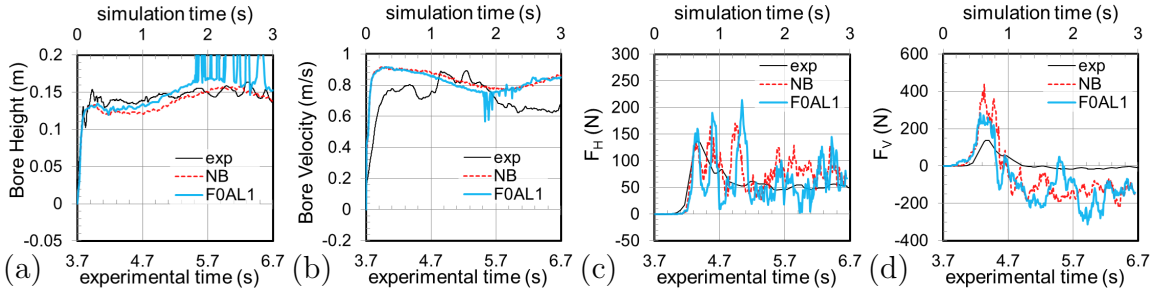


Figure 5.33: Model M1: (a) bore height and (b) bore velocity at the 1m-gauge; and (c) total horizontal force and (d) total vertical force on the bridge deck.

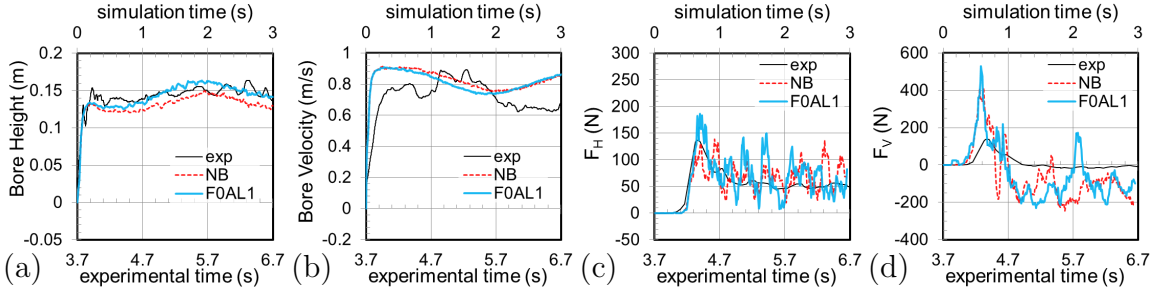


Figure 5.34: Model M1-L: (a) bore height and (b) bore velocity at the 1m-gauge; and (c) total horizontal force and (d) total vertical force on the bridge deck.

are induced by the particles flying upstream after the numerical instability observed in Figure 5.31. This nonphysical phenomenon does not have evident influence on the simulated gauge data.

Figures 5.33(c) to 5.38(c) compare the smoothed horizontal loads (Section 5.2.3) computed to act on the bridges against the experimental data. Figure 5.33(c) shows a very different horizontal loading history between the simulation results of M1 and experimental data, especially the first three periodic vibrations with gradually increased peaks. There is no certain explanation for this periodic loading in this study, but the most likely reason for this spurious loading pattern is the simplified setup of inflow boundary conditions, based on the observation that M1-L (Figure 5.34(c)) gives reasonable agreement in horizontal forces against the experimental data, but

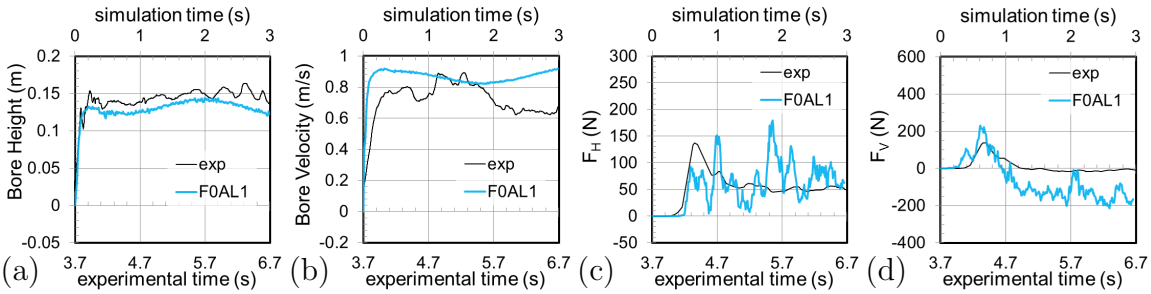


Figure 5.35: Model M1-R: (a) bore height and (b) bore velocity at the 1m-gauge; and (c) total horizontal force and (d) total vertical force on the bridge deck.

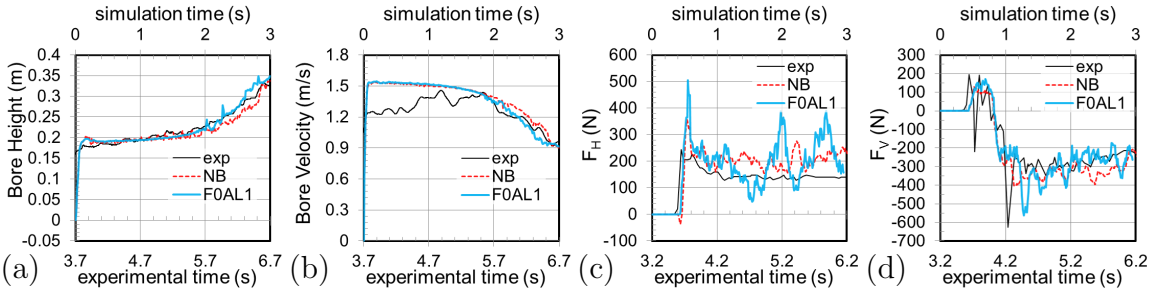


Figure 5.36: Model M2: (a) bore height and (b) bore velocity at the 1m-gauge; and (c) total horizontal force and (d) total vertical force on the bridge deck.

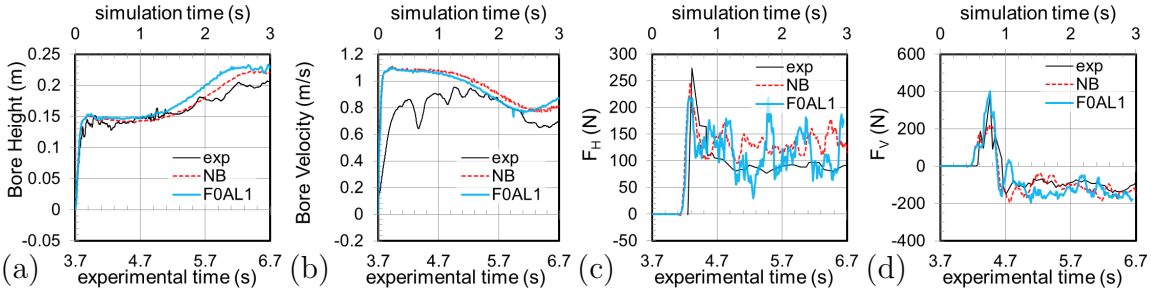


Figure 5.37: Model M3: (a) bore height and (b) bore velocity at the 1m-gauge; and (c) total horizontal force and (d) total vertical force on the bridge deck.

M1-R (Figure 5.35(c)) still includes the gradually increasing peaks. The other periodic vibration shown in Figure 5.34(c) after the first simulation second can also be

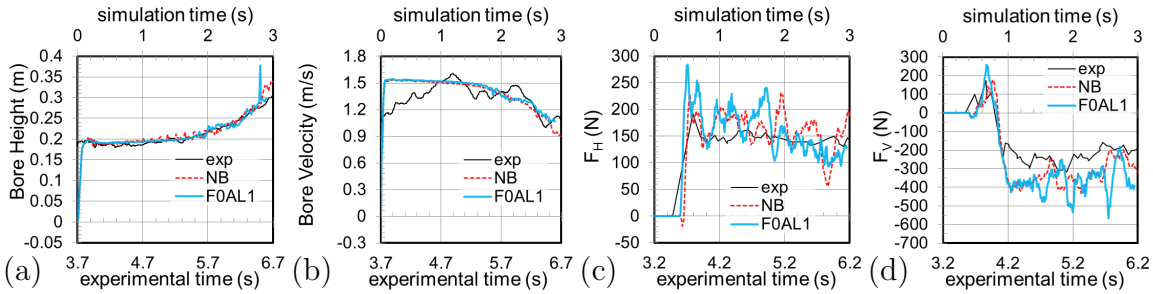


Figure 5.38: Model M4: (a) bore height and (b) bore velocity at the 1m-gauge; and (c) total horizontal force and (d) total vertical force on the bridge deck.

observed in other cases, and are explained in the next paragraph.

Figure 5.36(c) shows M2 can capture the horizontal loads of Case 2 within tolerable range with the relatively coarse mesh. The two periodic oscillations observed after the first simulation second are induced by the flow injecting back to the river after passing over the bridge floor. Figure 5.39 displays the time at which the first surface wave is created by the converging flow, and the time when the lateral load has extreme values beyond the first impact. The minimum loads occur if the surface waves smash on the bridge tail, and the maximum loads occur when the water tends to move downstream. In between, M2-FL has similar lateral loads with M2-NB when the water accumulates at the downstream side of the bridge with relatively low velocities. These surface waves can also be observed in M2-NB. However, because the node-based algorithm can dissipate the impact stress quickly, the influence of the tail waves in M2-NB is relatively insignificant on the smoothed horizontal loading history.

Figures 5.37(c) and 5.38(c) show the lateral forces for Models M3 and M4. In Case 3, the numerical model can capture the peak value but overestimates the forces after the bridge deck is flooded. In Case 4, M4-NB has good agreement with the experimental data, but M4-FL overestimates the lateral forces in the first two seconds. Again, oscillation can be observed in the time history of lateral loading in both cases.

Compared with the horizontal forces, vertical tsunami loads are relatively simple to

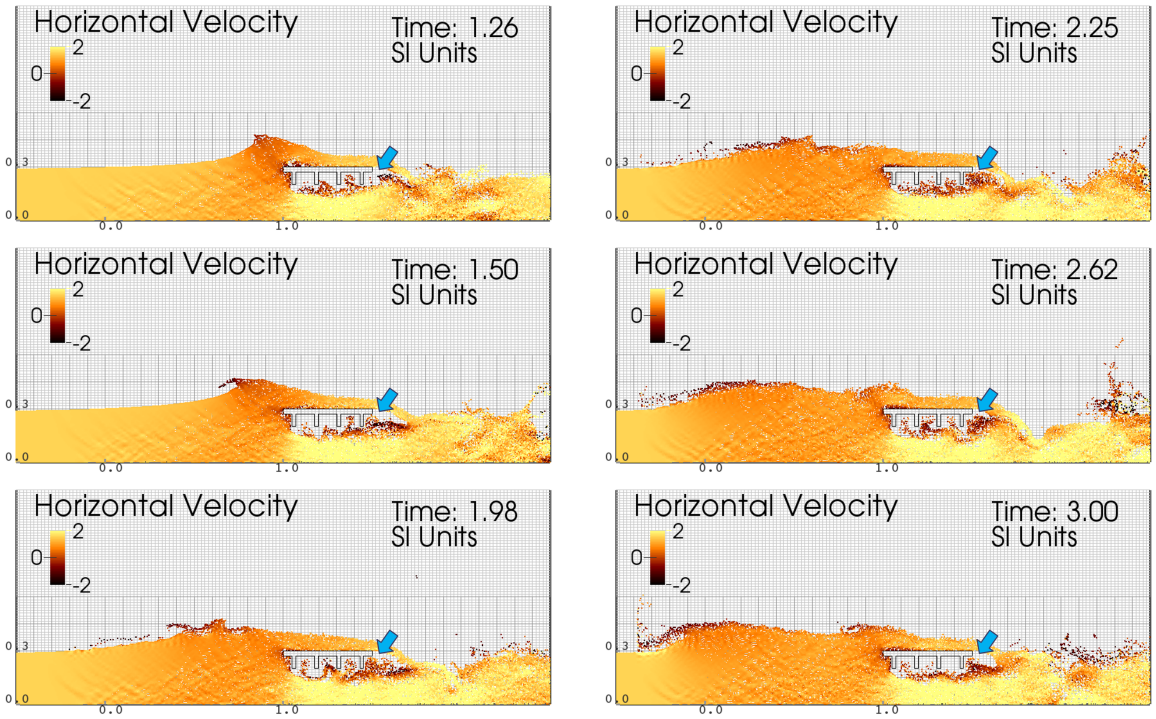


Figure 5.39: Surface waves observed at the bridge tail (pointed by the blue arrow) in Model M2 using F0AL1

