Evaluation of Fluid-Driven Debris Impacts in a High-Performance Multi-GPU Material Point Method

Justin Bonus

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

Doctor of Philosophy

University of Washington

2023

Reading Committee:
Pedro Arduino, Chair
Marc Eberhard
Michael Motley

Program Authorized to Offer Degree:
Civil and Environmental Engineering

University of Washington

Abstract

Evaluation of Fluid-Driven Debris Impacts in a High-Performance Multi-GPU Material Point Method

Justin Bonus

Chair of the Supervisory Committee:

Pedro Arduino

Civil And Environmental Engineering

Tsunamis, storm surges, landslides, lahars, and avalanches driving groups of debris (e.g. cars, trees, boats, collapsed structures) are complex coinciding hazards. Yet, in recent 3D animated films these events are rendered to an incredible visual benchmark at computational scales exceeding what most engineers can match. Disney's Frozen and Moana, in an effort to simulate snow and sand media (history-dependent, topology changing, large-deformation, and nonlinear), champions an interesting although expensive numerical tool: The Material Point Method (MPM). Their software optimization bypassed a computational barrier that had limited engineering use. Just as animators adopt engineering techniques to improve physical accuracy, we argue engineers should adopt the optimized codes of computer graphics professionals. Doing so can accelerate engineering simulations 10x - 100x, grow them by 2x - 1000x, and amplify their public impact. We introduce innovations by Disney, Pixar, Tencent, DreamWorks, and their research contemporaries into our own project with minimal overhead. Our open-source numerical tool, a high-performance multi-physics coupling of the MPM and finite elements (FEA), scales across

Multi-GPUs, devices typically used for video games and machine learning. In doing so, we broach the doorstep of exa-scale computation– Where billion particle simulations are common. Prioritizing flexibility, speed, and simplicity lead us to the most basic algorithm approach for debris-fluid-structure interaction (DFSI) identified yet, which we show to be a novel solution to fluid-driven debris-field loads on structures. This work develops an augmented MPM framework suitable for scaling on Multi-GPU infrastructure and designed for natural hazard simulations. We validate against analytical cases, alternative numerical methods, and experimental results. Five digital twins are developed for wave and lahar flumes located at USGS sites, Oregon State University, Waseda University, and the University of Washington. Structural loads from organized and random groups of debris at multiple scales are numerically replicated with novel accuracy. Preliminary steps are taken into a performance-based engineering framework and a generalized debris-field characterization scheme for multi-hazard events.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Fig. 1.1. Tsunami-driven debris-field impacts on Ofunato, Japan. Debris-field impact on a street in Ofunato, Japan, as a result of the Tohoku 2011 earthquake and tsunami. **(Left)** A street in Ofunato following post-disaster debris-field clean-up, taken January 15, 2012. **(Right)** The same street 3-days after the tsunami, taken March 14, 2011. Photographs courtesy of Toshifumi Kitamura / AFP.

Chapter 1

INTRODUCTION

1.1 *Structures, Tsunamis, and Debris*

Helping mitigate risk and uncertainty to life and property is the primary objective of this study. Eighty percent of people live along bodies of water, as does most infrastructure. A subset of these structures are exposed to natural hazards arising from local water bodies (e.g. tsunamis, floods). Natural hazards possess risk and risk uncertainty, which structures inherit. Occupants of these structures inherit risk to life, owners risk to property.

Secondary hazards (e.g. debris carried by fluid impacting structures) amplify risk but are not well quantified. Complex cases (e.g. debris-fields, a collection of potentially random debris) even less so. Efforts have been made to alleviate this, but observed chaotic variability is an obstacle to practical frameworks of understanding (i.e. small changes in initial conditions produce large changes in results). This work seeks to improve the state of the art for debris-field design.

Interactions of debris, fluids, and structures are nonlinear and chaotic. Chaotic, nonlinear phenomena often need a stochastic model to design with, which requires extensive, diverse data to create. Infrequency of these events, and the perishable nature of data produced, makes needed studies difficult and as such few have been performed.

Scaled experiments can help to fill data gaps, as fluid-driven debris-field impacts (i.e. debris-fluid-structure interaction) can be produced in lab environments. These may extrapolate on aspects of the physical behavior, but introduce limits, errors, and uncertainty of their own. Approximating high-energy fluid scenarios (e.g. tsunamis on coastlines) in a scaled facility (e.g. man-made wave or steady-state flow in a small flume) is tenuous. This is especially true for scaled debris tests because transport, impact, and damming behavior is sensitive to small changes in fluid and debris behavior deriving from scale.

Numerical models bypass many physical limits (e.g. scale, cost, etc.). A numerical model is simply a mathematical solution to material laws (e.g. water, concrete, etc.) within a chosen scenario (e.g. initial distribution of material, topography, and boundaries) over a time interval.

Studying the differential equations that define these chaotic, nonlinear events shines a light on their behavior. This assumes the chosen equations are accurate descriptors of reality and are solved with appropriate boundary conditions (BCs), initial conditions (ICs), numerical methods, and computational resolutions.

Multi-material, multi-phase interaction adds complexity to the problem and is the true subject of study. The physical phenomena, debris-fluid-structure interaction (DFSI), is a composition of subset interactions: **(i)** Debris-Structure, **(ii)** Fluid-Structure, **(iii)** Debris-Fluid, and **(iv)** Debris-Debris.

Each subset is a full-fledged field of study. Our work must maintain the basic standards of all while proposing new frameworks for a composite field. In this work, we aim to develop a numerical tool capable of replicating modes of behavior for fluid-driven debris-field hazards and using this tool to explore possible answer for the issues listed above.

1.2 *Gaps in Previous Works*

The best available science of the time has led to recommendations regarding debris impacts and damming that are included in the ASCE 7-16 building code (Chock 2016). It is communally accepted that there is room for improvement (Nistor et al. 2017a), which multiple groups are pursuing.

The theory behind some force recommendations on structures due to impact of debris are rooted in one-dimensional, idealized elastic collisions (Aghl et al. 2014a). These equations do not fully consider the influence of a driving and/or arresting fluid, nor do they truly approach the nonlinear dynamics of impact with complex debris. They are not fully appropriate for a single debris colliding with a structure, and this error compounds as more debris (i.e. debris-fields) are considered in advanced scenarios (e.g. tailing dam collapse, landslides, unusual inundation events).

In the case of debris-fields, distributed collections of interacting debris, engineers are forced to assume the whole debris-field effect is equal to the sum of its individual, near deterministic parts. This is not observably accurate in a field where behavior is chaotic. Debris-fields are not given credit as their own entity for abstracted analysis in design guidelines. The seemingly chaotic, yet potentially stochastically reproducible, nature of these fields are not truly understood. The literature lacks a means for effective characterization of debris-fields, and further it lacks a means to quantify the demand they exert on structures in meaningful terms.

Numerical studies done insofar, e.g. Yang 2016 and Hasanpour et al. 2021 who studied single debris impacts in the Material Point Method (MPM) and Smoothed Particle Hydrodynamics (SPH), were strong steps in the right direction but they have notable computational limits that are not applicable to ordered and randomized fields of debris. The problem of debris-fluid-structure interaction (DFSI) is exceedingly difficult. Most take the approach of co-opting fluid-structure interaction (FSI) focused codes, adding in debris as rigid, over-simplified geometries, but they fall short of properly modeling DFSI. Alternative tools, such as the Material Point Method, may better capture DFSI but are extremely computationally intensive. These advanced 3D approaches have the potential to produce compelling results but are unfortunately limited by resolution and domain size limits of the simulations.

In this document we make an argument that the Material Point Method (MPM) is a strong, high-performance approach to DFSI requiring only minor changes to the basic MPM formulation and numerical approach, but major changes to the hardware used to execute it. Claims are supported by order of magnitude improved speeds and size in our simulations.

1.3 *Material Point Method*

The problem of debris-fluid-structure interaction (DFSI), which is critical to fluid-driven debris-field hazards, is a demanding one. Many have attempted it but results are often found

wanting. In terms of physics it is complex: multi-phase, multi-material, and governed by large-deformation dynamics. A numerical method well-suited to this problem exists, only avoided in the past due to a perceived unreasonable computational demand– the Material Point Method (MPM, Sulsky et al. 1994). As we enter a new era of engineering simulations in terms of abstracted software, massively parallel hardware, and non-local computation and data-storage, previously shirked methods are finding critical value. The Material Point Method marks a simple but powerful approach. MPM is favorable to: (i) Large deformations, (ii) Complex multi-material, multi-phase interaction, and (iii) Multi-material, multi-phase boundary conditions. This dissertation shows that engineers may begin to make full use of MPM's potential through an unusual avenue– adoption of innovations from the field of computer graphics.

1.4 *Document Structure*

This dissertation is organized into eleven chapters, each addressing an important component of this project. They include:

Fundamentals of the Material Point Method. A proper study and extension using MPM requires an engineering description and derivation of the Material Point Method (MPM). This is done in Chapter 2, where we explain our reasoning for choosing this method, as well as the reasoning for avoiding others. Derivation of the traditional MPM (Sulsky et al. 1994) is near-identical to previous descriptions presented in documents by former PhD students from the University of Washington (Mast 2013; Yang 2016; Shin 2009). In this study we follow similar descriptions to maintain nomenclature consistency.

Further, we derive modern numerical improvements in the methodology, such as Affine Particle-in-Cell (APIC, Section 2.4) and Moving Least Squares Material Point Method (MLS-MPM, Section 2.5). These enhancements source from graphics literature, reviewed and rephrased here for engineering applications using our lab's established nomenclature. Innovations

such as APIC and MLS-MPM provide benefits ranging from improved angular momentum conservation to a 2x acceleration of simulation times.

Accelerator Hardware for Engineering Simulations. The Material Point Method (MPM), our chosen tool, faces the hurdle of practicality– No one can run it well on their desktop computer. Despite aspirations for advanced multi-phase, multi-material, large-deformation simulations (precisely what we need) computational times seem insurmountable. No algorithmic modification to MPM has fixed this issue, though advancements like the Moving-Least-Squares Material Point Method (MLS-MPM) do improve run-times. Simulation times must be accelerated by *at least* an order of magnitude.

We propose the way to make MPM usable is by switching the hardware infrastructure it is ran on. Graphics Processing Units, devices often used for video-games, may be the key to this engineering problem. An introduction to GPU hardware and software is described in Chapter 3.

Bringing the Material Point Method to Multi-GPU. In Chapter 4 we show the basic steps to make the Material Point Method (MPM) algorithmically manifest across a single or multiple graphics processing units (GPUs). Each device holds thousands of parallel computational cores that may accelerate the numerical method by over 100x if properly employed.

An argument is made for engineers to leverage open-source software and resources from other fields instead of starting from scratch. Claymore (Wang et al. 2020b) is a graphics-oriented software for Multi-GPU MPM and our chose foundation for our open-source project, ClaymoreUW. Claymore is surveyed in Chapter 4 in terms of its design and usage under an engineering-centric lens. To increase accessibility and viability for engineers, we rebuild many aspects of the project to create our own rendition which we call ClaymoreUW.

We cover novel modifications our lab has made to the Claymore software, including the coupling of finite elements, implementation of various advection methods, enhancing computational precision, and a complete overhaul to the user-interface. A more exhaustive list and description of

improvements and additions to Claymore made in development of ClaymoreUW are included in Chapter 4. Documentation, for both prospective users and developers, is provided for our ClaymoreUW software within Chapter 4 as well.

Design and Implementation of Numerical Methods for Debris-Fluid-Structure Interaction in a Multi-GPU Material Point Method. In Chapter 5 we develop a novel modelling approach to tackle fluid-driven debris-field hazards. Our numerical tool, a multi-physics coupling of material point method (MPM) (Sulsky et al. 1993) and finite elements (FEM, Section 5.13), scales across Multi-GPUs in high-performance computing centers (HPCs), diluting computational weight. Prioritizing flexibility, speed, and simplicity leads to the most basic algorithm approach for DFSI identified yet. We describe our novel coupling of Moving-Least-Squares Material Point Method, Finite Elements, Affine-Separable Fluid-Implicit Particles (ASFLIP, Section 5.6.7), and our novel, modified B-Spline F-Bar volumetric antilocking on Multi-GPU systems and the benefits provided. Further, we formulate a novel, high-performance material model called the PA-JB Fluid in Section 5.12 specifically optimizing for Multi-GPU usage.

Validating High-Performance Multi-GPU Material Point Method for Debris-Fluid-Structure Interaction. Our novel numerical approach requires validation. We do so by isolating physical phenomena of concern into characteristic scenes for simulation. Results from our software are then compared against analytical results and alternative approaches. Numerical results are solutions to the underlying partial differential equations of study. Controlling for boundary and initial conditions, results should be equivalent to those found in reality and other validated numerical codes.

Validation of debris-fluid-structure interaction (DFSI) requires **(i)** debris-structure interaction (DSI), **(ii)** debris-fluid interaction (DFI), **(iii)** fluid-structure interaction (FSI), and **(iv)** debris-debris interaction (DDI). Examples for each validation subset are presented in Chapter 6.

Debris-Field Impact Experiments and Simulations - Oregon State University's Large Wave

Flume. In Chapter 7 we demonstrate the integration of advanced computational techniques with coastal engineering through the application of a Multi-GPU Material Point Method, implemented in the open-source ClaymoreUW software. We successfully simulate wave-driven debris field dynamics in a digital twin of the Oregon State University's Large Wave Flume (OSU LWF). The results presented are compelling and highlight the potential of high-performance computing in advancing the understanding of complex, real-world engineering problems.

The physical specification and numerical replication of experiments performed at Oregon State University Large Wave Flume (OSU LWF) are described. Facilities are recreated in our numerical open-source MPM software, at a technologically novel level using Multi-GPUs to simulate the flume with 75 - 150 Million particles 50 times. Further, exa-scale capabilities for wave-flume simulations are demonstrated by modeling the OSU LWF digital twin with 1 billion particles. Numerical simulations are characterized and preliminary data-analysis of results are given with comparison to real-world results for validation of our approach in both ordered and randomized debris-fields. Preliminary extrapolation to the prototype scale through similitude scaling suggest a significant importance in the quantification of debris-field impact force relative to wave-only force. Our approach has shown an ability to replicate tests, predict tests not yet performed within expected results bounds, and may be of value in the prediction of prototype scale debris-field impacts in the real-world for tsunami events.

Motion of Tsunami Debris Through a Model Port City - Three Numerical Methods with Experiments - Waseda University Flume. In Chapter 8 a set of wave-flume experiments performed at Waseda University (Goseberg et al. 2016) are replicated with and compared between three advanced numerical methods: Multi-GPU Material Point Method (ClaymoreUW), GPU Smoothed Particle Hydrodynamics (DualSPHysics), and computational fluid dynamics (STAR-CCM+). This endeavor sheds light on the strengths and limitations of these numerical techniques in simulating debris motion in tsunami-like conditions. All three methodologies

demonstrated their unique capacities to accurately replicate complex fluid-debris-obstacle interactions in a controlled, yet highly representative, harbor environment.

Our findings suggest that while the Multi-GPU Material Point Method and GPU Smoothed Particle Hydrodynamics perform exceptionally well in capturing the debris longitudinal and lateral spreading, STAR-CCM+ has shown greater precision in estimating wave-gauge elevations respective to the pressure release system employed at Waseda University's tsunami wave basin. It is important to note, however, that the effectiveness of these methodologies can be scenario-dependent, thus emphasizing the importance of employing a combination of these techniques for comprehensive analysis.

While obstacles were found to have significant influence on the maximum longitudinal displacement of debris, the impact on the spreading angle was found to be less pronounced across all three methods. This study not only offers a critical foundation for further research in optimizing numerical models for tsunami debris motion but also paves the way for the development of more robust, efficient, and reliable disaster management strategies in the face of increasing threats from extreme maritime conditions. As we continue refining and improving these numerical methodologies, it is our hope that they will enable us to better protect coastal communities worldwide.

Multi-Hazard Simulation of Debris Driven by Flows of Water, Snow, Mud, and Soil Generated by Pumps, Pistons, or Gravity using the Material Point Method. In addition to the validation studies and applications to debris fields presented in Chapters 6, 7, and 8, Chapter 9 presents three additional comparative studies using our numerical method versus experimental results across different debris field hazard manifestations.

Each section in Chapter 9 covers a pilot study investigating the capabilities of our numerical method to generalize across new axis of critical natural hazard behavior. While not developed enough to justify individual chapters, we produce digital twins of three experiment sets at three

unique, scaled flume facilities, including: **(i)** Oregon State University's directional wave basin (OSU DWB), **(ii)** University of Washington's Wind-Air Sea Interaction Research Facility (UW WASIRF), and **(iii)** the U.S. Geological Survey's Volcanic Hazard Debris-Flow Flume (USGS Lahar flume).

Debris-Field Characterization. Chapter 10 reveals debris-fields as quantitative bodies using a novel characterization scheme. By exploring characteristics that describe debris-fields (e.g. debris count, occupation, orientation) we find independent variables for analysis. By considering basic needs of engineers designing for debris-field impacts we find dependent variables (e.g. impulse, max force, impact duration). Statistical analysis reveal relationships between knowable independent variables and desired dependent variables in the context of an in-air numerical study. These insights can be used to develop a novel but preliminary framework of probabilistic design, presented in Chapter 11.

Performance-Based Framework. Informed by experimental and numerical results, we take initial steps into a novel performance-based engineering framework of design for tsunami-driven debris-field hazards where a new characterization scheme is proposed. Limitations and lines of future research are discussed in Chapter 11

Conclusions and Future Work. Chapter 12 serves as a summary of work completed in the past three years and work to be done. To tie the knot on a novel, self-consistent dissertation the primary novel contributions are listed. Further, remaining lines of promising research are encapsulated to help stage future efforts in the study of tsunami debris and debris-fluid-structure interaction.

1.5 *Novel Contributions*

A shortlist of novel contributions by this dissertation are as follow:

1. Development of a novel multi-physics numerical platform on the frontier of hardware infrastructure (i.e. Multi-GPU supercomputers) for tsunami-driven debris-fields. More

broadly, this a numerical approach to debris-fluid-structure interaction (DFSI) problems suited to modern computational demands.

2. Publishing our high-performance, Multi-GPU MPM-FEA code-base as an open-source project, ClaymoreUW, with user and developer documentation. This builds off of highly optimized graphics software that has been retooled for engineers.

3. Validation of our novel numerical approach to DFSI across a range of demanding cases. Strengths and limitations of our approach are addressed.

4. Stochastic model of fluid-driven debris-field impacts based on experimental and numerical wave-flume tests at multiple wave-flumes.

5. Proposal of a novel characterization scheme for debris-fields at all scales. Establishment of preliminary regression models with statistical significance using simulated in-air and in-water cases.

6. Phrasing of debris-field impact hazards in a novel Performance-Based Engineering context.

Chapter 2

FUNDAMENTALS OF THE MATERIAL POINT METHOD

2.1 *Introduction*

This chapter provides an engineering description and derivation of the Material Point Method (MPM). We explain our reasoning for choosing this method and our reasoning for avoiding others. The derivation of traditional MPM presented here is similar to previous descriptions used by former PhD students at the University of Washington (Carter Mast, Wen-Chia Yang, and Wookuen Shin) (Mast 2013) (Yang 2016) (Shin 2009) to maintain nomenclature consistency.
Further, we derive modern numerical improvements in the methodology, such as the Affine Particle-in-Cell (APIC, Section 2.4) and Moving Least Squares Material Point Method (MLS-MPM, Section 2.5). These enhancements source from graphics literature, reviewed and rephrased here for engineering applications using our lab's established nomenclature.

2.2 *Why the Material Point Method*

For the problem of large-deformation debris-fluid-structure interaction (DFSI), found in fluid-driven debris-fields (e.g. tsunamis, landslides), the shortlist benefits of MPM appear ideal. In theory, it is, although limitations are found. Some of these are alleviated algorithmically (discussed later in this chapter, Sections 2.4 and 2.5) while others are a property of the hardware that computation is performed on (Chapter 3).
Briefly we will mention common alternatives to MPM to highlight why they were not pursued. Table 2.1 qualitatively compares various aspects of numerical methods that are pertinent to our

Table 2.1. MPM vs. Other Methods. Comparison of common numerical methods in relevant aspects. Qualitative review. Well-suited ✓. Feasible ✓. Poorly suited X.

Method / Attribute	MPM	FEM	DEM	SPH	CFD
Large Deformation	✓	X	✓	✓	✓
Multi-Material	✓	✓	✓	✓	X
Multi-Phase	✓	X	X	✓	X
Complex Materials	✓	✓	✓	✓	X
Fluids	✓	✓	X	✓	✓
Solids	✓	✓	✓	✓	X
Boundary Conditions	✓	✓	✓	✓	✓
Initial Conditions	✓	✓	✓	✓	✓

project.

2.3 *Material Point Method Derivation*

Loosely, a Material Point Method cycle can be described as follows. Space is discretized by a grid, where grid-nodes hold momentum. This introduces an Eulerian scheme. Material bodies are discretized by points (interchangeably referred to as particles), which hold positions and deformation gradients, as well as material parameters and velocities when necessary. This introduces a Lagrangian scheme. The equation of motion and boundary conditions are solved on the grid nodes. Material laws and advection are carried out on particles. Results of these computations pass between the particles and grid through a transfer-scheme (via shape-functions). MPM is thereby defined as a hybrid Euler-Lagrangian numerical method.

In this context, the Material Point Method (MPM) is a numerical technique that is best suited for modeling history dependent materials in a dynamic, large deformation setting. The formulation tracks moving points relative to stationary nodes, and can be used to capture the behavior of both fluids and solids in a unified framework. The standard, or traditional[1], implementation solves the

[1]The adjectives *standard* and *traditional* will be used interchangeably and represent the original framework presented by (Sulsky et al. 1993)

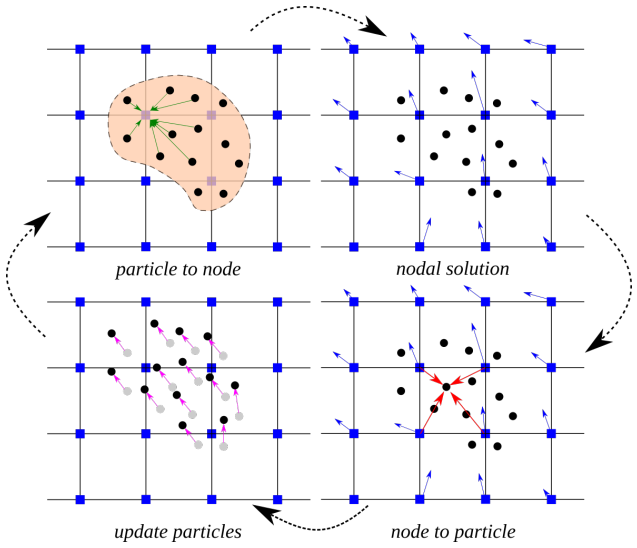

Fig. 2.1. Material Point Method as a flowchart. Basic MPM algorithm visualization.

governing equation of motion at fixed nodes that collectively form a grid. Each body or phase in the analysis is represented by a collection of discrete points known as *material points* or *particles*. This general concept is shown in the upper portion of Figure 2.1. Here the different components that make up an MPM simulation are classified as either Body/Phase-Based or Domain/Grid-Based, and properly understanding the role and interaction of these two categories will prove beneficial in later chapters.

The Body/Phase-Based group is comprised of the continuum body itself and the computational points that, collectively, describe the object. Each particle represents a portion of the total mass, and thus caries an implied volume as well as various state variables depending on the application. For example, in solid mechanics each material point is assigned initial values for position, velocity, stress, strain, and any other state variable needed for the constitutive relationship. That being the case, these particles form a Lagrangian frame of reference from which the state of the body is determined at any instant in time. A crucial and fundamental characteristic of the Material Point Method is the following: these objects serve only as integration points for the governing

equation in space (more on this shortly). As such, these entities are not physical particles, per se, and casting the MPM as a particle-based method is arguably incorrect or, at a minimum, misleading. This problem is compounded by two additional factors. The first is visualization, where viewing dynamic MPM simulations as moving material points is questionable if the true nature of the computational particles is not understood. Even simple effects like adjusting the plotting point size can severely distort the interpretation of what a material point is, can give false impressions of the implicit volume being occupied by the point, and lead to erroneous conclusions regarding the particle nature of the method. The second factor is the application of the method to discretely-based matter, such as granular materials, where the very nature of the medium lends itself to a particle description. In these applications the computational particles serve as integration points in space that contain the state of the continuum representation at that particular location, and not the individual constituent grains that make up the material.

The second category of objects, the Domain/Grid-Based entities, are responsible for defining the physical space a body moves in. The primary object is a node, and a collection of nodes forms a grid. Typically nodes are arranged in a repeating and regular pattern, forming a line in 1-dimensional (1D) applications, a rectangular pattern in 2D simulations, or a rectangular cuboid in fully 3D environments. This repeatable structure is not a requirement of the method but is the most common scheme to date. The nodal arrangement also defines cells, or the region contained between adjacent nodes, as well as nodal supports, defined by piece-wise continuous shape functions residing at each node in the domain. The interplay between the grid, nodes, cells, and nodal support (assuming linear shape functions, which will not be true later on) is shown in Figure 2.1. Strictly speaking the nodal positions are arbitrary and can potentially change without penalty at any point in time. However, nodes are most commonly assumed to reside in a single location effectively creating a static grid. This facilitates an Eulerian frame of reference when viewed relative to the particle motion. The sharing of information between particles and nodes is

governed solely by the shape function that serve as an effective weight for determining the importance of a given particle to any node in the domain. In general this process is referred to as mapping, and can occur from particle-to-node, or in the opposite direction, from node-to-particle. The primary goal of any analyses is to track the system in time while monitoring the evolution of key quantities in both the Body and Domain categories. This is accomplished by splitting a finite time increment into many smaller time intervals, $\Delta t = t_{n+1} - t_n$, and approximating key equations over each Δt. When the governing equation is conservation of linear momentum, material point quantities of mass, momentum, and force are mapped to the appropriate nodes as indicated in Figure 2.1(a). After collecting contributions from all particles in the support, the nodal acceleration and velocity vectors are determined over Δt as observed in Figure 2.1(b). The velocity gradient and the corresponding strain increment are mapped to the particle location using the updated nodal velocity. The particle stress and material state variables are computed from the desired constitutive model as part of the third step highlighted in Figure 2.1(c). Finally, the incremental changes in nodal velocity and position are mapped from the nodes to the particles, resulting in a fully updated system at the particle level. After Figure 2.1(d) the procedure begins again and the computational cycle is repeated for a prescribed time duration.

2.3.1 Traditional Implementation

The traditional approach (Sulsky et al. 1993) is built around conservation of linear momentum, which when expressed in differential form appears as follows:

$$\rho\, \dot{\mathbf{v}} = \operatorname{div} \boldsymbol{\sigma} + \mathbf{b} \ , \tag{2.1}$$

with the mass density $\rho(\mathbf{x}, t)$ at position $\mathbf{x}$ and time t, $\dot{\mathbf{v}}(\mathbf{x}, t)$ as the material time derivative of the velocity field—also known as the acceleration field. Stress divergence $\operatorname{div} \boldsymbol{\sigma} = \boldsymbol{\nabla} \cdot \boldsymbol{\sigma}$, where $\boldsymbol{\nabla}$ the

gradient operator, $\boldsymbol{\sigma}(\mathbf{x}, t)$ is the Cauchy stress tensor. $\mathbf{b}(\mathbf{x}, t)$ is the body force per unit volume. In the present derivation the end goal is to obtain an expression for $\dot{\mathbf{v}}(\mathbf{x}, t)$ consistent with the description of the MPM given in the previous section. Thus, it is necessary to build an approximation for the acceleration field in terms of the nodes and particles that make up a given analysis. For this purpose a weighted integral statement is constructed from 2.1 as

$$\int_{V_{\mathcal{B}}} (\rho\, \dot{\mathbf{v}} - \operatorname{div} \boldsymbol{\sigma} - \mathbf{b}) \cdot \boldsymbol{\eta}\, dV = 0 \ , \tag{2.2}$$

effectively transferring the strict, or strong, requirements of 2.1 to a weighted statement known as a weak form. Here the integration domain is over the spatial volume $V_{\mathcal{B}}$ of a continuous body, $\mathcal{B}$. The vector field $\boldsymbol{\eta}(\mathbf{x}, t)$ is an arbitrary vector-valued spatial function that is kinematically consistent with the desired boundary conditions. Separating each term according to

$$\int_{V_{\mathcal{B}}} \rho\, \dot{\mathbf{v}} \cdot \boldsymbol{\eta}\, dV - \int_{V_{\mathcal{B}}} \operatorname{div} \boldsymbol{\sigma} \cdot \boldsymbol{\eta}\, dV - \int_{V_{\mathcal{B}}} \mathbf{b} \cdot \boldsymbol{\eta}\, dV = 0 \ , \tag{2.3}$$

and using the product rule of differentiation yields

$$- \int_{V_{\mathcal{B}}} \operatorname{div} \boldsymbol{\sigma} \cdot \boldsymbol{\eta}\, dV = - \int_{V_{\mathcal{B}}} \operatorname{div} (\boldsymbol{\sigma} \cdot \boldsymbol{\eta})\, dV + \int_{V_{\mathcal{B}}} \boldsymbol{\sigma} : \boldsymbol{\nabla}^s \boldsymbol{\eta}\, dV \ . \tag{2.4}$$

The modified form produces a term that can readily be transformed via the divergence theorem as

$$\int_{V_{\mathcal{B}}} \operatorname{div} (\boldsymbol{\sigma} \cdot \boldsymbol{\eta})\, dV = \int_{\mathcal{S}} (\boldsymbol{\sigma} \cdot \mathbf{n}) \cdot \boldsymbol{\eta}\, d\mathcal{S} = \int_{\mathcal{S}_{\boldsymbol{\sigma}}} \tilde{\mathbf{t}} \cdot \boldsymbol{\eta}\, d\mathcal{S} + \int_{\mathcal{S}_{\mathbf{u}}} (\boldsymbol{\sigma} \cdot \mathbf{n}) \cdot \boldsymbol{\eta}\, d\mathcal{S} \ , \tag{2.5}$$

where $\mathcal{S}$ is the surface of the body $\mathcal{B}$ (sometimes written as $\mathcal{S} = \partial V_{\mathcal{B}}$ in the literature) and $\mathbf{n}$ is the outward normal defined on $\mathcal{S}$. The terms $\mathcal{S}_{\boldsymbol{\sigma}}$ and $\mathcal{S}_{\mathbf{u}}$ correspond to the portions of the surface $\mathcal{S}$ where loads and displacements are prescribed, respectively. These subsets collectively form the entire surface and do not overlap. The latter statement is summarized concisely as $\mathcal{S} = \mathcal{S}_{\boldsymbol{\sigma}} \cup \mathcal{S}_{\mathbf{u}}$

and $\mathcal{S}_{\boldsymbol{\sigma}} \cap \mathcal{S}_{\mathbf{u}} = 0$. The term $\tilde{\mathbf{t}} = \boldsymbol{\sigma} \cdot \mathbf{n}$ is a prescribed traction vector residing on the surface $\mathcal{S}_{\boldsymbol{\sigma}}$. Requiring that $\boldsymbol{\eta} = \mathbf{0}$ on $\mathcal{S}_{\mathbf{u}}$ removes the last integral and the remaining terms are collected to form

$$\int_{V_{\mathcal{B}}} \dot{\mathbf{v}} \cdot \boldsymbol{\eta}\, \rho\, dV = -\int_{V_{\mathcal{B}}} \boldsymbol{\sigma} : \boldsymbol{\nabla}^s \boldsymbol{\eta}\, dV + \int_{V_{\mathcal{B}}} \mathbf{b} \cdot \boldsymbol{\eta}\, dV + \int_{\mathcal{S}_{\boldsymbol{\sigma}}} \tilde{\mathbf{t}} \cdot \boldsymbol{\eta}\, d\mathcal{S}\ , \tag{2.6}$$

the very foundation of the MPM approximation scheme—not to mention several other numerical techniques (e.g. Finite Element Method).

The governing equations are solved at nodal points in the domain. That being the case it makes sense to build the unknown field quantities $\dot{\mathbf{v}}(\mathbf{x}, t)$ and $\boldsymbol{\eta}(\mathbf{x}, t)$ using the nodes themselves. These approximations are constructed as

$$\boldsymbol{\eta}(\mathbf{x}, t) \approx \boldsymbol{\eta}^h(\mathbf{x}, t) := \sum_i N_i(\mathbf{x})\, \boldsymbol{\eta}_i(t) \quad \text{and} \quad \dot{\mathbf{v}}(\mathbf{x}, t) \approx \dot{\mathbf{v}}^h(\mathbf{x}, t) := \sum_j N_j(\mathbf{x})\, \dot{\mathbf{v}}_j(t) \tag{2.7}$$

where $N_i(\mathbf{x})$ and $N_j(\mathbf{x})$ are the shape functions associated with nodes i and j, respectively. $\boldsymbol{\eta}_i(t)$ is an arbitrary, time-dependent nodal vector at a node i, and $\dot{\mathbf{v}}_j(t)$ is the time-dependent nodal acceleration vector at a node j. In this work the superscript h indicates a grid-based approximation. Closer inspection of the second integral term in Equation (2.6) reveals that $\boldsymbol{\eta}^h(\mathbf{x}, t)$ must be sufficiently smooth in order to accommodate non-zero action of the differential operator, $\boldsymbol{\nabla}$. Thus, at a very minimum, the shape functions $N(\mathbf{x})$ must be linear in $\mathbf{x}$ (at least C^0 continuous).

The next task is to identify an approximation scheme for the integral terms in (2.6). Representing the total body as a collection of particles of fixed mass m_p not only satisfies conservation of mass, but also allows integrals to be computed as sums over particles as follows:

$$\int_{V_{\mathcal{B}}} (\bullet)\ \rho\, dV = \sum_p \int_{V_p} (\bullet)\ \rho_p\, dV_p = \sum_p \int_{m_p} (\bullet)\ dm_p \approx \sum_p (\bullet)_p\ m_p\ . \tag{2.8}$$

The symbol $\sum_p$ indicates a summation over all particles while the subscript p refers to a particle

quantity. The approximation leading to the last term in (2.8) may be viewed either as a direct application of the *mean value theorem* of integration or as a single point numeric integration over the particle domain. This form is contingent upon the transformation to a mass element, defined as $dm = \rho\, dV$. Comparing to Equation (2.6), the proper mass element exists only for the first term and the other terms must be modified appropriately. The notion of a *mass-specific* term is introduced using the notation $(\bar{\bullet})$, which indicates the transformation of a volume-specific quantity, i.e., $(\bullet) = \rho\,(\bullet/\rho) = \rho\,(\bar{\bullet})$ to its mass-specific counterpart. In the present example this transforms the weak form equation to

$$\int_{m_{\mathcal{B}}} \dot{\mathbf{v}} \cdot \boldsymbol{\eta}\, dm = -\int_{m_{\mathcal{B}}} \bar{\boldsymbol{\sigma}} : \boldsymbol{\nabla}^s \boldsymbol{\eta}\, dm + \int_{m_{\mathcal{B}}} \bar{\mathbf{b}} \cdot \boldsymbol{\eta}\, dm + \int_{\mathcal{S}_\sigma} \tilde{\mathbf{t}} \cdot \boldsymbol{\eta}\, d\mathcal{S}\ , \tag{2.9}$$

where $\bar{\boldsymbol{\sigma}}$ and $\bar{\mathbf{b}}$ are the mass-specific Cauchy stress and body force, respectively. The primary integration domain has been transformed from the body volume $V_{\mathcal{B}}$, to the body mass $m_{\mathcal{B}}$.

2.3.2 Constructing the System of Equations

The discrete set of equations

$$\sum_j m_{ij}\, \dot{\mathbf{v}}_j = \mathbf{f}_i^{int} + \mathbf{f}_i^{ext}\ , \tag{2.10}$$

with

$$\mathbf{f}_i^{int} = -\sum_p \bar{\boldsymbol{\sigma}}_p \cdot \boldsymbol{\nabla} N_{ip}\, m_p\ . \qquad \text{and} \qquad \mathbf{f}_i^{ext} = \sum_p \bar{\mathbf{b}}_p\, N_{ip}\, m_p + \int_{\mathcal{S}_\sigma} \tilde{\mathbf{t}}\, N_{ip}\, dS \tag{2.11}$$

is obtained for the unknown nodal accelerations $\dot{\mathbf{v}}_j$ by substituting the grid-based definitions given in listing (2.7) and the integral approximation scheme outlined in (2.8) into the weak form Equation (2.9). The resulting system utilizes $m_{ij} = \sum_p N_{ip}\, N_{jp}\, m_p$, the consistent mass matrix coefficients with N_{ip} as the shape function evaluated at the particle location, i.e., $N_{ip} = N_i(\mathbf{x}_p)$.

Frequently the off diagonal coupling terms in m_{ij} are eliminated by approximating the mass matrix as a purely diagonal matrix: $m_i = \sum_p N_{ip}\, m_p$. In doing so the system in (2.10) is reduced to a series of i uncoupled equations for the i nodes describing the spatial domain.

The external surface force term in $(2.11)_2$ can be problematic in the MPM. The root of the problem lies in the fact that surface tractions must be applied on the body—a.k.a. the particles—and these objects move throughout nodal supports in time. Thus, the particle area and force orientation must be tracked appropriately so these terms can be applied at the correct nodes for any given position/orientation of the particle/surface. This is in contrast to other techniques, such as the Finite Element Method (FEM), where this term is applied directly to nodal values.

The primary goal of any analyses is to track the system in time while monitoring the evolution of key quantities at both the particle and nodal levels. This is accomplished in part by splitting a finite time increment, T, into many smaller time intervals, $\Delta t = t^{n+1} - t^n \ll T$. Over each time step Δt the current state is mapped to the nodes, a grid-based time integration is performed, and particle values are updated. In this section the details of each step are presented. The computational cycle is broken down and visualized as individual components in Figure 2.1.

The first step involves the transfer of particle quantities to the nodes. This is shown in Figure 2.1(a) and is accomplished by way of

$$\mathbf{p}_i^n = \sum_p N_{ip}\, m_p\, \mathbf{v}_p^n \quad , \quad \mathbf{f}_i^{int} = -\sum_p \bar{\boldsymbol{\sigma}}_p \cdot \boldsymbol{\nabla} N_{ip}\, m_p$$
$$\text{and} \qquad \mathbf{f}_i^{ext} = \sum_p \bar{\mathbf{b}}_p\, N_{ip}\, m_p + \int_{\mathcal{S}_\sigma} \tilde{\mathbf{t}}\, N_{ip}\, dS \tag{2.12}$$

for the momentum $\mathbf{p}_i^n$, internal force $\mathbf{f}_i^{int}$, and external force $\mathbf{f}_i^{ext}$ contributions, respectively. These values are used to solve the linear systems

$$\dot{\mathbf{v}}_i = \frac{1}{m_i}\left(\mathbf{f}_i^{int} + \mathbf{f}_i^{ext}\right) \qquad \text{and} \qquad \mathbf{v}_i^n = \frac{\mathbf{p}_i^n}{m_i}\,, \tag{2.13}$$

which yields the acceleration and velocity ($\dot{\mathbf{v}}_i$ and $\mathbf{v}_i^n$) at time t^n for each node in the domain. Here it is assumed that the consistent mass matrix is approximated using a diagonal, or lumped mass matrix as explained in the previous section. For the explicit integration scheme the nodal acceleration is assumed constant over the time step, resulting in the updated velocity field

$$\mathbf{v}_i^{n+1} = \mathbf{v}_{i,n} + \Delta\mathbf{v}_i \qquad \text{with velocity increment} \qquad \Delta\mathbf{v}_i = \Delta t\, \dot{\mathbf{v}}_i \tag{2.14}$$

describing the total field at the end of the time step Δt. The velocity field at the beginning and end of each time step are used to define the effective nodal velocity

$$\mathbf{v}_i^{n+\theta} = (1-\theta)\, \mathbf{v}_i^n + \theta\, \mathbf{v}_i^{n+1} \ , \tag{2.15}$$

where $\theta \in [0, 1]$ is an integration parameter that extracts the field at an arbitrary time, $t_{n+\theta} = t_n + \theta\Delta t$, between or at t_n and t_{n+1}. The effective velocity gives way to the position increment according to

$$\Delta\mathbf{x}_i = \mathbf{v}_i^{n+\theta}\, \Delta t \ . \tag{2.16}$$

The series of computations outlined in 2.12–2.16 are depicted in Figure 2.1(b) and together form the grid-based time integration portion of the MPM analysis. This series of nodal equations implies the nodes themselves are moving. Strictly speaking this statement is true. However, as noted previously, the nodal position is arbitrary at the beginning of each time step. It is common practice to continuously assume nodal positions coincide with their original position at $t = t^0$ for the start of each new time step. This may be interpreted as discarding the old grid and creating a new series of nodes each time step.

At this stage in the computational cycle the motion at the nodes is well defined over the time step and will not change. Therefore the resulting deformation, incurred in an incremental fashion as a

result of the change in motion, is determined based on the last known state. This stage is represented in Figure 2.1(c). The velocity gradient is computed at the particle level according to

$$\boldsymbol{\nabla}\mathbf{v}_p^{h,n+\theta} = \sum_i \mathbf{v}_i^{n+\theta} \otimes \boldsymbol{\nabla} N_{ip} \ . \tag{2.17}$$

Multiple deformation, or strain, measures exist depending on the type of analysis. A single presentation cannot possibly accommodate all the options in this regard. This section will focus on a large deformation measure obtainable from the incremental deformation gradient:

$$\mathbf{f}_p = \mathbf{1} + \Delta t \, \boldsymbol{\nabla}\mathbf{v}_p^{h,n+\theta} \ . \tag{2.18}$$

The particle strain is updated according to

$$\boldsymbol{\varepsilon}_p^{n+1} = \tilde{\boldsymbol{\varepsilon}}(\boldsymbol{\varepsilon}_p^n, \mathbf{f}_p) \tag{2.19}$$

where $\tilde{\boldsymbol{\varepsilon}}$ is a general strain function of the known deformation state at t^n and the incremental change over the time step.

The particle stress is determined from

$$\bar{\boldsymbol{\sigma}}_p^{n+1} = \frac{\partial \bar{\psi}(\boldsymbol{\varepsilon}_p^{n+1})}{\partial \boldsymbol{\varepsilon}_p^{n+1}} \ , \tag{2.20}$$

where $\bar{\psi}$ is the mass specific free energy function. Much like the strain function, $\tilde{\boldsymbol{\varepsilon}}$, the free energy function is typically cast in terms of several additional variables, including state dependent quantities required for elastoplastic constitutive laws. For the time being these details are omitted. The key point to take from this presentation is the stress is a function of the updated strain. This implies a hierarchical structure that will be exploited in later chapters. For the special case of a

linear elastic material, the particle stress is obtained as

$$\bar{\boldsymbol{\sigma}}_p^{n+1} = \frac{K}{\rho_0}(\operatorname{tr}\boldsymbol{\varepsilon}_p^{n+1})\,\mathbf{1} + \frac{2\,G}{\rho_0}\operatorname{dev}\boldsymbol{\varepsilon}_p^{n+1} \tag{2.21}$$

with initial mass density ρ_0, bulk modulus K, and shear modulus G. The terms $\operatorname{tr}(\bullet)$ and $\operatorname{dev}(\bullet)$ are the standard trace and deviatoric operators on a second order tensor.

Depicted in Figure 2.1, the final portion of the computational cycle is the particle velocity and position update

$$\mathbf{v}_p^{n+1} = \mathbf{v}_p^n + \sum_i N_{ip}\,\Delta\mathbf{v}_i \qquad \text{and} \qquad \mathbf{x}_p^{n+1} = \mathbf{x}_p^n + \sum_i N_{ip}\,\Delta\mathbf{x}_i\ , \tag{2.22}$$

obtained from the incremental change in nodal velocity and position over the time step. Upon completion of this last step the cycle repeats until the analysis time reaches a user prescribed value. The details provided here highlight the very basics of the Material Point Method. Of course, this traditional form is subject to change depending on the implementation strategy or details arising due to the extension of the traditional framework, such as the enhancements described in Chapters 4 and 5, or any one of the several modifications discussed in the remainder of this chapter. The current state of the literature is examined next.

2.4 *Affine Particle-In-Cell*

Throughout the past decade, the most significant leaps in the MPM methodology have derived from the computer graphic research communities (Jiang et al. 2015) (Fu et al. 2017). Whereas MPM found its genesis in the rigid bounds of engineering and physics (Sulsky et al. 1993), it finds innovation in the creative minds of animators and tinkerers.

A long-standing problem of MPM simulations has been damping of large rotation. Despite being designed for large-deformation dynamics, even a simple case of spinning an object in zero-gravity

with zero air-resistance leads to spurious slow-down of rotation. Using a finer grid with a better matched ratio of particles to grid-nodes improves this, and incorporating long-standing methods to maintain energy (such as Fluid-Implicit Particles, FLIP) (Brackbill and Ruppel 1986) helps as well, but no simple method in the engineering literature truly targets maintenance of angular momentum local to particles in a simple way.

The Affine Particle-In-Cell (APIC) (Jiang et al. 2015) method augments particles with information regarding their affine velocity ($\mathbf{C}_p$, 3x3 matrix), not just constant velocity ($\mathbf{v}_p$, 3x1 vector). APIC is a fairly minor modification to MPM that influences the grid-to-particle (G2P) and particle-to-grid (P2G) transfers. It is not fully derived here, as we later (Section 2.5) find it to be a specific case of a more general MPM analogue.

Transfer of momentum from particles in a support to a grid-node (P2G) is similar to conventional MPM (Eq. 2.12), the only difference being inclusion of an affine momentum term. Grid momentum ($\mathbf{p}_i^n$) from P2G, not considering internal forces, is now defined as:

$$\mathbf{p}_i^n = m_i^n \mathbf{v}_i^n = \sum_p m_p N_{ip} \left(\mathbf{v}_p^n + \mathbf{C}_p^n : (\mathbf{x}_i - \mathbf{x}_p^n) \right) \ . \tag{2.23}$$

After solving the equation of motion and applying boundary conditions in the standard MPM way, we then transfer from grid nodes in a support to particles (G2P). This is done similar to a typical Particle-in-Cell transfer (PIC), and similar to conventional MPM (Eq. 2.22). Two key terms are found via G2P transfer, linear velocity and the affine velocity matrix. The linear velocity on a particle is found as

$$\mathbf{v}_p^{n+1} = \sum_i N_{ip} \mathbf{v}_i^* \ , \tag{2.24}$$

and the affine velocity matrix ($\mathbf{C}_p^{n+1}$), the keystone of APIC, is constructed by

$$\mathbf{C}_p^{n+1} = \left(\sum_i N_{ip} \mathbf{v}_i^n \otimes (\mathbf{x}_i - \mathbf{x}_p^n) \right) : \mathbf{D}_p^{-1} \ , \tag{2.25}$$

where $\mathbf{D}_p$ is

$$\mathbf{D}_p = \sum_i N_{ip} (\mathbf{x}_i - \mathbf{x}_p^n) \otimes (\mathbf{x}_i - \mathbf{x}_p^n) \ . \tag{2.26}$$

The construction of the affine velocity matrix $\mathbf{C}_p$ on a particle p (Eq. 2.25) uses ingredients readily available during a G2P transfer. The only new term is $\mathbf{D}_p$ (Eq. 2.26), which depends on the shape-function being used and is unwieldy for common linear kernels. However, for quadratic and cubic B-Spline shape-functions we see $\mathbf{D}_p$ reduces to constant scalars according to

$$\mathbf{D}_p = \sum_i N_{ip} (\mathbf{x}_i - \mathbf{x}_p^n) \otimes (\mathbf{x}_i - \mathbf{x}_p^n) = \begin{cases} \frac{1}{4} \Delta x^2 \mathbf{I} \text{ for } N_{ip} = \text{ Quadratic B-Spline,} \\ \frac{1}{3} \Delta x^2 \mathbf{I} \text{ for } N_{ip} = \text{ Cubic B-Spline.} \end{cases} \tag{2.27}$$

Note that the quadratic B-Spline kernel is used for transferring in our study. This kernel has many strengths over a more standard tri-linear kernel (of which our lab has much experience with), including (i) avoiding cell-crossing instability, **(ii)** guaranteeing smoothness, and **(iii)** reducing complexity in the soon to be covered Moving-Least-Squares MPM formulation (MLS-MPM). Taking Δx as grid-spacing the quadratic B-Spline shape-function is

$$N(x) = \begin{cases} \frac{3}{4} - |x|^2 & 0 \leq |x| < \frac{1}{2} \\ \frac{1}{2} \left(\frac{3}{2} - |x| \right)^2 & \frac{1}{2} \leq |x| < \frac{3}{2} \\ 0 & \frac{3}{2} \leq |x| \ . \end{cases} \tag{2.28}$$

In 3D the shape-function is a dyadic product of Equation 2.28 (Jiang et al. 2016). Relative to

particles and grid-nodes, the quadratic B-spline shape-function is calculated as

$$N_{ip} = N_{\mathbf{i}}(\mathbf{x}_p) = N\left(\frac{1}{\Delta x}(x_p - x_{\mathbf{i}})\right) N\left(\frac{1}{\Delta x}(y_p - y_{\mathbf{i}})\right) N\left(\frac{1}{\Delta x}(z_p - z_{\mathbf{i}})\right) \ , \tag{2.29}$$

with 3D grid indices $\mathbf{i} = (i, j, k)$, grid-node position $\mathbf{x}_{\mathbf{i}} = (x_i, y_i, z_i)$, and particle position $\mathbf{x}_p = (x_p, y_p, z_p)$.

Using a quadratic B-spline reduces $\mathbf{D}_p$ to a trivial form ($\mathbf{D}_p = \frac{1}{4}\Delta x^2 \mathbf{I}$), as it also does for the cubic B-spline stencil ($\mathbf{D}_p = \frac{1}{3}\Delta x^2 \mathbf{I}$) (Eq. 2.27). In these cases the inversion of diagonal matrix $(\mathbf{D}_p)^{-1}$ is just a reciprocal. Taken against lumped mass formulations, it is simply a constant scalar multiplication, conceptually and computationally easy to execute despite the seemingly more complex stencils used (compared to tri-linear stencils). This is part of the beauty of APIC and B-spline kernels, a more ubiquitous stencil that tackles the unwieldy cell-crossing instability problem and actually simplifies the workflow.

The computational load of APIC is not large but is not negligible, in parallel code frameworks matrix operations can be optimized. Alternatively, methods that build on APIC reduce the total computational load in creative ways. Finding means to efficiently gather and broadcast to and from the 3x3x3 supports of a quadratic B-spline stencil (27 nodes), or 4x4x4 support of a cubic B-spline stencil (64 nodes) is a priority in memory management. More memory is also required on a per-particle basis to maintain the affine velocity matrix ($\mathbf{C}_p$) across the G2P and P2G operations. Later chapters (Chapter 4) will describe a parallel computational approach to tackle the enlarged kernel supports and a grid-to-particle-to-grid (G2P2G) fused operation to avoid saving $\mathbf{C}_p$ on each particle.

Polynomial Particle-in-Cell (PolyPIC, Fu et al. 2017) is a higher order scheme that encompasses APIC and deserves an honorable mention. It can help better maintain higher-order local velocity behavior (e.g. turbulence and and multi-phase fluid interactions). It is not pursued in this project,

as we were not excessively interested in these attributes, but others may find it intriguing. Going to higher-orders in PolyPIC tends to increase memory usage and computational load, so it should be used sparingly.

2.5 *Moving-Least-Squares Material Point Method*

A contribution of this project is the rephrasing, analysis, validation (Chapter 6), and application of the Moving-Least-Squares Material Point Method (MLS-MPM) to a complex engineering problem (Chapter 7). Invented by graphics researchers, it is a recent but popular advancement on MPM (Hu et al. 2018). So far it has only been used by a handful of engineers (Nakamura et al. 2021), likely due to its origins in computer graphics, the inaccessibility of its derivation, and the lack of a general engineering viewpoint on its viability. By placing MLS-MPM in our engineering lab's consistent nomenclature, summarizing its key engineering implication, and incorporating it into our own project we aim to bring MLS-MPM to the typical engineer and as a consequence bring engineers a bit closer to the booming field of computer graphics.

MLS-MPM **(i)** naturally leads to APIC and PolyPIC, **(ii)** removes calculation of nodal shape function gradients for force computation, **(iii)** favors complex interaction with arbitrary rigid bodies, **(iv)** doubles performance, and **(v)** is simpler to implement than traditional MPM.

Application of Moving Least Squares (MLS the general approach, not specifically MLS-MPM) is common in a number of numerical methods. Notably mesh-less and general particle based approaches like element free Galerkin method (EFG) by and Huerta et al. 2004 and Belytschko et al. 1994. Smoothed particle hydrodynamics (SPH) has made use of MLS Band et al. 2018 to improve local approximations of fields, field gradients, and general boundary behavior in SPH which has well-known difficulties at boundaries. MLS shows some resemblance to the Reproducing Kernel Particle Method (RKPM, Liu et al. 1995), which is another popular mesh-less approach for large deformation, and recenetly saw introduction of MLS directly into the

RKPM formulation in Chen et al. 2020 for multi-phase applications.

Here we present a summarized MLS-MPM deviation that highlights key differences with standard MPM. A rigorous derivation has many minor notes to cover which are not especially relevant to this study. However, we briefly touch on major mathematical aspects of MLS. This is to give substance to the equations we actually need to implement to run a simulation, and more importantly, to show how they differ from traditional MPM (Sulsky et al. 1994), as covered in Section 5.12.2 in the existing engineering notation of our lab (Mast 2013).

We begin the mathematical groundwork for MLS-MPM by defining the polynomial basis, $\mathbf{P}$, as **(i)** linear, **(ii)** complete, and **(iii)** centered about a grid-node i. Written mathematically, these conditions create a polynomial basis with form

$$\mathbf{P}(\mathbf{x}_i - \mathbf{x}) = \left[1, (\mathbf{x}_i - \mathbf{x})^T\right]^T \quad . \tag{2.30}$$

The specific properties set above are not technically needed for Moving-Least-Squares, but we will soon demonstrate the conditions simplify the approach substantially. They also naturally lead to a previously discussed method from computer graphics (Affine Particle-In-Cell, Section 2.4), now given a mathematical backbone.

With an established polynomial basis, we now seek to reconstruct a function value ($\tilde{u}$) and function value gradient ($\nabla\tilde{u}$) at an arbitrary point ($\mathbf{x}$) respective to grid-nodes ($1, 2...i$) within it's support. Ultimately, these are equivalent to particle velocity ($\mathbf{v}_p$) and the particle affine velocity matrix ($\mathbf{C}_p$) in APIC (Equations 2.24 and 2.25) (Section 2.4) due to our polynomial basis choice. Reconstruction of function values and function value gradients is defined as

$$[\tilde{u}, \nabla u]^T = \mathbf{M}^{-1}(\mathbf{x})\mathbf{Q}^T\Xi(\mathbf{x})\,[u_1, \ldots, u_N] \quad , \tag{2.31}$$

where $\Xi(\mathbf{x})$ is the diagonal weighting matrix $\Xi_{ii} = \zeta_i(\mathbf{x})$, $\mathbf{Q}(\mathbf{x})$ is a set of our polynomial basis

evaluated at sample points (i.e. grid-nodes) $[\mathbf{Q}(\mathbf{x})] = [\mathbf{P}(\mathbf{x}_1 - \mathbf{x}), \ldots, \mathbf{P}(\mathbf{x}_N - \mathbf{x})]^T$, $[\mathbf{M}(\mathbf{x})]$ is a moment matrix $[\mathbf{M}(\mathbf{x})] = [\mathbf{Q}]^T\ [\Xi]\ [\mathbf{Q}]$, and ζ_i is a weighting function about grid-nodes. These quantities will mostly disappear due to the choice of polynomial basis ($\mathbf{P}$) conditions an the conditions to be placed on ζ_i. For now, note that the reconstruction in equation 2.31 will reconstruct field values (i.e. velocity and velocity gradient) on an arbitrary point (i.e. MPM particle) respective to sample points (i.e. MPM grid-nodes in shape-function support of the particle) using a polynomial basis ($\mathbf{P}$) (Eq. 2.30) and weighting function (ζ_i) which are carefully curated with the end goal of simplifying the algorithm.

The MLS moment matrix, $[\mathbf{M}]$, reduces to something similar to the inertia-like matrix, $[\mathbf{D}]$, from APIC (Equation 2.27) when using certain shape-function kernels with our polynomial basis. Paired with restrictions to be placed on the weighting function at a grid-node (ζ_i), simplify the moment matrix at a particle p to

$$\mathbf{M}(\mathbf{x}_p) = \sum_i \zeta_i(\mathbf{x}_p) \begin{bmatrix} 1 & (\mathbf{x}_i - \mathbf{x}_p)^T \\ (\mathbf{x}_i - \mathbf{x}_p) & (\mathbf{x}_i - \mathbf{x}_p)(\mathbf{x}_i - \mathbf{x}_p)^T \end{bmatrix} = \begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_p \end{bmatrix}, \tag{2.32}$$

which further simplifies when shape-function $N_i(\mathbf{x})$ is a quadratic or cubic B-spline to $\mathbf{D}_p = \text{diag}\left(D_p\right)$, where $D_p = \frac{1}{4}\Delta x^2$ and $D_p = \frac{1}{3}\Delta x^2$ respectively for quadratic and cubic B-splines, exactly as in APIC (Eq. 2.27). Grid-spacing is Δx and assumed to be a constant (e.g. uniform grid in Cartesian coordinates). Note that scalar $M_p = D_p$ for quadratic and cubic B-splines and is used to represent the reduced moment matrix later in the formulation for convenience.

This ends the mathematical groundwork for MLS-MPM. Using this approach, a few keys differences arise respective to conventional MPM. Returning to our governing equation,

$$\rho\, \dot{\mathbf{v}} = \text{div}\, \boldsymbol{\sigma} + \mathbf{b}\ , \tag{2.33}$$

we discretize its weak-form in a Galerkin fashion. For brevity in these notes we drop the body force term. The updated Lagrangian time discretization of the weak form of conservation of momentum is then

$$\int_{V_{\mathcal{B}}} \rho \dot{\mathbf{v}} \cdot dV - \int_{V_{\mathcal{B}}} \operatorname{div} \boldsymbol{\sigma} \cdot \boldsymbol{\eta} dV - \int_{V_{\mathcal{B}}} \mathbf{b} \cdot \boldsymbol{\eta} dV = 0 \ , \tag{2.34}$$

the middle term is first decomposed as

$$- \int_{V_{\mathcal{B}}} \operatorname{div} \boldsymbol{\sigma} \cdot \boldsymbol{\eta} dV = - \int_{V_{\mathcal{B}}} \operatorname{div} (\boldsymbol{\sigma} \cdot \boldsymbol{\eta}) dV + \int_{V_{\mathcal{B}}} \boldsymbol{\sigma} : \boldsymbol{\nabla}^s \boldsymbol{\eta} dV \ , \tag{2.35}$$

which may then be taken according to the divergence theorem as

$$\int_{V_{\mathcal{B}}} \operatorname{div} (\boldsymbol{\sigma} \cdot \boldsymbol{\eta}) dV = \int_{\mathcal{S}} (\boldsymbol{\sigma} \cdot \mathbf{n}) \cdot \boldsymbol{\eta} d\mathcal{S} = \int_{\mathcal{S}_\sigma} \tilde{\mathbf{t}} \cdot \boldsymbol{\eta} d\mathcal{S} + \int_{\mathcal{S}_{\mathbf{u}}} (\boldsymbol{\sigma} \cdot \mathbf{n}) \cdot \boldsymbol{\eta} d\mathcal{S} \ , \tag{2.36}$$

to allow Equations 2.34, 2.35, and 2.36 to recombine (while dropping the last term in Equation 2.36) into

$$\int_{V_{\mathcal{B}}} \dot{\mathbf{v}} \cdot \boldsymbol{\eta} \rho dV = - \int_{V_{\mathcal{B}}} \boldsymbol{\sigma} : \boldsymbol{\nabla}^s \boldsymbol{\eta} dV + \int_{V_{\mathcal{B}}} \mathbf{b} \cdot \boldsymbol{\eta} dV + \int_{\mathcal{S}_\sigma} \tilde{\mathbf{t}} \cdot \boldsymbol{\eta} d\mathcal{S} \ . \tag{2.37}$$

For a discrete time-step (Δt) that bridges time t^n and t^{n+1} as $t^{n+1} = \Delta t + t^n$, Equation 2.37 is stated, dropping the body force term for brevity, as

$$\frac{1}{\Delta t} \int_{V^n_{\mathcal{B}}} \rho(\mathbf{x}, t^n)(\hat{\mathbf{v}}^{n+1} - \mathbf{v}^n(\mathbf{x})) \cdot \boldsymbol{\eta}(\mathbf{x}, t^n) dV = \int_{\mathcal{S}^n_\sigma} \tilde{\mathbf{t}}(\mathbf{x}, t^n) \cdot \boldsymbol{\eta}(\mathbf{x}, t^n) d\mathcal{S} - \int_{V^n_{\mathcal{B}}} \boldsymbol{\sigma}(\mathbf{x}, t^n) : \boldsymbol{\eta}(\mathbf{x}, t^n) dV \ . \tag{2.38}$$

The left side of the weak form (Eq. 2.38) is then rephrased as a summation over particles domains p with

$$\int_{V^n_{\mathcal{B}}} \rho(\mathbf{x}, t^n) \mathbf{v}^n(\mathbf{x}) \cdot \boldsymbol{\eta}(\mathbf{x}, t^n) dV = \sum_p \int_{V^n_p} \rho(\mathbf{x}, t^n) \mathbf{v}^n_\alpha(\mathbf{x}) \cdot \boldsymbol{\eta}_\alpha(\mathbf{x}, t^n) \ . \tag{2.39}$$

Approximating our velocity ($\mathbf{v}$) and test function ($\boldsymbol{\eta}$) using samples on grid-nodes in a MLS

fashion is then viable using:

$$\mathbf{v}^n(\mathbf{x}) = \sum_j \Phi_j(\mathbf{x})\mathbf{v}_j^n \ , \tag{2.40}$$

$$\boldsymbol{\eta}(\mathbf{x}, t^n) = \sum_i \Phi_i(\mathbf{x})\boldsymbol{\eta}_i^n \ . \tag{2.41}$$

The significant difference here between MLS-MPM and MPM is the use of MLS shape-functions Φ, defined as Equation 2.42:

$$\Phi_i(\mathbf{x}) = \zeta_i(\mathbf{x}_p^n) \left[\mathbf{P}(\mathbf{x} - \mathbf{x}_p^n)\right]^T \left[\mathbf{M}(\mathbf{x}_p^n)\right]^{-1} \left[\mathbf{P}(\mathbf{x}_i - \mathbf{x}_p^n)\right] \ , \tag{2.42}$$

where ζ_i is a weight function centered on grid-nodes. We set the weighting function equal to quadratic or cubic B-Spline shape-functions ($\zeta_i = N_i$). Using MLS shape-functions (Φ) in place of a typical MPM B-spline shape-function (N) is the key difference in the method. Further, using a weighting function in the MLS shape-function that is a standard B-spline converges MLS-MPM towards MPM while maintaining unique properties. In a nutshell, MLS-MPM differs from MPM in its use of a very specifically designed shape-function with respect to carefully curated constraints.

Plugging in the reconstructed values (Equations 2.40 and 2.41) into the weak form over particle domains (Equation 2.39) reduces the equation to

$$\sum_p \int_{V_p^n} \rho(\mathbf{x}, t^n)\mathbf{v}^n(\mathbf{x}) \cdot \boldsymbol{\eta}(\mathbf{x}, t^n) dV = \sum_{p,i,j} \boldsymbol{\eta}_i^n \cdot \mathbf{v}_j^n m_{ij} \ . \tag{2.43}$$

Here the mass matrix (m_{ij}) depends on the nodal MLS shape functions. That is

$$m_{ij} = \int_{V_p^n} \rho(\mathbf{x}, t^n)\Phi_i(\mathbf{x})\Phi_j(\mathbf{x}) dV \ , \tag{2.44}$$

which is typically converted to "lumped mass" form in various MPM variants and is applied here

for MLS-MPM. Note that mass is still conserved. To do this "lump" (i.e. sum) across rows of the mass matrix (m_{ij}) and save results to a single diagonal element, reducing into a diagonal form (m_i) that can decouple the equations. Note that this step is found in standard MPM (Section 2.3.2) and has well understood implications, though high-order methods often make use of the full mass matrix or expanded approximations of it. Our lumped mass matrix is defined as:

$$m_i^n = \sum_p \int_{V_p^n} \rho(\mathbf{x}, t^n)\Phi_i(\mathbf{x})dV \approx \sum_p m_p \Phi_i(\mathbf{x}_p^n) = \sum_p m_p N_{ip}^n \ . \tag{2.45}$$

Now we may see that linear and affine momentum transfer from particles-to-grid (P2G) is identical to APIC (Eq. 2.23) where grid momentum is

$$\mathbf{p}_i^n = m_i^n \mathbf{v}_i^n = \sum_p m_p N_{ip}^n (\mathbf{v}_p^n + \mathbf{C}_p^n : (\mathbf{x}_i - \mathbf{x}_p^n)) \ . \tag{2.46}$$

The grid-to-particle (G2P) transfer (i.e. reconstruction) of velocity and the affine velocity matrix is, again, as it was in APIC G2P transfer (Eq. 2.24 and 2.25), equal to

$$\mathbf{v}_p^{n+1} = \sum_i N_{ip}^n \hat{\mathbf{v}}_i^{n+1} \ , \quad \text{and} \tag{2.47}$$

$$\mathbf{C}_p^{n+1} = \left(\sum_i N_{ip}^n \hat{\mathbf{v}}_i^{n+1} \otimes (\mathbf{x}_i - \mathbf{x}_p^n) \right) : \mathbf{D}_p^{-1} \ . \tag{2.48}$$

With the basic momentum transfer found to be as it was in APIC, we move on to the stress term for MLS-MPM on the right side of the weak form of equation 2.38. This is where the two methods diverge from each other. Assuming there is no boundary integral contribution (null Neumann condition), stresses are summed over particle domains p such that

$$-\int_{V_B^n} \boldsymbol{\eta}(\mathbf{x}, t^n) : \boldsymbol{\sigma}(\mathbf{x}, t^n)dV = -\sum_p \int_{V_p^n} \nabla\boldsymbol{\eta}(\mathbf{x}, t^n) : \boldsymbol{\sigma}(\mathbf{x}, t^n)dV \ . \tag{2.49}$$

Internal force computation following particle advection, and thus relative to potentially new grid node indices u within particle p's shape-function stencil width N_{up}, is then determined after substituting the reconstructed test function and integrating over the domain. Assuming single-point quadrature for a particle (i.e. particles are points possessing volume) we find internal force on grid-node u as

$$\mathbf{f}_u^{\text{int}} = \sum_p \int_{V_p^n} \boldsymbol{\nabla}\boldsymbol{\eta}(\mathbf{x}, t^n) : \boldsymbol{\sigma}(\mathbf{x}, t^n) dV \approx - \sum_p V_p^n \left[\mathbf{M}_p^n\right]^{-1} \boldsymbol{\sigma}_p^n N_{up}^n (\mathbf{x}_u^n - \mathbf{x}_p^n) \ , \tag{2.50}$$

which is similar to that of standard MPM where $\mathbf{f}_u^{\text{int}} = -\sum_p V_p^n \boldsymbol{\sigma}_p^n : \boldsymbol{\nabla} N_{up}^n$. The difference between the MLS-MPM and MPM internal force update is that MLS-MPM does not determine the shape-function gradient $\boldsymbol{\nabla}_{up}^n$. This simplifies the force calculation and reduces computational time. However, it requires evaluation of both $\left[\mathbf{M}_p^n\right]^{-1}$ and $(\mathbf{x}_u^n - \mathbf{x}_p^n)$. The former is a known, constant scalar (M_p^{-1}) for quadratic and cubic B-Spline kernels (which we always advocate to use). The latter is simple for uniform Cartesian grids which is conveniently the most common grid set-up in MPM.

To find the particle stress we first must have the deformation the particle undergoes due to the G2P step. For updating our deformation gradient ($\mathbf{F}_p$) on a particle after the G2P momentum transfer, we return to an equation in the standard MPM derivation:

$$\mathbf{F}_p^{n+1} = \left(\mathbf{I} + \Delta t \frac{\partial \hat{\mathbf{v}}^{n+1}}{\partial \mathbf{x}}(\mathbf{x}_p^n)\right) \cdot \mathbf{F}_p^n \qquad \text{where the 2nd-order Identity tensor is } \mathbf{I} \ . \tag{2.51}$$

The partial derivative in MLS-MPM is the affine velocity gradient matrix that we reconstructed during the APIC-style G2P transfer (Eq. 2.48) because of our choice of polynomial basis (Eq. 2.30), phrased as:

$$\frac{\partial \hat{\mathbf{v}}^{n+1}}{\partial \mathbf{x}} = \mathbf{C}_p^{n+1} \ . \tag{2.52}$$

This means the deformation gradient is updated using only the previously stored deformation gradient on the particle ($\mathbf{F}_p^n$, 3x3 2nd-order tensor) and the affine velocity matrix ($\mathbf{C}_p^n$, 3x3 matrix) over the time-step (Δt):

$$\mathbf{F}_p^{n+1} = \left(\mathbf{I} + \Delta t \mathbf{C}_p^{n+1}\right) \cdot \mathbf{F}_p^n \ . \tag{2.53}$$

Knowing the particle deformation gradient, we may evaluate the stress relative to the material. Summing over particles in a grid-node support calculates the internal force (Eq. 2.54), now stated as a function of the particle deformation gradient and potential energy:

$$\mathbf{f}_i^{\text{int}} = -\frac{\partial E}{\partial \mathbf{x}_i} = -\sum_p N_{ip}^n V_p^0 \left[\mathbf{M}_p\right]^{-1} \frac{\partial \Psi}{\partial \mathbf{F}}(\mathbf{F}_p^n) \otimes \mathbf{F}_p^n : (\mathbf{x}_i^n - \mathbf{x}_p^n) \ , \tag{2.54}$$

where we have reverted from grid-index u notation to i for simplicity. In the above Equation 2.54 we assume that the constitutive model for the particle material is hyper-elastic and thereby definable in terms of energy potentials and the first Piola-Kirchhoff stress. More about hyper-elastic constitutive models and stress representations is discussed in Chapter 5.
For clarity, rephrase in terms of volume-specific Cauchy stress, $\boldsymbol{\sigma}$ (Eq. 2.55), as this is more natural to most engineers than defining by deformation gradients and the first Piola-Kirchoff stress:

$$\boldsymbol{\sigma} = \frac{1}{\det \mathbf{F}} \frac{\partial \Psi}{\partial \mathbf{F}} \otimes \mathbf{F} \ . \tag{2.55}$$

Initial particle volume (V_p^0) can also be presented as particle volume V_p^n at time t^n (Eq. 2.56) to allow for mass-specific representation of the Cauchy stress using the deformation gradients determinant (sometimes referred to as the Jacobian, J) as:

$$\det \mathbf{F} \, V_p^0 = V_p^n \ . \tag{2.56}$$

Substituting equations 2.55 and 2.56 into equation 2.54 brings us to a clean, engineering

representation of MLS-MPM's stress contribution to internal force,

$$\mathbf{f}_i^{\text{int}} = -\sum_p N_{ip} m_p M_p^{-1} \bar{\boldsymbol{\sigma}}_p^n (\mathbf{x}_i^n - \mathbf{x}_p^n) \ . \tag{2.57}$$

Reviewing standard MPM's equation (2.11), the difference between MLS-MPM and MPM is clear. MLS-MPM does not need to calculate the shape-function gradients ($\boldsymbol{\nabla} N_{ip}$) to determine internal forces.

Another useful aspect of MLS-MPM is that the writing of internal force and affine momentum contributed by a particle can be combined into a fused contribution matrix ($\mathbf{Q}_p$) to reduce the number of operations required to write results, computationally, onto the grid during P2G, written as:

$$\mathbf{Q}_p = m_p \mathbf{C}_p - \Delta t V_p^0 M_p^{-1} \frac{\partial \boldsymbol{\Psi}}{\partial \mathbf{F}} (\mathbf{F}_p^n) \otimes \mathbf{F}_p^n = m_p \left(\mathbf{C}_p - \Delta t M_p^{-1} \bar{\boldsymbol{\sigma}}_p^n \right) \ . \tag{2.58}$$

Making the full P2G operation in MLS-MPM (consisting of momentum, affine momentum, and internal particle force) reduce into a fused write of the form:

$$(m_i \mathbf{v}_i + \Delta t \mathbf{f}_i^{\text{int}}) = \sum_p N_{ip} (m_p \mathbf{v}_p^n + \mathbf{Q}_p : (\mathbf{x}_i^n - \mathbf{x}_p^n)) \ . \tag{2.59}$$

The fused write in Equation 2.59, in tandem with avoidance of shape-function gradients, allows MLS-MPM to save memory and computational time over MPM– roughly doubling its speed. However, engineers will often want to separate the internal force from momentum write during P2G (e.g. for FLIP-PIC mixes, discussed in Chapter 5). This is not a problem at all for MLS-MPM, though it will lose the savings of the fused write.

In summary MLS-MPM and MPM are most noticeably different in their particle deformation gradient update and grid-node internal force computation. MLS-MPM simplifies both by reusing quantities from APIC G2P transfer and by selecting shape-functions of favorable properties.

Ultimately, MLS-MPM doubles the speed of MPM while being much easier to implement and garners the benefits of previously discussed APIC (Section 2.4).

2.6 *Final Remarks*

An introduction to the Material Point Method (MPM) is provided in this chapter. New advancements on MPM (APIC, MLS-MPM), hailing from the computer graphics research community, are discussed in engineering terms. We rephrase derivations for consistency with our lab's nomenclature, making the formulation more accessible to engineers. These methods are used throughout our project for numerical simulations of fluid-driven debris-fields.
In the next section the most fatal drawback of MPM and its analogues is approached– Hardware limitations.

Chapter 3

ACCELERATOR HARDWARE FOR ENGINEERING SIMULATIONS

3.1 *Introduction*

The Material Point Method (MPM), our chosen tool, faces the hurdle of practicality– No one can run it well on a desktop computer. Despite aspirations for advanced multi-phase, multi-material, large-deformation simulations (precisely what we need) computational times seem insurmountable. No algorithmic modification to MPM has fixed this issue, though advancements like Moving-Least-Squares Material Point Method (MLS-MPM) and implicit implementations do improve run-times. Simulation times must be accelerated by *at least* an order of magnitude.
We propose the way to make MPM usable is by switching the hardware infrastructure it is ran on. Graphics Processing Units (GPU), devices often used for video-games, may be the key to this engineering problem.

3.2 *GPU vs. CPU*

Mohr's Law predicts a doubling of transistors (a fundamental hardware unit) in dense, integrated circuits roughly every two years. While this law held true for many years, it faltered in the mid-2000s, where Central Processing Units (CPUs), the typical computational body for engineering simulations, began to find diminishing returns in transistor counts year-over-year. In response, companies like Intel and AMD switched focus to developing CPUs with an increasing number of low-latency connected computational cores (both physical and virtual). Use of multiple cores in CPUs allowed for parallel computations to be ran. While the computation wasn't faster

(i.e. higher transistor count) tasks could be divided into parallel portions and executed alongside each other, as opposed to sequentially. This allowed for maintenance of computational growth through the 2000s.

During the same time period, computer engineers experimented with the design of specialized hardware units now known as Graphics Processing Units (GPUs) that maximized the number of computational cores and the memory available to these cores. Instead of focusing on developing extremely complex cores for advanced logic (e.g. running operating systems) with access to small but hyper-efficient memory pools (i.e. CPUs), cores were simplified for basic computation (which most engineering software is) and large memory pools were designed to foster high through-put computation. Figure 3.1 shows the basic hardware difference between a CPU and GPU, the former optimizing for few cores with advanced cache and the latter for many cores with simpler cache. In graphical use, where an extremely large number of basic computations must be executed in real-time, the use of GPUs is evident– Engineering simulations are not dissimilar.

Every four years we have observed roughly a doubling of three measures for GPUs: **(i)** core count, **(ii)** global memory size, and **(iii)** transfer speed. This does not highlight improvements found in smaller memory pools, core efficiencies, mass production capabilities, software, and the availability of GPUs in communal HPC systems and consumer machines alike. The absolute computational limit of what could be performed in our simulations at the beginning of this project (50 - 100 million particle MPM simulations) has become an easy case by the end, roughly three years, purely from the fast development of the GPU sphere which now enable simulations unimaginable to us previously (1 billion particles). Further, our access to higher-end hardware has expanded as far more institutions have invested in modern GPU-equipped computing centers, allowing our simulations to grow well over an order of magnitude in size. We anticipate these trends to continue throughout the decade as exa-scale HPC systems proliferate.

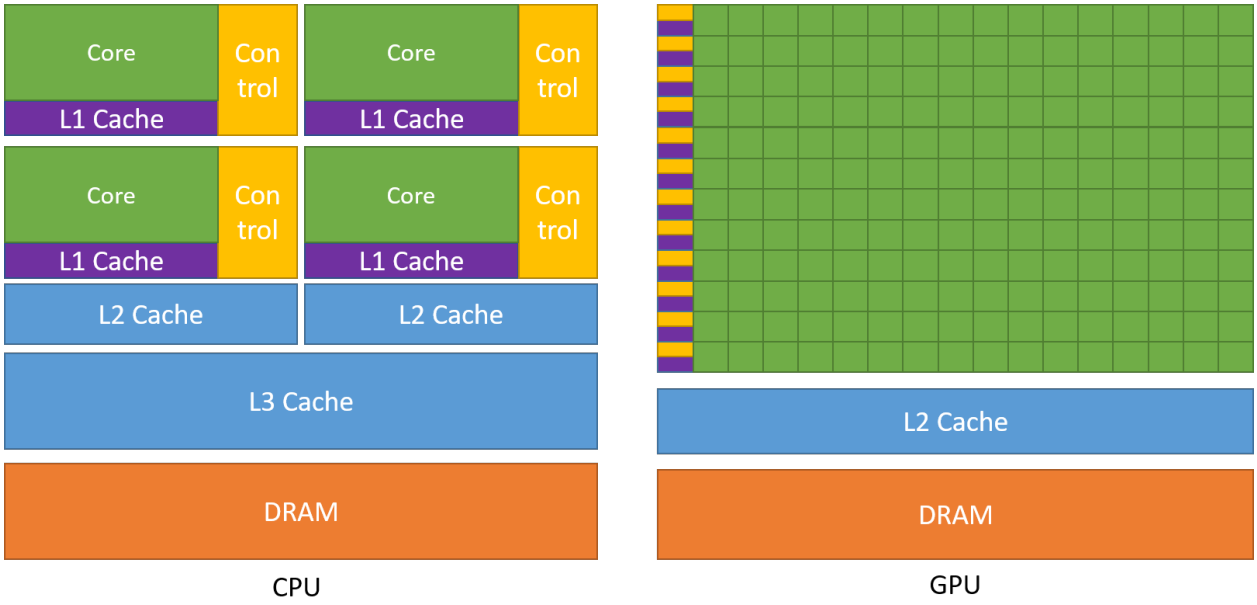

Fig. 3.1. GPU and CPU basic hardware layout. Note that more transistors in the GPU are dedicated to data-processing. Courtesy of NVIDIA.

3.3 *Graphics Processing Units*

Graphics Processing Units are specialized hardware units for high volume through-put of effectively parallel and logically simple computations. They enjoy extensive usage in the field of computer graphics, their namesake, but are very useful in engineering as well. Fields of engineering (e.g. Aerospace) have made thorough use of these devices to accelerate numerical simulations far beyond what CPU-centered systems can manage. However, civil engineering in general, and coastal engineering in particular, has been comparatively slow to embrace.

Two corporations provide the majority of GPUs: NVIDIA and AMD. Note that our project exclusively made use of NVIDIA devices, so some descriptions will be specific to the hardware/software architecture of NVIDIA Devices/CUDA programming. CUDA is the programming method developed by NVIDIA for use on their devices. Its model is Single-Instruction Multiple-Threads (SIMT), as opposed to Single-Instruction Multiple-Data

(SIMD) that is common in non-GPU parallel programming. The differentiation is important, but means little without covering some basics. To understand how parallel programs execute on a GPU you must first understand the **Thread/Block/Grid** structure of a CUDA program. Figure 3.2 demonstrates the hierarchy visually: grids contain blocks, blocks contain threads.

3.3.1 Thread/Block/Grid

A **thread** is a sequence of operations. It is the basic form of parallel instruction. Each thread possesses an index. A **block** is a collection of threads. Each block possess an index and a block dimension. When a device function is launched, threads are organized inside of blocks. A **grid** is a collection of blocks.

Indices of threads/blocks and the number of threads in a block are known by each thread. This allows threads to behave uniquely. In simple terms, the unique index lets a thread access unique data for its operations, i.e. an element in a 2D array. Hence, a single instruction can be given to "multiple threads", but the threads may operate on "multiple data" as each thread can reference its own unique indices to set which data it will work with.

Engineering tasks are parallel to different degrees. For instance, a central difference scheme must know information regarding the data in the vicinity of a point of computation (i.e. a thread executing a point must access data of adjacent points, operated by adjacent threads, but nothing beyond that). A block of threads that access spatially organized points will share most data, with the exception of **Halo** data. The thin seams between blocks must be shared by blocks.

A scheme may be effectively parallel globally but access to local data should be maintained in a hierarchical manner. Logically the CUDA thread/block/grid structure and memory pools are designed to obey tasks in hierarchical manner with appropriate access to data.

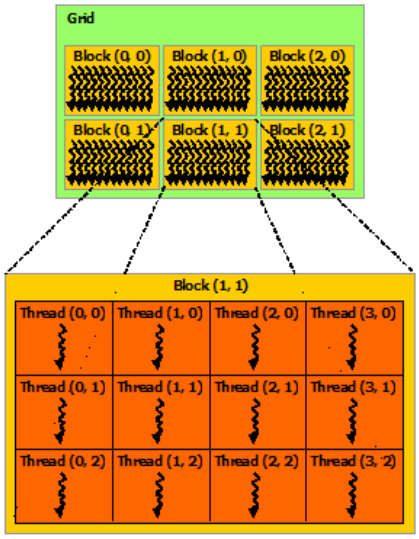

Fig. 3.2. Thread/Block/Grid structure of CUDA program. Hierarchical nature of CUDA operation sequences. Threads are grouped into blocks. Blocks are grouped into grids. Threads and blocks are indexed (indices that can be accessed within a thread/block). Courtesy of NVIDIA.

3.3.2 SP/SM/GPU

Hardware structure reflects software structure. Loosely, the Thread/Block/Grid hierarchy manifests as Stream Processor/Streaming Multiprocessors/Graphics Processing Units (SP/SM/GPU).

When the host launches a global function, also known as a **kernel**, blocks are sent to **Streaming Multiprocessors** (SMs) on the device. Their exact physical properties and count depend on the device, but SMs possess some number of **Stream Processors** (SPs, a.k.a. CUDA Cores). **Warps** are groups of 32 SPs on a SM that execute in lock-step. Blocks divide across warps in a balanced manner according to the CUDA scheduler so that threads may execute. Figure 3.3 shows a basic schematic of how a set of 8 blocks in a grid can distribute differently depending on the number of available SMs, resulting in a different schedule of execution. The number of warps executing on a SM at a given time depends on the number of SPs (e.g. 128 SPs could, ideally, execute 4 warps in

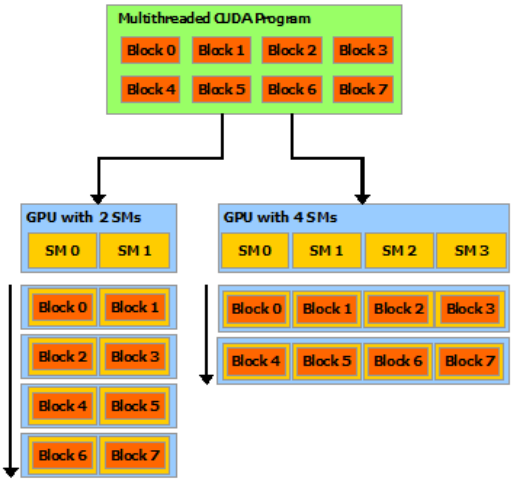

Fig. 3.3. Scaling across streaming multiprocessors. Blocks automatically distribute across streaming multiprocessors. Courtesy of NVIDIA.

parallel).

As a programmer, you will not know the order that warps execute in a block, but you do know that threads within a warp execution are done together. **Half-Warps** are noteworthy due to how CUDA schedules instructions and executes them at slightly finer hardware granularity than a full-warp. Modern card architectures and CUDA versions abstract away half-warps so it isn't often a concern. Going forward, warp is sometimes interchangeable with half-warp.

Having a brief glimpse into the structure of a CUDA program (Thread/Block/Grid) and its hardware reflection (SP/SM/GPU) we now address the management of data as it moves throughout hierarchical heuristics.

3.3.3 Memory and Data-Transfer

Software and computational hardware hierarchy is matched in memory hierarchy. Here the basics of memory on the **host** (CPU, RAM, HDD, SSD) and **device** (GPUs) are covered. Figure 3.4 shows basic device-side memory (local/shared/global) relative to the thread/block/grid structure of

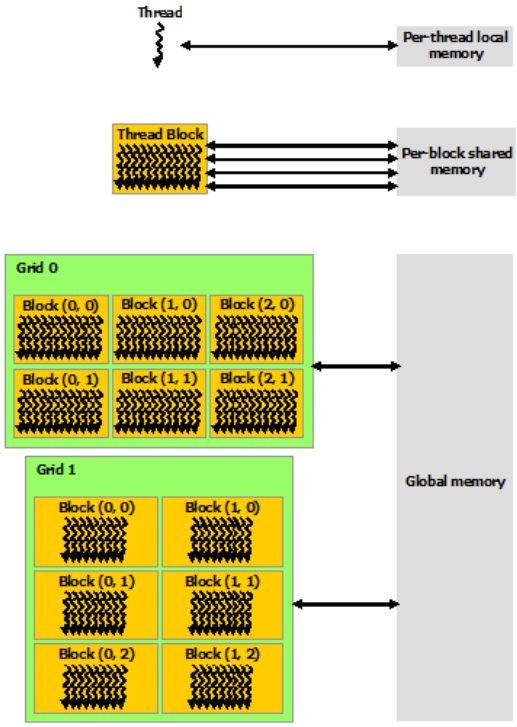

Fig. 3.4. Memory hierarchy for GPU thread/block structure. Courtesy of NVIDIA.

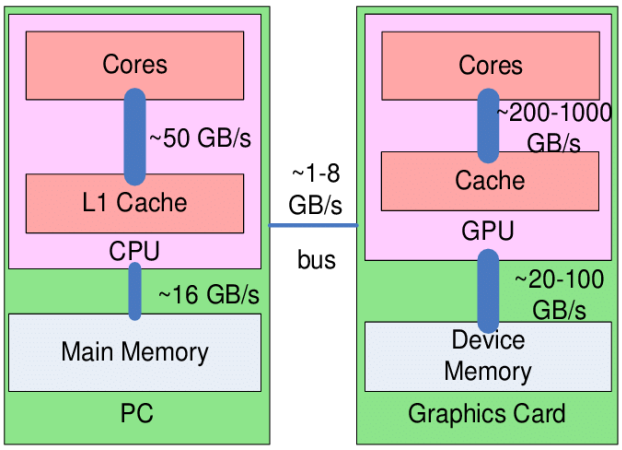

Fig. 3.5. Bottlenecks in data-transfer for GPU systems. Courtesy Binotto 2011.

a program.

Each thread has its own **register** (fast) for memory use. Each thread also possesses **local memory** (slow, but cached). On a per-thread basis, we aim to maximize register use and minimize local use.

Threads in a block have **shared memory** that only the block may access. This memory is "on-chip" and so it can be as fast as register memory assuming no bank-conflicts occur (discussed later).

All threads may access **global memory**. It is slow, uncached, and not located on chip but it has the largest available space. We typically want to minimize threads accessing global memory as it is 100x slower than register/shared memory without bank conflicts, however, the bulk of our program will reside here as it is even worse to move it off the GPU.