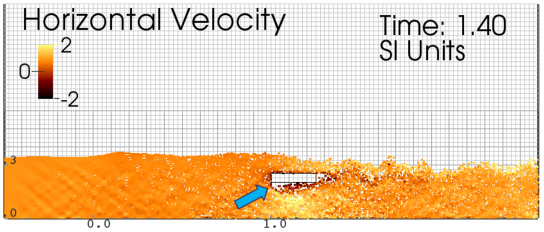


Figure 5.40: Illustration of void zone (pointed by the blue arrow) observed beneath the Type A bridge deck in a Case 1 (Model M1) simulation

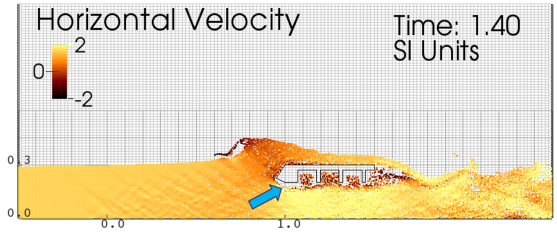


Figure 5.41: Illustration of void zone (pointed by the blue arrow) observed beneath the Type C bridge deck in Case 4 (Model M4) simulation

understand. In spite of the poor performance in horizontal forces, M1 derives similar vertical loads with M1-L as shown in Figures 5.33(d) and 5.34(d). The overestimated

uplifting forces introduced by the bore front can be improved with finer meshes as shown in Figure 5.35(d). The downward forces after the bridge is submerged in the tsunami are mainly caused by the lack of buoyancy and cannot be improved with mesh refinement. This is a result of the influence region of the MPM boundary conditions discussed in 5.2.4. This influence region acts like a protection layer surrounding the bridge. Therefore, particles need enough momentum or external force to break into the region. Consequently, the boundary cells beneath the bridge generally include fewer particles than those at the bridge deck (Figure 5.40), because of the gravity load. The same problem can be observed in the simulation results of Model M4 (Figure 5.41), which also has good agreement with the uplifting forces during the first impact and larger down-force after the bridge is under water (Figure 5.38(d)). For Cases 2 and 3, because the wet area beneath the girder bridge is small, both M2 and M3 can capture the vertical forces during the whole simulation as shown in Figures 5.36(d) and 5.37(d).

5.2.6 Summary

This section presented results for modeling the dynamic interaction of a tsunami with bridge structures of various shapes using the MPM. The investigated cases are as documented in (Nakao, Zhang, et al., 2013). Simulations were performed for a reduced domain, which utilizes the shallow water equations for a rectangular channel to define suitable inlet boundary conditions. However, limited by the efficiency of the self-developed single-threaded program, relatively coarse meshes (comparing to the general applications in the literature) are used in this study. Results show:

1. the simulated inflow and outflow boundary conditions introduce nonphysical reflection of stress waves, but the influence is minor compared to the integration errors of MPM;

2. the errors can be more severe around the outflow boundary (behind which the

particles are deleted), and in some cases introduce instability;

3. all models can capture the flow depth reasonable well but have noticeable errors in velocities;

4. longer simulation domains (M1-L) give better results in Case 1, which implies some turbulence happens beneath the water and hence the laminar flow assumption at the inlet boundaries of Model M1 is overly simplified;

5. surface waves, triggered by converging flow, impact the bridges from the tail and introduce lateral forces pointing upstream, which are mitigated (smoothed out) by NB;

6. the imperfections of boundary conditions in MPM result in down-forces in Case 1 and Case 4 by impeding particles from entering the boundary cells beneath the bridges, and this is a source of buoyancy.

5.3 Summary and Findings from Chapter 5

This section first validates the numerical flux algorithm (F0AL1) using hydrostatic problems. In the case without integration errors, results show F0AL1 is able to perform like the cell-based approach (C0); in the case including integration errors, F0AL1 starts to dissipate strain energy similar with the node-based algorithm (NB) to stabilize the simulations, in which C0 easily goes unstable. However, with a control mechanism, F0AL1 can keep more physical phenomenons than NB. For instance, the stress wave propagation observed in the second example (i.e. simulating tsunami-bridge interaction). In the complex hydrodynamic problem, F0AL1 demonstrates the capability handling integration-error-induced destabilization.

In both examples, especially the second one, the refinement is limited by the attainable computational speed of CAP, that suggests a more efficient program is

necessary for further practical studies. Besides, although both F0AL1 and NB can mitigate the error-induced destabilization, reaction forces computed on nodes are noisy because of the error-induced high frequency oscillation. Further study on this issue is necessary.

Chapter 6

DEBRIS-INDUCED LOADS ON BRIDGE SUPERSTRUCTURES DURING A TSUNAMI

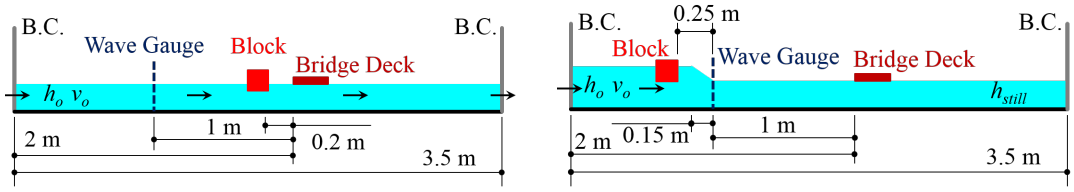
In this chapter demands on flat bridge decks due to tsunami-driven debris are studied. The idealized debris modeled here is relatively large compared to the bridge deck and waterway/channel, and could be thought of as being on the scale of houses or boats. Consequently, both damming effects and impact can be considered as the cause for additional demands on decks. With the efficiency limitation of the single-threaded program CAP, relatively coarse meshes are used in the following studies of debris-induced loads on bridges caused by the damming effects (Section 6.1) and impacts (Section 6.2).

6.1 Damming Effect

6.1.1 Numerical Models

Two types of tsunami-related flows are considered. Figure 6.1(a), Model I, shows a general situation where long waves raise the water level in a channel close to the bottom of the bridge deck. Figure 6.1(b), Model II, illustrates the case of a bridge subjected to a large tsunami bore in a coastal area.

Model M1-L (defined in Table 5.3) is used as the prototype for the numerical models. Model I (Figure 6.1(a)) has smaller water depth 0.2 m (i.e. 0 m bore height) at the left hand side but the same water depth at the right hand side; Model II keeps both the boundary conditions the same with M1-L. As is common in mesh-free methods, boundaries in MPM are not crisply defined, and so boundary reaction forces (i.e., bridge loading) are derived from cells adjacent to boundaries as they are



(a) Model I (b) Model II

Figure 6.1: Simplified models for (a) general cases and (b) great tsunami cases

Table 6.1: Dimension and material properties of blocks (debris)

	Dimensions (m)			Mass density	Mass	Young's modulus	Poisson's ratio
Model	X-dir.	Y-dir.	Z-dir.	ρ (kg/m^3)	m (kg)	E (GPa)	ν
B1	0.15	0.15	1.0	200	0.45	0.03	0.2
B2	0.15	0.15	1.0	400	0.90	0.03	0.2
B3	0.15	0.15	1.0	800	1.80	0.03	0.2
B4	0.30	0.15	1.0	200	0.90	0.03	0.2
B5	0.30	0.15	1.0	400	1.80	0.03	0.2
B6	0.60	0.15	1.0	200	1.80	0.03	0.2

populated with stressed material points, which can be observed in the 0.2-second snapshot of Model M1-FL shown in Figure 5.30. Therefore, the boundaries for the deck in both Model I and II are shifted one cell width away from the initial still-water level.

Dimension and material properties of the blocks representing idealized debris are listed in Table 6.1. The initial vertical location of each block is set at the equilibrium buoyancy level, and the initial velocity of each block is equal to the water flow speed (i.e., 0.8645 $^{\text{m}}/_{\text{s}}$ in both Models I and II). In addition, Young's modulus and Poisson's ratio of the debris are selected to make p-wave speeds in the blocks slightly smaller than the acoustic speed of fluid in the numerical models. Consequently, the minimum time step in the simulation will only be controlled by the bulk modulus of fluid.

Finally, the node-based algorithm (Chapter 3) is applied here for fluid simulation because, as observed in Section 5.2, NB can give relatively smooth and reasonable

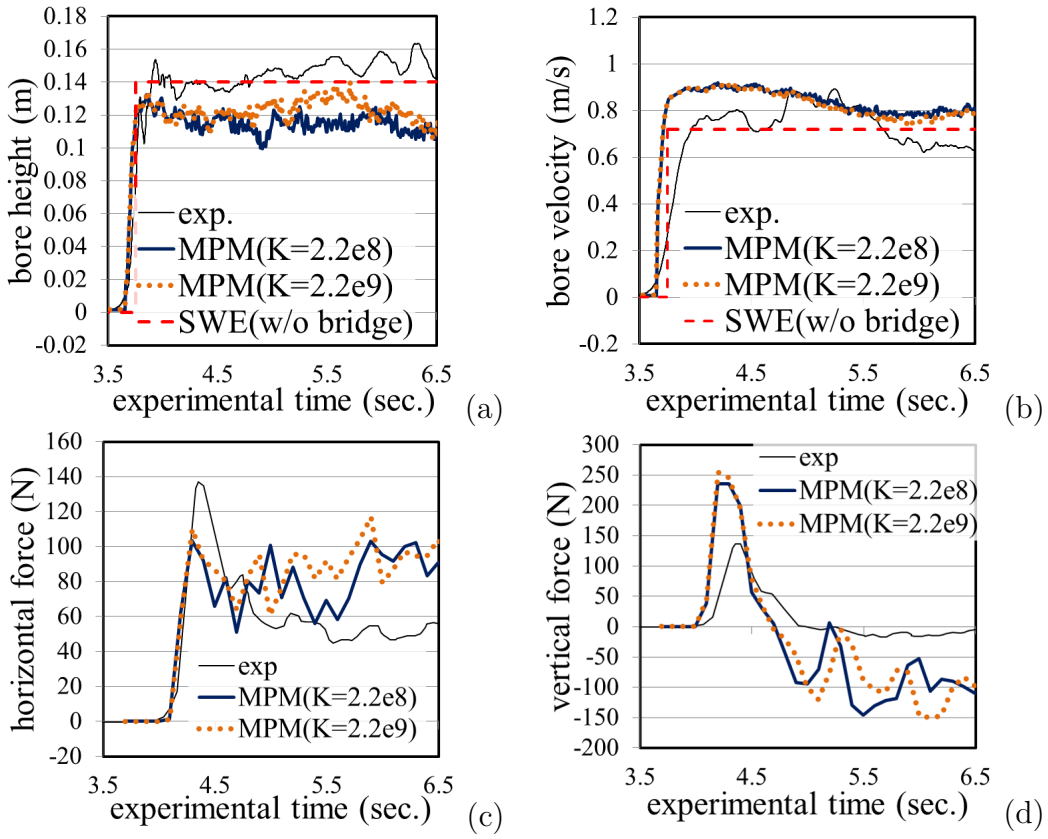


Figure 6.2: Model testing: (a) bore height at the 1m-gauge, (b) bore velocity at the 1m-gauge, (c) total horizontal force and (d) total vertical force on the bridge deck.

results in the reduced domain models (defined in Table 5.3). A non-slip contact algorithm (W. Hu & Chen, 2003) is used for modeling the interaction between fluid and floating blocks, or debris.

6.1.2 Model Testing

Prior to considering the effects of debris, experimental setups for fluid/bridge interactions without debris were modeled, and those results are described in this section. As indicated earlier, the numerically modeled bridge deck had to be placed one computational cell above the initial water level to allow free flow below the deck. This results in a small geometric deviation from the experimental setup and hence slightly

influences the simulation results compared to those observed in Figure 5.34.

A comparison of simulation results of Model II without debris against experimental observations is presented in Figure 6.2. Figure 6.2(a) and (b) show average bore height (Equation (5.3)) and bore velocity (Equation (5.4)) at the wave gauge (Figure 6.1(b)) for the first 6.5 seconds. The simulation time is mapped to experimental time by matching the moment when the bore front reaches the wave gauge (as described in Section 5.2.3). Results show the relocated bridge deck allows water to flow with less obstacle, that the bore height at the wave gauge in Model II is relatively flat in time compared to the height in Model M1-L, and the bore velocity in Model II has less deceleration than the velocity in Model M1-L around 5.7 experimental seconds.

Figure 6.2(c) shows the smoothed horizontal forces (Section 5.2.3) on the 1/20 model scale bridge deck. Compared with Model M1-L, the simulation has smaller initial peak force and larger horizontal forces once the flow completely engulfs the structure. The resulting vertical forces are shown in Figure 6.2(d). This figure shows the presence of an initial peak, which is slightly larger than the peak value observed in Model M1-L. The entire vertical force graph appears to be affected by a significant downward force of about 100 N throughout the event. This is caused by the imperfections of boundary conditions in MPM as discussed in (Section 5.2.5).

In the MPM, the analysis time step, Δt, has to satisfy the Courant-Friedrichs-Lewy (CFL) condition (Courant, Friedrichs, & Lewy, 1967). For one-dimensional cases and explicit solvers, the condition is

$$\Delta t \leq \frac{\Delta x}{c} \tag{6.1}$$

where Δx is cell size and c is the acoustic wave speed of the material, proportional to the square root of the bulk modulus K. Because water has a high bulk modulus, 2.2 GPa, computing efficiency is hampered because of the small time steps necessary to resolve the acoustic waves. To reduce computing time, a 0.22 GPa (i.e., one tenth

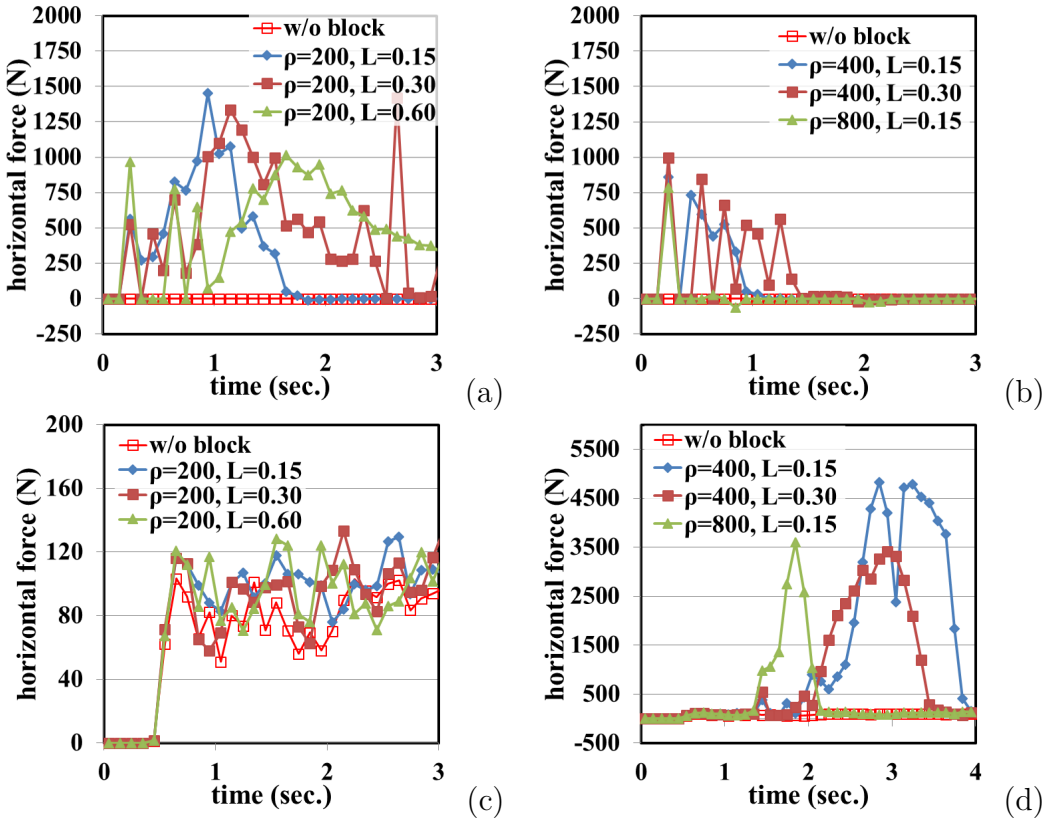


Figure 6.3: Time history of total horizontal forces caused by debris for Model I [(a) and (b)] and Model II [(c) and (d)]

of the original value) bulk modulus is used. Simulation results using the artificial bulk modulus are also plotted in Figure 6.2 and show good agreement with those using the one corresponding to real water.

6.1.3 Influence of Debris-Induced Damming

Compared to impact forces, which happen in a very short time interval, damming effects may introduce larger demands to bridge structures due to the longer loading duration, even though peak values of additional forces caused by damming are generally much smaller than those due to impact forces. Figure 6.3 shows time histories of total horizontal forces applied on bridge decks for Model I and Model II during

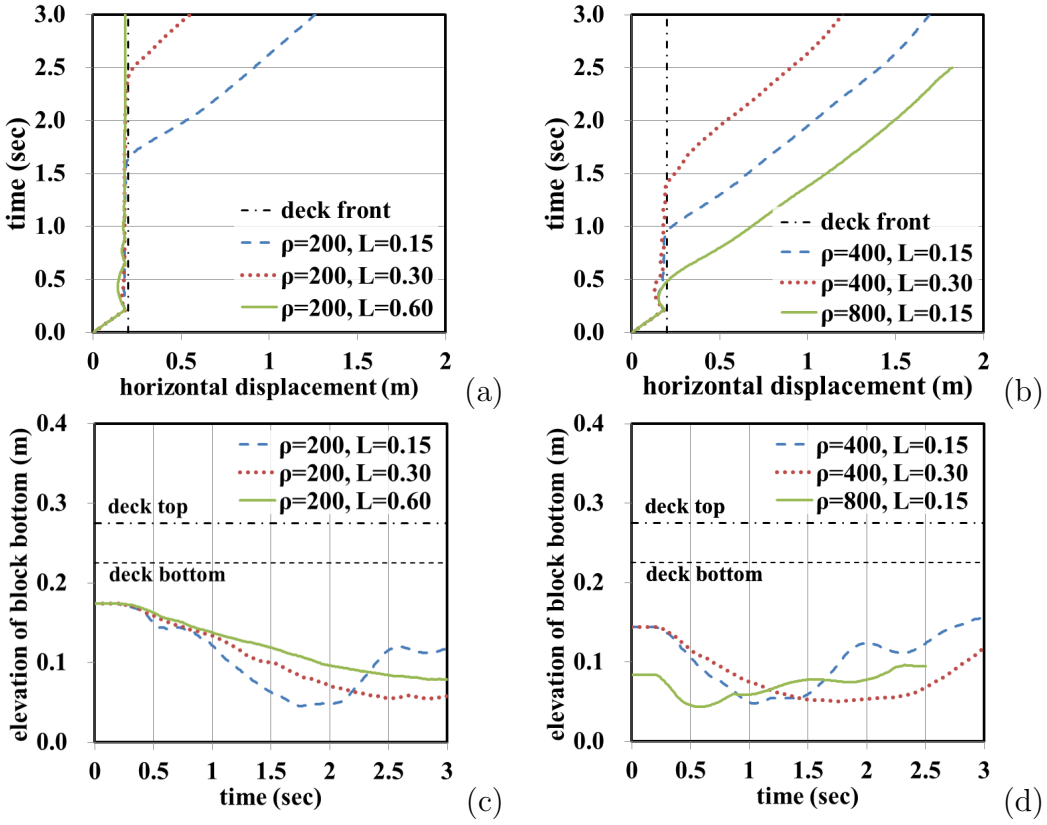


Figure 6.4: Time history of horizontal displacement of block centroids [(a) and (b)] and elevation of block bottom surface [(c) and (d)] for Model I

the first three seconds. In these figures, each point represents a time average of 10 time steps, which have fixed intervals of 0.01 second. This smooths the values of the impulse forces which could cloak peaks, but this procedure is appropriate in this case since forces with longer periods are those of interest for damming effects.

Time histories of debris centroid motions in the horizontal direction and debris bottom elevation for all models are plotted in Figure 6.4 and Figure 6.5. These figures display relative location between blocks and the bridge deck, and indicate when contact occurs. Peak horizontal forces F_{de} acting on bridge decks excluding impact forces are listed in Table 6.2. Results of Model I with block B3 ($\rho = 800\,^{\mathrm{kg}}/_{\mathrm{m}^3}$) are not included in this table because, as shown in Figure 6.4(b) and Figure 6.4(d),

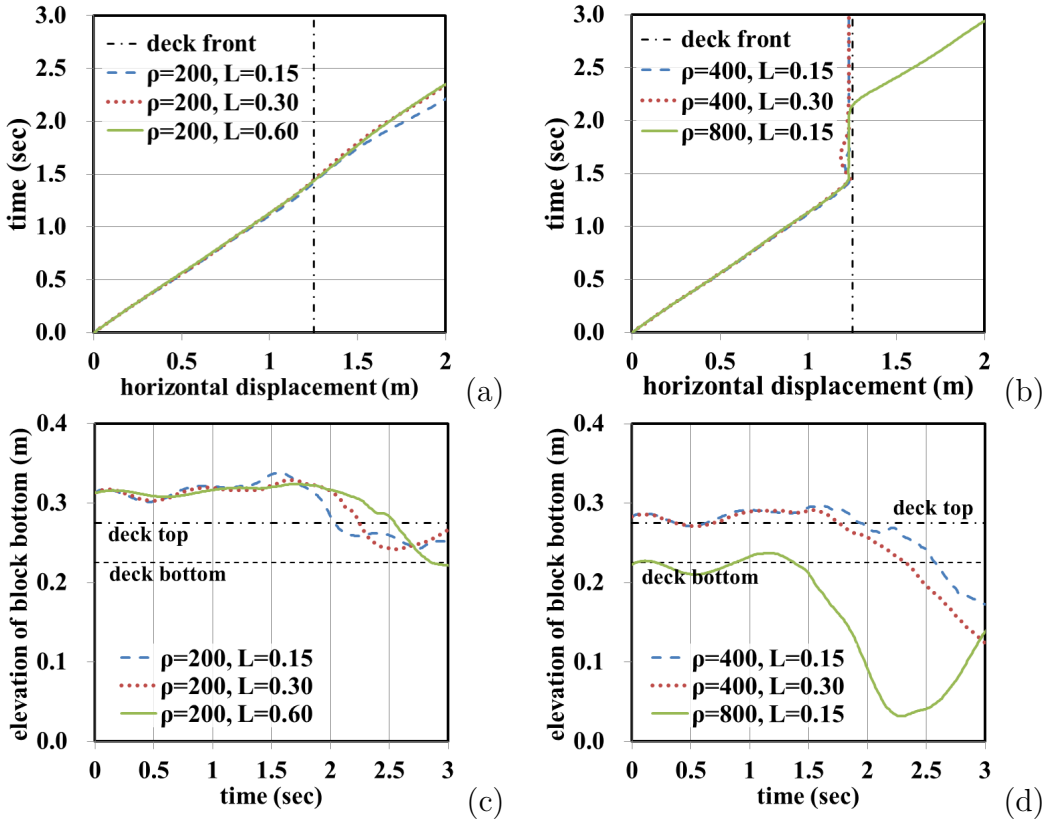


Figure 6.5: Time history of horizontal displacement of block centroids [(a) and (b)] and elevation of block bottom surface [(c) and (d)] for Model II

Table 6.2: Peak horizontal forces acting on bridges and minimum net cross-section area of water flow

	Model I					Model II		
	B1	B2	B4	B5	B6	B2	B3	B5
F_{de} (kN)	1.45	0.73	1.42	0.85	1.01	4.83	3.60	3.42
A_{if} (m^2)	0.170	0.140	0.170	0.140	0.170	0.225	0.218	0.225
A_{pf} (m^2)	0.129	0.115	0.124	0.111	0.112	0.183	0.144	0.124

the debris passes beneath the bridge deck after collision without damming the flow. In addition, Figure 6.5(a) and Figure 6.5(c) show that blocks with density 200 kg/m^3 in Model II are always floating above the bridge deck and do not bump against the deck.