There also exists **constant memory** (slow, cached) and **texture memory** (cached, optimized for 2D operations), which are read-only for all threads. These function similar to global memory, but for more specific purposes.

Always aim to minimize memory transfer between host and device as it is a primary bottleneck in program performance. We want as much of our simulation to exist on the GPU memory-wise,

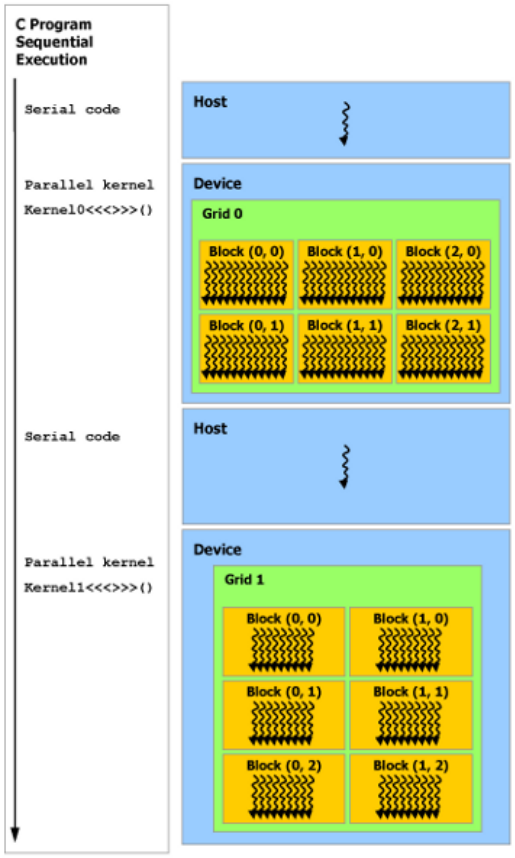

Fig. 3.6. GPU heterogeneous programming model. Heterogeneous programming model for computing across a host and GPU device(s). Courtesy of NVIDIA.

only using the CPU to send instructions to the GPU and to occasionally serve as file input/output for the program itself. Further, internal memory transfers on the host and device can be optimized. Figure 3.5 highlights this aspect, where the "bus" between the host (CPU-side) and device (GPU-side) is the primary bottle-neck for a whole program. Note that the "bus" (i.e. PCI-e interconnect bus) has a data transfer rate between the host and device that is orders of magnitude slower than any internal transfers on the host or device. PCI-e busses are actively improving in volume and speed of data transfer (doubling roughly every four years), but they are still slow and thus programmers should avoid moving information between the host and device when possible.

Figure 3.6 demonstrates how using a CPU and GPU for a program creates a natural serial bottle-neck at the host (i.e. CPU) from the software perspective. Thus it is best to limit host operations to I/O and launching parallel kernels on the device. Note that asynchronous transfer of data (device-to-host, host-to-device) can be performed while kernels are running, so transferring data that a thread is not waiting to access across the host and device isn't necessarily an issue (very useful for non-essential simulation tasks, like outputting frame data for visualization).

3.3.4 Problems and Tips

Data-Race is a condition where at-least two threads try to use memory at a single location. It occurs in write-write (WW), read-write (RW), and write-read cases (WR), though not read-read (RR). It is avoided by use of data-locks/synchronization. Syncing among threads in a warp/block is achieved in code while a global sync (all threads on device) is best done by exiting a kernel and launching a new one. Syncs/locks massively reduce program performance and should be avoided by not designing for threads to WW/RW/WR to the same address at the same time.

Bank-Conflicts occur when threads in a warp (i.e. half-warp) access data in the same bank of shared memory. As a result the process becomes serial, negating the benefits of parallel data-access that was likely intended. Banks are sequential 32 bits (4 bytes, i.e. a float, integer). To avoid this, have threads in warps access data in different banks, done in loose terms by organizing data accessed by adjacent threads– adjacently.

Coalesced Memory is the optimal way for threads in warps to read/write from/to global memory. Simply, threads should read adjacent memory addresses. Slightly more detailed, a warp should have a base address of 64/128/256 bytes and should access 32/64/128 bytes of contiguous memory. Often memory usage does not perfectly align with these specifications, so **padding** of contiguous memory chunks may be employed (i.e. adding well-sized, empty spacers between memory chunks to realign). Coalesced memory usage can vary across card architectures, some

allow finer granularity with varying performance, but this is more advanced.

Rule of 32. A very loose rule-of-thumb is to design programs around the number 32, as this is the typical number of SPs in a warp. This will tend to promote efficient memory transactions and avoid some common issues with minimal effort, though it is by no means perfect. For example, a Material Point Method simulation may group particles in bins of 32, with memory for the particles (e.g. position and mass, 4 floats each) column aligned in global memory (i.e. row for each particle property). Access for a bin thus takes 4 floats times 32 particles, or 4 * 4 * 32 bytes of memory. If well aligned, this results in 4 coalesced memory accesses of 128 bytes chunks. Kernel blocks launched for particles should have thread counts that are a multiple of 32, with each warp then executing a unique particle bin. This will substantially improve the speed of reading/writing particle data from/to global memory. More nuance appears with particle bins interacting with the MPM grid which would ideally be contiguous in memory. However, this is difficult because MPM allows particles to move throughout the grid so accessed memory addresses change frequently, negating coalesced access. This implies a need to spatially group these 32 particles for G2P and P2G optimization, possibly re-binning each time-step.

GPU Limits. There is a fundamental limitation for GPUs. If the bulk of the essential simulation data (i.e. all particle and grid-node data for MPM) can not exist on the device's memory, it will likely have poor performance. This derives from host-device transfers over a slow PCI-e bus (though they are improving). In past years limited global memory made GPUs untenable for many engineers accustomed to using high-memory CPU systems, however, GPU global memory size is growing rapidly, and more than one GPU can be used in a simulation.

Reaching the memory limit of a single GPU happens quickly in large simulations, necessitating dividing a simulation domain across multiple GPUs. A single GPU MPM simulation will tend to max out between 10 and 100 millions particles, depending on the grid-node resolution, hardware specifics, and memory optimization of the program. This requires a Multi-GPU code to support

quick messaging between Multi-GPUs (an execution cost) and the establishment of halo-regions within simulations (a memory cost). The next section will discuss basic logistics of Multi-GPU design and the hardware systems available to implement on.

3.4 *Multi-GPUs and HPCs*

Memory on GPUs is important, and limited. Multiple Graphics Processing Units (Multi-GPUs) can accelerate programs by increasing the available cores and memory. Moving from Single to Multi-GPUs is a conceptually basic extrapolation. However, it is not a simple step for most cases. While increasing core count is a catalyst to acceleration, communication between cores is a bane to parallel program performance. Tasks are ideally independent to avoid the cost of sending and receiving information across CUDA cores. This issue grows when using multiple GPUs, as sending information between these devices is even slower than between separate blocks, notably impacting performance.

Latency between devices (device-to-device) is significant. The key to Multi-GPU programming is: **(i)** Minimize communication, **(ii)** balance loads, and **(iii)** respecting scaling laws. The first concerns intelligent parallel design. The second proper partitioning of a program's data and computations across devices, and the third is a fundamental parallel obstacle discussed in the next section.

Common global memory sizes on each GPU range from 6 to 32 GB on NVIDIA devices. Global memory pools are reasonably large, but do not compete with standard High-Performance-Computing system (HPCs) CPU clusters which can have 128+ GB with ease through use of random access memory (RAM). Of course, host RAM also exists on GPU nodes and can pad the available memory, but its use would require transferring in and out of GPUs (host-to-device, device-to-host) which is a slow process governed by the PCI-e bus (orders of magnitude slower than GPU internal transfers).

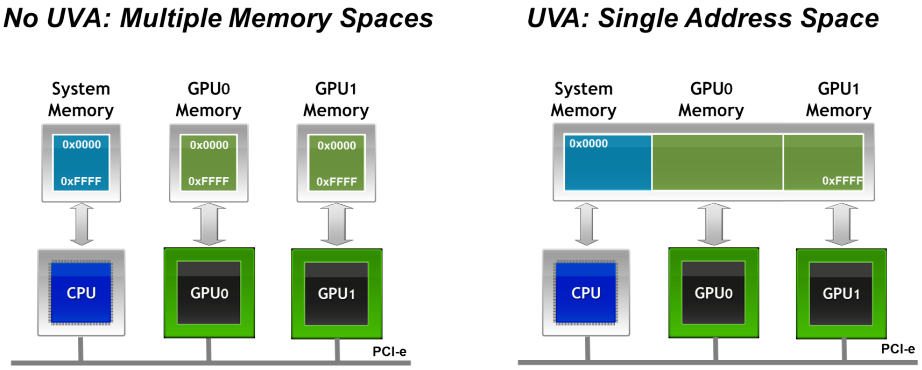

Fig. 3.7. Memory use in Multi-GPU systems using NVIDIA UVA. Memory management is greatly simplified when using multiple devices by employing NVIDIA UVA. Courtesy NVIDIA.

Table 3.1. HPC GPU Node Specifications

Parameter	Frontera TACC	Klone Hyak
Accelerators	4 Quadro RTX 5000 / node	4 RTX 2080 ti / node
GPU Memory	16GB GDDR6 / card	11GB GDDR6 / card
CPUs	2 Intel Xeon E5-2620 v4	2 Intel Xeon E5-2620 v4
RAM	128GB (2133 MT/s) DDR4	128GB (2133 MT/s) DDR4
Local Storage	144GB/tmp on 240GB SSD	144GB/tmp on 240GB SSD

Unified Virtual Addresses (UVA) in CUDA presents a simple way to orchestrate Multi-GPU programs. Figure 3.7 demonstrates how seperate memory pools on the host and devices can be virtually set as one large memory pool to simplify access. Note that this does not actually avoid latency of Multi-GPU communication; it just simplifies a programmers work.

High-Performance Computing systems (HPCs) allow us to make use of high-end, massively parallel hardware without investing in it directly. The two HPCs used in our project are located at the University of Washington (Hyak Supercomputer) and the University of Texas (TACC Supercomputer). Both feature nodes (smallest hardware unit available to access) equipped with multiple graphics processing units. Specification of typical nodes on either system are detailed in Table 3.1.

In modern times even a student can have multiple, if not dozens, of powerful GPUs at their disposal. However, as few truly make full use of one GPU, let alone two, attention should be paid to thoughtful design of program scaling so that resources are not misused in communal HPC systems.

3.5 *Scaling Laws*

More is not always better. Scaling a parallel numerical method across more and more computational cores often results in diminishing, and even decreasing, returns. While a single GPU MPM program is already scaled across thousands of CUDA cores, we may further amplify this by adding more GPUs. However, scaling laws are observed across our engineering simulations as we attempt to make them bigger and faster via hardware parallelism. The question is if there is marginal utility derived by adding another computational core, GPU, or node to a simulation. Two primary modes of scaling exist: **(i)** Strong and **(ii)** Weak. Respectively they concern the ability to "accelerate" a fixed "size" simulation and the ability to "grow" a fixed "execution time" simulation. For the Material Point Method this can roughly translate to accelerating a simulation with a fixed set of particles and grid-nodes by a factor related to GPU count (strong-scaling) or increasing the amount of particles and grid-nodes in a simulation with a constant execution time through a factor related to GPU count (weak-scaling).

3.5.1 Strong Scaling - Amdahl's Law

Scaling simulation time inversely with available hardware is a primary concern for HPC systems and parallel programs in general. How does adding an additional processor, or GPU, affect run-time? In other words, can we halve the simulation time of a MPM simulation if we double the number of GPUs?

Strong-scaling, Amdahl's Law, measures the marginal utility found from adding processors in

terms of simulation acceleration as

$$S = \frac{1}{(1-P) + \frac{P}{N}} \ , \tag{3.1}$$

where S is program "speed-up", P is proportion of the program that is "parallel", and N is the number of computational "cores".

As processor count approaches infinity ($N \to \infty$) one may expect speed-up to be limited by a fundamental barrier associated with the program's parallel proportion as

$$S = \frac{1}{1-P} \ . \tag{3.2}$$

A program will never be perfectly parallel ($P = 1$) due to overhead operations, such as host-side processing and inherently sequential operations. Thus, speed-up has a finite-limit regardless of processor count.

In practice, this barrier is not reached. The effect of ever-increasing processor count for numerical simulations has diminishing, and even decreasing, returns not accounted for by the idealized strong-scaling law. In some cases, increasing N may reduce P if communication between GPUs requires serial management. Communication latency between GPUs is also not considered in Amdahl's law and may completely negate theoretical speed-up. For MPM, where useful simulations might take days or weeks to complete on CPU systems, good strong-scaling behavior as GPUs are introduced is desirable.

3.5.2 Weak Scaling - Gustafson's Law

Scaling simulation size while maintaining execution time is another topic. How does adding an additional processor, or GPU, affect this? In other words, can we double the number of particles and grid-nodes in an MPM simulation without increasing simulation time if we double the

number of GPUs?

Weak-scaling, Gustafson's Law, measures the marginal utility found from adding processors in terms of simulation growth relative to constant execution time. It is phrased as

$$S = \frac{s + NP}{s + P} = s + NP = N + (1 - N)s \ , \tag{3.3}$$

where S is program "speed-up", P is proportion of the program that is "parallel", N is the number of computational "cores", and s is the serial proportion of the program. Note that $s + P = 1$ and $s = (1 - P)$. Weak-scaling is slightly harder to grasp than strong-scaling but is arguably more critical for programs which require increasing size over faster speeds (i.e. MPM, where simulation size limits many results).

Weak-scaling implies that instead of "speeding-up" a simulation with more processors, inherently limited by the sequential part of the program (s), we should seek to match the execution time of a small program (T_{Small}) ran in serial ($N = 1$) with a larger program of execution time (T_{Large}) on multiple cores ($N > 1$). Speed-up is defined as $S = \frac{T_{\text{Large}}}{T_{\text{Small}}}$. If simulation size linearly increases with cores N used, then perfect weak-scaling is when $S = 1$ at all sizes.

Realistically, S will drop below 1 as programs grow extremely large despite providing more and more cores to accelerate the parallel portion. This effect is greater for introduction of slow communication between core groups (i.e. Multi-GPU communcation) but isn't explicitly considered here.

Many serial CPU MPM programs simulate a million particles in a day, but simulating 50 million particles might take over a month– untenable for most engineers. Weak-scaling with GPUs can maintain execution time while having massive particle counts, something we achieve in our own study.

3.6 *Final Remarks*

This chapter addresses one of the flaws of MPM: computational time. For this purpose a crash course in Graphics Processing Units (GPU), CUDA, Multi-GPUs (Multi-GPU), and High-Performance Computing systems (HPCs) was provided. Optimistic forecasts for growth of these hardware infrastructures in engineering are drafted.

This introductory note on GPU programming can be intimidating. Most of our efforts going forward are to abstract away as much of the low-level CUDA operations and manual design as possible. An engineer's place is neither in fine-tuned register/local-memory optimization of a kernel, nor in careful use of shared memory in blocks to avoid bank conflicts. Rather, we aim to squeeze the maximum amount of performance from CUDA programs, both in execution time and implementation time, with the absolute minimum amount of effort.

Equipped with basic GPU competency, the next chapter brings our engineering methods, namely the Material Point Method (MPM, Chapter 2), onto these devices.

Chapter 4

BRINGING THE MATERIAL POINT METHOD TO MULTI-GPU

4.1 *Introduction*

Making the Material Point Method (MPM) algorithmically manifest on single or multiple graphics processing units (GPUs) requires several basic steps. Each GPU device holds thousands of parallel computational cores that may accelerate the numerical method by over 100x if properly employed. Therefore, GPU code must be carefully crafted to take advantage of the resources they provide.

An argument is made for engineers to leverage open-source software and resources from other fields instead of starting from scratch. A simplified guide to how other engineers may co-opt open-source software not originally intended for engineers is desirable.

Claymore Multi-GPU MPM, a graphical program and our chosen open-source software foundation, is surveyed. Rebuilding the user interface, making changes to the low-level code, and implementing additional methods are undertaken to increase accessibility to engineers. In sum, these modifications lead to ClaymoreUW, our open-source Multi-GPU MPM code for engineers. All the simulations presented in this dissertation use ClaymoreUW.

While retaining our respect and due credit to the original Claymore code, we describe in this chapter the added value of ClaymoreUW and provide documentation for users and developers alike.

4.2 *Single-GPU Material Point Method*

The Material Point Method in its various forms (MPM, Chapter 2) is a numerical method reasonably suited to parallel computation. The key-word, reasonably, is because particles must scatter information to shared, stationary grid-nodes in their supports (i.e. the local spatial region a particle influences, set by shape-functions). Weighted values on particles (e.g. mass, momentum) are summed (reduced) at a grid-node numerically to integrate, solving the equation of motion (EOM) at a local point of space. Solutions to the EOM at each grid-node in a support are then gathered onto relevant particles in the next time-step. Finally, particles independently advect (recording their updated postions) and solve their material models (recording deformation, scattering stresses and internal forces).

Note use of the terms gather/broadcast/scatter/reduce. These are basic parallel programming operations that form the spine of a parallel MPM implementation. A summation reduction could be an atomic addition between blocks, or a shuffle within warps. Terms "shared" and "independent" lead to a formative memory scheme. A grid region "shared" by particles can load into shared memory for parallel use at a CUDA block scope, particles can record "independent" information in their own global memory arrays or in registers at the CUDA thread scope.

Bottlenecks of computation are found anytime consensus is required (P2G, Grid Update), necessitating independent particles to aggregate information on a shared grid which they must draw from the following time step, breaking their parallel nature. Though particle advection, material models, and many boundary conditions are executed easily in parallel, transfer of results between particles and grid-nodes is a prerequisite. G2P and P2G transfers are heavy-weight operations. Choice of shape-function has a significant impact on performance. Linear, Quadratic B-Spline, and Cubic B-Spline respectively require a single particle to gather from and scatter to 2x2x2, 3x3x3, and 4x4x4 grid-nodes in 3D simulations. As the particle count in a cell (PPC) can often range from 0 to 64, hundreds of particles may access a single grid-node's information each

time-step which is prone to data-race errors unless many expensive syncs/locks are applied, or intelligent design is employed. Further, bank conflicts pose a significant barrier to performance if a parallel MPM program is not optimized at a low-level to avoid them.

4.3 *Multi-GPU Material Point Method*

A Material Point Method (MPM) implementation on multiple graphics processing units (Multi-GPUs) is a conceptually basic extrapolation of Single-GPU MPM. However, it is not a simple step. An extra layer of communication is established for separate GPUs to share results. This is referred to as Multi-GPU consensus.

Independent GPU information is regarded as Non-Halo data (i.e. particles or grid-nodes that influence a single GPU's partition). Shared GPU information is Halo data (i.e. data in multiple GPU partitions, "halo-regions"). The former is parallel to other GPUs while the latter must reach Multi-GPU consensus (P2G, Grid Update). Because Halo data may become Non-Halo data within a GPU as simulations progress, and vice versa, we view Non-Halo data as non-parallel to other GPUs over multiple time-steps. The key to Multi-GPU optimization of MPM is parallel execution of Halo and Non-Halo tasks within a time-step. Halo execution and communication (i.e. send and receive data between all GPUs) can be done in parallel to Non-Halo execution. Afterwards, each GPU reduces its aggregated Halo and Non-Halo data to achieve consensus for the MPM algorithm (P2G, Grid Update).

Well designed Multi-GPU MPM simulations have an order of magnitude more information (e.g. particles) as Non-Halo than Halo, meaning Halo execution and communication can asynchronously execute before Non-Halo execution completes. Subsequent reduction is thereby only dependent on Non-Halo performance, meaning a Multi-GPU code can outmatch Single-GPU performance despite device communication latency. Scaling laws discussed in Section 3.5 are important. Strong-scaling may accelerate fixed-size MPM simulations with additional GPUs if

Halo data and partition overlap growth are minimized. Weak-scaling may multiply MPM simulation size without increasing computation time if Non-Halo data size is kept constant across added GPUs, and assuming Multi-GPU PCI-e communication isn't overloaded.

4.4 *Existing Software for a High-Performance Material Point Method*

The primary obstacle of a parallel MPM implementation is potential data races between the computing cores writing to and reading from common memory addresses concurrently (Chapter 3) for the interpolation from particles to nodes (i.e. grid-to-particle, particle-to-grid). To promote the efficiency of MPM and enlarge its modelling capabilities, parallel computing on CPUs and single-GPUs has been explored by a limited, but growing, number of researchers, as remarked by Dong and Grabe 2018. Huang et al. 2008 and Zhang et al. 2010 developed a single-CPU parallel framework using a loop-based parallel library OpenMP, which constituted some of the earliest parallel approaches to the Material Point Method, although Parker 2006 debuted their multiple-CPU parallel approach a few years prior. Not long after, Ruggirello and Schumacher 2013 presented another multiple-CPU parallel framework with Message Passing Interface (MPI). The modern GPU approach first saw significant optimization and meticulous study perhaps in Gao et al. 2018a, who presented a parallel strategy on a single GPU based on the Compute Unified Device Architecture (CUDA, see Chapter 3). The approach of Gao et al. 2018b was evaluated and adjusted in the works of Hu et al. 2018 and Wang et al. 2020b, the latter of which extrapolated to Multi-GPU parallelism using peer-to-peer device messaging in CUDA. Fei et al. 2021b offered ways to improve Wang et al. 2020b's approach by performing low-level optimizations (e.g. concerning GPU assembly instructions) and seeking to minimize the heavy-load of data-structure operations compared to the MPM algorithm, specifically for low particle count simulations (e.g. below 10 million) at real-time or nearly real-time simulation speeds. Recently, Qiu et al. 2023 introduced a Multi-GPU, Multi-Node MPM framework based on the design philosophies of Hu

et al. 2019b, Wang et al. 2020b, the Kokos and Cabana libraries, among other inspirations. They achieved billion particle simulations with visual clarity.

Concerning the numerical partitioning across computational bodies, Huang et al. 2008 and Zhang et al. 2010 categorised the particles in groups by domains, and then proceeded to make-parallel the interpolation of the particles over the groups. Similarly, though designed for multi-CPU execution, Parker 2006 and Ruggirello and Schumacher 2013's works took computational domain and decomposed it into subdomains corresponding to the available CPUs and CPU cores. Over each subdomain the MPM algorithm was parallelized (each computational body handling a separate subdomain), while the interpolation of the particles in each subdomain was calculated sequentially on said computational body. The GPU work of Gao et al. 2018a created a list of particles associated with each node. Using this list, they were able to parallelize MPM for the interpolations over the nodes. This brief look at forerunners for Multi-Core CPU, Multi-CPU, and single GPU MPM implementations sets the stage for modern software now available and widely used. Further, they showed that there is a great amount of flexibility in the approaches that may be taken, and that none is necessarily superior until benchmarked as such.

CB-Geo MPM by Kumar et al. 2019 is an open-source, high-performance material point method software. It is designed for engineering use, namely geotechnical and structural. The software is optimized and well documented, making it almost ideal for any engineer who is starting out on the path of high-performance MPM. However, the code is not designed for GPUs, let alone Multi-GPU. It is a significant step-up from a serial MPM code and even most parallel MPM codes as in Parker 2006, Huang et al. 2008, and Zhang et al. 2010. While it was not the software used in our own project, we gave it serious consideration and have future intentions of coupling CB-Geo MPM into software endeavors such as HydroUQ for HPC uncertainty quantification problems and we would like to better align our user-interface schema with their intuitive example. Still, the study of tsunami-driven debris impacts required the absolute cutting-edge of speed, even if it is at

the cost of convenience and a preexisting track-record, hence we pursue a open-source code discussed later. We highly recommend this software to anyone who is not ready to make the jump to Multi-GPU MPM as CB-Geo is excellent for engineers and is actively in development.

Taichi by Hu et al. 2019b is an open-source, imperative, parallel programming language for high-performance numerical computation. In other words, it is a language designed for optimized numerical simulations. It is embedded in Python and uses just-in-time (JIT) compiler frameworks to offload the compute-intensive Python code to the native GPU or CPU instructions. The language has broad applications spanning real-time physical simulation, numerical computation, augmented reality, artificial intelligence, vision and robotics, visual effects in films and games, general-purpose computing, and more, though we are only concerned with numerical computation for engineering simulations in this project. Taichi has language syntax that is nearly identical to Python, making it a simple language to learn, and has an example 88 line implementation of the MLS-MPM method by the same lead author, Hu et al. 2018. Taichi is also well integrated into the Python ecosystem, including NumPy and PyTorch. Taichi is flexible and provides a set of generic data containers, an effective mechanism for composing hierarchical, multi-dimensional fields. This can cover many use patterns in numerical simulation (e.g. spatially sparse computing) and abstracts away the complexity of GPU data-structures. The JIT compiler automatically compiles Python functions into efficient GPU or CPU machine code for parallel execution. Currently, Taichi supports most mainstream GPU APIs, such as CUDA and Vulkan (i.e. NVIDIA and AMD devices, and perhaps Intel GPUs as well), along with standard CPU systems.

The Karamelo Multi-CPU/GPU code by Nguyen et al. 2023 is another interesting open-source development. Though its release was near the end of our project, and so we were unable to fully survey the capabilities it provides, it seems to be a very interesting engineering and physics focused code that we would recommend others interested in high-performance MPM consider for their own projects.

With these key players in high-performance MPM now known, we introduce the Multi-GPU MPM code, the graphics-focused Claymore MLS-MPM code by Wang et al. 2020b, of which we built our own open-source engineering advancement, ClaymoreUW, on top of.

4.5 *Claymore - Overview*

Claymore is a recently developed, open-source, Multi-GPU, computer-graphics software (Wang et al. 2020b) for running MLS-MPM (Hu et al. 2018) (Sec. 2.5). It boasts a 100x acceleration over optimized single-CPU codes and has been tested at particle counts in the ball-park of 100 million using 4-8 consumer GPUs in single-precision. We have since retooled the code into what is called ClaymoreUW, and validated it for double and mixed-precision engineering simulations with over a billion particles on just 4 GPUs.

Workflow in Claymore is similar to most MPM software, with a few key distinctions originating from its Multi-GPU oriented nature. The list below and subsequent sections identify the most prominent aspects of Claymore, including:

- **GPU oriented MPM Kernels.** All MPM algorithm steps are coded with respect to low-level GPU optimization. Most code is purely for squeezing out efficiency while the actual MPM algorithm is a minority of lines.

- **Multi-GPU Communication.** Simulations are partitioned across multiple GPUs. In "Halo-Regions", areas where GPU partitions overlap, the software must read/write information between devices without invalidating constraints of the MLS-MPM algorithm. Non-Halo tasks are done in parallel to Halo tasks, subsequently reduced as an aggregate each time-step.

- **Fused G2P2G Operation.** For extra efficiency, G2P + Advection + P2G are fused into one-step. This leads to a 2x improvement in speed for MLS-MPM and avoids storing the

affine velocity matrix from APIC on particles (Sec. 2.4). However, lack of global synchronization between G2P and P2G means certain things common in regular MPM codes are harder to implement here (e.g. any technique requiring particles to reference each other before P2G), and vice versa. Finite elements (Section 5.13) are particularly unwieldy to implement, though our lab has added this functionality along with expanded advection schemes such as Affine-Separable Fluid-Implicit Particles (ASFLIP, Section 5.6.7).

- **Data-Structure Design.** This code uses complex data-structures optimized for Multi-GPU use, fashioned similar to the approach Taichi uses. However, simply trying to retrieve information from a specific particle requires extra effort. Our lab has simplified user interaction with complex Multi-GPU data-structures in ClaymoreUW.

- **Computer Graphics Bias.** The Claymore software is not engineer friendly, though our project is improving this aspect. In its original version, graphical use was prioritized at every step, resulting in poorly validated physical behavior, engineering method implementation difficulty, and unwieldy I/O. Basic errors have been corrected by our group and we add functionality to expand the software to engineering use.

4.6 *From Graphics to Engineering*: *ClaymoreUW*

This section details some of our novel contributions and discusses the methodology taken to co-opt a Multi-GPU animation code for engineers (Wang et al. 2020b).
The low-level code optimization required to build an effective Multi-GPU numerical tool is high, but the benefits are greater. Leveraging the work of skilled programmers (i.e. graphics researchers) allows us to tap into optimized Multi-GPU codes with minimal overhead. Retooling a graphical software for engineering use is not a small task, but it is easier than starting from scratch. Here a summary of changes made to Claymore (Wang et al. 2020b) to make the software

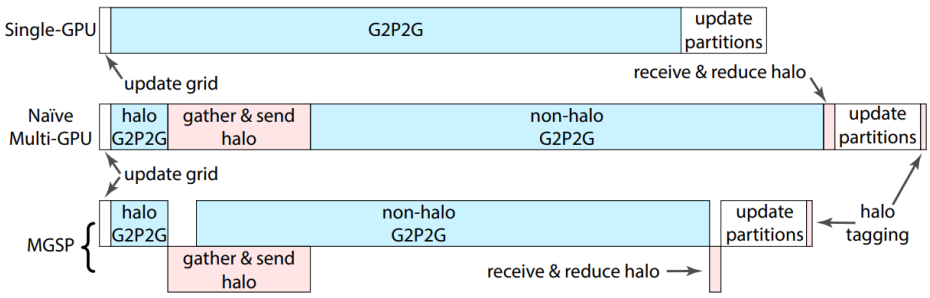

Fig. 4.1. Instructions for Single and Multi GPU Material Point Method - Claymore Software. Instruction pipe-line for both Single and Multi-GPU MPM in Claymore. Intelligent Multi-GPU MPM design uses CUDA asynchronous properties to hide latency, approaching Single-GPU speeds. Courtesy of (Wang et al. 2020b).

ClaymoreUW. We include reasons for doing so. Also, a more general guide for retooling is provided for developers.

We have made many changes to the ClaymoreUW software that span implementations of various algorithms, design of high-user-interfaces, optimization of low-level GPU instructions, and general bug-fixing. Novel contributions made by our lab to the code are:

1. Implementation of finite element coupling (Section 5.13),
2. FLIP advection (Section 5.6.2),
3. PIC-FLIP mixing (Section 5.6.3),
4. Affine-Separable Fluid-Implicit-Particles (ASFLIP, Section 5.6.7),
5. B-Spline F-Bar volumetric antilocking (Section 5.7),
6. Removed forced use of low-precision intrinsics (e.g. -fast-math flag, primitives like __fdiv()),

7. Enabled double-precision,
8. Enabled mixed-precision,
9. Fixed a variety of bugs (e.g. data-races for pre 7.5 compute capability, improper reservation of shared memory),
10. Loosened certain preset restrictions on CUDA kernel attributes and cache configuration (e.g. specify the l1 cache and carve-out ratios, specify min and max blocks and threads for compile-time optimization),
11. Tracking of memory usage for improved user-experience in memory intensive simulations,
12. Substantially overhauled user-interface,
13. Provided defaults for input values that are unspecified,
14. Introduced basic logging (info, tips, warnings, and errors),
15. Allow for outputs of 100+ variables at run-time,
16. Implemented numerical sensors for grids (e.g. load-cells on rigid boundaries),
17. Implemented numerical sensors for particles (e.g. wave-gauge on water),
18. Parallel optimized the arbitrary reduction operations on said sensors,
19. Made sensors accessible with the run-time scripts,
20. Implemented moving boundaries with either basic input script motion or load-in of a motion file,

21. Provided a simple way to divide particles across GPUs in Multi-GPU runs using partition domains,

22. Modified inputs to take care of any offsets and scaling required for the simulation to run (e.g. off-by-2 condition, makes it easier to use if you aren't familiar with the back-end code),

23. Set desired spatial-step (i.e. grid-cell length) at run-time,

24. Set desired time-step at run-time, automatically over-ride time-step to support desired CFL condition specified for a user-input material,

25. Track quantities on specific particles at high-frequency,

26. Output grid and particle sensors at high-frequencies,

27. Added determination and output of grid energies (kinetic, kinetic FLIP, gravity, strain FBAR),

28. Added determination and output of particle energies (kinetic, strain),

29. Expanded Multi-GPU communication for FLIP/FBAR/Coupled U-P usage,

30. Added digital twin of the Oregon State University's Large Wave Flume (OSU LWF),

31. Added digital twin of the Oregon State University's Directional Wave Basin (OSU DWB),

32. Added digital twin of the University of Washington's Wind-Air Scaled Research Facility flume (UW WASIRF),

33. Added digital twin of Waseda University's Tsunami Wave Basin (WU TWB),

34. Added digital twin of the U.S. Geological Survey's Volcanic Hazard Debris-Flow Flume (USGS Lahar Flume),

35. Added linear piston wave-makers,

36. Added gravity dam-break wave-makers,

37. Adjust gravity (i.e. external acceleration field) to be an alterable vector at run-time,

38. Above item also, in effect, allows simulations to easily change the orientation of the grid (i.e. at an angle)

39. Added box boundaries

40. Added arrays of boundaries

41. Added spacing specification for array boundaries

42. Added sphere geometries

43. Added cylinder geometries

44. Added box geometries

45. Added Boolean add/subtract/intersect/difference/union operations for geometries

46. Added run-time geometry load in with CSV and SDF files

47. Unified scheme to organize information of particles of any material for input and output (i.e. minimize hard coding and template specialization)

48. Run-time adjustment of memory reserved for particle output data-structure
49. Run-time specify domain to reserve memory for the 3D to 1D hash-map
50. Output check-point functionality (all pertinent information of a simulation)
51. Input prior check-point to resume a simulation
52. Fixed integer overflows that can occur when reaching particle simulations in the billions,
53. Standardized aspects of the spatial domain floating point representation to minimize relative floating point error for simulations at different magnitude scales
54. Run-time Froude similitude application on desired aspects of simulation,
55. Run-time check of proper data-type and data-bounds for input variables (e.g. is a length measure a positive number),
56. Added timed boundaries
57. Implemented J-Bar Fluid (JB Fluid, Section 5.11)
58. Implemented the Precision-Accelerated J-Bar Fluid (PA-JB Fluid, Section 5.12)
59. Implemented F-Bar volumetric antilocking mixing (Section 5.8)
60. Implemented F-Bar volumetric antilocking for the G2P2G algorithm (Section 5.10)
61. Implemented F-Bar volumetric antilocking for the MLS-MPM method (Section 5.9)
62. Implemented the Sub-J, i.e. recentered J, precision improvement for any material which uses the deformation Jacobian J (e.g. J Fluid, JB Fluid, PA-JB Fluid),

63. Fixed the weakly compressible fluid material model (J Fluid) to use the formal Murnaghan-Tait equation of state, as opposed to the graphical variants of Cole's equation, etc., which are often seen in graphics codes but are not equivalent to engineering counter-parts,

64. Fixed warp-reduction undefined data-race on some compute capabilities for the grid-update

65. Fixed improper reservation of shared memory span for the grid update

These changes are described in detail in their respective sections, when applicable. Each presented a technical challenge as they needed to be seamlessly implemented into an existing, highly-optimized Multi-GPU code that was designed primarily for animators using MLS-MPM. The original Claymore is fast, and it does run MLS-MPM, but it has very few features in the sense that any simulation that goes beyond loading in a provided list of particles and setting the material on one GPU would require changes to the source code. Further, it only ran MLS-MPM, which isn't enough for engineers on its own.

MPM advancements from the computer graphics literature (e.g. Sections 2.5 and 5.6.7) are poorly validated by engineering standards. Animators are primarily concerned if a simulation looks good (i.e. roughly accurate particle positions and velocities) and accept visual results as validation of a mathematically sound derivation. However, math that is good on paper may not result in quality engineering simulations.

Computing decent positions and velocities of particles is simpler than getting physically sound results. Force, stress, and other engineering aspects (e.g. nonlinear material parameters) are far more difficult to simulate accurately and must be carefully validated against reality.

Compile-time constants are used extensively in Claymore. This is a problem for physical simulations since in its original form minor modifications of a simulation required edits in the source code and a full recompilation. In our expanded version of the code many pertinent

simulation parameters (e.g. time-step, gravity, particles-per-cell, material model, etc.) have been changed to run-time, now set in a simple input script.

A common error in graphical codes is scene-scaling. Animators often scale just gravity and density, assuming that this will scale all physical laws in the scene appropriately. For instance, the Claymore MPM code creates erroneous stress values when the scene is scaled beyond a 1x1x1 box. This is due to APIC's stencil coefficient for quadratic B-Spline shape-functions, D_p^{-1}, not being adjusted in the code. Errors like this are likely missed by graphics researchers because they tend to focus on visual validation of their codes (i.e. are particle positions roughly correct). Errors in stress-strain behavior require further validation that necessitates a strong foundation in physics to undertake (i.e. engineers). For example, validating pressure-wave behavior of debris impacting rigid walls as elastic moduli varies in scaled scenes is simple for engineers but rarely in scope for animators. Small errors like this are not difficult to fix once identified, but the presence of multiple such errors in many open-source graphics codes makes the software give incorrect results "out-of-the-box". This may be why so few engineers use highly-optimized graphics codes, despite the fact that they run engineering algorithms, as test simulations will almost certainly not be up-to engineering standards.

4.7 *ClaymoreUW - Compilation*

The Claymore code's original source is at:

https://github.com/penn-graphics-research/claymore/

The modifications we've made in ClaymoreUW are kept at:

https://github.com/JustinBonus/claymore/

If you eventually publish research based on our modified Claymore, please cite the original authors, as well as our eventual publication on our open-source modifications. Original author citation as Wang et al. 2020b

Here is a brief, but generally effective compilation guide for *NIX based systems:

1. Use modern versions of CMake (3.10+ typically). Use a modern compiler. We have used used GCC 7.5+, 8, 9, 10, and 11 but any above 7.5 should work. People on Windows systems have used MSVC successfully though we have not attempted a Visual Studio build with MSVC on windows.

   CUDA version 10.2+ is required, we have compiled it with CUDA 10.2, CUDA 11.0, CUDA 11.4, and CUDA 12.2.

   Ensure a modern GPU is used and know the "compute capability" of the GPU. ClaymoreUW should work for any NVIDIA Compute Capability 6.1+ (most cards after 2016). We have primarily run on 6.1, 7.5, 8.0, 8.6, and 9.0.

2. Go into ./setup_cuda.cmake and change the TARGET_CUDA_ARCH flag to apply to your GPU card. Go-to the line where it says: "set(TARGET_CUDA_ARCH -arch=sm_61)", and change sm_61 to sm_XY where XY is your compute capability. Advanced: Change the line so it compiles directly for a single GPU card architecture (the code will be faster) or have it compile for a range of cards (code will be slower the first time running, but will work on more GPU cards).

3. Check that ./Projects/FBAR/settings.h has some basic values set properly:

   g_device_cnt = number of GPUs you are using, e.g. 1 (do not use 2+ if there are not 2+ available GPUs).

   g_models_per_gpu = number of models per GPU, e.g. 3 if you want to have a fluid, granular material, and a solid model running on one GPU

   g_max_particle_num = max number of particles, e.g. 2000000 for 2,000,000 particles on a GPU. (resizes as simulation runs, must be correct at start)

g_max_active_blocks = max number of active "blocks" on a GPU, e.g. 10000 for 10,000 blocks, which corresponds to 64 * 10,000 active grid-nodes (have mass) around the particles. (resizes within reason as simulation runs, must be correct at start). This has a big effect on memory/speed.

DOMAIN_BITS X <- X is the "bits" for the grid, e.g. 10 = 2e10 grid-nodes in any direction (Total grid-nodes in 3D = 2e10 * 2e10 * 2e10). There are 8 buffer grid-nodes on either end of each dimension, so for a 25.2 x 4 x 7.5 meter simulation the max length (25.2) is the bottleneck for grid-cell spacing dx. i.e. 25.2 meters / (2e10 - 16) = smallest possible dx = 0.025 meters. Increase DOMAIN_BITS to use more memory but allow higher-res simulations, or the inverse to save memory.

MAX_PPC X <- X is maximum particles-per-cell, e.g. 64. If you have a 8 PPC material that is compressible, you may expect as many as 64 PPC to happen in areas of high compression, so set it to something like that. Must be a power of 2 (...8, 16, 32, 64, 128...) in Claymore, ClaymoreUW allows non-power of 2 values. This number has a big effect on memory/speed.

4. Make sure CUDA is loaded on the computer, "nvidia-smi" will show available GPUs and the NVIDIA Driver version. Try "nvcc –version" to check your CUDA NVCC compiler version. On TACC systems (or any HPC), it may be neccesary to run something like "module load cuda/11.0" or "module load cuda/11.4" first to load the correct CUDA + NVIDIA drivers on a node before compiling. Again, use "nvidia-smi" to make sure GPUs are available on the node.

5. Compile using the steps given in https://github.com/penn-graphics-research/claymore/. Start by making a "build" folder in the claymore source folder. Starting from the ./claymore folder, you can run "cd ./build ; cmake .. ; cmake –build . ; cd ../Projects/FBAR" for GCC compilers.

6. To run the program use the command "./fbar" in the ./claymore/Projects/FBAR folder. This defaults to using the "scene.json" file in the same folder. To use a specific scene file you create, use "./fbar –file=scene_YourFile.json". The program will give you various info, warnings, and potential error messages as it initializes, the latter two require pressing ENTER on your keyboard to continue. Output files will be in the same directory that your terminal is in.

A set of tips that may help with getting the most out of ClaymoreUW and ensuring the set-up goes smoothly:

1. If using a GPU that is good at double-precision computation (e.g. NVIDIA Tesla series) expect a 8-32x performance boost over GPUs that are not good at double-precision (e.g. NVIDIA GTX/RTX/Workstation series), but they are all fairly fast. We recommend modern HPC systems such as TACC Lonestar6 if possible, it possesses 3 NVIDIA A100 GPUs per node (40 GB memory each, so 120GB total GPU memory). We have ran up-to 500 million particles on this system.

2. It is a very common issue for people that their CUDA NVCC version, CUDA Toolkit version, NVIDIA Driver version, and/or their GPU compute capability are mismatched in some way. This often happens when people install CUDA/NVIDIA Drivers incorrectly, in pieces, or improperly uninstall older versions before updating. This is not an issue with Claymore specifically, as it will prevent you from compiling any CUDA code. Check if you can compile and run NVIDIA's sample programs such as "/usr/local/cuda/samples/5_Simulations/fluidsGL/fluidsGL" (file-path may vary for you) before trying Claymore.

3. If using an IDE like Visual Studio Code (recommended) make sure your "Include" path is

set properly so that the program can compile/link the library dependencies. Also make sure it can find your CUDA install location.

4. Ensure you have modern C++ versions, e.g. C++ 14, 17, and 20+ should work (compilers like GCC will typically install these for you) with C++ 17 and higher being preferred.

5. We would recommend setting up PartIO and/or SideFX Houdini for visualization. SideFX offers free apprentice licensing of their software.

6. The RapidJSON library dependency causes compilation complications for some users. I haven't had any, but you can update to newer RapidJSON versions in "./setup_externallibs.cmake" to potentially fix issues.

We have compiled on a personal laptop (NVIDIA 1060m Max-q), desktop (NVIDIA 780 ti, may now be deprecated), TACC Frontera (NVIDIA Quadro 5000), TACC Lonestar6 (NVIDIA A100 40GB), UW Hyak Mox (NVIDIA Tesla P100), UW Hyak Klone (NVIDIA RTX 2080 ti), Texas A&M University's ACES (NVIDIA H100 PCIe 80GB). For help compiling or running ClaymoreUW on these systems, or a comparable one using a Linux operating system, contact me at bonusj@uw.edu for reasonable assistance.

The original Claymore code is fast but has limitations in its accuracy and usability. We have built in a lot of user-interface changes (almost everything controlled by a JSON script similar to other engineering codes, and there is no need to recompile every time input is modified) and fixed physical inaccuracies (incorrect material law implementations, dimension scaling errors, general precision improvements, etc.) and CUDA issues (overwriting memory bounds, undefined behavior for some warp operations on newer architecture GPUs IIRC, not enabling full use of Shared memory from the L1 cache/carve-out for critical GPU kernels, etc.) have been mostly fixed. There are a couple of geotechnical material laws (Drucker-Prager, Non-associative

Cam-Clay) that will need to be looked at a bit more closely (some "short-cuts" built into them that may need to be fixed).

Likewise we have added some materials like Neo-Hookean and another researcher, Dr. Javier Mroginski, is working on a Coupled U-P soil model. Multiple algorithms have also been added (FLIP, ASFLIP, F-Bar volumetric anti-locking, and basic coupling with Finite Elements). Additionally, multiple particle models per GPU of potentially different materials/algorithms are now supported in Multi-GPU usage (e.g. MPM sand with ASFLIP + FEM solids + MPM water with F-Bar anti-locking). We are currently looking to scale Claymore from Multi-GPU to Multi-Node + Multi-GPUs in the near-future. Nearly all features are currently accessible through a simple JSON script for Non-GPU programmers, the only instance that usually requires changes in the code is if adding a new feature or attempting to run a very memory intensive simulation.

4.8 *ClaymoreUW - User Guide*

Here we describe how a user can get started working with ClaymoreUW in their own custom simulations. This is not a technical task, as modifying input scene scripts in JSON is a very common and reasonably intuitive file format. The only true requirement is that the JSON "schema" is used, i.e. use the same variable names and structure as we will describe in this section.

Before continuing, make sure you have compiled and have a system capable of running ClaymoreUW by going through all steps in Section 4.7. Also, ensure that you can run an example simulation provided in "scene.json" before struggling to debug any custom input scene.

Basic use of ClaymoreUW is entirely accessible through: **(i)** an input JSON script, and **(ii)** potentially minor changes of a few variables in file 'settings.h' in the appropriate Project folder. Construction of the input scene JSON script will be described here, along with common changes made to the settings file.

We will break-down each possible component in an input scene file, which must be formatted as

JSON. By default, they should be placed within the same folder as the ClaymoreUW application, though this is not necessary if passing in the path to the input file when the application is called. An example scene file, scene.json, is as follows:

```
{
    "simulation": {...},
    "models": [
        {
            "device": {...},
            "material": {...},
            "algorithm": {...},
            "geometry": [{...}, ...],
            "output": {...},
        },
        ...
    ],
    "grid-boundaries": [{...}, ...],
    "grid-sensors": [{...}, ...],
    "particle-sensors": [{...}, ...]
}
```

4.8.1 Input Parameters -Simulation

The `simulation` object contains basic simulation settings. For instance, the max domain size to simulate in, the gravity vector, the default time-step, and the default grid-cell size.

```
"simulation": {
    "domain": [8, 8, 8],
    "default_dx": 0.05,
    "gravity": [0,-9.81,0],
    "time": 0,
```

```
    "default_dt": 1e-3,
    "fps": 30,
    "frames": 10,
    "save_path": "./",
    "save_suffix": ".bgeo",
    "particles_output_exterior_only": true,
    "froude_scaling": 1.0
},
```

where variables are defined as:

- `domain`: Defines max dimensions of the simulation domain in each direction of a 3D simulation. Simulations that try to exceed domain will fail. Array with three elements. Each element must be a positive number greater than zero. [meters].

- `default_dx`: The default grid-cell length for the Material Point Method's background grid, Δx. Must be a positive number. [meters].

- `gravity`: The 3D gravity vector. Array with three elements. Each element must be a number. [meters / second^2]

- `time`: Initial time datum when the simulation starts. Must be a number. [seconds]

- `default_dt`: The default time-step for the Material Point Method, Δt. Will be over-ridden (i.e. decreased) if required to maintain stability for pressure wave speed of a material in the simulation. Must be a positive number greater than zero. [seconds]

- `fps`: Frames-per-second to output full particle files from the simulation. Must be an integer greater than zero. [seconds^{-1}]

- `frames`: Total number of frames to output during the simulation. Divided by the frames-per-second, this defines the total time duration of the simulation. Must be an integer greater than zero. [ ]

- `save_suffix`: File suffix to save the full particle output files with. Supports geometry (.geo) and binary geometry (.bgeo) currently, will be expanded to CSV, VTK, TXT, etc., later through use of PartIO library. Must be a string that begins with a period. [ ]

- `particles_output_exterior_only`: Whether or not to output only the particles near the exterior surface of a given full particle file. Saves a ton of memory in big simulations, but does not output all particles. Must be the Boolean true or false value. [boolean]

- `froude_scaling`: Whether or not to apply Froude similitude scaling to the simulation input scene. If true, reads in the scene file and scales length, velocity, time, mass, force, acceleration, etc. according to Froude similitude (e.g. λ^1, $\lambda^{0.5}$, $\lambda^{0.5}$, λ^3, λ^3, and 1). Must be a number greater than zero. [ ]

4.8.2 Input Parameters - Models

An array of `models` objects is used to create MPM and/or FEA objects. This includes specification of the device they each occupy (i.e. which GPU), the material properties, the algorithms to run (e.g. F-Bar antilocking, ASFLIP), the geometry to define them (e.g. a box, a loaded geometry from a SDF file), and any output settings specific to the model.

```
"models": [
    {
        "device": {...},
        "material": {...},
        "algorithm": {...},
```

```
        "geometry": [{...}, ...],
        "output": {...},
    },
    ...
],
```

The device object sets the Graphics Processing Unit (GPU) that holds the given model, as well as the model ID of the model on the GPU.

```
"device": {
    "gpu": 0,
    "model": 0
}
```

where variables are defined as:

- `gpu`: Graphics Processing Unit (GPU) ID to send this model to be simulated on. Must be a number that is between zero and the max number of supported GPUs on your system and compiled ClaymoreUW applications (set by g_device_cnt in `settings.h` file). [integer]

- `model`: Model ID on this GPU. Will eventually be done automatically for the user, but currently they must be set to be sequential integers starting at zero and increment according to appearance in the scene file. I.e., for GPU 0, first model 0 must appear in the scene, then model 1, model 2 (where each model is specific to the same GPU 0). Different model sequences for different GPU IDs may be interleaved in the scene file. [integer]

The `material` object is where material laws and parameters are set for the model. For instance, you may define water as:

```
"material": {
    "type": "particles",
```

```
    "ppc": 8,
    "constitutive": "JFluid",
    "CFL": 0.5,
    "rho": 1000,
    "bulk_modulus": 1e7,
    "gamma": 7.1,
    "viscosity": 0.001
}
```

where variables are defined as:

- `type`: Is the material represented as particles for MPM or as a mesh for FEA. Currently, must be a string that says either "particles" or "mesh". [string]

- `ppc`: Particle-per-cell for the model. If a power-of-two, will align to Newton-Cotes quadrature points. Must be a positive number greater than zero. [ ]

- `constitutive`: The material law to apply. Available options are "JFluid" (weakly-compressible isotropic fluid), "FixedCorotated" (hyper-elastic solid), "NeoHookean" (common hyper-elastic solid), "DruckerPrager" (Drucker-Prager, good for granular media), and "NACC" (Non-Associative Cam-Clay, good for clay, concrete, snow, and other basic topology changing solids). Must be a string stating one of the above key-words. [string]

- `CFL`: Courant-Friedrichs-Lewy condition number C for stability of the numerical method. May over-ride the default time-step `default_dt`, Δt, in the `simulation` object by forcing $\Delta t \leq C\Delta x / v_{\text{p-wave}}$ where p-wave velocity $v_{\text{p-wave}}$ is specific to a material, see Section 5.4. Recommended to use a value between 0.1 and 0.6. Must be a positive number. [integer]

- `rho`: Density of the material at simulation start. [kilograms / meter^3]

- `bulk_modulus`: Elastic bulk modulus, B, of the material at simulation start. Measure of material incompressibility. Used for the "JFluid" constitutive model. [Pascals]

- `gamma`: Derivative of the bulk modulus w.r.t. initial pressure. Measure of nonlinear response to volume change. Used in the Tait-Murnaghan equation of state inside of the "JFluid" constitutive model, see Section 5.4. [ ]

- `viscosity`: Dynamic viscosity of the material. Measure of stress response to time-dependent deformation. Used in the "JFluid" constitutive model. Must be a positive number. [Pascals * seconds]

- `youngs_modulus`: Elastic Young's modulus of the material at simulation start. Measure of inline stress-strain response. Used in "FixedCorotated", "NeoHookean", "DruckerPrager", and "NACC". Must be a positive number greater than zero. [Pascals].

- `poisson_ratio`: Poisson's ratio of the material at simulation start. Measure that relates axial deformation to 3D transverse deformation. Value of 0.5 is a truly incompressible material, 0 is fully compressible, and negative values are unusual expansive materials (e.g. cork). Must be a number less than 0.5. [ ]

- `friction_angle`: Friction angle to be used in the "DruckerPrager" material model. [degrees]

- `cohesion`: Cohesive value for use in the "DruckerPrager" material model. Currently based on the logarithm of strain, i.e. adjust the yield surface to account for said amount of cohesive strain. To be updated. [ ]

The algorithms applied to a model are set in the `algorithm` object per model:

```
"algorithm": {
    "use_ASFLIP": true,
```

```
    "ASFLIP_alpha": 0.3,
    "ASFLIP_beta_max": 0.01,
    "ASFLIP_beta_min": 0.0,
    "use_FBAR": true,
    "FBAR_ratio": 0.8,
    "FBAR_fused_kernel": true,
    "use_FEM": false
}
```

where variables are defined as:

- `use_ASFLIP`: True/false for using Affine-Separable Fluid-Implicit-Particles. Must be true to use ASFLIP, FLIP, or PIC-FLIP mixing. [Boolean]

- `ASFLIP_alpha`: The ASFLIP and PIC-FLIP α varies velocity advection mixing between PIC ($\alpha = 0$) and FLIP ($\alpha = 1$). Higher number gives a more energetic simulation, but with greater risk for noise and typically smaller required time-step. Must be a number within 0 and 1, $\alpha \in [0, 1]$. [ ]

- `ASFLIP_beta_max`: The ASFLIP positional correction ratio maximum value, β_{max}. Larger number allow greater positional correction of particles in spurious tensile conditions and may improve the contact behavior between bodies and material. However, larger values risk greater noise and instabilities. Best to keep this small, e.g. $\beta_{max} \in [0, 0.2]$, though any value between 0 and 1 is permissible, $\beta_{max} \in [0, 1]$. [ ]

- `ASFLIP_beta_min`: The ASFLIP positional correction ratio minimum value, β_{min}. Larger number allow greater positional correction of particles in ambient conditions and may improve the contact behavior between bodies and material. However, larger values risk

greater noise and instabilities. Best to keep this small, e.g. $\beta_{min} \in [0, 0.05]$, though any value between 0 and 1 is permissible, $\beta_{min} \in [0, 1]$. [ ]

- `use_FBAR`: True/false for using F-Bar volumetric antilocking (see Sections 5.7, 5.8, 5.9, and 5.10). Can greatly improve pressure field and reduce volumetric antilocking. [Boolean]

- `FBAR_fused_kernel`: True/false for using F-Bar volumetric antilocking in G2P2G fused kernel (true, faster) or G2P + P2G split kernels (false, slower). See section 5.10 for more details. [Boolean]

- `FBAR_ratio`: Mixing ratio of F-Bar volumetric antilocking, ψ. Values of $\psi = 0$ and $\psi = 1$ reduce to no antilocking and full antilocking, respectively. In stiff simulations, we typically use between 0.5 and 0.9999, although all values between 0 and 1 are permissible, $\psi \in [0, 1]$. [ ]

- `use_FEM`: True/false for using finite element method. Pending update to improve user-interface for defining vertices and elements. [Boolean]

The geometries composing a model are set in the `geometry` array of objects per model:

```
"geometry": [
    {
        "object": "Box",
        "operation": "Add",
        "span": [0.25, 0.5, 1.0],
        "offset": [0, 1.6, 0],
        "rotate": [0,0,0],
        "fulcrum": [0, 0, 0],
        "array": [8, 4, 2],
        "spacing": [0.5, 1.0, 2.0]
```

```
    },
    ...
]
```

where variables are defined as:

- `object`: Type of geometry object to use. E.g., basic geometries like "box", "sphere", "cylinder". May also set it to "file' to load-in user-provided files (must provided file-path in different variables). [string]

- `operation`: Geometry operation to apply. E.g. "add", "subtract", and eventually "difference", "intersection", and "union". These are Boolean arithmetic operations that allow a user to define any geometry with multiple geometry objects using varying operations. Operation is applied in serial, e.g. "subtract" in the third geometry object subtracts from objects one and two, but "add" in the fourth will be impervious to the prior subtraction. [string]

- `span`: Dimensions of the geometry in each direction. Array of three elements. Elements must be numbers greater than zero. [meters]

- `offset`: Offset of the geometry's origin compared to the simulation origin. Array of three elements. Elements must be numbers. [meters]

- `rotate`: Euler angles of rotation to apply to the geometry. Array of three elements. Elements must be numbers. [degrees]

- `fulcrum`: Point to center rotations on. Array of three elements. Elements must be numbers. [meters]

- `array`: How many geometry objects to create in each direction (X, Y, Z). Array of three elements. Elements must be integers greater than or equal to 1. [integers]

- `spacing`: Spacing of geometry objects in each dimension (Sx, Sy, Sz). Array of three elements. Elements must be numbers, and numbers must be greater than or equal to the corresponding `span` element to avoid self-intersection between array bodies. [meters]

- `file`: If `object` selected is set to "file", you must provide the file-path in this variable. Relative to the ClaymoreUW AssetDir environment variable, default is "ClaymoreUW/Data/". [file-path]

- `padding`: If `object` is set to "file" and `file` was provided and SDF file-path, this declares the padding used in the SDF file. SDF file require a minimum padding of 1 to define an objects surface, so this should be an integer greater than or equal to 1 if using and SDF file geometry. [integer]

- `scaling`: If `object` is set to "file" and `file` was provided an SDF, CSV, BGEO, etc. file-path then this will scale the linear dimensions of that file's object by the value provided. I.e., 2 will double linear dimensions. Must be a positive number greater than zero. [ ]

Output settings for a model are given in the `output` object per model:

```
"output": {
    "output_attribs": ["Pressure", "ID"],
    "track_attribs": ["Position_X", "Position_Y", "Position_Z", "Velocity_Magnitude",
    ↪  "Velocity_X",  "Velocity_Y", "Velocity_Z"],
    "target_attribs": ["Position_Y"]
}
```

where variables are defined as:

- `output_attribs`: List of attributes to output per particle in each full output file. ID, Mass, Volume, Position_X, Position_Y, Position_Z, Velocity_X, Velocity_Y, Velocity_Z,

Velocity_Magnitude, DefGrad_XX, DefGrad_XY, DefGrad_XZ, DefGrad_YX, DefGrad_YY, DefGrad_YZ, DefGrad_ZX, DefGrad_ZY, DefGrad_ZZ, J, DefGrad_Determinant = J, JBar, DefGrad_Determinant_FBAR = JBar, StressCauchy_XX, StressCauchy_XY, StressCauchy_XZ, StressCauchy_YX, StressCauchy_YY, StressCauchy_YZ, StressCauchy_ZX, StressCauchy_ZY, StressCauchy_ZZ, Pressure, VonMisesStress, DefGrad_Invariant1, DefGrad_Invariant2, DefGrad_Invariant3, DefGrad_1, DefGrad_2, DefGrad_3, StressCauchy_Invariant1, StressCauchy_Invariant2, StressCauchy_Invariant3, StressCauchy_1, StressCauchy_2, StressCauchy_3, StressPK1_XX, StressPK1_XY, StressPK1_XZ, StressPK1_YX, StressPK1_YY, StressPK1_YZ, StressPK1_ZX, StressPK1_ZY, StressPK1_ZZ, StressPK1_Invariant1, StressPK1_Invariant2, StressPK1_Invariant3, StressPK1_1, StressPK1_2, StressPK1_3, StressPK2_XX, StressPK2_XY, StressPK2_XZ, StressPK2_YX, StressPK2_YY, StressPK2_YZ, StressPK2_ZX, StressPK2_ZY, StressPK2_ZZ, StressPK2_Invariant1, StressPK2_Invariant2, StressPK2_Invariant3, StressPK2_1, StressPK2_2, StressPK2_3, StrainSmall_XX, StrainSmall_XY, StrainSmall_XZ, StrainSmall_YX, StrainSmall_YY, StrainSmall_YZ, StrainSmall_ZX, StrainSmall_ZY, StrainSmall_ZZ, StrainSmall_Invariant1, StrainSmall_Invariant2, StrainSmall_Invariant3, Dilation = StrainSmall_Invariant1, StrainSmall_Determinant = StrainSmall_Invariant3, StrainSmall_1, StrainSmall_2, StrainSmall_3, VonMisesStrain, PorePressure, and logJp currently supported. Array of arbitrary number of elements. [strings]

- `track_attribs`: Attribute(s) on tracked particles to output. See `output_attribs` for supported particle attributes. Array of arbitrary number of elements. [string]

- `target_attribs`: Attribute(s) on particle-sensor to output. See `output_attribs` for supported particle attributes. Array of arbitrary number of elements. [strings]

There are a few remaining so-called global variables (within the scope of a model object) that require specification. Setting these outside of a geometry object will apply them to all geometry objects of the model. For instance:

```
"models": [
    {
        "velocity": [1, 0, 0],
        "track_particle_id": [0, 1000, 2000, 3000],
        "partition_start": [0,0,0],
        "partition_end": [4,4,4]
    },
    ...
]
```

where variables are defined as:

- `velocity`: Initial velocity of all particles in the model. Array of three elements. Elements must be numbers. [meter / second]

- `track_particle_id`: List of global model IDs to track, i.e. particles with these IDs are specifically tracked for output. Attribute(s) to track on tracked particles should be specified in `track_attribs`. Array of arbitrary number of elements. Elements must be integers greater than or equal to zero. [integers]

- `partition_start`: Any particle that is closer to the origin than this point at simulation start is removed. Used to help split things across GPU partitions. Array of three elements. Elements must be numbers. [meters]

- `partition_end`: Any particle that is further to the origin than this point at simulation start is removed. Used to help split things across GPU partitions. Array of three elements.

Elements must be numbers. [meters]

This concludes input parameters for the `models` object array.

4.8.3 Input Parameters - Grid Boundaries

Grid boundaries are set as an array of objects. Each will define a unique grid-boundary with associated contact type, motion, etc., where appropriate.

```
"grid-boundaries": [
    {
        "object": "Wall",
        "contact": "Slip",
        "friction_static": 0.0,
        "friction_dynamic": 0.0,
        "domain_start":[0,0,0],
        "domain_end": [8,8,8]
    },
    ...
]
```

where variables are defined as:

- `object`: Type of boundary object to use. E.g., Walls, Floor, Box, OSU_LWF_RAMP, OSU_LWF_PADDLE, USGS_RAMP, USGS_GATE, OSU_TWB_RAMP, OSU_TWB_PADDLE, WASIRF_PUMP, TOKYO_HARBOR, velocity_boundary. Note that some prefixes refer to boundaries specific to the digital twins of the Oregon State University Large Wave Flume (OSU_LWF, Chapter 7), Oregon State University Directional Wave Basin (OSU_TWB, Section 9.2), U.S. Geological Survey Debris-Flow Flume (USGS, Section 9.4), University of Washington Flume Wind-and-Sea Interaction Facility (WASIRF, Section 9.3), and Waseda University's Tsunami Wave Basin (TOKYO, Chapter 8). [string]

- `contact`: Contact type for application of boundary condition. Includes "sticky", "slip", and "separable". See Section 5.3 for more details. [string]

- `friction_static`: Static friction coefficient to apply on boundary. Not all boundaries support this currently, except Floor, TOKYO_Harbor, USGS_RAMP. Coulomb friction model used. Must be a number greater than or equal to zero. [ ]

- `friction_dynamic`: Static dynamic coefficient to apply on boundary. Not all boundaries support this currently, except Floor, TOKYO_Harbor, USGS_RAMP. Coulomb friction model used. Must be a number greater than or equal to zero. [ ]

- `time`: Time that the boundary is active. Array of two elements, first element defines time that it appears, second element is the time that is disappears. Elements must be numbers. [seconds]

- `domain_start`: Where to begin the boundary domain, i.e. the point nearest to the origin. Array of three elements. Elements must be numbers. [meters]

- `domain_end`: Where to end the boundary domain, i.e. the point furthest from the origin. Array of three elements. Elements must be numbers greater than corresponding element in `domain_start`. [meters]

- `motion_file`: File-path to the motion file for a moving boundary. Currently, only takes in CSV files with columns (no headers) specifying time, velocity in X, and position in X (relative to `domain_start` and `domain_end`). Used to define wave-maker piston motion. [file-path]

- `motion_freq`: Frequency of sampling in the motion file. Subsequent entries assumed to be at increments of its inverse. [Hz]

- `velocity`: Constant velocity boundary. Array of three numbers. [meters / second]

The original Claymore also featured a means to load-in generalized boundaries defined by an SDF file. This was, however, very memory inefficient (had to define the SDF file over and SDF discretization equal to the exact simulation grid domain) so it was deprecated. It will eventually be re-implemented in ClaymoreUW with a better user-interface and memory system (non-conformal discretization of SDF relative to grid). Will also include generalized application of friction on motion (translation and rotational) for ClaymoreUW.

4.8.4 Input Parameters - Grid Sensors

Sensors, i.e. instrumentation, to add to the simulation are split across the grid and the particles currently. Grid-sensor instruments, e.g. a load-cell that measures forces on a rigid MPM boundary using grid node forces, may be defined with:

```
"grid-sensors": [
    {
        "attribute": "Force",
        "operation": "Sum",
        "direction": "Z+",
        "output_frequency": 120,
        "domain_start": [-0.1, -0.1, 8],
        "domain_end": [8.05, 0.5, 8.1]
    },
    ...
]
```

where variables are defined as:

- `attribute`: Attribute on a grid-node that our sensor instrument will measure. "Force",

"Acceleration", "Momentum", "Velocity", "Mass", "Volume", and "JBar" currently supported. [string]

- `operation`: Reduction operation to perform on the `attribute`. E.g. "Sum", "Max", "Min", "Count", "Average", and "STDEV". Just about any sensor should be possible using these fundamental reductions. [string]

- `direction`: The direction to measure with respect to if `attribute` concerns a vector quantity. E.g., "X" will measure vectors projected onto the X axis, "X-" will only take vectors that point opposite of the unit X vector after project onto the X axis, "X+" will only take vectors that point in the unit X vector after project onto the X axis. "Y" and "Z" (and their variants) also applicable. Will eventually use a vector instead, for now must provide a string. [string]

- `output_frequency`: Frequency to perform and output the reduction operation of the sensor instrument. I.e. if a load-cell that measures force has a 120 Hz operating frequency, use 120. [Hz]

- `domain_start`: Where to begin the sensor domain, i.e. the point nearest to the origin. Array of three elements. Elements must be numbers. [meters]

- `domain_end`: Where to end the sensor domain, i.e. the point furthest to the origin. Array of three elements. Elements must be numbers. [meters]

4.8.5 Input Parameters - Particle Sensors

Likewise, the particle-sensors, e.g. a free-surface elevation gauge (a.k.a. wave-gauge), are defined similarly to grid-sensors:

```
"particle-sensors": [
    {
        "attribute": "Elevation",
        "operation": "Max",
        "output_frequency": 120,
        "domain_start": [4.0, 1.6, 0.0],
        "domain_end":   [4.1, 8.0, 0.1]
    },
    ...
]
```

where variables are defined as:

- `attribute`: Attribute on a particle that our sensor instrument will measure. ID, Mass, Volume, Position_X, Position_Y, Position_Z, Velocity_X, Velocity_Y, Velocity_Z, Velocity_Magnitude, DefGrad_XX, DefGrad_XY, DefGrad_XZ, DefGrad_YX, DefGrad_YY, DefGrad_YZ, DefGrad_ZX, DefGrad_ZY, DefGrad_ZZ, J, DefGrad_Determinant = J, JBar, DefGrad_Determinant_FBAR = JBar, StressCauchy_XX, StressCauchy_XY, StressCauchy_XZ, StressCauchy_YX, StressCauchy_YY, StressCauchy_YZ, StressCauchy_ZX, StressCauchy_ZY, StressCauchy_ZZ, Pressure, VonMisesStress, DefGrad_Invariant1, DefGrad_Invariant2, DefGrad_Invariant3, DefGrad_1, DefGrad_2, DefGrad_3, StressCauchy_Invariant1, StressCauchy_Invariant2, StressCauchy_Invariant3, StressCauchy_1, StressCauchy_2, StressCauchy_3, StressPK1_XX, StressPK1_XY, StressPK1_XZ, StressPK1_YX, StressPK1_YY, StressPK1_YZ, StressPK1_ZX, StressPK1_ZY, StressPK1_ZZ, StressPK1_Invariant1, StressPK1_Invariant2, StressPK1_Invariant3, StressPK1_1, StressPK1_2, StressPK1_3, StressPK2_XX, StressPK2_XY, StressPK2_XZ, StressPK2_YX, StressPK2_YY, StressPK2_YZ, StressPK2_ZX, StressPK2_ZY, StressPK2_ZZ, StressPK2_Invariant1,

StressPK2_Invariant2, StressPK2_Invariant3, StressPK2_1, StressPK2_2, StressPK2_3, StrainSmall_XX, StrainSmall_XY, StrainSmall_XZ, StrainSmall_YX, StrainSmall_YY, StrainSmall_YZ, StrainSmall_ZX, StrainSmall_ZY, StrainSmall_ZZ, StrainSmall_Invariant1, StrainSmall_Invariant2, StrainSmall_Invariant3, Dilation = StrainSmall_Invariant1, StrainSmall_Determinant = StrainSmall_Invariant3, StrainSmall_1, StrainSmall_2, StrainSmall_3, VonMisesStrain, PorePressure, and logJp currently supported. [string]

- `operation`: Reduction operation to perform on the `attribute`. E.g. "Sum", "Max", "Min", "Count", "Average", and "STDEV". Just about any sensor should be possible using these fundamental reductions. [string]

- `output_frequency`: Frequency to perform and output the reduction operation of the sensor instrument. I.e. if a wave-gauge that measures elevation has a 120 Hz operating frequency, use 120. [Hz]

- `domain_start`: Where to begin the sensor domain, i.e. the point nearest to the origin. Only measure particles past this point. Array of three elements. Elements must be numbers. [meters]

- `domain_end`: Where to end the sensor domain, i.e. the point furthest to the origin. Only measures particles before this point. Array of three elements. Elements must be numbers. [meters]

4.9 *ClaymoreUW - Developer Guide*

With an understanding of the MPM code implementation in Multi-GPU ClaymoreUW, this section summarizes the actual file-structure of a ClaymoreUW application for the convenience of

developers.

The Claymore open-source project is well written but not at all intuitive to those accustomed to engineering software. Below is a brief synopsis of the file structure and constituent functions so one may better navigate and add to the code-bases of either Claymore or ClaymoreUW.

- `scene.json` - Simulation run-time variables are set here. This JSON script includes file locations for models, specification of material properties, setting of force recorders, etc.

- `settings.h` - Simulation global compile-time variables are set here. This file includes max particle counts, max particles per cell, domain length, domain resolution, default material properties.

- `project.cu` - Here is where the simulation begins. This file handles the run-time input from an input script (`scene.json`) for the simulation. Initializes particle models, among other things, for passing into the actual simulation (found in `mgsp_benchmark.cuh`).

- `mgsp_benchmark.cuh` - This includes the simulation workflow. Multi-GPU consensus is organized here. This file serves as a wrapper of sort for launching GPU kernels. Output of entities is organized here.

- `mgmpm_kernels.cuh` - Here is where the majority of the GPU kernel functions reside. This file includes the full Material Point Method Algorithm (Grid update, G2P2G, etc.), as well as data-structure management and input/output schemes for device arrays. Many functions are material specific. Changes to the MPM algorithm are implemented here and called from the wrapper (`mgsp_benchmark.cuh`).

- `constitutive_models.cuh` - This file contains GPU device functions for the material laws (e.g. Fixed-Corotated, Weakly-Compressible Fluid a.k.a. JFluid).

- `particle_buffer.cuh` - This file organizes the data-structures for the particle models to be used in GPU kernels. Sets material specific properties.

- `grid_buffer.cuh` - This file organizes the data-structures for the grids to be used in GPU kernels. Sets scheme specific grid-node attributes (e.g. ASFLIP requires more grid-node velocity information than standard MPM). Dual-grids, staggered grids, etc. can be implemented here.

- `halo_buffer.cuh` - This file organizes the data structures for the halo regions to be used in GPU kernels.

- `hash_table.cuh` - Rarely need to edit this file unless trying to improve memory usage/change data-structure behavior. This file provides a hash-table mapping for fast access of sparse, complex GPU data-structures.

- `halo_kernels.cuh` - This file contains GPU device functions for operating the halo-regions between GPUs.

- `partition_domain.h` - This file partitions the domain. Not used in current state of code.

- `utility_funcs.hpp` - This file includes utility functions for convenience, such as finding the dyadic product for quadratic B-spline kernels. Not always utilized.

4.10 *ClaymoreUW - Algorithm*

Claymore uses a specific Moving-Least-Squares Material Point Method (MLS-MPM, Section 2.5) formulation with explicit time-integration and Quadratic B-Spline shape-functions. In this section, the grid-to-particle-to-grid fused kernel (G2P2G) (Wang et al. 2020b) used in Claymore

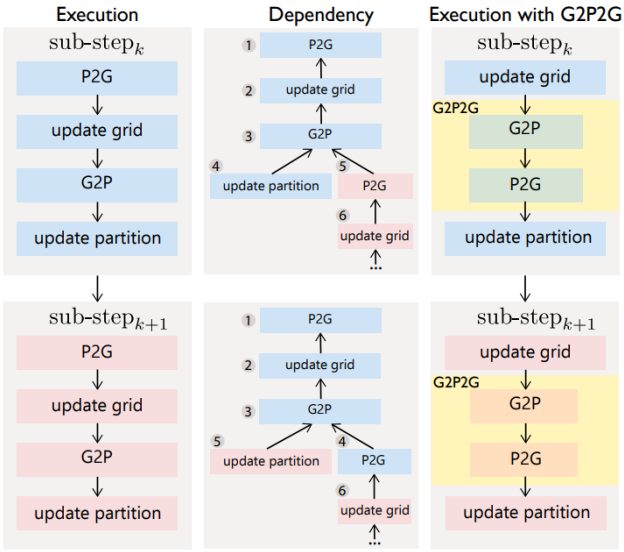

Fig. 4.2. G2P2G MPM Algorithm - Claymore Software. Grid-to-particle-to-grid fused kernel for Material Point Method in Claymore software. Courtesy of Wang et al. 2020b.

is briefly described relative to the MPM algorithm execution. The following steps are used and show in Figure 4.2:

Update Grid. The grid state is updated. Velocity evolves with respect to boundary conditions. $\mathbf{v}_i^n \rightarrow \mathbf{v}_i^{n+1}$

G2P2G. Fused step involving G2P, Particle Advection, and P2G: *G2P* is when all relevant grid velocity is transferred onto particles to reconstruct particle velocities and particle affine velocity matrices. $\mathbf{v}_i^n \rightarrow \mathbf{v}_p^{n+1}, \mathbf{C}_p^{n+1}$. *Particles Advection* occurs. $\mathbf{v}_p^{n+1} \rightarrow \mathbf{x}_p^{n+1}$. *Particles Evolve* deformation gradients and stress. $\mathbf{F}_p^n \rightarrow \mathbf{F}_p^{n+1}, \boldsymbol{\sigma}_p^{n+1}$. *P2G* transfers particle mass, momentum, affine momentum, and internal force to relevant grid-nodes. $m_p, \mathbf{v}_p^{n+1}, \mathbf{C}_p^{n+1}, \boldsymbol{\sigma}_p^{n+1} \rightarrow m_i, m_i\mathbf{v}_i^{n+1}$.

Update Partition. Data-structures are updated for the next time-step. Unoccupied grid-blocks are deactivated. This is essentially software overhead.

Claymore's parallel code implementation uses the MLS-MPM algorithm pipeline and G2P2G step described above. A few notes on the GPU kernel implementation is presented next.

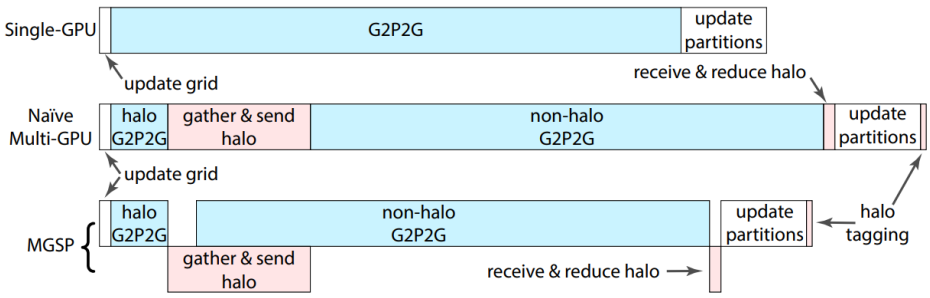

Fig. 4.3. Instructions for Single and Multi GPU Material Point Method - Claymore Software. Instruction pipe-line for both Single and Multi-GPU MPM in Claymore. Intelligent Multi-GPU MPM design uses CUDA asynchronous properties to hide latency, approaching Single-GPU speeds. Courtesy of Wang et al. 2020b.

4.11 *ClaymoreUW - GPU Kernels*

The high-level MPM workflow in Claymore and ClaymoreUW (Sec. 4.10) is organized as a series of kernels implemented in CUDA C++. Said code may be accessed in the respective open-source github repositories. This section presents the steps of the parallel MLS-MPM algorithm of ClaymoreUW used in each GPU kernel and how they are optimized to run on Multi-GPU systems. Only kernels relating to the numerical algorithm are described, we do not include all book-keeping functions used for managing back-end data-structures.

Thread/Block/Grid design, shared memory usage, Multi-GPU MPM consensus, and other aspects necessary for understanding the software are covered. For an introduction to GPUs and CUDA review see Chapter 3. The instruction pipeline of Claymore is shown in Figure 4.3 for single, naive Multi-GPU (non asynchronous Halo/Non-Halo), and Multi-GPU Static Partitioning (MGSP, i.e. partition scene across GPUs by undeformed particle location).

Update Grid. This function updates the grid state (i.e. velocities) and solves the equation of motion on the shared MPM grid. Kernel `update_grid_velocity_query_max()` is called here. CUDA blocks contain one or more grid-blocks each. Threads pull in data from a grid node

(corresponding to thread index) in the grid block (corresponding to the block index) from the grid buffer. Basic boundary conditions are determined and applied to the grid node velocities before writing results to the next grid buffer. Each CUDA block receives a shared memory array to hold max velocity values. These atomically reduce to a single max velocity per warp. Warp max values are reduced to a single max value, used for the CFL condition.

G2P2G. This function is the grid-to-particle-to-grid fused algorithm, as shown in Figure 4.2, that runs the bulk of MLS-MPM computations. Kernel `g2p2g()` is called on a material basis (i.e. unique `g2p2g()` exists and is optimized for each material). Shared memory per CUDA block is reserved for two arrays, representing the prior and posterior grid-arena state relative to a particle-block as shown in Figure 4.6. The first is the `g2pbuffer` which uses 4 x 3 x 512 bytes, the second is the `g2pbuffer` which uses 4 x 4 x 512 bytes. Respectively, this is for three and four floats per grid node in a grid arena (i.e. velocity vector in G2P, mass and momentum vector in P2G). `g2pbuffer` is populated efficiently by threads pulling values from grid nodes in the correct grid blocks in the grid buffer. This is done in a manner designed to avoid bank conflicts. `p2gbuffer` is zeroed initially. A shared pool of memory representing all needed grid information for G2P is now available in `g2pbuffer`. Each thread will now represent a particle in a particle bin of the particle block being executed by the CUDA block of `g2p2g()`. The thread pulls in particle data (e.g. position, deformation gradient) from global memory into the register. MLS-MPM is performed now, using particle data and the shared `g2pbuffer`. Particles advect velocity and position. Particles evolve deformation and stress. Updated particle values are written back into the global memory particle buffer. P2G now starts. Particle mass, momentum, and internal force contributions atomically add into the shared `p2gbuffer` of the CUDA block. When all threads complete of the block complete, the `p2gbuffer` is written (atomically added) into the global grid buffer in a coalesced manner for efficiency.

Update Partition. In this function overhead tasks and consensus are performed. This includes

Multi-GPU communication (lock, send, receive, unlock), particle bin reorganizing, grid block activation/deactivation, and halo block tagging.

Input/Output. These function ares used to input and output data into the MPM simulation. This is where input of new data on the host (CPU, RAM, HDD, SSD) to be written into the device (GPU) is done, such as updated boundary conditions. For this purpose, data is typically arranged as basic host arrays which asynchronously copy into a basic device array. Asynchronous operation prevents this process from slowing down the Multi-GPU program. Then, if needed, a kernel is launched to map basic device arrays into more complex data-structures described in Section 4.12. Output entails the same process in reverse. Our lab has added extensive functionality to output options so engineers may better understand the simulations they are performing without manually having to find a way to pull out data from complex Multi-GPU data-structures (e.g. summed force on a specified surface at a frequency).

4.12 *ClaymoreUW - Data Structures*

Data-structures are the largest barrier preventing engineers from enjoying the benefits of Multi-GPU numerical simulations. The arrangement of numerical bodies (particles, grid-nodes, elements), the means to access them, and the properties they possess are all managed by data-structures. Maintenance and use of data-structures on Multi-GPUs for hundreds of millions of numerical bodies is costly in time and memory. For programmers without traditional training in data-structures they are also incredibly unwieldy to use. Often times they are the singular obstacle preventing the implementation of numerical features (e.g. implicit time-integration, Poisson equation solvers) as these features may require data-usage that is difficult to organize on a Multi-GPU system.

To accommodate a typical user's development time, complex data-structures should be interacted with at only the highest, most abstracted level. However, at the lowest level these data-structures

should be optimized for memory and computation usage on multiple GPUs. This is a tall order.

Modern advancements in numerical simulations, especially in the field of computer graphics, have aimed to abstract away a users interaction with complex GPU data-structures. Taichi is an excellent high-performance programming language for numerical simulations and visualizations (Hu et al. 2019b) that, among other things, allows users to design complex data-structures (e.g. hierarchical, sparse, dynamic, etc.) with a front-end that requires minimal to no CUDA C++ and/or GPU expertise. In addition, Taichi makes the access to these structures convenient and their use well optimized. Claymore (Wang et al. 2020b) follows a similar design philosophy, as described in their technical supplement (Wang et al. 2020a) which is described in Section 4.12 as it is used in ClaymoreUW. However, neither code is specifically intended for engineering applications, and thus they present a learning curve that we hope to alleviate in some small part here.

A few terms will be used throughout this section to describe the data moving through a Multi-GPU MPM Claymore simulation. Some of the specifications are specific to supporting quadratic B-spline shape-functions and may grow or shrink with cubic B-splines or tri-linear shape-functions respectively. At a high level, data is organized as follows.

Particles, also referred to as "material points" in MPM, are the smallest unit of the material domain. They serve as quadrature points numerically and are free to move through the domain. Particles hold on to whatever independent information they need to update. Figure 4.4 shows how a simple fluid or solid particle may store information differently. An isotropic fluid may hold position (3x1 vector, 3 floats) and volume change ratio (scalar, 1 float). A linear elastic solid may hold position (3x1 vector, 3 floats) and deformation gradient (3x3 tensor, 9 floats). Shared, static particle properties are not saved on each particle (e.g. Poisson's Ratio for linear elastic material). Therefore, memory usage is material/scheme dependent. For certain applications, particles may need to hold velocity (3x1 vector, 3 floats) or unique material parameters (e.g. for plasticity). For example, Fixed-Corotated solids using a PIC-FLIP advection scheme requires position (3 floats),

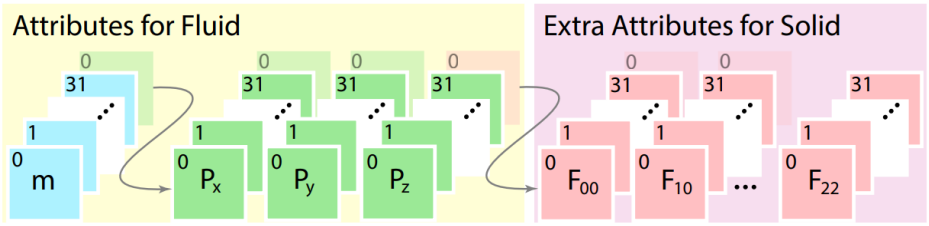

Fig. 4.4. Particle Bin Attributes by Material - Claymore Software. Attributes on particles in Multi-GPU MPM software Claymore visualized. Different material models require differing information, e.g. 4 floats per fluid particles and 13 floats per fixed-corotated solid. Arranged in bins of 32 to match NVIDIA warp size. Courtesy of Wang et al. 2020b.

velocity (3 floats), and deformation gradient (9 floats). In Claymore, particles are executed by an individual CUDA thread.

Grid Nodes are the smallest unit of the background, "scratch-pad" domain. They are used to solve the equation of motion (EOM) and to apply boundary conditions. Grid nodes hold whatever information they need to solve the EOM, as well as other values used for boundary conditions or augmented G2P transfers. Typically they hold mass (scalar, 1 float) and momentum/velocity (3x1 vector, 3 floats). Grid nodes are arranged in a uniform Cartesian pattern which resets to its initial position each time-step so they do not need to hold position information. Some schemes (e.g. FLIP/ASFLIP) require grid-nodes to hold an extra velocity vector, while advanced boundary conditions may require a surface normal (3x1 vector).

Particle Bins are groups of 32 nearby particles. 32 specifically optimizes for GPUs (See Chapter 3). Bins are executed in parallel by a CUDA warp. Particle bins reconstruct every time-step to maintain the spatial condition of 32 particles within a 4 x 4 x 4 grid-node domain (for Quad. B-Splines shape-functions). They are selected in a way that reduces the chance that any two particles in a bin occupy the same grid-cell, minimizing bank conflicts when particles in the warp read/write from/to shared grid-nodes for G2P/P2G transfers. Figure 4.4 shows how a particle bin organizes particle data as structures-of-arrays (SoA) in GPU global memory.

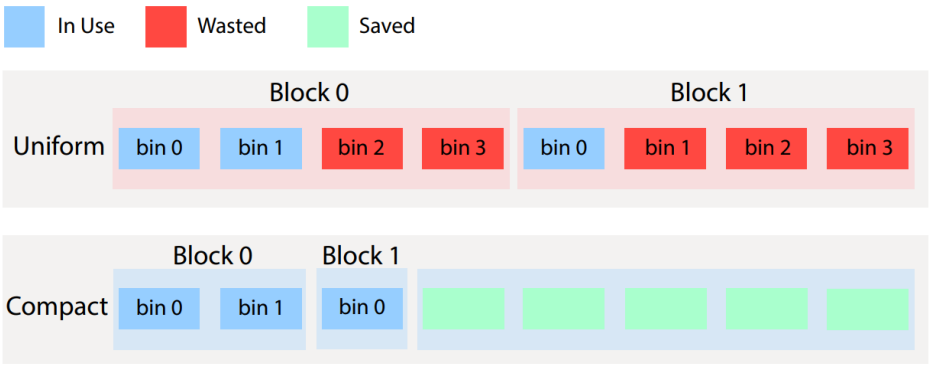

Fig. 4.5. Particles Binning in Particle Block - Claymore Software. Particle Blocks (4 x 4 x 4 cell size) dynamically resize to fit Particle Bins (32 particles) in its domain. Saves memory but has overhead. Courtesy of Wang et al. 2020b.

Particle Blocks are dynamic groups of Particle Bins that are located in the same 4 x 4 x 4 grid-node group (for Quad. B-Spline shape-functions). Their dynamic quality allows for better memory usage in simulations where particles are not evenly distributed. Figure 4.5 visually shows that the applied dynamic resizing buffer of compact particle bins scheme saves memory compared to a non-dynamic approach where no resizing occurs.

Grid Blocks are 4 x 4 x 4 adjacent grid-nodes (for Quad. B-Spline shape-functions). These are the smallest unit of grid-nodes that are actually used in most kernels, as G2P and P2G operations require many grid-nodes to support particle blocks. Grid blocks are considered active if any interior grid-node possesses mass and inactive if no grid-node has mass (i.e. memory freed for the next time-step).

Grid Arenas are groups of 2 x 2 x 2 adjacent grid blocks (8 x 8 x 8 grid-nodes = 512 grid-nodes). Grid arena size is set to maintain the off-by-two requirement for a particle block execution in the G2P2G kernel. Figure 4.6 show a schematic of a grid arena for a 2D case. Each grid-node holds a minimum of 3 floats for G2P operation, and 4 floats for P2G operation. They are the smallest set of grid-nodes a full particle block will read/write from/to during the G2P2G operation (Sec.