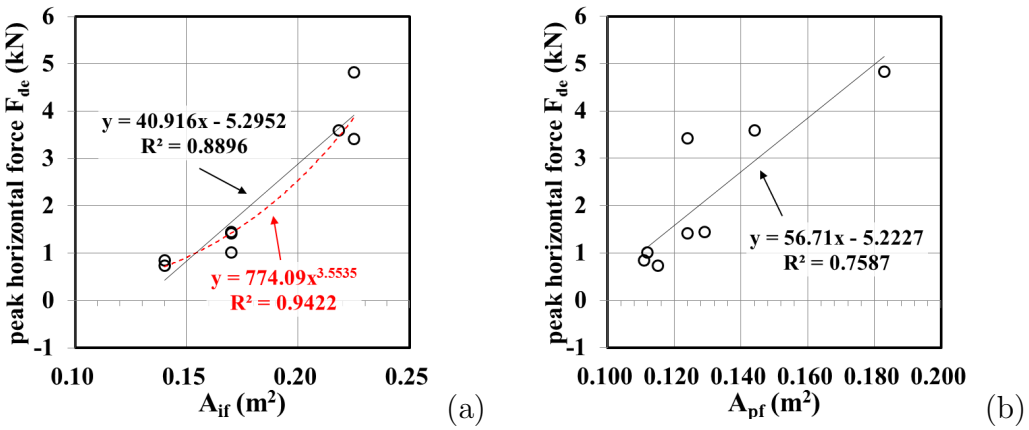


Figure 6.6: Correlation between peak horizontal force and minimum net cross-section area of water flow at (a) initial condition and (b) the moment when peak values happen

Consequently, debris does not result in additional force on the bridge (Figure 6.3(a)). For all other cases, debris significantly increases the demands on bridge structures, from 120 N to more than 3500 N in Model II cases. Minimum cross-section area (height) of water flow in the models at initial condition, A_{if} and at the moment when peaks happen, A_{pf}, are also listed in the table. Based on these data, Figure 6.6 shows values of F_{de} are linearly related to both A_{if} and A_{pf}, and are strongly related to $(A_{if})^{3.55}$. More study will be necessary to identify the key factors that influence peak forces when the fluid domain is partially or fully blocked by debris.

6.2 Debris Impacting Bridge Superstructures

6.2.1 Introduction

With huge peak values, impact forces due to debris (solid objects) can cause severe local damage on bridge components, even though they happen in a very short time. In the literature, the impact forces are evaluated using simple equations based on fundamental physics. In ASCE/SEI 7-10 (*Minimum Design Loads for Buildings and Other Structures*, 2013), for instance, maximum impact forces F_{Ip} are calculated using

impulse-momentum based equations of the form

$$F_{Ip} = \frac{\pi m_d v_d}{2\Delta t_I} \tag{6.2}$$

where m_d and v_d represent the mass and impact velocity of the debris, and Δt_I is the time interval over which debris stopped from its original velocity v_d. Recommended and measured values of Δt_I range from 10^{-3} seconds to 1.0 seconds. This diversity of Δt_I results in very different impact forces and hence demands on structural systems. To improve Equation (6.2) and avoid Δt_I, the flexibility of the solid object representing the debris is considered. By simplifying the collision of a debris into a bridge deck by a 1D spring model with equivalent stiffness k_{eq}, Equation (6.2) can be rewritten as,

$$F_{Ik} = c_I v_d \sqrt{k_{eq} m_d} \tag{6.3}$$

where c_I is a constant. Piran Aghl et al. (2014) suggested c_I be taken as 1.0 for impact forces caused by a 20-ft shipping container colliding with a wall in air, and evaluated the equivalent stiffness k_{eq} with

$$k_{eq} = \frac{E_d A_c}{L} \tag{6.4}$$

where E_d and L are the modulus of elasticity and length of debris (shipping container), respectively; and A_c is contact area. Consequently, with $m_d = \rho_d A_d L$ Equation (6.3) becomes

$$F_{Ic} = c_I v_d \sqrt{E_d A_c \rho_d A_d} = \left(v_I A_d \sqrt{E_d \rho_d}\right) \left(\sqrt{\frac{A_c}{A_d}}\right) \tag{6.5}$$

Ko et al. (2014) verify Equation (6.3) in their study but use experimental data to determine the equivalent stiffness k_{eq} instead of Equation (6.4) for their 1/5-scale shipping container. Furthermore, they found that impact forces for in-water cases are no greater than 17 percent of the corresponding impact forces for in-air cases, and

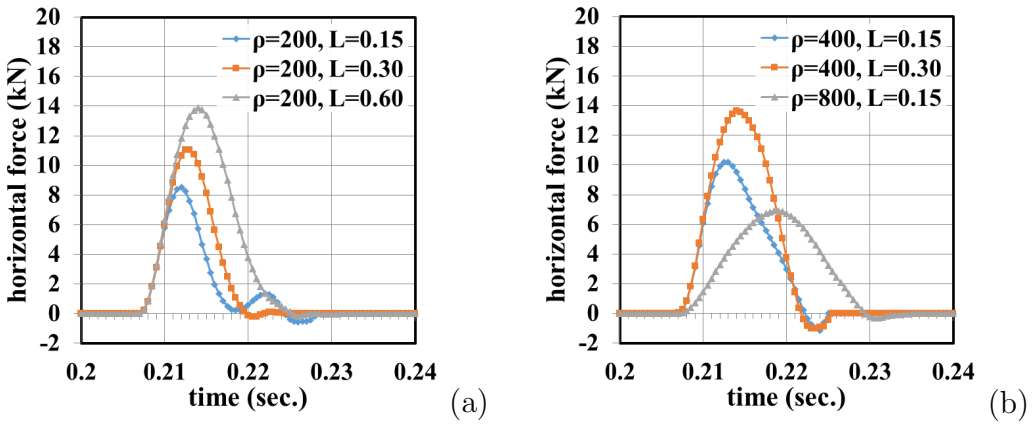


Figure 6.7: Time history of total impact forces caused by debris [(a) and (b)] for Model I

can be approximately predicted with Equation (6.3) using the same k_{eq} determined from in-air tests.

6.2.2 Preliminary Study of Debris Impacts

Model I, the simplified tsunami model without bore (Figure 6.1(a)), is used in the preliminary study of impact forces on bridge decks caused by tsunami-driven debris. As listed in Table 6.1, the material properties are the same for all six blocks (idealized debris) except for the mass density, which ranges from 200 $^{kg}/_{m^3}$ to 800 $^{kg}/_{m^3}$. Three different lengths, 0.15 m, 0.30 m and 0.60 m, in the direction of flow (impact) with the same cross-section area leads to three different block masses, 4.5 kg, 9 kg and 18 kg. The contact area of all blocks is equal to the cross section area of the bridge deck, 0.05 m^2, except for block B6, which has an A_c equal to 0.005 m^2.

Figure 6.7 shows time history results for total horizontal forces during the interval when the first collision happens. Impact forces observed from the models and calculated using Equations (6.2) and (6.3) are listed in Table 6.4. Related data for each debris model case can be found in Table 6.3. Figure 6.8 displays correlations between impact forces and the variables considered in this study.

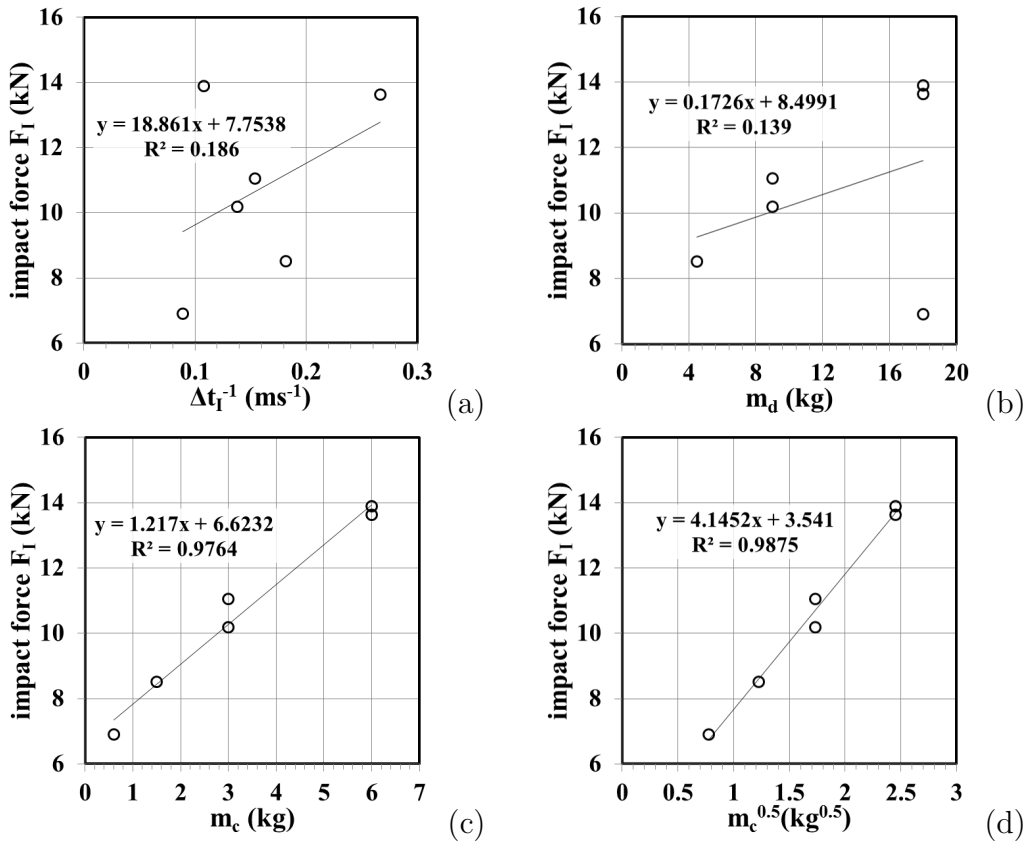


Figure 6.8: Relation between debris-induced impact force F_I and (a) Δt^{-1}, (b) m_d, (c) m_c and (d) $\sqrt{m_c}$

Figure 6.8(a) shows Δt_I^{-1} is irrelevant with respect to impact forces F_I for the six cases studied. Figures (6.8)(b) and (c) indicate that, instead of debris mass m_d, "contact mass" m_c,

$$m_c = \rho_d L A_c = m_d \frac{A_c}{A_d} \tag{6.6}$$

is more relevant to impact forces. Based on these observations it is concluded that Equation (6.2) is not a suitable equation for evaluating impact forces of tsunami induced debris, and Equation (6.3) can give better estimates of impact forces, but with the debris mass m_d replaced with "contact mass" m_c in the equation. The use of "contact mass" m_c also implies an additional factor related to L is necessary in

Table 6.3: Debris data required for evaluating impact forces in Model I

Model	L (m)	A_d (m^2)	m_d (kg)	E_d (GPa)	v_d (m/s)	Δt_I (ms)	A_c (m^2)	m_c (kg)
B1	0.15	0.15	0.45	0.03	0.8645	5.50	0.05	1.5
B2	0.15	0.15	0.90	0.03	0.8645	6.50	0.05	3.0
B3	0.15	0.15	1.80	0.03	0.8645	9.25	0.05	6.0
B4	0.30	0.15	0.90	0.03	0.8645	7.25	0.05	3.0
B5	0.30	0.15	1.80	0.03	0.8645	3.75	0.05	6.0
B6	0.60	0.15	1.80	0.03	0.8645	11.25	0.005	0.6

Table 6.4: Debris impact forces of debris for Model I

Model	F_I (kN)	F_{Ip} (kN)	F_I/F_{Ip}	F_{Ik} (kN)	F_I/F_{Ik}
B1	8.52	1.11	7.67	5.80	1.47
B2	10.19	1.69	6.04	8.20	1.24
B3	6.91	2.17	3.18	3.67	1.88
B4	11.05	1.88	5.88	5.80	1.91
B5	13.63	6.52	2.09	8.20	1.66
B6	13.90	2.64	5.26	5.80	2.40

Equation (6.5). This is true, otherwise the three time history lines in Figure 6.7(a) should overlap each other. Figure 6.8(d) shows $\sqrt{m_c}$ has slightly higher correlation with F_I compared to m_c, as suggested in Equation (6.3). Further studies about the effects of contact area and water flow on impact forces due to debris are discussed in the following.

6.2.3 Contact Area and Eccentricity

In this study, the influence of contact area and eccentricity on impact forces are evaluated with a parameter study using two-dimensional (plane strain) solid blocks. To simplify the problem, only in-air conditions are considered here(i.e. no water). Figure 6.9 illustrates the model setup. Model descriptions and symbols shown in

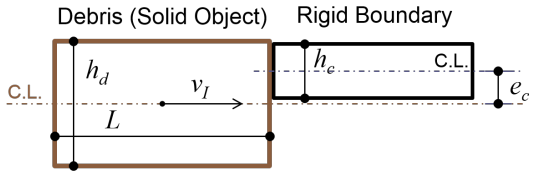


Figure 6.9: Illustration of model setup for studying the influences of contact areas on impact forces

Table 6.5: Variables considered in the study of impact forces due to an 2D solid block

Variable	Symbol	Unit	Values
height	h_d	(m)	3.0
thickness	t_d	(m)	1.0
length	L	(m)	3, 6, 9, 12, 15
initial velocity	v_I	($^{m}/_{s}$)	3, 6, 9, 12, 15
mass density	ρ_d	($^{kg}/_{m^3}$)	200, 400, 600, 800, 1000
Young's modulus	E_d	(GPa)	10, 5, 3, 1, 0.5, 0.3, 0.1
Poisson's ratio	ν_d		0.1, 0.2, 0.3
contact height [eccentricity]	h_c [e_c]	(m) [(m)]	3.0 [0.0]
			1.5 [0.75, 0.00]
			1.0 [1.00, 0.75, 0.00]
			0.5 [1.25, 1.00, 0.75, 0.00]

this figure are listed in Table 6.5 along with the corresponding values considered in this study. This results in 26250 combinations (models). Debris is treated as solid objects and is initially adjacent to a rigid boundary simulating the moment right before contact. Analysis time t_{ana} of each model is taken as

$$t_{ana} = \frac{4L}{c_d} \tag{6.7}$$

in which $c_d = \sqrt{E_l/\rho_d}$ is the stress wave propagation speed and E_l is a longitudinal elastic modulus. For a plane strain problem, E_l can be calculated with

$$E_l = \frac{E_d}{1-\nu_d^2} \tag{6.8}$$

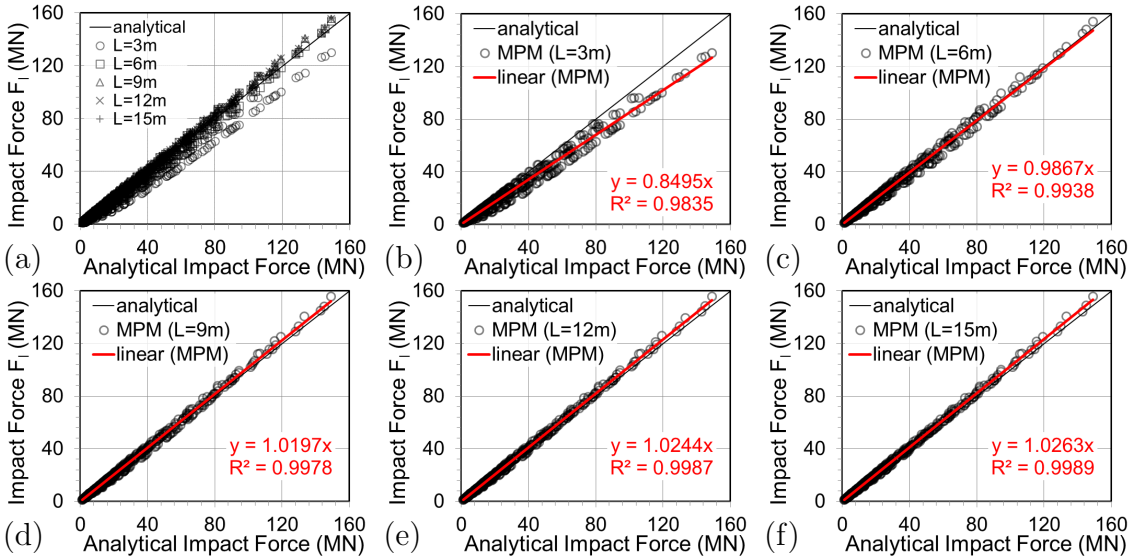


Figure 6.10: Validation of MPM models on impact forces

Regular square grids with cell size 0.5 m by 0.5 m, and nine (3 by 3) particles per cell are used.

Figure 6.10 compares the simulation results of all models with full contact surface (i.e. $h_c = 3\,\text{m}$) against theory values:

$$F_I^{th} = t_d v_I h_d \sqrt{\rho_d E_l} \tag{6.9}$$

The simulation results have good agreement with the theory when the solid blocks have longitudinal lengths L greater or equal to 6 m, but have about 15% error when $L = 3\,\text{m}$ as shown in Figure 6.10(b). This could be caused by the coarse mesh size (which is controlled by the efficiency of CAP) and hence is identified as the accuracy limitation of this study. To study the influence of the contact area $A_c = t_d h_c$ and eccentricity e_c on the impact forces, first the case of $e_c = 0\,\text{m}$ is investigated. Figure 6.11(a) shows that a decrease in contact area brings down the impact on the boundary, which is consistent with Equation (6.5) suggested by Piran Aghl et al.

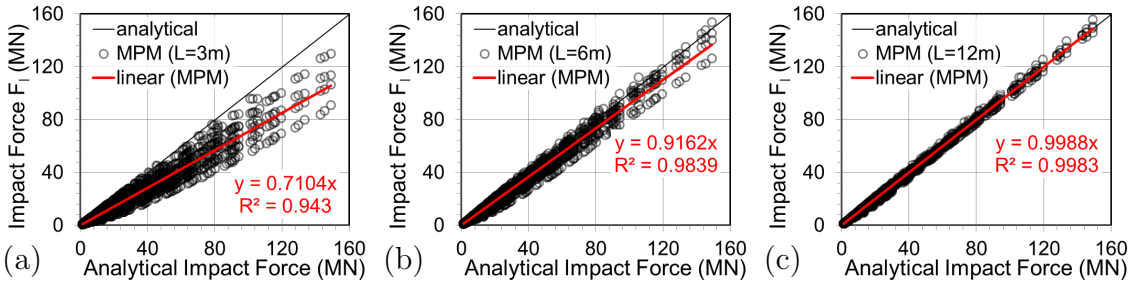


Figure 6.11: Influences of contact area on the debris-induced impacts ($e_c = 0\,\mathrm{m}$)

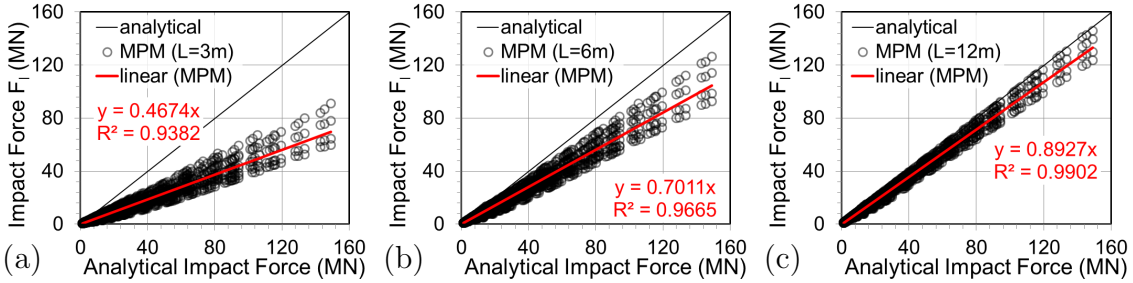


Figure 6.12: Influences of eccentricities of impact forces on the debris-induced impacts ($h_c = 0.5\,\mathrm{m}$)

(2014). Comparing Figures 6.11(a) to (c) the influence of A_c is lessen with increases of the block lengths L. Next, with a fixed contact area $A_c = 0.5\,\mathrm{m}^2$ ($h_c = 0.5\,\mathrm{m}$), which allows most variants of e_c, the eccentricity affect the impact forces slightly in this study and has less influence with longer longitudinal length L (Figure 6.12). The effect of L on the influence of A_c and e_c (on impact forces) can also be observed through Figure 6.13(b) to (f), each of which plots all model results corresponding to a specific L value. According to the observation in Figures 6.11 to 6.13, a modified height (area) $h_c^{'}$ with $\tan\theta = 5$ (as illustrated in Figure 6.14) is suggested to be used in Equation (6.9), i.e.

$$F_I^{mod} = t_d v_I h_c^{'} \sqrt{\rho_d E_l} = v_I A_c^{'} \sqrt{\rho_d E_l} \tag{6.10}$$

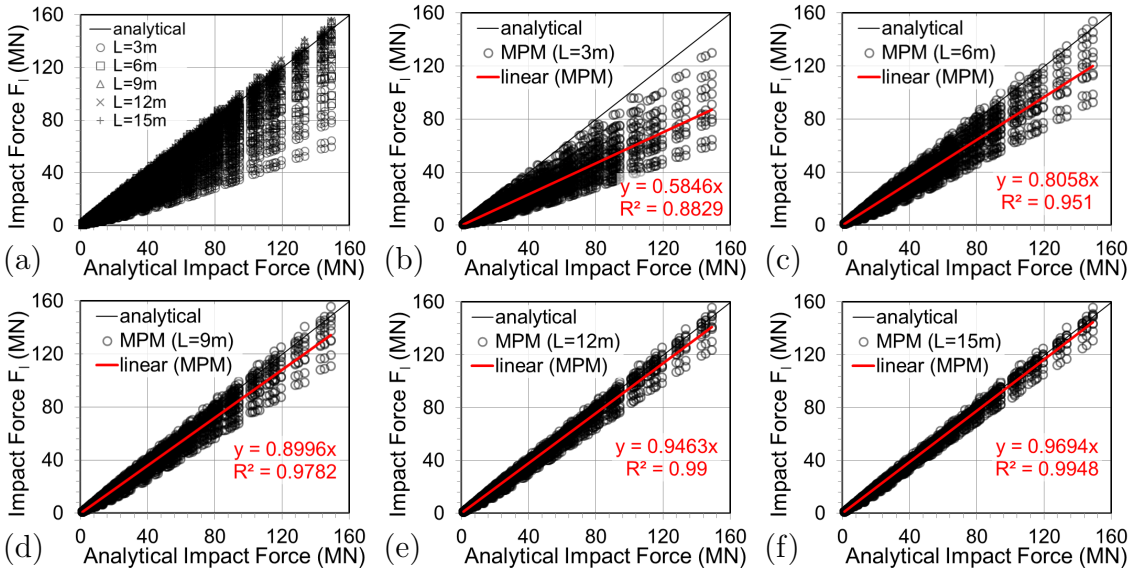


Figure 6.13: Influences of contact area and eccentricities of impact forces on the debris-induced impacts

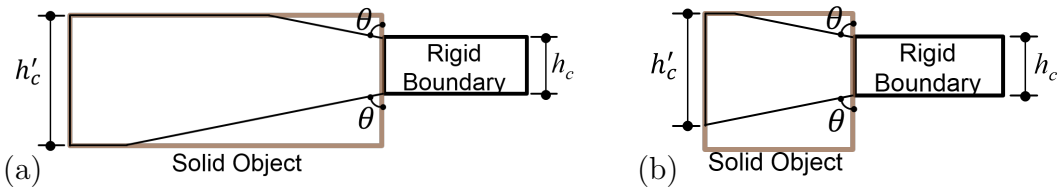


Figure 6.14: Illustration of modified contact height $h_c^{'}$ in the case when debris length is (a) larger or (b) smaller than the transition zone

Figure 6.15 shows this modified equation can capture the impact forces significantly better than Equation (6.9) (Figure 6.13). Furthermore, because the influence of length L and eccentricity e_c on impact force is considered, Equation (6.10) shall be a suitable equation for predicting impact forces introduced by solid objects. However, more study will be necessary to examine the capability of Equation (6.10) to predict impact forces introduced by objects containing spaces or gaps, e.g. shipping container, in which the stress waves have different propagation path.