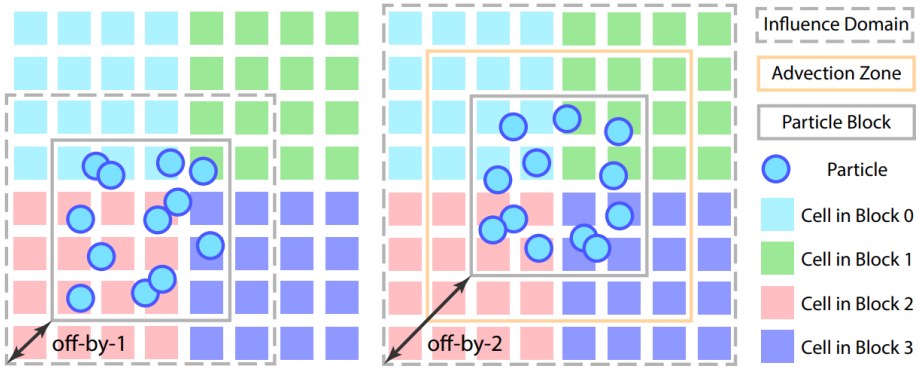

Fig. 4.6. Off-By-2 - Claymore Software. G2P2G kernel requires off-by-2 scheme, i.e. a given particle block entering G2P2G needs read/write access to a Grid Arena (2 x 2 x 2 Grid-Blocks) which is reserved in shared memory. It assumes CFL condition and Quadratic B-Spline shape-functions. Particle Block must be centered which creates a 2 grid-node buffer relative to the Grid Arena, hence "Off-By-2". Courtesy of Wang et al. 2020b.

4.11). Grid arenas are an array of grid-block information pulled into shared memory for optimized particle reads and writes within a CUDA block's scope. At the end of G2P2G, they hold inter-block atomically reduced particles contributions from G2P2G, which then intra-block reduce to achieve consensus on the full, shared grid. Memory for grid arenas is manually reserved as shared memory for the kernel.

Halo Blocks are grid and/or particle blocks in overlap regions between GPU devices. A halo-tagging process is undertaken to determine which blocks are relevant and which devices they must be sent to. These are the smallest unit for Multi-GPU communication. Transferring/receiving grid-nodes and/or particles in blocks, which are already optimized for CUDA reads/writes, is far more efficient than sending information from individual particles and grid-nodes. Multi-GPU MPM tasks are divided between operations that influence only halo blocks and operations influencing only device specific blocks (non-halo blocks). Halo block regions are seen as a yellow rectangle in Figure 4.7 for a dual-GPU simulation. Results in these regions are shared between the GPUs.

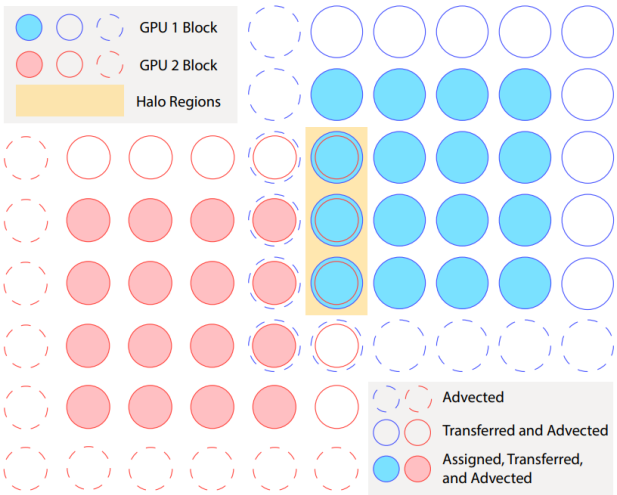

Fig. 4.7. Organizing MPM Grid Blocks for Multi-GPU - Claymore Software. Halo-Blocks are tagged for Multi-GPU transfer. Courtesy of Wang et al. 2020b.

Non-Halo Blocks are blocks that do not overlap in regions between partitions of the GPU device. They do not require special treatment. MPM operations on non-halo blocks do not need to communicate across Multi-GPUs so they are usually performed after halo block execution, but asynchronous of Multi-GPU halo block communication. Non-halo blocks are all blocks not in the yellow rectangle of Figure 4.7 and are unique to their GPU.

Particle Buffers are the full collection of particle blocks that constitute an object (e.g. a water body, a debris model) on a device. They collect particle bins (SoA) into a global memory array, making an array-of-structures-of-arrays (AoSoA) data-structure. They are material specific, holding basic material information (e.g. stiffness, density) and functions (e.g. for updating properties) common in all composing particles. Multiple objects can be on a device so there can be multiple particle buffers per device. Devices can transfer particle blocks to and from particle buffers of other devices depending on the partitioning scheme (Section 4.15) using marked halo blocks in the buffer.

Grid Buffers are the full collection of active grid-blocks on a device. Devices can transfer grid

blocks to and from grid buffers of other devices depending on the partitioning scheme (Section 4.15) using marked halo blocks in the buffer.

Partition is a data-structure for management of the grid and the relationship between particles and the grid. It also facilitates Multi-GPU organization. The partition reserves the largest amount of memory in most Claymore simulations. It keeps track of active grid-blocks to maintain grid sparsity, tags halo-blocks, provides a mapping from a scalar grid-block index to a 3D index (considering sparsity). It records which particles are in a grid-cell, number of particles in grid-cells, which particles are in a grid-block, and number of particles in a grid-block. The partition domain is set to encompass the entire simulation domain so it does not have to resize (as the grid buffer does), but this is at the cost of memory usage for sparse simulations. It pre-allocates enough memory to write 2 + `MAX_PPC` * 2 integers for each grid-cell in the full domain.

Vertices are a novel addition by our lab in Claymore to represent vertices in our coupled implementation of Finite Elements (see Section 5.13). Vertices are held in basic device arrays. The structure does not employ sparsity, hierarchy, etc., and does not reorder. Vertices hold their position (3 floats), internal force (3 floats), mass (1 float), volume (1 float), and optionally velocity (3 floats) or surface normal (3 floats). Each vertex is positioned in an array according to its unique integer ID. MPM particles tagged as "meshed" write advected positions to their corresponding vertex by knowing their ID, transcribing directly into the Vertice array index. This works despite MPM particles reorganizing throughout particle-bins and particle-blocks in Claymore. It is also compatible with Multi-GPU simulations. Meshed MPM particles read information from vertices (e.g. internal force) after FEM elements updates.

Elements are a novel addition by our lab to represent element-wise organization of our coupled implementation of Finite Elements (Sec. 5.13). Elements are basic device arrays, holding integer IDs of all its vertices. They also hold precomputed edge vectors and volume of the undeformed state (i.e. done at the start of a simulation). For a 4-point linear tetrahedron element, this

corresponds to 4 integers for vertex IDs, 9 floats for precomputed edge matrix, 1 float for precomputed volume. Elements read the information of its vertices from the Vertice array and use it along with its own information to compute the updated deformation and stress at element Gauss points. Stress is then mapped as internal force back to vertices, done as a reduction using atomic addition to vertices in a Vertice array. This prevents data-races, but further optimization could of course be found through warp-reduction within blocks before a global reduction with atomic addition.

Element Buffer is a novel addition by our lab, serving as a superstructure that organizes FEM elements. It is similar to a Particle Buffer as it holds material information that is shared across elements (e.g. stiffness). It allows for declaration of different material and element types with different information requirements (e.g. 4-point linear-elastic tetrahedron vs 8-point non-associative Cam-Clay brick).

This covers the broad data-structures used in our modified ClaymoreUW software. If a user becomes comfortable with them, and minds best practices of CUDA design in Chapter 3 and references NVIDIA 2022 and NVIDIA 2023, then implementing on Multi-GPUs is not much different to a CPU code. However, the results are orders of magnitude faster because of low-level optimization scaled across massively parallel systems.

In the next section, we provide an adapted technical supplement by Wang et al. 2020a for developers interested in adding to or repurposing the Claymore or ClaymoreUW code.

4.13 *ClaymoreUW - Technical Supplement*

For a more low-level view of the underlying data-structure system in ClaymoreUW, Claymore, Taichi, and to an extent, a variety of other hierarchically assembled GPU data-structure libraries written in CUDA C/C++, we provide a technical documentation written by the authors of the original Claymore code (Wang et al. 2020b), which was published in ACM conference

proceedings by Wang et al. 2020b, all right reserved.

In this section, we provide additional exposition so that the technical supplement by Wang et al. 2020b may be more easily understood. This is because for those not accustomed to CUDA and C++ data-structure design and optimization the technical implications of this supplement will not be apparent. We also include minor alterations to expository text that does not fully apply to ClaymoreUW, and direct readers back to Wang et al. 2020a for the original format.

However, we emphasize this is not our original work, beyond minor commentary, and we have not significantly altered this portion of the Claymore code in ClaymoreUW. While we use this system to alter the actual data structures that are used and to introduce new ones, the underlying system for construction is their original work and should be cited as such if publishing work based on it or if redistributing said work. Our reason for providing it below is purely for the convenience of readers who would like to better understand how to develop in the open-source code-bases of Claymore and ClaymoreUW.

4.13.1 Compile-Time Settings

To maximize the performance, Claymore uses compile-time constants for both simulation controls and material related settings. ClaymoreUW seeks to eliminate the majority of interaction with these settings for users without a performance hit through template specialization (though this increases compile times). Below, we provide additional details on compile-time settings for user convenience and reproduction purposes. For example in Wang et al. 2020b, the paper in which Claymore was introduced, they set the maximum number of particle-per-cell to be 64 for all example simulations. This setting is more than sufficient for most graphical MPM use since the typical particle-per-cell (PPC) is 8 when initializing scenes, and material are reasonably compressible but unlikely to exceed 8x the initial PPC in all but the most chaotic scenes. However, the particles will be discarded if the particle number inside one cell exceeds the compile-time

setting, leading to incorrect results, though graphical applications are not usually concerned with these errors when lost particles are not visual gaps in geometries and their subsequent errors are not visually dominant after post-processing. For engineering, a more modest MAX_PPC values is often applicable and saves significant amounts of GPU memory for large-scale simulations, e.g. between 2 to 4x the initialized PPC if scenes feature nearly-incompressible materials. For highly compressible scenes, a ratio of 8x will almost always prevent particle loss, but higher values can be selected for safety. In the future, we may pursue a dynamically resizing MAX_PPC value in ClaymoreUW so that only portions of the domain that need large values will reserve large pools of memory, but it remains static as of now. We do note that ClaymoreUW has expanded the setting of MAX_PPC from using integers of base-2 to any 32-bit integer (e.g. 12, 31, 64, 100, etc.), which gives much finer control of how much memory is pre-allocated. This is an important back-end feature for exa-scale simulations, and entailed undoing hard-coded bit-wise operations that relied on the assumption of base-2 integers to avoid more intensive but typical integer operations (e.g. modulo).

Claymore and ClaymoreUW preset the maximum number of particle blocks and grid blocks (g_max_grid_blocks), as well as the maximum particle number (g_max_particle_num). These settings are adopted to enable the pre-allocation of all spatial data structures. Still, the run-time application periodically check the current demand for memory and dynamically resizes to fulfill the need. If it is not possible to resize (i.e. more memory requested of a GPU than available) then the simulation will crash with an output to the C++ standard error stream.

Different scenes require different settings, and the program works as long as the whole memory allocated does not exceed the device memory limit. In general, we recommend not using more than 50% of a GPU total memory, as the application will run faster if their is ample room for resizing data-structures when needed and it will be far less error prone.

Another assumption is that Courant–Friedrichs–Lewy (CFL) condition always holds during

run-time, indicating that the particles would move at most one-cell distances in each time step. Claymore uses a Courant-number of 0.6 to compute the CFL-bounded default stepping time in all their benchmarks. However, we recommend use of ClaymoreUW with a CFL number between 0.5 and 0.3, with our own simulation typically at 0.45. This is because the added features (e.g. PIC-FLIP mixing, F-Bar antilocking) and tendencies for engineers to use much stiffer materials undergoing more complex constitutive laws necessitate more stringent time-stepping to avoid introduction of even small errors and poor resolution of pressure waves in the medium. We provide run-time control of CFL numbers specific to any particle object's material model in ClaymoreUW's input script. Wang et al. 2020b did consider material stiffness when computing the default stepping time for stability requirements, however, ClaymoreUW introduces an automatic calculation of this time appropriate to selected material parameters. This is for user convenience, as determining the required time-step manually in every simulation is tedious and prone to error. We refer readers to (Bai and Schroeder 2022) for a deeper understanding of potential time-step conflicts in explicit MPM and MLS-MPM (e.g. near-boundary and lone particle instabilities). During run-time, Claymore computes the maximum of the grid velocity and calculates a stepping time to ensure particles do not move more than one-cell distance. The final stepping time is chosen as the minimum of the computed time among all devices and the default stepping time, which is material stiffness considerate. The CFL condition is crucial for the correctness of the results, and the G2P2G kernel will crash if it is violated, leading to failures as would happen in traditional CPU and GPU solvers. This occurs because the G2P2G kernel must write particles back to the grid after it advects them, but it only has access to a limited view of the grid (namely, the 8 x 8 x 8 grid-cell arena around the centered 4 x 4 x 4 grid-cell particle-block) so any particle whose advection moves it further than one grid-cell will cause significant errors. For instance, the particle will not contribute to the grid (a fundamental MPM step) and its registration within the particle cell buckets data-structure will fail (holds the relative particle ID used to map

particles between particle bins) so the particle simply disappears in the next step.

4.13.2 Hierarchical Data Structure Composition

Since the efficacy of the data structure is usually hardware- and algorithm-dependent, it often requires non-trivial engineering efforts to explore different choices. Therefore, the ability to quickly design and benchmark new data structures for a specific task can significantly reduce code complexity.

4.13.3 Data-Oriented Design Philosophy.

Due to the increased overhead of memory operations, data-oriented design philosophy has been widely adopted in HPC. Following this design principle, Hu et al. 2019a introduces a high performance programming language, Taichi, wherein dedicated data structures can be developed by assembling components of different properties in static hierarchies. Taichi provides a powerful and easy to-use tool-chain for developing a wide range of high-performance applications. It implements an abstraction to define multi-level spatial data structures and kernel functions through a user-friendly python front-end and a robust LLVM back-end that automatically handles memory, manages executions, and deploys to CPU or GPU. Still, there are two major issues in Taichi that prevents us from directly adopting it when developing multi-GPU-tailored MPM algorithms: **(i)** no access to low-level operations, including CUDA warp intrinsics, and **(ii)** lack of multi-GPU support. Therefore, in our implementation, we refer to the data structure description described in Taichi as the mini-language and build up the infrastructure within our C++ code-base with the following improvements:

4.13.4 C++ Oriented Programming

Unlike developing a new compiler as in the Taichi programming language (Hu et al. 2019a), we intend to develop a tool that can be directly used in both native C++ and CUDA C++. The latest standard supported by CUDA is C++14, thus it is the minimum requirement for compilation. However, it is recommended, and likely the default if configuring a system in the past four years, to use C++17 compatible compiler version. Function definitions are decorated with constexpr keyword whenever applicable on both the host and the device-side.

4.13.5 Structural Composition

The C++ template meta-programming is adopted to implement the infrastructure. Most setups, including hierarchy, layout, the relationship of elements, etc., are known beforehand and can be statically specified as template parameters. Hence, the access interface and the internal composition of the customized data structure are specified.

4.13.6 Memory Management

The representations of memory handles vary across APIs for GPU computing. For CUDA C++ (NVIDIA 2023), the memory handle of the device memory is simply a pointer on the host; the cost of copying is trivial. Thus, the memory handle can be value-copied to CUDA kernel functions from the host-device. The specific type of memory (e.g., unified virtual memory or device memory) that the variable is allocated with is determined by the allocator given at the run-time. The instance does not own the handle of the allocator so that its lifetime could be managed by programmers explicitly. In our C++ code-base, we follow the same principle emphasized by the data-oriented design principle: the internal data structure should be highly compositional and shielded under a set of high-level access interfaces. Specifically, Structural Nodes can be associated with child nodes recursively for multi-level hierarchy composition, and

the accompanying Decorator specifies the property of the node itself. For high performance, most specifications of the structure are performed at compile-time. We provide these utilities through C++ variadic templates in the following form:

```
domain<Tn, Ns...>; // Tn: Index Type, Ns: Multi dimensional coordinates of type Tn
enum attrib_layout{aos,soa};
enum structural_type{entity, hash, dense, dynamic};
decorator<structural_allocation_policy, structural_padding_policy, attrib_layout>;
structural<structural_type, domain, decorator, structurals...>;
```

4.13.7 C++ Implementation.

For fine details, i.e. the full code, please refer to the open-sourced Claymore code-base by Wang et al. 2020b. The same back-end data-structure code is preserved in ClaymoreUW, with added comments to assist in navigating the various library files employed. The fundamental data structure composition infrastructure consists of four major components: Domain, Decorator, Structural Node, and Structural Instance, which are described as follows:

Domain. Domain describes the range for the index of a data structure. It maps from multi-dimensional coordinates to a 1D memory span.

```
template<typename Tn, Tn Ns...>
struct domain {
    template<typename... Indices>
    static constexpr Tn offset(Indices&&... indices);
};
```

Decorator. Decorator describes the auxiliary and detailed properties regarding the data structure it decorates.

```
enum class structural_allocation_policy : std::size_t {
    full_allocation = 0,
```

```
    on_demand = 1,
    ...
};
enum class structural_padding_policy : std::size_t {
    compact = 0,
    align = 1,
    ...
};
enum class attrib_layout : std::size_t {
    aos = 0,
    soa = 1,
    ...
};
template <structural_allocation_policy alloc_policy_, structural_padding_policy
↪ padding_policy_, attrib_layout layout_>
struct decorator {
    static constexpr auto alloc_policy = alloc_policy_;
    static constexpr auto padding_policy = padding_policy_;
    static constexpr auto layout = layout_;
};
```

Structural Node. Structural Nodes with particular properties are formed in a hierarchy to compose a multi-level data structure. Currently, we support three types of structural nodes (i.e., hash, dense, and dynamic), same as in Hu et al. 2019a. Hash is appropriate for hash-table type data-structures. Dense is for densely packed data of a given size (e.g. a grid-block structure). Dynamic is for dynamically resizing data-structures (e.g. the grid-buffer super-structure which contains dynamically changing counts of the dense grid-block structure as a simulation progresses).

```
enum class structural_type : std::size_t {
```

```
    /// leaf
    sentinel = 0,
    entity = 1,
    /// trunk
    hash = 2,
    dense = 3,
    dynamic = 4,
    ...
};
```

No matter what the internal relationship of elements is within a structure (either contiguous- or node-based), we assume there is at least one contiguous chunk of physical memory to store the data; the size is a multiple of the extent of the Domain and the total size of all the attributes of an element.

```
/// attribute index of a structural node
using attrib_index = placeholder::placeholder_type;
/// traits of structural nodes
template <structural_type NodeType, typename Domain, typename Decoration, typename...
↪ Structurals>
struct structural_traits {
using attribs = type_seq<Structurals...>;
using self = structural<NodeType, Domain, Decoration, Structurals...>;
template <attrib_index I>
using value_type = ...;
static constexpr auto attrib_count = sizeof...(Structurals);
static constexpr std::size_t element_size = ...;
static constexpr std::size_t element_storage_size = ...;
/// for allocation
static constexpr std::size_t size = domain::extent * element_storage_size;
template <attrib_index AttribNo> struct accessor {
```

```
16      static constexpr uintptr_t element_stride_in_bytes = ...;
17      static constexpr uintptr_t attrib_base_offset = ...;
18      template <typename... Indices>
19      static constexpr uintptr_t coord_offset(Indices &&... is) {
20          return attrib_base_offset + Domain::offset(std::forward<Indices>(is)...) *
            ↪  element_stride_in_bytes;
21      }
22      template <typename Index>
23      static constexpr uintptr_t linear_offset(Index &&i)
24      {
25          return attrib_base_offset + std::forward<Index>(i) * element_stride_in_bytes;
26      }
27  };
28  // manage memory
29  template <typename Allocator> void allocate_handle(Allocator allocator) {
30      if (self::size != 0)
31          _handle.ptr = allocator.allocate(self::size);
32      else
33          _handle.ptr = nullptr;
34  }
35  template <typename Allocator> void deallocate(
36  Allocator allocator) {
37      allocator.deallocate(_handle.ptr, self::size);
38      _handle.ptr = nullptr;
39  }
40  // access value
41  template <attrib_index ChAttribNo, typename Type = value_type<ChAttribNo>, typename...
    ↪  Indices>
42  constexpr auto &val(std::integral_constant<attrib_index, ChAttribNo>, Indices &&...
    ↪  indices)
```

```
{
    return *reinterpret_cast<Type *>(_handle.ptrval +
    ↪  accessor<ChAttribNo>::coord_offset(std::forward<Indices>(indices)...));
}
template <attrib_index ChAttribNo, typename Type = value_type<ChAttribNo>, typename
↪  Index>
constexpr auto &val_1d(std::integral_constant<attrib_index, ChAttribNo>, Index &&index) {
    return *reinterpret_cast<Type *>(_handle.ptrval +
    ↪  accessor<ChAttribNo>::linear_offset(std::forward<Index>(index)));
}
/// data member
MemResource _handle;
};
/// specializations of different types of structural nodes
template <typename Domain, typename Decoration, typename... Structurals>
struct structural<structural_type::hash, Domain, Decoration, Structurals...> :
↪  structural_traits<structural_type::hash, Domain, Decoration, Structurals...> {...};
...
```

We define two types of Structural Nodes, the root node and the leaf node, to form the hierarchy.

```
/// special structural node
template <typename Structural> struct root_instance;
template <typename T> struct structural_entity;
```

Structural Instance. A variable defined by a Structural Node is an Structural Instance spawned given an allocator at the run-time. The instance is customizable (e.g., accessing the parent node requires additional data) as it is assembled from data components.

```
enum class structural_component_index : std::size_t {
    default_handle = 0,
```

```
    parent_scope_handle = 1,
    ...
};
template <typename ParentInstance, attrib_index, structural_component_index>
struct structural_instance_component;
/// specializations for each data component
template <typename ParentInstance, attrib_index>
struct structural_instance_component<ParentInstance, attrib_index,
↪ structural_component_index::parent_scope_handle> {...};
...
```

Besides the data components, the Structural Instance also inherits from the Structural Node that specifies the properties of itself.

```
/// traits of structural instance, inherit from structural node
template <typename parent_instance, attrib_index AttribNo>
struct structural_instance_traits: parent_instance::attribs::template
↪ type<(std::size_t)AttribNo> {
    using self = typename parent_instance::attribs::type<(std::size_t)AttribNo>;
    using parent_indexer = typename parent_instance::domain::index;
    using self_indexer = typename self::domain::index;
};
/// structural instance, inherit from all data components and its traits (which is
↪ derived from structural node)
template <typename ParentInstance, attrib_index AttribNo, typename Components>
struct structural_instance;
template <typename ParentInstance, attrib_index AttribNo, std::size_t... Cs>
struct structural_instance<ParentInstance, AttribNo, std::integer_sequence<std::size_t,
↪ Cs...>> : structural_instance_traits<ParentInstance, AttribNo>,
↪ structural_instance_component<ParentInstance, AttribNo,
↪ static_cast<structural_component_index>(Cs)>... {
```

```
    using traits = structural_instance_traits<ParentInstance, AttribNo>;
    using component_seq = std::integer_sequence<std::size_t, Cs...>;
    using self_instance = structural_instance<ParentInstance, AttribNo, component_seq>;
    template <attrib_index ChAttribNo>
    using accessor = typename traits::template accessor<ChAttribNo>;
    // hierarchy traverse
    template <attrib_index ChAttribNo, typename... Indices>
    constexpr auto chfull(std::integral_constant<attrib_index, ChAttribNo>, Indices &&...
    ↪ indices) const {
        ...
    }
    template <attrib_index ChAttribNo, typename... Indices>
    constexpr auto ch(std::integral_constant<attrib_index, ChAttribNo>, Indices &&...
    ↪ indices) const {
        ...
    }
    template <attrib_index ChAttribNo, typename... Indices>
    constexpr auto chptr(std::integral_constant<attrib_index, ChAttribNo>, Indices &&...
    ↪ indices) const {
        ...
    }
};
```

4.13.8 Example Usage

To showcase the usages of Structural in C++ this section provides a set of examples that describes a GPU SPGrid. Firstly, the definitions below are useful short-hands:

```
/// leaf node
using empty_ = structural_entity<void>;
using i32_ = structural_entity<int32_t>; // single-precision signed integer
```

```
using f32_ = structural_entity<float>; // single-precision float
using i64_ = structural_entity<int64_t>; // double-precision signed integer
using f64_ = structural_entity<double>; // double-precision float
/// attribute index
namespace placeholder {
    using placeholder_type = unsigned;
    constexpr auto _0 = std::integral_constant<placeholder_type, 0>{};
    constexpr auto _1 = std::integral_constant<placeholder_type, 1>{};
    ...
}
/// default data components for constructing instances
using orphan_signature = std::integer_sequence<std::size_t, static_cast<std::size_t>(
    structural_component_index::default_handle)>;
```

The following code defines the SPGrid data structure within Claymore and ClaymoreUW:

```
Definition of GPU SPGrid.
// domain
using BlockDomain = domain<char, 4, 4, 4>; // block-domain in 3D, 4x4x4
using GridBufferDomain = domain<int, g_max_active_block>; // grid-buffer domain, element
    per max grid-blocks
// decorator
using DefaultDecorator = decorator< structural_allocation_policy::full_allocation,
    structural_padding_policy::compact, attrib_layout::soa>;
// structural node
using grid_block_ = structural<structural_type::dense, DefaultDecorator, BlockDomain,
    f32_, f32_, f32_, f32_>; // single-precision 4x4x4 grid-block
using grid_buffer_ = structural<structural_type::dynamic, DefaultDecorator,
    GridBufferDomain, grid_block_>; // single-precision grid-buffer
```

After defining the internal structure, it still requires an allocator and the list of data components to get the instance and actually have it occupy GPU memory in a simulation:

```
template <typename Structural, typename Signature = orphan_signature>
using Instance = structural_instance<root_instance<Structural>, (attrib_index)0,
↪  Signature>;
template <typename Structural, typename Componenets, typename Allocator>
constexpr auto spawn(Allocator allocator) {
    auto ret = Instance<Structural, Componenets>{};
    ret.allocate_handle(allocator);
    return ret;
}
auto allocator = ...;
auto grid = spawn<grid_buffer_, orphan_signature>(allocator);
using Attrib0 = structural_entity<float>;
using Attrib1 = structural_entity<double>;
using DecoratorA = decorator<structural_allocation_policy::full_allocation,
↪  structural_padding_policy::align, attrib_layout:aos>;
using DecoratorB = decorator< structural_allocation_policy::full_allocation,
↪  structural_padding_policy::align, attrib_layout:soa>;
using StructuralA = structural<structural_type::dense, DecoratorA, domain<int, 4, 4>,
↪  Attr0, Attr1>;
using StructuralB = Structural<structural_type::dense, DecoratorB, domain<int, 4, 4>,
↪  Attr0, Attr1>;
```

To access hierarchical elements of the GPU SPGrid in a function, it is neccesary to index as follows:

```
/// acquire blocknoth grid block
auto grid_block = grid.ch(_0, blockno);
/// access cidib\th cell within this block with 1D coordinate
grid_block.val_1d(_0, cidib); // access 0th channel (mass)
/// access cell within by 3D coordinates
grid_block.val(_1, cx, cy, cz); // access 1th channel (velocity x)
```

Memory layout can be visually interpreted for data-structures for simpler comprehension. Two types of Structural Nodes with different Decorators (Array-of-Structures and Structure-of-Arrays, i.e. AoS and SoA, respectively) are illustrated in Figure 4.8 to explain the underlying memory layout. Both are very important organization schemes to understand in high-performance computing. Further, they are often seen in Array-of-Structures-of-Arrays (AoSoA) format in recent years, i.e. a combination of AoS and SoA to promote hierarchical access of groups of particles, nodes, etc. which may then be loaded into GPU memory in a coalesced, efficient manner.

4.13.9 Concluding Remarks

This completes our brief summary of both a high-level view on the data-structures used in ClaymoreUW, as well as the low-level configuration in CUDA C++ that originates from Claymore, by Wang et al. 2020b, which is based on the Taichi programming language by Hu et al. 2019a. Note that there is still significant room to optimize these structures. Namely, for the hash-map which is not currently sparse (it creates an entry per each grid-block within the maximum 3D domain that can be very large for very large max domains), and for the cell and block particle buckets (which hold relative IDs of particles for a cell which aggregate into blocks, and are not currently sparse or dynamically resized so all cells in all active grid-blocks will reserve MAX_PPC entries for themselves. This takes a lot of memory if MAX_PPC is large and many grid-blocks are active).

4.14 *ClaymoreUW - Frequently Asked Questions*

A few common tasks a user of Claymore-for-Engineers (i.e. ClaymoreUW) may need to know are listed here. This is not an exhaustive user-manual but serves as a starting point for an otherwise high learning-curve software.

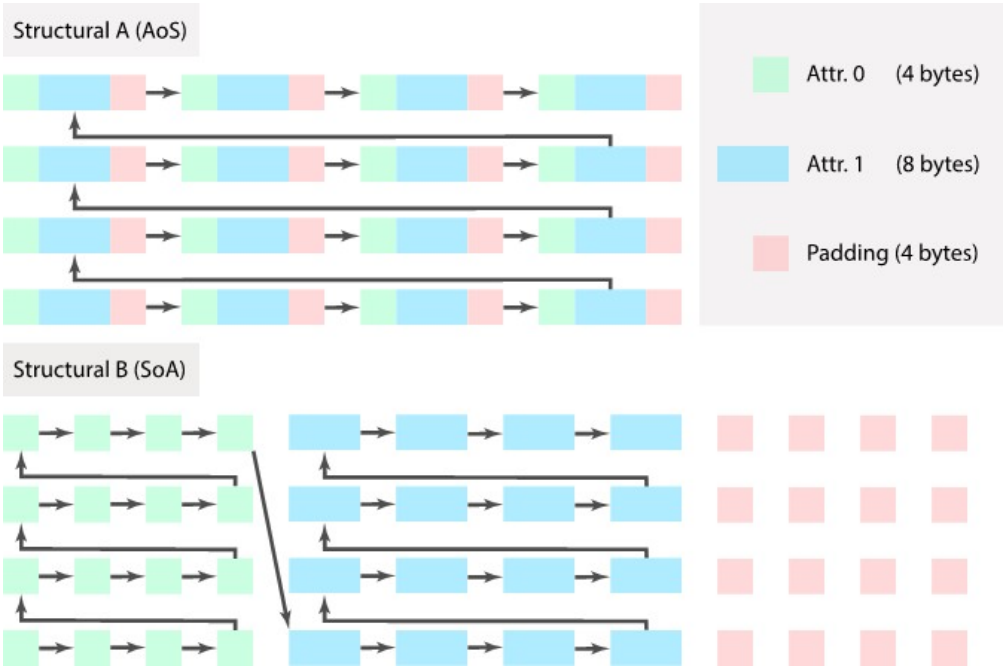

Fig. 4.8. Spatial structure specification in Claymore's hierarchical composition scheme for C++. Two data-structures are specified with different decorators. The arrows connecting all elements indicate the ascending order in a contiguous chunk of memory. The structures can be used as a child of another structural to form a multi-level hierarchy. Elements displayed in the grid view are accessed by a child structural index (marked with different colors) and a coordinate within its domain. Note that the memory size of each structural object is padded to the next power-of-2 due to the alignment decoration. Figure adapted from the supplementary technical document provided by Wang et al. 2020b. Their work was based on that of Taichi Hu et al. 2019a. This system is directly imported into ClaymoreUW. All rights of original authors and publishers are reserved.

Adding a material model. A couple of basic material models (weakly-compressible fluid, fixed-corotated solid, Drucker-Prager, modified Non-Associative Cam-Clay) are already implemented. Implementing a new material is a basic capability of an MPM software. Here we provide a walk-through for this task. Most difficulty is from navigating Claymore's file-structure and making new GPU functions to support a material, but this essentially amounts to copy-pasting specific code that is highlighted here. To add a material, go to `particle_buffer.cuh` and check if any currently defined particle representations apply to your particle (i.e. if your material needs one more float on particles than any previous model you will need to copy-paste existing code, e.g. `particle_bin8` as `particle_bin9`, and add an extra float value (f32). The means to do this is fairly apparent in the file. Then, define the material itself at the bottom of the file where the particle buffer is defined as a variant of possible materials, e.g. "Test-Material", and have it reference the appropriate particle data structure you just created. Copy code for any similar material Particle Buffer and alter it to reference "Test-Material", as well as changing it to include any material parameters (e.g. shear modulus). Go to `settings.h` and add "Test-Material" to the material list. Definition of the material in terms of memory allocation and structure is complete. Now go to `constitutive_models.cuh` and add the constitutive model that defines the material, i.e. given a deformation gradient and needed material parameters what is the resultant stress. This is no different than a material model in any other software, but it is written as an inline device function so be mindful that it operates on GPU data. Finally go to `mgmpm_kernels.cuh` and copy one of every material specific function, changing its header to use "Test-Material". The contents of the functions may or may not need to change, it will be material specific, but at the minimum it should account for any change in particle data usage as set by the particle bin structure defined at the start. For example, `g2p2g()` should call the appropriate constitutive model function for your material and it should read and write to the appropriate memory addresses in the particle bin structure you defined. Despite a fairly large number of steps, you mostly just copy-paste with minimal

modifications, leveraging highly optimized Multi-GPU code with low-overhead.

Input data. Currently the expanded Claymore code supports *.sdf, *.csv, and *.pos binary files to create MPM particles. The former will actually sample particles from a signed-distance-field according to a given PPC, while the latters just read in particle positions directly. Vertices and Elements for our FEM coupling use *.csv files only. Claymore does not natively support reading in stress, velocity, etc. on a particle basis, but we have built in functionality so we may read in *.bgeo files (a standard output format for particles and their attributes in ClaymoreUW) directly to imbue particles with initial attributes, i.e. a "checkpoint" simulation state. This can be useful for reusing previous frame outputs from other simulations (i.e. picking up a simulation where it left off).

Making a Signed-Distance-Field input file. Open-source command-line tool SDFGen translates common triangle mesh based *.obj files into signed-distance-field *.sdf files. As a signed-distance-field, each cell in a grid covering the object knows its distance from the manifold. This format is the preferred way to generate MPM particles and apply non-conforming boundaries because it is scale invariant if the SDF resolution is fine enough for the object, it provides distance from manifold (i.e. body) which is valuable for determining the surface/near-surface/body of an object, and it can provide surface normals easily which is often challenging for MPM particles and non-conforming boundaries.

Output particle data. If a user wants to pull, for example, particle Cauchy stress out of an object in a simulation, how would they do this? First, ensure that each particle in the object (i.e. Particle Buffer) is equipped with memory space to store a stress tensor (additional 3x3 floats). This is not set by default in `particle_buffer.cuh`. Second, tell particles to write their stress tensors to their memory within the material specific G2P2G kernel in `mgmpm_kernels.cuh`, this is just a copy-paste as particles already write values into their memory. Third, augment the current particle attribute retrieval functions `retrieve_particle_attributes()` to copy this new data from each particle in the Particle Buffer, going through it block by block, to a basic device array. Fourth, copy the device

array information to a host array asynchronously, already done in `mgsp_benchmark.cuh`. Fifth, `mgsp_benchmark.cuh` will output the host array containing the particle stresses to a desired output format (e.g. *.csv, *.bgeo) by calling output functions. PartIO, an open-source particle I/O toolbox, is used for this. Output functions are found in Claymore/Library/MnSystem/IO/ParticleIO.hpp and are easily modified as they are standard C++.

Output grid data. Our lab implemented two methods to output grid-data. The first method outputs grid-blocks only, with summed mass and momentum from composing cells. The second method outputs grid-cell information within defined domains (e.g. surfaces, volumes) in the simulation, usually providing the mass, momentum, and force on grid-nodes. The former is useful for observing overall grid behavior efficiently while the latter is good for high-resolution, targeted information. We also allow output to *.csv files for aggregated quantities, like summed force on a surface or max surface elevation in a set domain. Respectively, these replicate load-cells and wave-gauges in experiments and can be set by a user-specificed output frequency rate. Our implementation works on Multi-GPUs and the sparse data-structure of the grid in Claymore. It also leverages shared memory and warp level atomics for computing aggregate quantities. Asynchronous writes are used to go from device to host so the simulations are not interrupted when outputting large amounts of data.

Visualize data. Claymore output is provided in binary geometry (*.bgeo) format by default, though the PartIO library can convert to other common file types. To visualize *.bgeo files you can quickly use PartView (command in PartIO) or SideFX Houdini. The latter has a free license for non-commercial use and provides a massive toolbox for animation and visualization capabilities. Simply import the *.bgeo as geometry data and it will show position data in a viewing window for every available frame. Particle attributes (e.g. pressure) can be shown by opening a Scene Option in the Scene Viewer toolbar, clicking the Visualize tab, adding a new visualizer, setting it to reference the name of the attributes, and setting any other specifications for

visualization (e.g. visualize as color, vector, color-mapping, etc.). Advanced rendering (e.g. subsurface scattering, ray-tracing, etc.) can be done here as well.

4.15 *ClaymoreUW - Multi-GPU Domain Decomposition*

With scaling laws (Section 3.5), kernel design (Section 4.11), data-structure difficulties (Section 4.12), and Multi-GPU memory limits (Sections 3.3.3 and 3.4) in mind, one must consider a variety of partitioning schemes to optimize MPM simulations on multiple GPUs. Schemes considered by the author are:

- **Partition-by-Space.** Favorable to scenarios of predictable, even spatial occupation. The grid-block halo region is static and designated. Fluids, especially ones that mix and occupy a fixed domain volume, are usually done in this way.

- **Partition-by-Particles.** Ideal for particles that are expected to maintain their rough configuration relative to one-another throughout the simulation. Stiff solids behave well in this scheme.

- **Partition-by-Material.** Useful when there are multiple materials in a scene that require very different time-steps or undergo unique algorithms to advance through time. Incompressible fluids and compressible solids in the same scene may be best performed in this way.

- **Partition-by-Method.** Allows for different partitions to handle seperate numerical methods. For instance, an MPM-FEM coupling may split FEM meshed bodies from purely MPM bodies to simplify design. In our code, the FEM implementation is not as high-performance as the MPM implementation, but there are typically far more particles in a scene than elements so having one GPU handle FEM and the rest running MPM results in a time-balanced scene.

- **Mixed Partitions.** Fitting for complex scenes. For instance, we partition FEM meshed debris on one GPU, while partitioning the MPM water across multiple GPUs spatially. This leverages both Partition-by-Method and Partition-by-Space.

- **Adaptive Partitions.** Adapt throughout the simulations course. For instance, partitions may rebalance themselves according to the deformation occurring in the scene. This can account for nonlinear materials models which may slow-down as deformation grows, thus GPUs with too many particles undergoing nonlinear deformation may offload some to other, less taxed GPUs to balance computations.

4.16 *Final Remarks*

In this chapter we cover the basics of bringing the Material Point Method (MPM) onto multiple graphics processing units (Multi-GPUs). An open-source software by Wang et al. 2020b is evaluated for this task, but was not built for practicing engineers as the intended user-base. After notable modification by our group, we arrive at an augmented version of the open-source code-base, ClaymoreUW, that is retooled for engineers.
A developer and user's guide is provided for our open-source project, with the file-structure, data-structures, underlying algorithm, and input script construction all described.
Modifications made to the original Claymore (Wang et al. 2020b) by our lab, including user-interface changes, new antilocking methods, introduction of multiple advection schemes, improved computational precision, and coupling finite elements, among dozens of other changes, are described.

Chapter 5

DESIGN AND IMPLEMENTATION OF NUMERICAL METHODS FOR DEBRIS-FLUID-STRUCTURE INTERACTION IN A MULTI-GPU MATERIAL POINT METHOD

Numerical simulations, compared to experiments, are (i) cheaper, (ii) more flexible, and (iii) better posed for analysis. The benefits of augmentation with numerical models are evident, but only when using appropriate methods (Hasanpour et al. 2021). Simulating fluid-driven debris-field events (i.e. transport, impact, and damming) requires accurate debris-fluid-structure interaction in large domains undergoing large deformations– An exceedingly difficult demand even when approaching it using MPM. This chapter describes our general approach and any additional methods required, such as finite element coupling and advanced advection schemes, to simulate complex DFSI accurately.

5.1 *Numerical Approach For Debris-Fluid-Structure Interaction*

We propose a numerical approach to debris-fluid-structure interaction (DFSI) that combines various numerical methods listed in Table 5.1 with excellent scaling properties (notably on Multi-GPUs). Employing a curated set of existing methods from computer graphics and engineering approaches, the proposed method is a modification of the explicit Material Point Method (MPM) suited for accurate engineering simulations of generalized DFSI by means of high-performance computing.

Prioritizing simplicity, speed, and flexibility leads to an effective algorithm for generalized DFSI. Although some of these methods may initially seem straightforward or brute-force for practical

engineering applications, their collective integration offers the desired features for the problem we address. It is important to highlight that the true value of these methods is fully realized when applied to massive simulations, achieving remarkable acceleration rates that surpass most existing engineering software, particularly when leveraging Multi-GPUs. On the contrary, popular but complex methods may degrade with scale. A shortlist of requirements to properly characterize DFSI and our approach's answers are:

1. **Multi-Material and Multi-Phase Interaction.** MLS-MPM and MPM manage interaction across disparate materials and phases through the use of a shared grid (i.e. mass-velocity field) which balances momentum across all objects indiscriminately. This is feasible but harder to do in other popular methods such as finite element analysis (FEA), smoothed particle hydrodynamics (SPH) (Hasanpour et al. 2021) (Domínguez et al. 2022), and computational fluid dynamics (CFD) (Lewis et al. 2023). This natural property of MPM has significant value for the stable interaction of fluids, structures, and debris in a multitude of engineering scenarios.

2. **Contact Mechanics.** Necessary for DFSI, MLS-MPM and MPM have basic, reliable contact out-of-the-box. However, it is at a grid-cell resolution and "sticky" (e.g. water may

Table 5.1. DFSI numerical requirement specification sheet

DFSI Requirement	Numerical Solution
Multi-Material / Phase Interaction	MPM / MLS-MPM
Contact Mechanics	MPM / MLS-MPM, ASFLIP
Material Models	Weakly-Compressible Fluid, Fixed-Corotated, Neo-Hookean, Drucker-Prager, Cam-Clay
Large-Deformation Dynamics	MPM / MLS-MPM
Damping and Noise Mitigation	APIC, PIC-FLIP Mixing, F-Bar Antilocking
Compatibility With Other Methods	MPM / MLS-MPM
Timely Execution	MLS-MPM, ASFLIP, F-Bar, Multi-GPU Design

latch onto debris, debris may stick together). Sub-grid and "slip" contact is needed in many problems. We use the Affine-Separable Fluid-Implicit-Particle method (ASFLIP, Fei et al. 2021a), a graphics technique (e.g. animating thin hair strands), for sub-grid non-sticky contact. Although inferior to advanced contact models, ASFLIP is very easy to implement and fast while producing reasonable contact for DFSI. Magnifying grid-resolution with GPU implementations also improves contact substantially.

3. **Material Models.** DFSI involves anything from cars to mud slurries. Broad-reaching models for fluids and solids are desired for DFSI and are supported in MPM. For fluids (e.g. water, mud), we take a common weakly-compressible model seen often in SPH (Becker and Teschner 2007). It is a simple combination of the Murnaghan-Tait equation of state by Murnaghan 1944 and Tait 1888 with added viscous stress from Pradhana et al. 2017 in our Multi-GPU optimized implementation (PA-JB Fluid, Bonus 2023). Incompressible fluid models are viable (Zhang et al. 2017) (Vittoz et al. 2019), but have a reduced scope and increased complexity. For debris and structures we use the Fixed-Corotated model (efficient, large-deformation, hyper-elastic) (Jiang et al. 2016). Non-Associative Cam-Clay and Drucker-Prager solid models are also available, expanding the available DFSI behavior to materials such as concrete, clay, and sand.

4. **Large-Deformations Dynamics.** Needed for DFSI (e.g. debris transport), MLS-MPM and MPM are well-suited due to "resetting" background grids (Sulsky et al. 1994), avoiding common large-deformation numerical obstacles (e.g. mesh tangling in FEA).

5. **Damping and Noise Mitigation.** Detrimental to sensitive DFSI phenomena, MPM is poorly-suited without modifications in four key aspects: **(i)** Particle cell-crossing errors, derived from the discontinuous gradients of linear shape-functions, cause "checker-boarding" errors in stress and strain. **(ii)** Null-space errors present in MPM if

particles greatly outnumber grid-nodes (i.e., the majority of MPM simulations) which are potentially detrimental for simulations with stiff materials and large rotations. These errors derive from the reduced degrees-of-freedom for MPM particle-to-grid transfer (P2G) and from the inverse in grid-to-particle transfer (G2P) (Tran and Sołowski 2019) (Hammerquist and Nairn 2017). **(iii)** Computational costs constraining grid resolution and thereby physical behavior (e.g. damps small vortexes). **(iv)** Volumetric locking of stress fields in explicit MPM that deteriorate even basic hydrostatic simulations (Yang 2016). To tackle these four issues we propose: **(i)** Use simple shape-functions, such as quadratic B-Splines, to eliminate cell-crossing error and garner additional benefits later. **(ii)** Apply the Affine Particle-in-Cell method (APIC, included in MLS-MPM and ASFLIP) (Jiang et al. 2015) to better conserve angular velocity in a simple, stable manner. Any remaining velocity damping is treated with PIC-FLIP velocity mixing (Stomakhin et al. 2013). PIC-FLIP mixing is a linear combination of PIC (Harlow 1962) and FLIP (Brackbill and Ruppel 1986) that bypasses part of MPM's grid filtering while maintaining good stability. It is a subset of ASFLIP's implementation. **(iii)** Magnify grid resolution via Multi-GPU acceleration, brute-forcing granularity 10 - 1000x to capture refined physical phenomena. **(iv)** Vastly improve strains/stresses at minimal effort with simple volumetric anti-locking by Zhao et al. 2023, a scheme compatible with B-Splines that uses no new data-structures and only introduces two extra values on grid-nodes and one on particles. Other more traditional and/or advanced schemes exist to improve MPM noise and damping (Mast 2013, Tran and Sołowski 2019, Yang et al. 2019, and Nairn and Hammerquist 2021), but are too involved/costly for high-performance DFSI or conflict with other methods and parallel implementations.

6. **Compatibility.** Specific DFSI problems may require further numerical capabilities. MLS-MPM and MPM are highly compatible with both Eulerian and Lagrangian numerical schemes because they are hybrid Eulerian-Lagrangian methods. MPM can be coupled with

relative ease to FEA, SPH, CFD, etc., to leverage their strengths (e.g. improved fluid flow, structural behavior). We couple Finite Elements (Irving et al. 2004, Irving et al. 2006, and de Vaucorbeil et al. 2020) in our Multi-GPU implementation as proof-of-concept and apply it in Section 6.6.

7. **Timely Execution.** Computation cost for large DFSI simulations can be prohibitive, especially in MPM. To address this, we apply MLS-MPM to halve simulation time and add both ASFLIP (Fei et al. 2021a) and F-Bar anti-locking (Zhao et al. 2023) to optimize cost-benefit ratios in the aforementioned aspects. The algorithmic simplicity of this approach is suited to acceleration via Multi-GPUs over complex alternatives, shown in Section 6.4 via a billion particle fluid simulation.

To demonstrate the efficacy and portability of these methods, we have applied them to a computer graphic's Multi-GPU software originally by Wang et al. 2020b which we have repurposed for engineering problems, accelerating simulation of DFSI by 100x over traditional engineering codes– bringing simulation times from days to hours. This breakthrough not only showcases the immense potential of our approach, but also enables the execution of high-resolution 3D parametric analyses within realistic time frames. In Chapter 6, we employ benchmark studies to validate the effectiveness of the proposed method. The remainder of this chapter covers the necessary technical details for understanding the implementation and formulation of all novel designed and existing selections of DFSI methods that we have assembled.

5.2 *Time-Integration*

Explicit integration is used in our implementation. Noting the study by Bai and Schroeder 2022 which characterized the complex influence of solitary particles, boundaries, and material pressure waves on time-step instabilities in explicit B-Spline PIC, CPIC, and APIC-based MPM schemes,

we still choose exclusively explicit time integration. Benefits of explicit time integration are: **(i)** implementation simplicity, **(ii)** computational simplicity, and **(iii)** greater compatibility with many interesting methods. For instance, APIC (Sec. 2.4) (Jiang et al. 2015) naturally conserves more angular momentum for explicit formulations of MPM, but the case of implicit time-integration will require extra treatment for the same result (Jiang et al.). The drawbacks of explicit time integration are: **(i)** smaller time-steps, **(ii)** increased risk of "checkerboard" stress behavior (i.e. for stiff materials), and **(iii)** compounding of noise deriving from advection modifications (e.g. PIC/FLIP mixing, Section 5.6.7) due to more time-steps (Bai and Schroeder 2022).

The Courant-Friedrichs-Lewy Condition (CFL Condition) is used to determine the time-step of all our simulations. Critical time-step Δt is set by

$$\Delta t = c\frac{\Delta x}{v_{\max}}\,, \tag{5.1}$$

where c is the CFL factor (typically 0.5), Δx is the grid-node spacing in the MPM simulation, and $v_{\max}$ is the maximum velocity in the simulation which is typically taken as the pressure wave velocity for an isotropic, weakly-compressible fluid:

$$v_{\max} = \sqrt{\frac{k}{\rho}}\,, \tag{5.2}$$

where k is bulk modulus and ρ is density. Note that stiffer materials will inherently need a smaller time-step. For water, the max velocity could be changed from the pressure wave speed to shear wave speed if using an imcompressible formulation. Shear waves are substantially slower the pressure waves and thus allow larger time-steps, though we do not pursue this.

If coupling Finite Elements with MPM, the CFL condition may need to consider a Δx that is set by a characteristic length. For tetrahedron elements this is the minimum tetrahedron internal diameter. One should also use a c factor inline with standard FEM as opposed to MPM. Our

simulations tend to use FEM elements that are not substantially smaller than the MPM grid-cells, hence we still achieve stability with $c = 0.5$ and a Δx set by the MPM grid.

5.3 *Boundary Conditions*

Boundary conditions are essential. Numerical simulations, in most cases, resolve partial differential equations carried out for some scenario. Scenarios are set to behave properly at locations where we know, or can reasonably claim, specific behavior must occur (e.g. rigidity, temperature, velocity). By bounding properly, non-boundary locations will exhibit good behavior if the numerical tool can accurately solve the PDEs over the domain.

The Material Point Method (MPM) applies boundary conditions to its shared grid (i.e. a mass and velocity field). Because the grid is shared and automatically handles multi-phase and multi-material interaction, applying BCs here holds the property invariant and is thereby extremely valuable. MPM typically uses a uniform Cartesian grid (i.e. evenly spaced grid with cells making squares or bricks for 2D or 3D). This further simplifies boundary conditions if boundaries align with the grid. However, scene boundaries that do not perfectly align with the MPM grid often occur, resulting in MPM errors proportional to grid-spacing (Δx). This is a limitation of all MPM implementations unless more advanced methods are used. Keeping this in mind, a projection operator (**Proj**) (Hu et al. 2018) for boundary conditions used in our code can be broadly stated as:

$$\mathbf{v}_i^* = \mathbf{Proj}(\mathbf{v}_i, \mathbf{n}_i, \mathcal{B}, \mu_c) = \begin{cases} \mathbf{0} & \text{where } \mathcal{B} \text{ is sticky,} \\ \mathbf{v}_t & \text{where } \mathcal{B} \text{ is slip,} \\ \zeta \mathbf{v}_t & \text{where } \mathcal{B} \text{ is separate and } \mathbf{v}_r \cdot \mathbf{n}_i \leq 0, \\ \mathbf{v}_i & \text{where } \mathcal{B} \text{ is separate and } \mathbf{v}_r \cdot \mathbf{n}_i > 0 \ , \end{cases} \tag{5.3}$$

where $\mathcal{B}$ is the boundary type (e.g. sticky, slip, separable), $\mathbf{v}_i$ is the grid-node velocity, $\mathbf{v}_b$ is the

boundary velocity, $\mathbf{v}_r$ is relative velocity ($\mathbf{v}_r = \mathbf{v}_b - \mathbf{v}_i$), $\mathbf{n}_i$ is the boundary surface normal on the grid-node, $\mathbf{v}_t$ is relative velocity tangent to the boundary ($\mathbf{v}_t = \mathbf{v}_r - \mathbf{v}_r \cdot \mathbf{n}_i$), μ_c is a Coulomb friction parameter (either static or dynamic), and ζ is a tangential contact factor (i.e. set using μ_c).

Coarse grids and non-conforming boundaries present a challenge to MPM. Even a simple inclined surface may produce "stair-stepping" effects (i.e. a smooth ramp appears as a staircase with steps of Δx) because it is at an angle relative to the grid. For our wave flume simulations this is especially worrisome since water travels up what is intended to be a smooth ramp but meets fictitious resistance. Worse, grid coarseness can result in particles going past the intended boundary surface, substantially altering the dynamics of the simulation.

Many methods exist to handle irregular boundary conditions in MPM, ranging from ghost-particles to multi-grid methods. In this work we settle on a simple approach that adds a "decay" layer to boundaries (Yang 2016). BCs on grid-nodes are tuned proportional to distance from the boundary surface (r) over a characteristic distance (h), usually a grid-cell spacing ($h = \Delta x$), by a factor γ_{BC}. The grid is split into three regions: a free zone (nodes not influenced by BC, $r > h$), decay zone (nodes influenced by BC, $0 < r \leq h$), and a constrained zone (nodes on or inside the boundary, $r \leq 0$). Over these domains γ_{BC} takes the form of:

$$\gamma_{\text{BC}} = \begin{cases} 0 & \text{where } \mathbf{x}_i \text{ is in free zone.} \\ \gamma_{\text{Decay}} & \text{where } \mathbf{x}_i \text{ is in decay zone,} \\ 1 & \text{where } \mathbf{x}_i \text{ is in constrained zone,} \end{cases} \tag{5.4}$$

where γ_{Decay} is a function that bridges conditions in the free and constrained zones. Ideally γ_{Decay} should be a smooth function that connects the free and constrained zones so no jump is observed in γ_{BC} or its derivative. Inside γ_{BC} we define a decay function with the following forms:

$$\gamma_{\text{Decay}} = \begin{cases} 0 & \text{Free Decay.} \\ (1 - \frac{r}{h}) & \text{Linear Decay.} \\ (1 - \frac{r}{h})^2 & \text{Quadratic Decay.} \\ 1 - (\frac{r}{h})^2 & \text{Flipped Quadratic Decay.} \\ \frac{1}{2}\left(1 - \cos\left(\frac{\pi r}{h}\right)\right) & \text{Cosine Decay.} \\ 1 & \text{Constrained Decay.} \end{cases} \tag{5.5}$$

In our simulations we favor Quadratic decay because it allows particles to get very close to boundary surfaces without penetration for our choice of boundary geometries and grid resolution. Note that γ_{Decay} is ultimately arbitrary, it only exists to bridge the gap between free-space and a boundary that is not aligned with the MPM grid. Users should choose Constrained Decay if they want to guarantee non-penetration of a boundary (at the cost of over-extending a boundaries influence) and Free Decay for regular MPM contact.

5.4 *Material Models*

The primary strength of MPM is its inclusion of simple and complex material models alike, with interaction between them being handled naturally. Here we favor "simple" models in the material sense but complex in their ability to express dynamics through large-deformations.

Recall that the large-deformation mechanics being used tend to favor computation in the form of the first Piola-Kirchoff stress ($\mathbf{P}$) and deformation gradient ($\mathbf{F}$), instead of cauchy stress ($\boldsymbol{\sigma}$) and infinitesimal strain ($\boldsymbol{\varepsilon}$). Within our constitutive models on the particles we maintain the prior form but typically convert to the latter for certain force calculations and/or grid-transfers.

For reference, we include here useful equations to relate undeformed to deformed states and the

first Piola-Kirchoff stress to Cauchy stress:

$$\boldsymbol{\sigma} = \frac{1}{J}\mathbf{P}\mathbf{F}^T \quad \text{where} \quad J = \det(\mathbf{F}), \quad V = V^0 J \ . \tag{5.6}$$

Conversion formula between common elastic moduli relevant to implemented models are also provided. Note that any two elastic moduli can be used to find any other elastic moduli, however we only include a few equations here:

$$\mu = \frac{E}{2(1+\nu)}, \quad \lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}, \quad k = \frac{E}{3(1-2\nu)} \ , \tag{5.7}$$

where μ is the shear modulus, λ is the first Lame parameter, E is the Young's modulus, and ν is Poisson's ratio.

For water, a Weakly-Compressible Isotropic Fluid (a.k.a J-Fluid) (Pradhana et al. 2017) is implemented as a broad fluid material. The key feature is that: **(i)** it is compressible, **(ii)** it is performed by MPM explicitly like any other material, and **(iii)** stresses are computed solely from the volume change in the particle deformation gradient (J_p^n, hence J-Fluid). The model relies on an equation of state (EOS) to define its constitutive relationship, relating volume change to pressure. The Murnaghan-Tait Equation of State is the simplest to implement and fairly robust for weakly-compressible scenarios (e.g. not the bottom of the ocean). It is isothermal and assumes bulk modulus (k) is a linear function with pressure (P):

$$k = k_0 + Pk_0' \ , \tag{5.8}$$

which gives rise to a pressure relationship with volume change. Respective to an initial state, the pressure is defined as:

$$P = \frac{k_0}{k_0'}\left(\frac{V}{V_0}^{-k_0'} - 1\right) + P^0 \ , \tag{5.9}$$

where the bulk modulus (k_0) and derivative of bulk modulus with respect to pressure (k_0') for pure water (no salinity) are experimentally taken as 2.2e9 Pa and 7.1 for typical sea-level conditions. Ambient pressure (P_0) is not a concern for our near-surface simulations because we do not observe nonlinear bulk modulus behavior in a reasonable pressure range of it for coastal engineering.

In MPM simulations, the EOS on a particle is integrated over time with respect to the particle domain. For an explicit scheme with discrete time-step Δt that bridges times t^n and t^{n+1}, operating on a particle p that possesses an initial volume V_p^0, current volume V_p^n, and a deformation gradient $\mathbf{F}_p^n$, the pressure (P_p^n) is defined as:

$$P_p^n = \frac{k_0}{k_0'}\left(\left(J_p^n\right)^{-k_0'} - 1\right) \quad \text{where} \quad J_p^n = \det(\mathbf{F}_p^n), \quad V_p^n = V_p^0 J_p^n \ . \tag{5.10}$$

Note that Equation 5.10 does not use the full deformation gradient, but only its determinant (J_p^n). This saves computational time and memory usage on a particle basis. Viscous stresses are added into the model to find the total Cauchy stress for a particle p at time t^n:

$$\boldsymbol{\sigma}_p^n = \left(C_p^n + C_p^{n,T}\right)\mu_d - P_p^n \mathbf{I} \ , \tag{5.11}$$

where C_p^n is the APIC affine velocity matrix (Section 2.4, Equation 2.25), $\mathbf{I}$ is the 3 x 3 identity tensor, and μ_d is the dynamic viscosity of the fluid. For water this is 1 centipoise (0.01 poise, or $1*10^{-3}$ Pa*s) for typical conditions (e.g. sea-level, 20C). Note that viscosity (μ_d) can vary greatly due to salinity, temperature, etc. and that more advanced equations of state can be used to account for these effects. Artificial viscosity may also be incorporated during this step to reduce numerical noise in the fluid, if desired.

This weakly-compressible model for isotropic fluids is robust and easy to implement in explicit time-integration for both MPM and MLS-MPM. However, it is limited by known issues of standard, explicit MPM, namely checker-board stress fields in stiff fluids. Imcompressible fluids

or implicit time-integration may alleviate this, but we opt to instead make minor changes to MPM's advection to improve fluid behavior. This will be described in Section 5.6.7.

For basic solid materials, a Fixed-Corotated model has been selected and validated. This is a Hyper-Elastic material (i.e. constitutive relationship based on potential energy) and as such is effective for large-deformation (rigid and non-rigid) dynamics (Jiang et al. 2016). As we are approximating the elastic behavior of HDPE plastic debris, a model that allows for transport through a domain and collision with other debris/structures is needed, of which the Fixed-Corotated model fits.

The model is popular in computer graphics for its relative simplicity. It is a corotated linear elastic model with a small modification (hence "fixed") for more robust large-deformation computation (Jiang et al. 2016).

To start, deformation gradient ($\mathbf{F}$) is taken as a polar singular-value-decomposition and polar decomposition:

$$\mathbf{F} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \quad \text{and} \quad \mathbf{F} = \mathbf{R}\mathbf{S} \quad \text{where} \quad \mathbf{R} = \mathbf{U}\mathbf{V}^T, \quad \mathbf{S} = \mathbf{V}\mathbf{\Sigma}\mathbf{V}^T \ , \tag{5.12}$$

where $\mathbf{U}$, $\mathbf{\Sigma}$, and $\mathbf{V}$ are the left singular value matrix, diagonal singular value matrix, and right singular value matrix respectively. Because the deformation gradient ($\mathbf{F}$) is a real d x d matrix (where d is the dimension, 3 x 3 for 3D, 2 x 2 for 2D), $\mathbf{U}$ and $\mathbf{V}$ are real, orthonormal d x d matrices. Diagonal singular values in $\mathbf{\Sigma}$ are defined as $\sigma_i = \mathbf{\Sigma}_{ii}$. $\mathbf{R}$ and $\mathbf{S}$ are the rotation matrix and the symmetric right-stretch matrix respectively for a polar decomposition.

Singular value decomposition (SVD) and polar decomposition (Equation 5.12) are highly optimized for real 3 x 3 matrices / second-order tensors (i.e. the deformation gradient tensor in 3D), decomposing the full deformation gradient into components that describe rigid translation, rotation, and stretching. We prefer to work with this form for large deformation material models.

Energy for the Fixed-Corotated model ($\Psi(\mathbf{F})$) is defined as the energy respective to the SVD decomposition of the deformation gradient ($\hat{\Psi}(\mathbf{\Sigma}(\mathbf{F}))$) to evaluate internal energy respective to non-rigid deformation:

$$\Psi(\mathbf{F}) = \hat{\Psi}(\mathbf{\Sigma}(\mathbf{F})) = \mu \sum_{i=1}^{d} (\sigma_i - 1)^2 + \frac{\lambda}{2}(J-1)^2 \ , \tag{5.13}$$

where μ is the shear modulus, λ is the first Lame parameter, and J is the deformation gradient determinant which is also write-able as $J = \Pi_{i=1}^{d}(\sigma_i - 1)^2$.

This deviates from regular corotated models, which are less robust and may have poor performance when $\mathbf{F}$ becomes near-singular, that define energy as:

$$\Psi(\mathbf{F}) = \hat{\Psi}(\mathbf{\Sigma}(\mathbf{F})) = \mu \sum_{i=1}^{d} (\sigma_i - 1)^2 + \frac{\lambda}{2}(\sum_{i=1}^{d}(\sigma_i - 1))^2 \ . \tag{5.14}$$

Taking the derivative of the energy (Eq. 5.13) against the deformation gradient provides a stress measure. The first Piola-Kirchoff stress ($\mathbf{P}$) for the Fixed-Corotated model is:

$$\mathbf{P}(\mathbf{F}) = \frac{\partial \psi}{\partial \mathbf{F}}(\mathbf{F}) = 2\mu(\mathbf{F} - \mathbf{R}) + \lambda(J-1)J\mathbf{F}^{-T} \ , \tag{5.15}$$

which may be implemented on a MPM particle p at time t^n, using an explicit time-integration scheme with discrete time-step Δt, to form:

$$\mathbf{P}_p^n = 2\mu\left(\mathbf{F}_p^n - \mathbf{R}_p^n\right) + \lambda\left(J_p^n - 1\right)J_p^n\mathbf{F}_p^{n,-T} \ . \tag{5.16}$$

Alternatively, a Neo-Hookean material is applicable to solids. It is a nonlinear hyperelastic model for elastic material undergoing large deformation. Both the Neo-Hookean and Fixed-Corotated model are linear-elastic at small strain, by definition, but their formulation specifics produce

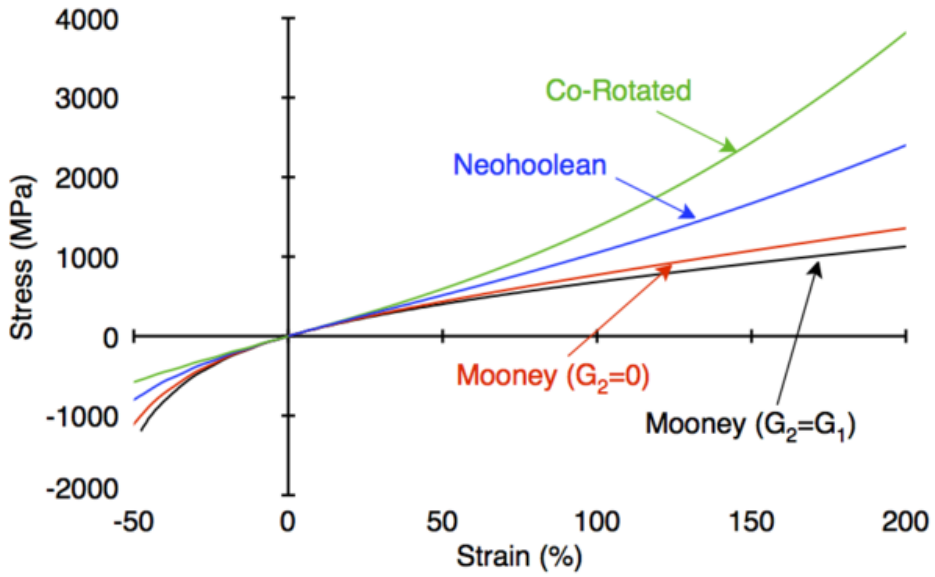

Fig. 5.1. Comparison of common large-deformation solid models. Comparing stress-strain response of Neo-Hookean, Fixed-Corotated (clamped Neo-Hookean), and Mooney models. At small deformations they are near-identical. Courtesy NairnMPM.

different results at large deformations. Figure 5.1 demonstrates the deviation in stress between different hyper-elastic models as strain grows. There is no correct material model for all instances, Fixed-Corotated vs. Neo-Hookean vs. any other model will depend on the real-world behavior one is attempting to simulate. In the case of stiff plastic undergoing large rigid motion but small non-rigid deformations, the Fixed-Corotated model is favorable.

Drucker-Prager (DP) and the Modified Non-Associative Cam-Clay (NACC) material models are also implemented in Claymore. They have not been rigorously vetted for engineering use by our group as of now, and some parameters in their formulations are still compile-time set.

5.5 *Contact Algorithms*

Contact algorithms are a vast field of study, often involving nonlinear contact mechanics and other intensive subjects. For a minimalist MPM approach to debris-fluid-structure interaction (DFSI)

we want to avoid complex schemes. For this purpose we propose the use of a novel, simplistic advection scheme (Affine-Seperable Fluid-Implicit-Particles, Section 5.6.7), for use in engineering applications (e.g. tsunami-driven debris-fields) instead of a complex, formal contact algorithm.

5.6 *Advection*

First we will describe advection schemes, old and new, that have been incorporated into MPM. These schemes control the update of a particle p in terms of velocity and position. All schemes listed here have been implemented by our lab into our ClaymoreUW Multi-GPU MPM software. At a time t^n, there is a particle velocity $\mathbf{v}_p^n$. The particle-to-grid (P2G) operation, which is respective to the method used, transfers particle velocities onto grid-node i as $\mathbf{v}_i^n$. Applying boundary conditions, external forces, etc. over a discrete time-step Δt on the grid in the Grid Update step then produces the updated grid-node i velocity $\mathbf{v}_i^*$. Note that the only difference between $\mathbf{v}_i^n$ and $\mathbf{v}_i^*$ is application of boundary conditions and external force contributions. The following time is t^{n+1} where the particle again constructs a velocity, $\mathbf{v}_p^{n+1}$. the process repeats until simulation termination. Using just these three velocity quantities give rise to numerous advection schemes.

5.6.1 Particle-In-Cell

The basic Particle-In-Cell approach (PIC, Harlow 1962) is commonly used in graphical MPM implementations due to its robustness, though avoided by many engineers as it is highly damped compared to alternatives. The method only uses the updated grid-node velocities, v_i^*, to construct particle velocities, v_p^{n+1}, for position advection, x_p^{n+1}. As PIC MPM uses the same shared grid for all materials and velocity boundaries, it serves as a filter where noise developed on a given material point(s) is canceled out when summed with the many other contributions that map to the same grid-node. This makes PIC MPM very stabile for explicit formulations applying the CFL

condition (although it is not guaranteed, especially for isolated points and constraining boundary conditions (Bai and Schroeder 2022)) and ensures MPM's no-slip contact is applied between bodies of any phase and material. However, most MPM simulations are done at a coarse grid-resolution or with a high ratio of particle-per-cell (PPC). This means sending all velocity information through the grid via the P2G operation, and retrieving all velocity information via the grid-to-particle (G2P) operation, will dampen/average out high-frequency velocity modes advected on the particles. The P2G and G2P operations for PIC are:

$$\text{PIC P2G:} \qquad m_i^n \mathbf{v}_i^n = \sum_p N_{ip} m_p \mathbf{v}_p^n \tag{5.17}$$

$$\text{PIC G2P:} \qquad \mathbf{v}_p^{n+1} = \sum_i N_{ip} \mathbf{v}_i^* \tag{5.18}$$

$$\mathbf{x}_p^{n+1} = \mathbf{x}_p^n + \Delta t \sum_i N_{ip} \mathbf{v}_i^* \ . \tag{5.19}$$

5.6.2 Fluid-Implicit-Particles

Fluid-Implicit Particles (FLIP, Brackbill and Ruppel 1986) is another particle method of renown. When implemented in MPM, it seeks to maintain high-frequency velocity modes on individual particles that would be lost in PIC due to an order of less grid-nodes than particles occupying a typical simulations. This mismatch is a dissipating information bottleneck incurred in PIC but also gives a more stable method overall, unlike FLIP's partial avoidance of the grid. Destabilizing noise in FLIP may grow as spurious velocity modes on particles are not filtered entirely through

the MPM grid. The P2G and G2P operations for FLIP are:

$$\text{FLIP P2G:} \qquad m_i^n \mathbf{v}_i^n = \sum_p N_{ip} m_p \mathbf{v}_p^n \tag{5.20}$$

$$\text{FLIP G2P:} \qquad \mathbf{v}_p^{n+1} = \sum_i N_{ip} \mathbf{v}_i^* + (\mathbf{v}_p^n - \sum_i N_{ip} \mathbf{v}_i^n) \tag{5.21}$$

$$\mathbf{x}_p^{n+1} = \mathbf{x}_p^n + \Delta t \sum_i N_{ip} \mathbf{v}_i^* \ . \tag{5.22}$$

5.6.3 PIC-FLIP Mixing

PIC and FLIP advection can be combined with a scalar mixing ratio, which has been popularized by Stomakhin et al. 2013 and Zhu and Bridson 2005. The ratio is denoted as α in this document, with bounds of $\alpha \in [0, 1]$. Values greater than 0 but less than 1 reduce noise relative to FLIP yet still garners improved energy maintenance compared to PIC. Likewise, it reduces the damping of velocity modes observed in PIC while maintaining most of its robustness. PIC-FLIP mixing has near-identical computational cost to FLIP and is simple to implement, so we generally recommend its use over its constituents. The P2G and G2P operations for PIC-FLIP mixing are:

$$\text{PIC-FLIP P2G:} \qquad m_i^n \mathbf{v}_i^n = \sum_p N_{ip} m_p \mathbf{v}_p^n \tag{5.23}$$

$$\text{PIC-FLIP G2P:} \qquad \mathbf{v}_p^{n+1} = \sum_i N_{ip} \mathbf{v}_i^* + \alpha(\mathbf{v}_p^n - \sum_i N_{ip} \mathbf{v}_i^n) \tag{5.24}$$

$$\mathbf{x}_p^{n+1} = \mathbf{x}_p^n + \Delta t \sum_i N_{ip} \mathbf{v}_i^* \ , \tag{5.25}$$

where it can be observed that setting $\alpha = 0$ reduces PIC-FLIP to PIC, and $\alpha = 1$ to FLIP. This mixing factor, α, reintroduces velocity lost by a particle due to the P2G and G2P operation, i.e. the MPM grid's filtering effect. The mixing ratio α is subjective but not especially difficult to set in our applications. However, it may produces unstable results if set too high so it is best to be

conservative. For compressible materials we typically use a value of 0 to 0.5 (i.e. PIC or a half-and-half PIC-FLIP mix). However, when damping is dominating our results (e.g. for stiff materials), we often use values ranging from 0.5 to 0.9999. Major instabilities in PIC-FLIP simulations are visually apparent (e.g. the scene goes unstable in an unrealistic, explosive fashion), as shared by pure FLIP. Minor instabilities may only appear briefly in a small region of a body or on isolated particles (Bai and Schroeder 2022). To remove minor instabilities, we tend to find the largest α that does not produce major instability and then reduce its last digit by 50-100%. Lower CFL numbers often improve stability in high α simulations, e.g. CFL = 0.3 instead of 0.5, as greater magnitudes velocity noise can appear without violating the CFL condition (which halts the simulation). However, using more time-steps may contribute to compounding of errors from other sources so sub CFL = 0.05 - 0.10 is not recommended.

5.6.4 Affine Particle-In-Cell

One of the most substantial improvements to MPM produced by the computer graphics / applied mathematics research community is the Affine Particle-In-Cell method (APIC, Jiang et al. 2015 and Jiang et al.). APIC has stability similar to PIC while conserving energy in velocity modes inline with FLIP– without the majority of FLIP's spurious noise. Further, APIC's formulation directly considers affine velocity, i.e. angular velocity and momentum are incorporated into G2P and P2G for a particle, and thus improves the conservation of energy for large rotations compared to PIC, FLIP, and PIC-FLIP. While APIC has been covered in more detail in Section 2.4, the basic

formulas for its G2P and P2G operations are:

$$\text{APIC P2G:} \qquad m_i^n \mathbf{v}_i^n = \sum_p N_{ip} m_p \left[\mathbf{v}_p^n + \mathbf{C}_p^n \left(\mathbf{x}_i^n - \mathbf{x}_p^n \right) \right] \tag{5.26}$$

$$\text{APIC G2P:} \qquad \mathbf{v}_p^{n+1} = \sum_i N_{ip} \mathbf{v}_i^* \tag{5.27}$$

$$\mathbf{x}_p^{n+1} = \mathbf{x}_p^n + \Delta t \sum_i N_{ip} \mathbf{v}_i^* \tag{5.28}$$

$$\mathbf{C}_p^{n+1} = \sum_i N_{ip} \mathbf{v}_i^* \left(\mathbf{x}_i^n - \mathbf{x}_p^n \right)^T (\mathbf{D}_p^n)^{-1}$$

Although some graphical practitioners of MPM use PIC advection, it can dampen out energy substantially and make contact between objects overly "sticky". Implementing FLIP advection, fairly common in engineering and graphics until recently, reintroduces velocity modes through preservation on particles when the grid would filter them out if using PIC. However, FLIP may preserve spurious velocity modes on these particles (i.e. null-space error, Hammerquist and Nairn 2017 and Nairn and Hammerquist 2021). FLIP-PIC mixing, as seen in Zhu and Bridson 2005, is a middle-ground that garners benefits of both PIC and FLIP, but its stability will depend on a subjectively chosen mixing factor, $\alpha \in [0, 1]$, and its lack of an affine velocity matrix (along with truncation further high-order components) will continue to filter out velocity modes. APIC takes an alternative path by augmenting a PIC style scheme with more transferred information regarding the velocity field to avoid a good amount of damping and preserve PIC's stability (almost always observed for CFL ≤ 0.5, but note that neither PIC nor APIC are proven to remain stability (Bai and Schroeder 2022)). APIC also improves PIC, FLIP, and PIC-FLIP MPM's rotational behavior significantly, giving an efficient means to run a much larger array of simulation scenarios.

5.6.5 Polynomial Particle-in-Cell

We note that APIC has been extrapolated to the Polynomial Particle-in-Cell (PolyPIC, Fu et al. 2017) method, which augments the G2P and P2G projection operations with local, mass-orthogonal polynomial basis as opposed to the simple affine set chosen for APIC. In other words, PolyPIC stores additional coefficients per particle to capture the velocity field more accurately (i.e. preserves velocity modes better) on a selected number of polynomials. This has interesting implications for complicated, turbulent flow simulations which are rarely studied in MPM. Further, it may pair well with G2P2G kernels as the added coefficients need only be temporarily held– not saved on particles. In the next section, we will discuss two additional methods which offer similar capabilities for MPM as PolyPIC, but derived and interpreted with greater mathematical rigor in their respective papers.

5.6.6 Extend Particle-in-Cell and Full-Mass Matrix Material Point Method

Insofar we have described fairly simple advection schemes (PIC, FLIP, PIC-FLIP, and APIC) and their respective benefits in MPM. Briefly, the PolyPIC method (Section 5.6.5) was discussed as a means to higher-order behavior, but its formulation seems to dance around the idea of a full mass matrix MPM without committing, although this may be our misinterpretation.

More advanced and rigorous approaches have been developed and may be necessary for MPM applications in engineering and physics. Despite not pursuing in our research, we would have enjoyed the opportunity to implement both the Extended Particle-in-Cell (XPIC, Hammerquist and Nairn 2017) scheme and the Full Mass Matrix Material Point Method (FMPM, Nairn and Hammerquist 2021) if time had allowed. Both XPIC(k) and FMPM(k) provide superior reduction of noise and minimization of dissipation compared to APIC, FLIP-PIC, FLIP, and PIC when implemented at high-orders.

In effect, an implementation of XPIC(k) projects out null-space errors from MPM velocities as k

grows large. For k=1 XPIC is PIC, for k=2 to 3 XPIC is on similar ground to FLIP and APIC, and for k=4 to 8+ (common choices) XPIC is often superior in accuracy to all aforementioned advection methods, although some dissipation may occur. FMPM(k) for the same k used in an XPIC(k) implementation is less dissipating. This is because it considers the full mass matrix at order k, whereas XPIC just uses an asymptotic expansion of the inverse mass matrix for grid-calculations (Nairn and Hammerquist 2021). Note that standard MPM uses the lumped mass matrix, a useful simplification for high-performance simulations, but a non-ideal choice for high-accuracy. Both XPIC and FMPM aim to remove MPM's P2G and G2P projection of null-space errors at the added computational cost of growing k values to converge towards the ideal of a full mass matrix. Neither XPIC nor FMPM are favored for high-performance implementations, though they can be computationally competitive in some ranges, e.g. $k \in [2, 8]$. XPIC and FMPM are certainly high-accuracy methods developed with rigor and they provide fine-tuned control over the performance-accuracy trade-off inherent to adjustable order of approximation of the full mass matrix in MPM.

5.6.7 Affine-Separable Fluid-Implicit Particles

The MPM advection methods described in the previous sections (e.g. PIC, FLIP, PIC-FLIP, and APIC) still resolve object contact as sticky and feature a gap of roughly one grid-spacing, Δx, at all points of collision. Seemingly minor, this results in poor contact shearing of low-friction objects and can substantially impact basic phenomena (e.g. buoyancy and drag forces, Chp. 6). An advection scheme that is similar to APIC but can selectively break sticky contact and reduce the contact gap is desirable.

To alleviate this problem, Affine-Seperable Fluid-Implicit Particles (ASFLIP) has been implemented by our lab as an advection scheme in the open-source Multi-GPU MLS-MPM code Claymore (Chapter 4). It serves as a contact algorithm to improve fluid-debris-structure

interaction (DFSI), specifically contact between phases and materials.

ASFLIP is a computer graphics method (Fei et al. 2021a) associated with Tencent, a giant in many fields (e.g. video-games), developed to improve contact between objects in MPM and to reduce MPM's damping similar to APIC. In essence, it adds an affine and separable component into FLIP-PIC advection. Recall that FLIP-PIC, and now ASFLIP, use the subjective mixing ratio $\alpha \in [0, 1]$, as described in Section 5.6.3. ASFLIP's affine portion is identical to the Affine Particle-In-Cell (APIC, Section 2.4, Jiang et al. 2015) method's use of affine velocity matrices for G2P and P2G augmentation. ASFLIP's novelty is is inclusion of a separable factor, $\beta \in [0, 1]$. It allows PIC-FLIP's partially-filtered velocity advection update on a particle to affect position advection (within the same time-step) proportional to $\beta\alpha$. The throttled position correction factor $\beta_p\alpha$ for a particle p is controlled by identification of assumed spurious tensile modes on particles.

To understand ASFLIP's utility it is best to visualize an MPM simulation step at two distinct phases of the computation. The first is at a grid-node, where the equation of motion (EOM) is solved. The second is at a particle (i.e. point), where constitutive rules (i.e. material models) are executed. The act of transferring all particle results within a grid-node "support" (i.e. local domain, e.g. 3x3 grid spacing for quadratic B-Spline transfer functions) to said grid-node localizes the EOM relative to particles. The act of transferring all grid-node results within a particles "support" is a means to localize constitutive computation relative to grid-nodes. In essence, grid-nodes and particles act as filters for each others computations. Usually the grid-node is the primary filter, as it may feature an order of magnitude less computational points than the respective particles in its support.

This filtering leads to **(i)** damping and **(ii)** spatially restricted computations. The former lessens significance of energetic simulations (i.e. violent free-surface wave breaking) and the latter sub-grid cell behavior (i.e. refined multi-material contact). PIC advection possesses this quality. FLIP and FLIP-PIC mixing reintroduce energy but risk noise. APIC is an overall improvement on

PIC but is still damped and handles contact at a grid-cell spacing. XPIC and FMPM are intriguing solutions, but at a noted computational and implementation cost. Refined multi-material, multi-phase interaction at sub-grid cell resolutions necessitates a closer look at numerical advection. Namely, FLIP-PIC mixes only allow the velocity update during advection to be influenced by unfiltered particle velocity– The position update at advection uses only the filtered quantity. Sticky contact between materials at the grid-cell resolution can be taken as a result of updating particle positions with spatially filtered velocity, even if the advected particle velocities are updated with partially unfiltered FLIP values. The consequences of this is a "positional trap" that occurs at roughly a grid-cell width, disallowing adjacent particles to move away from one another unless acted on by a very strong tensile force.

To break the "positional trap" of MPM, unfiltered Lagrangian velocity can advect particle positions. This risks exposure to numerical noise (as particles can carry spurious velocity modes) and so it is throttled by a scalar mixing factor ($\alpha \in [0, 1]$) from PIC-FLIP mixing. This method is sometimes described as Natural Fluid Implicit Particles (NFLIP, Stomakhin et al. 2013; Fei et al. 2021a).

For stability, it is advised to introduce lagrangian velocity into particle velocity advection and not into particle position advection. A throttling factor (β_p) can be further added to weight Lagrangian position advection. It is determined on a material, tension vs. compression, and boundary contact basis to limit spuriously energetic behavior at critical portions of the simulation. This is Separable Fluid-Implicit-Particles (SFLIP).

Introducing affine velocity from APIC (Sec. 2.4) into SFLIP improves energy conservation (especially in rotation). This is Affine-Separable Fluid-Implicit Particles (ASFLIP) (Fei et al. 2021a), a surprisingly simple yet powerful advection method that reduces null-space error of PIC, is more stable / controllable than FLIP, includes rotational benefits of APIC, and introduces sub-grid correction for spurious position locking of particles.

Starting from APIC with PIC-FLIP mixing, a common formulation, there is nearly no computational overhead in B-Spline MPM or MLS-MPM. The P2G and G2P operations for ASFLIP are:

$$
\begin{aligned}
\text{ASFLIP P2G:} \qquad & m_i^n \mathbf{v}_i^n = \sum_p N_{ip} m_p \left[\mathbf{v}_p^n + \mathbf{C}_p^n \left(\mathbf{x}_i^n - \mathbf{x}_p^n \right) \right] \\
\text{ASFLIP G2P:} \qquad & \mathbf{v}_p^{n+1} = \sum_i N_{ip} \mathbf{v}_i^* + \alpha \left(\mathbf{v}_p^n - \sum_i N_{ip} \mathbf{v}_i^n \right) \\
& \mathbf{x}_p^{n+1} = \mathbf{x}_p^n + \Delta t \left[\sum_i N_{ip} \mathbf{v}_i^* + \beta_p^{n+1} \alpha \left(\mathbf{v}_p^n - \sum_i N_{ip} \mathbf{v}_i^n \right) \right] \\
& \mathbf{C}_p^{n+1} = \sum_i N_{ip} \mathbf{v}_i^* \left(\mathbf{x}_i^n - \mathbf{x}_p^n \right)^T \left(\mathbf{D}_p^n \right)^{-1} ,
\end{aligned} \tag{5.33}
$$

where β_p is set to a value $\beta_{\min}$ or $\beta_{\max}$ if the particle deformation gradient determinant (J_p) is less than or greater than a characteristic value (J_c) respectively.

J_c is material dependent and effectively represents reaching a yield surface in a constitutive model by purely volumetric means. For an isotropic weakly-compressible fluid J_c is set to 1 in some implementations to guarantee that fluid particles in tension due to sticky contact have the opportunity to break their positional trap (i.e. fluid particles don't stick to debris).

When a particle is moving into a boundary, correction ratio β_p is set to zero to disallow spurious advection through the boundary. The same adjustment is made when near particle inflows and outflows. However, often we have not needed these conditions as we use small values of $\beta_{\min}$ (e.g. 0 to 0.01) and $\beta_{\max}$ (e.g. 0 to 0.25). Small factors with reasonably small time-steps and a precise floating-point code design tended to be stable for our studies. However, this may not hold for all domains, e.g. high-velocity ballistics which exhibits extreme jumps in tensile stress.

A limitation in the original ASFLIP formulation is the binary switch between a minimum and maximum position correction value. We investigated position correction as a function $\beta_p = \mathcal{B}(J_p, J_c, \delta_c)$, where δ_c is the range over which correction develops (e.g. 0.01 for 1% volume

tensile growth), when $J_c < J_p < J_c + \delta_c$. If $J_p < J_c$, then $\beta_p = 0$. If $J_p = J_c$, then $\beta_p = \beta_{\min}$. If $J_p \geq J_c + \delta_c$, then $\beta_p = \beta_{\max}$.

Many functions could be tested, e.g. a sharp shock function replicates ASFLIP binary behavior, but the best may be application specific. For instance, order-b polynomials of the form:

$$\beta_p = \beta_{\max} \max(\left((J_p - J_c)/\delta_c\right)^b, 1) + \beta_{\min} \quad \forall \quad J_p \geq J_c,\ \delta_c > 0,\ b \geq 2\ , \tag{5.34}$$

have worked well for gentle adjustment of slight exceeding over J_c with an increasingly strong correction near the upper-bound of $J_c + \delta_c$. This is logical for a weakly defined discontinuity, e.g. discontinuities which tend to feature some cohesive-like attraction. For instance, nearby debris in water may tend to move as a group until the cohesive-like attraction (i.e. surface tension and any shared driving-flow) is exceeded by a significant stress, e.g. impact or splitting flow. For small disturbances in flow they may jostle lightly but continue as a group with minimal, transient motion at discontinuities.

We applied ASFLIP to our coupled MPM-FEM (see Section 5.13). Only element vertices on the surface of the 3D mesh were allowed non-zero position correction ratios, β_p, a condition set at initialization. This was to prevent spurious advection within the volumetric body, as we are only interested in contact near the surface which is conveniently defined by the tetrahedron meshes. This improved contact distances that were previously mediated by non-ideal gaps of two or so grid-cells. Error in advection was thus primarily constricted to surface waves on debris, which aren't important for our study of simple impacts, but notable error was not observed. A benefit of MPM with ASFLIP when applied to hyperelastic finite elements is that any erroneous deformation modes will be penalized by the element. This is because they trend towards undeformed states unless persistently counter-acted. This behavior allowed for the compelling non-sticky behavior of sub-grid hair strand interaction, with hair being Lagrangian beam elements coupled with MPM, in

the original ASFLIP paper by Fei et al. 2021a, as each strand trended towards a resting state (i.e. straight, curly) irregardless of small transient errors that would have produced unrealistic, sticky, tangled hair in standard MPM.

ASFLIP was developed by the computer graphics research community in (Fei et al. 2021a) and, in some ways, bypasses the limits of previous engineering methods in the simplicity of implementation and use. It is not yet clear if it bypasses first-principals entirely, but we note that there are somewhat similar methodologies that have been used in the past. For instance, NairnMPM allows for a PIC-FLIP mixing style that can adjust positions along with velocities in a way similar to ASFLIP. However, their usage is not based on the flexible tension contact throttling but a periodic application every set-number of time-steps in order to avoid introduction of undesired frequencies.

While mass is still conserved in this method we wouldn't recommend ASFLIP for studies requiring near-perfect thermo-mechanical conservation as the already tenuous convergence properties of MPM and its many low-order variants are certainly not made more predictable with ASFLIP. Further study is needed to accept or reject the method for applications sensitive to energy conservation of high-frequency modes, but for now it opens the doors for energetic, multi-material, multi-phase simulations at sub-grid cell contact fidelity. Further, it requires unusually small implementation effort and computational resources.

5.6.8 Remarks on Advection in MPM

In Sections 5.6.1, 5.6.2, 5.6.3, 5.6.4, 5.6.5, 5.6.6, and 5.6.7 we discussed the many competing approaches for advection in the Material Point Method. Overall, we recommend ASFLIP for high-performance simulations that are not highly sensitive to energy errors as it removes a large proportion of the damping and noise seen in traditional MPM usage (e.g. PIC, FLIP, and PIC-FLIP) and also provides what is, in effect, a basic but tunable contact algorithm to alleviate

MPM's no-slip contact. However, it is advised to reduce ASFLIP to AFLIP (APIC combined with PIC-FLIP mixing, occurs when $\beta = 0$) unless this contact modification is neccesary.

Although some graphical practitioners of MPM use PIC advection it can dampen out energy substantially and makes contact between objects overly "sticky". Implementing FLIP advection, fairly common in engineering and graphics until recently, reintroduces velocity modes through preservation on particles when the grid would filter them out if using PIC. However, FLIP may preserve spurious velocity modes on these particles (i.e. null-space error, Hammerquist and Nairn 2017 and Nairn and Hammerquist 2021). PIC-FLIP mixing, as seen in Zhu and Bridson 2005, is a middle-ground that garners benefits of both PIC and FLIP, but its stability will depend on a subjectively chosen mixing factor, $\alpha \in [0, 1]$, and its lack of an affine velocity matrix (along with truncation further high-order components) will continue to filter out velocity modes. APIC takes an alternative path by augmenting a PIC style scheme with more transferred information regarding the velocity field to avoid a good amount of damping and preserve PIC's stability (almost always observed for CFL ≤ 0.5, but note that neither PIC nor APIC are proven to remain stability (Bai and Schroeder 2022)). APIC also improves PIC, FLIP, and PIC-FLIP MPM's rotational behavior significantly, giving an efficient means to run a much larger array of simulation scenarios. We give an honorable mention to PolyPIC, as it offers a path forward for extrapolating APIC to higher-order velocity field augmentation. For more stringent requirements on energy conservation and preservation of velocity modes, we direct readers to evaluate if the high-order instances of XPIC and FMPM are feasible in their computational frameworks, as they remove null-space error in P2G and G2P by approximating (with high computational cost) a full mass matrix.

Before considering higher-order MPM advection and mass matrix formulations, it may be far easier and effective to test the usage of a antilocking strategy. In the next sections we describe a very simple way to vastly improve the volumetric strain and stress behavior of MPM and how to implement it with our own linear mixing modification for stability, in MLS-MPM for

compatibility, and within a fused G2P2G kernel for applicability in high-performance parallel codes.

5.7 *F-Bar Volumetric Antilocking for B-Spline MPM*

In 8.4 it was indicated that regular MPM suffers from volumetric locking effects, even after resolving the infamous check-board pressure errors, when simulating nearly-incompressible materials. To overcome this we have considered several antilocking strategies. However, to maintain the high-performance Multi-GPU viability of our approach one strategy was identified as suitable– the F-Bar volumetric antilocking algorithm by Zhao et al. 2023 for B-Spline MPM.

In short, an assumed deformation approach is taken, commonly seen in finite elements literature (Neto et al. 2005), where an effectively reduced integration determines the volumetric deformation's contribution to a particle's stress update to alleviate constraints from degrees of freedom. This manifests as having particles aggregate their expected volume change ratio, i.e. the Jacobian J_p, on grid-nodes in their B-Spline range with a volume weight. On the grid, contributions are unweighted by dividing out the total volume of contributing particles per node. Particles then read the grid filtered Jacobian scalar field as the assumed Jacobian, "J-Bar" $\bar{J}_p$, and use it to adjust their deformation gradients $\mathbf{F}$ and relative deformation gradient $\Delta\mathbf{F}_p$ into the assumed "F-Bar" forms of $\bar{\mathbf{F}}_p$ and $\Delta\bar{\mathbf{F}}_p$. This essentially scales the volumetric portions of each particles deformation measures to alleviate the oscillating volume change / pressure, that is characteristic of a locked MPM, within a B-Spline support, with the exact means showed later in this section, of their deformation gradient for use in their material law's stress computation. This is a very effective and efficient way to remove the poor volumetric behavior of B-Spline MPM in nearly-incompressible simulations and improve the resolved pressure field notably.

Reduced integration methods common in various schemes including F-Bar projection methods, notably in the geotechnical literature for stable multi-phase schemes with pore pressure projection

Zheng et al. 2021, are often used to alleviate the inf-sup condition which is a necessary, but not sufficient (consistency and ellipticity are also requisite), condition for nearly-incompressible simulations. In other words, spurious pressure modes will appear if the appropriate balance between the number of degrees of freedom for displacement/velocity, number of DOF for pressure, and the order of the complete interpolating polynomial space (1st order in MLS-MPM, where the 0th order is constant and 1st is affine) is not met. Simplified, MPM represents velocity/displacement with typically an order of magnitude less degrees of freedom on the grid than it represents the Jacobian/pressure on particles because particles-per-cells is usually high. Typically, PPC = $N^{\text{Dim.}}$ where N is the number of particles in the linear direction of a grid-cell length (Δx) and is usually set as 2, 3, or 4 to produce PPC of 8, 27, or 64.

If, say, 64 particles occupy each cell and we consider quadratic B-Splines on a grid that extends infinitely in 3D, it is apparent that the deformation gradient Jacobian, J_p, on each particle, p, must be determined at an inappropriate information asymmetry for enforcement of nearly-incompressible behavior via momentum and internal forces contributed to the far fewer grid-nodes by far-more particles. In this typical case, the grid will simply exhibit volumetric locking that prevents meaningful deforming flow of nearly-incompressible materials, as needed for stiff fluids like water, and the particles will have their true pressure mode dominated by spurious pressure modes which may appear as standing waves of oscillatory high and low pressure which are non-logical for a true solution of a simulated problem.

Analytical guarantee of the inf-sup condition can be found in some special cases but this is often an unwieldy task. Tests of numerical convergence toward the ideal behavior is commonly taken in its place (Bathe 2001), e.g. a patch test. For MPM and its variants (e.g. mixed v-p formulation as presented in Chen et al. 2018), which have large motion/deformation of particles throughout the static grid, the underlying formulation convergence properties are understudied compared to the FEA literature. Though it is possible to effectively project particles into ideal quadrature positions

from a mathematical stance to put MPM into a higher-order regime, but this is not trivial and will have a computational cost that raises the question of why advanced FEA is not being used instead. Most MPM practitioners will start MPM particles at either Newton-Cotes or Gaussian locations with respect to the grid, and then allow particles to advect through space and time while deviating from (or back into) ideal quadrature positions. This leads to an MPM fabric of inconsistently positioned particles in time and space and thus unpredictably varying inf-sup behavior throughout the domain. This isn't desired, but a fairly simple way to alleviate, but not guarantee, the inf-sup condition is reduced integration of particle Jacobians/pressures. This adds a grid-averaging projection of the Jacobian to cell-centers (i.e. a staggered grid), which is then reconstructed on particles. However, this has cell-crossing error concerns among other issues.

Instead, the work of Zhao et al. 2023 takes a pragmatic approach of continuing to use the same B-Spline interpolating function to grid-average the Jacobians, now within the stencil support instead of a specific cell. In effect, this filters the Jacobians/pressures through grid degrees of freedom on top of the prior filtering of velocity on the grid (used to construct the Jacobian originally) and thereby tends to alleviate the disparity in DOF that produces spurious pressure modes. Obviously, the mathematical guarantees of this approach are non-rigorous, especially for unusual particle distributions along the Cartesian grid, but it is observed to work surprisingly well in our studies of natural hazards, structural impacts, as well as multi-material and multi-phase validation benchmarks (see Section 5.12.5). Further, in the context of high-performance Multi-GPU MPM at massively parallel computational scales, it will likely outperform the more complicated approaches for nearly-incompressible flow in terms of error convergence as these methods are typically not in high-performance implementations due to complexity and/or are not as well-suited to acceleration on modern parallel hardware.

Before we formulate the B-Spline F-Bar method for MPM, we recommend reviewing the basics of MPM's deformation gradient $\mathbf{F}$, Jacobian J, and other standard steps presented in Sections 8.4.1

and 2.5. To begin, the assumed deformation gradient $\bar{\mathbf{F}}_p$ is stated relative to the grid-averaged deformation gradient determinant $\bar{J}_p$ reconstructed on a particle p with respect to the standard deformation gradient $\mathbf{F}_p$ and its determinant $J + p$ as:

$$\bar{\mathbf{F}}_p = \left(\frac{\bar{J}_p}{J_p}\right)^{1/\text{Dim.}} \mathbf{F}_p \qquad \text{where Dim. is the number of dimensions ,} \tag{5.35}$$

which shows the relationship between the assumed and standard deformation gradient is based solely on the real part of the reciprocal root (i.e. reciprocal cube or square root in 3D and 2D respectively) of $\bar{J}_p$ over J_p. This is considerate of numerical dimensions, i.e. 3D simulations use Dim = 3. Grid-averaging of the Jacobian is effectively a particle-to-grid (P2G) operation that applies a volume weight to a particles Jacobian contribution. Once all particles have summed contributions to grid-nodes, and said summation has synchronized if in a parallel code, the grid updates to remove weighting from the Jacobians, stated by:

$$\bar{J}_i = \sum_p N_{ip} J_p \frac{V_p}{V_i} \ , \qquad V_i := \sum_p N_{ip} V_p \ , \tag{5.36}$$

The grid-averaged Jacobian, $\bar{J}_i$, on a particle p, i.e. $\bar{J}_p$, is reconstructed relative to all grid-nodes within the stencil of its B-Spline function, i. For quadratic B-Splines in 3D there are 27 grid-nodes within the stencil support. By determining the shape-function weight relative to the particle and a grid-node, N_{ip}, we may multiply against said grid-nodes Jacobian, $\bar{J}_i$, which in summation over all grid-nodes within the stencil produces the reconstructed particle Jacobian as:

$$\bar{J}_p = \sum_i N_{ip} \bar{J}_i \ , \tag{5.37}$$

which has removed the bulk of volumetric locking common to MPM with unusual ease. This is no different than projecting and reconstructing any scalar from the grid onto a particle in MPM,

though the weight is particle volume instead of mass. A key observation is that no new shape-function was introduced, as other methods tend to use a lower-order shape-function such as linear or quadratic for quadratic and cubic primary shape-functions respectively. The saved implementation effort makes it more likely to actually be used broadly (many very good ideas in papers aren't applied if they can't be understood and implemented within an afternoon), as well as the computational savings from applying the shape-function weights that are already calculated in the standard MLS-MPM algorithm.

Insofar we have discussed the assumed Jacobian. It is trivial to compute, and useful for simplistic isotropic constitutive laws, but more advanced material will require the assumed deformation gradient $\bar{\mathbf{F}}$ and assumed relative deformation gradient $\Delta\bar{\mathbf{F}}$. To acquire said variables, it is useful to view the assumed Jacobian on a particle as a projection operator, $\Pi(\cdot)$, on the particle Jacobian as:

$$\Pi\left(J_p\right) := \bar{J}_p = \sum_i N_{ip}\bar{J}_i \ , \tag{5.38}$$

where the grid-averaged assumed Jacobian, $\bar{J}_i$, is of course determined only after the MPM grid update is performed to remove volume weighting done during P2G with Equations 5.36 and synchronization occurs. Note that we have only considered this formulation without the influence of discrete time integration needed for numerical application of MPM. For a time $t = n$, take a discrete time-step Δt that advances time as $t + \Delta t = n + 1$. Over this interval, we define ΔJ to be the relative Jacobian and $\Delta\mathbf{F}$ as the relative deformation gradient. For a particle p, three useful definitions relating the various Jacobian values to deformation gradients are:

$$\Delta J_p := \det\left(\Delta\mathbf{F}_p\right), \qquad \bar{J}_p^n := \det\left(\bar{\mathbf{F}}_p^n\right), \qquad J_p^n := \det\left(\mathbf{F}_p^n\right) \ , \tag{5.39}$$

where $\Delta\bar{F}_p$ is the assumed relative deformation gradient over Δt, which is defined as:

$$\Delta\bar{\mathbf{F}}_p := \bar{\mathbf{F}}_p^{n+1}\left(\bar{\mathbf{F}}_p^{n+1}\right)^{-1} \quad . \tag{5.40}$$

We also define the discrete time representation of the assumed Jacobian at time $t + \Delta t = n + 1$ using Equations 5.37 and 5.38 as:

$$\bar{J}_p^{n+1} = \sum_i N_{ip}\bar{J}_i \tag{5.41}$$

$$= \Pi\left(\bar{J}_p^n \Delta J_p\right) \ , \tag{5.42}$$

which shows that the assumed Jacobian on a particle at a given time, $t + \Delta t = n + 1$, is simply the assumed Jacobian at the prior time-step, $t = n$, which was projected during its update (i.e. multiplication by ΔJ_p) in a prior P2G step. The projection operator $\Pi(\cdot)$ starts the grid-averaging, is finished when the grid updates to remove volumetric weights, and then is reconstructed on the particle in G2P during Equation 5.42.

Note that a global synchronization of devices and computational cores must occur for parallel implementations to guarantee the MPM grid update step has removed a grid-node's assumed Jacobian's volumetric weighting, as seen in Equation 5.36, *after* all relevant particles have contributed to said grid-node. Failure to do so will cause a data-race condition, which is a source of error with unpredictable behavior.

Equation 5.40 is an appropriate but not yet useful definition. Notably, the value of $\Delta\bar{\mathbf{F}}_p$ is desired at time $t + \Delta t = n + 1$ but appears to depend on the acquisition of the assumed deformation gradient of the same time-step. This is a problem because while it may be possible to find the deformation gradient at this time, $\mathbf{F}_p^{n+1}$, simply by using the updated velocity and velocity gradient of the particle (see Sections 5.6 and 8.4.1) we can not do the same for the corresponding assumed

deformation gradient $\bar{\mathbf{F}}_p^{n+1}$. This is because we removed the direct determination of deformation via velocity parameters to construct the assumed deformation method that is referred to as F-Bar antilocking.

For context, this was done as a means to prevent over-reliance on velocity degrees of freedoms for the typically greater DOF's of volumetric deformation (i.e. Jacobian) and volumetric stress (i.e. pressure). F-Bar achieves this by effective reduction of DOF with the information-reducing averaging operation on grid-nodes shared by many particles, which is good because it is less likely that spurious (and often high frequency and amplitude) pressure modes are preserved following said information loss. This tends to damp out the erroneous modes and primarily preserves real modes, although with potential for dissipation.

To acquire the assumed deformation gradient at the latter time-step, $\bar{\mathbf{F}}_p^{n+1}$, we must determine a way to compute it using only the grid-averaging projection of Equation 5.38. We may start by

using simple algebraic manipulation of Equation 5.35 via Equations 5.39 to show:

$$\bar{\mathbf{F}}_p^{n+1} = \left(\frac{\bar{J}_p^{n+1}}{J_p^{n+1}}\right)^{1/\text{Dim.}} \mathbf{F}_p^{n+1} \tag{5.43}$$

$$= \left(\frac{\bar{J}_p^{n+1}}{J_p^n \Delta J_p}\right)^{1/\text{Dim.}} \mathbf{F}_p^{n+1} \tag{5.44}$$

$$= \left(\frac{\bar{J}_p^n \bar{J}_p^{n+1}}{\bar{J}_p^n J_p^n \Delta J_p}\right)^{1/\text{Dim.}} \mathbf{F}_p^{n+1} \tag{5.45}$$

$$= \left(\frac{\bar{J}_p^{n+1}}{\bar{J}_p^n \Delta J_p}\right)^{1/\text{Dim.}} \Delta\mathbf{F}_p \mathbf{F}_p^n \left(\frac{\bar{J}_p^n}{J_p^n}\right)^{1/\text{Dim.}} \tag{5.46}$$

$$= \left(\frac{\bar{J}_p^{n+1}}{\bar{J}_p^n \Delta J_p}\right)^{1/\text{Dim.}} \Delta\mathbf{F}_p \bar{\mathbf{F}}_p^n \tag{5.47}$$

$$= \left(\frac{\Pi\left(\bar{J}_p^n \Delta J_p\right)}{\bar{J}_p^n \Delta J_p}\right)^{1/\text{Dim.}} \Delta\mathbf{F}_p \bar{\mathbf{F}}_p^n \tag{5.48}$$

$$= \left(\frac{\Pi\left(\bar{J}_p^n \Delta J_p\right)}{\bar{J}_p^n \Delta J_p}\right)^{1/\text{Dim.}} \Delta\mathbf{F}_p \left(\frac{\bar{J}_p^n}{J_p^n}\right)^{1/\text{Dim.}} \mathbf{F}_p^n \ , \tag{5.49}$$

which relies only on saving the assumed deformation gradient Jacobian at time $t = n$ (with the $\bar{\mathbf{F}}_p$ at time $t = n$ found through scalar multiplication in Equation 5.35 and a grid-averaging projection, $\Pi(\cdot)$ (see Equation 5.38), which began at time $t = n$ and completes via reconstruction on the particle at time $t + \Delta t = n + 1$.

This form may already be applicable for hyper-elastic materials that solely use the full deformation gradient. The additional scalar assumed Jacobian value saved per particle is only 4 to 8 bytes. Overall, the performance hit is in the range of 10 to 20% with substantial benefits to pressure resolution.

The relative change of the assumed deformation gradient, $\Delta\bar{\mathbf{F}}_p$, on a particle p over the discrete

time-step Δt bridging times $t = n$ and $t + \Delta t = n + 1$ is found by substituting Equation 5.42 and Equation 5.49 into the original definition of $\Delta\bar{\mathbf{F}}_p$ in Equation 5.40. The desired, compute-ready form is found to be:

$$\Delta\bar{\mathbf{F}}_p = \left(\frac{\bar{J}_p^{n+1}}{\bar{J}_p^n \Delta J_p}\right)^{1/\text{Dim.}} \Delta\mathbf{F}_p \bar{\mathbf{F}}_p^n \left(\bar{\mathbf{F}}_p^n\right)^{-1} \tag{5.50}$$

$$= \left(\frac{\bar{J}_p^{n+1}}{\bar{J}_p^n \Delta J_p}\right)^{1/\text{Dim.}} \Delta\mathbf{F}_p \tag{5.51}$$

$$= \left(\frac{\Pi(\bar{J}_p^n \Delta J_p)}{\bar{J}_p^n \Delta J_p}\right)^{1/\text{Dim.}} \Delta\mathbf{F}_p \ . \tag{5.52}$$

Equation 5.52 is needed for input into material models that consider relative change of the deformation gradient, which we replace with the assumed relative change of the deformation gradient to alleviate volumetric locking. Of course, most material models aren't phrased in terms of deformation gradients or their relative change, so this value may be transmuted to determine some other measure of strain or deformation following standard rules of large-deformation mechanics.

If we need the deformation gradient as input as well, we can construct it using Equation 5.49 and 5.52 in the form exposing dependency only on the prior deformation gradient $\bar{\mathbf{F}}_p^n$, prior assumed Jacobian $\bar{J}_p^n$, the relative change of Jacobian ΔJ_p, and the completion of a projection of the assumed Jacobian update to the grid, grid-update, and subsequent reconstruction on the particle. This is stated as:

$$\bar{\mathbf{F}}_p^{n+1} \quad = \quad \Delta\bar{\mathbf{F}}_p \bar{\mathbf{F}}_p^n \quad = \quad \left[\left(\frac{\Pi\left(\bar{J}_p^n \Delta J_p\right)}{\bar{J}_p^n \Delta J_p}\right)^{1/\text{Dim.}} \Delta\mathbf{F}\right] \cdot \left[\left(\frac{\bar{J}_p^n}{J_p^n}\right)^{1/\text{Dim.}} \mathbf{F}_p^n\right] \ . \tag{5.53}$$

In summary, the necessary variables for this B-Spline F-Bar volumetric antilocking approach are

computable for time $t + \Delta t = n + 1$ using

$$\bar{J}_p^{n+1} = \Pi\left(\bar{J}_p^n \Delta J_p\right) , \quad \Delta\bar{J}_p = \frac{\bar{J}_p^{n+1}}{\bar{J}_p^n} , \tag{5.54}$$

and

$$\bar{\mathbf{F}}_p^{n+1} = \left[\left(\frac{\Pi\left(\bar{J}_p^n \Delta J_p\right)}{\bar{J}_p^n \Delta J_p}\right)^{\frac{1}{\text{Dim.}}} \Delta\mathbf{F}\right] \cdot \left[\left(\frac{\bar{J}_p^n}{J_p^n}\right)^{\frac{1}{\text{Dim.}}} \mathbf{F}_p^n\right] , \quad \Delta\bar{\mathbf{F}}_p = \Delta\mathbf{F}_p \left(\frac{\Pi(\bar{J}_p^n \Delta J_p)}{\bar{J}_p^n \Delta J_p}\right)^{\frac{1}{\text{Dim.}}} . \tag{5.55}$$

Note that we have defined the relative Jacobian as:

$$\Delta J_p := \det\left(\Delta\mathbf{F}_p\right) = \frac{J_p^{n+1}}{J_p^n} = 1 + \text{trace}\left(\sum_i v_i^{n+1} \otimes_{\text{Outer}} \nabla N_{ip}\right)\Delta t , \tag{5.56}$$

and relative deformation gradient as:

$$\Delta\mathbf{F}_p := \mathbf{F}_p^{n+1} \cdot \left(\mathbf{F}_p^n\right)^{-1} = \mathbf{I} + \left(\sum_i v_i^{n+1} \otimes_{\text{Outer}} \nabla N_{ip}\right)\Delta t \ . \tag{5.57}$$

This is an initial value problem, so we must take pertinent variables to be set to some real number so that we may start a numerical simulation. Initial values may be whatever fits the simulated problem, but we demonstrate here the trivial and common case of no particle deformation at time $t = 0$, stated as:

$$J_p^0 = 1 , \qquad \bar{J}_p^0 = 1 , \qquad \Delta J_p^0 = 1 , \qquad \mathbf{F}_p^0 = \mathbf{I} , \qquad \bar{\mathbf{F}}_p^0 = \mathbf{I} , \qquad \Delta\mathbf{F}_p^0 = \mathbf{I} , \tag{5.58}$$

which includes all necessary components for the B-Spline F-Bar method in explicit MPM as stated in Equations 5.55, 5.56, 5.57, and 5.58.

We believe the added grid update operation and parallel core synchronizations can be aligned to

match certain ordering schemes of stress-strain updates in MPM. Note that we have not fully explored the applicability of Update-Stress-First, Update-Stress-Last, Modified-Update-Stress-Last, etc., on the energy preserving behavior of this B-Spline F-Bar method and alternatives, but there is certainly a potential paper that could be written on the topic. The applicability of various F-Bar / reduced integration methods to stress-update ordering schemes, in terms of energy conservation and parallel computation, is of significant value. Some schemes will be less impacted computationally by the cost of the respective projections and synchronization. This is because they may be naturally aligned with an antilocking formulation's proposed time-points for scattering, syncing, un-weighting, syncing, and gathering relevant values amidst data-structures for high-performance parallel computation.

The primary caveat of the use of F-Bar in this dissertation is that *we do not use the above formulation in our simulations*, but rather a modification on a few key aspects. Namely, we **(i)** apply it to MLS-MPM (Hu et al. 2018) (see Section 5.9), **(ii)** within the constraints of the G2P2G fused algorithm (Wang et al. 2020b) (see Section 5.10), **(iii)** throttled via a linear mixing factor (see Section 5.8), and **(iv)** with improved floating-point precision using a precision-accelerated, subtracted J-Bar form (see Section 5.12). These steps take the B-Spline F-Bar method of Zhao et al. 2023 from being very efficient to being overwhelmingly optimized, down to a GPU's assembly instruction level, for massively parallel, exa-scale simulations while remaining easier to implement than most antilocking methods.

5.8 *Mixed F-Bar Volumetric Antilocking*

We modify the B-Spline F-Bar volumetric antilocking scheme by Zhao et al. 2023 to use a linear mixing ratio, $\psi \in [0, 1]$, for an adjustable balance of pressure field smoothness and low-dissipation stabilization. This linear combination of B-Spline F-Bar in an explicit MPM may or may not be novel, as we haven't seen other papers apply it, but it is so simple that we would

expect it has been used prior. Our antilocking mixing approach is reminiscent structurally to the PIC-FLIP velocity mixing popularized in graphics communities by Zhu and Bridson 2005 and Stomakhin et al. 2013, which we covered in Section 5.6. The assumed deformation gradient determinant for the stress update is defined as:

$$\bar{J}_p^{n+1} = (\psi) \sum_i N_{ip} \bar{J}_i + (1 - \psi) \left(J_p^n \Delta J_p \right) \tag{5.59}$$

$$= (\psi)\, \Pi \left(\bar{J}_p^n \Delta J_p \right) + (1 - \psi) \left(J_p^{n+1} \right) \ , \tag{5.60}$$

which simply adds a weighted grid-averaged deformation gradient from Zhao et al. 2023 to the counter-weighted deformation gradient determinant of standard MPM or MLS-MPM. For $\psi = 0$, the scheme reduces to standard MPM or MLS-MPM. For $\psi = 1$, the scheme becomes that of Zhao et al. 2023. Typical ψ values we use for stiff fluid simulations are between $\psi = 0.9$ to 0.9999, but note this may be application specific.

5.9 *F-Bar Volumetric Antilocking for B-Spline MLS-MPM*

The Moving-Least-Squares Material Point Method by Hu et al. 2018 does not use the velocity gradient, needed for the F-Bar volumetric antilocking of Zhao et al. 2023 in Section 5.7. In this section we discuss substitution of the velocity gradient for the affine velocity matrix of APIC, defined in Jiang et al. 2015, on a particle, C_p, which is utilized in MLS-MPM.

The APIC affine velocity matrix on a particle at time $t + \Delta t = n + 1$ is found by:

$$\mathbf{C}_p^{n+1} = \sum_i N_{ip} \mathbf{v}_i^{n+1} \left(\mathbf{x}_i - \mathbf{x}_p^n \right)^T \left(D_p \right)^{-1} \ , \tag{5.61}$$

where the inertia-like tensor, $\mathbf{D}_p$, is a constant scalar, D_p, under the condition that the

shape-function N_{ip} is a quadratic or cubic B-Spline, set respective to the grid-cell length Δx as:

$$D_p = \frac{\Delta x^2}{4} \text{ or } \frac{\Delta x^2}{3} \text{ if } \quad N_{ip} \quad \text{is a quad. or cubic B-Splines respectively .} \tag{5.62}$$

To begin to apply F-Bar antilocking to MLS-MPM, we simply redefine the relative Jacobian (ΔJ), over a time-step Δt that bridges time $t = n$ to time $t + \Delta t = n + 1$ as:

$$\Delta J_p := 1 + \text{tr}\left(C_p^{n+1}\right)\Delta t \quad \text{where tr}(\cdot) \text{ is the trace operator, i.e. } \text{tr}\,(A) = \sum_{i=0}^{n} A_{ii} \ , \tag{5.63}$$

which is in a high-performance form as only the diagonal of the affine velocity matrix C_p^{n+1} is computed. Next, we redefine the relative deformation gradient of a particle ($\Delta \mathbf{F}_p$), as:

$$\Delta \mathbf{F}_p := \mathbf{I} + \Delta t C_p^{n+1} \text{ where } \mathbf{I} \text{ is the second order Identity tensor .} \tag{5.64}$$

All other definitions and relations provided in Section 5.7 apply to the MLS-MPM implementation of F-Bar volumetric antilocking, given that they do not redefine either $\Delta \mathbf{F}$ or ΔJ as stated in Equations 5.63 and 5.64. In the next section we demonstrate how to apply this F-Bar formulation to fused grid-to-particle-to-grid MLS-MPM algorithms, as implemented in our ClaymoreUW software.

5.10 *F-Bar Volumetric Antilocking for G2P2G Fused Kernels*

The F-Bar volumetric antilocking which was proposed by Zhao et al. 2023 for B-Spline Material Point Methods was covered in Section 5.7 and applied to the B-Spline Moving-Least-Squares Material Point Method of Hu et al. 2018 in Section 5.9. In Section 5.8 we also modified the determination of the assumed deformation gradient and its Jacobian (i.e. determinant) to allow linear mixing with the standard MPM and MLS-MPM forms with an adjustable value, $\psi \in [0, 1]$.

In this section we will show that F-Bar volumetric antilocking by Zhao et al. 2023 is not directly applicable to the fused Grid-to-Particle-to-Grid algorithm of Wang et al. 2020b, which was developed as a way to avoid extra GPU kernel launches and host-device / device-device synchronizations in the high-performance Claymore code ,which our ClaymoreUW code is forked from, and to avoid writing the affine velocity matrix of Jiang et al. 2015 on particle memory for use in MLS-MPM.

In Section 5.7 there is an underlying assumption that we can: **(i)** perform a projection operation $\Pi(\cdot)$ (see Equation 5.38) during G2P to average a particle's assumed Jacobian update ($\bar{J}_p^n \Delta J$) on the shared background grid within the particles shape-function stencil, **(ii)** synchronize all computing cores to guarantee contribution of all particles to all relevant grid-nodes, **(iii)** perform a grid-update to divide out the volume weighting on all grid node assumed Jacobians $\bar{J}_i$, **(iv)** synchronize all computing cores to guarantee completion of the grid-update, and **(v)** finally reconstruct the assumed Jacobian $\bar{J}_p^{n+1}$ during P2G. As the Grid-to-Particle-to-Grid (G2P2G) fused algorithm does not allow for global synchronization of computing cores in the middle of its execution, this is not a feasibly approach to antilocking.

The simplest adjustment would be to simply split apart the G2P2G kernel, but this loses its benefits as we must now perform additional global synchronization (a bottleneck for GPU and Multi-GPU codes) and can no longer hold the affine velocity matrix, C_p, as a temporary value on a given particle p. Note that the latter would add an additional nine values per particle in 3D, equating to 36 - 72 gigabytes of extra GPU data to process and atomically sync across the parallel system per one billion particles, in single and double-precision respectively.

Our proposed solution is to use two variables for evaluating the relative Jacobian, $\Delta J^{n-1/2}$ and $\Delta J^{n+1/2}$, when computing the assumed Jacobian. This allows us to avoid two additional global synchronizations imposed by the form of Zhao et al. 2023 for G2P2G. Because the former value may be determined, applied, and discarded in a previous G2P2G step to kick-start the

grid-projection operation, and the latter value may be determined and applied in the following G2P2G step concurrent to completion of the grid-projection, we avoid a substantial bottle-neck (accelerating the traditional form by 2-3x in G2P2G). Our proposed relative Jacobian variables are constructed as:

$$\Delta J_p^{n-1/2} := \left(\frac{J_p^n}{J_p^{n-1}}\right) \qquad \text{and} \qquad \Delta J_p^{n+1/2} := \left(\frac{J_p^{n+1}}{J_p^n}\right) , \tag{5.65}$$

where $\Delta J_p^{n+1/2}$ is the same as ΔJ_p from the original method in Section 5.7.

We take the following steps in our G2P2G B-Spline F-Bar method for volumetric antilocking: **(i)** project our assumed Jacobian update to the grid using the proposed $\Delta J_p^{n-1/2}$ during a G2P2G step, **(ii)** allow the G2P2G kernel to complete its cycle, causing global synchronization of parallel cores, **(iii)** update the shared grid to remove volume weights from the assumed weighted Jacobians concurrently with the regular grid update in G2P2G, resulting in the assumed Jacobian $\bar{J}_i$ at any grid-node i, **(iv)** allow the shared grid update to end to subsequently force global synchronization of the parallel cores and devices, as standard, and **(v)** reconstruct $\bar{J}_p^{n+1}$ on the particle at the start of a new G2P2G kernel. No new parallel synchronizations are enforced so the high-performance strengths of G2P2G fused kernels from Wang et al. 2020b persist.

With the assumed Jacobian $\bar{J}_p^{n+1}$ available, the next step is to to determine any further required variables variables (e.g. the assumed deformation gradient or assumed relative deformation gradient, $\bar{\mathbf{F}}_p^{n+1}$ and $\Delta\bar{\mathbf{F}}_p$). To do so, use the relative Jacobian, $\Delta J_p^{n+1/2}$, which is calculated respective to the updated grid velocities, $\mathbf{v}_i^*$, determined in the standard MPM Grid Update (see Section 8.4.1 for review) concurrent to the assumed Jacobian's nodal update. This is written as:

$$\Delta\bar{\mathbf{F}}_p^{n+1/2} = \left(\frac{\Pi\left(\bar{J}_p^n \Delta J_p^{n-1/2}\right)}{\bar{J}_p^n \Delta J_p^{n+1/2}}\right)^{1/\text{Dim.}} \Delta\mathbf{F}_p^{n+1/2} , \tag{5.66}$$

which is no different than the original Equation 5.52 besides projecting a prior relative Jacobian to

avoid global synchronizations. This value was computed at a prior particle position that interpolated a prior velocity field from the grid. Subsequently, it is taken for granted that interpolating the velocity field, which updated concurrently with the F-Bar projection, in the next G2P2G step in order to compute the traditional relative Jacobian $\Delta\bar{J}_p^{n+1/2}$, and then applying it with the reconstructed F-Bar projection of the Jacobian (based on $\Delta\bar{J}_p^{n-1/2}$), to solve Equation 5.66 is a valid sequence of events.

However, we have not done an intensive study on the energy conservation and convergence properties of the modified B-Spline F-Bar volumetric antilocking proposed here for the fused G2P2G algorithm insofar. Likewise, there has not been a comprehensive study done for the formulation of Zhao et al. 2023, so both remain as a future line of study.

Our G2P2G B-Spline F-Bar antilocking approach may or may not share the stability and efficacy of the original split kernel approach, although we have not observed any meaningful difference in pressure field smoothness or required time-step between the two in our own simulations. However, our G2P2G F-Bar antilocking is certainly faster as it fits into the fused G2P2G algorithm optimization for high-performance Multi-GPU MPM codes by Wang et al. 2020b without introducing any new parallel synchronizations. Our G2P2G antilocking scheme is implemented in our open-source ClaymoreUW Multi-GPU software, which builds off the original Claymore by Wang et al. 2020b. Examples of exa-scale fluid simulations with antilocking may be found in Section 6.4 and 7.7.3, where we simulate a billion nearly-incompressible fluid particles over a million time-steps with smooth, accurate pressure fields. This demonstrates stable behavior over one quadrillion executions of our anti-locking algorithm in G2P2G MLS-MPM. Significantly, both simulations took 24 hours to complete on just four GPUs occupying a single node of a supercomputer (Texas A&M University's ACES cluster, four NVIDIA H100 PCIe5 80 GB cards). To our knowledge, no other antilocking scheme in MPM has been demonstrated to achieve stable, high-performance of the same magnitude.

In the next section, we present a reduced formulation of this antilocking scheme optimized for materials that do not require the entire deformation gradient, rather just its determinant (i.e. the Jacobian, $J := \text{Det}(\mathbf{F})$). This include compressible and weakly-compressible isotropic fluids, which we described in Section 5.4 for the Murnaghan-Tait equation of state that is commonly referred to as "J Fluid".

5.11 *J-Bar Fluid*

A common material model for fluids is the J Fluid, i.e. any isotropic fluid model which only requires the Jacobian, J, of its deformation gradient, $\mathbf{F}$, to resolve its equation of state (i.e. relationship of deformation and pressure). A common EOS in MPM and SPH is the Murnaghan-Tait model (see Section 5.4) which is applicable for many weakly-compressible fluids, including water for coastal engineering purposes as used in Chapter 7, 8, and 9. As many multi-phase problems studied numerically require a large fluid volume interacting with potentially small solid volumes, a method such as MPM is forced to represent a massive number of fluid particles on top of a very high resolution background grid in order to resolve the small length-scale of fluid-solid interaction accurately. This fluid is often very stiff, as it is for water, so volumetric locking occurs in MPM unless adjusted for. However, having many stiff fluid particles on a fine grid with additional overhead from an antilocking method is computationally expensive. It is desirable to have a fluid model that is very computationally fast, memory-light, antilocked, and simplistic in conception.

In this pursuit, we develop the "J-Bar Fluid" ($\bar{J}$ Fluid, i.e. JB Fluid) using a reduced form of the full B-Spline F-Bar volumetric antilocking of Zhao et al. 2023. Our simplification is applied to a J Fluid material, specifically one using the Murnaghan-Tait equation of state (see Section 5.4). This section describes the JB Fluid formulation and implementation. JB-Fluid is compatible with both MPM and MLS-MPM (Hu et al. 2018, Section 2.5) along with the standard G2P + P2G or the

fused G2P2G algorithms. Included is our antilocking linear mixing strategy (see Section 5.8) to provide balance between antilocking and dissipation through a controlled ratio $\psi \in [0, 1]$.
Our JB Fluid model is viable for massive simulations of, volumetric antilocked fluid particles in high-performance with near optimal memory usage and minimized read-write instructions. In Section 5.12 we discuss the Precision-Accelerated J-Bar Fluid (i.e. PA-JB Fluid) which is a novel high-performance reformulation of our JB Fluid that substantially improves accuracy, memory usage, and computational time suitable to exa-scale computation.

5.12 *Precision-Accelerated J-Bar Fluid*

We propose a straightforward reformulation of common nearly-incompressible material models, garnering significant gains in precision, memory footprint, and relative computational speed. The Tait-Murnaghan equation of state (EOS), a relationship between volume change and pressure for isotropic fluids, has found wide-spread use in engineering and computer graphics. However, despite its use in simulations with billions of numerical bodies undergoing millions of time-steps, little optimization is done on the implementation. This oversight can lead to inefficient computations and instabilities for near-incompressible flow at low pressure, e.g. for shallow-water simulations. In this section we propose an approach that resolves this issue and is well-suited to modern high-performance computing (HPC) hardware, e.g. Graphics Processing Units (GPUs), due to advantages for mixed-precision, minimal use of registers, and compact format for communication between GPUs and HPC nodes (i.e. Multi-GPU, Multi-Node systems). Our formulation in single precision shows superior performance compared to the traditional approach in both single and double precision. As example, we simulate shallow water in the Material Point Method to demonstrate massive improvement in pressure calculations within +/- 1000 Pascals. We provide an open-source Python, C++, and CUDA C++ implementation of both the traditional and precision-accelerated material model at `https://github.com/JustinBonus/Optimized-Fluid`.

5.12.1 Introduction

Computation is shifting– From isolated systems to massive data-centers. From CPUs to Multi-GPU clusters. From serial implicit simulations to massively parallel explicit cases. As vertical scaling of Mohr's law nears physical limits our sights move towards non-classical means of computing. This decade marks our collective first-step into the exa-scale paradigm. Year-over-year, individual supercomputers running a quintillion double-precision floating-point operations per second is to become more common, with scientific simulations and machine learning models making ample but not ideal use. To capitalize on this change, long-standing approaches must be re-evaluated in new contexts. Algorithms performing quadrillions of times in a day should be optimized at a low-level, as done with sorting algorithms (Mankowitz et al. 2023) through reinforcement learning on low-level assembly instructions. Likewise, material models executing quintillions of times per day, the subject herein, should be optimized first by programmers and in future works via machine learning for the benefit of engineers, animators, and energy efficiency at large.

This section reformulates implementation of a long-standing material model common to weakly-compressible, i.e. nearly-incompressible, fluid and solid simulation. Novelty is **(i)** negation of the expensive act of raising a double-precision floating-point number to a power, **(ii)** a bypass to floating-point catastrophic cancellation, a source of significant error for nearly-imcompressible material laws, and **(iii)** suitability to next-generation HPC hardware, e.g. Multi-GPU Multi-Node HPC systems. We achieve this by: **(i)** replacing the broad "raise to the power of" (pow) operation with specialized exponential and natural logarithm operations in "exponential of a number, minus 1" (expm1) and "natural logarithm of a number plus 1" (log1p) forms, **(ii)** working with variable forms more favorable to floating-point representation, and **(iii)** in-depth evaluation of our formulations mixed-precision viability, register usage, floating-point error near the imcompressible limit, shared memory, global memory, and Multi-GPU

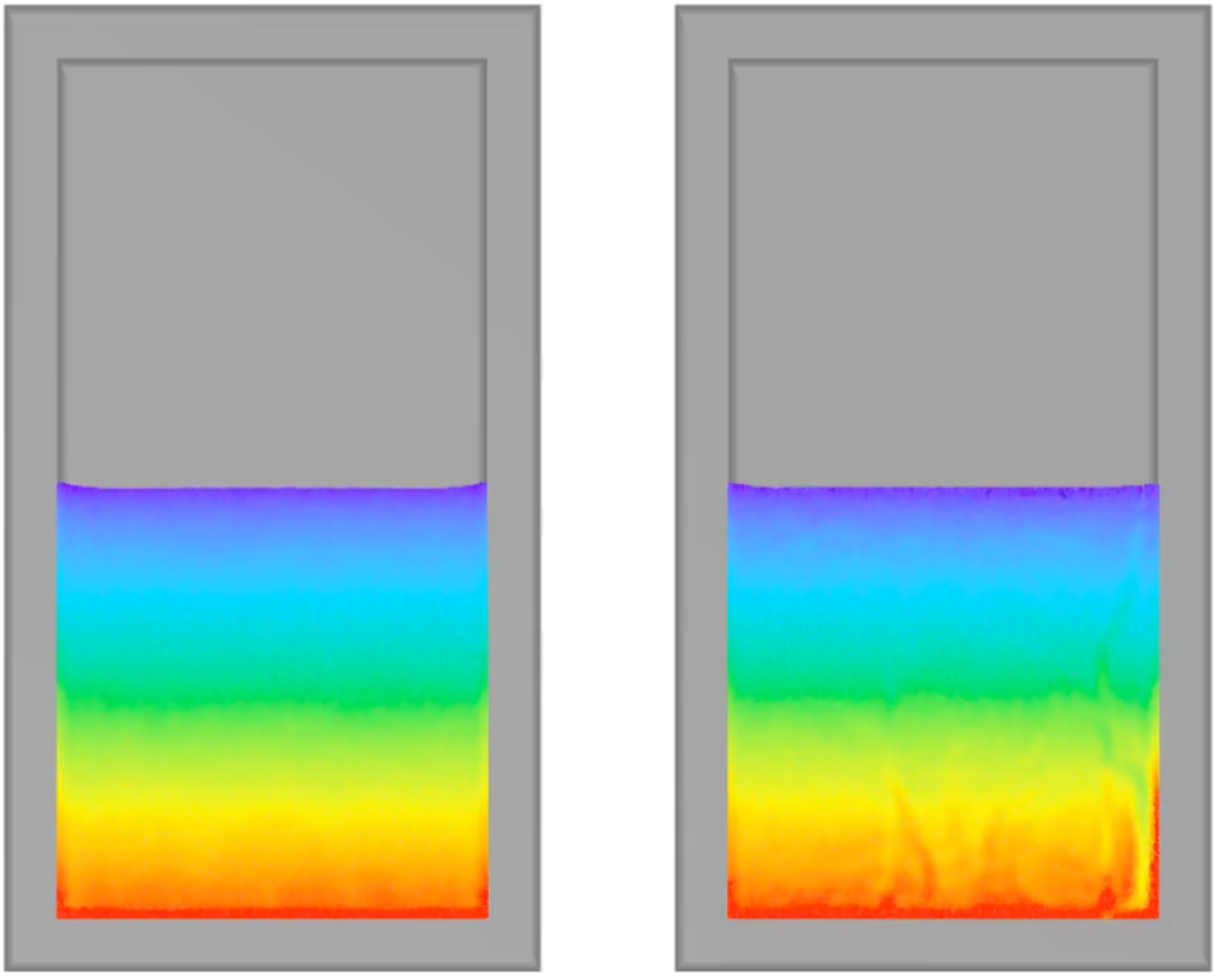

Fig. 5.2. Improved Pressure Field Of Hydrostatic Tank of Water Via Proposed Material Law Formulation. Visualization of a 3.2 x 3.2 x 0.4 m tank of water. (Left) PA-JB Fluid formulation in single-precision. (Right) Traditional J Fluid formulation in single-precision. Snapshots taken three seconds into the simulation with pressure visualized on fluid particles.

communication in the context of next-generation hardware. All variables within this section are listed in Table 5.2 for convenience.

Many simple equations of state (EOS), a relationship between a materials volume change and pressure, are used to describe liquids, gasses, and crystalline solids. Popular EOS include the Tait equation (Tait 1888), Murnaghan-Tait equation (Murnaghan 1944), Cole equation (H. 1948),

Table 5.2. List of variables / function names, their descriptions, and units.

Symbol(s)	Description	Units
P	Pressure	Pa
P_0	Ambient Pressure	Pa
B	Bulk Modulus	Pa
B_0	Initial Bulk Modulus	Pa
B'_0	Bulk Deriv. w.r.t. Pressure	-
V	Deformed Volume	m^3
V_0	Initial Volume	m^3
$\mathbf{F}$	Deformation Gradient	-
J	Volume Change Ratio $J = V/V_0 = \det(\mathbf{F})$	-
j	Rel. Vol. Change Ratio $j = 1 - J$	-
e	Euler's Number	-
exp()	Math Exponential Func. $\exp(x) = e^x$	-
expm1()	Math Exponential - 1 Func. $\mathrm{expm1}(x) = e^x - 1$	-
log()	Math Logarithm Func. $\log(x) = \log_e(x)$	-
log1p()	Math Logarithm + 1 Func. $\mathrm{log1p}(x) = \log_e(x + 1)$	-
pow()	Math Power Func. $\mathrm{pow}(x, y) = x^y$	-
fma()	Fused Multiply-Add $\mathrm{fma}(x, y, z) = (x * y) + z$	-
FP16	Half-Prec. Floating-Point, 16 bit	-
FP32	Single-Prec. Floating-Point, 32 bit	-
FP64	Double-Prec. Floating-Point, 64 bit	-

MacDonald equation (MacDonald 1966) (MacDonald 1969), and third-order Birch-Murnaghan equation (Dass and Kumari 1985). All benefit from optimizations discussed here, although we focus on the Murnaghan-Tait EOS.

Note that applicability of the Murnaghan-Tait equation to a problem is not a given, and that there are valid criticisms regarding its mathematical derivation, as noted in Poirier 2000, as is the case for many common EOS. However, it has been successfully used to replicate experimental observations of real isotropic, isothermal media when volume change is within 10%. Examples include shallow-water, e.g. wave-flumes studies with sub-5 cm flow depths (Goseberg et al. 2016 and Park et al. 2021), and visual applications including animated movies and video-games. The traditional but flawed implementation is currently used within open-source codes DualSPHysics (Domínguez et al. 2022), Taichi (Hu et al. 2019a), and Claymore MPM (Wang et al. 2020b), to name a few. This implementation introduces compounding errors into shallow, nearly-incompressible pressure fields and increases computational times for a given precision. This is a widespread flaw that should be addressed before extending to exa-scale magnitudes. In the following section, the common Murnaghan-Tait EOS implementation is described.

5.12.2 Traditional Formulation

The Murnaghan-Tait equation enjoys ongoing popularity as a broad fluid and solid material. It is often used as a Weakly-Compressible Isotropic Fluid (Pradhana et al. 2017), a.k.a J-Fluid. The key feature is that **(i)** it is compressible, **(ii)** it is easily performed explicitly, and **(iii)** stresses are computed solely from the volume change ratio expressed by the material's deformation gradient (J, hence J-Fluid). The model relies on an equation of state (EOS) to define its constitutive relationship, relating volume change to pressure.

The Murnaghan-Tait Equation of state is simple to implement and fairly robust for weakly-compressible scenarios (e.g. not the bottom of the ocean). It is isothermal and assumes

the bulk modulus (B) is a linear function with pressure (P):

$$B = B_0 + PB_0' \ , \tag{5.67}$$

which gives rise to a pressure relationship with volume change. Respective to an initial state, the pressure is defined as:

$$P = \frac{B_0}{B_0'}\left(\frac{V_0}{V}^{B_0'} - 1\right) + P_0 \ , \tag{5.68}$$

where the bulk modulus (B_0) and derivative of bulk modulus with respect to pressure (B_0') for pure water are experimentally taken as 2.2e9 Pa and 7.15 for typical sea-level conditions. Ambient pressure (P_0) is not a concern for most near-surface simulations as we do not observe nonlinear bulk modulus behavior near this range.

Using $J = \frac{V}{V_0}$, where V_0 and V are initial and deformed volume of the material, equation 5.68 may be written as:

$$P = \frac{B_0}{B_0'}\left(J^{-B_0'} - 1\right) + P_0 \ , \tag{5.69}$$

and subsequently phrased to use the pow() function as is done in traditional computational implementations:

$$P = \frac{B_0}{B_0'}\left[\text{pow}\left(J, -B_0'\right) - 1\right] + P_0 \ , \tag{5.70}$$

which proves to be an easy line of code to implement. However, the formulation's simplicity distracts from its flaws regarding precision, memory use, and speed.

5.12.3 Improved Formulation - or Straightforward re-Formulation

Recognizing that nearly-incompressible fluids inherently have minimal deformations, except in extreme scenarios, we can posit that J will tend to be near 1. Thus, $1 - J$ will be near 0. We will

call this relative deformation term j, stating it as:

$$j = 1 - J \ , \tag{5.71}$$

where positive and negative values correspond to volume compression and tension respectively. For nearly-incompressible material j, Eq. 5.71 will be within a small neighborhood of zero (e.g. +/- 1e-10). This property is problematic for the traditional J approach, which both lacks the floating-point precision to represent J near 1 and includes catastrophic cancellation. However, we leverage this property to improve precision with a j formulation.

By starting from Equation 5.70, we avoid raising an arbitrary number, J, to the power of another arbitrary number B_0' by using an exponential and natural logarithm.

$$P = \frac{B_0}{B_0'} \left(e^{-B_0' \log_e(J)} - 1 \right) + P_0 \ , \tag{5.72}$$

$$P = \frac{B_0}{B_0'} \left(e^{-B_0' \log_e(1-j)} - 1 \right) + P_0 \ , \tag{5.73}$$

$$P = \frac{B_0}{B_0'} \left[\exp\left(-B_0' \log\left(1 - j\right) \right) - 1 \right] + P_0 \ , \tag{5.74}$$

Note that Equation 5.74 does not avoid catastrophic cancellation. It is also near identical to the actual implementation of pow(x,y), which is typically exp2(y * log2(x)) in many math libraries. By reformulating to use expm1 and log1p, available in almost all common programming languages (e.g. Python, C++, CUDA), we avoid catastrophic cancellation and greatly improve precision when J is close to 1. Pressure is implemented as

$$P = \frac{B_0}{B_0'} \ \mathrm{expm1}\left(-B_0' \ \mathrm{log1p}\left(-j\right) \right) + P_0 \ . \tag{5.75}$$

which is a far more optimal implementation of the EQS than seen in the wide-spread Equation 5.70. Note that large values of P_0 can reduce the precision represented in the change in pressure the EOS discerns, but there are many numerical codes that already separate pressure change from ambient pressure at all times except when it is absolutely necessary to combine the two (e.g. centering relative to background pressure) and for user outputs. If ambient pressure P_0 is non-zero, we can directly optimize the multiplication and addition being performed with a fused multiply-add operation $\text{fma}(x, y, z) = (x * y) + z$. This improves speed and precision slightly. Occasionally compilers will perform the conversion automatically, but the operation can be forced as shown symbolically with:

$$P = \text{fma}\left(\frac{B_0}{B_0'},\ \text{expm1}\left(-B_0'\ \text{log1p}\left(-j\right)\right),\ P_0\right)\ . \tag{5.76}$$

The key to using j is to work directly with it, i.e. do not calculate it from J. This means initializing the simulation using $j = 0$ if using an undeformed initial state, calculating incremental updates to j directly, and performing averaging operations in the algebraically correct form. These are lax requirements that are sufficiently met by algebraic reorganization of a few equations in common numerical methods.

5.12.4 High-Performance Benefits

Our improved equation of state (EOS) formulation in Equation 5.75 in single-precision is well-suited to modern hardware, e.g. for massively parallel simulations on Graphics Processing Units (GPUs). It's high-performance computation benefits are:

Mixed Precision Viability. As demonstrated in Figures 5.3 and 5.4, the improved formulation in single-precision has less pressure error than the traditional approach in single and double precision. As most engineering simulations run in double-precision, this means one could pair our

Table 5.3. Units in Last Place Error of functions used for implementing the Tait-Murnaghan equation of state in CUDA C++. Lower is better. (NVIDIA 2023)

Function	ULP (FP32)	ULP (FP64)
powf / pow	4	2
expf / exp	2	1
logf / log	1	1
expm1f / expm1	1	1
log1pf / log1p	1	1

single-precision implementation with an otherwise double-precision method (e.g. regarding shape-functions, advection, etc.). Mixing precision better utilizes GPU hardware because NVIDIA cards possess independent single and double precision floating-point operation units on-chip. If an entire simulation is double precision then lower-precision computational hardware is idle and double precision computations may have to wait longer to execute due to over-demand of double precision hardware. Our formulation lessens the demand on double precision units, makes use of single precision units, accelerates the simulations, and still produces improved pressures over the traditional approach.

Smaller ULP Error. Table 5.3 details ULP error for the mathematical functions of interest across floating point representations. Both functions in our reformulation have ULP of just 1 in single and double precision, whereas the traditional form has 4 and 2 respectively. As lower ULP is better, this shows that these math functions as they are implemented in CUDA C++ for GPU usage are objectively more precise in our formulation (NVIDIA 2023) (NVIDIA 2022).

Fewer Registers Used. Register usage is often a bottleneck for GPU occupancy. Applications may request more registers per CUDA core than available, restricting the number of CUDA cores running in parallel. When implementing material models using math functions, such as pow() or sin(), register usage can grow far past programmer expectations and reduce application speeds. Even a simple model such as the Murnaghan-Tait EOS can bottleneck simulations. Register usage

Table 5.4. Register use (32 bit) for implementations of the Tait-Murnaghan equation of state in CUDA C++. Lower is better.

Register Use	Traditional	Improved	Saved
FP32 (sm_62)	20	18	2
FP32 (sm_75)	19	18	1
FP32 (sm_80)	-	-	-
FP64 (sm_62)	28	24	4
FP64 (sm_75)	40	26	14
FP64 (sm_80)	-	-	-

can vary depending on card architecture, functions used, and various compiler optimizations, so numbers are subject to change (NVIDIA 2023). Table 5.4 shows that in CUDA C++ our formulation uses 2 and 4 less 32 bit registers for single and double-precision respectively when compared to the traditional form. Note that this should be profiled directly by users for their specific implementations, especially on compute-capability 5.2 cards and older, or versions of CUDA before 10.2. Further, our implementation in single precision is more accurate than the traditional approach in double-precision for low pressure cases found in computer-graphics and coastal engineering. We assert that our single-precision improved formulation saves 50-65% registers compared to the traditional formulation in double-precision 5.4.

Halved Shared-Memory. Shared memory is a low-latency memory pool shared by groups of GPU cores. It is at least 48 kilobytes, so programmers design with respect to this number, as it is 10x faster to access than global GPU memory and 100x faster than standard computer memory (e.g. RAM). For MPM, FEM, SPH, etc. simulations it is often used as an intermediate memory pool for averaging, integrating, or reconstructing values from shape-functions. One notable use is for assumed deformation anti-locking schemes, e.g. F-Bar volumetric anti-locking. Implemented in MPM software such as Claymore (Wang et al. 2020b) in double precision would require 8 kilobytes of additional shared memory, or 16.7% of the pool per CUDA block on a SM. In single-precision, feasible in our improved formulation, only 4 kilobytes is used (8.3% total). This

Table 5.5. GPU global memory usage for 1 billion MLS-MPM particles using Tait-Murnaghan equation of state in CUDA C++. Lower is better. Double-buffering assumed, as in (Wang et al. 2020b).

Case	Traditional	Ours	Reduction
-	64 GB	56 GB	12.5%
F-Bar	80 GB	64 GB	20.0%

is a substantial reduction that can have a significant impact on GPU occupancy (effective utilization of all available hardware, e.g. CUDA cores) and thus performance. Further, as we read and write less memory to shared memory, operations which can dominate kernel run-times in MPM simulations, said operatations are halved. Another significant advantage of our formulation is that we use $j = 1 - J$ instead of J, thus we average a number which is better represented in floating-point form without loss of information and thereby a method like F-Bar volumetric anti-locking is improved in precision with no added effort.

Halved Global-Memory. Global memory is the largest memory pool on a GPU, typically ranging from 4 - 80 gigabytes. For 100 million+ numerical body simulation this can dominate memory use and provide an upper bound on how large GPU simulations can be before scaling to multiple GPUs and multiple supercomputer nodes. In MLS-MPM (Hu et al. 2018) as implemented by (Wang et al. 2020b) a fluid particle using Murnaghan-Tait EOS requires only four unique parameters: position (x,y,z) and volume change ratio J. Assuming double-precision, this requires 32 bytes of data per particle. Table 5.5 shows global memory used in a billion particle MLS-MPM simulation using the traditional and our reformulated model with and without F-Bar anti-locking. Note that saved memory exceeds what most consumer GPUs possess, highlighting the benefits of low-level optimization as we approach exa-scale simulations. Note that these memory reductions don't just save memory, they save the time cost of reading from and writing to global memory every time-step (300 clock cycles) which is a common bottle-neck of performance. Our

Table 5.6. GPU communication memory usage reduction for MLS-MPM using Tait-Murnaghan equation of state reformulation in CUDA C++. Higher is better. Assuming memory layout used in Claymore MPM, (Wang et al. 2020b).

Partition	Case	Reduction
Particles	-	12.5%
Particles	F-Bar	20.0%
Grid-Nodes	-	-
Grid-Nodes	F-Bar	20.0%

seemingly small optimizations in memory use of a simple material law has profound consequences for memory use and run-times in high-performance applications.

Fast Multi-GPU Communication. Truly massive high-performance simulations, 1 billion+ numerical bodies, exceed capabilities of a single GPU. Thus, we must be able to split simulations across **(i)** multiple GPUs on a node and **(ii)** multiple nodes in a system. This partitioning requires GPUs and nodes to communicate relevant data among themselves so that the simulation is unified. Even modern Multi-GPU interconnects, e.g. PCIe Gen. 5, which can reach 900 GB/s, communicating data every time-step can slow application performance and limit scaling. For multi-node communication, e.g. using InfiniBand at 128 GB/s, this limit is more notable. It is highly desirable to reduce the size of data being communicated every time-step. Our method applied to MLS-MPM can reduce size of communicated data, as shown in Table 5.6. Using particle partitions (particle data exchanges between GPUs) reduces data size by 12.5% and 20% for the Murnaghan-Tait EOS and Murnaghan-Tait EOS with volumetric anti-locking respectively. For grid partitions (grid-data exchanges between GPUs), it reduces data size by 20% when using volumetric anti-locking. These savings can proportionally improve application performance when bottle-necked by Multi-GPU Multi-Node communication, as common in massive simulations of the modern era.

5.12.5 Benchmarks

To show the benefits of the proposed reformulation we benchmark the Murnaghan-Tait equation of state with four formulations, using two levels of floating point precision, on two different benchmarks. In the first benchmark, both models are implemented in Python 3.7 for accessibility and readability. The second benchmark implements formulations in CUDA C++ within an open-source Multi-GPU MPM software developed by our lab (ClaymoreUW, see Sec. 4). The benchmark cases simply involve taking N log-distributed points between either 0 and +/- 1e-7 or 0 and +/- 1e-15 (i.e., the ranges of compressiblity where errors are observed to grow large for single and double precision, respectively), where $N = 10,000$, as sample J values and inputting them into respective traditional and precision-accelerated formulations of the Murnaghan-Tait equation of state (EOS). The EOS is tuned to represent water (bulk modulus $B = 2.2e9$ Pa, $B_0' = 7.15$). Resulting pressure values are plotted against their input j value in the top plots of Figures 5.3 and 5.4. On the bottom plots, we show the percent error of the calculated pressure of all given EOS formulation and precision relative to what we take as "ground-truth" pressure (i.e. a quadruple precision calculation of the EOS) as a normalization effort. This allows the quality of an approach and precision to be represented simply by remaining close to the X-axis (i.e. maintaining a percent error close to zero across the span of relative deformations).

Figure 5.3 shows that traditional formulations for Murnaghan-Tait EOS cannot resolve pressures +/- 1000 Pascals (or +/- 10 cm of the free-surface) without notable error in single-precision. Note that we tune the model to replicate water (B_0 = 2.2e9 Pa, B_0' = 7.15). The top plot demonstrates a "stair-stepping" of pressures near zero, i.e. inability to smoothly approximate the underlying function, for the traditional J and modified tradition $(1 - J)$ models. Contrarily, our improved formulation in single-precision has no discernible issues, laying on the nearly flat slope as ideally desired. All models in double precision perform well in this range. The bottom plot quantifies the percent error relative to the improved formulation in quadruple precision (taken as the

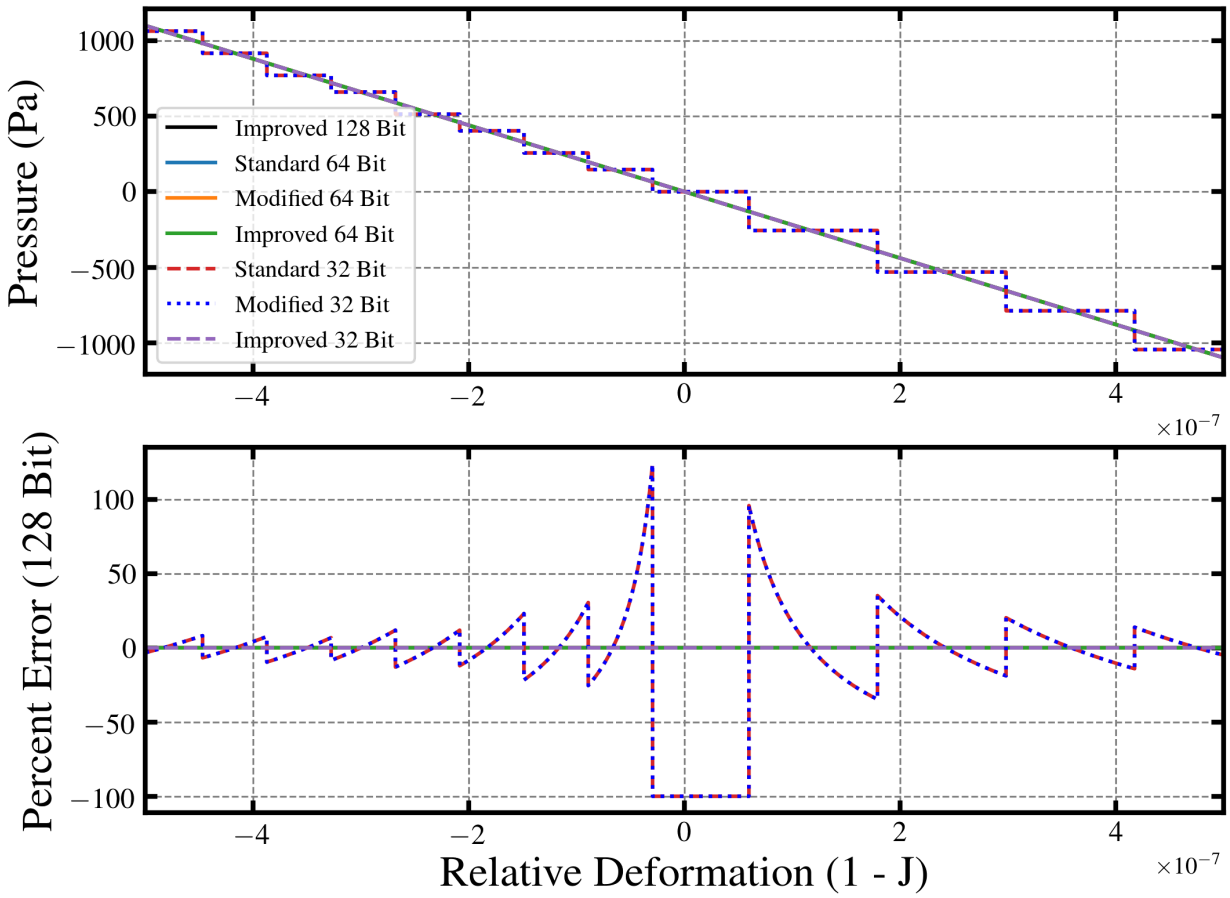

Fig. 5.3. Water Pressure Comparison of Formulations Within 10 centi-meters of Free-Surface. The traditional, modified, and improved formulations of the Tait-Murnaghan equation of state are set to model water and are compared for deformations relating to +/- 10 cm of a free-surface. Single and double precision implementations are used. The improved formulation in qadruple precision is taken as the true pressure for error comparison. Note that the improved formulation in single-precision outperforms both the traditional J and modified sub-J formulations in single-precision.

"ground-truth"). Error exceeds 100% near zero and features multiple discontinuities for the traditional models in single-precision, whereas our single-precision improved model has error of nearly zero throughout the range. Errors for all double-precision models are negligible.
Figure 5.4 shows model performance for water pressures +/- 2e-6 Pa, or +/-200 nano-meters of the free-surface (less than the width of a hair). This incredibly thin range is selected not due to relevance in coastal engineering and computer graphics, but to demonstrate limits of even double precision models. The top plot shows that traditional formulations again fail with "stair-stepping" pressures in this range, yet our improved formulation in both double and single-precision still have no discernible pressure faults. This is a substantial finding as our single-precision implementation outperforms the traditional double-precision approach, whereas the single-precision traditional forms are unable to predict pressures not equal to zero in this range. The bottom plot quantifies percent error of the models relative to the improved formulation in quadruple precision (128 bit). Again, the traditional models show saw-blade errors while the improved forms near zero. The single-precision traditional model have constant percent error of -100% in this range as they can only predict zero pressure.

5.12.6 Concluding Remarks

Computation is shifting– From memory-restricted systems to massive data-centers. From CPUs to GPU clusters. From individuals to communities. To capitalize on this change, long-standing methods must be re-evaluated in new contexts.

5.13 *Finite-Element Coupling*

Finite elements offer the opportunity to model solids with better accuracy. Given the close relationship between FEM and MPM they have been coupled in our implementation. This is established on both single and Multi-GPU systems in the open-source MLS-MPM software

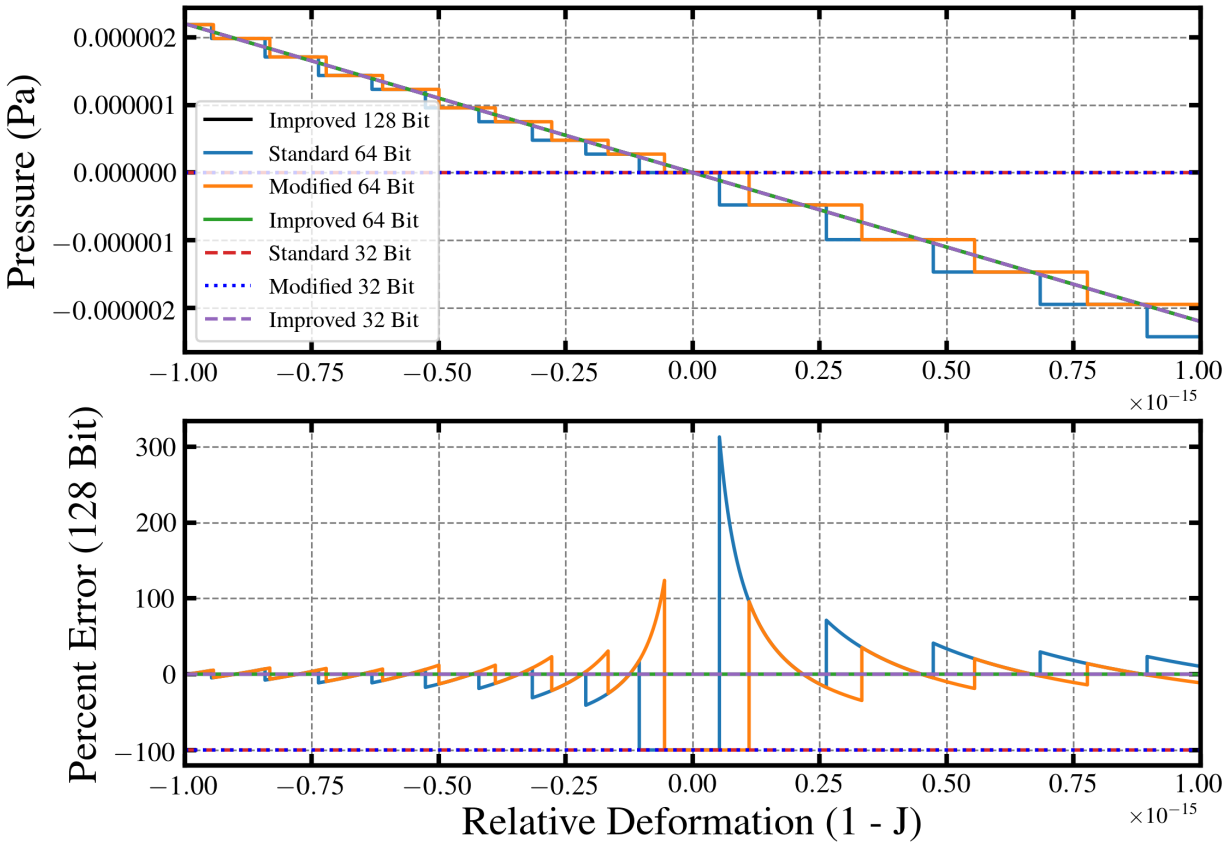

Fig. 5.4. Water Pressure Comparison of Formulations Within 200 nano-meters of Free-Surface. The traditional, modified, and improved formulations of the Tait-Murnaghan equation of state are set to model water and are compared for deformations relating to +/- 10 cm of a free-surface. Single and double precision implementations are used. The improved formulation in qadruple precision is taken as the true pressure for error comparison. Note that the improved formulation in single-precision outperforms both the traditional J and modified sub-J formulations in double and single-precision.

Claymore (Chp. 4) but the formulation can also apply to traditional CPU and Multi-CPU systems.

Coupling was undertaken to: (i) Augment the functionality of MPM, (ii) improve debris-fluid-structure interaction, (ii) impose constraints on the ASFLIP contact algorithm, (iv) leverage the mature body of FEM literature, and (v) provide an open-source Multi-GPU FEM-MPM code for the public.

Finite elements improve debris behavior by refining stress-strain behavior of stiff debris with lessened memory usage. FEM is preferable for simulating small-deformations, as relatively few elements can produce better results than many more MPM particles. This is mostly from the use of Gaussian points within elements for deformation and stress calculation, as opposed to quadrature points (particles) within grid-cells for MPM.

Finite Elements improve debris-fluid-structure interaction (DFSI) by decoupling the constitutive updates of elements. This prevents stress generation between elements and thereby between separate debris. The Material Point Method uses grid-cells as temporary elements and material points as quadrature points (which move freely between grid-cells). Therefore stress may generate between points that may not belong to the same body (e.g. two separate pieces of debris) if they are in the same grid-cell. In contrast, FEM element stress converts to internal forces on particles, which writes to shared grid-nodes by shape-functions. In practice, this improves sub-grid contact but does not eliminate erroneous contact completely.

ASFLIP (Section 5.6.7) is powerful in its simplicity, but the positional escape coefficient (β_p) can lead to stiff, solid particles escaping from their bodies. FEM imposes a constraint on ASFLIP. Performing the constitutive update in elements, not on particles, penalizes particles that attempt to escape their body due to ASFLIP noise with large internal forces. This prevents undesired topology change for hyper-elastic constitutive models (e.g. Fixed-Corotated solid, Sec. 5.4). FEM coupled with ASFLIP garners contact benefits without losing the form of bodies.

With this context, we now formulate a simple but robust MPM-FEM coupling, which has been

implemented in the open-source Multi-GPU MLS-MPM software Claymore (Chp. 4). Linear displacement, constant stress tetrahedron elements (four vertices, one quadrature point) for large-deformations were selected and were implemented as follows below.

Starting from the rest coordinates of a given element (a.k.a. undeformed, material, initial), an edge matrix is assembled to describe the element. We choose to make it relative to an arbitrary, undeformed vertex $\mathbf{x}_a^0$, as

$$\mathbf{D}_\mathrm{m} = \begin{bmatrix} \mathbf{x}_b^0 - \mathbf{x}_a^0 & \mathbf{x}_c^0 - \mathbf{x}_a^0 & \mathbf{x}_d^0 - \mathbf{x}_a^0 \end{bmatrix} = \begin{bmatrix} x_b^0 - x_a^0 & x_c^0 - x_a^0 & x_d^0 - x_a^0 \\ y_b^0 - y_a^0 & y_c^0 - y_a^0 & y_d^0 - y_a^0 \\ z_b^0 - z_a^0 & z_c^0 - z_a^0 & z_d^0 - z_a^0 \end{bmatrix} , \tag{5.77}$$

which is a precomputable quantity, as is its inverse, when implemented.

The undeformed, resting volume of the tetrahedron element (V^0) is precomputed now for later use as

$$V^0 = \mathrm{abs}\left(\frac{\det |\mathbf{D}_\mathrm{m}|}{6} \right) . \tag{5.78}$$

In deformed (i.e. world, spatial) coordinates a deformed edge matrix ($\mathbf{D}_\mathrm{s}$) is assembled as

$$\mathbf{D}_\mathrm{s} = \begin{bmatrix} \mathbf{x}_b^n - \mathbf{x}_a^n & \mathbf{x}_c^n - \mathbf{x}_a^n & \mathbf{x}_d^n - \mathbf{x}_a^n \end{bmatrix} = \begin{bmatrix} x_b^n - x_a^n & x_c^n - x_a^n & x_d^n - x_a^n \\ y_b^n - y_a^n & y_c^n - y_a^n & y_d^n - y_a^n \\ z_b^n - z_a^n & z_c^n - z_a^n & z_d^n - z_a^n \end{bmatrix} . \tag{5.79}$$

The deformation gradient ($\mathbf{F}$) at the tetrahedron's single quadrature point is simply the deformed edge matrix ($\mathbf{D}_\mathrm{s}$) multiplied with the undeformed edge matrix inverse ($\mathbf{D}_\mathrm{m}{}^{-1}$):

$$\mathbf{F} = \mathbf{D}_\mathrm{s} \mathbf{D}_\mathrm{m}{}^{-1} . \tag{5.80}$$

For a deformation gradient at the element quadrature point, the constitutive law produces the first

Piola-Kirchoff stress as

$$\mathbf{P} = P(\mathbf{F}) \ . \tag{5.81}$$

We used a fixed-corotated hyper-elastic model (Sec. 5.4) in our simulations, but this formulation works for other models as well (e.g. modified non-associative Cam-Clay) so nuanced finite element material behavior can be achieved.

Knowing the first Piola-Kirchhoff stress ($\mathbf{P}$) at the element quadrature point, we now seek to determine the internal force contribution it creates at each vertex. This will require a mapping ($\mathbf{B}_{\mathrm{m}}$) which is essentially a collection of the weighted vertex normal vectors in undeformed space.

The area weighted normal vectors of each face on the tetrahedron element in undeformed space are calculated as

$$N_{abc} = \frac{(\mathbf{x}_b - \mathbf{x}_a) \times (\mathbf{x}_c - \mathbf{x}_a)}{2} \ , \tag{5.82}$$

$$N_{adb} = \frac{(\mathbf{x}_d - \mathbf{x}_a) \times (\mathbf{x}_b - \mathbf{x}_a)}{2} \ , \tag{5.83}$$

$$N_{acd} = \frac{(\mathbf{x}_c - \mathbf{x}_a) \times (\mathbf{x}_d - \mathbf{x}_a)}{2} \ , \tag{5.84}$$

$$N_{bcd} = \frac{(\mathbf{x}_d - \mathbf{x}_b) \times (\mathbf{x}_c - \mathbf{x}_b)}{2} \ , \tag{5.85}$$

and then the average area weighted normal vectors of all three faces incident to a vertex are found to determine the weighted vertex normal vector in undeformed space, calculated as

$$\mathbf{B}_b = \mathrm{sign}(V^0) \frac{N_{bcd} + N_{adb} + N_{abc}}{3} \ , \tag{5.86}$$

$$\mathbf{B}_c = \mathrm{sign}(V^0) \frac{N_{bcd} + N_{acd} + N_{abc}}{3} \ , \tag{5.87}$$

$$\mathbf{B}_d = \mathrm{sign}(V^0) \frac{N_{adb} + N_{acd} + N_{bcd}}{3} \ , \tag{5.88}$$

where each vector is then assembled into the undeformed vertex normal matrix ($\mathbf{B}_{\mathrm{m}}$) for

convenience:

$$\mathbf{B}_{\mathrm{m}} = \begin{bmatrix} \mathbf{B}_b & \mathbf{B}_c & \mathbf{B}_d \end{bmatrix} \ . \tag{5.89}$$

This same matrix ($\mathbf{B}_{\mathrm{m}}$) may also be more directly found using components from (Eq. 5.78, 5.77). This shows that the quantity can be precomputed, however it doesn't give the physical intuition that assembly through weighted normal vectors on tetrahedron faces in (Eq. 5.85 and 5.88) gave. It is written as

$$\mathbf{B}_{\mathrm{m}} = V^0 \mathbf{D}_{\mathrm{m}}{}^{-T} \ . \tag{5.90}$$

The tetrahedron's internal force contribution matrix ($\mathbf{G}$) is then assembled by taking the Piola-Kirchhoff stress ($\mathbf{P}$) against the undeformed weighted vertex normal matrix ($\mathbf{B}_{\mathrm{m}}$) as

$$\mathbf{G} = \mathbf{P}\mathbf{B} \ , \tag{5.91}$$

where internal force contributions at three of four vertices are contained within as column vectors:

$$\begin{bmatrix} \mathbf{f}_b & \mathbf{f}_c & \mathbf{f}_d \end{bmatrix} = \mathbf{G} \ . \tag{5.92}$$

The internal force contribution on the remaining vertex (the arbitrary vertex initially chosen for Eq. 5.79 5.77) is calculated through force equilibrium as

$$\mathbf{f}_a = -\mathbf{f}_b - \mathbf{f}_c - \mathbf{f}_d \ . \tag{5.93}$$

Moving back to MPM, the total internal force contribution given to a particle (p) by all incident tetrahedrons (e) is a summation over element contributions:

$$\mathbf{f}_p = \sum_e \mathbf{f}_e \ , \tag{5.94}$$

which assumes the particle occupies the same spatial coordinate as the vertices so that no shape-function is needed, as done in our implementation. Particle internal force ($\mathbf{f}_p$) can then transfer from particle-to-grid (P2G) to find the grid-node internal force contribution using any desired advection scheme (Sec. 5.6 and 5.6.7). Total internal force on a grid-node is taken as a summation over particles in the nodal support as in standard MPM:

$$\mathbf{f}_i = \sum_p N_{ip}\mathbf{f}_p \ . \tag{5.95}$$

This completes the basic MPM-FEM coupling. The formulation also lends itself to determining vertex normals (useful for contact algorithms) during the simulation. A deformed weighted vertex normal matrix ($\mathbf{B}_\mathrm{s}$) can be found using the precomputed undeformed, weighted vertex normal matrix ($\mathbf{B}_\mathrm{m}$). Determination of the deformed normal vectors requires an evaluation of the inverse deformation gradient. This isn't a massive computational cost but it is not negligible. It is calculated as

$$\mathbf{B}_\mathrm{s} = J\mathbf{F}^{-T}\mathbf{B}_\mathrm{m} \ , \tag{5.96}$$

where the matrix consists of columns defining deformed, weighted vertex normal vectors set by

$$\mathbf{B}_\mathrm{s} = \begin{bmatrix} \mathbf{b}_b & \mathbf{b}_c & \mathbf{b}_d \end{bmatrix} \ , \tag{5.97}$$

and the remaining arbitrary datum vertex is found through equilibrium once more as

$$\mathbf{b}_a = -\mathbf{b}_b - \mathbf{b}_c - \mathbf{b}_d \ . \tag{5.98}$$

Because meshed particles can represent vertices in multiple elements, the weighted normal of a meshed particle ($\mathbf{b}_p$) is taken as the sum over elements to find the vertex normal on the MPM

particle p. Compute this summation over all incident elements e as

$$\mathbf{b}_p = \sum_e \mathbf{b}_e \ . \tag{5.99}$$

With the coupled MPM-FEM formulation for a linear tetrahedron element complete, we now give attention to a high-level pipe-line for Multi-GPU implementation (Chp. 4). The pipeline is as follows:

1. Grid Update - Boundary conditions and external forces are applied to grid-nodes.

2. Grid-to-Particle (G2P) - Reconstruct velocity and affine velocity gradient matrix on meshed particles. Advect particles using any desired advection scheme.

3. Particle-to-Vertice (P2V) - Write new particle positions to their vertex counterpart. May also write velocity or other quantities if desired.

4. Vertices-to-Element-to-Vertices (V2E2V) - Launch kernel for elements. Each element contain a precomputed, undeformed edge matrix $\mathbf{B}_\mathrm{m}$ and undeformed volume V^0. Positions of the four vertices are read and the element updates deformation and stress at the Gauss point. Lagrangian internal force contribution is then atomically added back onto the vertices.

5. Vertice-to-Particle-to-Grid (V2P) - Summed internal forces on vertices are read by the particles.

6. Particle-to-Grid (P2G) - Particles transfer momentum, affine momentum, and Lagrangian internal force to the grid-nodes.

The above MPM-FEM pipeline has been implemented into the Claymore Multi-GPU MLS-MPM software (Wang et al. 2020b). While it is high-performance, the low-level design has room for improvement. For now it performs well and is not observed to slow down simulations, but as a future task the MPM-FEM coupling should be optimized.

Our approach to MPM-FEM coupling is very minimalist yet it interfaces with modern techniques like MLS-MPM and ASFLIP. Further, it has been implemented into an open-source, Multi-GPU MPM software (Wang et al. 2020b) for public use– Making this the first open-source Multi-GPU MPM-FEM code to our knowledge.

5.14 *Final Remarks*

In this section the needs of fluid-driven debris-field hazard simulations are addressed by a novel numerical approach designed by this lab. Points of implementation, such as material models and time integration are discussed. Alternative advection schemes are described in the context of improving damping and contact. MLS-MPM is coupled with FEM in an open-source Multi-GPU code. Novel modifications are made to an F-Bar volumetric antilocking scheme and a novel, high-performance material model is formulated and presented.

Chapter 6

VALIDATING HIGH-PERFORMANCE MULTI-GPU MATERIAL POINT METHOD FOR DEBRIS-FLUID-STRUCTURE INTERACTION

Development of numerical software requires proper verification. The aim of this chapter is to showcase the efficacy of our numerical approach by presenting a series of debris-fluid-structure interaction (DFSI) validation examples. Each validation case in this study focuses on a specific and relevant DFSI phenomenon. Our approach replicates each at massive computational scales (100,000 - 1,000,000,000 particles simulated within hours). All simulations in this dissertation use mixed-precision, i.e. particles and grid-nodes are in double and single-precision respectively, excluding those in Section 6.6 and an example case from Section 6.4 which is a foil for its improved mixed-precision twin. Note that the original Claymore by Wang et al. 2020b is a single-precision graphics code, all mixed and double-precision engineering functionality was implemented into ClaymoreUW by our group. Benchmark results in ClaymoreUW are compared against analytical equations, experimental observation, and alternative numerical methods.

6.1 *Validation Overview*

DFSI is a complex problem, for which few clean validation cases exists. Given the context, DFSI validation across benchmarks of varying complexity is taken on the following subset interactions. These include:

- **Debris-Structure Interaction (DSI).** One validation case is performed for DSI. In-air debris of varied densities impact a rigid structure at a prescribed velocity. Forces on the structure are measured and compared to a real-world design equation with agreement in

magnitude and duration.

- **Fluid-Structure Interaction (FSI).** Due to its relevance, four validation cases are performed for FSI. **(i)** The first is a gravity-loaded hydrostatic tank of water that checks for water volume maintenance and accurate tank wall reaction forces. **(ii)** The second is a sloshing tank of water checked for accurate first-mode sloshing frequency. **(iii)** The third is an elastic fixed-fixed beam loaded by initially uniform fluid, which redistributes with beam deflection. Beam center-span deflection is tracked and compared to other numerical tools and analytical predictions. **(iv)** The fourth is an elastic column impacted by a water dam break. Column displacement is tracked. FSI results are within bounds produced by multiple publications using alternative numerical methods.

- **Debris-Fluid Interaction (DFI).** One validation case is performed for DFI. A piece of debris is placed in a tank of water. We check for proper sinking/buoyancy of the debris under gravity loading. DFI substantially improves, relative to analytical models, when MLS-MPM is combined with either FLIP-PIC mixing or ASFLIP as they alleviate fluid damping and fluid-debris contact, especially at low grid resolutions.

- **Debris-Debris Interaction (DDI).** One validation case is performed for DDI. A line of uniformly spaced dominoes is perturbed on one end. Toppling propagates through the domino ensemble until all have collapsed. Simulations are checked for allowing accurate collision, sliding, and separation behavior between dominoes and the steady-state domino wave-speed is compared against analytical, experimental, and FEA results. Our DFSI approach improves DDI significantly over standard MPM and is in agreement with FEA studies performed in ABAQUS.

See Tables 6.1, 6.2, and 6.3 for simulation and material properties. Each validation case is

Table 6.1. Benchmark Solid Material Settings

Benchmark Name	ρ_s [kg/m^3]	E_s [Pa]	ν_s [-]	FLIP [Ratio]	ASFLIP [Ratio]	FBAR [Ratio]
DSI: Debris Impact	500	1e6	0	0	0	1
-	1000	1e6	0	0	0	1
-	1500	1e6	0	0	0	1
-	2000	1e6	0	0	0	1
FSI: Hydrostatic Tank	-	-	-	-	-	-
FSI: Sloshing Tank	-	-	-	-	-	-
FSI: Fluid On Beam	100	Varies	0	Varies	0	0
FSI: Dam-Break Column	1000	1e6	0.3	0.8	0	0
DFI: Debris Buoyancy	750	1e7	0.4	0	0	0
-	1500	1e7	0.4	0	0	0
-	3000	1e7	0.4	0	0	0
DDI: Domino Toppling	1400	2e9	0.3	0.999	0.025	0

enumerated below.