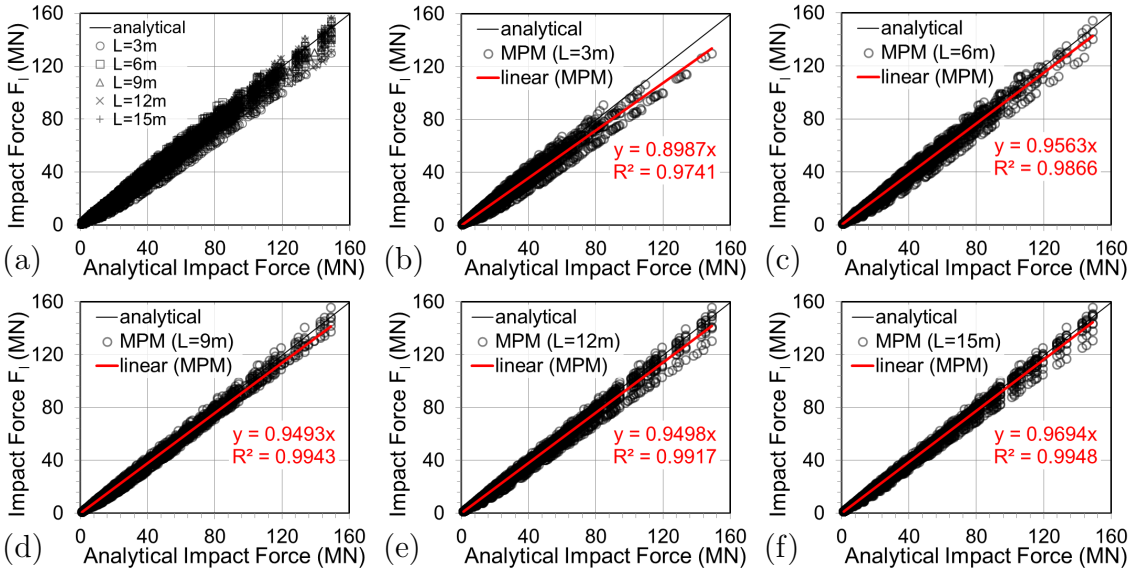


Figure 6.15: Comparisons between MPM simulation results and analytical solutions calculated with Equation (6.10)

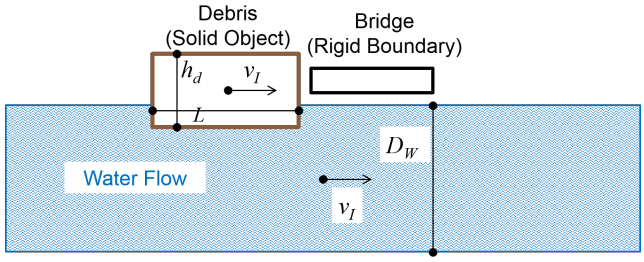


Figure 6.16: Illustration of model setup for studying the contribution of water flow to debris-induced impact forces

6.2.4 Contribution of Water Flow

In this section, the influence of water flow on debris-induced impact forces are discussed with a parameter study using two-dimensional (plane strain) solid blocks. Figure 6.16 illustrates the numerical model used in this study and Table 6.6 lists all variable values under consideration. This model is based on Model I (Figure 6.1(a)) in full scale (i.e. all the dimensions are scaled up with the factor 20) and hence includes

Table 6.6: Variables considered in the study of impact forces due to tsunami-driven debris

Variable	Symbol	Unit	Values
height	h_d	(m)	3.0
thickness	t_d	(m)	1.0
length	L	(m)	3, 6, 9
initial velocity	v_I	($^{\mathrm{m}}/_{\mathrm{s}}$)	3, 6, 9, 12, 15
mass density	ρ_d	($^{\mathrm{kg}}/_{\mathrm{m}^3}$)	200, 400, 600, 800
Young's modulus	E_d	(GPa)	10, 5, 3, 1, 0.5, 0.3, 0.1
Poisson's ratio	ν_d		0.1, 0.2, 0.3
water depth	D_w	(m)	4, 6, 8

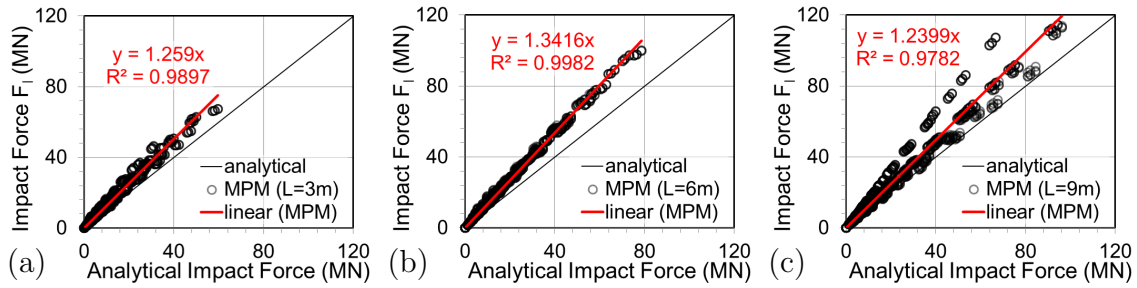


Figure 6.17: Comparisons of tsunami-driven debris-induced impact forces against the analytical solution for in-air cases calculated from Equation (6.10)

a 1 m-by-5 m bridge (modeled with slip wall boundary conditions). The acceleration due to gravity is $g = 9.81\,^{\mathrm{m}^2}/_{\mathrm{sec}}$; The water has Bulk modulus $K_w = 2.2\,\mathrm{GPa}$, mass density $\rho_w = 1000\,^{\mathrm{kg}}/_{\mathrm{m}^3}$, and viscosity $\mu = 0.001\,\mathrm{Pa}\cdot\mathrm{s}$. F0AL1 (defined in Table 3.1) is used for water modeling not only to save computational time but also to keep stress waves caused by impacts. Regular square grids with cell size 0.5 m by 0.5 m, and nine (3 by 3) particles per cell are used. Figure 6.17 indicates the impact forces due to debris (solid objects) in the water flow are about 25 to 35 percent larger than the corresponding impact forces for in-air cases, evaluated with Equation (6.10). The variance shown in Figure 6.17(c) is related to the mass density ρ_d of debris. However,

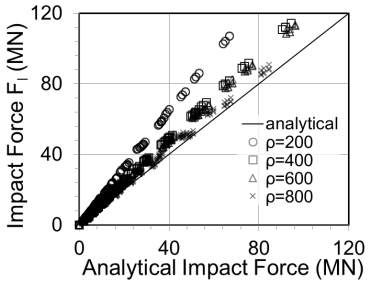


Figure 6.18: Influences of debris mass density on the impact forces for in-water cases ($L = 9\,\mathrm{m}$)

ρ_d in this case is also related to the contact area, eccentricities of impact forces, and wet areas of the debris. More study will be necessary to throughly understand the contribution of water flow to the debris-induced impact forces.

6.3 Summary and Findings from Chapter 6

This chapter studies the influence of tsunami-driven debris on bridge superstructure demands using MPM. Setup of numerical models is based on a reduced domain of a water flume experiment conducted in Japan (Nakao, Zhang, et al., 2013). Major observations about the influence of debris are listed below.

1. When flow is partially blocked by debris, peak horizontal forces are proportional to net cross-section area of the water flow at initial conditions and at the moment when peak forces occur.

2. When debris collides against bridge decks, demand on the bridge increases significantly by a factor no less than 25, compared with the no-debris case.

3. Equation (6.4) is not an appropriate solution of equivalent stiffness for general cases.

4. In plane strain problems considered in this study, using a modified (contact)

height $h_c^{'}$ in the classic equation for impact forces (Equation (6.9)) increases the capability of capturing the effects of contact areas and eccentricities of loading.

5. Tsunami-driven debris-induced impact forces are 25 to 35 percent larger than the corresponding in-air cases.

Additional study is necessary to validate the proposed impact model (as illustrated in Figure 6.14) and understand more about the contribution of water flow to the debris-induced impact forces.

Chapter 7
CONCLUSIONS AND FUTURE WORK

In this research, tsunami-driven-debris-induced loads on bridge superstructures have been studied with the material point method. To this end, a numerical flux smoothing algorithm was proposed to address the over-smoothing problem in the node-based algorithm and hence enhance the capability of modeling fluid with MPM. In addition, an enhanced boundary modeling technique was presented to increase the flexibility of choosing the computational grid geometry used in MPM. A hydrostatic problem (water in a tank) and a complicated hydrodynamic problem (modeling tsunami bores impacting against bridge decks) are used to examine the performance of the newly developed algorithms. With the new algorithms, debris-induced loads, impacts and loads caused by damming effect, on bridges during a tsunami were studied. Conclusions from the work are listed below:

1. With tri-linear shape functions, integration errors arising from the MPM particle forms, especially the nodal internal force term, introduce noise in stress fields and may destabilize the simulations. Particle refinement can reduce the integration errors and hence reduce the effects of destabilizing sources; in contrast, mesh refinement may increase the destabilizing sources with motion of particles and hence not necessarily increases the accuracy of simulation results.

2. The term "anti-locking algorithm" presented by Mast et al. (2012) actually combines the effects of anti-locking (used in FEM) and smoothing applied on material (stress) states. These effects have been separated out in the current work.

3. The cell-based smoothing algorithm (C0) dissipates the least strain energy (compared to the node-based and numerical flux smoothing algorithms) but may be insufficiently dissipative to stabilize the destabilization introduced by the integration errors.

4. The node-based smoothing algorithm (NB) provides a strong stabilization mechanism through strain energy dissipation, but without controls NB may overdissipate the strain energy and introduce evident nonphysical "plastic"-like behaviors in simulations.

5. The constant flux smoothing algorithm (F0) provides a stabilization mechanism similar with NB but uses piece-wise constant shape functions, which decreases the resolution of the stress states but increases the efficiency compared to nodal shape functions. Hence, F0 can be an efficient alternative to NB if strong smoothing is needed for stabilization.

6. The flux (linear, quadratic and oscillation) limiters for F0 provide different levels of flux control, and hence different levels of control of the strain energy dissipation.

7. The constant flux smoothing algorithm with both linear and oscillation limiters (F0AL1) offers the best flux (energy dissipation) control mechanism among other limiter combinations.

8. The linear flux smoothing algorithm (F1) offers higher resolution fluxes and higher order interpolants than F0, but does not provide a suitable selection to stabilize the analyses of MPM using linear shape functions. This is because F1 not only preserves too much strain energy like C0 (and hence offers an insufficient stabilization mechanism), but also keeps more interpolation errors than C0 when particles move across cell boundaries.

9. With its control mechanism of strain energy dissipation, F0AL1 not only can solve hydrostatic problems as well as the cell-based approach, but also is as stable as the node-based algorithm.

10. The approach of using specific force fields for applying boundary conditions decouples the dependency between the boundary geometry and the grid geometry, and, thus, allows using a regular rectangular cuboid grid pattern freely for efficiency.

11. The specific force field (used in the enhanced boundary treatment) includes a constrained zone, in which nodes are constrained by applied boundary conditions; a free zone, in which no constraints are applied on nodes; and a transition zone between the constrained and free zones, or a decay zone, in which the nodal constrained forces are reduced with distance away from the edge of the constrained zone, according to a prescribed decay function.

12. The suggested decay function is a convex function: $\gamma_d(r/h) = (1 - r/h)^2$, which provides enough supporting forces preventing particles from penetrating a prescribed boundary surface (which is not necessarily aligned on the grid lines), and allows particles to move closer to the boundary, compared to other tested functions: $\gamma_d = 0$, $0.5[1 - \cos(\pi r/h)]$, $1 - r/h$, $1 - (r/h)^2$ and 1.

13. Execution time limitations of the self-developed single-threaded program, Continuum Analysis Program, controls the refinement levels and hence only relatively coarse meshes (compared to the applications in the literature) can be used for validations and applications in this research.

14. Simplified inflow and outflow boundary conditions have nonphysical reflection of stress waves, but their influence is minor in the reduced domain model for

the tsunami-bridge interaction modeling, compared to the integration errors of MPM.

15. Deleting the particles behind the outflow boundary may also reduce the effectiveness of the smoothing stabilization (when smoothing algorithms are used) and hence the errors can be more severe around the outflow boundary and in some cases introduce instability.

16. With relatively coarse meshes and simplified flow boundaries, the simulations still match the experimental data to a reasonable degree.