6.2 *Debris-Structure Interaction*: *Elastic Debris Impacting a Rigid Structure*

Debris-structure interaction (DSI) is essential to debris-fluid-structure interaction (DFSI). Arbitrary debris, in both material and geometry, colliding with a structure must produce the appropriate impact force and duration, as well as the dynamic response in the debris. An example of the behavior of varied debris geometries in our DFSI approach can be seen, qualitatively, to exhibit nuanced dynamics in Figure 6.1. Here we investigate two debris types that have been used at two real-world wave-flume experiments: Oregon State University Large-Wave-Flume (OSU LWF) and the University of Washington's Harris Flume (UW NASIRF). The former is large (0.5 x 0.1 x 0.05 m) and the latter is small (0.1016 x 0.0254 x 0.0254 m) debris but both are made of the same material (HDPE, ρ = 1000 kg/m^3). The OSU LWF debris will be used to evaluate impact dynamics as density is artificially varied. The UW NASIRF debris will be used to evaluate how well MLS-MPM + FEM (Sec. 5.13) can resolve stiff debris-impacts at low-resolutions as damping

Table 6.2. Benchmark Fluid Material Settings

Benchmark Name	ρ_w [kg/m^3]	k_w [Pa]	μ_w [Poise]	FLIP [Ratio]	ASFLIP [Ratio]	FBAR [Ratio]
DSI: Debris Impact	-	-	-	-	-	-
FSI: Hydrostatic Tank	1000	2.2e9	0.001	0	0	1
-	1000	2.2e8	0.001	0	0	1
FSI: Sloshing Tank	1000	2.2e9	0.001	0.25	0	0.25
FSI: Fluid On Beam	1000	5.0e7	0.001	0	0	0
FSI: Dam-Break Column	1000	2.2e9	0.001	0.8	0	0
DFI: Debris Buoyancy	1000	2.2e9	0.001	0.3	0	0.99
DDI: Domino Toppling	-	-	-	-	-	-

Table 6.3. Benchmark General Simulation Settings

Benchmark Name	Δx [m]	PPC [#]	Particles [#]	Duration [sec]	CFL [Ratio]
DSI: Debris Impact	0.01	27	67,500	0.06	0.5
FSI: Hydrostatic Tank	0.05	8	262,144	3	0.3
-	0.0125	8	16,777,216	3	0.3
-	0.0032	8	1,000,000,000	1	0.5
FSI: Sloshing Tank	0.025	8	2,097,152	3	0.3
FSI: Fluid On Beam	0.025	8	768,000	20	0.4
FSI: Dam-Break Column	0.001	8	4,184,832	0.5	0.3
DFI: Debris Buoyancy	0.0125	8	33,554,432	3	0.45
DDI: Domino Toppling	0.0005	27	35,831,808	1.5	0.45

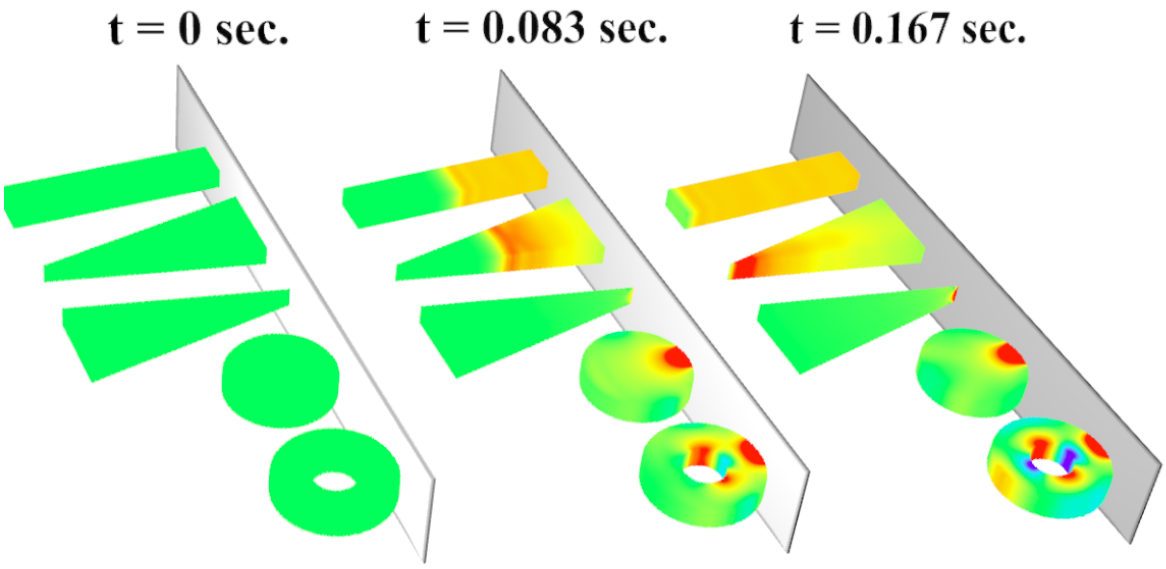

Fig. 6.1. Debris-Structure Interaction: Debris pressure waves simulated in MLS-MPM for varied geometries. Pressures visualized for varied MLS-MPM debris geometries (bar, wedges, disc, and ring) during impact against a rigid wall. Magnitudes and propagation sped of the pressure wave along debris geometries match expectations for pressure waves in corresponding shapes.

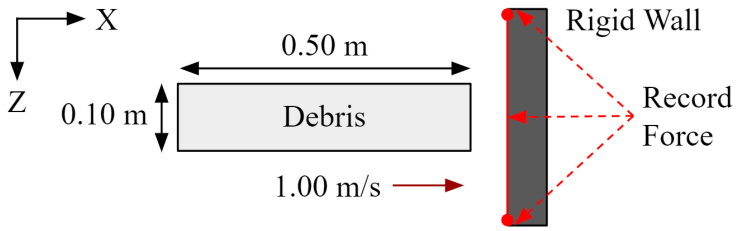

Fig. 6.2. Debris-Structure Interaction: Diagram of debris impact validation case. Schematic for a rectangular prism debris (0.5 x 0.05 x 0.1 meters) impacting a rigid wall at 1.0 m/s. Reaction forces of the wall are measured through time. This case is used to validate basic DSI for debris densities of 500, 1000, 1500, and 2000 kg/m^3.

is varied. This is enacted with FLIP-PIC mixing's α parameter (see Sec. 5.6.3). This is to check if a nonlinear, damped impact model improves predictions of force-time series over an undamped analytical model for our damped numerical impacts.

To begin DSI validation, we examine a simple rectangular prism debris impacting on a rigid structure and compare against an established methodology for quantifying debris impact. The plan view schematic that illustrates this setup is shown in Figure 6.2. Debris dimensions (0.5 x 0.1 x 0.05 m) and velocity (1 m/s) replicate real-world debris used in hundreds of wave-flume debris impact experiments conducted at Oregon State University's Large Wave Flume (OSU LWF) in Mascarenas 2022.

6.2.1 Undamped Debris Impact

Some structural design guidelines suggest a simple debris impact model as characterized in Aghl et al. 2014b. Simplified from ASCE 7-22 Chapter 6 Section 6.11.2, the model assumes an elastic collision with debris taken to be 1D bars. Stiffness of debris (k), impact duration (t), and max force of impact (f) are found as

$$k = \frac{EA}{L} \quad \text{and} \quad t = 2\sqrt{\frac{m}{k}} \quad \text{and} \quad f = v\sqrt{mk}\ , \tag{6.1}$$

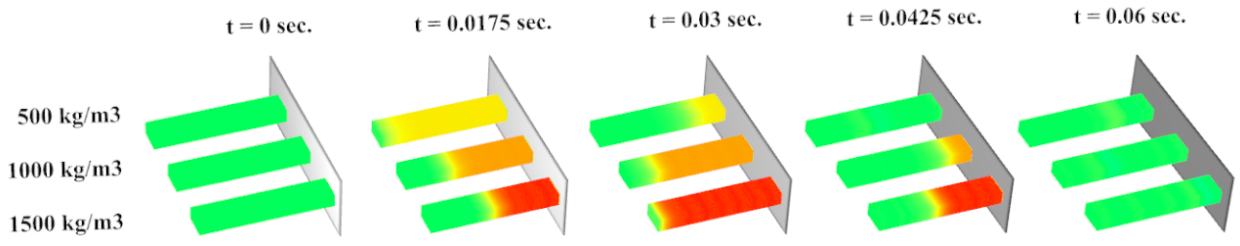

Fig. 6.3. Debris-Structure Interaction: Debris pressure waves simulated in MLS-MPM for varied densities. Pressures visualized for varied MLS-MPM debris densities during impact against a rigid wall. Magnitude and propagation of the pressure wave along debris fits expectations for pressure waves in rectangular prism elastic bodies.

where E is the debris' Young's modulus, A is area of impact, L is length of debris, v is velocity of impact, and m is mass of debris. Volume for rectangular prisms is $V = LWH$ and mass is $m = V\rho$. Equation 10.1 may be phrased more geometrically as

$$f = v\sqrt{\rho EWHA} = vA\sqrt{\rho E} \ . \tag{6.2}$$

where it should be noted that length (L) does not affect force of impact, whereas area of impact (A) does. Root dependency is also seen for density (ρ). Equations 10.1 and 6.2 present a simplistic step-function force-time impact profile, but it is reasonable for rectangular, elastic debris in head-on collisions. Calculated force-time profiles do not extrapolate to nonlinear materials, varied debris geometries, or oblique impacts, all of which our numerical approach is suitable for.

In our MLS-MPM approach we recreate the DSI case in Figure 6.2 to evaluate impact dynamics as debris density is changed (500, 1000, 1500, and 2000 kg/m^3). Debris pressures at select times are shown in Figure 6.3, demonstrating the effect of debris density on pressure wave speed and magnitude. Force-time series at the rigid structure from our approach are shown in Figure 6.4 along with the analytical model in Equation 10.1.

Our MLS-MPM results for the numerical OSU LWF debris impacts agree with a simple analytical

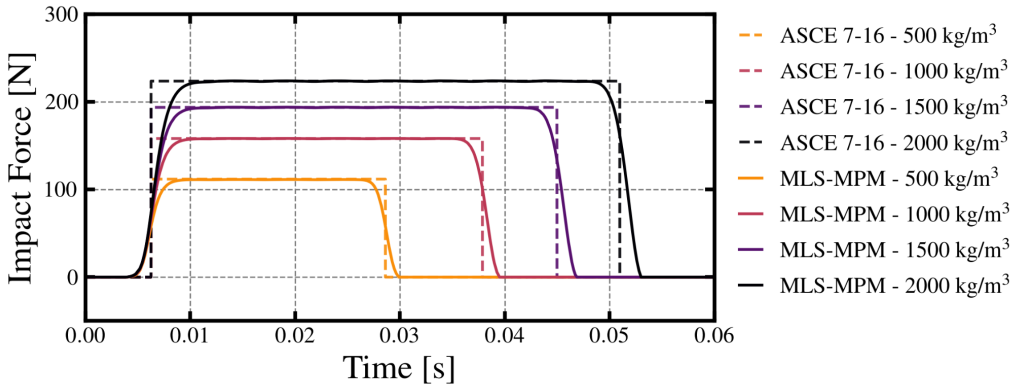

Fig. 6.4. Debris-Structure Interaction: MLS-MPM debris impact forces on a rigid wall. Debris density is varied. ASCE-7 guideline for single-debris impacts plotted as comparison. MLS-MPM closely matches impact max force and duration. Differences in lead-in/lead-out forces are caused by coarse grid-cell size and wide shape-functions.

model of bar debris impact used in practice by structural engineers. Both impact force magnitude and duration match closely across cases, with additional pressure-wave dynamic behavior captured.

6.2.2 Damped Debris Impact

The model in Equations 10.1 and 6.2 is very simplistic. It produces a step-function force-time profile for all impacts. It reasonably approximates rectangular, elastic debris in head-on collision (Aghl et al. 2014a). However, wood logs, steel bars, and shipping containers have, respectively, been recorded to produce half-sine, step-function, and trapezoidal force-time series on structures. To describe these we investigate alternative equations.

For instance, we can construct our own dynamic impact model to introduce a damping-inclusive perspective, e.g. dynamic debris impact forces can be phrased as a single degree-of-freedom system dependent on damping and stiffness parameters. Many approaches are viable (e.g. Maxwell, Bingham, Kelvin-Voigt) (Polukoshko et al. 2007). A Kelvin-Voight form is a simple

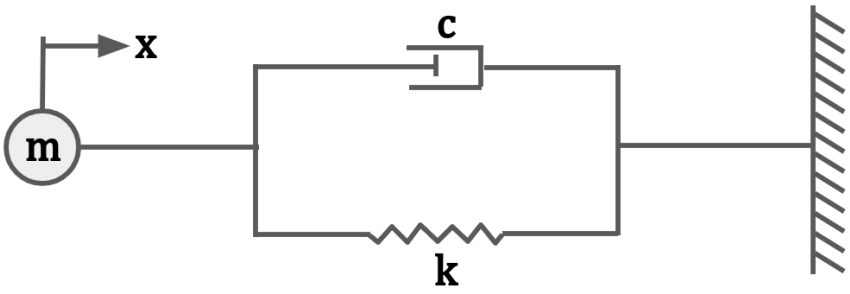

Fig. 6.5. Kelvin-Voigt Rhealogical Model. Visualization of linear Kelvin-Voight rhealogical model for debris impact.

approach to collision. This is visualized in Figure 6.5.

The Kelvin-Voight model idealizes a point mass connected to a rigid body where connection is composed of a linear spring (stiffness k, i.e. imposes force when the mass is displaced) and dash-pot (damping c, imposes force proportional to the mass' velocity) in parallel. An equation of motion (EOM) is constructed for the system as

$$m\ddot{x} + c\dot{x} + kx = 0 \ , \tag{6.3}$$

which we may rephrase to use mass specific factors with

$$\ddot{x} + 2\zeta\omega_0\dot{x} + \omega_0^2 x = 0 \quad \text{where} \quad \omega_0 = \sqrt{\frac{k}{m}}, \quad \zeta = \frac{c}{2\sqrt{km}} \ , \tag{6.4}$$

where ω_0 is the natural frequency. We may also define a linear damping ratio $\beta = \frac{cm}{2\omega_0}$.

The model in Equation 6.4 is still limited. Tensile forces develop when debris bounce off of a structure, something not seen in reality unless there is cohesion. Further, damping does not increase with mass displacement (i.e. deformation), which is needed for some elasto-plastic debris types. An easy modification can be made to the Kelvin-Voight model of Equation 6.4 to address

these issues and make it nonlinear. Simply alter the EOM to

$$m\ddot{x} + c\dot{x}x + kx = 0 \quad \text{or} \quad m\ddot{x} + x(c\dot{x} + k) = 0 \ , \tag{6.5}$$

which is an EOM that may be solved numerically with an explicit time integration method while including mass-displacement damping and no erroneous tensile forces. Acceleration of the debris ($a_d = \ddot{x}_d$) at a time-step (t^{n+1}) is taken as the total force acting on the debris divided by its mass, relative to a previous time-step (t^n) with

$$\ddot{x}_d^{n+1} = \frac{1}{m}\left(-kx_d^n - cx_d^n\dot{x}_d^n\right)\gamma \ , \tag{6.6}$$

where $\gamma = 1$ if $x \geq 0$ and $\gamma = 0$ if $x < 0$ to define contact of the debris with the rigid structure. The debris velocity ($v_d = \dot{x}_d$) is updated in time using the constant acceleration found in Equation 6.14 over a discrete time-interval (Δt) as

$$\dot{x}_d^{n+1} = \ddot{x}_d^{n+1}\Delta t + \dot{x}_d^n \ , \tag{6.7}$$

and debris elevation (x_d) is then updated according to

$$x_d^{n+1} = \dot{x}_d^{n+1}\Delta t + x_d^n \ . \tag{6.8}$$

Figure 6.6 compares the simplified ASCE 7 bar-impact methodology of equation 10.1 against a numerical solution to our nonlinear Kelvin-Voight model in 6.5. The nonlinear Kelvin-Voight model is computed for a set of damping ratios, where a damping ratio of 0 makes the model elastic. Note that Equation 10.1 from Aghl et al. 2014b predicts a step-function shaped impact in the force-time plot, while elastic Kelvin-Voight predicts a half-sine. Both models, in undamped form, predict the same max force. As damping ratio increases in the nonlinear Kelvin-Voight

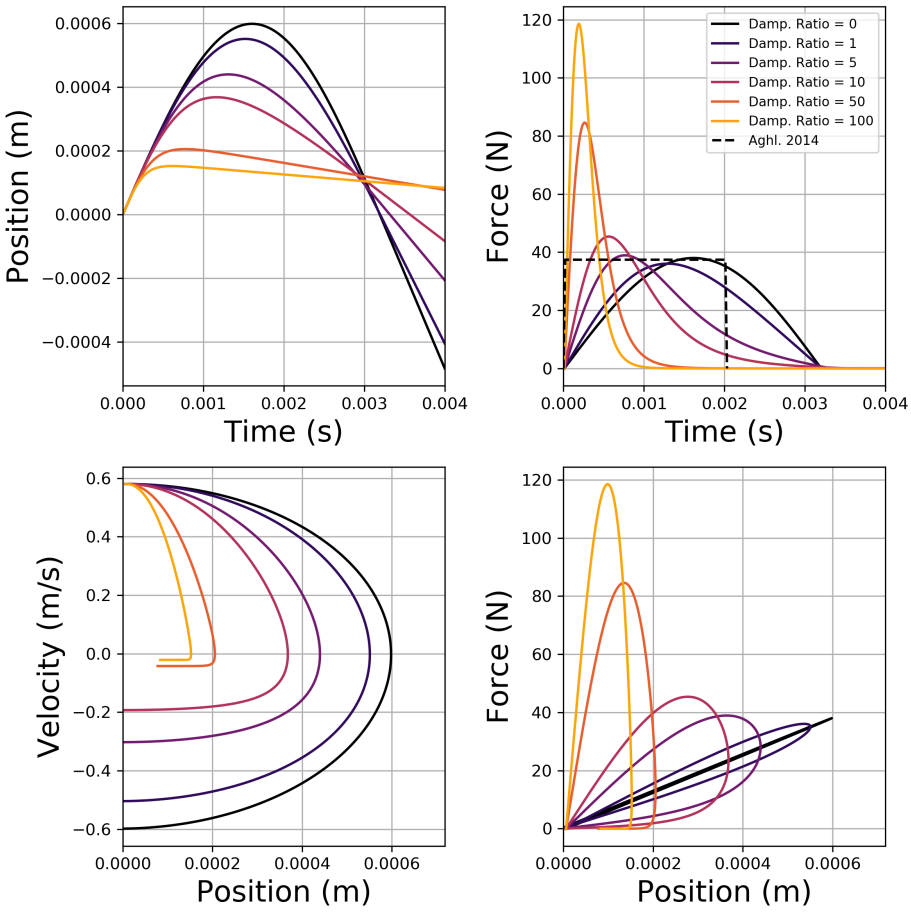

Fig. 6.6. Debris Impact Model as Nonlinear Kelvin-Voigt vs. ASCE 7-16 - UW WASIRF Debris. Visualization of nonlinear Kelvin-Voight constitutive model for debris impact. OSU LWF debris modeled with dimensions 0.1016 x 0.0254 x 0.0254 m. Density 1000 kg/m^3. Young's Modulus 1e7 Pa. Initial velocity is 0.58 m/s.

model, however, the force-time plot begins to resemble an asymetric shock-wave, reaching a higher max force but for a shorter duration. This is an important effect to understand for real-world debris, which vary in material property regarding damping and elasto-plasticity.

While the initial case for the 0.5 x 0.1 x 0.05 m debris was based on the OSU LWF debris tests of Shekhar et al. 2020 and Mascarenas 2022, we may evaluate an alternative experimental set to attempt to replicate discrepancies in analytical models (Figure 6.6) with MPM simulations. We

move on to the UW NASIRF debris Type B (0.1024 x 0.0254 x 0.0254 m, HDPE plastic) from the University of Washington wave-flume experiments by Nicolette Lewis. Initial velocity is 0.58 m/s. Debris is placed 0.01 m from the structural face. There is no impact obliquity (i.e. head-on collision). No gravity is present. Density of the debris is 1000 kg/m3. Young's modulus and Poisson's Ratio is 1e7 Pa and 0.4 respectively. Particles-per-cell is 2 on a grid spacing (Δx) of 0.3175 cm. A PIC-FLIP mixing factor, α (Section 5.5), iterates through a set of values (0, 0.5, 0.9, 0.99, and 0.999) to demonstrate its effect on damping. Recall that PIC-FLIP advection becomes less damped as $\alpha \in [0, 1]$ moves towards 1, at which point it reduces to pure FLIP advection. Results are shown in Figure 6.7.

6.2.3 Remarks

For large rectangular prism debris modeled as MLS-MPM fixed-corotated solids (Sec. 5.4) in head-on, in-air collisions we see good agreement between our MLS-MPM implementation and ASCE 7 guidelines (Aghl et al. 2014a) as debris density is varied. Figure 6.4 shows force-time impact results which match closely with noted oscillatory (i.e. dynamic) behavior in the MPM simulations due to modeling the actual pressure wave traveling through the debris.

For small rectangular prism debris modeled as MPM-FEM (Sec. 5.13) fixed-corotated solids in head-on, in-air collisions we see weaker agreement with an analytical model (nonlinear Kelvin-Voight). Figure 6.7 shows force-time impact results. Low-resolution of simulations (2 PPC) and the stiffness of debris were primary limitations, as well as use of large shape-functions relative to grid-size slowing-down debris before impact. Results improve as FLIP/PIC mixing (Sec. 5.6) is increased, demonstrating that energy damped out by the low-resolution MPM grid can be reintroduced by modifying advection.

The two different modes of MLS-MPM debris impact behavior for different sized rectangular prism debris (step-function and half-sine) suggest that using just the simplified ASCE 7 guideline,

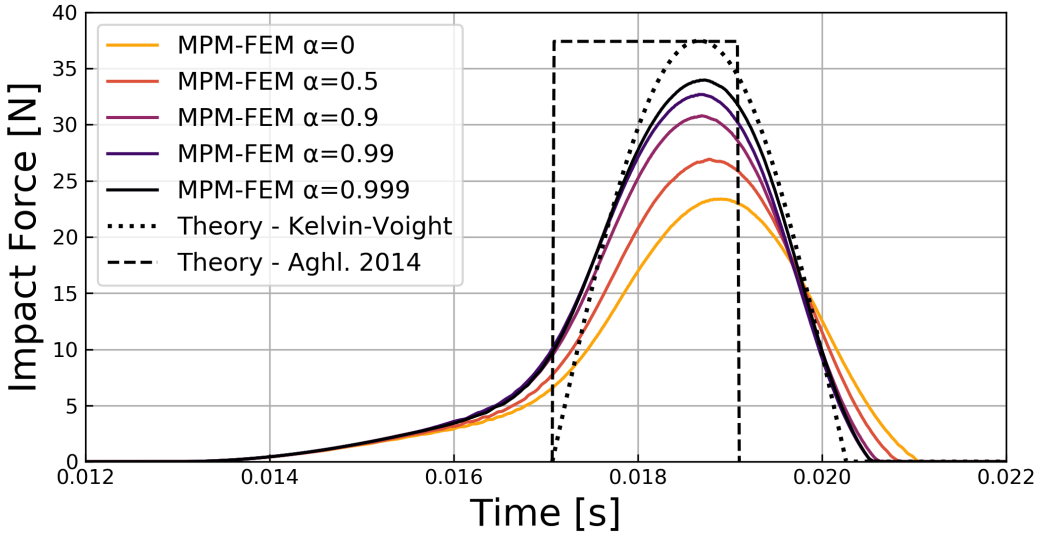

Fig. 6.7. Debris Impact Validation - MPM-FEM vs. Theory for Small, Stiff Debris. 0.1016 x 0.0254 x 0.0254 m debris (UW WASIRF) impacting a rigid wall. Density 1000 kg/m3, Young's modulus 10 MPa, and Poisson's Ratio is 0.4. PPC is 2 and grid-spacing is 0.3175 cm. Initial velocity of debris is 0.58 m/s. Time-step set by CFL condition. Model in ASCE 7-16 guidelines (Aghl et al. 2014b) plotted as comparison, as well as a nonlinear Kelvin-Voight model with no damping. Note that simulations for this short, stiff debris resemble the half-sine profile of the Kelvin-Voight model over the step-function, suggesting that the former is better in predicting debris which give half-sine profiles when undamped and asymmetric shock profiles when damped. Note that simulations used an earlier state of the ClaymoeUW software that only allowed single-precision computation, so results may be greatly refined if replicated.

which only consider a step-function, may not be adequate for real-world debris impact design. The difference likely arises due to the pressure wave speed and wave-length (dynamic properties) traveling through the debris at impact relative to debris length (geometric property). The simulated UW WASIRF debris is short and stiff so a half-sine profile appeared, whereas the simulated OSU LWF debris was long and not very stiff and thus produced a step-function profile. More complex debris will feature further considerations, of which there is not currently an adequate means to approach.

6.3 *Fluid-Structure Interaction*: *Validation Overview*

Given the significance within the scope of this study, we assess the feasibility of our approach in fluid-structure interaction (FSI) using four scenarios of increasing complexity, namely:

1. **Hydrostatic Fluid-Tank Interaction.** This includes evaluation of a level fluid in a rigid tank– A very basic case that must behave well over millions of time-steps and particles. The scope is to check for volume maintenance, accurate tank wall reactions, and appropriate fluid pressures relative to fluid properties (bulk modulus) and numerical method (with/without anti-locking).

2. **Sloshing Fluid-Tank Interaction.** This includes an initially slanted water mass sloshing in a rigid tank– A simple case with dynamic fluid behavior. The scope is to check for an appropriate sloshing frequency via surface elevations and wall forces, comparing to first-mode analytical sloshing solutions.

3. **Dynamic Fluid-Beam Interaction.** This includes a layer of fluid loading a fixed-fixed beam of variable stiffness– A complex case of a redistributing, nonlinear fluid continuously loading a structural element, producing both small and large deformations. The scope is to

compare the center-span deflection to simplified analytical equations and alternative numerical approaches, as well as transition of linear to nonlinear structural deformation.

4. **Dynamic Fluid-Column Interaction.** This includes a water dam-break impacting an elastic column– A complex dynamic case without a closed form solution relevant to chaotic DFSI problems. The scope is to check column horizontal tip displacements and compare to multiple alternative numerical approaches.

6.4 *Fluid-Structure Interaction*: *Hydrostatic Convergence in a Rigid Rectangular Tank*

Arguably the most basic fluid-structure interaction (FSI) case is still water in a rigid, rectangular-prism tank structure loaded suddenly by gravity. This case is not trivial, as any numerical method at massive scales may undergo millions of time-steps across billions of weakly-compressible numerical bodies, compounding errors beyond expectations. Reaction forces on the tank's walls and converged surface elevations should match analytical predictions closely. Further, results should change predictably with bulk modulus (i.e. fluid resistance to compression) as it is commonly lowered by practitioners to increase time-steps but rarely quantified in its side-effects.

Analytically, fluid pressure (P_{H}) and the total force on the right wall of a rigid rectangular tank (f_{H}) for ideal hydrostatic conditions are:

$$P_{\mathrm{H}} = g\rho H \quad \text{and} \quad f_{\mathrm{H}} = \frac{1}{2} g\rho H^2 W \ , \tag{6.9}$$

which we may solve by setting gravity g (-10 m/s^2), water density ρ (1,000 kg/m^3), water height H (3.2 m), and wall width W (0.4 m) as specified to receive a hydrostatic wall force of 20,480 Newtons. A hydrostatic validation case is depicted in Figure 6.8 for a 3D rectangular tank (3.2 x 6.4 x 0.4 m) and a water mass (3.2 x 3.2 x 0.4 m). Gravity is applied at t = 0 seconds, but no initial

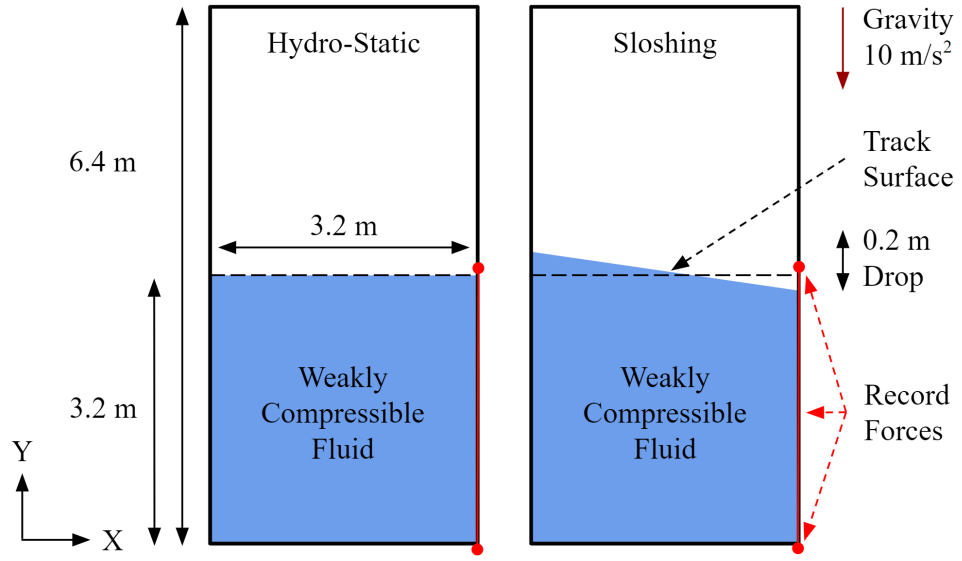

Fig. 6.8. Fluid-Structure Interaction: Diagrams of hydrostatic and sloshing water in a rigid-tank cases. (Left) Hydrostatic case. (Right) Sloshing case. Separable velocity condition boundaries.

pressure field is set. See tables 6.3 and 6.1 for simulation and material properties, and refer to Yang 2016 for this case in MPM with different anti-locking approaches. Here we investigate the influence of various MPM grid-cell lengths (Δx = 0.05, 0.0125, and 0.0032 meters), single versus mixed-precision implementation, and fluid compression ratios of 1/100, 1/10, 1 relative to water (i.e. bulk modulus = 2.2e7, 2.2e8, and 2.2e9 Pascals).

To start, we set grid-cell length, Δx, to 0.05 m to check initial pressure wave behavior, and then we will refine to 0.0125 m to study free-surface deviation, force convergence, and the effect of computational precision. Figure 6.9 visualizes the pressure on fluid particles and the force vectors on the tanks' rigid boundary grid-nodes. Figure 6.9 (a) depicts the initial pressure wave as it begins to reflect off the tank bottom, along with forces exerted on the rigid tank walls as vectors on MPM grid-nodes. Simulated reactions visibly approximate the analytical profile, not just the final summed force. Figure 6.9 (b) contrasts the pressure fields at t = 3 seconds for single vs. mixed

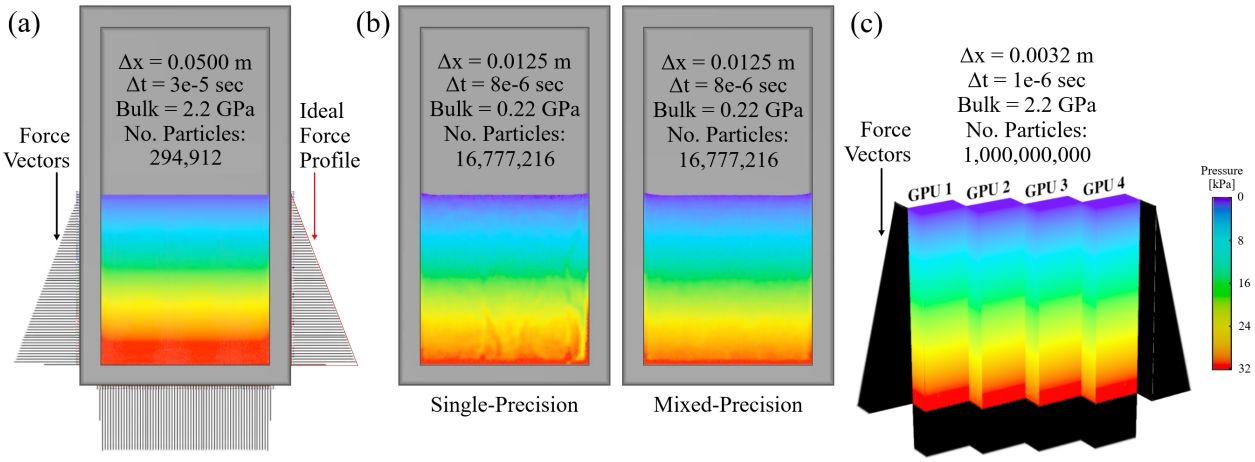

Fig. 6.9. Fluid-Structure Interaction: Hydrostatic tank's fluid pressure and wall forces across varied MLS-MPM simulations. (a) Initial pressure wave after sudden gravity loading reflects off bottom of tank. Boundary forces are instantaneously near the hydrostatic ideal. **(b)** Simulations in single and mixed-precision respectively undergo and avoid pressure errors after 3 seconds. **(c)** Exascale simulation (1,000,000,000 particles over 1,000,000 time-steps) of hydrostatic convergence. One node with four GPUs on Texas A&M's ACES used for 23.5 hours.

precision floating point computation. Figure 6.10 plot percent error time convergence of MLS-MPM wall forces relative to Eq. 6.9, i.e. $e_{\mathrm{H}} = \frac{f_{\mathrm{MPM}} - f_{\mathrm{H}}}{f_{\mathrm{H}}}$, for varied bulk modulus. We claim convergence at time t_n if $e_{\mathrm{H}} < 0.1\%$ for all times t_{n+m} where m is any positive integer. This accounts for transient simulation dynamics while comparing against hydrostatic equation 6.9 on short time scales. Figure 6.11 shows free-surface elevation at t = 3 seconds for a set of bulk moduli.

The 2.2e9 Pa bulk modulus case has forces that converge by t = 0.2 seconds, i.e. percent error remains below 0.1%, and the surface is nearly unperturbed. The 2.2e8 Pa bulk modulus case ($1/10^{\mathrm{th}}$ of water's) converges more slowly (t = 1.7 seconds) and the surface has small, but more notable deviations. The 2.2e7 Pa bulk modulus case ($1/100^{\mathrm{th}}$ of water's) does not converge in the simulation time-frame due to dynamic oscillations, though error does decay substantially, and the surface has notably changed from the incompressible baseline, signifying the expected

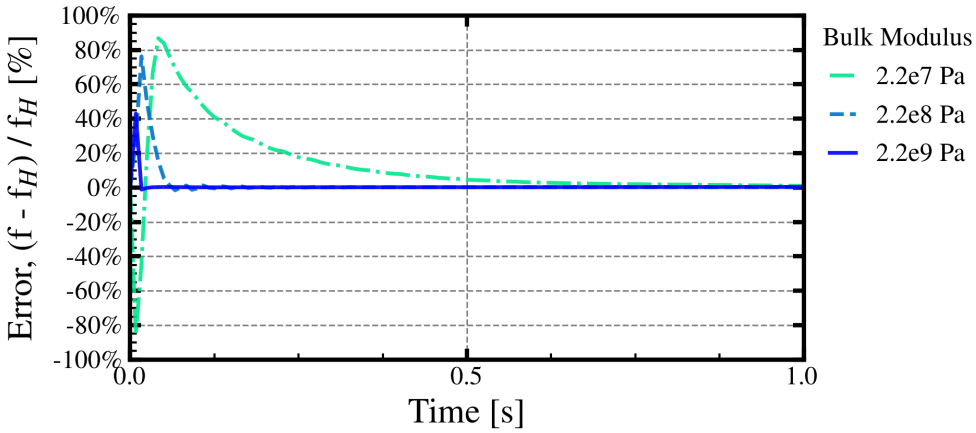

Fig. 6.10. Fluid-Structure Interaction: Hydrostatic tank wall loads in MLS-MPM. Hydrostatic mass of water is loaded suddenly by gravity for three seconds. Reaction forces on the rigid right wall are summed (f) and then taken relative to analytical hydrostatic force ($f_H = 1/2 g \rho H^2 W$) as a percent error measure.

compression of the fluid.

Confident in performance at typical computational scales, we now recreate the case in MLS-MPM at an exceedingly high resolution as proof of high-performance capability. In Figure 6.9 (c), one billion water particles are simulated for one second using one million time-steps (Δt = 1e-6 seconds) on 125,000,000 grid-cells at a spacing of Δx = 0.0032 meters. One quadrillion total MPM steps are executed in ClaymoreUW using a single node on Texas A&M's ACES system for 23.5 hours. Double-precision operations per MPM step per particle are on the order of 1,000, equating to one quintillion (1,000,000,000,000,000,000) total calculations conservatively. This locates our approach's fluid validation at the doorstep of the exa-scale computational paradigm. Whether implemented in ClaymoreUW or elsewhere, our curated high-performance method is ready for next-generation FSI and DFSI problems.

Our approach is able to simulate the reaction force of weakly-compressible hydrostatic fluid under gravity, producing convergence of wall reaction forces relative to dynamics at both low and high spatial resolutions (262,144 - 1,000,000,000 particles, 0.05 - 0.0032 meter grid-cells) that span

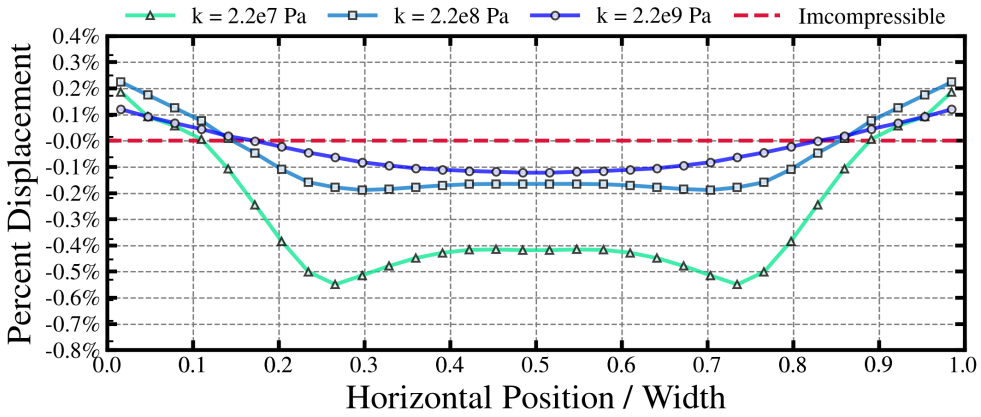

Fig. 6.11. Fluid-Structure Interaction: Free-surface of hydrostatic tank in MLS-MPM simulations. Rigid tank filled by still water is loaded by gravity. Simulations are ran for bulk moduli of 2.2e9, 2.2e8, and 2.2e7 Pa. Free-surface level readings normalized by initial still water level are compared at t = 3.0 seconds. Greater free-surface deviation is seen to correlate with a lower bulk modulus.

nearly real-time performance to the borders of exa-scale computation. The stability at very high resolutions is noteworthy as the exposure to floating-point misrepresentation, Multi-GPU data-race conditions, and MPM null-space error can become detrimental unless a method is implemented for massively parallel use. Converged surface elevations are appropriate for fluid bulk moduli evaluated and maintain accuracy amidst potential integration errors from particles displacing relative to ideal Gauss point positions. We observe that, for even this simple case, fluid pressure fields beyond a second degrade in MLS-MPM, but the volumetric antilocking F-Bar scheme by Zhao et al. 2023 that we applied was found to be a suitable and computationally expedient solution. Advection with ASFLIP from Fei et al. 2021a is not necessary for hydrostatics. Information we garnered concerning the effect of bulk modulus, floating-point precision, and anti-locking on fluid behavior is kept in mind as we move on.

6.5 *Fluid-Structure Interaction*: *Sloshing Water in a Rigid Rectangular Tank*

For the second FSI validation benchmark the same tank as the hydrostatic case is occupied by an initially slanted water mass, introducing a small hydrodynamic factor for validation. Our approach is tested in its capability to replicate fluid sloshing behavior without volume loss/gain or spurious pressure as time progresses through hundreds of thousands of time-steps.

The schematic is shown in Figure 6.8 (right). The left side of the water mass has an initial elevation of 3.3 m, the right 3.1m (0.2 m linear drop over a distance L of 3.2 m). The rigid tank (3.2 x 6.4 x 0.4 m) has a slip boundary condition. Gravity loads at t = 0 seconds without any initial stress or velocity set.

The water surface should oscillate between the starting slant, a flat surface, and a reflected slant according to the first-mode sloshing frequency ($f_{\text{Slosh.}}$) of the rigid rectangular tank. This was defined in Housner 1963 and Jaiswal et al. 2008 as:

$$f_{\text{Slosh.}} = \frac{1}{2\pi}\sqrt{\frac{3.16g\tanh\left(3.16\frac{H}{L}\right)}{L}} \quad \text{and} \quad T_{\text{Slosh.}} = \frac{1}{f_{\text{Slosh.}}}\,, \tag{6.10}$$

where H is tank height, L is tank width, g is gravity, and tanh refers to the hyperbolic tangent. For our validation case dimensions, the sloshing frequency is expected to be 0.499 Hz, corresponding to a sloshing period ($T_{\text{Slosh.}}$) of 2.003 seconds.

Figure 6.13 shows water surface profiles across the width of the tank, at select times. The displayed surfaces correspond to the first-mode sloshing at regular quarter sloshing periods (n/4 $T_{\text{Slosh.}}$, where n is an integer) and are provided for comparative purposes. These surfaces include the ideal initial slant, level surface, and reflected slant. Back-and-forth sloshing of the numerical surface can be seen to follow the ideal first-mode sloshing trends, with some deviation near the boundaries. The jagged surfaces is due to its discrete representation with MPM points.

Figure 6.14 shows hydrodynamic sloshing forces ($f - f_H$) on the right wall of the tank for the

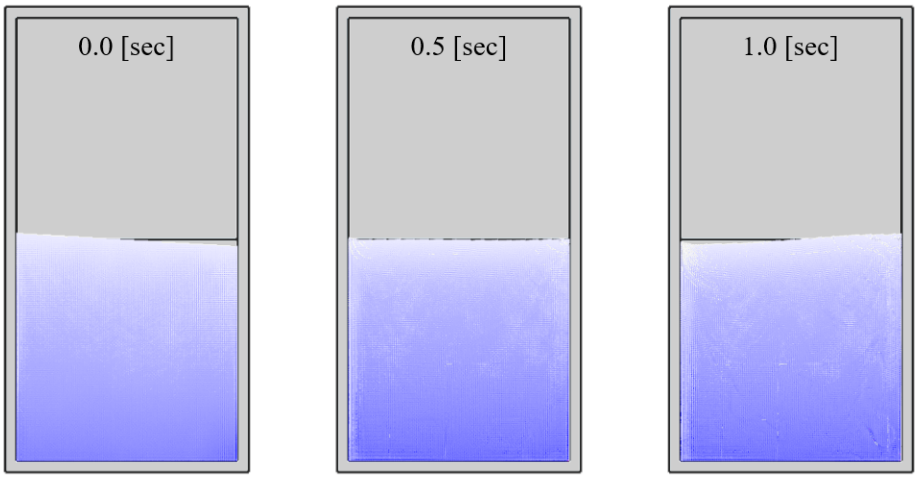

Fig. 6.12. Fluid-Structure Interaction: Sloshing water tank simulated in MLS-MPM. Dynamic interaction of sloshing water with a tank. Surface level reading across tank at select times compared to ideal starting and ending surface. Periodic behavior is inline with analytical sloshing frequency of 2.003 seconds.

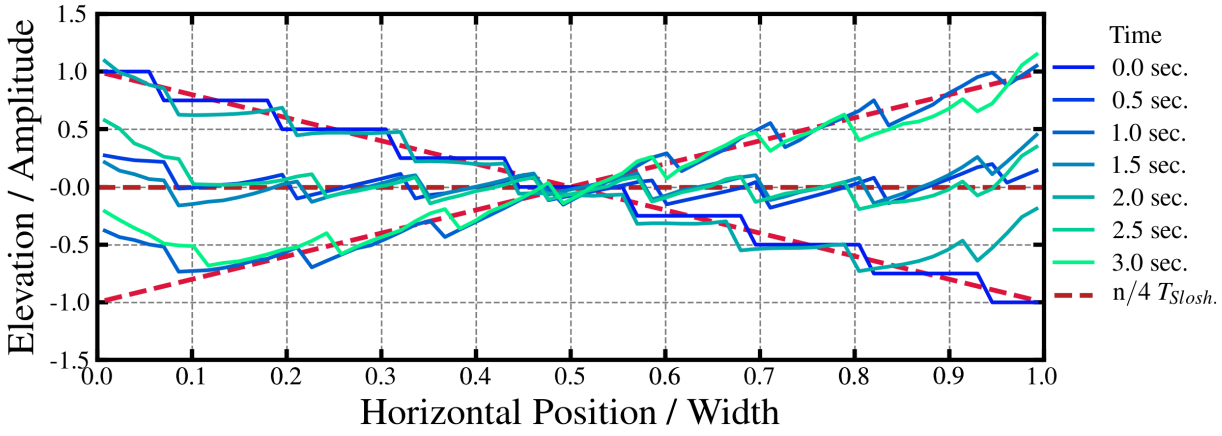

Fig. 6.13. Fluid-Structure Interaction: Sloshing water tank's oscillating surface simulated in MLS-MPM. Free-surface reading across tank width at select times compared to ideal surfaces for first-mode sloshing theory. Surface appears jagged due to the finite resolution of the simulation. Periodic behavior is inline with the analytical sloshing frequency of 2.003 seconds.

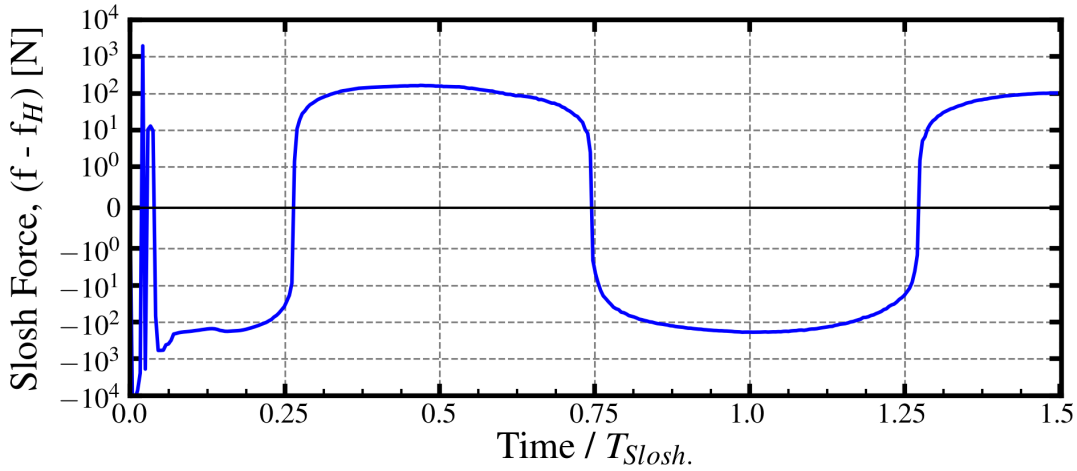

Fig. 6.14. Fluid-Structure Interaction: Sloshing tank's wall load in MLS-MPM. Hydrodynamic force (simulated force minus hydrostatic prediction) of sloshing water tank in MLS-MPM behaves periodically. Simulated sloshing period is observed to be close to the analytical first-mode of 2.003 seconds.

MLS-MPM simulation. It is worth noting the periodic nature of these forces. The minima and maxima occur at regular intervals of half sloshing period, which aligns with the analytical expectation derived from Equation 6.10.

This case demonstrates that MLS-MPM can simulate sloshing of stiff fluids using explicit time integration and a weakly-compressible fluid model, replicating the first-mode analytical sloshing frequency of rectangular tanks. We now move on to more complex fluid-structure interaction validation.

6.6 *Fluid-Structure Interaction*: *Validating Nonlinear Sag of a Fixed-Fixed Beam Loaded by Water*

For the third FSI validation case a fixed-fixed beam is loaded by an initially uniform layer of fluid. A schematic diagram of this problem is depicted in Figure 6.15. As the fluid load acts on the beam, it induces deflections that lead to a nonlinear redistribution of the fluid, resulting in further deflections. This case demonstrates the capability of our MLS-MPM tool and its ease in coupling

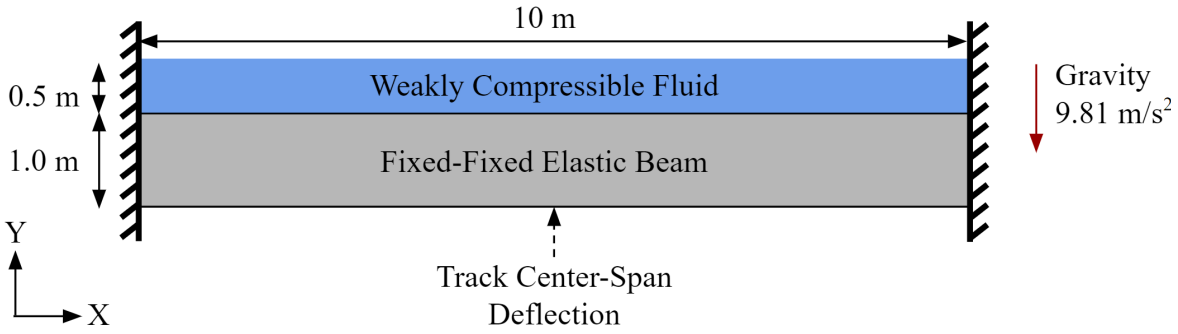

Fig. 6.15. Fluid-Structure Interaction: Schematic of fluid loading an elastic beam validation case. FSI validation case for refined fluid-beam interaction. Concerns a fixed-fixed beam deflecting under a nonlinear redistributing fluid load. Center-span displacement of the beam is tracked through time.

with FEM for sustained, two-way fluid-structure interaction with a focus on refined beam displacements, both small and large.

The beam length, height, and width are 10.0, 1.0, and 0.1 meters, respectively. The water's initial length, height, and width are 10.0, 0.5, and 0.1 meters respectively. The water surface should begin and end at a uniform level across the beam, given enough time, while the beam should begin level but converge to a deflected state. To reduce dynamic effects, gravity is applied quadratically over time, reaching a full value of 9.81 m/s^2 at 10 seconds. This load is then sustained until the benchmark completion at 20 seconds.

The center-span deflection (d_U) of a rectangular prismatic beam under uniform load, using small displacement theory, is characterized by

$$\mathrm{d}_U = \mathrm{d}_{\mathrm{Flexure}} + \mathrm{d}_{\mathrm{Shear}} = \frac{wL^4}{384EI} + \frac{wL^2}{8A_sG} \ , \tag{6.11}$$

where w is the load per length, L is the beam length, E is the beams' Young's modulus, I is the beam's bending moment of inertia, A_s is the strain area, and G is the beam's shear modulus.

MLS-MPM simulation frames for a range of beam stiffness values, i.e. Young's moduli varies

from 2e5 - 5e8 Pa, are visualized in Figure 6.16 to illustrate our tools robustness in stable deflection of beams far beyond ranges allowed by small-deflection models. From left to right we observe that an initially uniform fluid load deflects the beam, the fluid then redistributes due to the deflection, the fluid sloshes as it attempts to reach equilibrium with the nonlinearly deflecting beam, and finally the fluid-beam system approaches a near static solution. Note that a layer of water "sticks" to the beam due to low computational precision and grid resolution. These simulations are from an earlier version of ClaymoreUW when single-precision floating-point arithmetic was still used and before our group had consistent access to modern GPU hardware. Despite past technical limitations, these results are still quantitatively strong.

Figure 6.17 shows how varying the beam's Young's modulus affects center-span deflection vs time in the left plot, and deflection over beam length vs Young's modulus in the right plot (each color representing a different stiffness E). Oscillations are observed in deflections over time as these are dynamic simulations involving quadratic gravity loading (i.e. gravity starts at zero and approaches its full value at a quadratic rate over 10 seconds) on a fluid and elastic beam. These oscillations are proportional to the beam stiffness, as expected. Oscillations could be minimized by applying gravity more slowly. The plot on the right of Figure 6.17 shows good agreement at higher stiffness between MPM and a small-deflection beam equation, but deviation grows as stiffness reduces. This suggests that an underlying analytical assumption is violated for low-stiffness beams modeled in MPM compared to a small-deflection beam equation. The name of the analytical model is a clue, as deflections are not "small" for the low-stiffness cases, but this may be revealed more quantitatively.

Figure 6.18 compares our MPM-FEM results with results obtained using three different approaches: OpenSees (McKenna 2011) with a uniform load (no fluid-redistribution), OpenSees with a one-way / iterative redistributing fluid load, and a two-way coupled simulation of OpenSees and OpenFOAM which will herein be referred to as FOAMySees. OpenSees and FOAMySees

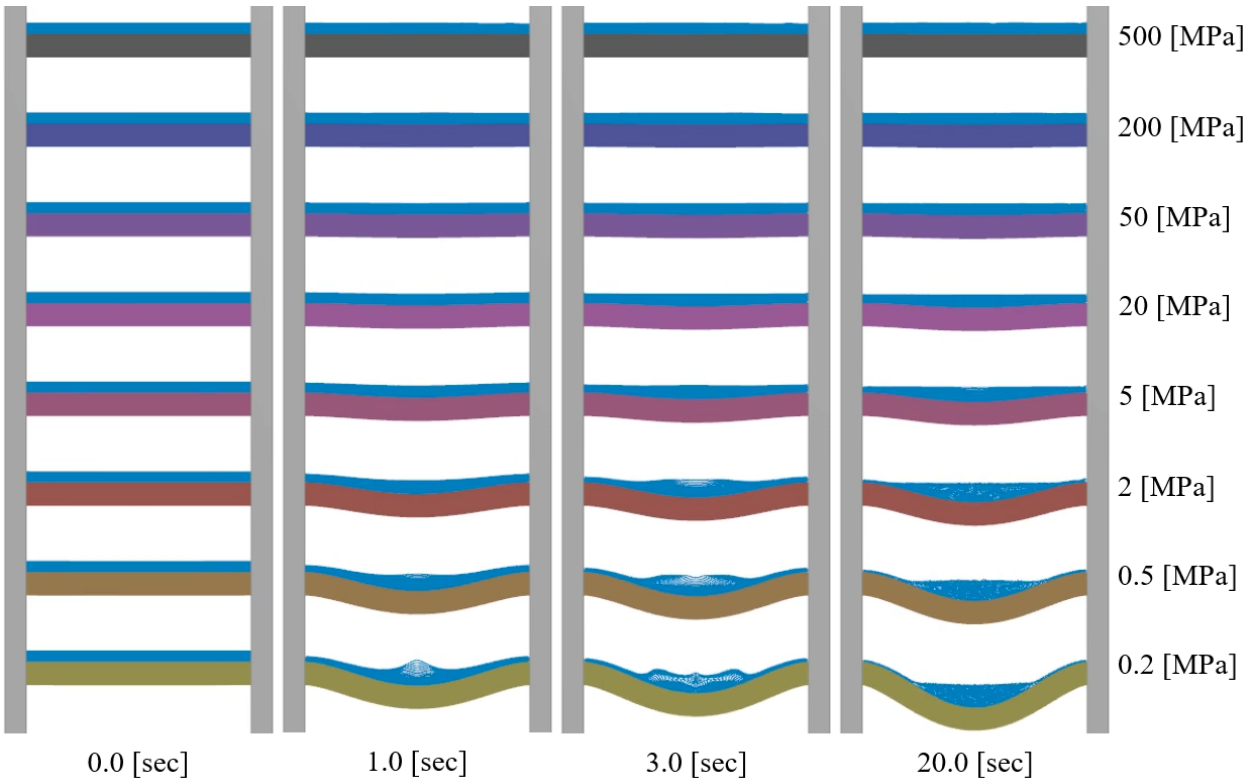

Fig. 6.16. Fluid-Structure Interaction: MPM-FEM simulations of fixed-fixed beams under a fluid load for varied beam axial stiffness. Beam stiffness values decrease from top to bottom, resulting in a changing profile of beam deflection under a nonlinearly redistributing fluid load. Gravity loads quadratically over ten seconds to reduce dynamic oscillations. Beam center deflection is tracked.

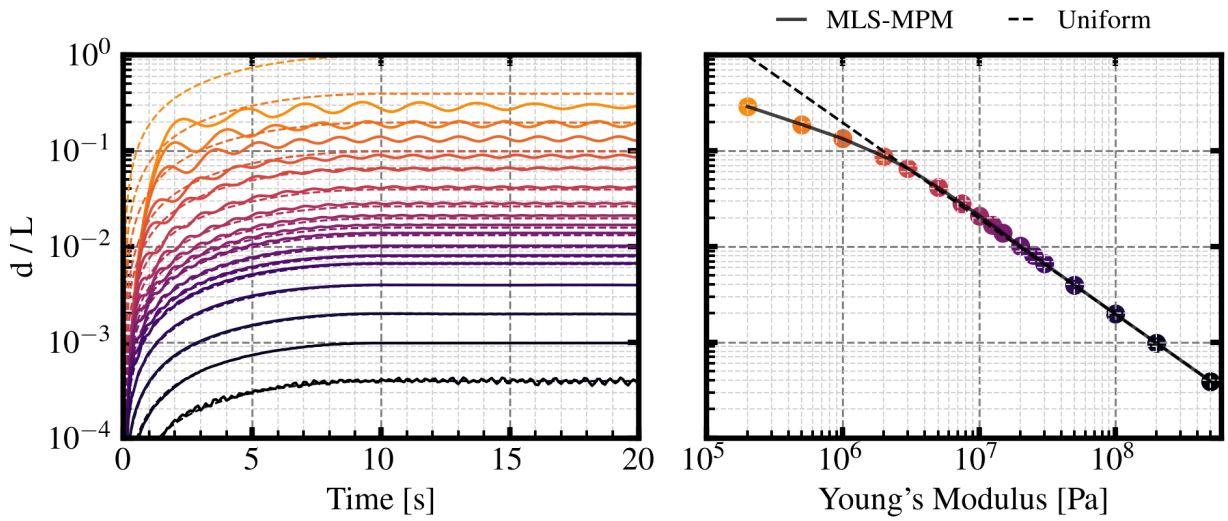

Fig. 6.17. Fluid-Structure Interaction: Displacement of an MPM-FEM fixed-fixed elastic beam center-span under a fluid load for varied beam stiffness. Comparison of our MPM-FEM model deflections to the small-deflection equation across a range of beam Young's moduli.

results originate from Lewis et al. 2023.

The top plot in Figure 6.18 normalizes numerical deflections by the small-deflection beam equation to identify relative deviation from small-deflection theory between the four different approaches. There appears to be three primary "phases" in the curves that are sorted by intervals of d_U / L, and hence directly related to beam stiffness by Equation 6.11. We interpret these three fluid-beam center-span displacement phases as follows:

1. **Small-deflections.** When d_U/L is less than 1%, the small-deflection assumption of uniform loading with fluid is valid. Simulations should match very closely to the analytical small-deflection prediction in Equation 6.11.

2. **Moderate-deflections.** When d_U/L is greater than 1% but less than 6%, the small-deflection assumption of uniform loading with fluid is invalidated. Fluid mass redistributes towards the center of the beam and results in larger than small-deflection

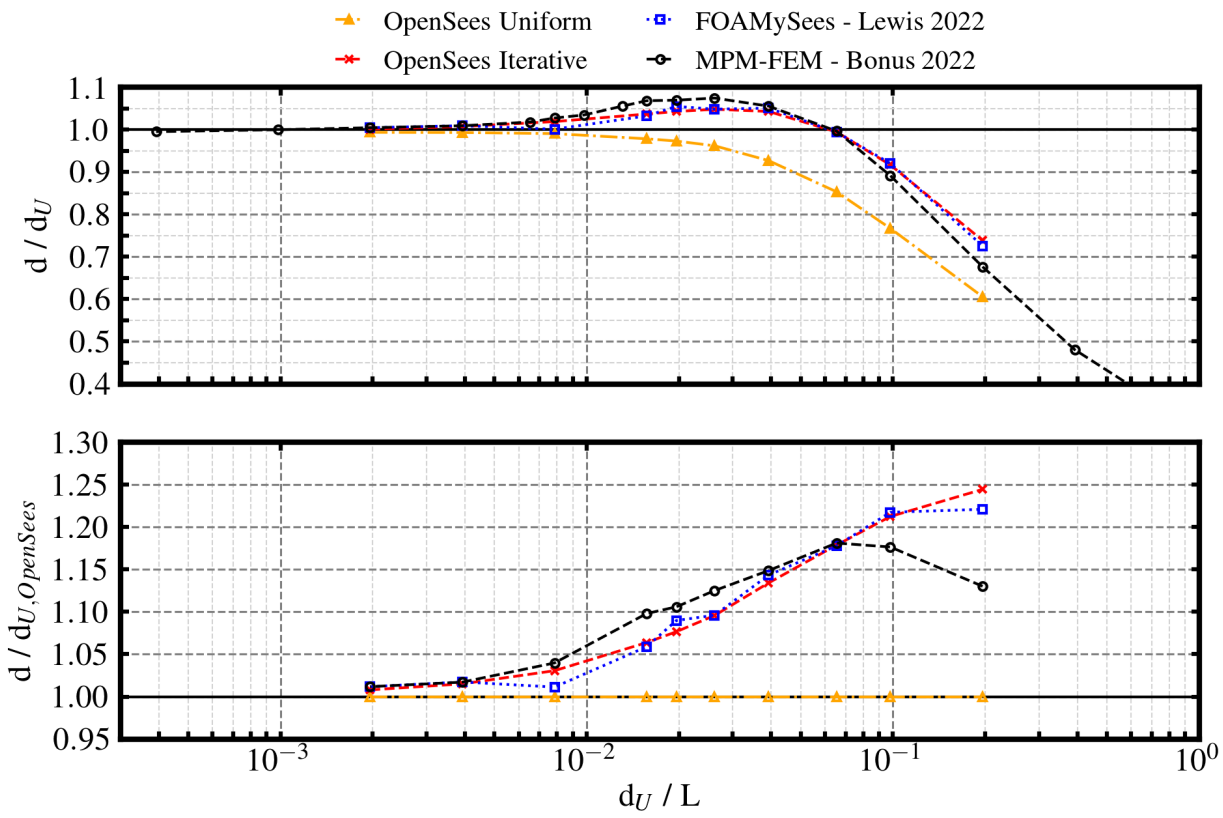

Fig. 6.18. Fluid-Structure Interaction: Comparing displacement of fixed-fixed elastic beam under a fluid load across numerical models. Comparison of our MPM-FEM model deflections to the small-deflection equation, OpenSees FEM, and FOAMySees FEM-CFD across a range of beam Young's moduli.

equation predicted displacements. This effect is referred to herein as "ponding". In this range, the deflections are not great enough for nonlinear axial stiffening to override the ponding effect.

3. **Large-deflections.** When d_U/L exceeds an estimated 6%, nonlinear fixed-fixed beam stiffening becomes substantial, as corroborated by Jing et al. 2002 which saw nonlinearity that started at 1-2% becoming dramatic by 5-6% deflections. This overrides the influence of ponding amplification, despite a more pronounced centralization of fluid, and continues to invalidate the small-deflections assumption. As a result, deflection should increasingly deviate from the small-deflection prediction as d_U/L grows, predicting smaller displacements in general.

Referring back to the top plot of Figure 6.18, we may now view the respective numerical results through the lens of the three-phase behavior. Stiff beam cases in phase one show a near-perfect match between the analytical small-deflection equation and all numerical models. This is because simulation deflections are small relative to beam length and thus they follow the assumptions set by small-deflection beam theory. In phase two, moderate deflections are anticipated and we see, as expected, that the beams deflect enough to invalidate the small deflection equation. This is due to ponding, i.e. fluid redistributes towards the center of the beam in a non-negligible way that breaks from the uniform load assumption of Equation 6.11. This results in simulation deflections larger than expected analytically. In the third phase, displacements are predicted to be large relative to both the beam length and the ponding amplification of phase two. While the trend of deviating from small-deflection physics holds from the prior phase, the direction of the trend reverses. We observe a smaller deflection compared to the analytical equation in all numerical approaches at large deflections. Stiffness is very low relative to the beam's length and thus one would expect the beam to deflect substantially under its load. Subsequently, the ponding effect should amplify due

to greater coalescing of fluid towards center-span. This could lead one to anticipate numerical deflections far greater than the small-deflection predictions. However, previously undisclosed beam effects, such as axial stiffening, are present in all numerical fluid-beam simulations but were not introduced into the simple small-deflection equation. These effects grow with displacement and manage to take a dominant role in phase three. They counteract the displacement amplification of the ponding that was already notable in phase two despite the fact that the ponding has intensified, as visible in Figure 6.16.

The bottom plot in Figure 6.18 normalizes numerical deflection by the non-redistributing fluid load OpenSees beam model. This enables the examination of the varying influence of a redistributing fluid load in three numerical methodologies. The findings of note in this plot are that MPM results have slightly increased relative deflections for moderate d_U / L values compared to OpenSees-rooted methods, meaning that ponding in the moderate deflection regime is more pronounced. The reverse is true for larger d_U / L values, with nonlinear axial stiffening playing a larger role in our MPM results. Note that FOAMySees employs OpenSees beam elements, so it is not inherently significant that their results match more closely to the iterative load calculations on an OpenSees beam than our approach using 3D MPM-FEM elements. Further discussion of FOAMySees results can be found in Lewis et al. 2023.

This benchmark demonstrates that our MPM-FEM approach captures the nonlinear fluid-beam interaction behavior beyond basic analytical equations. Further, it well replicates multi-mode fluid-beam interaction trends observed in dedicated structural engineering software, namely OpenSees and FOAMySees, while remaining viable for use in the wide range of phenomena show-cased in alternative sections of this paper. Overall, both structural FSI (FOAMySees) and generalized DFSI (MPM) approaches capture the three-phase behavior of the beam-fluid case with modest discrepancies in magnitude, reinforcing their respective capabilities in a range of dynamic, deforming fluid-structure interaction scenarios.

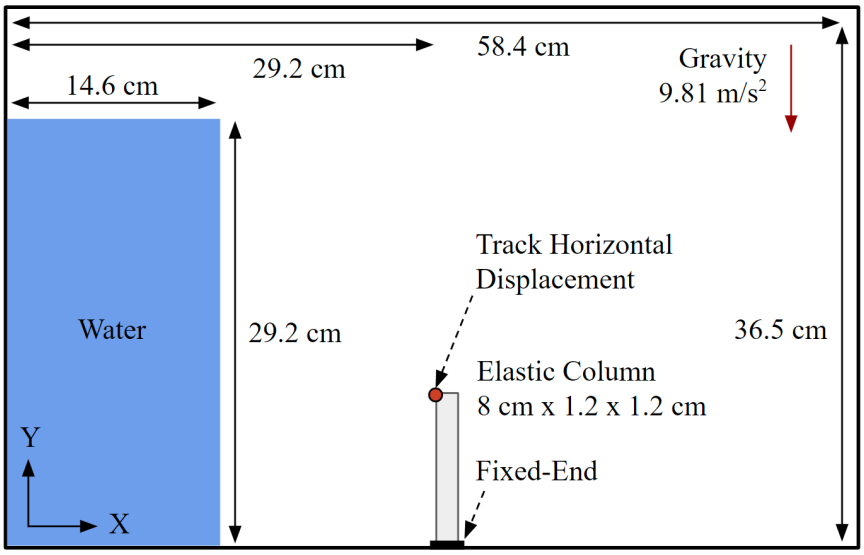

Fig. 6.19. Fluid-Structure Interaction: Schematic of dam-break impacting an elastic column validation case. Validation case schematic for fluid-structure interaction for a dam-break wave, as in Walhorn at al. 2005. Entails an unsupported water mass collapsing into an elastic column with a fixed base and a free tip. Horizontal tip displacement is tracked.

6.7 *Fluid-Structure Interaction*: *Water Dam-Break Impact on an Elastic, Fixed-Free Column*

For this fourth case we choose a demanding FSI benchmark of a dynamic water dam-break impacting an elastic column. It serves to study dam-break behavior, large structural deflections, and the interface between the two.

The validation case schematic is shown in Figure 6.19. Unsupported on one side, a water mass is instantly loaded by gravity at t = 0 seconds, causing a dam-break run-out, a relevant form to study for tsunami wave simulations at certain time and length scales (Madsen et al. 2008). In the wave's path is an elastic column, which will interact with the fluid. Displacement of the upper left tip of the elastic column is tracked in the horizontal direction, i.e. deflection of the cantilever under hydrodynamic loading. Max tip displacement demonstrates the ability of hydrodynamic impact force to manifest in a dam-break and transfer to a structure. See Tables 6.3 and 6.1 for simulation

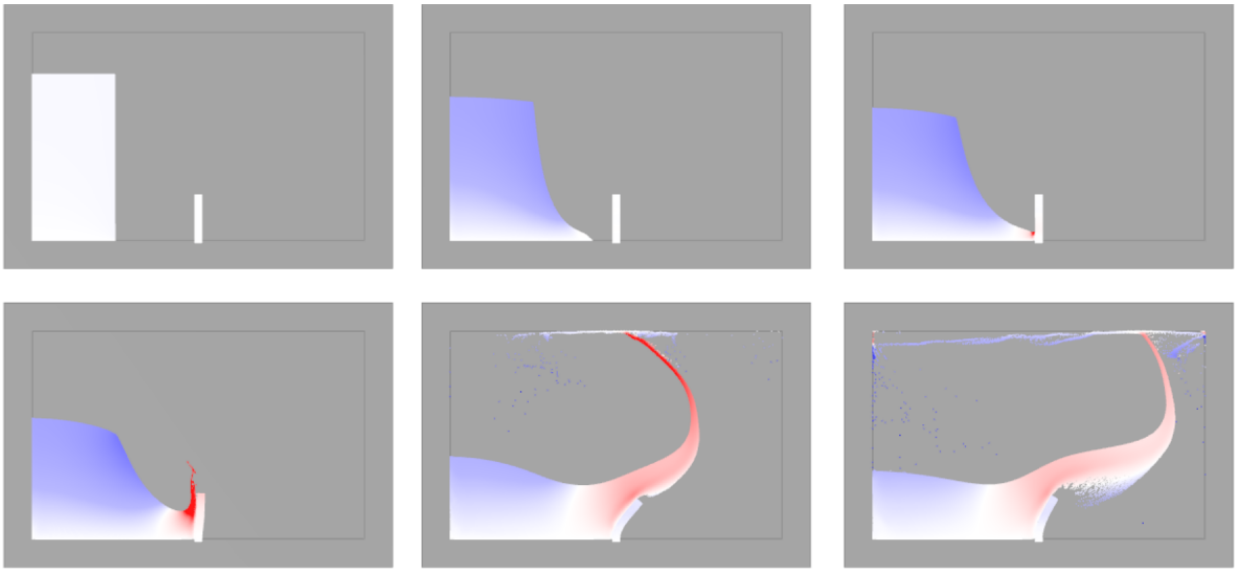

Fig. 6.20. Fluid-Structure Interaction: MLS-MPM simulation of dam-break impacting an elastic column. Vertical velocity is visualized on particles, with blue being negative and red positive. Water slides along the column with minimal damping, but noted disturbance of its lower surface.

and material properties.

We compare our new implementation approach against prior numerical results using finite elements (FEM, Walhorn et al. 2005), particle finite elements (PFE, Idelsohn et al. 2008), and coupled CFD-FEM (FOAMySees, Lewis et al. 2023) to test our approach's capabilities in dynamic large-deformation FSI.

Figure 6.20 shows the fluid and column at key times with fluid velocity visualized. Figure 6.21 records the column tip displacement in a variety of numerical approaches as the case progresses in time. Our results reinforce that first-mode behavior is approximated by all respective numerical tools, but none are identical.

Our MLS-MPM particle-based approach most closely matches the particle-based PFEM results in max displacement and behavior following the peak tip displacement. There is some notable

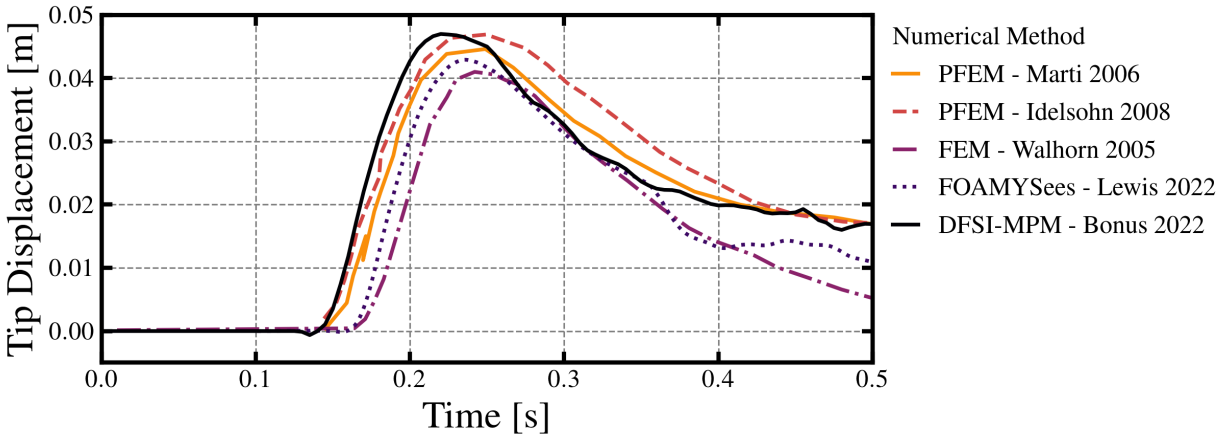

Fig. 6.21. Fluid-Structure Interaction: Displacement of elastic column impacted by fluid across numerical methods. Comparison of simulated tip displacements from FOAMySees by Lewis et al. 2023, PFEM by Idelsohn et al. 2008, MPM by our group, and FEM by Walhorn et al. 2005 for a FSI validation case from the latter paper.

deviation from CFD-FEM (FOAMySees, Lewis et al. 2023) and FEM, which resemble one another. Deviation is potentially from how fluids are modeled in respective methods, as arrival times of the dam-break at the column differ. Our numerical DFSI approach is comparable to dedicated FSI tools in this demanding dynamic benchmark.

6.8 *Debris-Fluid Interaction*: *Buoyancy of Submerged Cube Debris in a Water Tank*

A critical aspect of debris-fluid interaction (DFI) is the balance of gravity, drag, and buoyancy forces on arbitrary debris in arbitrary fluids, as noted in Nistor et al. 2017a and Wu et al. 2014. Doing so captures debris sinking, floating, and transport, all critical to DFSI. Here we characterize sinking and floating as basic 1D analytical equations for a DFI benchmark involving debris in a hydrostatic water column. However, 3D DFI simulations are difficult, with many methods going unstable without in-depth contact schemes or case simplification (e.g. only incompressible fluids, rigid debris). In our proposed DFSI approach we produce stable simulations of debris elevation and sink rate across varied debris properties at high resolutions to establish DFI capabilities.

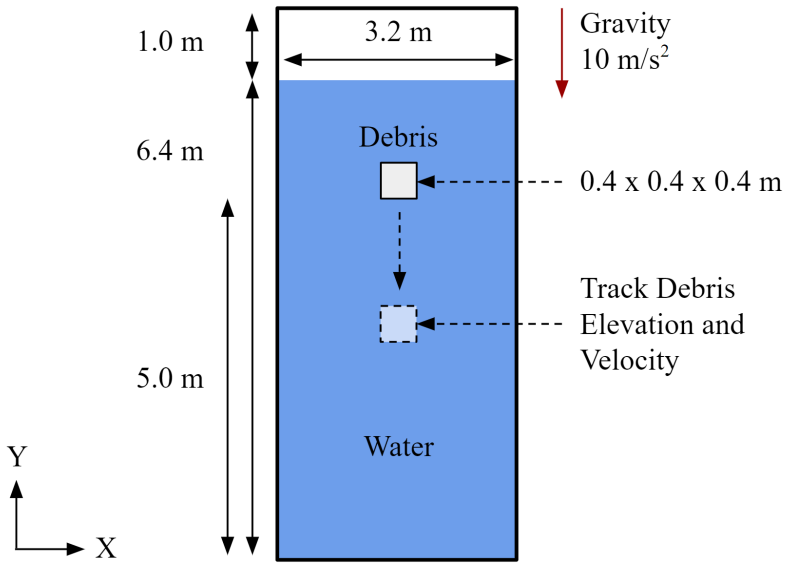

Fig. 6.22. Debris-Fluid Interaction: Schematic of debris buoyancy validation case. Cuboid debris submerged in a tall tank of still water is mobilized by gravity. Track elevation and velocity for debris of various densities and compare against an analytical buoyancy motion model. Tank walls are rigid with a separable velocity condition.

A rigid tank (3.2 x 7.4 x 0.4 m) is occupied by water (3.2 x 6.4 x 0.4 m), similar to Yang 2016. A cube debris (0.4 x 0.4 x 0.4 m) is centered in the tank's horizontal-direction with its top face at 5.4 m elevation, 1.0 m below the water's surface. Gravity is -10 m/s^2 vertically. No initial velocity or pressure field is prescribed. Walls are separable boundaries. Figure 6.22 shows the benchmark schematic.

For a simple 1D equilibrium model, combining basic forces on a piece of debris (gravity, buoyancy, and drag), we get the following:

$$f_T = f_G - f_B - f_D \quad \text{where} \quad f_G = m_d g \ , \quad f_B = \rho_w g V_d \ , \quad \text{and} \quad f_D = c_d \rho_w \dot{x}_d^2 \frac{A_d}{2} \ , \qquad (6.12)$$

where m_d is mass of debris, g is gravity (-10 m/s^2), c_d is the drag coefficient (c_d = 2.1, common values for a perpendicular square are 2.05 - 2.25), V_d is boundary-layer adjusted submerged

volume of debris (0.4+2Δx x 0.4+2Δx x 0.4 m^3), A_d is boundary-layer adjusted debris drag area (0.4+2Δx x 0.4+2Δx m^2), and $\dot{x}_d$ is debris velocity. A simplistic model for debris moving through a fluid column is derived as

$$m_d\ddot{x} + \rho_w g V_d + c_d \rho_w \frac{A_d}{2}\dot{x}^2 = m_d g \ , \tag{6.13}$$

which we solve with explicit time-integration. Acceleration of the debris ($\ddot{x}_d$) in a time step (t^{n+1}) is the total force divided by the mass relative to a previous time-step (t^n), stated by

$$\ddot{x}_d^{n+1} = \frac{1}{m_d}\left(m_d g - \rho_w g V_d - c_d \rho_w \frac{A_d}{2}(\dot{x}_d^n)^2\right) \ . \tag{6.14}$$

Debris velocity ($\dot{x}_d$) updates by taking acceleration in Equation 6.14 as constant over a discrete time-interval (Δt), which then allows debris elevation (x_d) to advect for the time-step. Both are phrased as

$$\dot{x}_d^{n+1} = \ddot{x}_d^{n+1}\Delta t + \dot{x}_d^n \quad \text{and} \quad x_d^{n+1} = \dot{x}_d^{n+1}\Delta t + x_d^n \ , \tag{6.15}$$

to complete our simple analytical model for debris in a water column. No dynamic inertial factors, tank vertical/horizontal constraints, surface effects (e.g. penetration, breaching, surface-tension), fluid shearing viscosity, or turbulent phenomena are considered. Initial position (x_d = 5.0 m, base elevation of debris) and velocity ($\dot{x}_d$ = 0 m/s) are set.

Three benchmark cases set debris density as 750, 1500, and 3000 kg/m^3. Respectively, we anticipate the debris to rise to the surface, sink slowly, and sink quickly before impacting the rigid bottom of the tank.

Key simulation frames are shown in Figure 6.23 with fluid vertical velocity visualized. Simulation and material settings are in Tables 6.3 and 6.1. Note that we apply both Simple F-Bar antilocking

with a ratio of 0.99 and FLIP velocity mixing with ratio of 0.30 on a grid with $\Delta x = 0.0125$ meters. Debris elevation and velocity through time are tracked in Figure 6.24. Elevations follow the analytical model reasonably. Velocities track along similar trends, but there is a noted difference in behavior initially, when approaching the tank's bottom, and when breaching the surface. These may, in part, be explained by lack of inertial factors, tank constraints, and surface effects in the analytical model. We do not model turbulence in the debris' wake despite using a coefficient of drag, $c_d = 2.1$, that assumes standard turbulence in the eddies trailing the cuboid. Because of this, the initially slowed movement of our simulated debris is expected due to larger pressure drag deriving from slowed development of initial eddies. FLIP velocity mixing allowed us to better approach turbulent-like DFI compared to MPM's purely laminar default flow, but only after roughly half a second of motion.

The coefficient of drag (c_d) is an important parameter for debris movement in a water body but we only give it minor consideration insofar. It affects the force balance relative to debris velocity and geometry and can be a non-constant, non-linear variable in some scenarios. c_d can vary depending on Reynold's number (R_e), debris surface roughness, and many other factors, and we can experimentally and numerically (e.g. in CFD software like ANSYS) find reasonable constant values for basic geometries in simple cases, for instance relative to water-flow direction a square at 0, a square at 45 degrees, a cube at 0 degrees, and a cube at 45 degrees respectively result in c_d of 2.05, 1.55, 1.05, and 0.8. While our simulations are in 3D in this validation case, they are better characterized by the c_d of a square due to out of plane symmetry constraints (i.e. water does not extend past the out-of-plane faces of the debris). Note that the debris we use can tilt slightly as it moves in the water column, lowering c_d and potentially creating oscillations in the drag force. Likewise, the debris' leading face can deform as debris velocity and water pressure increase (i.e. further down debris goes) which also drops c_d, although this is fairly negligible unless at low debris stiffness in highly dynamic scenarios.

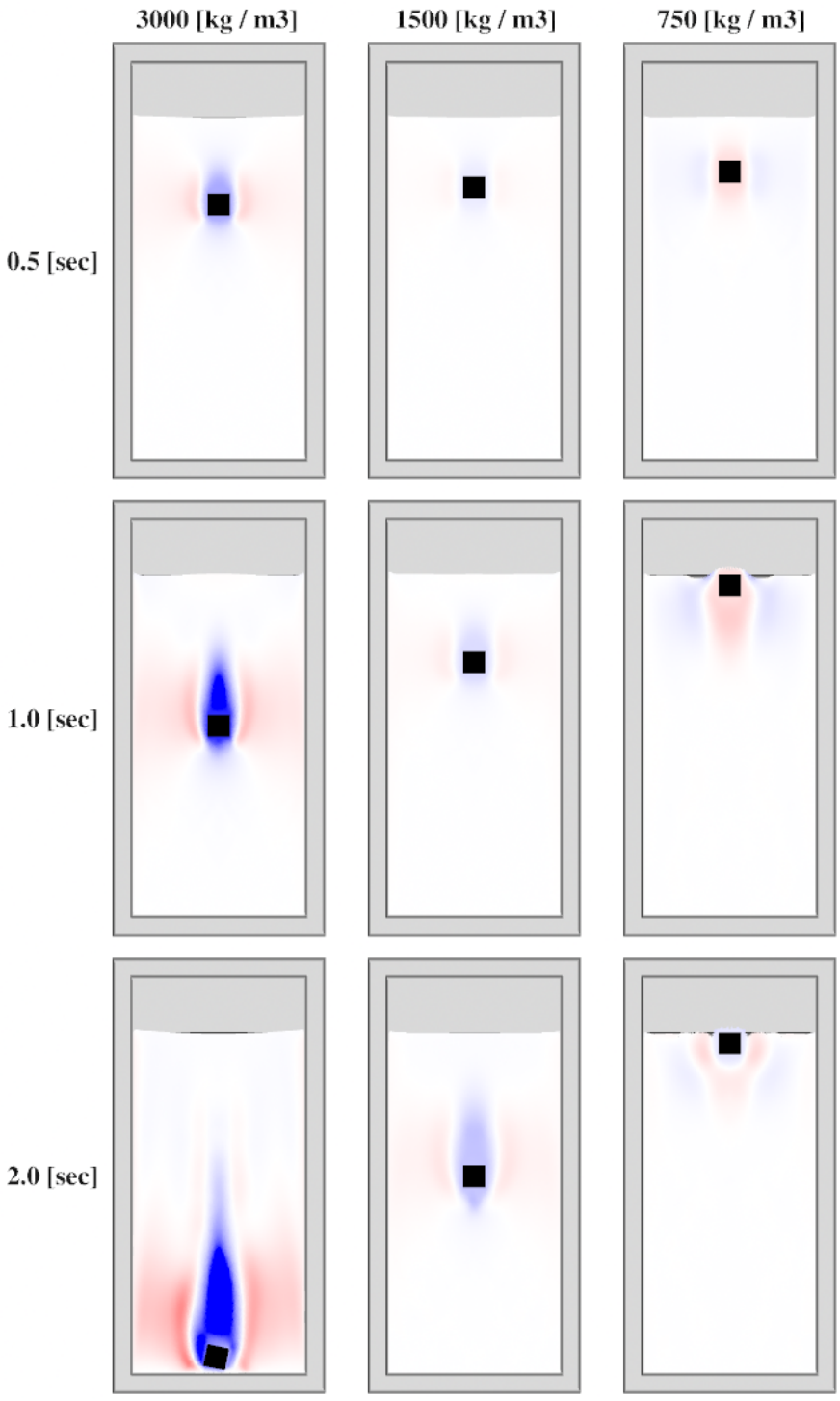

Fig. 6.23. Debris-Fluid Interaction: Debris buoyancy simulations in MLS-MPM. Cuboid debris of densities **(Left)** 3000 kg/m^3. **(Center)** 1500 kg/m^3. **(Right)** 750 kg/m^3 are placed in a tall tank of water and simulated in MPM to study debris-fluid interaction. Vertical velocity is visualized on water particles, with Blue = -3 m/s, White = 0 m/s, and Red = 3 m/s.

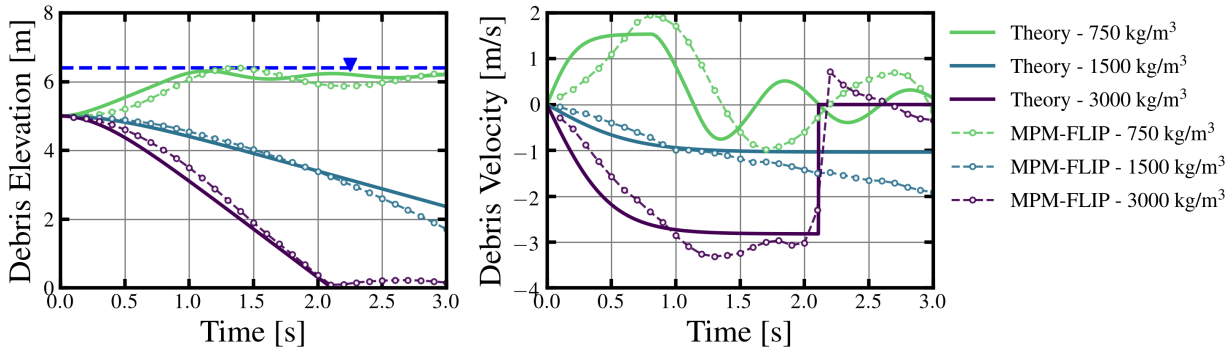

Fig. 6.24. Debris-Fluid Interaction: Simulated motion of debris in water tanks using MLS-MPM with FLIP advection and F-Bar antilocking. Varying MLS-MPM debris densities seen to affect debris motion direction and speed. Water particles use FBAR = 0.99 and FLIP = 0.3. (Left) Elevation curves match reasonably between our MPM results and the analytical motion model. Using FLIP improves MPM's standard damped motion substantially. (Right) Velocity trends generally match, though there are differences in initial speed as MPM does not model turbulence.

Results demonstrate that our numerical MLS-MPM DFSI approach reasonably captures debris-fluid interaction (DFI). The largest limitation is observed to be in the debris' boundary layer width and the fluid turbulence in the wake of the cuboid debris. The former is artificially thickened by the grid size in MPM, while the latter is not directly replicated in MPM and was instead imitated using FLIP velocity mixing. For a debris buoyancy and sinking validation case with varied debris material properties, our approach demonstrates primary DFI behavior, surpassing standard MPM among other methods. However, we highlight a need for further investigation into high-performance turbulence models in MPM for when DFI is the dominant interaction in a DFSI problem of interest.

6.9 *Debris-Debris Interaction*: *Uniform Line of Dominoes Toppling*

Debris-debris interaction (DDI) concerns the dynamic influence of an arbitrary number of debris on each other. This is a difficult component of DFSI, particularly if explicitly determined stress waves are computed throughout material continuum as is done in MPM. While small-scale

interactions, e.g. two debris colliding head-on, are relatively trivial, large sets of ordered or disordered debris pose challenges and are often the subject of study in the fields of sea-ice, tsunami debris, and many other fields.

Basic DDI (e.g. direct collisions) is handled respectably by MPM's automatic no-slip contact, even with fluids present, but can fall short in less ideal scenarios when compared to DEM and FEM which enjoy extensive validation for pure DDI problems. MPM and MLS-MPM may produce **(i)** damped separation after collisions, **(ii)** frictional shearing behavior along the contact surface, and **(iii)** a fictitious contact-gap of roughly one to two grid-cells, Δx. Furthermore, if debris is instantiated side-by-side (i.e. within a grid cell) they have little ability to separate at all during transportation or impact unless acted upon by a large tensile stress. We apply ASFLIP to MLS-MPM in this section, alongside a GPU enabled high-fidelity mesh, to alleviate these problems without involving a dedicated contact algorithm at all in the following DDI benchmark.

Figure 6.25 is a schematic of the DDI validation case. Twenty-four vertically oriented, uniform dominoes are placed in a straight line at even intervals. Dimensions of a domino is 6 x 48 x 24 millimeters (t = 0.006 m, h = 0.048 m, w = 0.024 m). Neighbors are spaced 30 mm center-to-center ($s + t$ = 0.03 m), i.e. a 24 mm internal gap spacing (s = 0.024 m). The domino material replicates common plastics in commercial dominoes by using a Young's modulus of 2 GPa, Poisson's ratio of 0.3, and density of 1,400 kg/m^3. The leftmost domino is perturbed by a 1.0 m/s velocity applied to its upper-left tip for 0.01 seconds. A frictional floor, with static and dynamic coefficients of friction set to 0.3, counteracts most sliding. This causes the first domino to rotate, closing the gap with the adjacent piece until collision occurs. The effect propagates until all the dominoes are destabilized. A "domino wave" is to be observed moving through the ensemble.

Analytically, the seminal paper by McLachlan et al. 1983 proposed a simplified domino wave speed, stated as:

$$v_{\mathrm{D}} = G\left(\frac{s}{h}\right)\sqrt{gh}\ , \tag{6.16}$$

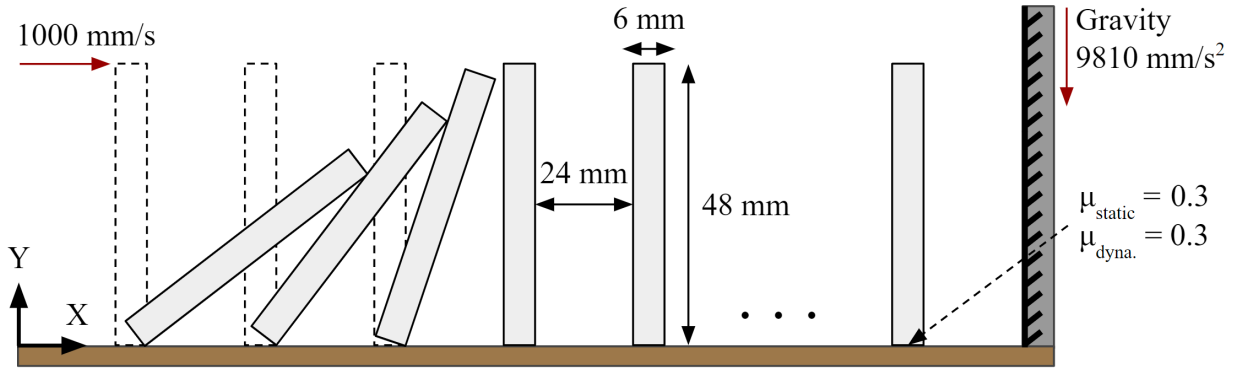

Fig. 6.25. Debris-Debris Interaction: Schematic of domino toppling validation case. Twenty-four dominoes are standing in a straight line, loaded by gravity, at even intervals center-to-center. The leftmost domino has its tip velocity imposed for a time-interval. Frictional contact at the floor counteracts sliding, leading to rotation and eventual collision with its neighbor. Measure domino-wave steady-state speed.

where $G\left(\frac{s}{h}\right)$ is some function that relates velocity to the ratio of spacing and domino height. This function was given an approximation by Efthimiou and Johnson 2007 for a domino system with key restrictive assumptions: no domino sliding, no energy dissipated at contact, collision are instantaneous, collisions are perfectly elastic, collisions are single-hit, collisions are head-on, dominoes are rigid rods, and a domino's mass occupies a point at their tips. For our case, where $s/h = 0.5$, we can take $G\left(\frac{s}{h}\right) = 1.70$. This gives an idealized domino-wave steady-state speed of $v_{\mathrm{D}} = 1.17$ m/s. However, this is not fully applicable to a real-world dynamic domino system and thus is only considered an upper-bound.

A potentially complimentary approach was suggested by Leeuwen 2010, who improved on soliton wave interpretations of Stronge 1987 and gave greater consideration to domino-domino friction, moments of inertia, domino thickness, and inelastic collisions which in sum served to reduce predicted values of models like Efthimiou and Johnson 2007 to be closer to experimental observation. A soliton may loosely be thought of as a self-preserving wave-form translating through a material media. In our case the medium is an array of deforming dominoes with wide

discontinuities. We must replicate a soliton by means of both translation and rotation from a brief initial disturbance, with input energy having to overcome frictional floor-contact, frictional debris-debris interfaces, and deformation of hyper-elastic dominoes in a stable manner. Success strengthens a methods claim to supporting DDI, as arrival at a steady-state domino-wave via rapid exchange of energy between gravity potential, kinetic, and strain energy of many debris bodies in the presence of friction is demanding.

Our MLS-MPM simulation grid-cells are 0.0005 meters and the particles-per-cell is 27. The time-step is 3.125e-7 seconds, corresponding to a CFL number of roughly 0.45. We set the ASFLIP velocity and max position correction ratio to 0.999 and 0.025 respectively. Figure 6.26 is composed of key frames from a ClaymoreUW MPM simulation using our DFSI approach. Qualitatively, it visualizes the primary destabilizing mode, i.e. toppling, and the domino wave propagation speed. Both are found to be reasonably consistent with expectations set by FEM results in Sun 2020 and our own table-top experimental recreation in Figure 6.27.

Numerically, we imitated a laser distance finder at elevation 0.048 meters (initial domino tip elevation) at the far left of Figure 6.25, pointing to the right, in ClaymoreUW. Steady-state domino-wave speed was determined to be 0.53 m/s in our MLS-MPM DFSI approach and is reached after an estimated 5-7 domino collisions. It is below the highly simplified, but useful, analytical model of Efthimiou and Johnson 2007 by 54%, which is expected as their assumptions are suited more towards defining an upper-bound than estimating speed in a frictional-slip system. For a more appropriate comparison of values, we refer to Sun 2020 and Song et al. 2021, who performed very similar domino studies in Abaqus FEA and found a domino wave-speed scaling law of:

$$v_{D,\text{Straight}} = C_{\text{Straight}}\left(s^{\alpha}\right)\sqrt{\frac{gt}{h}} = 3.488\left(s^{\frac{1}{2}}\right)\sqrt{\frac{gt}{h}}\,, \tag{6.17}$$

where α describes the relation between domino spacing and wave speed for linear arrangements ($\alpha = 0.5$) and C_{Straight} is a linear array coefficient determined to be 3.488 for cases with

domino-domino interface friction. Song et al. 2021 observed in their domino validation case, which we primarily based our own on, that for typically sized dominoes arranged in straight lines the velocity (Eq. 6.17) plateaus in magnitude for a spacing of 0.019 meters and maintains that speed until about 0.025 meters. This is proposed to be due to the balance of friction and conversion of horizontal to vertical motion during toppling. Thereby we compute the predicted value of 0.5324 m/s using the spacing clamp in equation 6.17, i.e. $s = 0.024 \rightarrow 0.019$ meters, which is within 1% of our MLS-MPM simulated speed of 0.53 m/s. Even if the clamping were bypassed the predicted domino-wave velocity is 0.5984 m/s, still within 13% of our observed MLS-MPM value.

While simplified analytical models and alternative numerical findings are useful, there is simply no substitute to an experiment when validating. We undertook our own table-top experiment of the domino case in Figure 6.25, with photographs in Figure 6.27 showing striking similarity to our ClaymoreUW MLS-MPM simulation's behavior in Figure 6.26. The experimental wave speed is found to be 0.42 m/s. This value is 20% smaller than our MLS-MPM simulation's speed of 0.53 m/s, lending confidence to our ability to replicate primary behaviors of domino systems but showing that there is room for improvement. We emphasize that our preliminary experiment's frictions, initial velocity generation, material parameters, and other factors are not fully identical to the validation case diagram in Figure 6.25. Our table-top wave-speed should not be used beyond a sanity-check for basic behavior, but reasonable agreement is clear and it forms a strong lower-bound. Readers are encouraged to replicate our table-top results for their own validation efforts, as the set-up cost and required time is low.

Overall, our MLS-MPM simulated domino wave-speed of 0.53 m/s is 54% smaller than the simplified analytical upper-bound proposed by Efthimiou and Johnson 2007, within 1% and 13% of the clamped and unclamped scaling law of Sun 2020 and Song et al. 2021 which derived from parametric Abaqus FEA, and is 26% larger than the table-top experiment value produced by our

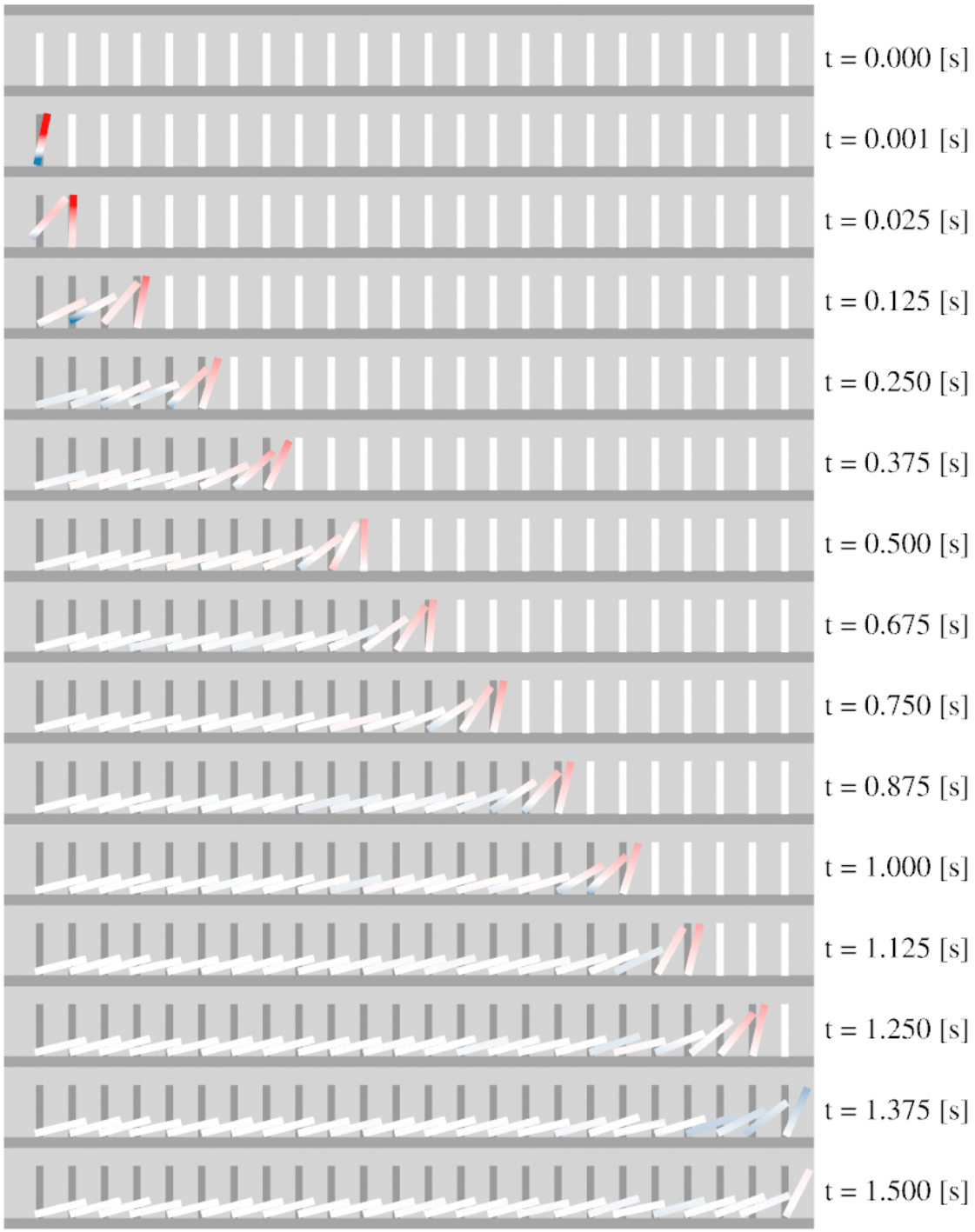

Fig. 6.26. Debris-Debris Interaction: Toppling dominoes via MLS-MPM with ASFLIP. 24 dominoes in MLS-MPM stand in a straight line at intervals of 0.03 meters center-to-center. First domino's tip is hit by a 1 m/s velocity boundary for 0.01 seconds. Floor is frictional. Toppling attains steady-state speed of 0.53 m/s after 4-7 dominoes. Horizontal particle velocity shown in blue, white, and red which map to -0.2, 0.0, and 0.6 m/s respectively.

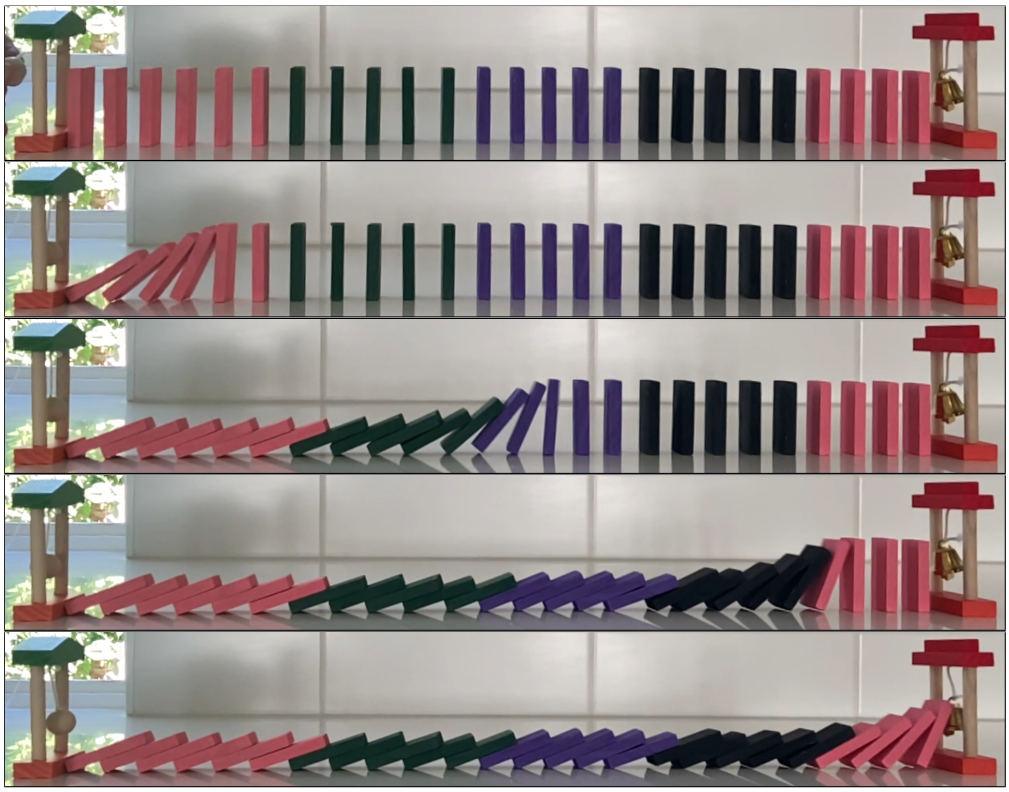

Fig. 6.27. Debris-Debris Interaction Experiment: Table-top domino toppling experiment. Preliminary experiment with similar specifications to the DDI validation case. Steady-state domino wave-speed reaches 0.42 m/s after six collisions.

group. For such a demanding DDI validation case, we are pleased to find our DFSI MLS-MPM approach in a sweet-spot among varied approaches to manifesting domino wave-speeds. We also corroborate the observation of Sun 2020 that it takes an estimated six collisions for the steady-state domino-wave speed to be attained via both our MLS-MPM simulations and our table-top experiment. Further, the striking resemblance between simulated and experimental domino ensemble collapse in Figures 6.26 and 6.27 respectively lend confidence to replication of DDI behavior beyond just domino-wave speed, opening new avenues for high-performance simulation in the study of sea-ice and tsunami debris-fields.

6.10 *Debris-Debris Interaction*: *Collapsing Discs and Force-Chains*

As a qualitative example of multi-directional debris-debris interaction (i.e. not uni-directional as in Section 6.9), Figure 6.28 shows our simulation of 648 collapsing disc debris interacting to demonstrate our approaches applicability to hundreds of debris and manifestation of the complex "force-chain" phenomena, i.e. preferential paths of loading in seemingly homogeneous debris-fields and granular media. In a similar vein, we will evaluate our MPM approach for simulations of sand-gravel masses as they flow down hill-slopes in Section 9.4.

6.11 *Final Remarks*

This paper presents seven validation cases used to find the strengths and weaknesses of our open-source, high-performance Multi-GPU MPM approach to debris-fluid-structure interaction (DFSI). Composite validation of debris-structure interaction (DSI), fluid-structure interaction (FSI), debris-fluid interaction (DFI), and debris-debris interaction (DDI) was carried out in this pursuit. Cases executed thousands to millions of time-steps at scales of 100,000 to 1,000,000,000 particles within hours, broaching the door-step of exa-scale computation. Our results show the flexibility and validity of our numerical approach across DFSI phenomena, but also identifies

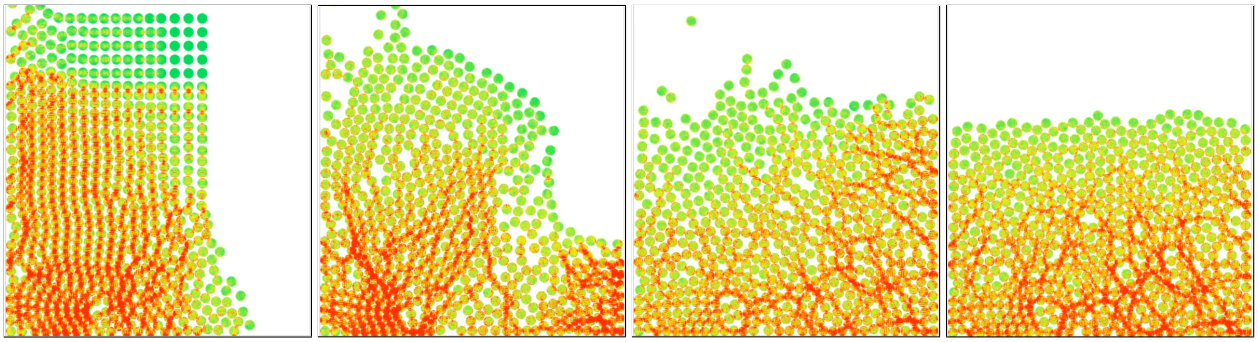

Fig. 6.28. Debris-Debris Interaction: Force Chains Appear Within a Large Debris-Field's Collapse. Non-analytical case to demonstrate debris-debris interaction for hundreds of debris. Notable force-chain manifestation (i.e. preferential stress paths in seemingly homogeneous media) is observed. Four key-frames shown (left-to-right, time = 1, 2, 3, 7 sec.) for 648 disc debris (18 x 36) collapsing in a rigid tank. Modeled via the proposed Material Point Method DFSI approach. Density 1000 kg/m^3, Young's Modulus 0.1 MPa, Poisson Ratio 0.3. Pressure visualized on debris. Each disc modeled as 1000 MPM particles, 648,000 total, with a Fixed-Corotated solid material.

room for improvement, such as in turbulence modeling and in frictional contact between debris. Debris-structure interaction (DSI) results in Sec. 6.2 agree well with a simple analytical model of rectangular prism debris impact used in practice by structural engineers (Aghl et al. 2014b). Both impact force and duration match closely across varied debris densities, with additional dynamic pressure wave behavior captured by our MLS-MPM approach.

Fluid-structure interaction (FSI) results in four benchmark cases (Sections 6.4, 6.5, 6.6, and 6.7) capture all phenomena of interest seen in corresponding analytical equations and alternative numerical approaches. We achieved this using FLIP-PIC velocity mixing (Stomakhin et al. 2013) to reduce fluid damping and explicit volumetric antilocking by Zhao et al. 2023 to minimize erroneous pressure scillations. The latter was recently extended to modified implicit form by Xie et al. 2023 and may allow high-performance FSI with porous foundation soils in our future works. Our approach showed ease of coupling for MPM-FEM in Section 6.6, where basic four-point tetrahedrons composed a fluid-loaded beam. The coupling was quick to implement on Multi-GPUs and was nearly equivalent to results from dedicated structural engineering tools (e.g.

FOAMySees). This has spurred our interest in exploring more Lagrangian body couplings in MLS-MPM, such as frictional beam elements from Kang et al. 2021, plates, or shells Finally, our DFSI approach showed significant capabilities for accurate, stable, and expedient FSI results at massive computational scales, e.g. for Section 6.4's billion particle fluid tank simulated for a million time steps in 23.5 hours on one HPC node, and at length parity with wave-flume experiments and real waterborne hazards.

Debris-fluid interaction (DFI) results in Section 6.8 demonstrate that our numerical DFSI approach strongly captures the complex DFI of submerged debris motion and laminar flow around elastic debris. It does not capture turbulence as we do not use a turbulence model. Balance of the debris' buoyant, drag, gravitational, and elastic force is stable across varied debris densities and motion modes (plummeting, sinking, rising, and re-submerging), surpassing standard MPM without coupling additional numerical methods as often done elsewhere (e.g. for Smoothed Particle Hydrodynamics and the Finite Volume Method). However, we emphasize that relative lack of research into turbulence models for MPM limits the quality of pressure drag on even a basic cube debris and thus their steady-state motion. We assert the lack of readily available and validated turbulence models for MPM is the primary bottleneck on our DFSI approach until the MPM literature advance. Despite this, we achieve imitation of debris motion applicable for coastal engineering contexts that feature, but are not dominated by, turbulence by applying the computationally expedient ASFLIP (Fei et al. 2021a) and FLIP-PIC velocity mixing methods (Zhu and Bridson 2005) (Stomakhin et al. 2013).

Debris-debris interaction (DDI) results in Section 6.9 show that our DFSI approach characterizes the dynamic nuances of large groups of deforming debris and stably propagates destabilizing waves through media with wide discontinuities. A line of uniform dominoes is disturbed (i.e. pushed) at one end to induce a toppling "domino-wave" in the discontinuous medium. The MLS-MPM wave-speed (0.53 m/s) is resolved within 1% of scaling-laws produced by parametric

Abaqus FEA by Sun 2020 and Song et al. 2021, found to be 54% lower than a simplified analytical upper-bound by Efthimiou and Johnson 2007, and is 26% larger than a preliminary table-top experiment conducted by our group. This spread shows the variance that different approaches give to the same problem, but we find that our DFSI MLS-MPM occupies a sweet-spot amongst them. Significantly, our approach does not require an advanced contact model, the use of which inflates computational time and memory use due to tracking surface normal vectors, impulses, etc., and yet we observe ideal no-penetration contact across all dominoes and boundaries. The most common criticism of MPM for DDI problems is its overly sticky contact and generally damped behavior, yet our results bypass the bulk of this through high-resolution computation and a simple advection scheme from computer graphics (Fei et al. 2021a).

To our knowledge, our engineering approach to DFSI using computer graphics rooted methodologies, namely MLS-MPM (Hu et al. 2018), ASFLIP (Fei et al. 2021a), and B-Spline F-Bar volumetric anti-locking (Zhao et al. 2023), has never been attempted or validated for simulation of DFSI, let alone at the lower-bound of exa-scale computation. This recipe, curated to be favorable for massively parallel computation, shows uniquely strong promise for the rapid study of multi-material, multi-phase, constitutive nonlinear, and large deformation phenomena as the decade progresses.

Chapter 7

DEBRIS-FIELD IMPACT EXPERIMENTS AND SIMULATIONS - OREGON STATE UNIVERSITY'S LARGE WAVE FLUME

In this chapter debris-field wave-flume tests at Oregon State University's Large Wave Flume (OSU LWF) are briefly summarized before leading into the numerical simulations that were the primary goal of our project. Here we investigate the strengths and limitations of our approach to debris-fluid-structure interaction (DFSI) in a coastal engineering application. Fine details of experiments are available in adjacent publications from our group. Namely, the work of Andrew Winter (Winter 2019) (Winter et al. 2020), Krishnendu Shekhar (Shekhar et al. 2020), and Dakota Mascarenas (Mascarenas 2022) (Mascarenas et al. 2022). Experiments were performed in the NHERI OSU LWF, a 100 meter long flume with adjustable bathymetry, in order to quantify stochastic impact loads of ordered and disordered debris-fields on effectively rigid, raised structure. This research aims to produce a robust database (numerical and physical) from which to eventually be able to extract both the first-principals of wave-driven debris-field phenomena and design guidelines on induced forces. Because the structural impact loads of debris-fields driven by waves are notoriously chaotic, meaning very small changes in initial conditions can greatly alter structural demands between near identical cases, this represents a step towards better understanding one of the least understood phenomena faced by modern coastal engineers.

We recreate all experiments using our Multi-GPU Material Point Method code ClaymoreUW. Particle counts range from 75 to 150 million as seen in Figure 7.1, allowing us to recreate 50+ experimental cases with excellent replication of debris-field behavior as shown in experimental photos of Figure 7.2.

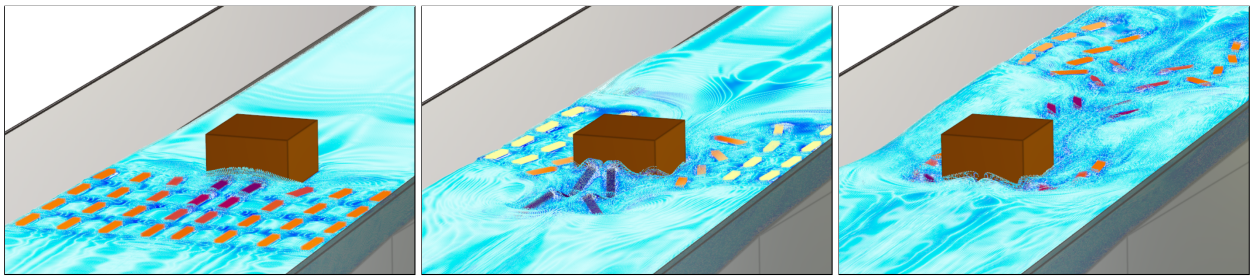

Fig. 7.1. Debris-fluid-structure interaction: 140 million particle simulation of fluid-driven debris colliding with and damming on a raised structure. Key-frames of 32 buoyant debris driven into a structure by a solitary wave. Modeled via the proposed Material Point Method approach. Total of 20 hours to simulate 140 million nearly incompressible MPM particles for 25 seconds. Stream-wise velocity shown on debris. Based on experiments at Oregon State University's Large Wave Flume by Mascarenas 2022, Shekhar et al. 2020, and Winter et al. 2020.

Fig. 7.2. First impact, decoupling, and second impact of 16 longitudinal debris on a raised structure. Photographs taken at the OSU LWF during experiments by Mascarenas 2022.

Three categories of debris-fields are studied herein: **(i)** longitudinally ordered debris-fields (long-axis parallel with flow), **(ii)** transversely ordered debris-fields (long-axis perpendicular to flow), **(iii)** randomized debris-fields, and **(iv)** ordered debris-fields scaled to a prototype tsunami according to Froude and Froude-Cauchy similitude. We studied slightly positive buoyancy, stiff rectangular prism debris with 10:2:1 length:width:height ratios with counts between 2 to 32 per debris-field. These specifications apply to design against automobile and shipping crate debris-field hazards observed in the catastrophic Tohoku 2011 tsunami (Naito et al. 2014), a

tragedy that is important to learn from due to a similar events anticipation in the American Pacific Northwest.

For ordered debris-field arrays, our simulated structural loads are within 5 - 15% of experimental medians, typically fall within interquartile ranges, and never qualify as outliers relative to experimental trial envelopes. We validate against two very similar (but not identical) physical studies done in the OSU LWF by Shekhar et al. 2020 and Mascarenas 2022, indicating high accuracy of our model and low bias to minor experiment specifications.

For randomized debris-fields, we replicate all max peak load behavior observed in analysis by Mascarenas 2022 (e.g. debris-field density influence on structural peak loads) and fill in gaps in the sampled variables (e.g. untested ratios of debris-field area to frame area) to take initial steps towards design equations. Trends of peak impact loads are found relative to debris-field density, streamwise length, transverse length, and total debris count (i.e. measure of total mass).

For ordered debris-field arrays extrapolated to prototype scale, we find that the relative significance of debris-field loads compared to wave-only loads can be highly sensitive to the similitude applied. For Froude-Cauchy similitude, which consider the balance of inertial, gravity, and elastic forces, debris impact loads are 210 - 270% greater than expected relative to hydrodynamics. This derives from elasticity's affect on wave energy reflections off of the inclined ramps, aspects of solitary wave breaking, and from a debris-field's ability to displace a structure's protective fluid buffer without undue momentum loss. Because most models (e.g. OSU LWF experiments) and prototypes (tsunami event) use water as the wave's fluid there will be compression dependent behavior of engineering concern at real scales that cannot be directly observed in water-based wave-flumes. Further, as many numerical models apply incompressible material, they are unable to directly evaluate nonlinear compressive behavior in the fluid, debris, and the structure. To fully account for this effect, we claim that wave-driven debris-fields should be modeled with tools able to replicate the scaling of elastic and elasto-plastic materials or risk

greatly underestimating structural loading. If this finding persists in alternative wave-flumes of various scales, it will signify that debris-field impacts from a multitude of experimental studies are twice as significant than their respective authors were able to report originally. This effect could be even greater at Froude scales less than 1:10, common for small to medium sized flumes. Additionally, this work has produced a validated digital twin of the OSU LWF for the scientific community. This digital twin will be included in the Natural Hazard Engineering Research Institute (NHERI) SimCenter's HydroUQ tool for uncertainty quantification in coastal engineering events to allow for high-performance simulation of a swath of experiments performed in the facility. We highlight the value of advanced computational methods like Multi-GPU MPM in understanding complex coastal phenomena, such as flow-driven debris fields during large-scale inundation events, and emphasize that our results are only feasible because of this capability. Four to eight years prior, they would have been unrealistic to achieve without reserving a significant portion of a supercomputer per run at great expense.

7.1 *Literature Review*

Common guidelines for tsunami-driven debris-field impacts originate from Naito et al. 2014, Chock 2016, and other seminal works which were primarily based on a limited set of experiments and case studies, namely the Tohoku 2011 tsunami. The study of tsunami-driven debris-fields in a

Fig. 7.3. Photographs of Oregon State University's large wave flume. (Right) OSU LWF with large breaking waves. (Left) Preparing random debris test in front of the orange structural box.

more generalized phenomenon-centered context has been experimentally pursued in recent years using scaled wave-flume facilities (Stolle et al. 2016; Goseberg et al. 2016; Nistor et al. 2017b; Shekhar et al. 2020; Park et al. 2021; Mascarenas 2022; Cinar et al. 2022; Majtan et al. 2022; Moris et al. 2023) that aim to understand the hazard at a reduced model scale. If done carefully, extrapolation of model-scale conclusions to a prototype tsunami scale may be possible. A handful of researchers have recently sought to recreate experiments numerically, as in Yang et al. 2017a, Majtan et al. 2022, and Hasanpour 2023, to eventually assist in the extrapolation to prototype scales. These efforts logically lead to various attempts to lay the ground-work for scale-invariant and probabilistic design frameworks for tsunami-driven debris-field hazards (Bonus et al. 2022; Nistor et al. 2017a). Although there is no consensus on the most effective and efficient numerical approach to tsunami debris, as identified by Takabatake et al. 2021 and Nistor et al. 2017a, two-way coupled solid-fluid models, smoothed particle hydrodynamics (SPH) and material point method (MPM) schemes show promise and have enjoyed growing popularity. Furthermore, the apparent need for high-resolution 3D DFSI models has prompted high-performance multi-CPU and GPU adoption in open-source tools, notably CB-Geo MPM (Kumar et al. 2019), Karamelo (Nguyen et al. 2023), DualSPHysics (Domínguez et al. 2022), Claymore (Wang et al. 2020b), and now ClaymoreUW.

A critical review of the state of the art in tsunami-debris modeling was provided by Nistor et al. 2017a. Key conclusions include the need to: **(i)** determine appropriate experimental and numerical scales for turbulence (She and Leveque 1994) and debris dynamics (Nistor et al. 2017b), **(ii)** account for robust hydrodynamic boundary conditions numerically, **(iii)** provide rigorous high-resolution studies for 3D model calibration, **(iv)** investigate fundamental debris-debris and debris-fluid behavior in the context of debris entrainment and momentum transfer, **(v)** support of numerical probabilistic approaches (i.e., develop high-performance tools to simulate at higher frequencies) due to the stochastic nature of debris motion, **(vi)** and numerically

characterize the demand of multiple debris impact and damming events on structures.

In Yang et al. 2017b the authors employed MPM to examine the impact of tsunami-driven debris on bridge superstructures. The study found that the influence of contact area and impact eccentricity on debris-structure interaction is mitigated with an increase in the debris longitudinal ratio, and highlighted that tsunami-induced debris impacts can increase impact forces by up to 35% compared to debris impacts in air estimated using empirical equations (Aghl et al. 2014a) for equivalent debris velocities (Aghl et al. 2014b). However, this finding applies to cases where a buffer of water does not develop between the debris and structure at impact and thus is primarily limited to highly buoyant debris impacting elevated structures. When a fluid buffer is observed between the structure and debris, as in Mascarenas 2022 and Shekhar et al. 2020, loads are experimentally recorded to decrease relative to in-air calculations. This suggests that while the debris-fluid interaction can increase loads, DFSI often occurs across a difficult to predict fluid buffer between debris and structures at impact and may dampen loads in a scale-dependent way, suggesting a need for further numerical DFSI study.

Aslami et al. 2023 applied SPH to solve shallow-water equations (SWE-SPH) with solid boundaries in DualSPHysics coupled with the Discrete Element Method (DEM) to simulate and predict debris behavior and hazards in tsunami-like conditions, providing good agreement with experimental data and analytical equations for simple debris materials and geometries.

Wu et al. 2014 introduced a two-way coupled dynamic numerical model for moving solids in free-surface flows. They utilized a Large Eddy Simulation (LES) model, the Volume-of-Fluid (VOF) method for tracking the free surface, the Discrete Element Method (DEM) for debris displacement and rotation, and it models the fluid response from solid motion via cell-face velocity with partial-cell treatment (PCT). The effectiveness is validated through two laboratory experiments, showing accurate predictions of the trajectory of moving blocks in a water tank, similar to our own debris-fluid interaction validation case in Section 6.8.

Hasanpour et al. 2021 coupled Smoothed Particle Hydrodynamics with the Finite Element Method (SPH-FEM) model to simulate the interaction of single debris in a wave-flume environment, as well as loads on scaled coastal structures. It found acceptable accuracy across variables, with deviations of maximum wave height and fluid velocities between 4-22% from experimental tests, and predicted debris impact velocities were 20% higher. The maximum predicted impact forces of the debris on the structure deviated 14- 25% from the experimental data. Interestingly, debris allowed to tilt obliquely to streamwise flow produced higher impact velocities but smaller impact forces, approximately 56% less, compared to restrained debris due to oblique impact angles as observed in Yang et al. 2017b. The study suggests considering water depth and velocity range in developing future debris load equations, acknowledging the need for high-resolution 3D models to better account for the debris' rotational movement. Recently Hasanpour et al. 2023 extrapolated this work to study single 3D debris in their SPH-FEM approach and achieved good results regarding overall motion of the debris as well as resolving interesting trends in multi-directional loading on bridge structures (i.e. streamwise, vertical, transverse). Their work is potentially the most comprehensive numerical study of single debris impacts on a bridge superstructure as it has nuanced implications regarding the debris' ability to inherent an intensity measure (i.e. flow-speed, flow-depth) from a wave, manifest it as its own intensity measure (i.e. debris momentum, debris persistence through the slowing mass of water at submerged structural faces), and impart it onto a structure with preliminary trends appearing for multiple axis of loading that designers must consider. However, this work still studies single-debris without providing guidance for extrapolation to multiple-debris loading that can occur in real events. Our work aims to fill this gap for both ordered and randomly configured groups of debris, allows for preliminary extrapolation to prototype scales, and is to eventually give numerical insight to elasto-plastic constitutive effects of debris and structures. Combining insights from Hasanpour 2023 and this work may provide a novel framework to understanding wave-driven

debris and debris-field hazards on raised structures for multiple axis of loading.

Participants in a tsunami debris modeling workshop compared different approaches to numerically simulate tsunami debris (Takabatake et al. 2021). Accurate modeling of the tsunami-induced flow field was identified as a necessary, but not sufficient, condition for debris motion. This was especially notable for debris dragging and bouncing across frictional topography (i.e. large, dense debris not fully lifted by the inundation), as the changing fluid drag, ground friction, and inertia coefficients affected the speed and trajectory of the debris. Debris-debris interaction was a significant phenomenon worth refining, as models tended to incorrectly predict the lateral spread of debris. Two-way coupled simulation, which include the debris-fluid interaction drag force, were identified as favorable for studying larger debris. They suggested incorporating 3D models and exploring probabilistic approaches to address variations in debris trajectory.

All of these studies have pushed the field of numerical tsunami debris modeling forward in their own right. However, challenges persist in high-resolution 3D models (100+ million numerical bodies), multiple debris interactions in fluid-structure deformation and topology changing environments, hyper-elastic and elasto-plastic debris and structural material models, and handling arbitrary debris and structure geometries. This work aims to address some of these issues, demonstrated across consistent results in ordered debris-fields, randomized debris-fields, and up-scaled debris-field hazards. Our high-performance Multi-GPU Material Point Method approach, implemented in the open-source ClaymoreUW, holds great promise for addressing these challenges and advancing the field of numerical tsunami-driven debris-field hazards.

7.2 *Facility Overview*

Large-scale testing was performed at Oregon State University's Large Wave Flume (OSU LWF, Fig. 7.3), a facility for two-dimensional characterization of ocean-shore wave advancement (Lomonaco et al. 2018). Breaking and non-breaking solitary waves were studied as they progress

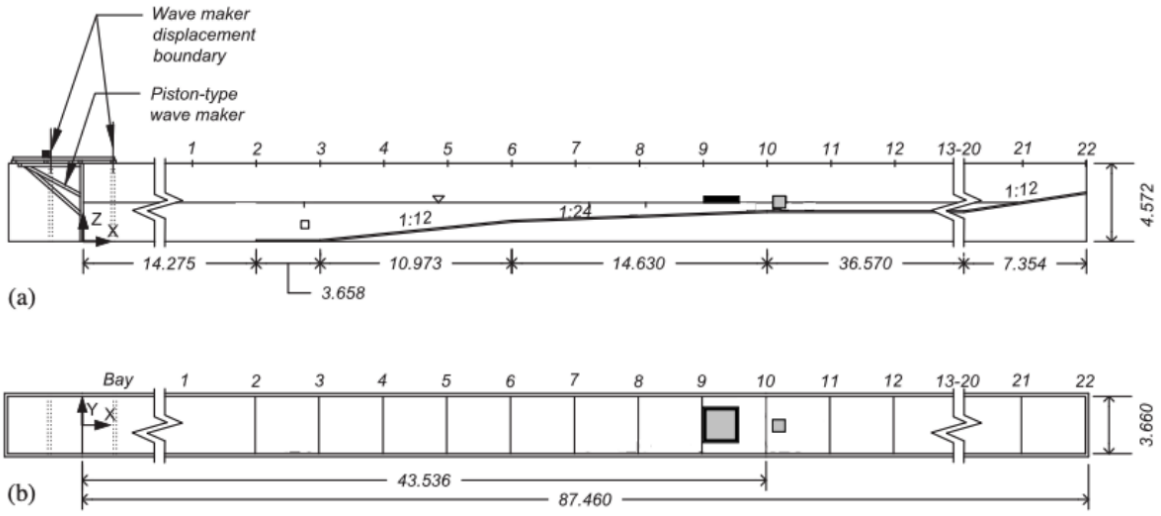

Fig. 7.4. OSU LWF bathymetry. Bathymetry diagram of the OSU LWF adapted from Winter 2019. (a) Side-view. (b) Plan-view. Adapted from Alam 2019 and Mascarenas 2022.

up a series of ramps, representing the shore, before impacting a scaled structure. Previous debris-field studies by our group were performed here by Shekhar et al. 2020. Our most recent study was by Mascarenas 2022 and has shown promise for resolving structural demand trends with not just debris count, but statistical aspects of debris-field configurations as well (e.g. field density and orientation).

Full documentation of the OSU LWF facility specifications and usage can be found in Winter 2019, Winter et al. 2020, Shekhar et al. 2020, and Mascarenas 2022. Here the basics of the flume, placement of sensors, debris, and the raised structural box are summarized. Figure 7.4 presents a schematic of aforementioned details.

Hundreds of fluid-driven debris-field impact and damming experiments by Shekhar et al. 2020 and Mascarenas 2022, covering a swath of ordered and randomized debris configuration, have been executed. Performed at a large wave flume (OSU LWF, 100m length), these trials aimed to **(i)** observe large model scale-effects (Froude length ratio of 1:10), **(ii)** compare different

debris-field configurations (longitudinal versus transverse versus random), **(iii)** reduce bias towards small changes in facility usage (e.g. water-level), **(iv)** quantify the medians and interquartile ranges of stochastic loading from chaotic debris-field impacts, and **(v)** determine the relationship between debris-field impact loading and initial debris-field fabric parameters (e.g. field density/porosity, debris count, orientation) through the analysis of hundreds of experiments.

Both experimental works used a raised aluminum structural box to represent a scaled building, seen in Figure 7.5. The box was assumed to be effectively rigid, although it was further stiffened during the work of Mascarenas 2022. It was equipped with load-cells and pressure sensors to measure structural demands, with the former being the focus of our numerical replication. Water velocity and depth was measured at key locations in the flumes. Varied number of debris (1 - 24) made of High Density Poly Ethylene (HDPE, buoyant on water) were used to represent fields of roughly 1:10 Froude scale automobiles, or roughly 1:20 Froude scaled shipping crates. Debris were placed in ordered and random configurations to create debris-fields at a distance of two structure widths upstream from the structure. Collisions of debris-fields with the structures were recorded by load-cells in three dimensions, though only streamwise loads are considered herein. Debris-field case medians and interquartiles ranges were reported. The aim of our work is to be able to broadly capture trends in the medians across the chaotic range of behavior debris-fields exhibit while ideally staying within interquartile ranges in individual cases and within outlier bounds (+/- the IQR from the interquartile bounds) and experimental trial envelopes (minimum and maximum reported values) for all cases we numerically simulate. Achieving this would demonstrate the stochastic viability of our MPM approach in the modeling of wave-driven debris-field impacts on raised structures.

Basic flume model and tsunami prototype specifications are included in Table 7.1. Note that scales were chosen to maintain Froude similitude and for the analysis of primary debris-field inundation impacts. The raised structure's specifications are found in Table 7.2. For further detail,

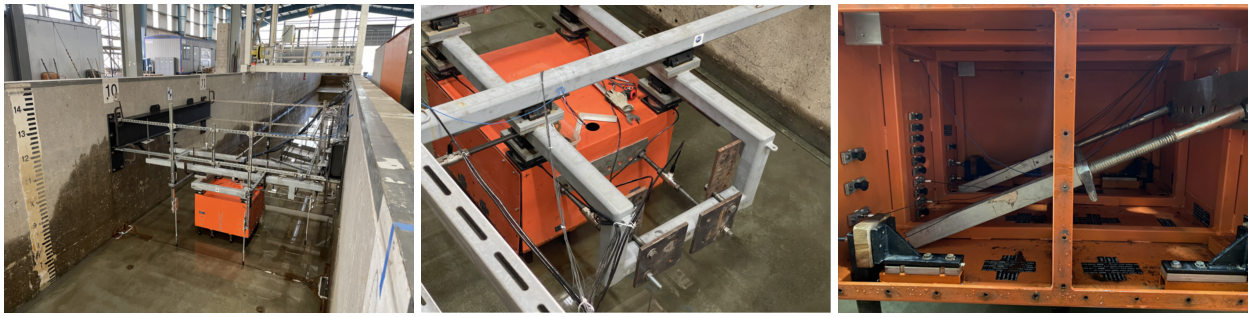

Fig. 7.5. Raised structural box used in OSU LWF debris-field tests. The raised "orange box" from Winter 2019, Shekhar et al. 2020, and Mascarenas 2022 which is used to measure wave and debris loads. Interior stiffened with cross-bars.

Table 7.1. Specifications of the wave event in the prototype and model.

Parameter	Prototype	Model (OSU LWF)
Flume Length (m)	1000	100
Flume Width (m)	36.5	3.65
Flume Height (m)	-	4.6
Standing Water Level (m)	20.0	2.0
Max Flow Velocity (m/s)	0.5	4.0
Inundation Event Duration (s)	189	60.0
Debris-Field Event Duration (s)	31.6	10.0
Flow-Speed at structure (m/s)	4.1	1.3
Flow-Depth at structure (m)	5.28	0.528
Flow-Depth above SWL at structure (m)	2.78	0.278

the OSU LWF model's structural box, referred to as the "orange box", is also present previous studies including Shekhar et al. 2020, Winter et al. 2020, and others. Details on sensor location and rigidity of the mounting is discussed briefly but is best referenced in the original experimental work of Mascarenas 2022 with additional details in Winter 2019.

Prototype and model simulation parameters are stated in Table 7.3, with further specifications provided when our numerical digital twin of the wave flume is discussed later in this chapter. All simulations use our Multi-GPU MPM code ClaymoreUW (Chp. 4).

In both experimental endeavors, a set of hydrodynamic (no-debris) trials were performed to find a

Table 7.2. Specifications of the raised structure in the prototype and model. Raised structural box, i.e. Orange Box, used in the OSU LWF experiments is assumed to be effectively rigid.

Parameter	Prototype	Model
Structure Length (m)	10.16	1.016
Structure Height (m)	6.15	0.615
Structure Width (m)	10.16	1.016
Dist. from Wave-Origin (m)	438.0	43.80
Elevation above Seabed (m)	20.0	2.00
Elevation above Ground (m)	2.50	0.25
Elevation above SWL (m)	0.00	0.00
Adjacent Gap-Widths (m)	13.2	1.32
Blockage Ratio	3.6	3.6

Table 7.3. OSU LWF simulation specifications.

Parameter	Prototype	Model (OSU LWF)
Fluid Bulk Modulus (GPa)	2.2	2.2
Debris Young's Modulus (GPa)	0.8	0.8
Grid-Cell Spacing (m)	2.5	0.025
Particles (No.)	150,000,000	150,000,000
Time-Step (s)	2.4e-5	7.5e-6

baseline for fluid behavior and demand on the structure without debris interaction. Fluid event consistency was found to be very good as the interquartile range of the wave-load was only 6% of the median. Two wave-modes were studied: "Broken" and "unbroken" (note that both waves technically break, but the former does so more dramatically) with driving paddle displacement profiles shown in Figure 7.6 (error function piston stroke of 4 meters with a scaling factor of 500 for the unbroken case, see Winter 2019 and Mascarenas 2022). Respectively, the cases represent a tall wave (1.3 meter model, 13 - 26 meter prototype) that has a plunging break as it travels up the inclined ramps well before the structure, and a small solitary wave (0.2 meter model, 2 - 4 meter prototype) that breaks in a spilling mode a few meters in front of the structure. This work is primarily focused on the latter case, which we will find to be well replicated numerically in explicit MPM, although with some limitations in pressure field smoothness compared to incompressible formulations.

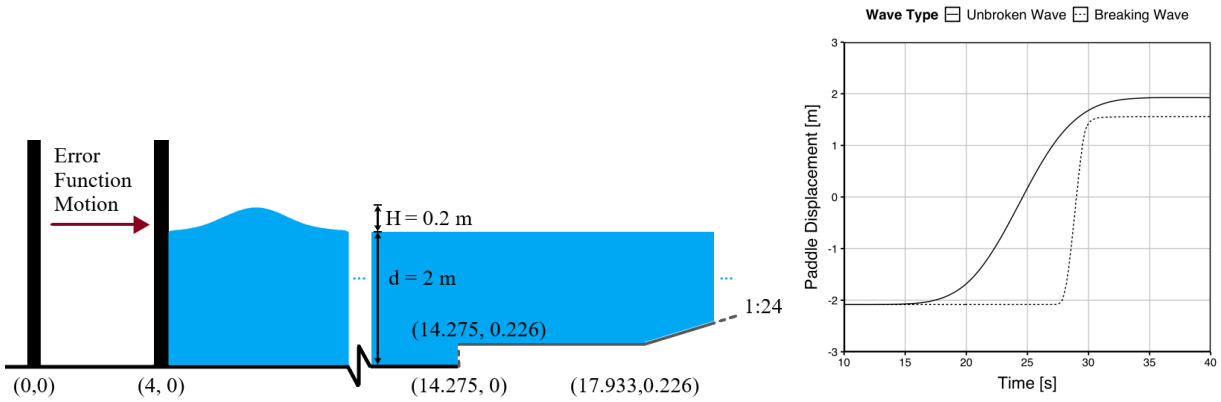

Fig. 7.6. OSU LWF paddle movement. Wave-maker paddle movement for unbroken and broken waves.

Before seeking to numerically recreate the wave-only load, we can perform an analytical sanity-check to ensure that experimental loads from Mascarenas 2022 are valid. The

hydrodynamic drag force (F_s) on the raised structural box may be predicted as:

$$F_s = \frac{1}{2}\rho_w C_s A_s u_w^2 \ , \tag{7.1}$$

where ρ_w is water density (1000 k/m^3), C_s is the structures coefficient of drag, A_s is the structural box's submerged area perpendicular to stream-wise flow, and u_w is the average velocity of the fluid profile acting on the structural box. We assume that the partially submerged depth of the box at peak wave-loading is 0.278 meters (measured at ultra-sonic wave-gauge 3 in Mascarenas 2022), the average flow velocity is 1.4 - 1.0 m/s (near-structure ADV flow speed from Shekhar et al. 2020 and Mascarenas 2022), and that the coefficient of drag is 0.8 - 1.45 (standard for a fully submerged cube and halfway between a rectangular box and fully submerged cube). We may then calculate an approximate hydrodynamic load of 219.0 - 201.6 Newtons using Equation 7.1. This is within good tolerance of the experimentally measured median load of 214.4 Newton with a 6 Newton interquartile range and thereby suggests that the experimental structure was effectively rigid and load cells had only minor bias / drift across tests. However, note that some small amount of drift was still observed in experiments regardless of instrument calibration.

Debris was introduced into the flume 2.0 meters away from the structure's leading face, measured to the debris placement-frame's downstream face. Debris geometry and composition are shown in Figure 7.7 and 7.8. They are nearly identical to previous work by Shekhar et al. 2020. Steel plates were attached to debris for mounting instruments and holes were drilled to preserve neutral buoyancy.

Varying sets of debris were tested. Most were done with the unbroken wave due to its consistent nature and non-dominance over debris loading, hence we choose to study only the unbroken experiments in numerical detail. Any broken wave debris experiment had a corresponding unbroken case for comparison, so in future works we may pursue parity in replicating large

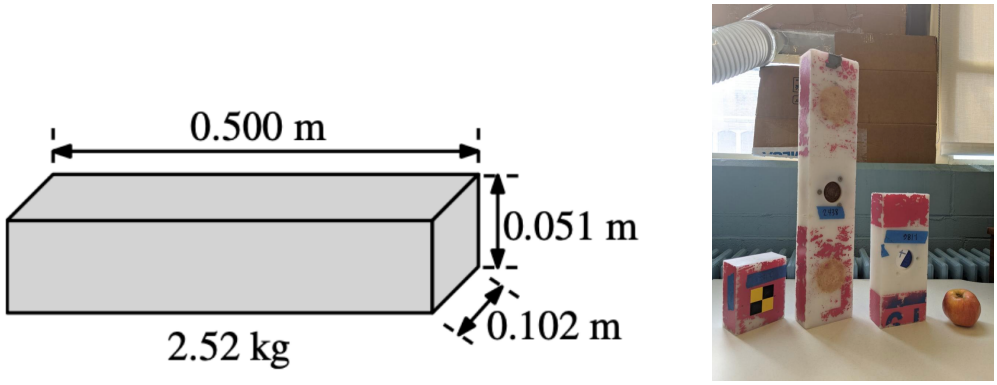

Fig. 7.7. OSU LWF debris schematic. (Left) Schematic of typical debris used in OSU LWF tests. Note that debris were modified with a steel plate and had holes drilled. Courtesy of Alam 2019. (Right) Debris used in experiments by Mascarenas 2022. Only the longest debris was used in our MPM simulations.

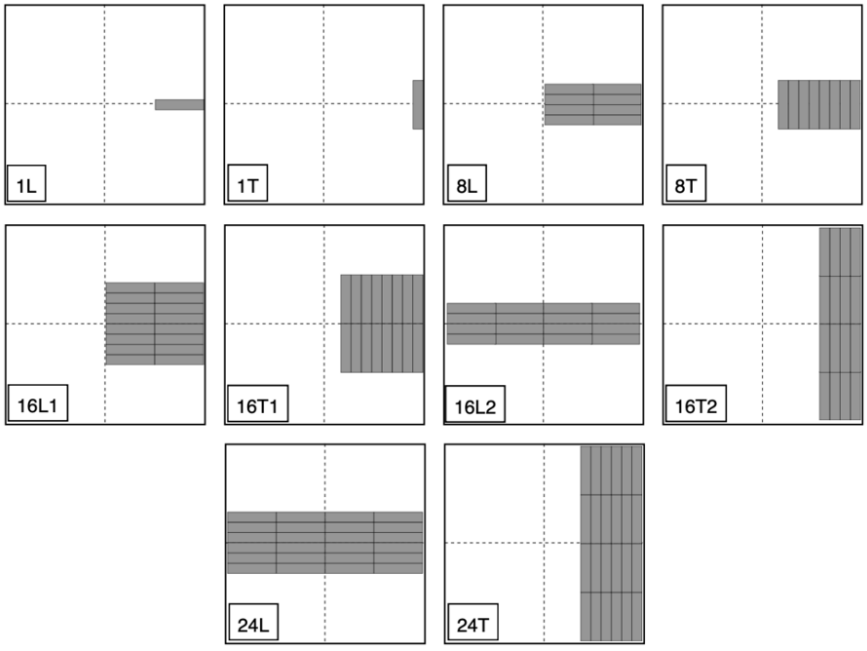

Fig. 7.8. Debris-field ordered configurations used in OSU LWF experiments. Adapted from Mascarenas 2022. Fluid flows from left-to-right. The structural box is two meters to the right of each debris field downstream face. Note that these configurations are not identical to cases in Shekhar et al. 2020 but there are some that are shared.

breaking waves to the same standard that we achieved numerically for unbroken waves. Aforementioned numerical simulations of the unbroken wave are described in the following section.

7.3 *Numerical Approach*

The greatest challenge in modelling the OSU LWF is its size. In the past, MPM and SPH models alike attempt to reduce memory demand by using truncated models in Yang 2016 and Shekhar et al. 2020 (i.e. only part of the flume was simulated, with complex boundary conditions forming in/out-flows) or 2D representation as in Hasanpour et al. 2021.

For this project our group has ran 50+ cases of full-length 3D, start-to-finish simulations of the OSU LWF, not compromising on representation while continuously pushing resolution to greater heights. This is achieved computationally in our Multi-GPU MLS-MPM code ClaymoreUW. We do, however, use a stream-wise mirrored domain when applicable, i.e. if the debris-field configuration is mirrored about the center-line in experiments. Hydrodynamic checkpoints are also used to save time when studying the exact same underlying wave over dozens of cases. These are purely for expedience in the hundreds of simulations we've ran throughout this study and are not at all necessary. As we have successfully simulated the OSU LWF with a billion fluid particles, the typical 75 - 150 million particle simulations (mirrored and unmirrored) are certainly not the limit of what is achievable.

In our numerical study of the OSU LWF debris-field experiments, we apply the Material Point Method (Sulsky et al. 1994, MPM) with modifications for computational speed, i.e.e the Moving-Least-Square Material Point Method (MLS-MPM, Hu et al. 2018), and an assumed deformation volumetric antilocking scheme (F-Bar antilocking, Zhao et al. 2023) for improved pressure fields in our flume study herein for all simulations. Grid-cell length Δx is 2.5 centimeters (1 inch). Particles-per-cell (PPC) is 8 for both water and debris. The Courant-Friedrichs-Lewy

(CFL) number is set to 0.45 to maintain stability, corresponding to 7.5e-6 second time-steps, Δt, for the model scale. Three NVIDIA A100 GPUs host the water particles, while one of the GPUs also simulates the debris-field particles. Pertinent information regarding the digital-twin model of the OSU LWF we have developed in ClaymoreUW Multi-GPU MPM is given in the following sections.

7.3.1 Flume Bathymetry

Flume walls used rigid walls with separable velocity contact applied on grid-nodes, which are set according to Table 7.1. **Flume bathymetry panels** (i.e. ramps) are recreated on the MPM grid with a linear decay layer, as described in Yang 2016, applied to prevent "stair-stepping" effects of a coarse Cartesian grid defining tilted surfaces. The bathymetry is defined according to Figure 7.4, with an added condition that the "gap" below the raised slab in Bay 2 is adjusted to prevent water from leaking under the ramps. In the experiments, the space beneath the ramp was filled with water so leakage under the slab was not a concern and had little hydrodynamic impact as noted in Winter et al. 2020. For the simulation, we do not want to model the water under the ramp because it would double simulation sizes while having no discernible hydrodynamic effect. We initially capped the gap in Bay 2 to avoid flow under the ramp, but found that simply raising the panels prior to Bay 2 to 0.226 meters (i.e. flush) to allow a fully flat bathymetry prior to the ramps gave a far less disturbed MPM fluid fabric and thereby a cleaner flow up the ramps. This slight decrease in water depth in the far-field has a very minor affect on the solitary waves far-field speed, but it should be noted. The ramp is made of concrete slabs but we do not account for any material friction or boundary layers, although MPM is seen to create somewhat thickened boundary layers coming off the convex ramp joints automatically.

7.3.2 Wave-Maker

The **wave-maker** in the OSU LWF is a linear actuator (i.e. piston). In MPM, its modeled as a velocity boundary condition on the MPM grid that moves according to actual experimental motions of the wave-maker. Starting from the at-rest position (wave-maker neutral), velocities are fed into grid-nodes behind and on the paddle face. As time accrues, the displacement and velocity of the paddle updates. A separable condition is set at the paddle face to allow water particles to decouple from it seamlessly if they are moving quicker than the paddle in the forward stream-wise direction. Note that the coarseness of the grid can result in coarse behavior of the wave-maker. Ideally, a dual-grid approach would be used to better fit small movements, or a finite-element mesh/Lagrangian moving boundary used to recreate the wave-maker at sub-grid resolutions. The wave-maker operates at the frequency of the experimental data (e.g. 1200 Hz).

7.3.3 Structural Box

The raised **structural box**, i.e. the orange box as it is called in some related papers, is modeled according to specifications in Table 7.2. It is taken to be rigid– No elastic structural responses are considered. The interior grid-nodes of the box are set to sticky contact. The exterior faces are separable-slip contact in their respective axis. Overlap on exterior face edges XY and XZ are overridden to be an X boundary condition. Overlap on exterior face edge YZ and corners XYZ are set to a free boundary condition to reduce the effect of MPM's grid-constrained boundary layer and non-turbulent pressure drag on the box. These are all velocity boundary conditions applied directly to the MPM grid-nodes. The box's influence as a boundary condition is affected by grid-coarseness, i.e. very coarse grids may not perfectly conform to our box dimensions. There are techniques for applying non-coinciding boundary conditions to MPM grids which are not taken here because the box matches the grid fairly well at our grid resolutions (less than 5cm). Structural box columns are not modeled here as we are interested in peak debris-impacts.

Damming may be the subject of a future MPM study, at which point we may revisit appropriate thin-column boundaries.

7.3.4 Water

Water at-rest conditions is recreated in simulations via the recorded still water level (SWL) from respective experiments. No other field or particle variable is applied initially (e.g. pressure) as all particles are considered to be undeformed at initialization, as gravity loading will develop said pressure fields when the simulation runs. Water in the OSU simulations is divided across GPU partitions with partition surfaces transverse to stream-wise flow to reduce total area and thus required size of memory transfers per time-step. This partitioning provides a means to reduce computational load on a single GPU by adding another level of parallel behavior, and to avoid memory restrictions on a single GPU's global memory pool. Water is typically split across three GPUs on the Lonestar6 HPC system, with two modeling the high-volume far-field water and one modeling the shallower near-field water with debris. Stiff fluids are exposed to damping and locking from MPM which can be somewhat alleviated with PIC-FLIP mixing factors and reducing PPC, though we use Simple F-Bar volumetric antilocking by Zhao et al. 2023, with novel modifications for computational expedience and stability (see Sections 5.8, 5.9, and 5.10), to achieve the same effect. In future work we may investigate reintroducing FLIP or ASFLIP to improve water flow around the structural box, but here we choose to reduce variables and increase stability at potentially the cost of some debris motion damping and damped flow around the structural box, though neither effect was notable enough to invalidate our results. Water is modeled as a weakly-compressible isotropic fluid using the Murnaghan-Tait equation of state with viscous shear stress (Pradhana et al. 2017, Murnaghan 1944, and Tait 1888. See Section 5.4), implemented as our novel optimized, precision-accelerated J-Bar fluid form (PA-JB Fluid, see Section 5.12) for use in high-performance GPU computation. Density is 1000 kg/m^3 which is

reasonable for fresh water at sea-level in the neighborhood of 20° C. Bulk modulus (i.e. fluid incompressibility) is 2.2 GPa to match typical values of minimally aerated water. Viscosity is 0.1 centipoise (0.001 Pa*s). Standing water level is 2.0 meters from the flume basins minimum at the wave-maker. Thickness of water layer below the structural box is 0.25 meters, though the total surface elevation is still 2.0 meters.

7.3.5 Debris

Debris are taken to be solid rectangular prisms made of buoyant HDPE plastic of size 0.5 x 0.1 x 0.05 meters. We neglect the small cut-outs for instrumentation mounting/ballasting in Mascarenas 2022. Material behavior is approximated using the fixed-corotated hyper-elastic material model (Jiang et al. 2016, see Section 5.4). Debris possess a uniform interior particle spacing of 0.0125 meters for a particle-per-cell count of 8 on the 2.5cm grid-cells. Higher values (e.g. 27, 64) can improve dynamics of debris pressure waves but we chose to keep parity with the PPC values of the water to reduce evaluated variables. Density of debris is 981 kg/m^3, again noting that we do not account for the small holes drilled into debris or the metal discs that were inserted in the work of Mascarenas 2022. Poisson ratio is set to 0.4 and young's modulus is 0.8 GPa to match common values for HDPE plastic.

7.3.6 Debris-Fields

Debris-fields are all initialized on the water's surface, i.e. , in configurations designated in Figure 7.8 which are placed two meters from the structural box. Debris are spaced four grid-cells apart (10 centimeters) to improve debris-debris interaction behavior during transport. Note that experiment video-files taken by Shekhar et al. 2020 and Mascarenas 2022 show that initial debris-field drift may disturb the fabric so there is natural variability, hence initial packing of numerical debris is not required to feature no gaps at all. No modifications are made for the

methods of placing debris from the experiments, although the effect of the placement methods has been investigated in experimental analysis by Mascarenas 2022 regarding a magnetic dropping and mechanical lifting mechanism. Our simulations are most similar to the lifting frame as we assume debris begin in an effectively static state on the fluid surface. Debris have been tested as MPM bodies, MPM-FBAR bodies, MPM-FEM bodies, ASFLIP-FBAR bodies, and ASFLIP-FEM bodies, but only results for the case of MPM debris are shown to avoid use of any subjective antilocking or velocity mixing variables (note that the water, however, uses F-Bar antilocking). A fixed-corotated material model is used to model the HDPE plastic (Sec. 5.4). No adjustments are made to account for the stiffening of the HDPE debris with a metal plate or for the small instrumentation holes drilled in them. However, this could be done by selectively altering material properties of material points and/or finite elements in the debris or with a slightly higher-resolution debris model. We assume that effects on moment of inertia are negligible.

7.3.7 Free-Surface Wave-Gauges

Wave-gauges 1, 2, and 3 (WG1, WG2, WG3) from Mascarenas 2022 are recreated numerically as an output of the highest MPM particle's elevation in a specified sensor domain, which are +/- Δx of each experimental wave-gauge's center-position streamwise and transversely. They are operated at sample-rates of 120 and 1200 Hz to match conditions of respective experiments and placed according to the experimental schematics. This method is exposed to a small amount of noise due to grid-coarseness and PPC magnitude but we observe smooth gauge curves for our 2.5cm grid-cells with 8 PPC.

7.3.8 Load-Cells

Load-cells are recreated numerically as an output of summed force on the leading face of the structural box. We have only investigated stream-wise forces, but vertical and transverse forces are

available and may eventually be pursued similar to the work of Hasanpour et al. 2023 on multi-directional bridge loads. Force at each grid-node on the specified structural face is calculated through momentum change assuming a separable rigidity condition (i.e. all positive stream-wise momentum is set to zero over time-step Δt) and then added together. Numerical load-cell output frequency is identical to experimental instruments (1200 Hz). We define the boundary for the load-cell measurements as all contained grid-nodes on the structural box's front face. Note that load sensors on the structure are somewhat sensitive to the method of defining the structure as a boundary condition, as over-representing the geometry of the box on the MPM grid can lead to a nonlinear increase in the arrested fluid volume at the face of the structure during loading.

7.3.9 Breaking Wave Experiments

Broken wave tests have yet to be fully simulated, but we have replicated the basic hydrodynamic elevations and forces. Standing water level (SWL) is set to 1.85 m. These tests are lower priority, since fewer debris were used in them, and are also more suited to turbulence modeling methods. In preliminary runs we have found that MPM, using the Affine Particle-In-Cell scheme (APIC) and FLIP, can simulate the breaking behavior due to its improved maintenance of angular velocity, as seen in Figure 7.9 where a plunging break forms. Grid-cell sizes can be fairly coarse (e.g. 5cm) and bulk modulus can vary between 10 to 100% of real water with fairly minimal change in the far-field, though discrepancy grows as the breaking wave travels up the inclined bathymetry and collides with the structural box. Note that high bulk modulus values (e.g. 2 GPa for real water) can introduce errors in the pressure field at high particle-per-cell counts which in turn can negate the breaking behavior, while dropping PPC too low may prevent expression of a breaking wave at all. This can be countered by changing grid-resolution, using Fluid-Implicit-Particles (FLIP, Section 5.6.3), or by applying volumetric antilocking (F-Bar antilocking, 5.7) in some cases. If the wave fold-over gap is artificially maintained after the break (from the inherent sticky contact of

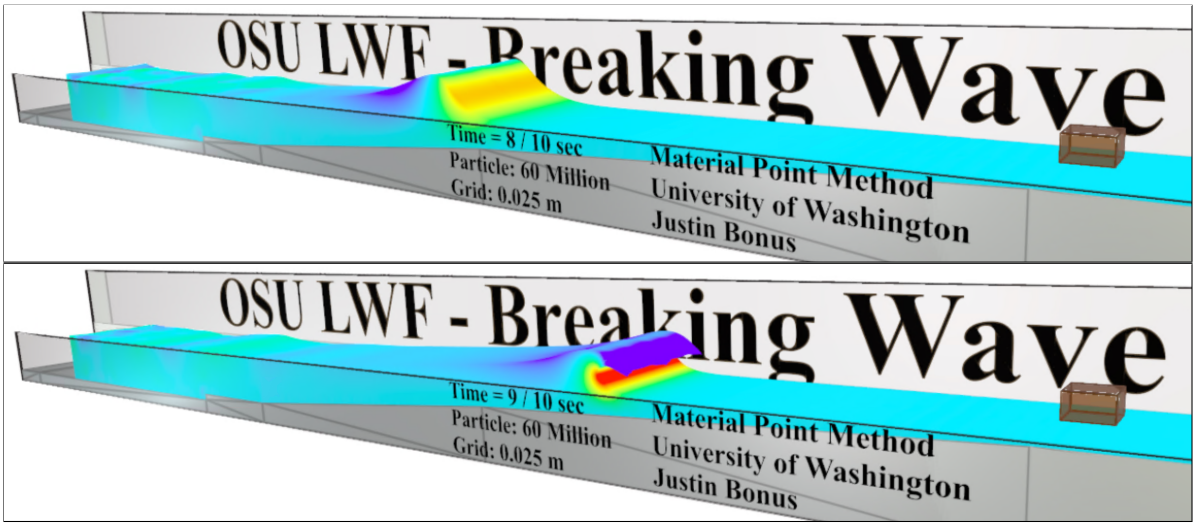

Fig. 7.9. Breaking wave simulations in OSU LWF digital twin. Open-source digital twin of the OSU LWF, implemented in ClaymoreUW Multi-GPU MPM, simulates a breaking wave based on Winter et al. 2020 and Mascarenas 2022.

MPM particles) then FLIP or ASFLIP can help to resolve this. Note that both may alter the pre-break and post-break wave dynamics slightly, so it is best to restrict them to the portion of the domain where breaking is desired.

7.3.10 Unbroken Wave Experiments

Unbroken wave tests are recreated in depth. Standing water level (SWL) is set to 2.0 meters and the wave-maker motion is set by the experimental paddle motion. Simulations run until 45 seconds (experiment time) to capture the long paddle motion and subsequent wave travel. Typically, we use an MPM checkpoint file, which contains a saved state of a simulation which can be restarted/resumed from, for a hydrodynamic simulation at 27.5 seconds for our debris simulations. This allows us to just simulate 17.5 seconds. Results are strong even for an explicit weakly-compressible fluid model, which are commonly thought to be error prone. However, resolving the small wave (20-25 cm) in a 100 m flume requires high resolution (e.g 2.5 cm

grid-cells). We may investigate cells of 1.25 to 1.75 cm more in the future, the latter shown in Section 7.7.3 for one billion particles, as they are achievable on modern GPU hardware.

7.4 *Hydrodynamic Simulations*

Hydrodynamic cases are the focus of preliminary validation efforts between the numerical and experimental results for the OSU LWF. The OSU LWF wave we primarily study is approximately an "unbroken" solitary wave far from the structural box (far-field), but it's front slows as it travels up the ramp bathymetry relative to its tail and thus creates and build-up of fluid mass in the bore front. This produces an asymmetric wave with a steep bore-front.

This is a better approximation to a tsunami wave than a pure solitary form. We note that the time-scale that our model wave operates on is far smaller than the time-scale of real tsunami inundation waves. As argued by Madsen et al. 2008, a dam-break wave-maker is more effective for producing these sorts of time-scales, and work by Goseberg et al. 2013, Goseberg et al. 2016, and Stolle et al. 2016 also show that tsunami waves may be better modeled in flumes with non-piston wave-makers. However, our prototype wave scale falls reasonably within 10-15 seconds (loosely speaking, as there is not "one" period for a irregular wave-train or asymmetric raised front solitary-like wave). This range is arguably appropriate for studying first and max peak load debris-field impacts, which tend to be primarily controlled by the first inundating wave of a tsunami as opposed to the tail-end of an event's time-scale. Future work may address concerns of the wave's time-scale by adjusting wave-generation numerically and evaluating if results are significantly affected.

To validate the basic hydrodynamics of our MPM approach in the OSU LWF unbroken wave experiments, we must observe the following key properties in simulations:

Wave generation. Numerical paddle movement must be identical to the experimental paddle. The wave generated at the face of the paddle must match the one observed in the wave-flume facility.

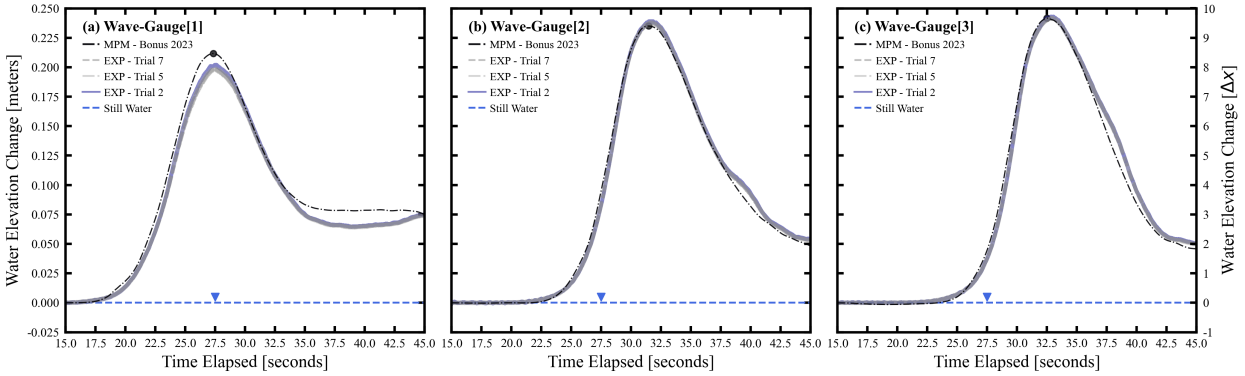

Fig. 7.10. Free-surface elevation at wave-gauges in the OSU LWF. Hydrodynamic results (no debris) of wave evolution along the OSU LWF. Experiments by Mascarenas 2022 closely match our MPM simulations.

This is automatically met by use of a moving velocity boundary in MPM that tracks experimental recordings of wave-maker motion. We also observe that MPM particles do not pass-through the wave-maker as they are moved realistically by the piston face.

Far-field wave properties. Consensus with wave-gauges readings placed deep in the flume demonstrates validity of far-field wave-manifestation following paddle driving. Figure 7.10 shows the wave-gauge 1 in three experimental hydrodynamic trials and our hydrodynamic simulation match very closely, with error within half an MPM grid-cell (1.25 cm). Note that our hydrodynamic simulation did not feature the orange box as the experiments did, so there may be some deviation due to the reflected wave in the later portion of the curve.

Near-field wave properties. Numerical wave-gauge readings must match experimental results as the wave moves up the bathymetry ramp, growing or sinking appropriately for the unbroken and broken case. Figure 7.10 demonstrates the ability of MPM to replicate free-surface readings at wave-gauge 2 and 3 as the wave travels up two inclined bathymetry panels. Deviation is constrained to half an MPM grid-cell prior to wave reflection of the structural box, which was not included in our hydrodynamic simulation. **Hydrodynamic loads on the structural box.**

Hydrodynamic forces on the structural box are ultimately a function of wave-velocity and height for the hydrodynamic case, but they must agree well with experimental data. Hydrodynamic loads (no debris) on the structural box front face in the stream-wise direction are plotted in Figure 7.11. We observe that our force profile arrives later and leaves earlier, however this is anticipated as we raised our simulated structural box three grid-cells above the water's surface to prevent erroneous readings and excessive fluid buffering at the structural box's face. This hydrodynamic reading does not perfectly match the load magnitude of the experiments but it is within 25% and features the characteristic profile of the force-loading so we find it to be an acceptable result. This is because our aim is to study debris-field impact forces which will be an order of magnitude larger than wave-only loads, so the discrepancy shrinks to roughly 2.5%. Impulses for experiments between times 30 and 45 seconds are 1217 +/- 17 N * s, whereas the original MPM curve is 1182 N * s, so the primary time-interval of concern has equivalent impulses between simulations and experiments. Hydrodynamics should also be contextualized with the fact that this is a very small box located far down a massive flume, and that we model the entirety of the wave from the generating piston, as it travels up the inclined ramps, when it undergoes a breaking-spill, and then until it impacts and passes the box. This gives significant room for error, which we have minimized in our simulations. There is some sensitivity to the mixing ratio used for the fluid's F-Bar volumetric antilocking (see Section 5.8), so we settled on using a value of 0.999 for all simulations to eliminate bias in individual simulations.

We also plot a boundary stabilized MPM result in Figure 7.11, which is seen to have a smoother but smaller force profile. This is due to the stabilized boundaries not disturbing the MPM fluid particles as notably as they travel up the flume bathymetry, preventing spurious increases of wave elevations which means the wave height will be accurate and thus the force will be lower when the wave hits the raised box compared to the artificially steepened MPM wave. The stabilized hydrodynamic result is used later as a checkpoint for results regarding extrapolating to prototype

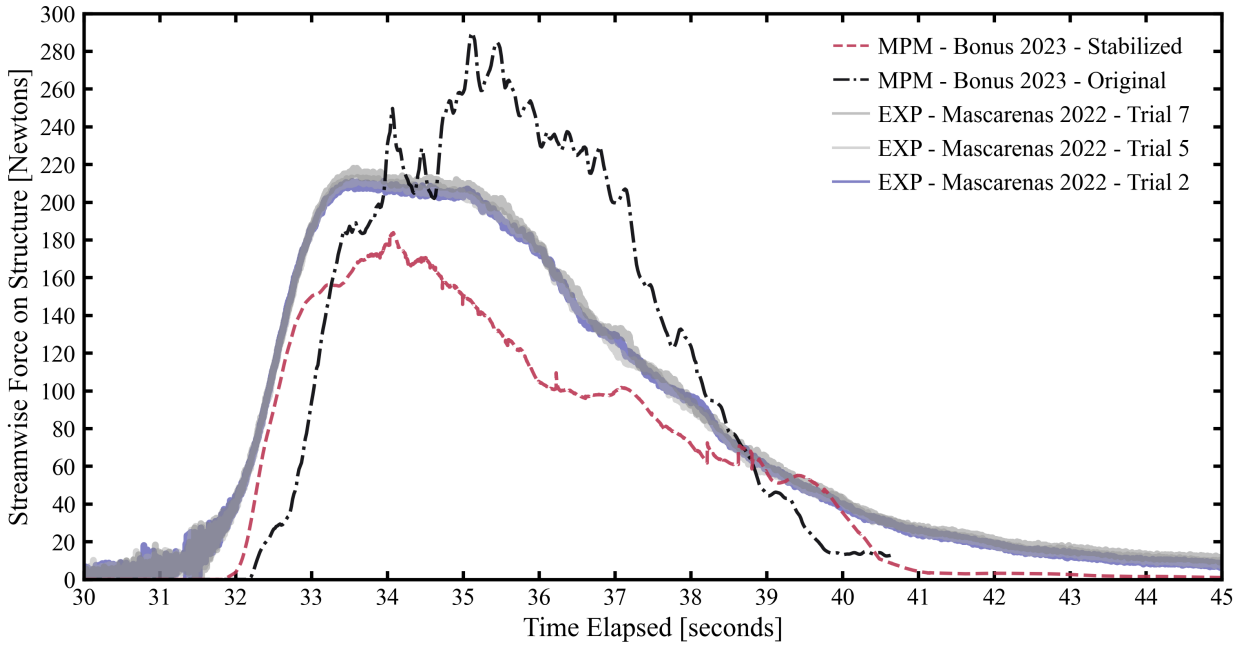

Fig. 7.11. Hydrodynamic streamwise load on OSU LWF structural box. Wave-only load on a raised structure in the OSU LWF. MPM's force profile is smoother when boundaries are stabilized. Experiments by Mascarenas 2022 shown. They reasonably match our MPM simulations (25% error, acceptable as this manuscript studies the dominant debris impacts). Simulations raised the box an extra three inches to account for shape-function width, resulting in a second slower wave-load arrival (adjusted for visual comparison in plot).

scale. The checkpoint is taken at a time before the box or debris are able to influence the wave (25 seconds after wave-maker motion starts). All results regarding model-scale ordered and random debris-fields use the original MPM hydrodynamic curve (i.e. non-stabilized curve in Figure 7.11).

7.5 *Ordered Debris-Field Simulations*

A set of sixteen ordered debris-fields were replicated numerically. Two orientations were considered, longitudinal and transverse, and debris were all arranged as arrays within the constraints of the 2.0 by 2.0 meter wooden frame used during experiments. Streamwise debris velocity visualization from all ordered debris-field simulations are shown, superimposed on each other and without water visualized, on the left-side Figure 7.12.

First peak forces from simulations for both orientations are plotted in Figure 7.13 along with all experimental data-points from Mascarenas 2022 and Shekhar et al. 2020. Simulations by Shekhar et al. 2020 are included, and box-and-whisker plots for Mascarenas 2022 are drawn. We see a remarkably strong agreement in our simulated results with trends in both experiments by Mascarenas 2022 and Shekhar et al. 2020 for first-peak debris-field impacts. Our MPM results always lay within the experimental trial envelopes Mascarenas 2022 and often are within the interquartile range and near to the experimental median.

Note that experiments by Shekhar et al. 2020 are not entirely identical to those by Mascarenas 2022 because the standing-water-level used in the former was slightly below the box, whereas it was flush with the box's base in the latter. Also, some transverse debris tests by Shekhar et al. 2020 did not have the downstream edge of the debris-field located 2 meters away from the structure's leading face, resulting in slightly different debris-field velocities and displacements streamwise and thereby structural impact loads. For this reason transverse debris tests in Shekhar et al. 2020 have larger values than Mascarenas 2022 and our own simulations, as the water is at a lower elevation and thus a smaller fluid buffer develops between the large surface area of

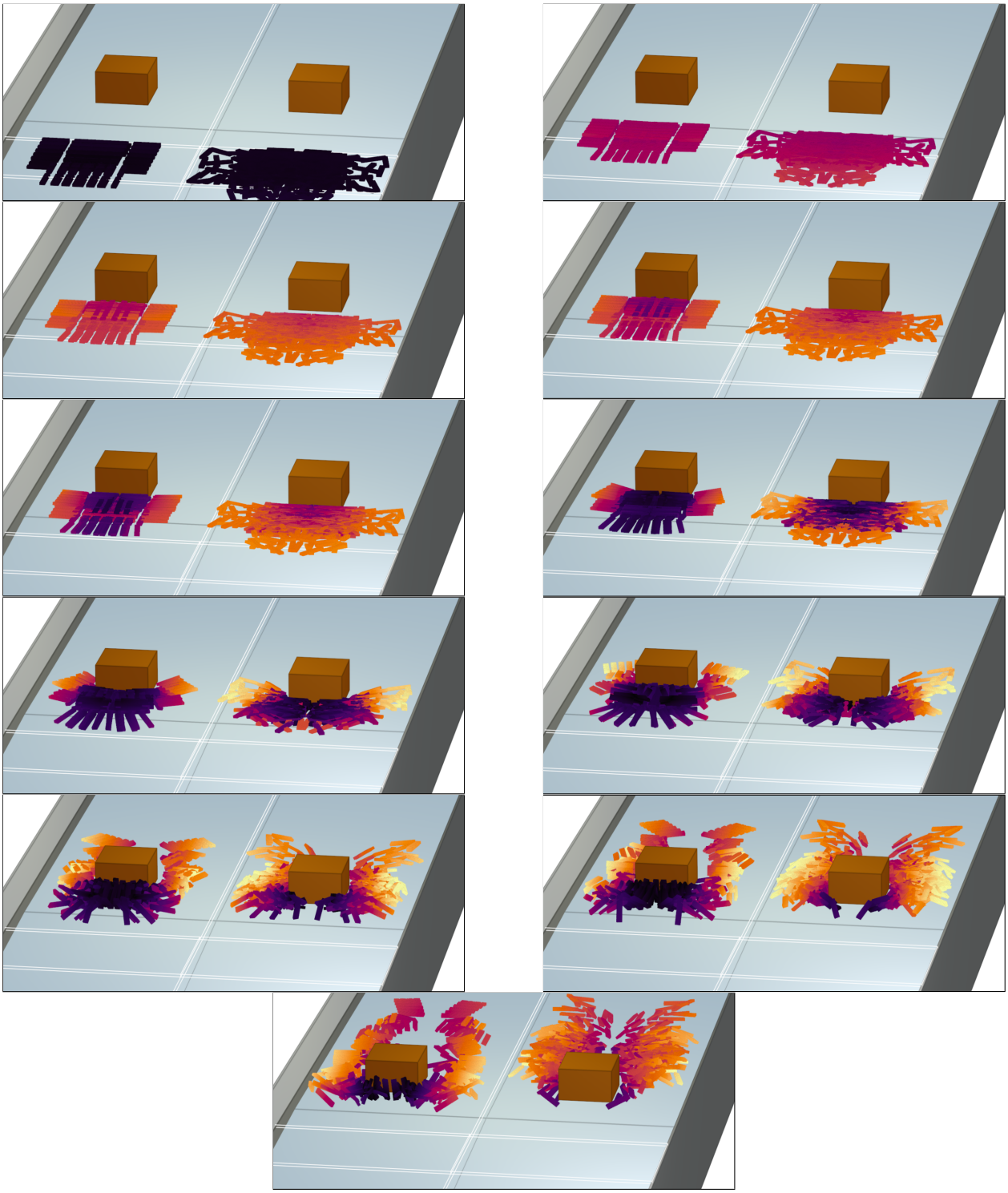

Fig. 7.12. Ordered and randomized debris field impacts in OSU LWF digital twin. ClaymoreUW Multi-GPU MPM simulates impacts of 16 ordered and 34 randomized debris fields on a raised structure (simulations superimposed to visualize motion envelopes), based on experiments by Mascarenas 2022. Debris streamwise velocity, from 0 to 1.5 m/s, map to black and bright yellow.

transverse debris and the structure. This does not appear to affect the smaller impact surface area of longitudinal debris, although their greater streamwise length changes the velocity imparted onto them by the slightly different wave used in Shekhar et al. 2020. Overall, we see good agreement in trends between our simulations and experiments by Shekhar et al. 2020, lending confidence to our model's ability to predict debris-field cases not performed experimentally in Mascarenas 2022 and thereby demonstrating capabilities in extrapolating and overall low-bias to fairly small details of the flume experiments.

Significantly, our Multi-GPU MPM simulation results improve massively over previous CPU MPM results by Shekhar et al. 2020. As their work was constrained by numerical resolution and hardware capabilities, coarse simulations were used and unsurprisingly loads were overestimated by 150 - 400%. Primarily by changing to a high-performance implementation, our lab has refined previous attempts into potential engineering conclusions for debris-field impact behavior.

First and max peak impacts can be normalized by the max longitudinal impact load for a single debris observed experimentally in Mascarenas 2022 to show how a multi-debris simulation compares to the worst case scenario single-debris experiment. Figure 7.14 shows said information with a noted linear and square-root trend in the longitudinal and transverse debris-field trials, respectively.

Figure 7.15 show the relationship of streamwise debris-field length and peak loads shows good linear fits for both the longitudinal and transverse debris-field trials. However, we note that this is probably due to the correlation between longer debris-fields and more debris (i.e. more momentum), there should be steps taken to control for correlated variables in order to understand the purely geometric influence of debris-field streamwise length on the impact loads.