17. The imperfections of boundary conditions in MPM result in down-forces in Case 1 and Case 4 by impeding particles from entering the boundary cells beneath the bridges, and this is a source of buoyancy.

18. Tsunami-driven debris can significantly increase demands on bridges compared with no-debris cases.

19. When flow is partially blocked by debris, peak horizontal forces are proportional in the order of about 3.6 to net cross-section area of the water flow at initial conditions, or A_{if} (Section 6.1.3).

20. Impact forces are proportional to the modified contact area (illustrated in Figure 6.14).

21. Contribution of water flow to the debris-induced impact forces can vary from 25 to 35 percent of those in the corresponding in-air cases.

Based on the lessons (indicated above) learned in this study, recommendations for future research are listed below:

1. A more efficient MPM program is necessary for practical applications. For general cases, parallel computing is a strait forward way to achieve this goal, e.g. (Huang, Zhang, Ma, & Wang, 2008; Dong, Wang, & Randolph, 2015). Although in CAP parallelization is based on CPU, considering most computation occurs independently in particles and nodes, GPU parallel computing for MPM may be also an interesting issue; Implicit methods are also commonly used approaches to improve computational speed, e.g. Sulsky and Kaul (2004).

2. A study on mitigating integration errors would be a critical move to popularize the material point method in more engineering applications.

3. The limiters for the constant flux smoothing algorithm could be improved with the flux limiters used in the control volume method.

4. Smoothed finite element method (S-FEM) is a FEM-based method includes mesh-free technique (Nguyen & Liu, 2010). This method uses a modified strain field from the compatible one, or simply from the assumed displacement field. These features are similar to MPM enhanced with the anti-locking algorithm presented by Mast et al. (2012). Therefore, a branch of S-FEM: a node-based smoothed finite element method (NS-FEM) may shed a light on the improvement of the node-based algorithm (Mast et al., 2012), which gives a continuous stress field.

References

Azadbakht, M., & Yim, S. C. (2015). Simulation and estimation of tsunami loads on bridge superstructures. *Journal of Waterway, Port, Coastal and Ocean Engineering*, *141*(2). doi: 10.1061/(ASCE)WW.1943-5460.0000262

Bardenhagen, S. G., Guilkey, J. E., Roessig, K. M., Brackbill, J. U., Witzel, W. M., & Foster, J. C. (2001). An improved contact algorithm for the material point method and application to stress propagation in granular material. *Computer Modeling in Engineering & Sciences*, *2*, 509–522.

Bardenhagen, S. G., & Kober, E. M. (2004). The generalized interpolation material point method. *Computer Modeling in Engineering & Sciences*, *5*(6), 477–495.

Belytschko, T., Liu, W. K., & Moran, B. (2000). *Nonlinear finite elements for continua and structures*. Chichester: Wiley.

Brackbill, J. U., Brackbill, J., & Repel, H. (1986). FLIP: A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions. *Journal of Computational Physics*, *65*(2), 314–343.

Bricker, J., & Nakayama, A. (2014, feb). Contribution of Trapped Air, Deck Superelevation, and Nearby Structures to Bridge Deck Failure during a Tsunami. *Journal of Hydraulic Engineering*, *140*(5), 5014002. Retrieved from `http://dx.doi.org/10.1061/(ASCE)HY.1943-7900.0000855` doi: 10.1061/(ASCE)HY.1943-7900.0000855

Burgess, D., Sulsky, D., & Brackbill, J. U. (1992). Mass matrix formulation of the FLIP particle-in-cell method. *Journal of Computational Physics*, *103*(1), 1–15. doi: 10.1016/0021-9991(92)90323-Q

Cockburn, B., & Shu, C.-W. (2001). Runge-Kutta Discontinuous Galerkin Methods for Convection-Dominated Problems. *Journal of Scientific Computing*, *16*(3), 173–261. Retrieved from `http://www.worldcat.org/oclc/1990363890893`

Courant, R., Friedrichs, K., & Lewy, H. (1967). On the Partial Difference Equations

of Mathematical Physics. *IBM Journal of Research and Development*, *11*(2), 215–234. doi: 10.1147/rd.112.0215

Dong, Y., Wang, D., & Randolph, M. F. (2015). A GPU parallel computing strategy for the material point method. *Computers and Geotechnics*, *66*, 31–38. doi: 10.1016/j.compgeo.2015.01.009

Hearn, E., & Hearn, E. (1997). Chapter 4 Rings, Discs and Cylinders Subjected to Rotation and Thermal Gradients. In *Mechanics of materials 2* (pp. 117–140). doi: 10.1016/B978-075063266-9/50005-6

Hesthaven, J. S., & Warburton, T. (2008). *Nodal discontinuous Galerkin methods algorithms, analysis, and applications*. New York: Springer. Retrieved from `http://site.ebrary.com/id/10230312`

Hiraishi, T., Haruo, K., & Saitoh, E. (2010, June). Experimental study on impulsive force of drift body due to tsunami flow. *Journal of Earthquake and Tsunami*, *04*(02), 127–133. doi: 10.1142/S1793431110000741

Hu, H.-C. (1954). On some variational principles in the theory of elasticity and the theory of plasticity. *Acta Physica Sinica*, *10*(3), 259–290. Retrieved from `http://wulixb.iphy.ac.cn/CN/Y1954/V10/I3/259`

Hu, W., & Chen, Z. (2003). A multi-mesh MPM for simulating the meshing process of spur gears. *Computers & Structures*, *81*(20), 1991–2002.

Huang, P., Zhang, X., Ma, S., & Huang, X. (2011). Contact algorithms for the material point method in impact and penetration simulation. *International Journal for Numerical Methods in Engineering*, *85*(4), 498–517.

Huang, P., Zhang, X., Ma, S., & Wang, H. (2008). Shared Memory OpenMP Parallelization of Explicit MPM and Its Application to Hypervelocity Impact. *Cmes-Computer Modeling In Engineering & Sciences*, *38*(2), 119–147.

Ko, H., Cox, D., Riggs, H., & Naito, C. (2014, October). Hydraulic Experiments on Impact Forces from Tsunami-Driven Debris. *Journal of Waterway, Port, Coastal, and Ocean Engineering*, 4014043. doi:

10.1061/(ASCE)WW.1943-5460.0000286

LeVeque, R. J. (1992). *Numerical methods for conservation laws*. Basel: Birkhauser Verlag.

LeVeque, R. J. (2002). *Finite volume methods for hyperbolic problems*. Cambridge: Cambridge University Press.

Mast, C. M., Mackenzie-Helnwein, P., Arduino, P., Miller, G. R., & Shin, W. (2012). Mitigating kinematic locking in the material point method. *Journal of Computational Physics*, *231*(16), 5351–5373. doi: 10.1016/j.jcp.2012.04.032

Minimum design loads for buildings and other structures (ASCE/SEI 7-10 ed.). (2013). Reston, VA: American Society of Civil Engineers. doi: 10.1061/9780784412916

Nakao, H., Itonaga, W., Nozaka, K., Izuno, K., & Kobayashi, H. (2013). Hydrodynamic Force on Plate Girder Bridges during Tsunami and Effect of Baffle Plates to Reduce Force. *Journal of Japan Society of Civil Engineers, Ser. A1 (Structural Engineering & Earthquake Engineering (SE/EE))*, *69*(3), 572–585. doi: 10.2208/jscejseee.69.572

Nakao, H., Zhang, G., Sumimura, T., & Hoshikuma, J. (2013). Effect of cross-sectional configuration of superstructure on hydrodynamic behavior of bridges subjected to tsunami-induced loading. *Journal of Japan Society of Civil Engineers, Ser. A1 (Structural Engineering & Earthquake Engineering (SE/EE))*, *69*(4), I_42–I_54. doi: 10.2208/jscejseee.69.I_42

Nguyen, T. T., & Liu, G. R. G.-R. (2010). *Smoothed finite element methods* (T. T. Nguyen, Ed.). Boca Raton: Boca Raton : CRC Press.

Pan, X.-F., Xu, A.-G., Zhang, G.-C., Zhang, P., Zhu, J.-S., Ma, S., & Zhang, X. (2008, May). Three-dimensional multi-mesh material point method for solving collision problems. *Communications in Theoretical Physics*, *49*(5), 1129–1138. doi: 10.1088/0253-6102/49/5/09

Piran Aghl, P., Naito, C., & Riggs, H. (2014, February). Full-Scale Experimental

Study of Impact Demands Resulting from High Mass, Low Velocity Debris. *Journal of Structural Engineering*, *140*(5), 4014006. doi: 10.1061/(ASCE)ST.1943-541X.0000948

Sadeghirad, A., Brannon, R. M., & Burghardt, J. (2011). A 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*(12), 1435–1456.

Sadeghirad, A., Brannon, R. M., & Guilkey, J. E. (2013). Secondorder convected particle domain interpolation (CPDI2) with enrichment for weak discontinuities at material interfaces. *International Journal for Numerical Methods in Engineering*, *95*(11), 928–952. doi: 10.1002/nme.4526

Sulsky, D., & Kaul, A. (2004). Implicit dynamics in the material-point method. *Computer Methods in Applied Mechanics and Engineering*, *193*(12), 1137–1170. doi: 10.1016/j.cma.2003.12.011

Sulsky, D., Zhou, S.-J., & Schreyer, H. L. (1995, May). Application of a particle-in-cell method to solid mechanics. *Computer Physics Communications*, *87*(1-2), 236–252. doi: 10.1016/0010-4655(94)00170-7

Washizu, K. (1982). *Variational methods in elasticity and plasticity* (3rd ed.). Oxford: Pergamon Press.

Zhang, D. Z., Ma, X., & Giguere, P. T. (2011). Material point method enhanced by modified gradient of shape function. *Journal of Computational Physics*, *230*(16), 6379–6398.

Zienkiewicz, O. C., & Taylor, R. L. (2005). *The finite element method for solid and structural mechanics* (6th ed.). Amsterdam; Boston: Elsevier Butterworth-Heinemann.

Appendix A

CONTINUUM ANALYSIS PROGRAM NOTES

Continuum Analysis Program (CAP) [1] is a material-point-method-based program developed with the C++ programming language for this research and hopefully any related studies in the future. In this appendix, instead of going through the details of CAP, notes about possible extensions and improvements of this program or strategies used in the implementation are addressed.

A.1 Parallel Computing

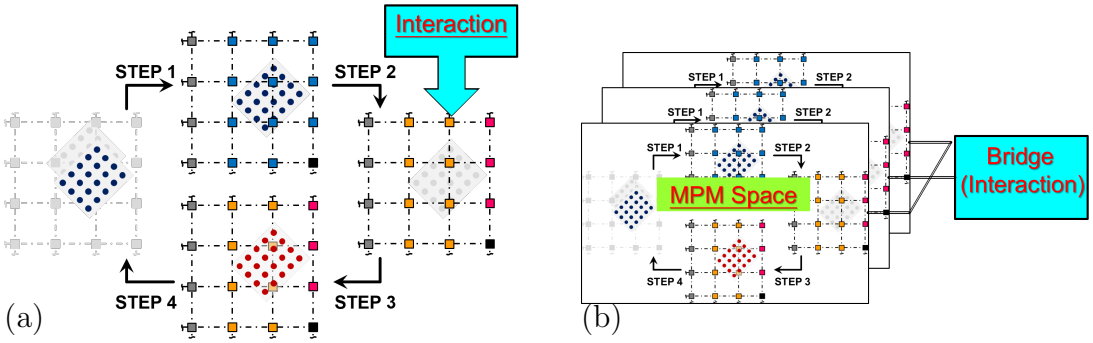


Figure A.1: Illustration of the multi-mesh philosophy from (a) local and (b) global viewpoints

Although CAP is a single-threaded program, extension to parallel computing is considered during its design phase and programming. The fundamental concept of the program design is based on the multi-mesh strategy (Figure A.1) presented by

[1]Continuum Analysis Program is available on https://yangwc@bitbucket.org/yangwc/cap.git.

W. Hu and Chen (2003). This strategy opens a parallel space for each particle-represented object and its own computational grids. Although these grids have the same pattern and hence share the same nodal locations with the others, they cannot see one another unless contact algorithms have been applied on the nodes and bridged these grids. Since in the MPM using a no-slip contact algorithm between partitions of an object gives exactly the same results as simulating the object in whole, threads used in parallel computing can be considered as parallel spaces in the multi-mesh strategy. Therefore the strategy presented by W. Hu and Chen (2003) for contact problems is taken as the framework for parallel computing in the program design of CAP.