Strangely, Figure 7.16 shows that transverse debris-field length has poor trends for linear regression against peak loads, especially for transverse debris-fields. This is in part due to, again, correlation with other variables and the fact that the structural face is only wide enough to support

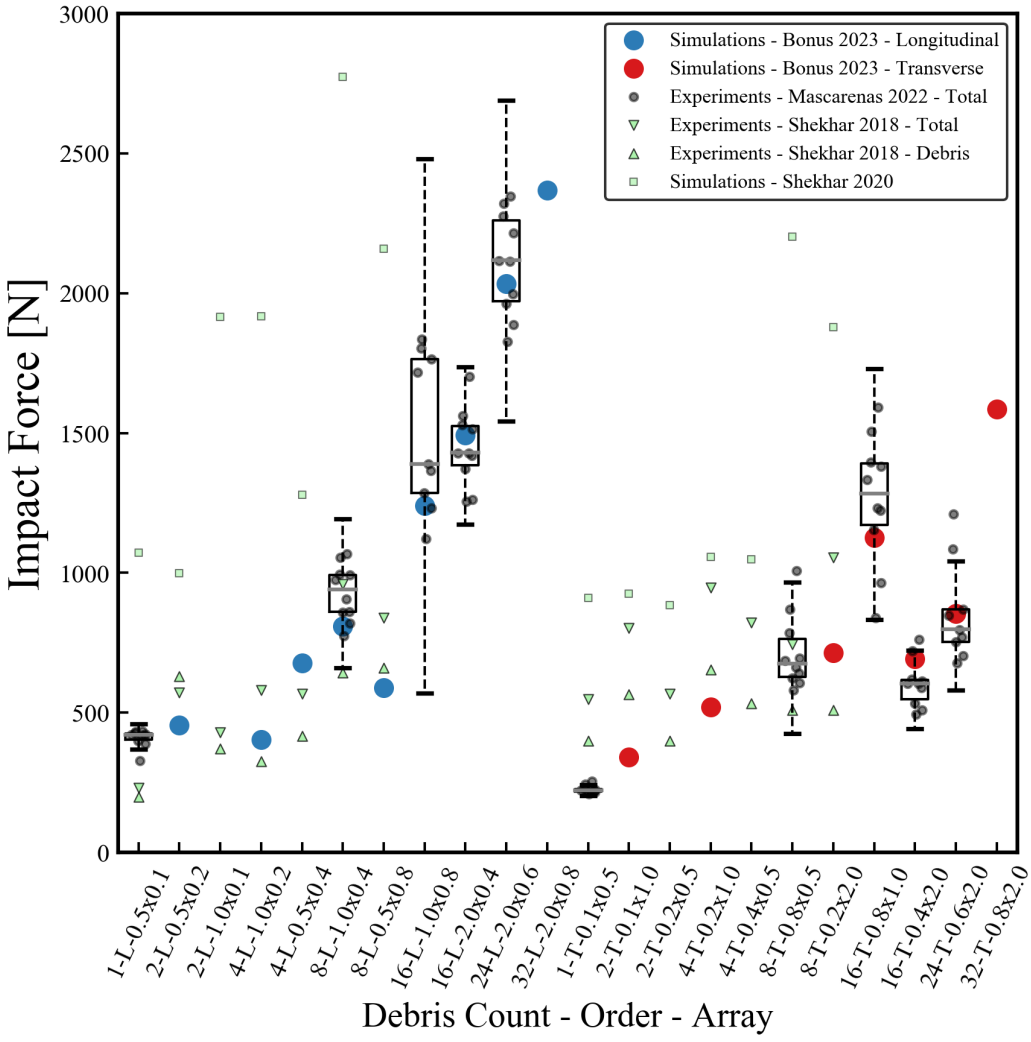

Fig. 7.13. First peak load of ordered debris field impacts in the OSU LWF as a box-and-whisker plot.

collision with two pieces of transverse debris at a time (ignoring 3D debris-field arrays) so some trials cannot fully contribute to a linear trend as they are cropped out of a full-impact by geometry relative to the structure.

Figure 7.17 plots the simulated load results versus the expected load value suggested by experimental medians from Mascarenas 2022 for both (a) first and (b) max peak loads.

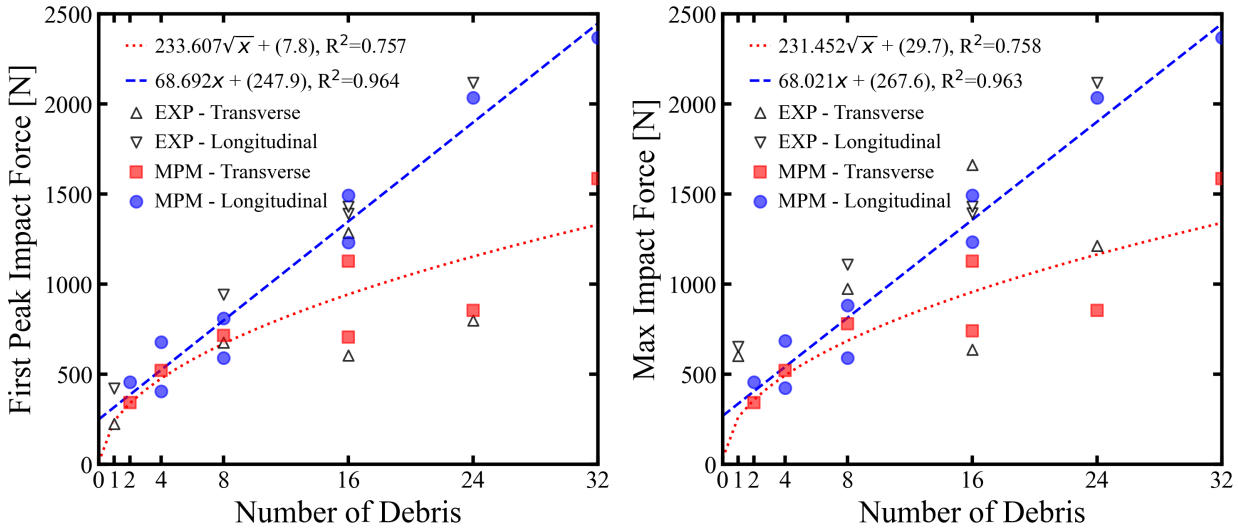

Fig. 7.14. First and max peak loads of ordered debris field impacts vs. number of debris in the field at the OSU LWF. Linear and square-root trends observed for first and max peak load predictions of longitudinal and transverse oriented debris-fields with respect to number of debris.

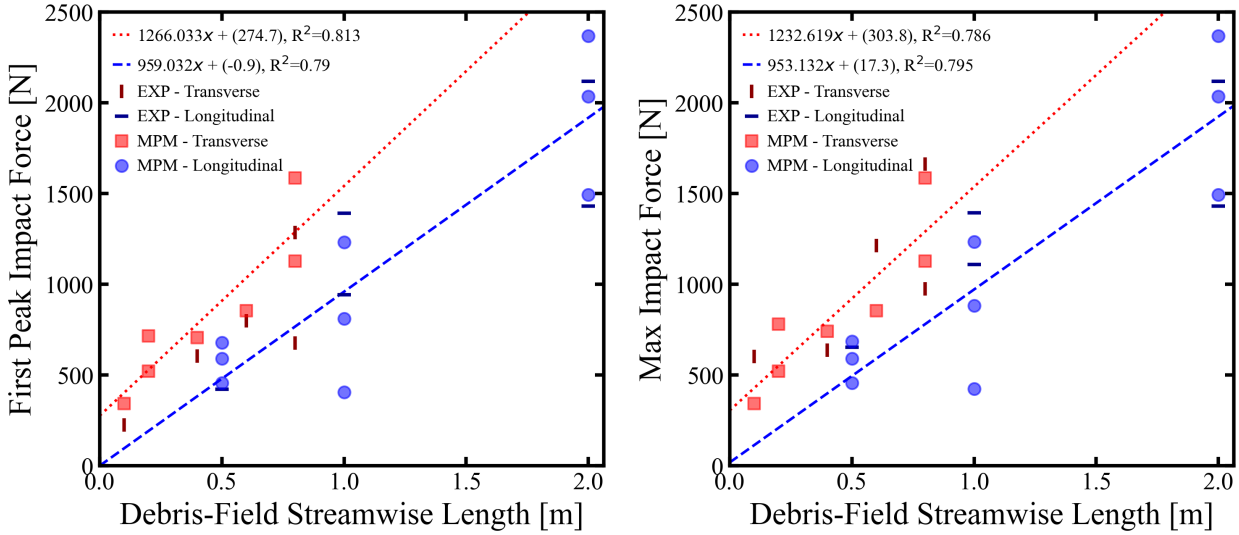

Fig. 7.15. First and max peak loads of ordered debris field impact vs. debris field streamwise length in the OSU LWF. Linear trends observed for first and max peak load predictions of longitudinal and transverse oriented debris-fields with respect to streamwise length.

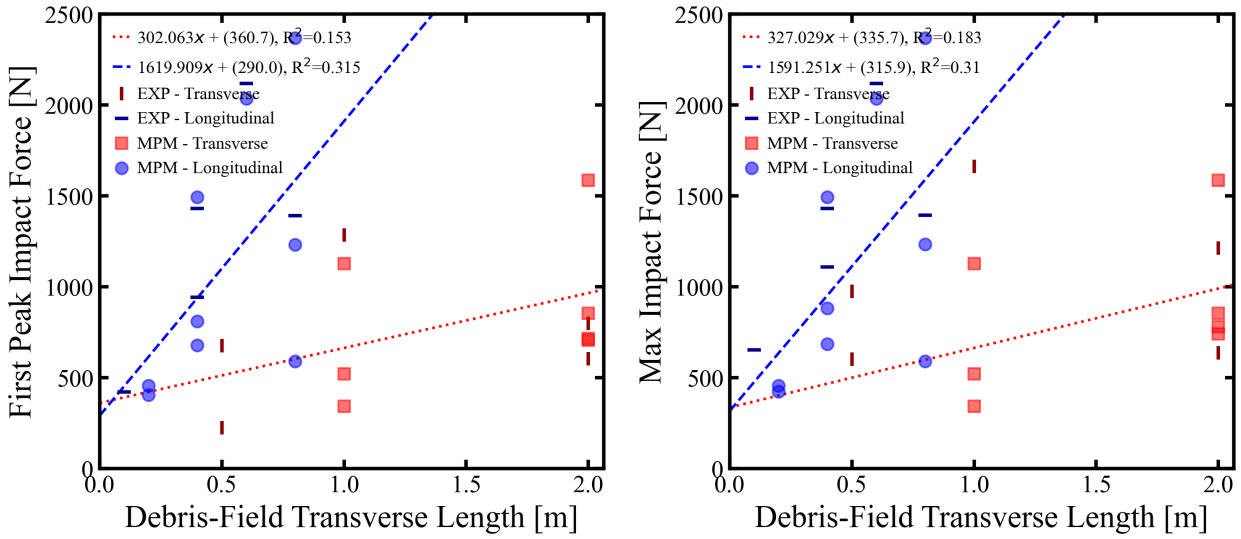

Fig. 7.16. First and max peak loads of ordered debris field impacts vs. debris field transverse length in the OSU LWF. Linear trends observed for first and max peak load predictions of longitudinal and transverse oriented debris-fields with respect to transverse length.

Respectively we find a 0.933 and 0.898 slope in fitted trend-lines, and R^2 values of 0.973 and 0.885. Intercepts are at 33.0 and -27.5 respectively which are nearly negligible but suggest slight bias. Our simulations are strongly predicting first-peaks median trends and magnitudes, and reasonably so for max peak trends and magnitudes. Both first and max peaks have a less steep trend than experiments, but are only off by 2.7 and 11.5% which falls within the envelopes of all load results per experimental trial and often within interquartile ranges.

7.6 *Random Debris-Field Simulations*

Random debris-field impact loads on structures exhibit highly chaotic behavior that has presented challenges to characterization for engineering design. In an effort to better understand this phenomena, 34 simulations were performed in ClaymoreUW MPM in a digital twin of Oregon State University's Large Wave Flume (OSU LWF). An additional 85 experiments from Mascarenas 2022 were brought into the random debris-field data-set for completeness. Cases

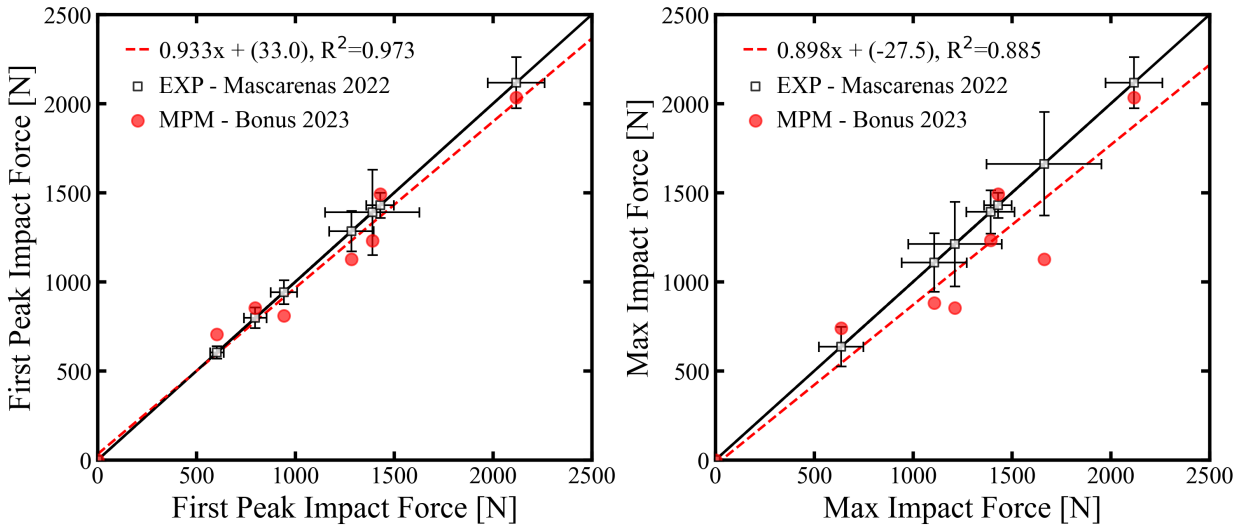

Fig. 7.17. Comparing experimental streamwise first and max loads per experiment case against measured streamwise Loads in MPM simulations on the raised structural box. Ideal simulated match between simulations and experiments by Mascarenas 2022 would give a trend-line with slope 1:1, intercept 0, and R^2 of 1. Results for both first and max peak loads are good, with first peak loads having a slightly better trend. Error bars represent the loading interquartile range of each experimental case.

center on moderate to large debris counts of 8, 16, and 24. They are in variously "dense/porous" and "angular" configurations that arise from random placement within a wooden frame of adjustable size.

We developed a simple scripts for generating random debris-fields and reporting their initial geometric meta-data (e.g. streamwise debris-field length, debris-field porosity, coefficient of uniformity). However, note that the geometric constraints of the wooden frame area used for debris placement restricts true randomness of subsequent debris placement, although this is not accounted for in a statistical manner in our analysis. Our numerical results aimed to augment the experiments which had suffered from a somewhat poor sampling of key variables due to time-constraints and costs of running hundreds of tests in experimental facilities. For example, we simulated less densely packed random debris-fields than the high density experiments to better characterize impact loads when debris-fields become increasingly sparse or when they geometrically have a high chance of missing the structure entirely during inundation.

All randomized debris-field simulations and experiments are represented as their max peak impact load on the structural box in the upper-left and lower-right of Figure 7.18. Respectively, the plots compare max debris-field impacts on the raised structure against number of debris c, the ratio of total debris-field occupied area to the allowed frame area for debris placement D, the initial stream-wise length of the debris-fields L_x, and the initial transverse length of the debris-fields L_y.

Inverse square root trends are observed for first peak loads in the upper-right and lower-left of Figure 7.18 respective to both transverse and streamwise debris-field length normalized by the structural box's width. Regression coefficients are nearly identical across the two variables, which is not expected as this behavior was not seen in ordered-debris fields which held moderate and strong linear trends for the streamwise and transverse length variables respectively. An explanation may be that the randomized placement of debris in a typically square frame statistically tends to have debris-field lengths in either direction that are, on average, nearly

identical if there is no experimental facility bias towards the random fabrics skew. This is apparent in the near identical data-points in both plots. A strong geometric correlation may mean the dominant trend observed in both length variables is a manifestation of a joint multiplicative variable, such as debris-field density or porosity relative to the frame size. This is supported by Figure 7.18's lower-right plot, which shows the debris-field density (0 to 1, higher means more of the frame area is filled with random debris) has a loosely square-root relationship with loads. Thus, For a given number of debris, increasing streamwise or transverse debris-field length will decrease the debris-field density so an inverted trend would be logical. This is potentially the source of the inverse square-root trend we see in our simulation results.

In summary, we observed recreation of all trends seen in the disordered / randomized debris-field experimental data in our Multi-GPU MPM simulations. Further, magnitudes of first peak impacts reasonably match medians for random debris-fields and tend to be within interquartile ranges. However, there were difficulties simulating very dense debris-fields, as debris-field to frame area ratios over 0.4 were difficult to construct in a randomized fashion without debris being placed too closely and thus sticking erroneously to each other. This could be fixed by simulating at a higher resolution or exploring contact algorithms to avoid sticky debris-field behavior. Overall, results we have produced using ClaymoreUW MPM for wave-driven randomized debris-field impacts are replicating trends observed in Mascarenas 2022 and reasonably matching experimental load magnitudes measured at the OSU LWF.

7.7 *Extrapolating to Tsunami Prototype Scales via Simulations*

Scaled-down experiments and simulations serve an important purpose in the study of complicated, large-scale phenomena like tsunami-debris impacts– especially regarding costs. However, they are limited by the physical realities of similitude laws. Not all force ratios (e.g. inertial to gravity, inertial to elastic, etc.) can be maintained across scales for these complex cases, and thereby the

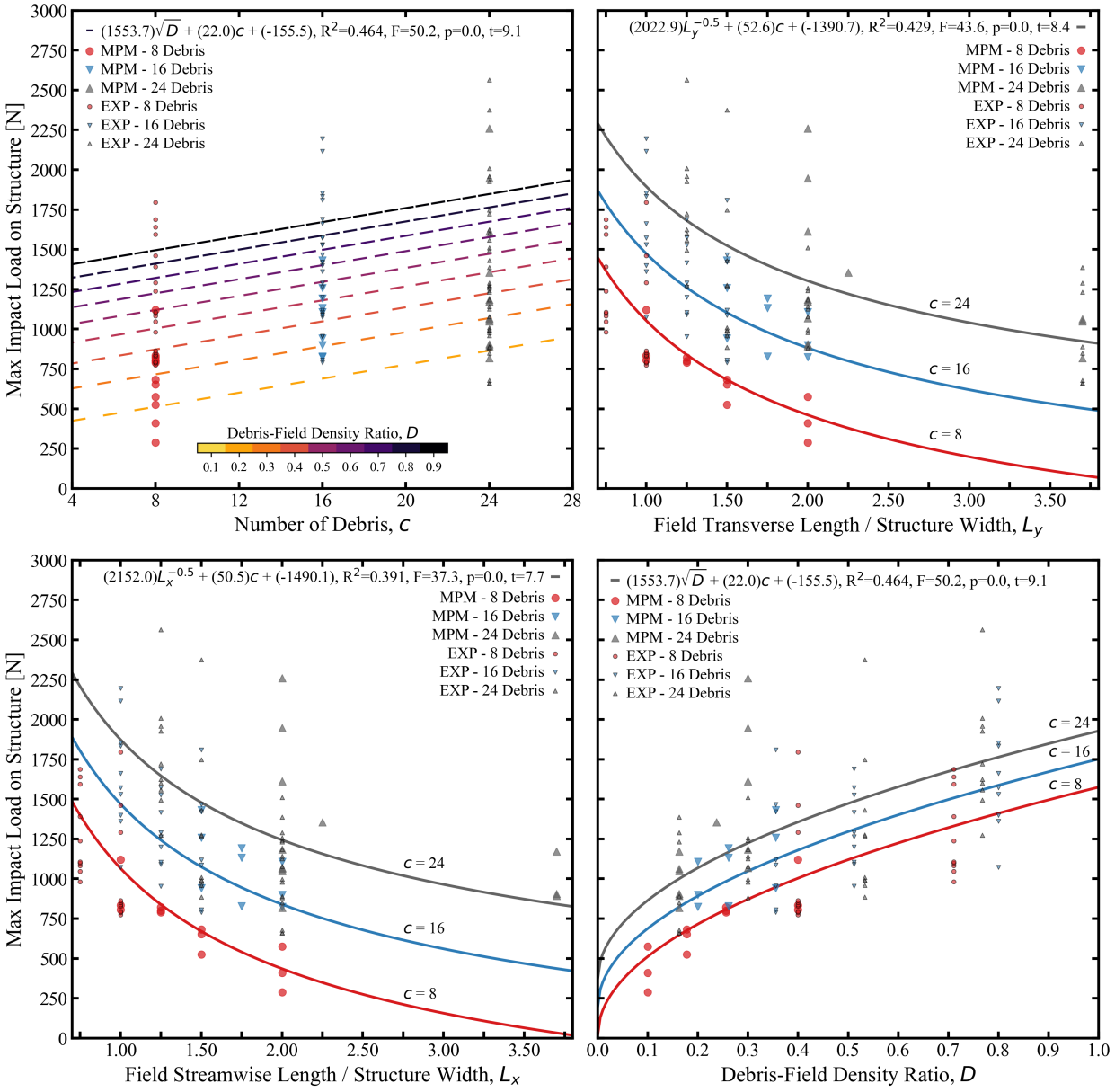

Fig. 7.18. Random debris-field max peak loads regressed against basic variables. Ordinary Least Squares regression is applied to random debris-field experiments by Mascarenas 2022 and MPM simulations from this work. Simple trends appear regarding debris-count, debris field area relative to an occupied frame area, and streamwise / transverse length normalized by the structure's face width. F-statistics, t-values, and p-values suggests the value of chosen variables. The Omnibus and R^2 imply that there is still something lacking in the model's representation. Kurtosis, skew, and Durbin-Watson are in ranges typical of thick tailed, somewhat asymmetric distributions.

phenomena we are attempting to study changes in fundamental and possibly debilitating manners. Balance of Froude scaling of tsunami-like waves has been extensively studied, as the balance of inertial to gravity forces is widely shown to be the dominant descriptor of gravity waves like the solitary form used at the OSU LWF. For smaller flumes and/or tests with dramatic wave breaking or air entrapment, Reynolds' and Weber similitude can be important though they are negligible here. If frequency effects are significant, as in wave attenuators, then Strouhal similitude is applicable but it is not apparent in our study as debris have already impacted the structure before notable eddies flow off the ramp joints and structural box. Material elasticity, speed-of-sound, and broader elasto-plastic behavior stand-out as similitude types worth investigating as we scale the OSU LWF tests to the prototype tsunami's scale.

Tandem similitude of all structure, debris, and fluid material properties (e.g. speed-of-sound, elasto-plasticity) is fairly unexplored due to difficulties of doing so in a lab environment, though it may be non-negligible for debris-field impacts as noted in Nistor et al. 2017a and Goseberg et al. 2016. There are fairly straightforwards ways to account for elastic aspects of debris scaling with Mach-Cauchy similitude approaches, used later in this chapter, when debris at the model and prototype scale are both elastic materials. However, it is insufficient for more complex materials, such as plasticly deforming automobiles and splintering tree-trunks. Debris in prototype tsunami events may exhibit highly nonlinear, large deformation, elasto-plastic, and possibly topology changing behavior which is not well observed in wave-flumes. To fully account for these affects it is possible that simple Froude similitude may be invalidated, or at the very least come with caveats, as it is not the only dominant similitude ratio for complex debris and structures.

Numerical simulations thus offer great promise in characterizing scaling laws and scale lengths for use in experiments so that we may better understand respective behavior at prototype scale. The Material Point Method, in particular, is an extremely strong candidate for this endeavor due to its support of advanced material models in the context of multi-material and multi-phase interaction.

To begin, we will look at a simple numerical Froude scaling of the OSU LWF model to prototype scales. Then, we will apply Froude-Cauchy scaling to show the affect of fluid compression during evolution of a solitary wave along moderately inclined bathymetry panels and when debris-fields displace a fluid-buffer immediately before structural collision. Finally, we will posit on the challenges of scaling tsunami-debris impact wave-flume tests to prototype scales and how our efforts can provide initial steps in solving them.

7.7.1 Froude Similitude in Flume Simulations

Froude dynamic similitude is achieved when the ratio of inertial forces to gravity forces, for some characteristic length L and velocity u of interest, are kept constant while scaling the prototype's dimensions (e.g. length, time, velocity, power) to a model case by appropriate Froude scaling laws. This ratio, the Froude number, is stated as

$$\left(\frac{F_{\text{Inertial}}}{F_{\text{Gravitational}}}\right)_{\text{Prototype}} = \left(\frac{F_{\text{Inertial}}}{F_{\text{Gravitational}}}\right)_{\text{Model}} = \frac{u}{\sqrt{gL}} \ . \tag{7.2}$$

Achieving Froude similitude in Equation 7.2 is often adequate to allow study of various fluid dynamics and coastal engineering problems which naturally occur at large scales (e.g. motion of boats in a wave, where dynamics are typically dominated by gravity and inertial forces) in a smaller and more cost effective wave-flume environment. This typically applies to solitary and solitary-like waves at the scale of the OSU flume when attempting to study the scale of a prototype tsunami event.

Figure 7.19 shows that when Froude similitude is performed at the wave-maker, i.e. scaling the wave-maker motion and subsequently generated solitary wave in the far-field prior to the bathymetry ramps, that the evolution of the solitary wave along inclined bathymetry panels (1:12 and 1:24 slopes) may not scale ideally according to Froude similitude when looking at wave-gauge

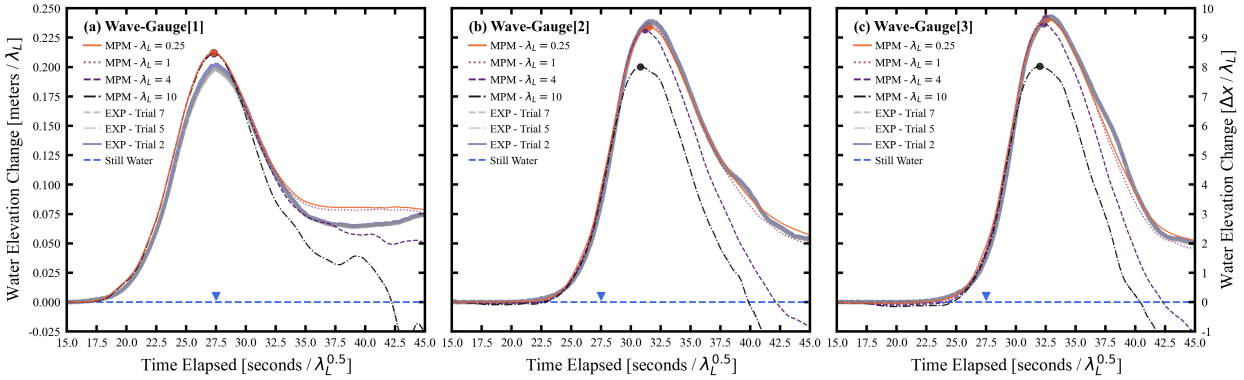

Fig. 7.19. Wave-gauge elevations of free-surface for various Froude length scales. Effect of Froude length scale on OSU LWF solitary wave evolution along a moderately inclined bathymetry (1:12 and 1:24 slopes) manifests as growing deviations. Froude similitude applied at the wave-maker. We observe that the breaking mode of the wave changes significantly enough to show notable differences in the free-surface in our compressible fluid MPM simulations.

free-surface elevations. The initial wave at the piston and the wave as it is seen in the lead-in and peak of WG1 (located before inclined bathymetry) do scale according to Froude similitude, however, we observe notable deviation in the free-surface by the second wave-gauge peak (located above the inclined bathymetry) which continues to grow when reaching WG3 and the structural box for the prototype tsunami length scale (10:1) relative to the OSU LWF model scale (1:1). This is seen as a drop in wave-elevation at WG2 and WG3. This is likely caused by a change in the wave's spilling-break's energy transition, which is potentially sensitive to density and pressure fluctuations of the weakly-compressible water model at prototype scales. This may or may not be due to compressibility effects of the fluid, as compressibility was kept constant (i.e. bulk modulus 2.2 GPa) for each tested Froude similitude prototype-to-model length ratio of 0.25, 1, 4, and 10.

Figure 7.20 plots load measurements on the structural box's face for varied Froude length scales in hydrodynamic conditions (no debris) with axes normalized to allow force-time comparison across length scales. We observe that scaling up to the prototype length scale of 10 deviates nonlinearly from our model scale of 1. Likewise length scale 4 deviates, but less significantly at first before

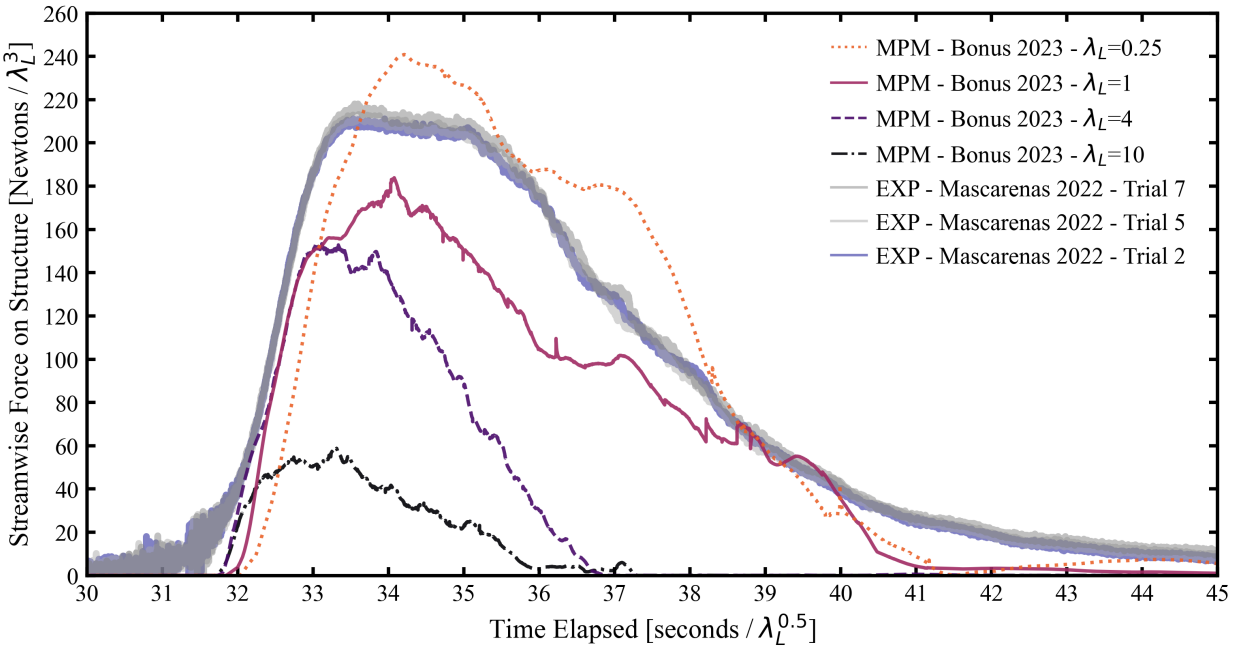

Fig. 7.20. Hydrodynamic streamwise load on structural box relative to Froude length scales in the OSU LWF. Load-cell measurements streamwise for varied Froude length scales of a solitary wave in the OSU LWF.

appearing to go through a type of mode transition when the wave reaches the top of the bathymetry at 30 seconds. Length scale 0.25 has very little deviation from the model scale, seemingly obeying Froude similitude expectations for the box load.

A key concept to note is that while water is nearly incompressible from the perspective of a 100 meter wave-flume with its bulk modulus of 2.2 GPa, this is not especially incompressible at the kilometer+ scale of prototype tsunamis. Considerable compression, reflections, and generally altered energy transitions of solitary waves into their asymmetric breaking form can occur when going up bathymetry of notable incline.

7.7.2 Froude-Cauchy Similitude in Flume Simulations

While Froude similitude is a standard expectation in coastal engineering, there are scenarios where other laws may play a significant role. Due to the slant of our experiments bathymetry panel's causing non-negligible wave reflection, and our debris-field impacts and interactions being highly elastic phenomena, we set out to investigate whether Mach-Cauchy similitude is of importance. The basic force ratio, the Cauchy number, is stated as:

$$\left(\frac{F_{\text{Inertial}}}{F_{\text{Elastic}}}\right)_{\text{Prototype}} = \left(\frac{F_{\text{Inertial}}}{F_{\text{Elastic}}}\right)_{\text{Model}} = \frac{u}{c} = \frac{u}{\sqrt{K/\rho}} \quad \text{where } c = \text{Pressure Wave Speed} . \tag{7.3}$$

The influence of fluid compressibility on free-surface readings for solitary waves undergoing a spilling-break over long, inclined bathymetries is shown by Figure 7.22 for our digital twin simulations of the OSU LWF as configured experimentally in Mascarenas 2022. The deviations in wave-gauge measurements between the OSU LWF results at length scale 1 (simulated and experimental OSU LWF model) relative to the Froude length scale 10 simulation (tsunami prototype) are notable when their is no Mach-Cauchy dynamic similitude (Eq. 7.3), i.e. prototype and model fluid bulk modulus are equivalent ($\lambda_C = 1$). Free-surface deviations are effectively negligible with Cauchy dynamic similitude ($\lambda_C = \sqrt{10} = 3.16$), meaning we scaled up the bulk modulus of the prototype by 10 to a value of 22.0 GPa (nearly that of mercury, which is 28.5 GPa). This is because the OSU LWF model (experiments and simulations) already used water as the primary fluid and thereby constrained the characteristic speed-of-sound for use in Equation 7.3. This reinforces our hypothesis that fluid compressibility has a non-negligible affect on solitary wave breaking expectations over moderate bathymetry slopes when Froude similitude is applied in the far-field.

Figure 7.21 shows horizontal velocity on fluid particles across the OSU LWF for Froude-Cauchy length ratio and bulk ratio pairs of 1-1, 10-10, and 10-1 as the waves begin to flow over the top of

the bathymetry ramps and as the waves nearly reach the end of the flume. Minor deviations are seen between the Cauchy similitude considerate and inconsiderate cases by the time the wave crests the ramp, showing that fluid compression occurs, and the deviations have grown substantially by the time the wave reaches the end of the flume and can be viewed in the rip current of the 10-1 scaled case versus the lack of such current in 10-10 and 1-1 cases. Figure 7.23 shows significant deviation of wave-gauge data against Froude force and time scales for a purely Froude scaled case versus the base simulation model of the OSU LWF, whereas adjusting the Froude scale along with the Cauchy-Mach speed-of-sound ratio produces the predicted Froude similitude scaled forces (i.e. a cube law relative to length scale). This suggests that elastic properties of the fluid medium can play a notable role in the effort of scaling, at the very least, spilling solitary-like waves traveling up long, moderate slope bathymetries in regards to hydrodynamic loads on raised structures.

The key insights are that **(i)** the inclined ramps and raised structure create reflection waves which are compressible phenomena, **(ii)** the solitary-like wave undergoes a spilling break mechanism which features an energy transition sensitive to compressive fluctuations in the fluid medium along the bathymetry ramps, and **(iii)** that despite wave-flume experimental models scaling down lengths and velocities according to Froude dynamic similitude they do not adjust the model fluid (water) from the fluid of a prototype tsunami (water). This is an underlying, important implication for any system seeking dynamic similitude where compressive behavior is non-negligible, as the prototype to be scaled up to would have to feature a stiffer, mercury-like fluid in order to meet Froude similitude force scaling predictions on a structure

Hydrodynamic forces for a characteristic unbroken wave experiment, four Froude length scale simulations, and one Froude-Cauchy length scaled simulation are plotted in Figure 7.23. We see that while a Froude scale of 10 deviates strongly from Froude Scale 1 simulations and experiments, Froude-Cauchy scale 10-10 very closely matches Froude Scale 1 simulations with

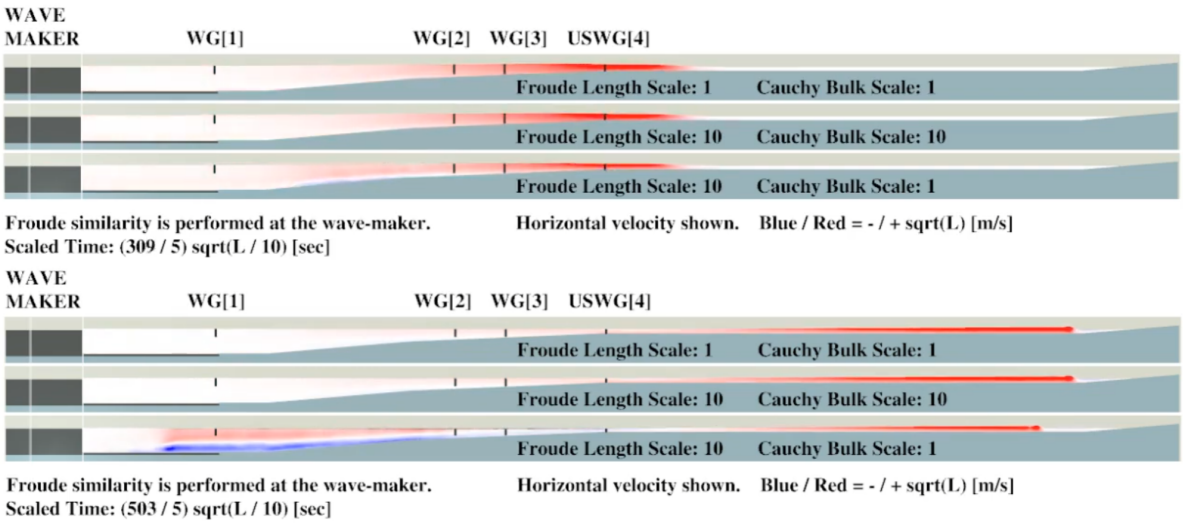

Fig. 7.21. Hydrodynamic velocity in OSU LWF for varied Froude-Cauchy scales. Streamwise fluid velocity visualized for the OSU LWF model wave and prototype tsunamis (with and without Mach-Cauchy dynamic similitude) when applying Froude dynamic similitude at the wave-maker (far-left).

reasonable deviation from experiments that has been discussed previously. This demonstrates the importance of fluid compressibility for even the basic hydrodynamic behavior of our experimental and simulated model of the prototype tsunami.

We see the trends of our hydrodynamic similitude studies continue into a debris-field case for 24 longitudinal debris. Figure 7.24 plots Froude length ratios of 0.25, 1, 4, and 10 (where 1 currently refers to the OSU LWF scale and 10 is the prototype tsunami) to show that the up-scaled simulation loads again deviates nonlinearly. A set of experimental loads are included as a sanity check for our observations. While none of the MPM loads are identical to experimental experiments, Froude length scales of 0.25, 1, and 4 show general load trends which signify simulation of the same phenomena as the OSU LWF experiments, even though their are some scale dependent affects not fully accounted for. This is evident in their similar normalized peak loads and sustained forcing after initial impact of the debris-fields. However, the Froude length

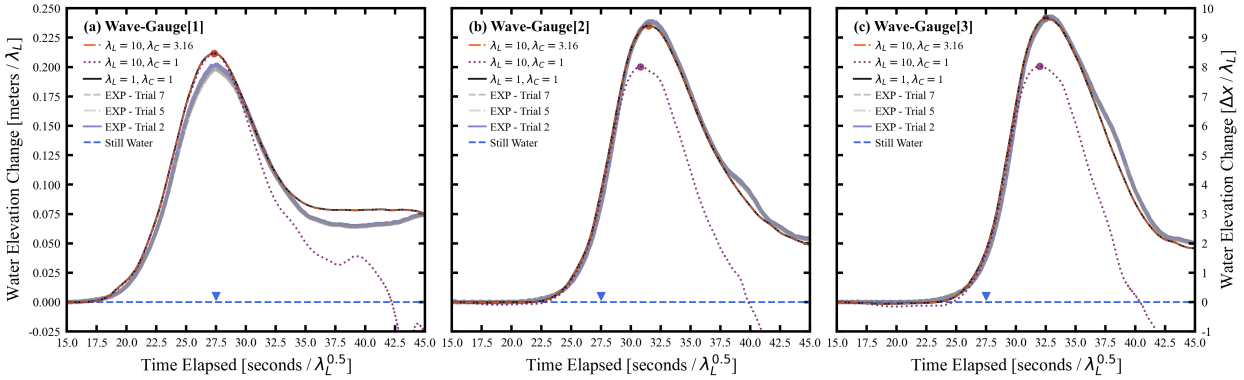

Fig. 7.22. Wave-gauge elevations of free-surface for various Froude-Cauchy scales. MLS-MPM simulations in ClaymoreUW of the OSU LWF. Plots show elevation deviations relative to the OSU LWF model ($\lambda_L = 1$, $\lambda_C = 1$) for measured wave-gauge measurement normalized by anticipated scaling using Froude dynamic and geometric similitude at the prototype ($\lambda_L = 10$). This can be remedied if Cauchy-Mach dynamic similitude is also achieved in tandem ($\lambda_C = \sqrt{10} = 3.16$), entailing a 10x increase of the bulk modulus of the fluid at the prototype tsunami scale.

scale 10 simulation results far exceeds experiments and trends of the other simulated scales in its normalized first peak impact loads, damming loads prior to the second peak impact loads, and secondary collision loads, along with its sudden drop-off to zero. The drop-off may be a sign of faster inundation draw-down (as seen in Figure 7.21) and/or of the hydrodynamic loading time-scale not scaling purely with Froude time scaling.

It is not clear why Froude-Cauchy scaling results produces debris-field loads relative to hydrodynamic loads so much larger than predicted by the Froude similitude λ_L^3/λ_L^3 relationship, here by 270% for 24 longitudinal debris. Froude similitude has been successfully applied throughout coastal engineering for many decades, with inclusion of Mach-Cauchy similitude typically reserved for Mach numbers > 0.1. Our model and prototype wave celerity is around 5.5 and 17.5 m/s, producing Mach numbers of 0.004 and 0.012 respectively so it is unusual that Cauchy-Mach similitude plays a notable role.

However, we emphasize that the common assumption regarding compressibility in coastal

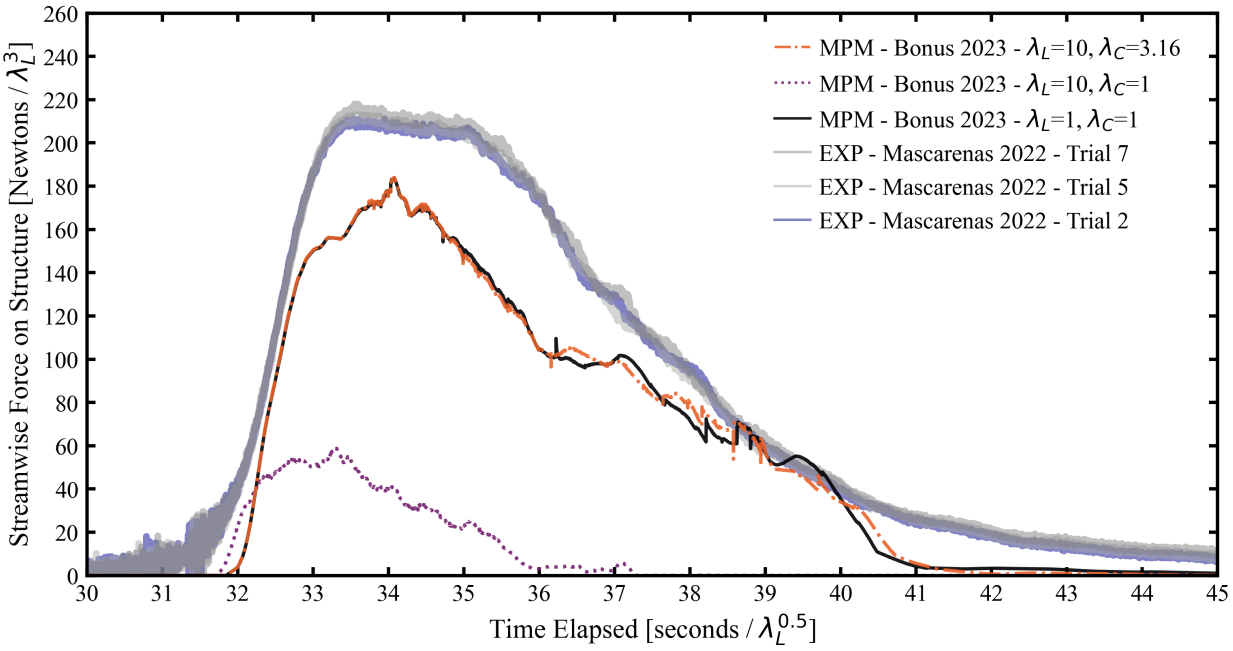

Fig. 7.23. Hydrodynamic streamwise load on a structural box for various Froude-Cauchy scales. Loads from varied Froude-Cauchy scales and experiments of the OSU LWF normalized by Froude similitude scaling laws. Simulated forces have near perfect agreement between the OSU LWF model (1-1 length and speed-of-sound ratios) base simulation and the scaled Froude-Cauchy prototype simulation (10-3.16 length and speed-of-sound ratios), whereas the prototype with only Froude similitude (10-1) does not. This suggests that elastic properties of fluid media are non-negligible for achieving the Froude scaling prediction of hydrodynamic structural loads for this specific scenario in the OSU LWF.

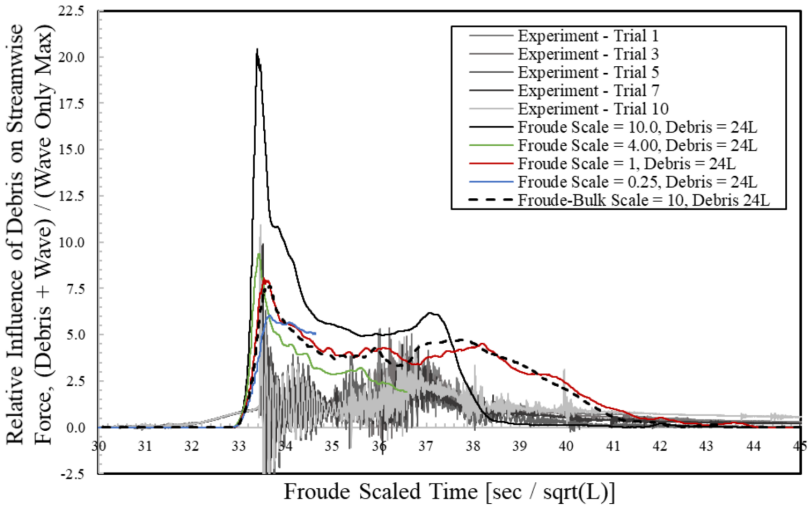

Fig. 7.24. Hydrodynamic streamwise load on structural box for 24 longitudinal debris. Measurements for varied Froude-Cauchy length scales simulations and the experimental results.

engineering, i.e. Mach numbers > 0.1 required for Cauchy-Mach similitude significance, are only meant to be applied to quasi-steady and steady-state flow. The former is, loosely, interpreted as flows with effectively constant, or only small transient changes in, flow-properties over appropriate time-scales relative to the length scale at points for a given reference frame.

For bathymetries of only slight influence on flow, i.e. milder slopes than the 1:12 used in the 100 meter OSU LWF experiments, quasi-steady states are often seen for aspects of solitary waves (potentially multiple waves interacting). However, as our slopes are non-negligible they present increasing risk of invalidating the quasi-steady flow assumption for our spilling-break solitary-like wave as we scale up to the prototype tsunami. This is because the slope ratio is maintained from model to prototype (ratio of 1, i.e. reserves 1:12 grade, which is 4.76°), yet the length scale magnifies by 10, velocity scale by 3.16, time scale by 3.16, and speed-of-sound scale is by just 1. In other words, the constricted time-scale of the tsunami-like wave event in both its flow-speed and fluid media's speed-of-sound (relative to length-scaling of effective wavelength , as well as the bathymetry length the wave must travel up and spill over) leads to non-negligible temporal

variation in flow. Yet, the bathymetry slope is unscaled so relative wave velocity increase and wave compressibility, producing stronger reflections off the bathymetry over a much larger length (and with much more relative compression) and thus more of the horizontal wave energy redirects vertically to destabilize the waves breaking-spill and thereby its flow-speed, depth, and overall loading on the structural box. Their is also some diffraction at the 1:12-meets-1:24 panel slope joint which may play a role in these scaling phenomena.

We could also describe this occurrence in Iribarren parameter terms or the more broad family of surf similarity methods, noting that the fictitious wave steepness calculated in the naive form (not adjusting far-field measurements for the near-shore scaling affects, assuming direct applicability instead) does scales proportional to bathymetry slope, which suggests similitude. However, the approach to calculating surf similarity parameters aside, the wave celerity outpaces both fictitious wave steepness and bathymetry slope by 3.16x and the relative fluid compression scales by 10x without Cauchy similitude.

In sum, the wave interacts with an effectively steeper bathmetry when adjusting for its velocity at the prototype scale, bringing a 1:12 (4.76°) slope to something notably more impeding from the perspective of compressible fluid motion. This may be enough to change the wave's breaking-spill behavior, or even alter it into a break more similar to a collapse or surge break characteristic of very steep bathmytries. This may explain the apparently erosion-inducing back-flow current in Figure 7.21 when Cauchy similitude is not enforced for the prototype relative to the OSU LWF model, yet disappearing when it is attained.

We simplify our scope, now focusing on Figure 7.25. Two curves are plotted for Froude length scales of 1 (model) and 10 (prototype) with a shared speed-of-sound ratio of 1, which does not achieve Cauchy similitude, as Froude velocity scale is 3.16. One curve is plotted for a simulation at a Froude length scale of 10, Froude velocity scale of 3.16 (i.e. $\lambda_L^{0.5}$), and with an additional Mach-Cauchy adjustment made to the fluid's bulk modulus (bulk modulus ratio $\lambda_B = 10$) which

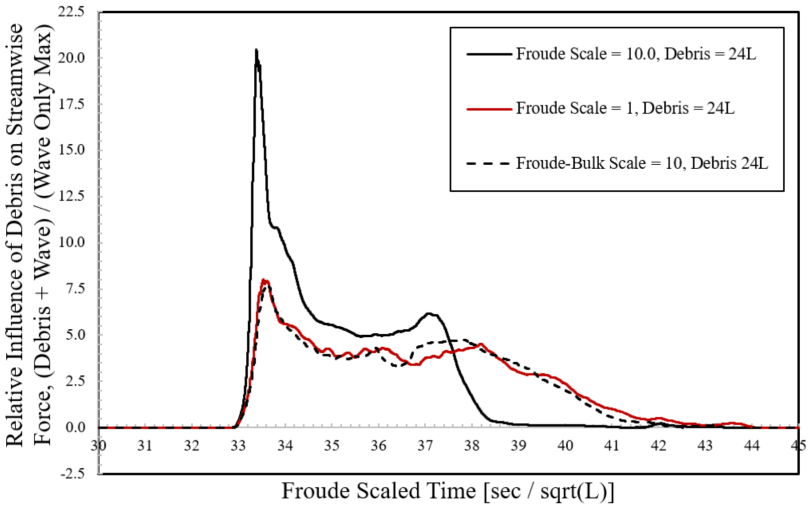

Fig. 7.25. Hydrodynamic streamwise load on structural box for 24 longitudinal debris. Measurements for varied Froude-Cauchy length scales simulations.

produces a speed-of-sound ratio of 3.16 (i.e. $\lambda_C = \lambda_B^{0.5} = \lambda_L^{0.5} = 3.16$). This achieves dynamic Mach-Cauchy similitude (Eq. 7.3) and Froude dynamic similitude (Eq. 7.2). However, the latter prototype case had to use a bulk modulus of 22.0 GPa, ten times higher than waters (2.20 GPa). The plot in 7.25 shows that the Froude-Cauchy similitude (length ratio and bulk ratio, 10-10) loading result lays nearly perfectly on top of the original OSU simulation model scale of (1-1), i.e. it attains force scaling proportional to λ_L^3, as predicted in Froude similitude, but the pure Froude similitude simulation does not. Some deviation follows after impact due to the improperly scaled speed-of-sound / compressibility of the debris material, the relative effect of the fluid viscosity, and floating point error. However, this deviation and/or error is minimal. This shows that the anomaly of the Froude similitude only scale simulation is from its fluid's material properties not maintaining Cauchy-Mach dynamic similitude as Froude similitude is enforced, meaning the fluid is effectively more compressible and thereby behaves different traveling up the bathymetry and when displaced by the debris-field.

Does this imply that we should use the Froude-Cauchy dynamic similitude prototype when

extrapolating from wave-flume experiments, as it produces results inline with Froude scaling law expectations? No, because doing so means we are no longer studying water in the prototype but a fluid with incompressibility on par with pure mercury. What is important is that we understand that a straightforward Froude scaling of a simple wave from a large model (OSU LWF) to prototype events (tsunamis) can produce counter-intuitive anomalies in wave-gauge data, wave loading, and importantly the debris-field loading relative to wave-loading. The last point implies debris-fields can be significantly more hazardous than predicted with Froude scaling (270% in this longitudinal case). The former two highlight the potential for water's compressibility to alter hydrodynamics on even moderate bathymetry's when studying the prototype.

A relative increase in debris-field to wave-only loading of 270% is substantial, but only supported by one data-point so far. To test its robustness, we simulate a longitudinal debris-field case with just 16 debris to observe sensitivity to debris count. Figure 7.27 shows that, again, a 270% increase in debris to wave-only loading occurs when Froude scaling to the prototype. Next, we try two transverse cases, containing 24 and 16 debris, to check if the load magnification applies to debris-fields with a different orientation. Figures 7.26 and 7.28 show full support for the trend. These transverse debris-fields both have 210% amplification, a decrease from the longitudinal cases 270%. This is still an extreme increase in the anticipated hazard of prototype debris-field impacts. They far exceed Froude predictions that are easy to rely on when working in large model wave-flumes, e.g. the OSU LWF, and with incompressible fluid simulations, i.e. impervious to discussed material requirements for similitude.

With the given information, 2.1 - 2.7x is our estimate of amplification at prototype scale. Admittedly, our four debris-field data-points do not prove that this will apply in other situations. Our experiments and simulations concern a solitary wave in intermediate depths traveling up two moderately inclined ramps (1:12 and 1:24) during which its front grows steep and causes the wave to begin breaking / spilling at the bathymetry crest. The spilled wave front then mobilizes the

debris-fields into a raised rigid structure, at which point the debris must displace buffers of water that slow their approach and subsequently their maximum impact loads.

This scenario is sensitive to fluid compression along the bathymetry ramps because the solitary wave spilling relies on downstream energy transfers to achieve Froude similitude at the intended heights. Energy reflects off the ramp and back towards the wave-maker for more compressible fluids (relative to length scale), so it is logical that forward spilling is reduced in the prototype while the reflected wave and a reversed current on the bathymetry panels become stronger. Reduced wave depth at impact may allow the prototype debris-field to collide with the structure with a less significant water buffer volume in its way that is lagging behind the increased debris momentum, so less debris energy is spent displacing the now easy to compress and displace water at prototype scale and is instead seen in the load-cell measurements as greater debris to wave loading ratios. In reverse this trend does not hold, as unchanging water grows stiff compared to Froude down-scaled debris momentum, reducing impact loads for debris, and the energy reflections on the ramps that altered the prototype wave via fluid compression are not able to dominate solitary wave's forward spilling. Experiments performed in these small wave-flumes, say 10 meters or less in length stand to benefit the most from understanding elastic and material similitude laws for debris-field impacts as they are currently dominated by water's effective incompressibility relative to the momentum accrued in model debris and thus struggle to resolve the true structural loading hazard of prototype events.

However, an anomaly still remains. It appears that despite not scaling the debris' Young's modulus, which remain at 0.8 GPa throughout all simulations, we still observe increases in impact loads that are typically associated with increasing stiffness of impact when keeping a constant fluid bulk modulus while up-scaling geometric length scale. This may be explained by MPM replicating the "effective stiffness" of the collision across the grid-cell at the structural box's face as a rheological parallel or serial circuit. This circuit may combine stiffness from both the debris

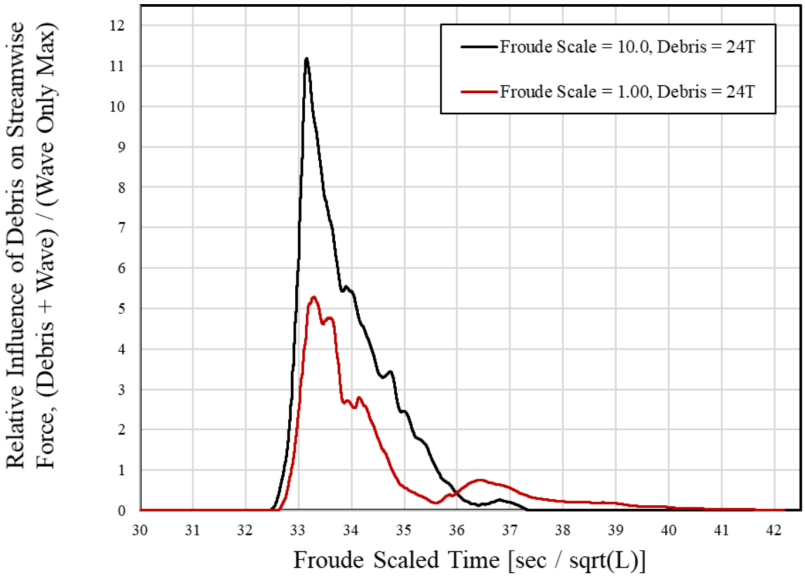

Fig. 7.26. Hydrodynamic streamwise load on structural box for 24 transverse debris. Measurements for varied Froude-Cauchy length scales simulations.

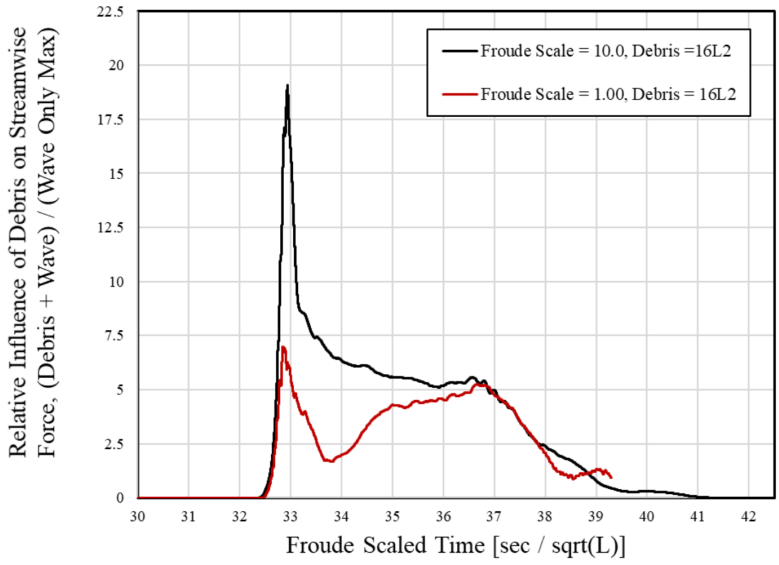

Fig. 7.27. Hydrodynamic streamwise load on structural box for 16 longitudinal debris. Measurements for varied Froude-Cauchy length scales simulations.

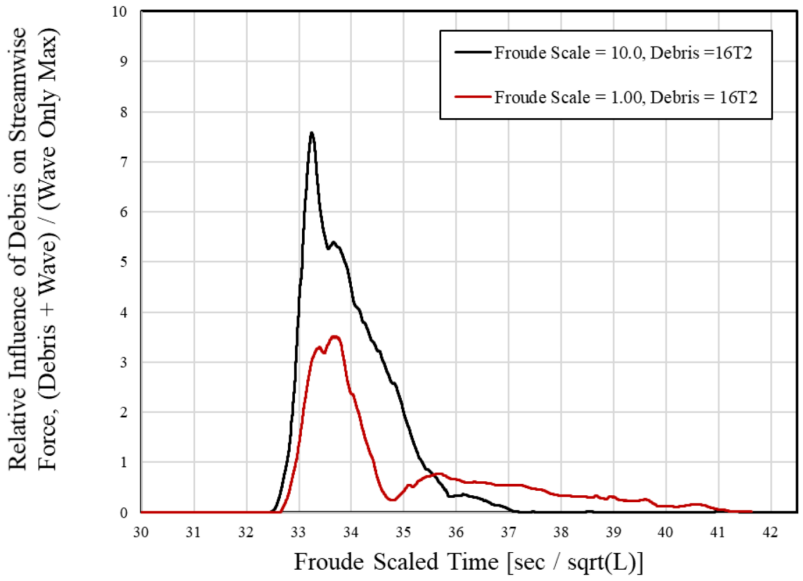

Fig. 7.28. Hydrodynamic streamwise load on structural box for 16 transverse debris. Measurements for varied Froude-Cauchy length scales simulations.

and the water at impact, of which the latter is dominant. This would explain why Figure 7.25 shows the Froude-Cauchy similitude prototype case is a near perfect up-scaling of the OSU LWF model's structural loads (relative to Froude scaling law predictions) despite leaving the debris speed-of-sound / stiffness unchanged.

It may be that the discrepancy is from "added mass", or perhaps "effective impact stiffness". The former may be influenced by the ability of debris with given momentum for the model and prototype scales to displace fluid volumes in their path, which may be easier to do if the fluid is more compressible relative to debris momentum (i.e. bulk modulus and speed-of-sound do not increase with length and wave velocity scaling).

The latter "effective impact stiffness" seeks to describe the stiffness of the debris-fluid buffer system at the box face. Assuming that impact stiffness is dominated by the 10x scaled fluid stiffness over the constant debris stiffness and then considering its square root relationship to max impact force in the standard bar impact equation, it is logical that it multiplies $L^{0.5}$ into the former

$L^{2.5}$ to form a L^3, a convenient explanation but not proof. More experimental and case-history research is necessary to draw actionable conclusions regarding the relevance of effective impact stiffness to prototype event loads, and further, how to accurately approximate the effective stiffness without back-calculating from structural loads.

While our argument of effective stiffness allowing the Froude-Cauchy up-scaled fluid to dominate the Froude up-scaled debris aligns with certain lines of logical rhetoric, we maintain healthy skepticism as the various impact models in the literature could easily support a number of incorrect hypothesis. Our observations could be an artifact of our numerical method until corroborated by other approaches. MPM naturally leads to thin fluid buffer systems at debris-structure impact interfaces because the presence of debris and fluid particles constrained by structural boundary conditions in a grid-cell (1" wide) or B-Spline stencil (2" wide) creates a mixed 8-point brick element in the finite element view. Said elements could be nearly-incompressible due to a rheological hybrid stiffness of all constituent material points. In other words, the up-scaled fluid bulk modulus over-rides cell deformation contributed by compressible debris at impact with a disproportionate pressure response to the cell's volume change. The culprit may also just be volumetric locking in the low-order MLS-MPM grid background grid not alleviated by our 0.999 linear mixing of F-Bar antilocking. Finally, floating point error could create numerical stiffness, but we have been diligent in its minimization as shown in our optimized GPU fluid implementation (Section 5.4).

Some guidance is available in the literature. Debris impact in-water phenomena were studied semi-analytically in Paczkowski et al. 2012, where they sought to describe the validity of "added mass" and "impact stiffness" in impact equations and alternative explanations of the phenomena. For instance, reflection and transmission coefficients R and T for the debris-fluid impact wave

transfer medium at a rigid structure may be defined as:

$$T = 1 - R, \quad R = \frac{\bar{\rho}\bar{c}\bar{A} - 1}{\bar{\rho}\bar{c}\bar{A} + 1} \quad \text{where } 1 \leq R \leq 1 \ . \tag{7.4}$$

Likewise the effect on impact duration, where $R \geq -0.5$ causes systems where impacting debris do not bounce off of the structure following the first impact cycle but remain until 3D fluid flow conditions cause a disturbance and thus decoupling from the impact face. In simulations at prototype scale we observe this lengthened contact time between the prototype debris and structure, an inertial calm following impacts which we see little off at smaller scales. This delayed decoupling affects impact force according to:

$$\bar{F} = 1 + 2R + 2R^2 + 2R^3 + 2R^4 + 2R^5 + 2R^6 + 2R^7 + 2R^8 + \dots \ , \tag{7.5}$$

which is a series that we choose to truncate at the eighth term. For a R value of 0.5 - 0.8 (the upper-end of 0.8 is stated for shipping containers, our prototype debris) this can produce a 2.5 - 3.5x amplification of impact force. Our MPM scaling produces R values in this range when multiplying the fluid bulk moduli B by 10 even if the debris Young's moduli E is left unchanged at scale 1. This could offer an analytical explanation of the apparent L^3 scaling of debris-field impacts when only $L^{2.5}$ is accounted for by the similitude law alone. Hence, the Froude-Cauchy 1-1 and 10-10 (stiffened fluid but not debris) cases are equivalent when normalized by the Froude force relationship of L^3.

Impacts are complicated, especially when considering multiple bodies, materials, and phases as they scale along a myriad of similitude. Our observation that relative debris impacts are amplified 2.1 - 2.7x in certain conditions, here for the prototype debris-field event, was unexpected but not unreasonable. Asking Froude similitude to describe all aspects of our compressible system is an unfair expectation. It is easy to over rely on incompressible simulations as they perform well in

many cases. For instance, in small scale flumes where water is effectively incompressible, and in large scale ocean events where extensive research has allowed for elegant mathematical models that do not consider all aspects of compressible physics. However, moving between scales raises significant risks of ignoring potentially dominant material scaling behavior if a compressible continuum with accurate properties is not used (prototype debris may even require full elasto-plasticity for constitutive similitude). This was observed in our compressible scaling of the OSU LWF debris-field experiments to a prototype tsunami event. Namely, debris field loads relative to wave-loads outpace Froude scaling expectations by 2.1 - 2.7x as we extrapolated outside the comfort of a controlled wave-flume.

7.7.3 Towards Exa-Scale Tsunami Debris Simulations

Higher numerical resolutions at the impact interface are the best chance at clarity regarding unexpected scaling behavior. Future works should focus on adaptive grid refinement and/or exa-scale simulations to the tune of a billion+ particles within sub-inch grid-cells. The latter we have applied to our OSU LWF digital-twin as an initial step into exa-scale coastal engineering. Using ClaymoreUW Multi-GPU MPM, we simulated one billion stiff fluid particles (1,000,000,000) as visible in Figure 7.29. Particles undergo one million time-steps (1,000,000) to model 3.25 seconds in the digital twin (time-steps of Δt = 3.25e-6 sec). The background MPM grid-cells contain 27 particles each and are comparable in width to US dime coins (Δx = 1.75 cm). One quadrillion (1,000,000,000,000,000) individual executions of the MPM algorithm occurred. As double-precision operations per MPM step is approximately one thousand (dependent on the compiler, etc.), summing all computations totals at one quintillion (10^{18} = 1,000,000,000,000,000,000) double-precision floating-point calculations. Comparatively, all time transgressed, from the big-bang to now, is estimated at 465 quadrillion seconds ($4.65*10^{17}$ = 465,000,000,000,000,000). The simulation took 24 hours on a single HPC node.

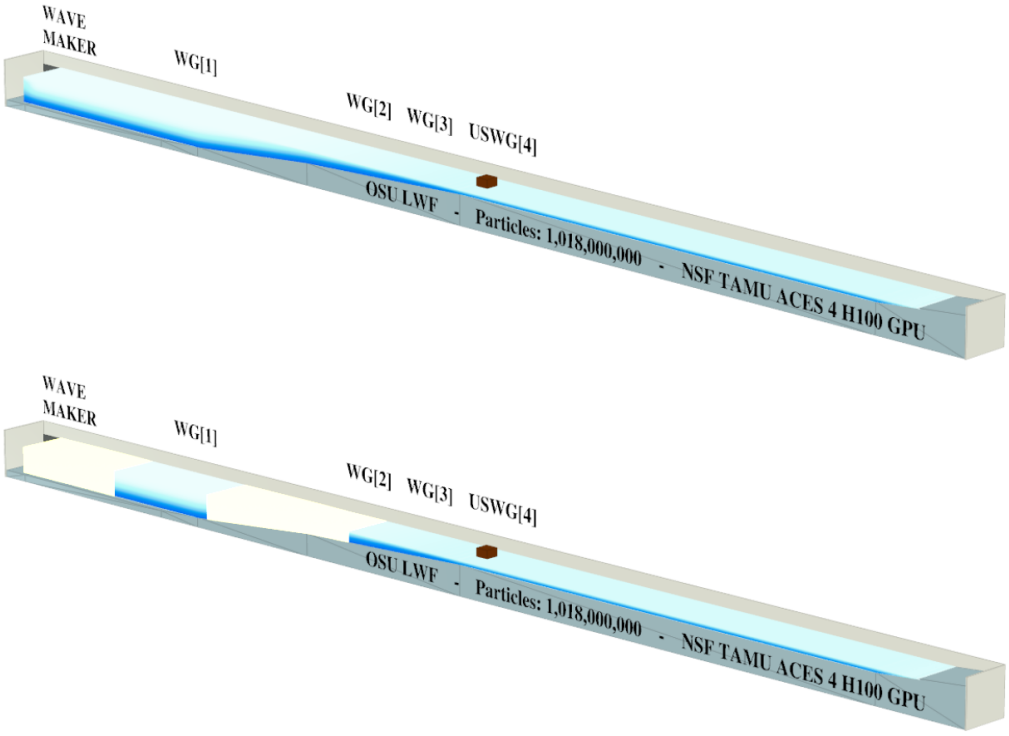

Fig. 7.29. Billion particle wave-flume digital twin simulates a pressure wave. Open-source digital twin of the OSU LWF, implemented in ClaymoreUW Multi-GPU MPM, holds a billion water particles undergoing a million time-steps. (Top) Initial pressure wave travels through the fluid as gravity suddenly loads the wave-flume. (Bottom) Particle distribution over four H100 PCIe 80GB GPUs on Texas A&M's ACES system. Individual GPUs hold particle counts of 240 to 275 million.

Figure 7.29's top image is a snapshot of an initial pressure wave's motion through a billion gravity-loaded fluid particles. The bottom scene visualizes the four GPU partitions, each device holding 240 - 270 million double-precision particles, across which gigabytes of data are transferred every time-step to approximately solve the underlying differential equations.

Our exa-scale OSU LWF was modeled in ClaymoreUW Multi-GPU MPM on a single node with four GPUs (NVIDIA PCIe H100 80 GB). The node is hosted on Texas A&M's ACES high-performance computing system, which we were able to access via DesignSafe and ACCESS / XCEDE resources. The exponential growth in our computational capabilities from the beginning

of this project three years prior is greatly dependent on the massive and growing parallel scales offered by GPU and interconnect hardware year-over-year, as discussed in Chapter 3. We emphasize the profound importance that NSF funded community initiatives for high-performance computing has played in research. Namely DesignSafe, NHERI SimCenter, and ACCESS / XCEDE. These have proved to be invaluable resources which must be maintained. Firstly, in the network of talented multi-disciplinary researchers that are brought under these funding umbrellas, and secondly, in order to allow natural hazard engineering to reap the unknowable utility within the exa-scale to form a bulwark against the inevitable hazards of the coming decade.

7.8 *Conclusions*

This research demonstrates the integration of advanced computational techniques with coastal engineering through the application of a Multi-GPU Material Point Method, implemented in the open-source ClaymoreUW software. We successfully simulate wave-driven debris field dynamics in a digital twin of the Oregon State University's Large Wave Flume (OSU LWF). The results presented are compelling and highlight the potential of high-performance computing in advancing the understanding of complex, real-world engineering problems.

- Debris Impact Forces: The simulation's ability to predict debris impact forces within experimental inter-quartile ranges is a significant achievement, as it offers a reliable numerical tool for understanding debris interaction with coastal structures during inundation events.

- Debris Damming Forces: While the general behavior of debris damming was successfully replicated, the under-prediction of sustained damming forces but over-prediction of intermediate damming forces (between first and second debris-field impact peaks) indicates the need for refinement in the debris-fluid interaction within the model. This area represents

an opportunity for further research, potentially through higher-resolution modeling or incorporation of additional physical processes.

- Wave Hydrodynamics: The replication of wave hydrodynamics within the digital twin of the flume was convincingly achieved. The model can simulate fluid motion in a realistic manner, which is crucial for analyzing inundation events and their effects on coastal structures.

- Open-source Software and Digital Twin Model: By making the software and digital twin model openly accessible, this research contributes valuable tools to the scientific community. The integration of computer graphics techniques into engineering simulations is likely to foster innovations in computational methods for coastal engineering and beyond.

Future research should focus on enhancing the resolution of debris-fluid interactions and examining additional physical processes to improve the prediction of debris damming forces. Additionally, the model could be extended to explore a broader range of debris types and sizes, and further validation against real-world tsunami data could be pursued.

In summary, the Multi-GPU Material Point Method presented in this study stands as a promising approach for the numerical simulation of inundation-driven debris fields. While refinements are necessary, particularly concerning debris damming forces, the model represents a substantial step forward in the convergence of computer science and coastal engineering. As coastal communities globally face the growing threat of natural disasters, these advancements are pivotal in the quest for developing robust solutions for disaster mitigation and resilience building.

In this chapter the physical specification and numerical replication of experiments performed at Oregon State University Large Wave Flume (OSU LWF) are described. Facilities are recreated in our numerical open-source MPM software, at a technologically novel level using Multi-GPUs. Numerical simulations are characterized and preliminary data-analysis of results are given with comparison to real-world results for validation of our approach. Preliminary extrapolation to the

prototype scale through similitude scaling suggest a significant importance in the quantification of debris-field impact force relative to wave-only force. Our approach has shown an ability to replicate tests, predict tests not yet performed within expected results bounds, and may be of value in the prediction of prototype scale debris-field impacts in the real-world for tsunami events.

Chapter 8

MOTION OF TSUNAMI DEBRIS THROUGH A MODEL PORT CITY - THREE NUMERICAL METHODS WITH EXPERIMENTS - WASEDA UNIVERSITY FLUME

This chapter concerns an ongoing numerical, comparative study on tsunami debris between researchers in the United States and Germany. Documented herein is the most recent results provided by all groups, with analysis done from our individual study on the Material Point Method. A manuscript is to be submitted and will contain the final results of all groups in the months following this dissertation.

This document serves to critically examine the value of three numerical modeling methods for next-generation, high-performance tsunami debris loading and motion simulations. Further, three digital twins of the pressurized dam-break wave-flume at Waseda University in Tokyo, Japan are created. The tsunami debris wave flume experiments of Goseberg et al. 2016 are replicated. All implementations are High-Performance computationally (Multi-GPU/Multi-CPU), with nearly identical settings synced across each. Using the Material Point Method in our open-source ClaymoreUW software is myself, Professor Mike Motley, and Professor Pedro Arduino. Smoothed Particle Hydrodynamics within open-source DualSPHysics is ran by Felix Spröer, Clemens Krautwald, and Professor Nils Goseberg. Siemens STAR-CCM+ CFD is represented by Andrew Winter.

8.1 *Introduction*

We present an international comparative analysis in simulating 3D tsunami debris hazards on three state-of-the-art numerical methods: the Material Point Method (MPM, ClaymoreUW, Multi-GPU), Smoothed Particle Hydrodynamics (SPH, DualSPHysics, GPU), and Eulerian grid-based fluid dynamics (Siemens STAR-CCM+, Multi-CPU / GPU). Three teams, two from the United States and one from Germany, apply their unique expertise gained in industry and academics to shed light on the state of advanced tsunami debris numerical modeling in open source and professional software. A mutually accepted and meaningful benchmark is set as a 1:40 Froude scale model of shipping crates mobilized into and amidst a port city's structures, as observed in seminal Tohoku 2011 tsunami case histories. The sophisticated wave-flume at Waseda University in Tokyo, Japan hosts the experiments. Across dozens of trials, an elongated dam-break wave surges and spills over a harbor, mobilizing 3 - 6 hollow debris in 1 - 2 vertical layers against friction. 0 - 2 rows of 5 square obstacles are placed upstream or downstream of the debris, with widths and gaps of 0.66x and 2.2x of debris length, respectively. Teams report results on irregular wave generation, longitudinal displacement of debris forward and back, lateral spreading of debris forward and back, interaction of stacked debris and impact forces measured with accelerometers or obstacles. Each team writes a foreword on their digital twin model, which are all open-sourced. Then, preliminary statistical analysis contrasts simulations with experiments. Afterwards, team's give value propositions for their numerical tool. Finally, a transparent cross-interrogation of results is performed.

The principal aim of this research is not only to evaluate the efficacy, precision, and reliability of these numerical techniques in modelling complex debris-fluid-structure interactions, but also to discern the degree to which obstacles affect the maximum longitudinal displacement and spreading angle of debris. The outcome of this research promises to yield insights that will guide the selection of simulation techniques in the study of debris motion under extreme coastal

conditions. In doing so, it lays the foundation for more effective disaster management strategies, which holds the potential for far-reaching impacts on coastal communities worldwide.

8.2 *Literature Review*

Common guidelines for tsunami-driven debris-field impacts originate from Naito et al. 2014 and Chock 2016, among other seminal works, which were mainly based on a limited set of experiments and case studies, namely the Tohoku 2011 tsunami (Yeh et al. 2013). The study of tsunami-driven debris-fields in a more generalized phenomenon-centered context has been experimentally pursued in recent years using scaled wave-flume facilities (Stolle et al. 2016, Goseberg et al. 2016, Park et al. 2021, Shekhar et al. 2020, Kennedy et al. 2021, Mascarenas 2022, Moris et al. 2023, and Cinar et al. 2022) which aim to understand the hazard at a reduced model scale. If done carefully, extrapolation of model-scale conclusions to a prototype tsunami scale may be possible. A handful of researchers have recently sought to recreate experiments numerically, as in Yang et al. 2017a and Hasanpour et al. 2021, eventually to assist in extrapolating them to prototype scales. These efforts logically lead to various attempts to lay the ground-work for scale-invariant and probabilistic design frameworks for tsunami-driven debris-field hazards using probability (Nistor et al. 2017a) for performance-based analysis (Bonus et al. 2022). Although there is no consensus on the most effective and efficient numerical approach to tsunami debris, as identified by Takabatake et al. 2021 and Nistor et al. 2017a, two-way coupled solid-fluid models, smoothed particle hydrodynamics (SPH) and material point method (MPM) schemes show promise and have enjoyed growing popularity. Furthermore, the apparent need for high-resolution 3D debris-fluid-structure interaction models has prompted high-performance multi-CPU and GPU adoption in open-source tools, notably CB-Geo MPM (Kumar et al. 2019), Karamelo (Nguyen et al. 2023), DualSPHysics (Domínguez et al. 2022), Claymore (Wang et al. 2020b), and now ClaymoreUW.

Comprehensive and extensive fluid suites, such as STAR-CCM+, are applicable to a wide-range of phenomena and enjoy professional use at scales not truly matched by comparatively academic methods like SPH and MPM. Though SPH is seen increasingly in software like LS-DYNA, it has also began to occupy the niche of coastal engineering. Open-source SPH code DualSPHysics Domínguez et al. 2022 even supports various wave-maker schemes to assist in wave flume and wave event modeling for coastal hazard analysis. Meanwhile, MPM's primary engineering point of adoption has been geotechnical events involving large-deformations, as in Zhang et al. 2023, Zheng et al. 2021, Tran et al. 2022, Tran et al. 2023, Qiao et al. 2023 and Mast 2013, while its public-facing position has been as a scheme for computers graphics simulations, as in Disney's animated movie Frozen (Stomakhin et al. 2013) and the broader research community which has proposed modifications in Jiang et al. 2015, Fu et al. 2017, Hu et al. 2018, and Fei et al. 2021a for seamless improvement of visuals which offer merit to engineering problems.

MPM and SPH are seen to occupy a similar range of capabilities, both showing strong and weakly-compressible particle method capabilities for a wide range of materials, see Issa et al. 2010 and Chen et al. 2018, but are distinct in their key strengths. MPM finds its stride in simulation of complicated, history-dependent multi-material interaction at large-deformations. However, SPH favors multi-phase and free-surface flows with comparatively good pressure-field determination. A few comparative studies have been done on SPH versus MPM. The most relevant to tsunami debris is Sun et al. 2018, whose authors found both SPH and MPM to be applicable to replication of fluid experiments but noted that MPM is computationally favorable over SPH. This is because it required no neighbor search algorithm, which tends to scale-up in operations required, slowing simulations, than MPM's particle-grid scatter operations. They also stated that MPM slightly outperforms SPH in accuracy for their fluid cases. We do not personally share the opinion that MPM is superior to SPH for fluids as they studied small particle counts on implementations that are not fully representative of modern usage of either tool. MPM is

commonly in low-order forms proposed by Jiang et al. 2016 and Hu et al. 2018 to achieve high computational performance that may degrade stiff fluid pressures while SPH is refined for use in high accuracy fluid simulations needed for shoaling of nonlinear waves and cavitation studies. Although this is a generalization on our part, all participants in this manuscript's comparative study consider SPH, not MPM, to be the more logical choice in pure fluid studies. On the contrary, MPM sees better results in collisions of complex solids versus SPH, as shown by Ma et al. 2009 who found MPM to have an edge over SPH in hypervelocity impacts, and general large-deformations of complicated solids (e.g. splintering wood, plastic metal, elasto-plastic soils) and mixed phases (e.g. partially liquefied debris-flows) while keeping interactive capability with fluids. This suggests MPM may have untapped potential in quantifying tsunami debris loads on structures, assuming the driving fluid can be modeled on-par with SPH. While SPH is not inherently strong for modeling complicated solid materials, there are ways to model non-flowing materials. It is often seen that SPH tools are coupled with DEM or FEA to handle solids such as debris and structures. For instance, DualSPHysics by Domínguez et al. 2022 incorporates solids from Khronos which is applied in this manuscript for tsunami debris.

STAR-CCM+ takes a similar approach with coupled rigid bodies used for debris, though its professional and reliable fluid capabilities are far more developed than open-source projects typical in SPH. MPM and STAR-CCM+ do not overlap in their common use, and it may be that they are better seen as compliments than competitors, but we contrast their ability to simulate full tsunami debris wave-flume experiments here while noting that coupling high-performance MPM into STAR-CCM+ would be of value.

A critical review of the state of the art in tsunami-debris modeling was provided by Nistor et al. 2017a. Key conclusions include the need to: **(i)** determine appropriate experimental and numerical scales for turbulence (She and Leveque 1994) and debris dynamics (Nistor et al. 2017a), **(ii)** account for robust hydrodynamic boundary conditions numerically, **(iii)** provide

rigorous high-resolution studies for 3D model calibration, **(iv)** investigate fundamental debris-debris and debris-fluid behavior in the context of debris entrainment and momentum transfer, **(v)** support of numerical probabilistic approaches (i.e., develop high-performance tools to simulate at higher frequencies) due to the stochastic nature of debris motion, **(vi)** and numerically characterize the demand of multiple debris impact and damming events on structures.

In Yang et al. 2017b the authors employed MPM to examine the impact of tsunami-driven debris on bridge superstructures. The study found that the influence of contact area and impact eccentricity on debris-structure interaction is mitigated with increasing debris longitudinal ratio, and highlighted that tsunami-induced debris impacts can increase impact forces by up to 35% compared to debris impacts in air estimated using empirical equations (Aghl et al. 2014b; Aghl et al. 2014a; Paczkowski et al. 2012) for equivalent debris velocities. However, this finding applies to cases where a buffer of water does not develop between the debris and structure at impact and thus is primarily limited to highly buoyant debris impacting elevated structures. When a fluid buffer is observed between the structure and debris, as in Mascarenas 2022 and Shekhar et al. 2020, loads are experimentally recorded to decrease relative to in-air calculations. This suggests that while the debris-fluid interaction can increase loads, DFSI often occurs across a difficult to predict fluid buffer between debris and structures at impact and may dampen loads in a scale-dependent way, suggesting a need for further numerical DFSI study.

Hasanpour et al. 2021 coupled Smoothed Particle Hydrodynamics with the Finite Element Method (SPH-FEM) model to simulate the interaction of single debris in a wave-flume environment, as well as loads on scaled coastal structures. It found acceptable accuracy across variables, with deviations of maximum wave height and fluid velocities between 4-22% from experimental tests, and predicted debris impact velocities were 20% higher. The maximum predicted impact forces of the debris on the structure deviated 14-25% from the experimental data. Interestingly, debris allowed to tilt obliquely to streamwise flow produced higher impact velocities

but smaller impact forces, approximately 56% less, compared to restrained debris due to oblique impact angles as observed in Yang et al. 2017b. The study suggests considering water depth and velocity range in developing future debris load equations, acknowledging the need for high-resolution 3D models to better account for the debris' rotational movement– a characteristic supported in this comparative study.

Aslami et al. 2023 applied SPH to solve shallow-water equations (SWE-SPH) with solid boundaries in DualSPHysics coupled with the Discrete Element Method (DEM) to simulate and predict debris behavior and hazards in tsunami-like conditions, providing good agreement with experimental data and analytical equations for simple debris materials and geometries. Shallow-water equations may assist in off-loading computation from SPH (or MPM) while improving relative results for far-field wave-generation. It would be of interest to see a coupled implementation Abdolali and Kirby 2017 of SPH or MPM in future works to simulate compressible Boussinesq waves in the far and near-shore while the particle based methods are used for the increased complexity found in the reduced computational domain of on-shore tsunami debris loading on structures.

Wu et al. 2014 introduced a two-way coupled dynamic numerical model for moving solids in free-surface flows. They utilized a Large Eddy Simulation (LES) model, the Volume-of-Fluid (VOF) method for tracking the free surface, the Discrete Element Method (DEM) for debris displacement and rotation, and it models the fluid response from solid motion via cell-face velocity with partial-cell treatment (PCT). The effectiveness is validated through two laboratory experiments, showing accurate predictions of the trajectory of moving blocks in a water tank, similar to our own debris-fluid interaction validation case in Sec. 6.8.

Participants in a tsunami debris modeling workshop compared different approaches to numerically simulate tsunami debris (Takabatake et al. 2021). Accurate modeling of the tsunami-induced flow field was crucial, especially for debris on the ground, as the changing drag and inertia coefficients

affected the speed and trajectory of the debris. Debris-debris interaction was a significant phenomenon worth refining, as models tended to incorrectly predict the lateral spread of debris. Two-way coupled simulation, which include the debris-fluid interaction drag force, were identified as favorable for studying larger debris. They suggested incorporating 3D models and exploring probabilistic approaches to address variations in debris trajectory.

All of these studies have pushed the field of numerical tsunami debris modeling forward in their own right. However, challenges persist in high-resolution 3D models (100+ million numerical bodies), multiple debris interactions in fluid-structure deformation and topology changing environments, hyper-elastic and elasto-plastic debris and structural material models, and handling arbitrary debris and structure geometries. This work aims to elucidate some of these issues, demonstrated across a comparative study of three numerical methods that show great promise for addressing these challenges and advancing the field of numerical tsunami-driven debris-field hazards.

8.3 *Facility Specifications*

Facility details are described in the original experimental paper by Goseberg et al. 2016, with additional information found in (Stolle et al. 2016) and (Nistor et al. 2017a). Figure 8.1 is a schematic of the Tsunami Wave Basin at Waseda University, Tokyo, Japan, as it was configured for the aforementioned experiments. All flume instrumentation information (e.g. location, sampling rate) is aggregated in Table 8.2. Further description of the flume is noted in below sections when relevant to assumptions made or limitations noted by respective creators of each numerical digital twin.

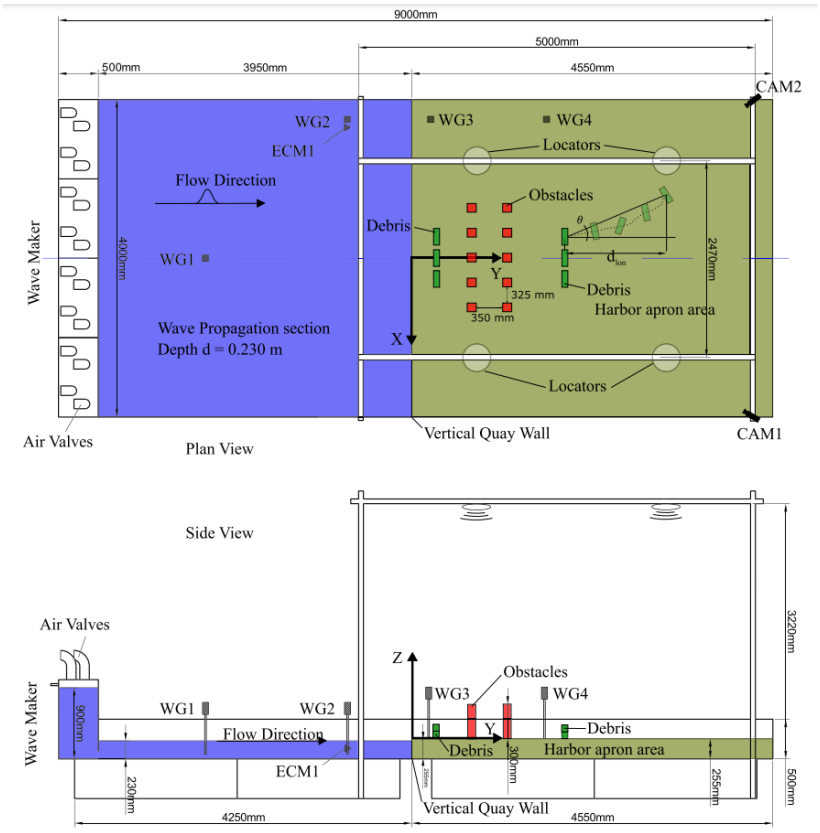

Fig. 8.1. Schematic of Waseda University's tsunami wave basin. Configuration used in Stolle et al. 2016 experiments and our comparative numerical study.