A.2 Integration Error

In the program, tri-linear nodal shape functions are used. The nodal shape function N_i at node i is a product of three linear functions:

$$N_i(\xi, \eta, \zeta) = N_{\xi i}(\xi) N_{\eta i}(\eta) N_{\zeta i}(\zeta) \tag{A.1}$$

in which for $\alpha \in \{\xi, \eta, \zeta\}$, $|\alpha| \leq 1$

$$N_{\alpha i}(\alpha) = \frac{1}{2}(1 + \alpha_i \alpha) \tag{A.2}$$

and

$$\frac{\partial N_{\alpha i}}{\partial \alpha} = \frac{\alpha_i}{2} \tag{A.3}$$

Because the differential of the linear shape functions is constant, numerical issues can occur in calculating nodal accelerations due to nodal internal forces when particles are around the edges of cells. Hence, a modification is made to the shape

functions to avoid this numerical issue. The modified shape functions

$$N'_{\alpha i}(\alpha) = \max\left(\min\left(0.001, N_{\alpha i}(\alpha)\right), 0.999\right) \tag{A.4}$$

and

$$\left(\frac{\partial N_{\alpha i}}{\partial \alpha}\right)' = \min\left(\frac{1-|\alpha|}{0.01}, 1\right)\frac{\alpha_i}{2} \tag{A.5}$$

for any $\alpha \in \{\xi, \eta, \zeta\}$, $|\alpha| \leq 1$ are used to slightly reduce the impact of integration errors and hence only applied for finding nodal values, but not for particle updating.

A.3 Numerical Damping

In CAP, the user can add numerical damping (artificial viscosity) in materials without viscosity, e.g. linear elastic material, by giving a positive value of damping ratio ξ to stabilize analyses. The specific stress corresponding to numerical damping, $\bar{\boldsymbol{\sigma}}^{\xi}$, is calculated as

$$\bar{\sigma}_{ij}^{\xi} = \begin{cases} 2ld_{ij}\xi\sqrt{\frac{M}{\rho}} & \text{, if } i = j \\ 2ld_{ij}\xi\sqrt{\frac{G}{\rho}} & \text{, if } i \neq j \end{cases} \tag{A.6}$$

in which M is the bulk modulus; G is the shear modulus; ρ is mass density; l is a characteristic cell dimension and is equal to the average of cell sizes; and d_{ij} is a component of the rate of deformation tensor. In one-dimensional cases, $\xi = 0$ means undamped; $\xi < 1$ means under-damped; $\xi > 1$ means over-damped; and $\xi = 1$ is critically damped.

A.4 Stress State Updating

In the material point method, the rate of deformation $\boldsymbol{d}^{(n)}$:

$$\boldsymbol{d}^{(n)} = \frac{1}{2}\left(\operatorname{grad}\boldsymbol{v}^{(n+1)} + \operatorname{grad}\boldsymbol{v}^{T(n+1)}\right) \tag{A.7}$$

is used for material (stress) state updating at the end of the n^{th} time step. The velocity gradient $\left(\operatorname{grad} \boldsymbol{v}^{(n+1)}\right)$ shown in Equation (A.7) is

$$(\operatorname{grad} \boldsymbol{v})^{(n+1)} = \frac{\partial \boldsymbol{v}^{(n+1)}}{\partial \boldsymbol{x}^{(n+1)}} = \left(\bigtriangledown_{\xi} \boldsymbol{x}^{(n+1)}\right)^{-1} \left(\bigtriangledown_{\xi} \boldsymbol{v}^{(n+1)}\right) \tag{A.8}$$

in which

$$\bigtriangledown_{\xi} (\,\cdot\,) := \frac{\partial (\,\cdot\,)}{\partial \boldsymbol{\xi}} = \sum_{i} \frac{\partial N_i(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}} (\,\cdot\,)_i \tag{A.9}$$

and $\boldsymbol{\xi}$ is the parent element (natural) coordinates. Assuming the deformation in a single time step is small and hence

$$\boldsymbol{x}_i^{(n+1)} \approx \boldsymbol{x}_i^{(n)} \tag{A.10}$$

Equation (A.8) can be approximated as

$$\boldsymbol{x}_i^{(n+1)} \approx \left(\bigtriangledown_{\xi} \boldsymbol{x}^{(n)}\right)^{-1} \left(\bigtriangledown_{\xi} \boldsymbol{v}^{(n+1)}\right) \tag{A.11}$$

If the shape of the cells at the beginning of each time step is selected as a rectangular cuboid, $\bigtriangledown_{\xi} \boldsymbol{x}^{(n)}$ is a diagonal second order tensor and hence is easier to determine its inverse. Consequently, using Equation (A.11) saves computational cost. In the next paragraph, the difference between using Equation (A.8) and Equation (A.11) is examined using a spinning cylinder (2D strain) example, an equilibrium problem with non-zero stress updating.

Consider a slice of a spinning cylinder, which has Young's Modulus $E = 1\,\text{MPa}$, Poisson ratio $\nu = 0.25$, mass density $\rho = 3000\,{}^{\text{kg}}/_{\text{m}^3}$ and radius $r = 20\,\text{cm}$. 6910 particles discretize the circular cross-sectional area of the cylinder, and have initial angular velocity $\omega = \pi\,{}^{\text{rad.}}/_{\text{s}}$ and the corresponding equilibrium stresses (Hearn & Hearn, 1997). Cells are 12.5 mm by 12.5 mm squares and have area about nine times the represented domain of a particle. The simulations stop at 4 seconds, two

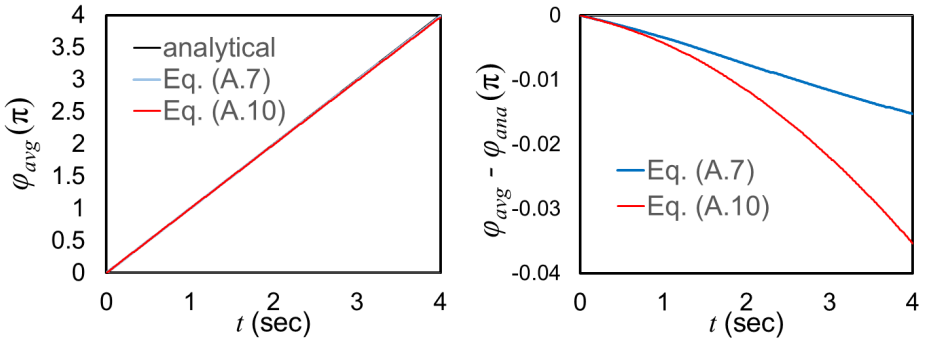


Figure A.2: Comparison of averaged angle of rotation ϕ_{avg} of a spinning cylinder between stress state updating using deformed (Equation (A.8)) and undeformed (Equation (A.11)) nodal coordinates in velocity gradient

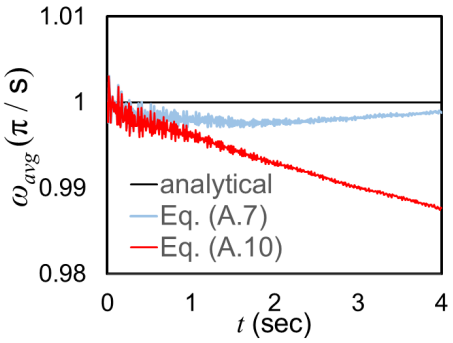


Figure A.3: Comparison of averaged angular velocity ω_{avg} of a spinning cylinder between stress state updating using deformed (Equation (A.8)) and undeformed (Equation (A.11)) nodal coordinates in velocity gradient

rotating cycles in theory, which is considered as the most rotation to happen in the applications of this research. As shown in Figure A.2, both Equations (A.8) and (A.11) allow the cylinder rotate as expected, and the difference between the theoretical solution and the simulation results of the averaged angle of rotation ϕ_{avg} at the end of two rotation cycles is less than 0.04 π (1 % error). The error increase rate of ϕ_{avg} in the simulation results using Equation (A.11) seems quadratic, which is one order higher than using Equation (A.8). However, both of them give

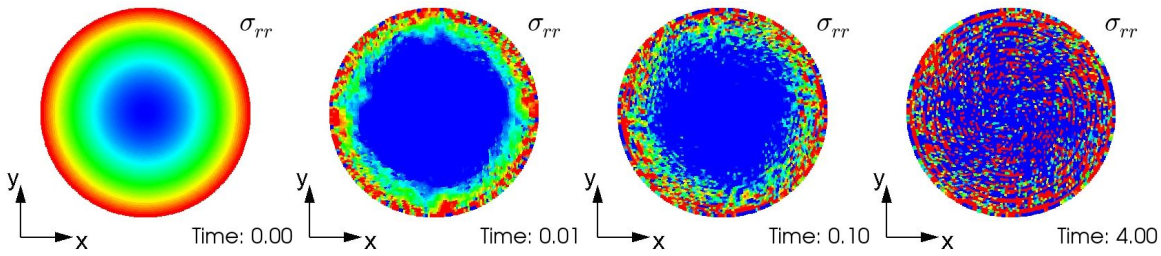


Figure A.4: Snapshots of the radial stresses in the spinning cylinder using deformed (Equation (A.8)) nodal coordinates in velocity strain for stress updating

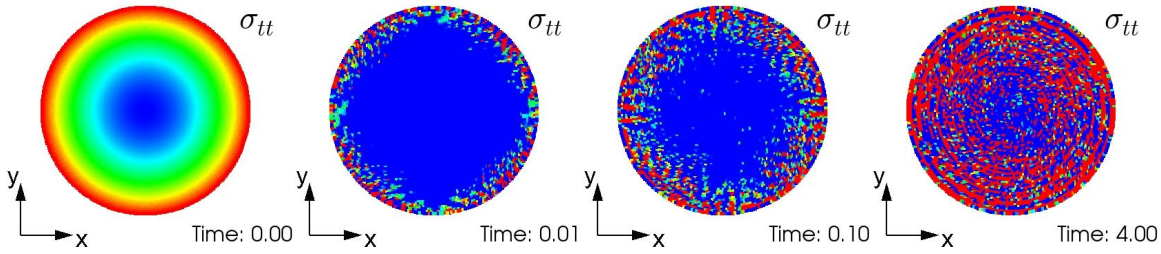


Figure A.5: Snapshots of the hoop stresses in the spinning cylinder using deformed (Equation (A.8)) nodal coordinates in velocity strain for stress updating

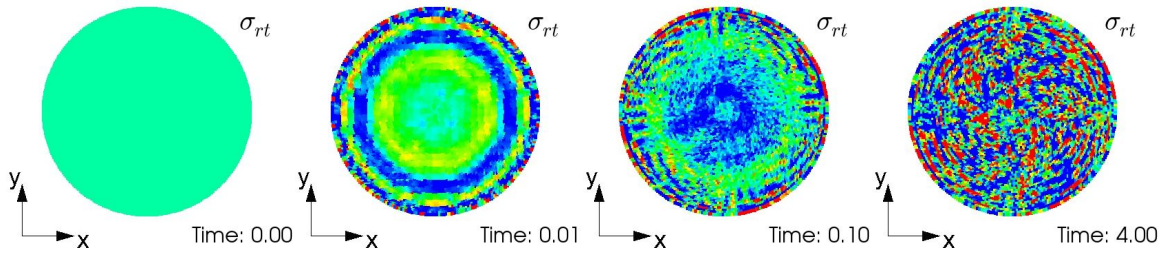


Figure A.6: Snapshots of the shear stresses in the spinning cylinder using deformed (Equation (A.8)) nodal coordinates in velocity strain for stress updating

reasonable behavior in a considered time range and hence can be used in this research. Figure A.3 compares the averaged angular velocities ω_{avg} of the spinning cylinder between Equations (A.8) and (A.11). ω_{avg} is decelerated linearly when

Equation (A.11) is applied, but decelerated from the beginning and accelerated after the first rotating period when Equation (A.8) is used. This acceleration probably is caused by the large errors in the stress shown in Figures A.4 to A.6. Again, as observed in Section 3.6.2 the computational grids seem to offer some filtering that the MPM analysis gives reasonable kinetic behavior even with severe noise in the stresses.