8.4 *Numerical Methods and Digital Twins*

A summary of each of the three numerical method's applied for replication of the debris experiments by Stolle et al. 2016 is given by respective authors. Details on their digital twin model, experiment assumptions, and known limitations of their numerical approach are provided briefly here. As this dissertation chapter is written by the user of the MPM approach while the comparative analysis of methods is still underway, only MPM relevant sections are fully described. Further exposition by the respective authors of SPH and STAR-CCM+ results is intended to appear in a published article in the coming year following this dissertation, with

Fig. 8.2. Tabulated instrumentation in Waseda University flume experiments. Instrumentation used in Goseberg et al. 2016.

Instrument name	Instrument ID tag	X [m]	Y [m]	Z [m]	Sampling rate [Hz]	Notes
WG1	CHT6-30-1	0.00	−2.60	–	100	
WG2	CHT6-30-2	−1.75	−0.81	–	100	
WG3	CHT6-30-3	−1.75	0.24	–	100	
WG4	CHT6-30-4	−1.75	1.70	–	100	
ECM1(X)	VMT2-200-04P	−1.65	−0.81	−0.10	100	Collocated with WG2
ECM2(Y)	VMT2-200-04P	−1.65	−0.81	−0.10	100	Collocated with WG2
CAM1	pi1900-32gc	2.00	4.20	2.95	25	Pointed to area of interest
CAM2	pi1900-32gc	−2.00	4.20	2.95	25	Pointed to area of interest
LOC1	9			3.15	22–48	Downward looking
LOC2	10			3.15	22–48	Downward looking
LOC3	11			3.15	22–48	Downward looking
LOC4	12			3.15	22–48	Downward looking

partial results presented in a limited capacity.

8.4.1 Material Point Method

The Material Point Method (MPM, Sulsky et al. 1994) is applied to study these flume experiments. We use a Moving-Least-Square Material Point Method (MLS-MPM, Hu et al. 2018) with an assumed deformation volumetric anti-locking scheme (i.e. B-Spline F-Bar from (Zhao et al. 2023), see Section 5.7) with our custom modifications discussed in Sections 5.8, 5.9, and 5.10 for improved stability and compatibility with the G2P2G fused kernel used in ClaymoreUW's MLS-MPM implementation..

We model the flume walls, harbor, and obstacles as rigid boundaries with separable velocity conditions on the MPM background grid. The divider between the main flume and the water reservoir column is taken as 0.05 meters thick to avoid fluid particles from interacting across the barrier, and this appears to roughly equate to the length of the "lip" at the reservoir gates edge so channeling of the outflow is similar. We neglect back-flow over the lip that could produce moderate turbulence in the area between the thin reservoir wall and the lip's end. The reservoir opening, i.e. sluice gate, is a 0.1 meters gap in the boundary condition that separates the reservoir

wall from the flume's base. The vertical quay harbor wall, i.e. the front panel separating the water basin and the dry harbor apron, is 0.255 meter high and applies a separable velocity boundary condition on streamwise flow only. The grid-nodes at the exact corner of wall and harbor platform were beveled to so as to favor neither the wall or harbor apron, so as to not impede the over-topping of the wave onto the harbor. The harbor apron is horizontal and applies a separable velocity condition against any downwards vertical motion. It also applies Coulomb static and dynamic friction to orthogonal sliding motion on the condition that there are solid debris particles reading or writing information from a grid-node within the harbor apron's boundary domain. This is to prevent spurious friction damping the fluid inundation over the harbor, though we note that a small amount of added friction is applied to the water that is within a grid-cell of the debris as a side-effect of the shared velocity field in MPM. This could be accounted for by using multiple velocity field's, but is not pursued in this implementation. The harbor continues until it reaches the back wall of the 9 meter flume.

The square column obstacles are placed as in Goseberg et al. 2016 for applicable cases. They are assumed to be rigid and apply a separable boundary condition respective to each face of the square column to the MPM grid. We use beveled surface normal vectors (e.g. corner XZ has a normal pointing at 45° on axis X and Z) for edge grid-nodes to prevent over-representation of the leading face's geometry onto the fluid inundation and to minimize the thickened laminar flow boundary layer that can occur in MPM around naive box boundaries.

Load-cells are numerically set on the leading face of obstacles, recording the forces imposed streamwise by computing load from momentum change at a grid-node caused by the obstacles boundary condition. Although the experiments did not have load-cells on obstacles, only moderate sampling frequency accelerometer estimate forces acting on debris bodies implanted in said debris, we model them here as we were interested in evaluating the quality and content of motion-based force data in experimental and SPH debris accelerometers to the total load imposed

on the various obstacles acquired in MPM.

Water is modeled with our high-accuracy, high-performance sub-J Fluid formulation. It is mathematically identical to an isotropic weakly-compressible fluid using the Murnaghan-Tait equation from Murnaghan 1944 and Tait 1888 as it was applied in Pradhana et al. 2017 for the inclusion of viscous stress. Density is 1000 kg/m^3 and bulk modulus is 2.2 GPa. Viscosity is 0.1 centi-poise (0.001 Pa * seconds) and no turbulence model is applied . Standing water level in the basin is set to 0.23 meters and water elevation in the reservoir is 0.9 meters. No initial pressure field is set in the reservoir or basin. F-Bar volumetric antilocking Zhao et al. 2023 is used on water particles

Debris are modeled as fixed-coroted solids (Jiang et al. 2016). We choose to model debris as solid blocks instead of hollow, as done in the experiment. This is to avoid using thin walls defined by only 1-2 particles. Density is set at 419 kg/m^3, instead of the experimental 921 kg/m^3, to account for applying all mass of the debris HDPE + instrumentation to the solid volume. Poisson ratio is to 0.3. Young's modulus is 1 GPa.

8.4.2 Smoothed Particle Hydrodynamics

Exposition for the SPH approach to be provided in this work's manuscript form. Software used is DualSPHysics from Domínguez et al. 2022.

8.4.3 STAR-CCM+

Exposition for the STAR-CCM+ approach to be provided in this work's manuscript form.

8.5 *Comparison*

This section contrasts numerical results in MPM, SPH, and STAR-CCM+ and from the experiments of Goseberg et al. 2016. The following text segments comparison along five

categories of special interest tsunami debris modeling: **(i)** Hydrodynamics, **(ii)** Debris Longitudinal Displacement, **(iii)** Debris Lateral Spreading Angle, **(iv)** Debris Motion, and **(v)** Debris Accelerations / Obstacle Loads.

8.5.1 Hydrodynamics

The first-step in validation of numerical flume results is the validation of a method's capability in resolving the wave-form throughout the flume's length. Elevation results for the four wave-gauges along the flume are shown in Figure 8.3 and discussed below.

Wave-gauge 1 is placed near the wave-maker so as to capture the initial fluid elevation surge from the dam-break reservoir and the elongated tail before the wave has moved near the harbor apron. The goal of this wave-maker configuration is to generate "wave-trains" as the irregular tsunami wave model, and thus they are preferred to have a steep front which is reasonably taken as a solitary wave while the tail elongates to replicate the long time-period of real tsunamis. Of course this facility can't achieve an ideal model time scale of the wave relative to its amplitude, as prototype tsunamis can last hours, but for the initial inundation that concerns debris impacts it only needs to scale up-to 10 - 30 seconds.

Wave-Gauge 2 is still over the basin but near to the harbor quay wall. We observe a slightly increased wave-peak which tapers off in an elongated fashion. Half a second after, reflected waves from the quay wall begin to dominate elevation.

Wave-Gauge 3 is over the harbor and positioned just before the debris. Replicating its measurements is important as they govern initial mobilization of debris against friction. The two peaks respectively are the irregular waves initial surge over the wall, whereas the second is from the elongated train of fluid that must transfer energy through the front's reflection. There is effectively no inundation at this gauge after 2.75 seconds of arrival, equating roughly to $T_{Inundation} = 2.75\sqrt{40} = 17.4$ seconds. This is a reasonable time-scale for observing debris motion

and impacts during initial inundation.

Wave-Gauge 4 is beyond both obstacle rows but prior to the downstream debris position. Again, replication is vital to mobilize downstream debris correctly. This gauge will also ascertain if a method can create a persisting, thin flow over the dry horizontal harbor without losing its primary two peaks to numerical damping (e.g. from coarse grid-resolution near the harbor apron's surface). Even attaining a thin flow in the first place requires the tempered computation of a dam break fluid mass flowing below a gate, forming a solitary front on the other side, having said front travel across the basin, reflect partially off the vertical quay wall while the remainder overtops the surge that develops at the wall, and then said surge must flow across a frictional (for debris) harbor apron, between obstacles, until reaching the wave-gauge far inland.

MPM is seen to fair respectably in initial hydrodynamic replication in Figures 8.3, 8.4, 8.5, 8.6, and 8.7, quantified by first peaks at four wave-gauges constrained within 4 mm to 2 cm of deviation for grid-cells of 1 cm. This is unexpectedly positive as our MPM implementation lacks a turbulence model. Interestingly, MPM's laminar flow introduced an initial dam-break damping that resembles the brief reservoir pressure release lag our group suspects elongated the wave during the experiments of Stolle et al. 2016, though not documented. However, MPM does not elongate the initial wave at WG1 into a plateau from 1-1.5 seconds which may limit time-scales that the model irregular wave can apply to prototype events. The reduced wave-tail reaches WG2 down the basin but no longer appears shorten time-scales. On the way to WG3, MPM produces a surge at the quay wall during wave impact. This allow subsequent spilling of the trailing wave-train onto the harbor when it adds to the surge mass. Simulated MPM measurements arrives at WG3 (before debris) and WG4 (after obstacles) on the flat harbor. At both, MPM characterizes thin flows accurately in depth and and speed, demonstrating that an ideal elongated tail at WG1 and WG2 is not required to attain the critical elongated thin flow on the harbor for studying debris. The thin flow starts on and sustains large debris mobilization along the frictional harbor surface

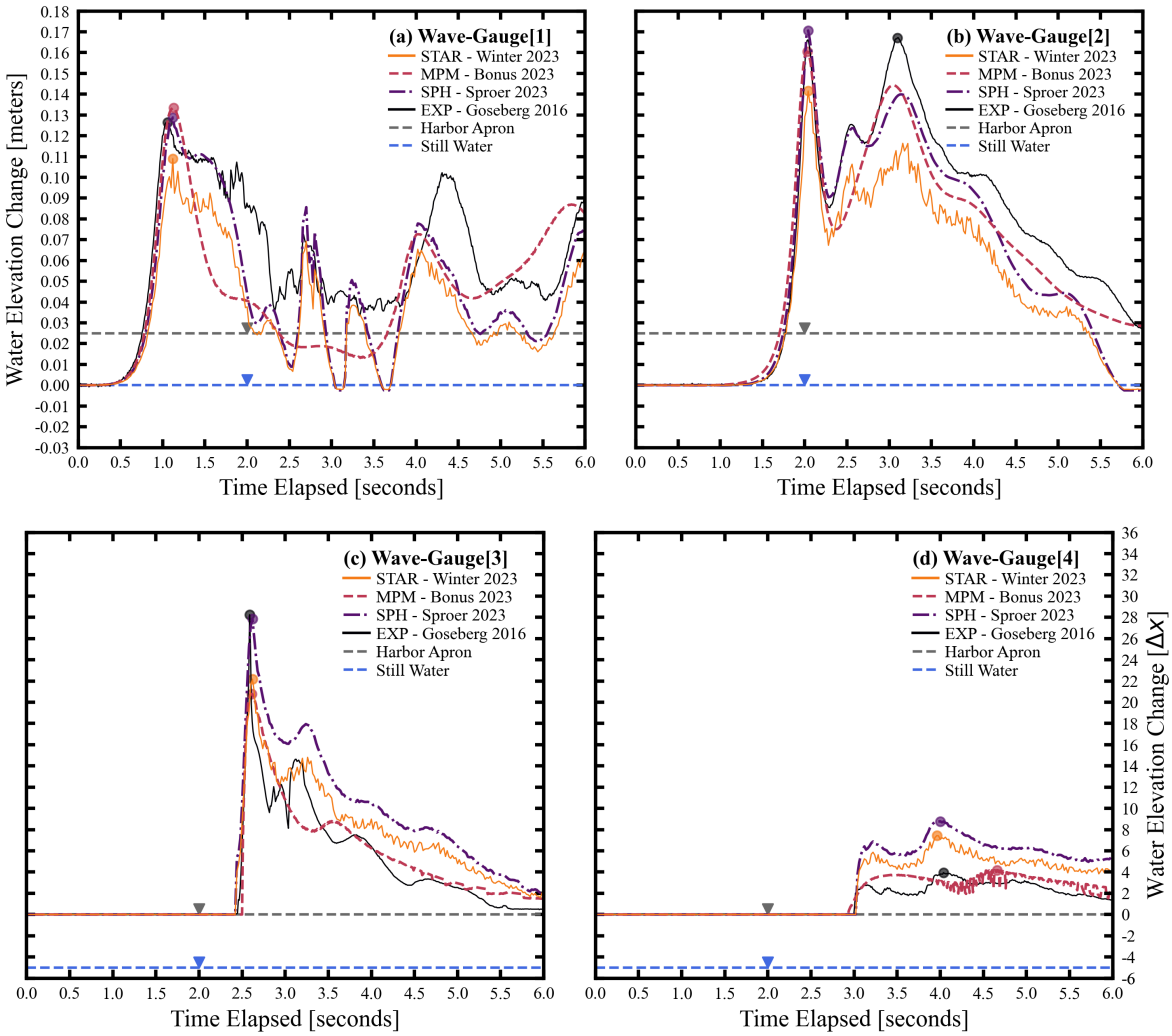

Fig. 8.3. Free-surface elevation of numerical methods and experiments at four wave-gauges in Waseda University's tsunami wave basin. Based on experiments by Stolle et al. 2016. Compares the Material Point Method, Smoothed Particle Hydrodynamics, Siemens STAR-CCM+, and experiments by Stolle et al. 2016.

despite imperfect free-surface readings in the basin. An insufficiently strong (i.e. weak) flow will not be able mobilize the debris. Likewise, an overly heavy flow would over-mobilize debris and ruin comparisons to stochastic experiments . First peaks of the flow at WG3 and WG4 are very good and usable respectively. They are constrained within 4 to 8 mm, the effective limit of 1 cm

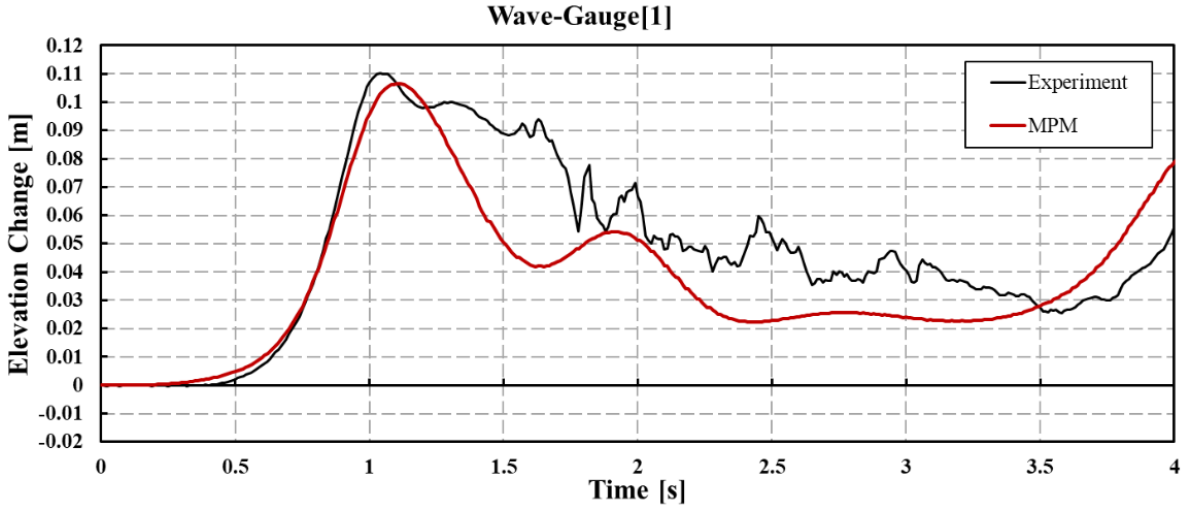

Fig. 8.4. Wave-gauge 1 free-surface elevations in Waseda University's tsunami wave basin.

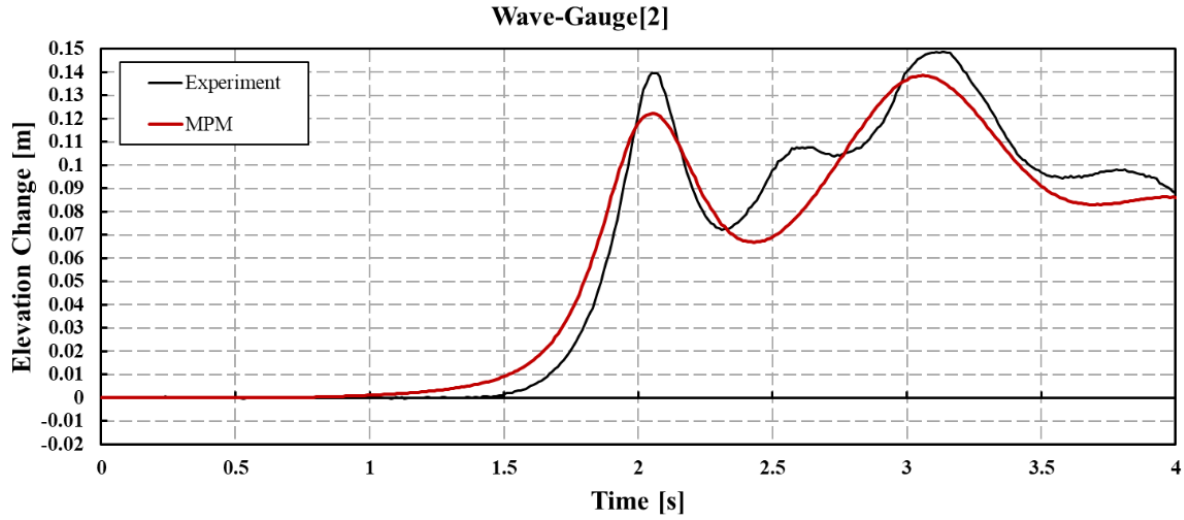

Fig. 8.5. Wave-gauge 2 free-surface elevations in Waseda University's tsunami wave basin.

grid-cells for fluid flow. However, frequency of waves on the horizontal harbor shifts down past 3.5 seconds, suggesting some damping accrued though it is not inherently detrimental. STAR-CCM+ accomplished the same with turbulence included, as did SPH with a caveat that its thin flow appears to be up-to twice the experimental depth, potentially signifying floating point

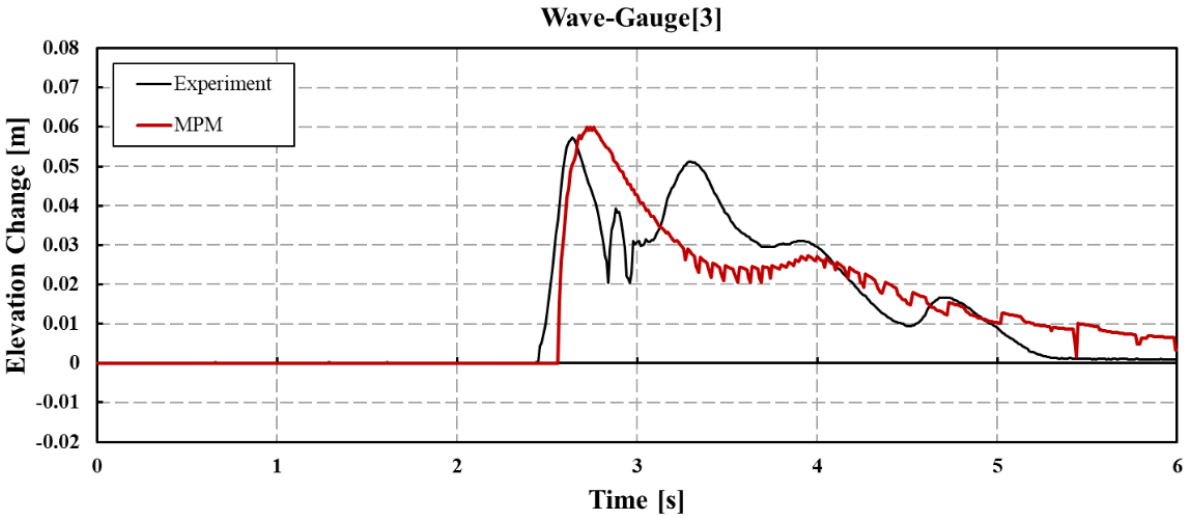

Fig. 8.6. Wave-gauge 3 free-surface elevations in Waseda University's tsunami wave basin.

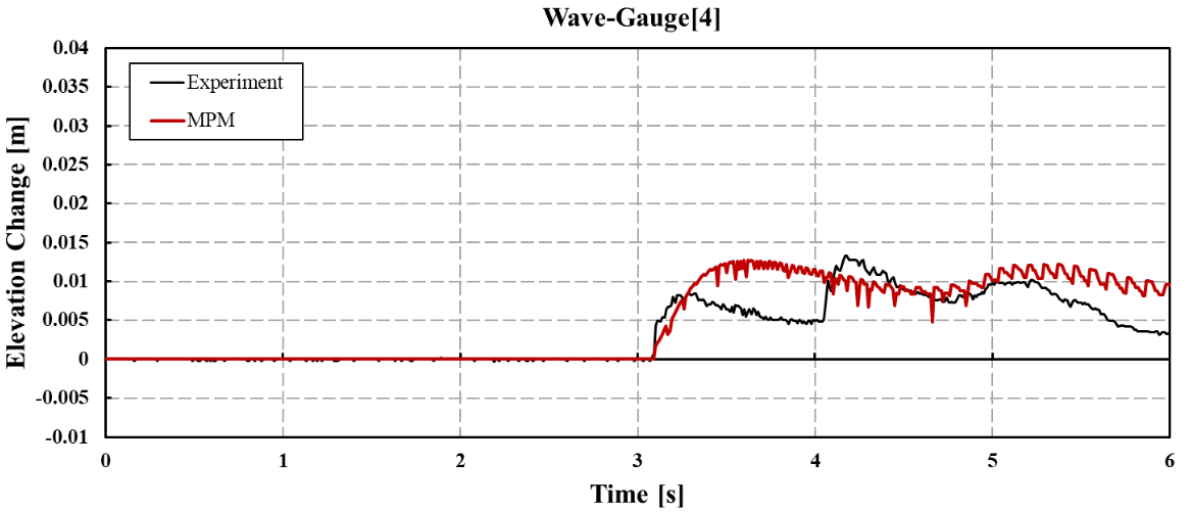

Fig. 8.7. Wave-gauge 4 free-surface elevations in Waseda University's tsunami wave basin.

pressure errors in thin flows of nearly-incompressible fluids in single precision. Use of single precision was confirmed by the SPH team as they used an NVIDIA 3090 GPU to achieve their results, an RTX card which strongly favors single-precision (as opposed to HPC hardware). Tangentially, our proposed PA-JB fluid in Section 5.12 may be a potential fix to the

precision-sensitive equation of state used in DualSPHysics. We note that MPM has quicker decay following the elongated dam-break waves front when compared to experiments (which may have used a timed release to elongate waves), SPH results (which applied an empirical velocity boundary condition in the reservoir to match experiments), and STAR-CCM+ (which applied a pressure boundary condition after a study on the pump-valve mechanisms of the flume's reservoir).

8.5.2 Longitudinal Displacement

The longitudinal displacement of adjacent and stacked shipping crates in harbor ports were observed during the Tohoku 2011 tsunami. These large debris have simple geometries, pose significant hazards, begin in logical arrangements, and were uniquely identified by serial numbers which allowed researchers to estimate the motion of each debris Naito et al. 2014 during inundation. These value data-points gave preliminary bounds on high hazard debris-field impacts as a function of a structures distance from the debris source and inland from the coastline and expected inundation flow vector. Flumes like Waseda University's tsunami basin allows experiments, such as Stolle et al. 2016 and Nistor et al. 2017a, to control important variables in these events (i.e. spacing of adjacent buildings) for scaled down models. This has allowed discovery of empirically tuned equations for predicting chaotic debris motion events, though they can be limited to certain scales unless considerable effort is employed in similitude control. Numerical replication of experiments and tsunami events gives engineers another tool to explore the variables believed to govern this important phenomena, and further, a means to later explore any of the dozens of parameters that may need to be investigated during a high-quality similitude up-scaling of wave-flume results (e.g. Reynolds, Weber, Strouhal, Mach, constitutive). However, multiple numerical contenders (MPM, SPH, STAR-CCM+) show promise and thus we undertake an international, quantitative comparison of advanced numerical methods for estimating tsunami debris longitudinal displacement. We will be assessing motion of a few large debris (scaled

shipping crates) through adjacent obstacles (a scaled port city) with wooden floor friction (proxy for pavement/asphalt) impeding displacement with respect to the wetting of the interface.

To begin, we consider an experimentally fitted predictive model from Nistor et al. 2017a, defined as:

$$\Delta_{\text{Long.}} = 3.58 - 0.09N \ , \tag{8.1}$$

where N is the number of debris in the trial. This is a reasonable prediction on median longitudinal displacement, based solely on the number of debris and trends observed in the Waseda University flume, though it is limited in this form to just the Waseda flume and similar specification debris. Note the constant of 3.58 only makes sense for the harbor length in the Waseda flume, it is not for prototype tsunami design. The negative linear trend with debris count N does imply a more general tendency for larger groups of debris to aggregate and dissipate momentum in a way counterproductive to longitudinal displacement, which is worth keeping in mind. This basic predictive model will serve as a sanity check outside of experimental data from Stolle et al. 2016 so we may know if our numerical methods are wildly incorrect (e.g. never exceeding 10% of Equation 8.1).

Figure 8.8 displays all numerical results, experimental trials, and fitted predictive equations considered for debris longitudinal displacement. It is segmented by the number of obstacle rows and the starting position of the debris-field. Box-and-whisker plots show experimental medians, quartiles (Q1, Q3), and outlier points according to whisker caps at (Q1 - 1.5*IQR) and (Q3 + 1.5*IQR), where IQR is the interquartile range (IQR = Q3 - Q1). All individual experimental and numerical trials are included as points for completeness. The fitted prediction, Equation 8.1, from the experiments of Nistor et al. 2017a is superimposed to provide a general guideline for typical debris motion behavior that exists outside of Stolle et al. 2016. Said predictive model also shows that the experiments of Goseberg et al. 2016 are close (percent error of 12%) for upstream debris with no obstacles, but experiments feature less displacement when testing rows of debris and/or

downstream position debris-fields as neither are part of the predictive model. The highest allowed value in the plot is at the back wall of the flume and thus provides and upper bound on displacement. A blue hatch is plotted at -0.2 meters and below to represent the lower-bound of the water basin. Downstream debris have been shifted to begin adjacent to upstream debris in our visualization. Note that they are placed physically further in the flume, past any obstacles, and thus experience different wave loading.

All MPM simulations performed for debris-field cases in Goseberg et al. 2016 have final debris forward inundation longitudinal displacement added to the plot. Measurements are taken when debris attain rest or hold velocities below 5 cm/s as some short simulations have mild debris drift that is not believed to be significant to results.

All SPH simulations have both forward and drawback inundation longitudinal spread plotted per debris. Debris in SPH tend to go a bit further in forward inundation than in MPM but remain within reasonable expectations of experiments. In drawback inundation, SPH results show a consistent trend of moderate longitudinal displacement towards the basin. These trends, if further developed, may allow for a back-calculation of forward inundation longitudinal displacement if given only the drawback resting position. This would have notable value for post-tsunami reconnaissance when perishable drawback resting positions for debris can be recorded in order to retrieve potentially more critical, but harder to determine, forward inundation positions. This is due to both the forward longitudinal displacement of debris pinning the start point of the hazardous drawback path of debris and because they mark the furthest inland points at risk from tsunami debris.

8.5.3 Lateral Spreading Angle

Lateral spreading angle is an incredibly significant variable for tsunami debris design. Increase in spreading angle has a linear relationship with the potential spreading area (for a fixed longitudinal

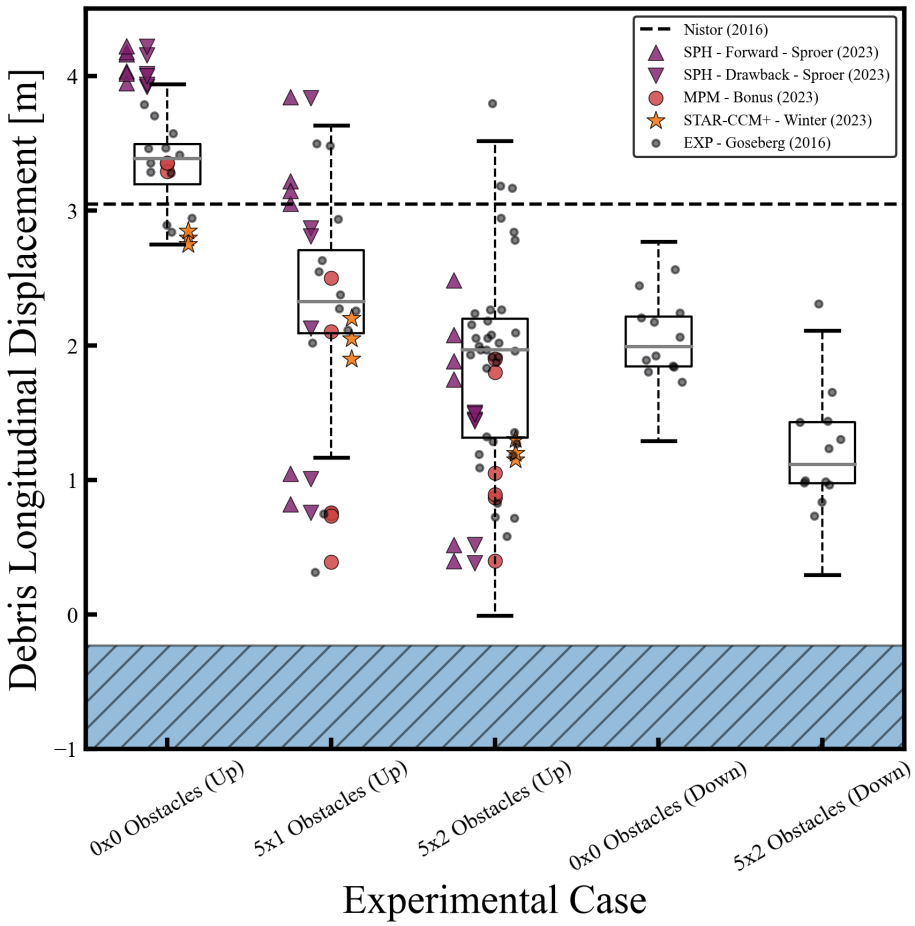

Fig. 8.8. Debris Longitudinal Displacement.

displacement) and thereby causes a proportional increase in the inventory of structures at risk of debris-field impact. However, radiating spread from an initial debris-field centroid, such as for a collection of shipping crates, can reduce the total density of debris along the advancing fronts arc-length (assuming that debris do not form strong aggregates, which may occur). This means the probability of damming and impact loads on an individual building are expected to decrease along some other proportionality relationship to lateral spread. The commonly used conservative guideline from Naito et al. 2014 suggests flat spreading angles of 22.5, which are often applied in both forward and backwards inundations. It is of special interest to quantify the nonlinear effect of

topography, flow characteristics, debris geometry, and infrastructure arrangement (e.g. marginal effect of each added row of structures) on debris-field lateral spreading angle. In flume experiments of Stolle et al. 2016, added rows of obstacles tightened interquartile ranges for spreading angle but increased points exceeding the 95% confidence interval, i.e. outlier events became more likely and the risk of a large group of debris contacting an individual structure within the tight spreading path was increased. This effect is probably due to **(i)** flow through obstacle gaps becoming channelized and thus altering debris linear and angular velocity and **(ii)** increased likelihood of obstacle impact means kinetic energy of debris redirects transversely more frequently. The second contributor is seen, in wave-flumes, to decrease longitudinal displacement which equates to a decrease of inland infrastructure exposed to debris impacts and damming. However, we emphasize that this is dependent on the tsunami wave's time-scale as longer lasting flows may re-mobilize debris that lose momentum from impacts. Although an elongated dam-break wave is applied, the time-scale studied and in nearly all wave-flumes without pumps will have shorter inundation time-scales than a significant tsunami event. This suggests that decrease in longitudinal displacement due to obstacles is probably less apparent at prototype scales but the lateral spreading angle is as apparent or greater for long time-scale inundation.

Predictions from Naito et al. 2014 which apply to conservative design for the spreading of large shipping crate like debris in tsunamis and Nistor et al. 2017a are plotted as interval bounds. We anticipate the former to very rarely be exceeded (it is conservative) while the latter may feature the occasional transgression. It is of special interest to see how obstacles in each numerical method affect the spreading of debris, as impacts and weaving motion through the columns could lead to debris simulations with large lateral spreads or dammed formations between columns if errors are present. As in Nistor et al. 2017a, lateral spreading angle may be defined as the trigonemtric relatinship between longitudinal and lateral displacement, which is then fitted with regression

coefficients for similar Waseda University flume experiments as:

$$\pm\theta_{\text{Spread}} = \arctan\left(\frac{\Delta_{\text{Lat.}}}{\Delta_{\text{Long.}}}\right) = \pm 3.69^{\circ} \pm 0.80^{\circ} N \ , \tag{8.2}$$

which forms a more reasonable bound on lateral spreading compared to the over-conservative guidelines of Naito et al. 2014, although this is very specific to the Waseda flume experiments. Figure 8.9 displays statistical data for debris lateral displacement across experimental cases as a box-and-whisker plot. SPH and MPM appear to capture the general lateral spread of experiments by Stolle et al. 2016, with SPH being a bit better at characterizing the extents of the spread while MPM is seemingly moreso confined to modest spreading values as it never exceeds 12°. STAR-CCM+ results are purely placeholders currently, only plotted as an example of the intended visual style of the figure which is to appear in a future manuscript.

8.5.4 Debris Motion Through a Scaled City

Motion of a few large debris due to a driving inundation wave in a scaled cityscape is a primary focus of our comparative analysis of numerical tools. Distinct phases of motion should be observed with particular emphasis on affects of added rows of obstacles, debris stacking, and debris location on the harbor (upstream/downstream). Phases are interpreted as:

- **Motion begins.** Debris overcome static friction with the floor.
- **Collapsing stacks.** If applicable, debris in vertical stacks may decouple if relative accelerations exceed debris-debris interface friction.
- **Obstacle collision.** Debris rapidly decelerate at impact with obstacles. May bounce transversely
- **Channel flow.** Passing between adjacent obstacles may gain velocity.

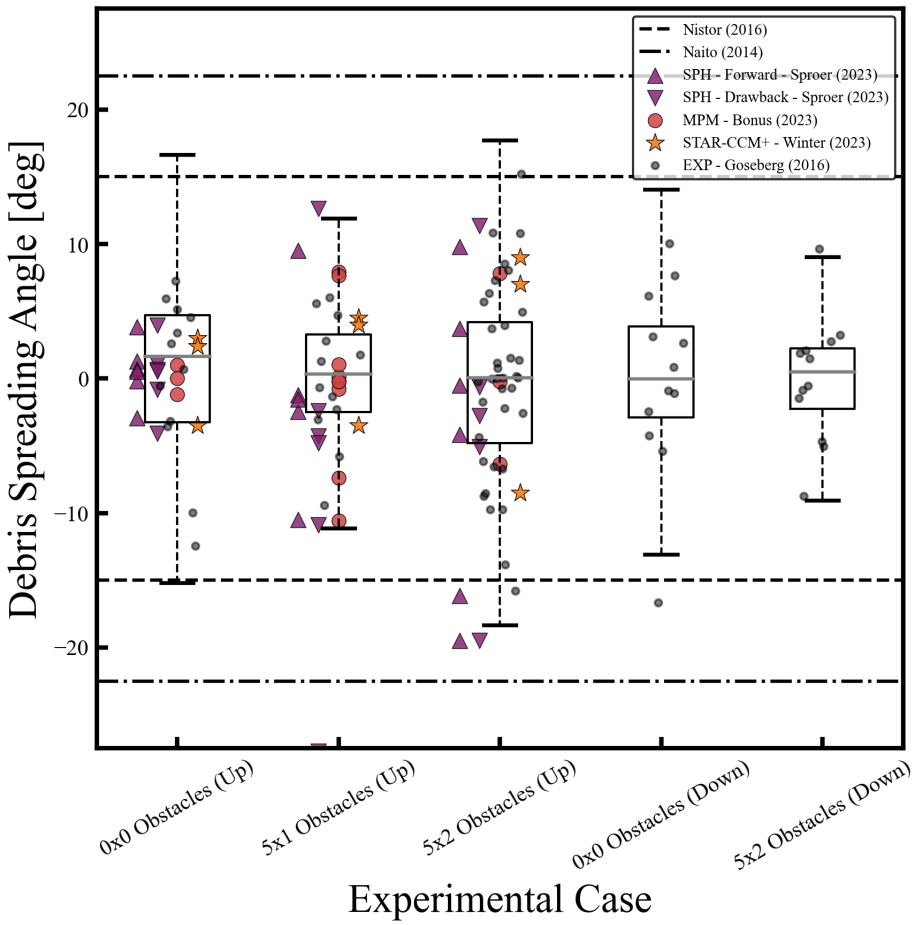

Fig. 8.9. Debris Lateral Spreading Angles.

- **Debris damming on obstacles** and debris-fields jamming between them should be seen in some cases. Hydrodynamic loads should grow with the dammed surface area.

- **Debris spread** across a thinning, slowing inundation wave.

- **Debris come to rest**, or nearly so, at appropriate distances.

- **Inundation begins to drawback**, debris must mobilize in reverse for brief drawback spreading if they are unprotected by obstacles.

- **Debris return to rest.** Wave drawdown becomes too weak for continued debris mobilization against friction

Debris motion has been plotted for SPH data, but not for MPM or STAR-CCM+ data, in Figure 8.10. Strong characterization of both forward motion of debris in the initial inundation, as well as the drawback from the reflecting flow are visible. Damming can be seen on columns, as in the center case, while also observing tandem motion of debris throughout the obstacles according to driving fluid-flow. All debris are mobilized, all debris reach rest or near rest, debris in the path of drawback re-mobilize, and said debris attain rest again. This is, broadly speaking, excellent characterization of the motion phases we aim to see. However, there is a noted tendency for debris to glide further than desired for the no obstacle case on the right of 8.10. This may results from flow-depths that are too great or friction that is too weak.

As the planned manuscript is still in progress, we have not yet replicated these plots for MPM and STAR-CCM+. In MPM's case, we can say that simulations insofar are fairly similar to that of SPH with a few key differences. MPM debris tend to not spread out from the centerline as far, likely due to lack of turbulence increasing pressure drag on obstacles and thereby pulling debris into their wakes. MPM debris generally are transported a shorter distance, however in the case of no obstacles or one row of obstacles in Figures 8.11 and 8.12 this is the preferential results as it better matches experiments. MPM debris dam more easily and for longer on obstacles, especially the central column in the first row. In part this may be because of a slight over-sizing of the obstacles on the 1cm MPM grid (soon to be refined), excessive pressure drag on obstacle geometry's altering adjacent flow (beveling column corners may help), and the high floating point accuracy of ClaymoreUW in this symmetric flume provides little incentive to destabilize the head-on contact between the center debris and center column (explicitly adding 5mm of debris placement asymmetry may solve this).

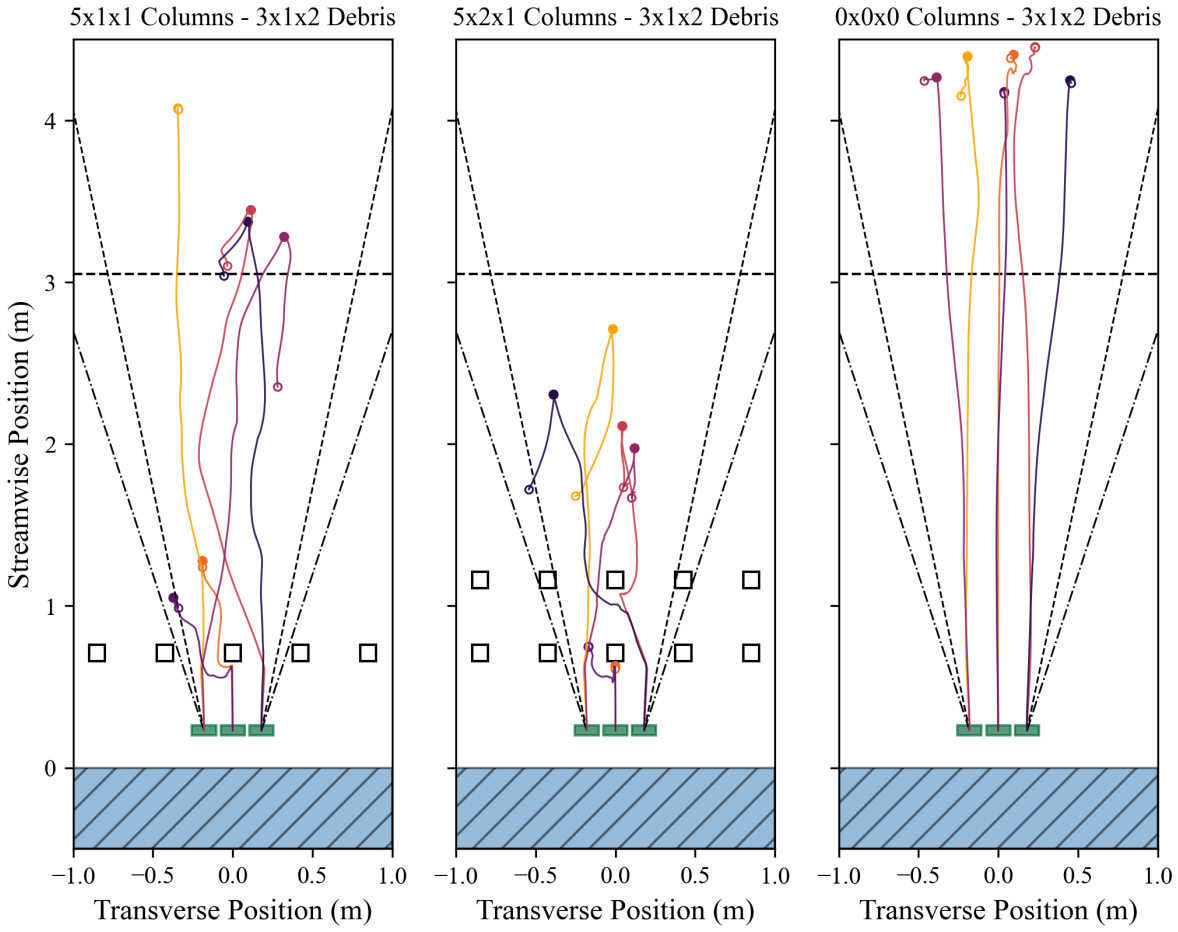

Fig. 8.10. Debris motion in SPH at the Waseda University's tsunami wave basin. SPH replication of experiments by Goseberg et al. 2016, simulated by Felix Sproer, Klemens Krautwald, and Nils Goseberg with visualization by Justin Bonus and Pedro Arduino.

8.5.5 Obstacle Loads

Using a basic drag force calculation on a column, we can calculate an expected hydrodynamic force. This will give a more realistic value than the prediction of a full solitary wave impact on a vertical wall as determined in the work of Cross 1967, which is the absolute hydrodynamic upper-bound due to assuming complete wave reflection.

A collection of curves is plotted in Figure 8.13 to characterize MPM stream-wise load results on 5

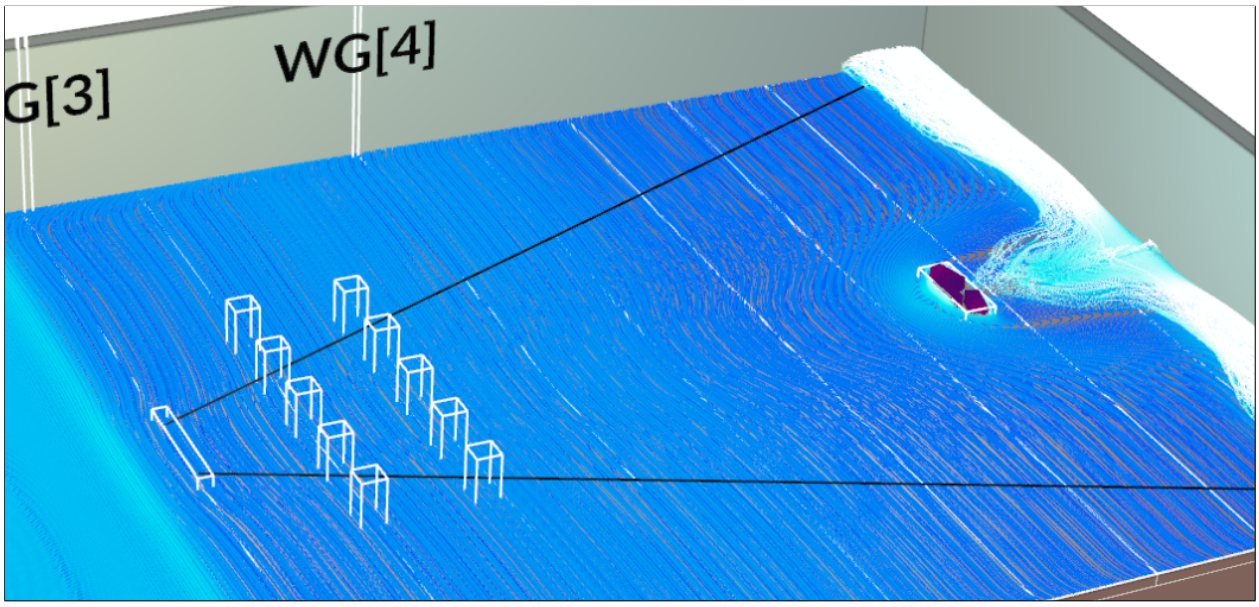

Fig. 8.11. MPM - Simulation of three debris mobilized by a bore-front. No obstacles are present, debris mobilize by unimpeded inundation and come to rest as a result of friction against the harbor floor. MPM debris longitudinal rest location is within 10 centimeters of the experimental median.

select column obstacles. Analytical predictions for the debris load using a basic impact equation (Aghl et al. 2014a), a solitary wave load on a confined wall of unit width 0.1 meters (Stolle et al. 2016), and an estimated column drag force that could result given an assumed wave elevation and flow velocity around the obstacle geometry. Arrival time of the wave surge, not including debris which are yet to fully mobilize, from Goseberg et al. 2016 is also included and seen to closely match MPM results. We include SPH results, which are accelerometer based, to see if there is coincidence with MPM obstacles loads. It appears that the lower relative sampling of the accelerometer method may reduce the peak force, but general timing and trends appear similar. Note that the first force bump in the SPH accelerometer results is the mobilization acceleration initiated on debris, while the first force bump on MPM results is the first contact of the wave's bore-front contacting the obstacles briefly after debris mobilization. Both are roughly 8 N,

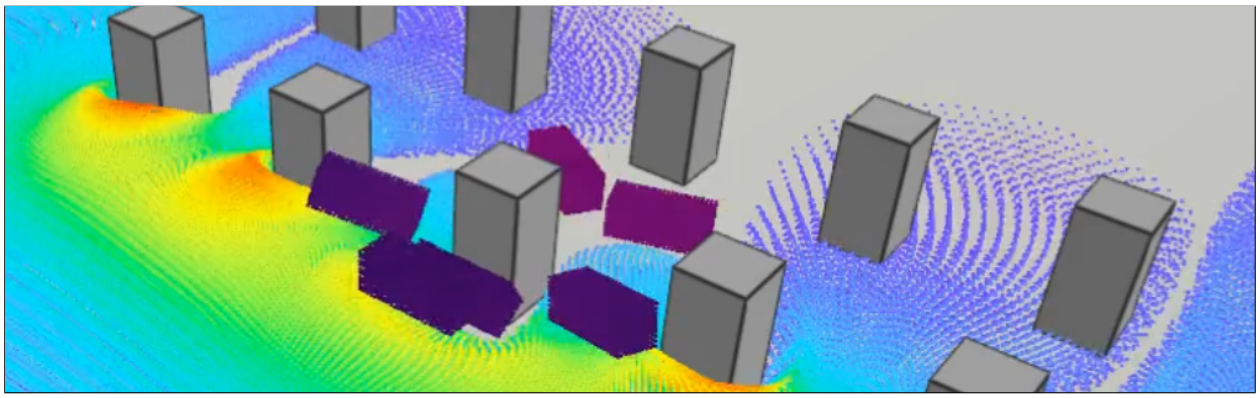

Fig. 8.12. MPM - Simulation of six debris in three stacks mobilized through scaled buildings by wave inundation. Direct collision with the middle column observed. All upper layer debris decoupling from their base when base acceleration exceeds debris-debris friction. Two exterior debris are rapidly pulled past the central obstacle, flowing in to its wake. Experiments and SPH feature less attraction of debris to obstacle wakes, suggesting that MPM's purely laminar flow creates excessive pressure drag that pulls debris inwards. Bore-front closes an estimated 0.5 meters past the obstacles, as observed experimentally in Goseberg et al. 2016. Back row of obstacles are a visual artifact (not simulated).

suggesting that the respective numerical flows produce a similar forcing when first contact a flat obstacle of approximately 10 to 15 cm in width on the early portion of the harbor.

8.6 *Conclusions*

In this chapter a set of wave-flume experiments performed at Waseda University (Goseberg et al. 2016) are replicated with and compared between three advanced numerical methods: Multi-GPU Material Point Method (ClaymoreUW), GPU Smoothed Particle Hydrodynamics (DualSPHysics), and computational fluid dynamics (STAR-CCM+). This endeavor has shed light on the strengths and limitations of these numerical techniques in simulating debris motion in tsunami-like conditions. All three methodologies demonstrated their unique capacities to accurately replicate complex fluid-debris-obstacle interactions in a controlled, yet highly representative, harbor environment. Table 8.1 shows a subjective ranking of each numerical method in a set of topics

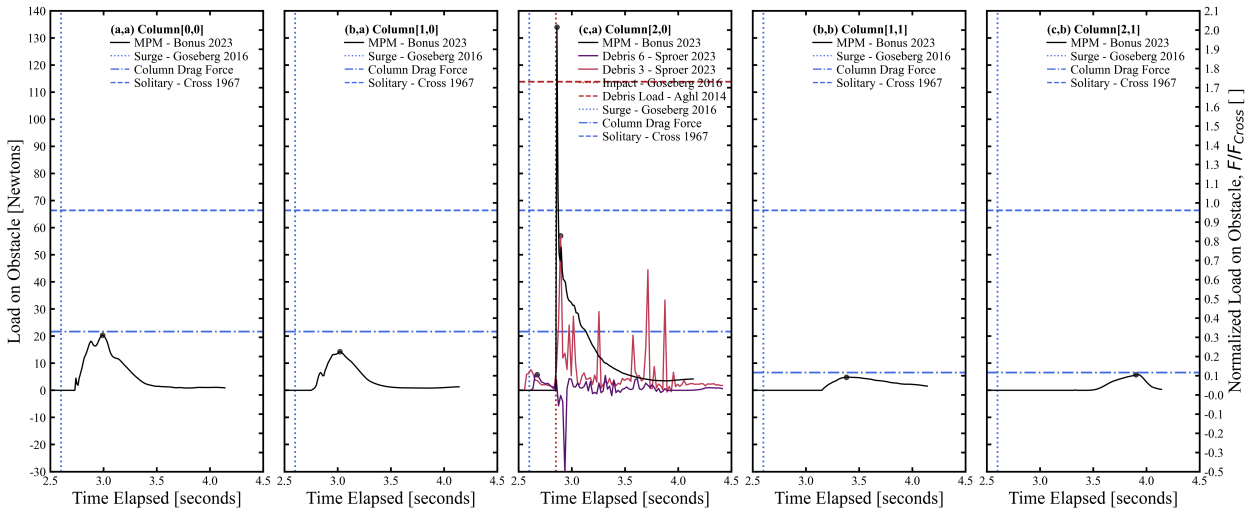

Fig. 8.13. Debris impact force recorded on obstacles.

Table 8.1. Qualitative Comparison of Numerical Methods For Simulating Large Debris in Tsunami-Like Waves

	MPM	SPH	CFD
Hydrodynamics	★★	★★	★★★
Debris Longitudinal Spread	★★★	★★	-
Debris Lateral Spread	★★	★★★	-
Debris Loads	★★	★★★	-
Obstacle Loads	★★★	-	★★★

relevant to tsunami-driven debris hazards.

Our findings suggest that while the Multi-GPU Material Point Method and GPU Smoothed Particle Hydrodynamics perform exceptionally well in capturing the debris longitudinal and lateral spreading, STAR-CCM+ has shown greater precision in estimating wave-gauge elevations respective to the pressure release system employed at Waseda University's tsunami wave basin. It is important to note, however, that the effectiveness of these methodologies can be scenario-dependent, thus emphasizing the importance of employing a combination of these techniques for comprehensive analysis.

While obstacles were found to have significant influence on the maximum longitudinal displacement of debris, the impact on the spreading angle was found to be less pronounced across all three methods. This study not only offers a critical foundation for further research in optimizing numerical models for tsunami debris motion but also paves the way for the development of more robust, efficient, and reliable disaster management strategies in the face of increasing threats from extreme maritime conditions. As we continue refining and improving these numerical methodologies, it is our hope that they will enable us to better protect coastal communities worldwide.

Chapter 9

MULTI-HAZARD SIMULATION OF DEBRIS DRIVEN BY FLOWS OF WATER, SNOW, MUD, AND SOIL GENERATED BY PUMPS, PISTONS, OR GRAVITY USING THE MATERIAL POINT METHOD

9.1 *Introduction*

In addition to the validation studies and applications to debris fields presented in Chapters 6, 7, and 8, this chapter presents three additional comparative studies using our numerical method versus experimental results across different debris field hazard manifestations were explored. Each section in this chapter covers a pilot study investigating the capabilities of our numerical method to generalize across new axis of critical natural hazard behavior. While not developed enough to justify individual chapters, we produce digital twins of three experiment sets at three unique, scaled flume facilities, including:

- Debris field motion experiments by Park et al. 2021 at the Oregon State University's O.H. Hinsdale Wave Research Laboratory Directional Wave Basin (OSU DWB) in Corvalis, Oregon. This includes piston-driven tsunami-like wave tests that mobilize many small debris.

- Debris field impact experiments by Lewis 2023 at the University of Washington's Harris Hydraulic Lab's Washington Air-Sea Interaction Research Facility (UW WASIRF) in Seattle, Washington. This includs pump-driven steady-state flow tests that mobilize a varied number of debris into a raised structure.

- Lahar / debris-flow runout experiments by Iverson et al. 2010 at the US Geological Survey (USGS) Volcanic Hazards Debris-Flow Flume near Blue River, Oregon. This includes gravity-driven debris-flow test (sand-gravel-mud) that mobilize down a steep slope onto a flat, frictional runout plane.

Finally, we discuss the applicability of our Multi-GPU MLS-MPM approach to a broader range of natural hazards, including: tsunamis, floods, storm surges, avalanches, landslides, submarine landslides, lahars, pyroclastic flow, lava flow, and rock-fall. All aforementioned hazards may combine naturally with each other within MPM, and with debris fields, to form a large permutation of multi-hazard scenarios. This will set the stage for the following chapter, which seeks to discern the invariant behavior of debris field hazards separate from the driving mechanism (e.g. tsunami waves, lahar flows).

9.2 *Heterogeneous Density Debris Field Motion in a Tsunami-Like Wave - Oregon State University's Directional Wave Basin*

Heterogeneous debris field motion is one of the many observed outcomes of tsunami events. In this section we demonstrate the efficacy of our MLS-MPM approach in modeling the motion characteristics of wave-driven debris fields respective to varied material parameters (e.g. density, stiffness, friction), specifically for buoyant wood and plastic. These heterogeneous debris fields, arguably, better represent real tsunami debris events, e.g. Tohoku 2011 or Indonesia 2004, where debris-fields were typically not observed as homogeneous. We first describe an experimental set conducted by Park et al. 2021, a high quality work that introduces this variability of initial conditions into a large set of trials while minimizing confounding factors. Then we elucidate the assumptions and limitations of our digital twin modeling approach

The motion of twenty small, buoyant debris in a tsunami-like wave was experimentally studied by Park et al. 2021, conducted at the O. H. Hinsdale Wave Research Laboratory at Oregon State

University in the Directional Wave Basin (OSU DWB), an NSF funded facility. Their work examined how using two different buoyant materials, wood and HDPE, would alter debris-field motion (e.g. lateral and longitudinal spreading) relative to multiple ordered configurations, debris-ground friction, debris density, and obstacle arrangements. Four primary cases were considered, three without obstacles and one including them. The latter uses obstacles as a means to build on prior obstacle-dependent debris-motion works by the likes of Goseberg et al. 2016. Tests primarily used ordered, rectangular debris arrays to characterize debris motion with reproducible initial conditions. Other trials had randomized debris placement, allowing observations on the chaotic motion seen in real tsunami events (Yeh et al. 2013). Full details regarding the experiments motivations and specifications are available in the original study by Park et al. 2021, or located within their references. The most important information regarding the flume and experimental set-up is briefly summarized below to only the extent needed to understand the digital twin we have developed in ClaymoreUW MLS-MPM, our numerical software, for the OSU DWB experiments of Park et al. 2021

Figure 9.1 shows scale and number of particles used for these experiments. The different color debris indicate different materials (wood or HDPE), and trials were run with and without obstacles.

A **wve-flume** schematic for the OSU DWB is shown in Figure 9.2. The flume consists of a rectangular shape with the dimensions of 48.8 m long, 26.5 m wide and 2.1 m high. Beginning at the wavemaker, the bathymetry was comprised of a constant depth section for 0 m < x < 11.29 m at Z=0 m followed by a 1:20 slope for 11.29 m < X < 31.29 m, and ending with a raised flat section through X=41.29 m at elevation Z=1 m above the basin floor (Fig. 2, elevation view). Note that the back-end of the flat, shelf-section ends into a drainage basin at X=41.29 m, so this boundary is best modeled as an outgoing boundary, rather than a vertical wall. In MPM, we modeled the drainage basin itself to conserve total simulation mass. The impermeable bathymetry

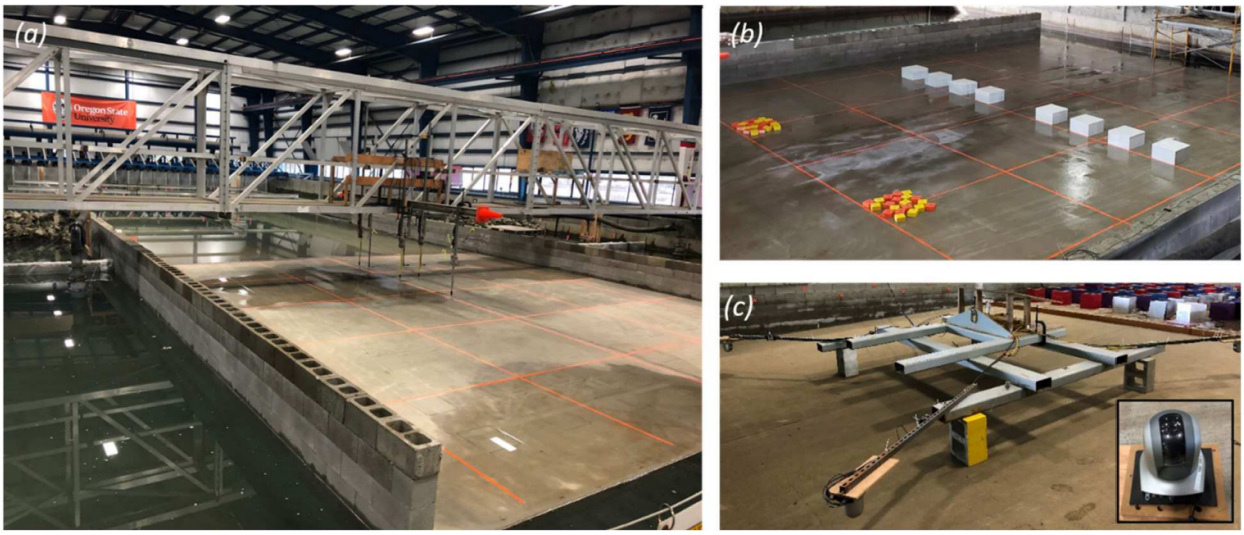

Figure 2: Photographs of the debris test setup. (a) Overview of testbed without debris, (b) Overview of the testbed in other direction with two setups of debris and eight obstacles, (c) Steel frame for the camera mounting and snapshot of the camera (inner photo).

Fig. 9.1. Debris field tests at Oregon State University's Directional Wave Basin. Photographs from Park et al. 2021. (a) Overview of raised testbed, (b) Overview of testbed with debris and obstacles, (c) Mounting frame for optical tracking system.

was constructed of smooth concrete with a float finish, and the roughness height was estimated to be 0.1–0.3 mm. Figure 9.2 also shows relative position of instrumentation, debris-fields and obstacles relative to the coordinate system, which is center transverse to flow at the wave-maker's starting position. Along the flume is a set of wire and ultra-sonic wave-gauges (WG and USWG) to measure the free-surface elevation of the water as hydrodynamics progress.

Debris are rectangular boxes with square footprints (10.2 x 5.1 x 10.2 cm). The two materials tested are high-density polyethylene (HDPE, painted orange-red) and lumber (wood, painted yellow). Both are buoyant in water. Table 9.1 states our digital twin's debris parameters which aim to replicate values from Park et al. 2021 while taking reasonable assumptions on debris material Young's modulus and Poisson ratio's in order to model their elastic response beyond rigid models. Note that both debris material types use a plastic base-plate made of HDPE. This is to

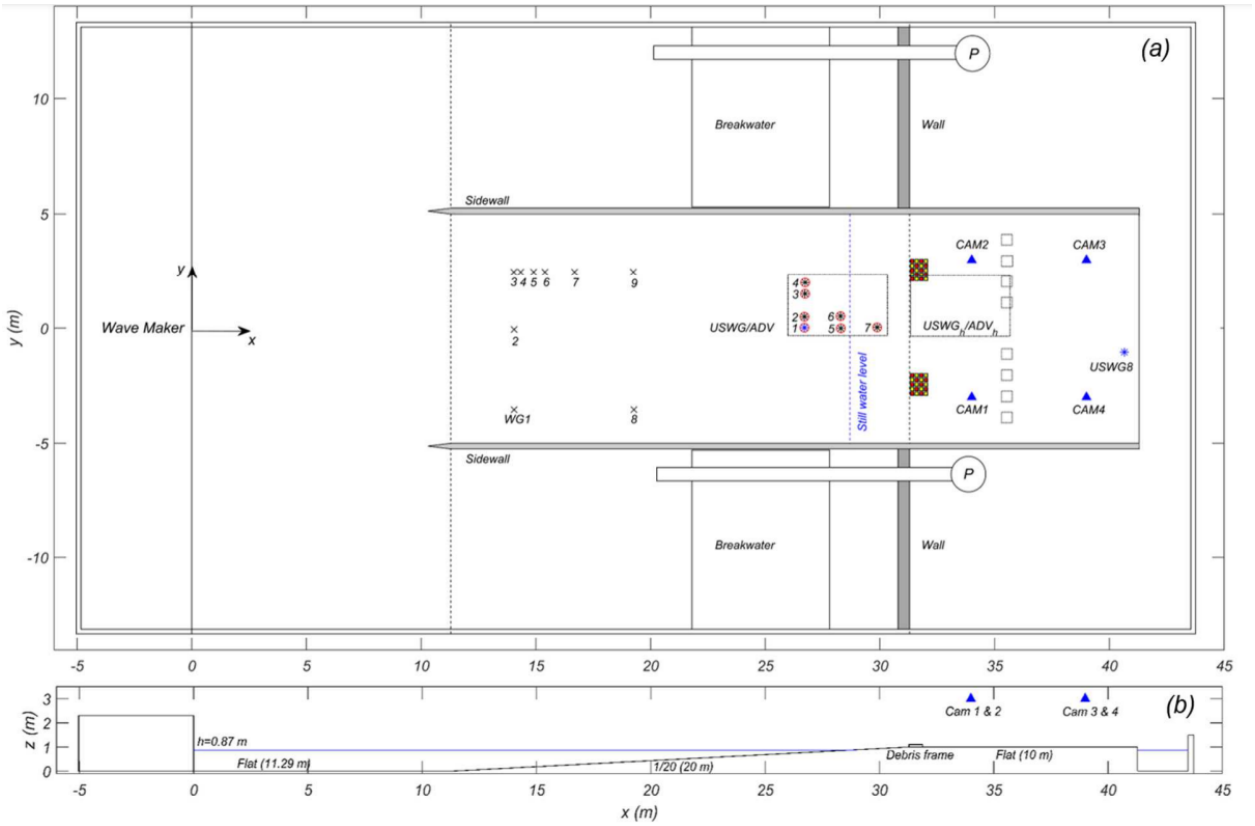

Fig. 9.2. Plan-view schematic of the Oregon State University's Directional Wave Basin. Schematic of the OSU DWB during debris field motion tests. Adapted from Park et al. 2021.

reduce frictional affects unrelated to mass-density (e.g. wood grain orientation, as shown in Figure 9.3). The debris field location was chosen at the leading edge of the flat test section (x = 31.29 m). We kept the footprint of the initial debris field constant (constant area) as in experiments which utilized a 71.4 cm width (flow perpendicular) by 56.1 cm length (flow parallel) frame to place a matrix of 5 by 4 debris elements. Gap spacing in experiments is S_{gap} = 5.1 cm, however for our numerical model we increased the gap to S_{gap} = 10 cm to improve our low resolution pilot simulations. This helped resolve debris-debris interaction with relative accuracy, future work may improve on this. As the increased gap is smaller than debris widths, so it is not expected to dominate results, however, it does constitute a less dense starting configuration which

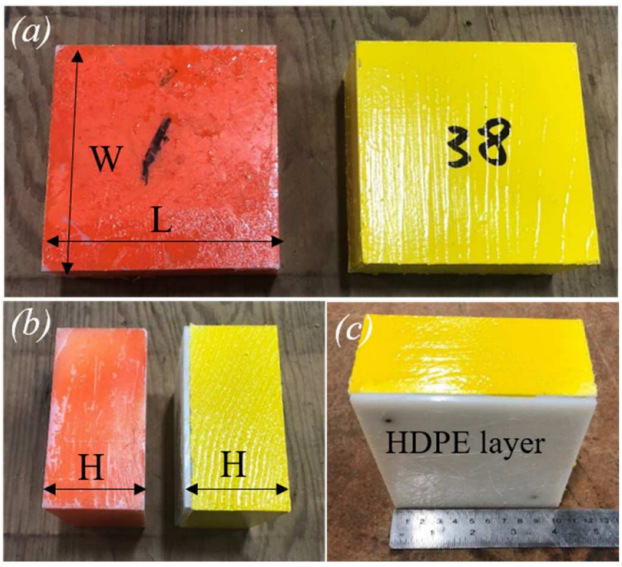

Fig. 9.3. Rectangular debris made of wood and plastic used in OSU DWB experiments. Photographs from Park et al. 2021. Note that both debris have HDPE plastic base-plates to standardize some aspects of friction response to inundation of debris. Debris dimensions are 10.2 x 5.1 x 10.2 cm.

is theoretically easier for a tsunami-like bore to spread across a flat plane.

Obstacles: Downstream of debris on the flat platform are obstacles, if applicable to the trial, seen as gray boxes in Figure 9.3 (b). Each is 0.4 x 0.3 x 0.4 meters of concrete, making them effectively rigid relative to the debris and wave impact loads they resist. At 1:50 geometric scale the prototype represents small to mid-sized commercial structures (20 x 15 x 20 meters, 400 m^2 footprint, 2 - 3 stories). Obstacles in the flume model are 4 meters (200 m prototype) away from the debris field, i.e. a parking lot. The spacing between obstacles is 0.4 m (20 m prototype) which gives a blockage ratio of 0.5 amidst obstacles. In our MPM digital twin, obstacles were represented as box boundaries, located according to Park et al. 2021, with separable velocity conditions. Edge grid-nodes on boxes were beveled. No friction is applied on obstacles.

Incident Wave: As stated in Park et al. 2021, the generated wave is not a solitary wave but rather a solitary-like wave that is able to make better use of the piston wavemaker to get a longer period,

Table 9.1. OSU DWB debris in the prototype and model. Based on experiments by Park et al. 2021.

Parameter	Prototype	Model	Prototype	Model
Material	HDPE	HPDE	Wood	Wood
Geometric Scale	1:1	1:50	1:1	1:50
Debris Length (m)	5.1	0.102	5.1	0.102
Debris Width (m)	5.1	0.102	5.1	0.102
Debris Height (m)	2.6	0.051	2.6	0.051
Debris Gaps (m)	5.1	0.102	5.1	0.102
Density (kg/m^3)	987±1.7	987±1.7	648±17.6	648±17.6
Young's Modulus (Pa)	1e9	1e9	2e9	2e9
Poisson's Ratio (s)	0.4	0.4	0.2	0.2
Static Friction (Dry)	0.66±0.07	0.66±0.07	1.28±0.13	1.28±0.13
Dynamic Friction (Dry)	0.66±0.07	0.66±0.07	1.28±0.13	1.28±0.13
Static Friction (Wet)	0.38	0.38	0.71	0.71
Dynamic Friction (Wet)	0.38	0.38	0.71	0.71

an important feature for tsunami time-scales as shown by Madsen et al. 2008. while generating a long period wave. Due to this generation approach, the wave is not permanent, like a solitary wave. Numerically, the wave can be generated using two different methods: **(i)** The wavemaker displacement time series can be used if a moving wall boundary condition is available in the numerical model. **(ii)** The time series of incident wave elevation along the wavemaker paddle can be used to force the numerical model at X=0 m. We choose the former approach, using a moving velocity boundary condition on the MPM grid. Water depth at the wavemaker is 0.87 meters (SWL = 0.87 m).

Instrumentation: Free-surface water elevation comparisons at WG1, WG9, USWG5, and USWG5-h were used to evaluate if the underlying hydrodynamics is reasonably simulated, as accurate flow conditions is a minimum requirement for accurate debris mobilization response, as mutually agreed in another debris modeling workshop by (Takabatake et al. 2021). Error metrics could be evaluated, though for now we take visual appraisal as adequate to check for first-mode

wave behavior. Figure 9.4 compares experimental against our numerical results, where we see that error is bounded with 0.5 to 1.5 grid-cells for the first peaks of WG1, WG9, USWG5, and USWG-h5. Our frequency content appears to be somewhat damped, not surprising as we did not investigate use of more energetic MPM advection for this pilot study. Also note that we are only concerned with capturing the initial wave in these plots due to the very long wave generated, and the fact that breakwaters in the side channels reflect wave energy which arrives at the wave gauges about 10 seconds after the crest of the leading wave and thus may amplify the subsequent peak. In other words, the entire offshore section of the basin is sloshing horizontally and constructively amplifying later peaks in our measurements. These reflected waves do not play a significant role in the debris motion, as noted by the author in Park et al. 2021, so we chose to neglect modeling the $\tilde{6}0\%$ water volume in the flume outside the center channel for expedience. Of course, MPM is not ideal for modeling fluids, especially in such simple conditions where reduced complexity models will perform excellent. For instance, a Boussinesq-predicted wave could have been taken as an inflow boundary condition, but we chose not to do this as proving MPM's viability in simulating a wave from the wave-maker to the end of the experimental duration is of significance. The following experimental cases from Park et al. 2021 were considered in our MLS-MPM simulations:

- Case 1: Array of HDPE Boxes, 5 debris wide by 4 debris long is placed. The boxes are spaced 5.1 cm apart in both the x and y directions experimentally, although our coarse pilot simulations apply 10 cm distances. The offshore edge of the boxes is at X=31.29 m.

- Case 2: Array of Wood Boxes. Identical initial configuration as Case 1.

- Case 3: Random array of 10 HDPE and 10 Wood boxes. The boxes are placed randomly in space and individual rotation, as shown in the flume in Figure 9.1 (c) and our numerical random debris fields in Figure 9.6. Numerous experimental trials for this case were run, all

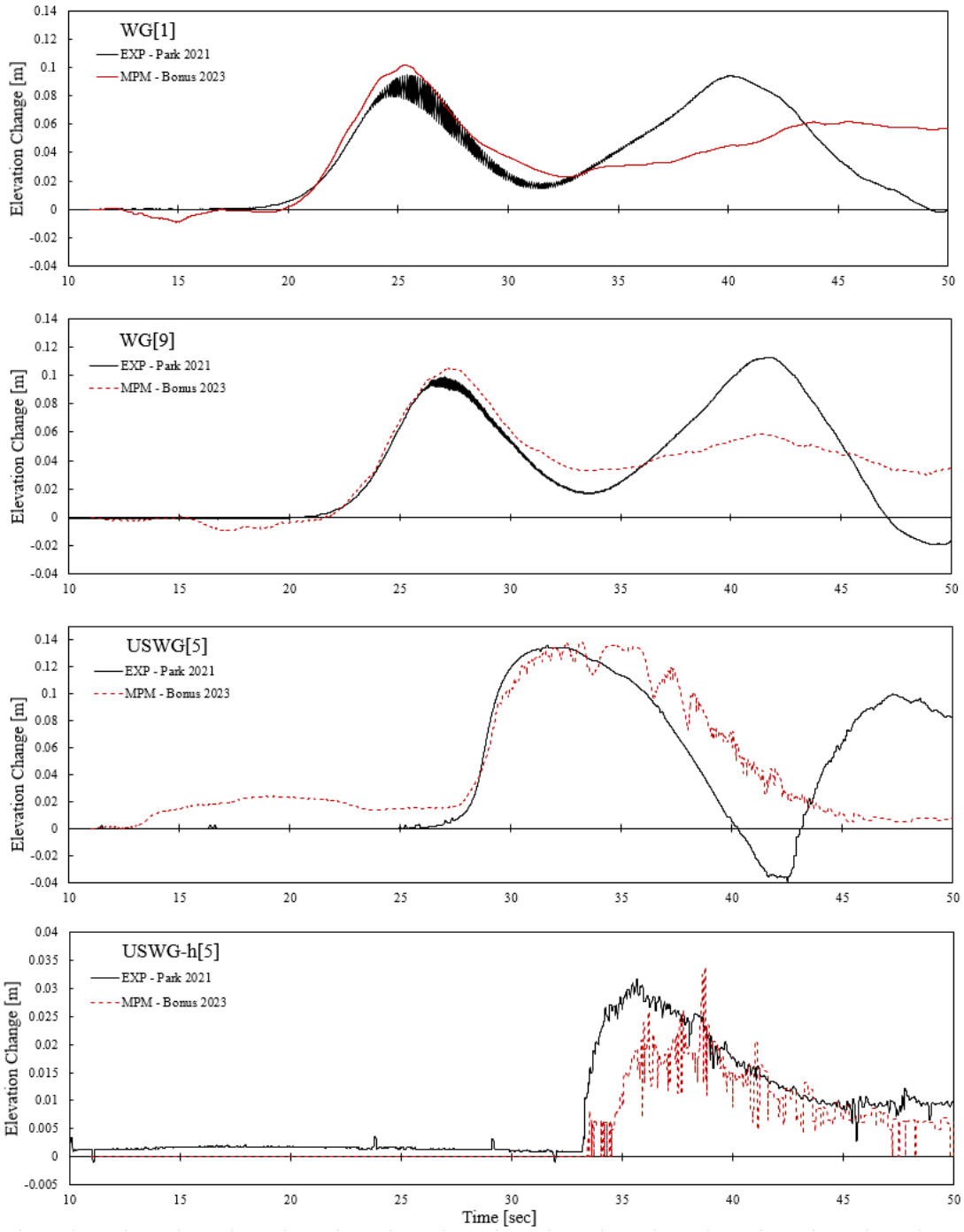

Fig. 9.4. Free-Surface elevation comparison in the Oregon State University's Directional Wave Basin. Compares wave-gauge measurements from experiments by Park et al. 2021 and MLS-MPM simulations by this study (Bonus 2023).

with different initial placements of the random array. The debris boxes are initially placed inside of a wood frame (71.4 cm by 56.1 cm) without an overlap among debris.

- Case 4: Same initial debris configuration as Case 3. Park et al. 2021 provides the position of the two obstacles (white boxes). The size of an obstacle (W by L by H) is 0.4 m by 0.4 m by 0.3 m. Obstacles are made of concrete and firmly fixed on the ground. In simulations we take them to be rigid boundaries. The space between two obstacles is 0.4 m. The exact coordinate of the left bottom edge of the left obstacle is x = 35.29 m and y = -0.6 m.

The non-obstacle cases (Cases 1, 2, and 3) for this benchmark are ideally suited to relatively simple debris transport models, perhaps even well-captured with a calibrated one-way coupled model. Modelers were asked to simulate four different cases in Park et al. 2021; one with obstacles (Case 4) and three without (Cases 1, 2, and 3). Modelers provided comparisons with the mean distance traveled, lateral spreading, and longitudinal spreading for the cases. Modelers were encouraged to discuss various parameter changes used during the comparisons, such as grid size, time step, friction, breaking models, material properties, and any initiation and stopping thresholds for debris motion.

We used the random debris field generating algorithm described in Chapter 7.6 for the OSU LWF randomized impact tests. This gave us three random fields with locations/orientations, wind-rose plots of orientation, and a cumulative distribution and histograms of debris orientations. These are all shown in Figure 9.6 and were used to run random debris field simulations in ClaymoreUW MLS-MPM.

The tsunami-like wave was seen to disturb the initially ordered debris field fabrics of Cases 1, 2, and 3 in a predictable way. Notably, it compressed the field elements nearer to one another, began to widen the back of the field into a trapezoidal shape, pivots the external debris forward about the fields front point, and eventually scatters the debris slightly laterally and notably forward with

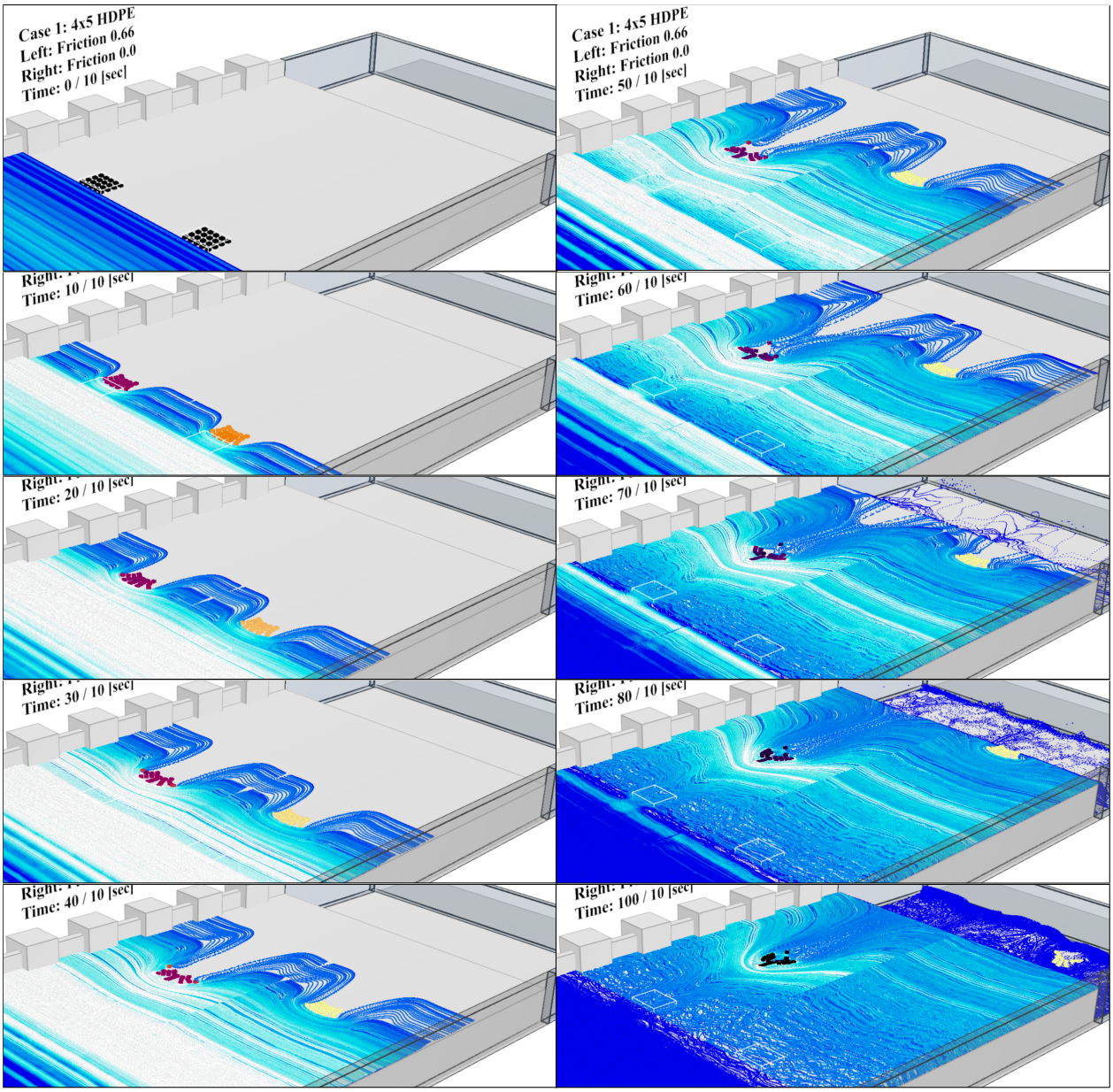

Fig. 9.5. Motion of dry debris fields with and without friction in a wave via MLS-MPM. Twenty plastic HDPE debris are simulated in MLS-MPM using our Digital Twin of OSU's Directional Wave Basin. Experiments by Park et al. 2021. Two debris fields rest on concrete with friction coefficients of 0.66 and 0. Velocity and elevation shown on debris and water.

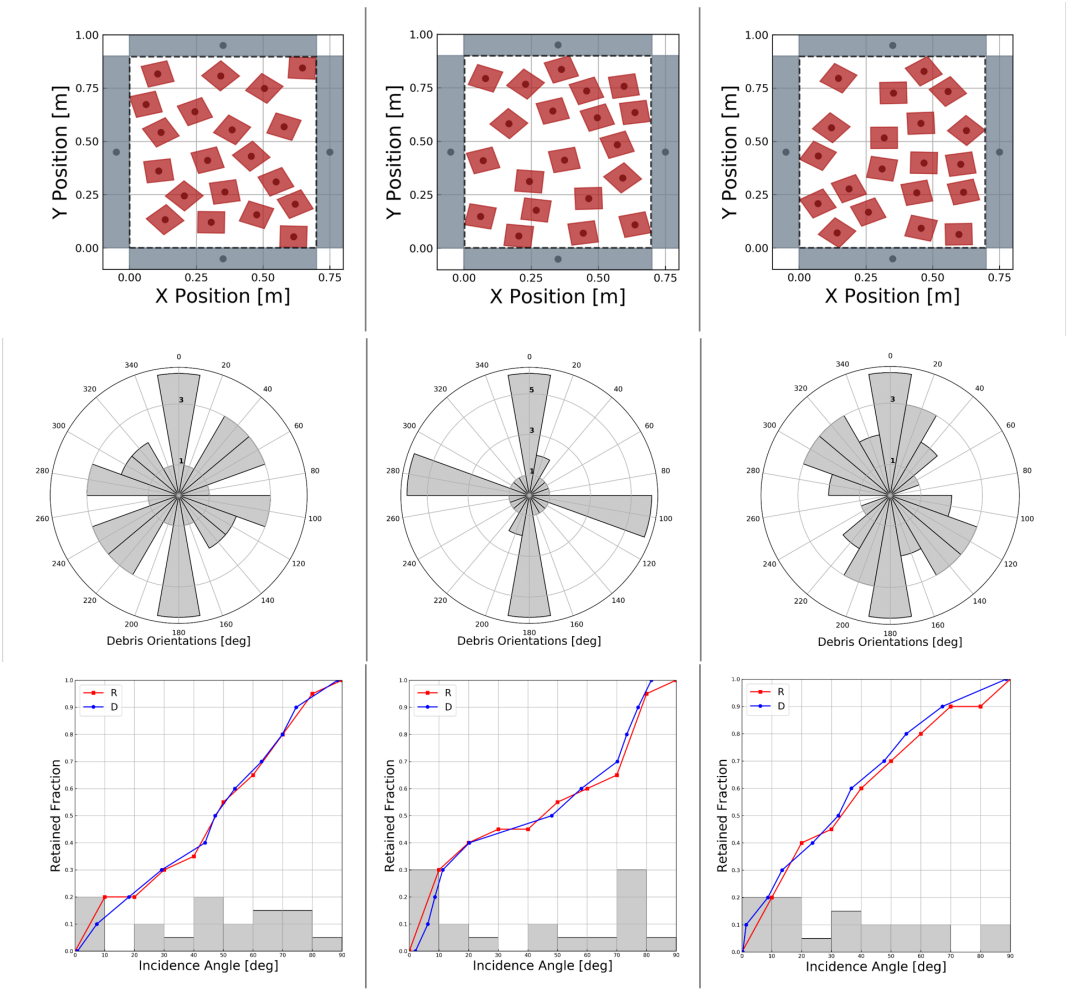

Fig. 9.6. Three random debris fields as diagrams, rose plots, and histograms. Used for OSU DWB simulations in MLS-MPM. Based on experiments of Park et al. 2021.

some random variations in motion. This is shown in Figure 9.7 in both experiments and our simulations. We note that our simulations were more damped thus taking longer to reach equivalent stages of fabric change, that the rotation overshoots since the debris are "sticky" and did not separate and thus conserved too much angular momentum for the group instead of individual derbris, as well as bringing attention to the very low computational resolution enacted in this MPM case. Grid-cells of 2.5 cm are simply inadequate for modeling such tightly packed debris-debris interaction, especially with a thin but stiff fluid attempting to infiltrate between the points of debris contact as it lifts pieces off the frictional floor. This limits the quality of the debris-debris and debris-fluid interactions, and if redone a cell size of 1.5 to 0.5 cm would fare significantly better. This is achievable at the high-end on modern Multi-GPUs without much difficulty, and the low-end if we were to truncate out the $\tilde{8}0\%$ of water volume in the far-field that does not hold significance after the first wave (could just be an inlet boundary condition of a Boussinesq solution to save GPU memory and time). However, even at low resolutions of 2.5 cm grid-cells, the overall replication of the debris-fields rotational response to an inundating wave is a very promising sign that the dynamics of the system are being resolved in MLS-MPM. To the best of our memory's ability, we do not recall other participants of the tsunami modeling workshop showing results that so clearly modeled these rotational dynamics, with the faces of the square debris and the gaps between them showing nuanced responses to the thin fluid flow as it builds on them to overcome friction with the floor. There were, however, two groups with notably better predictions of debris lateral spreading and the transport of different density debris past one another as the fabric reorganizes, but those methods imposed a force balance specific to fluid debris problems (i.e. calculates drag, buoyancy, added mass effects on point debris) whereas MPM solves general dynamic multi-material, multi-phase, large-deformation 3D problems.

We are surprised at how well our MPM approach aws able to perform on short-notice for this nuanced study of debris motion respective to debris material, debris geometry, debris-field

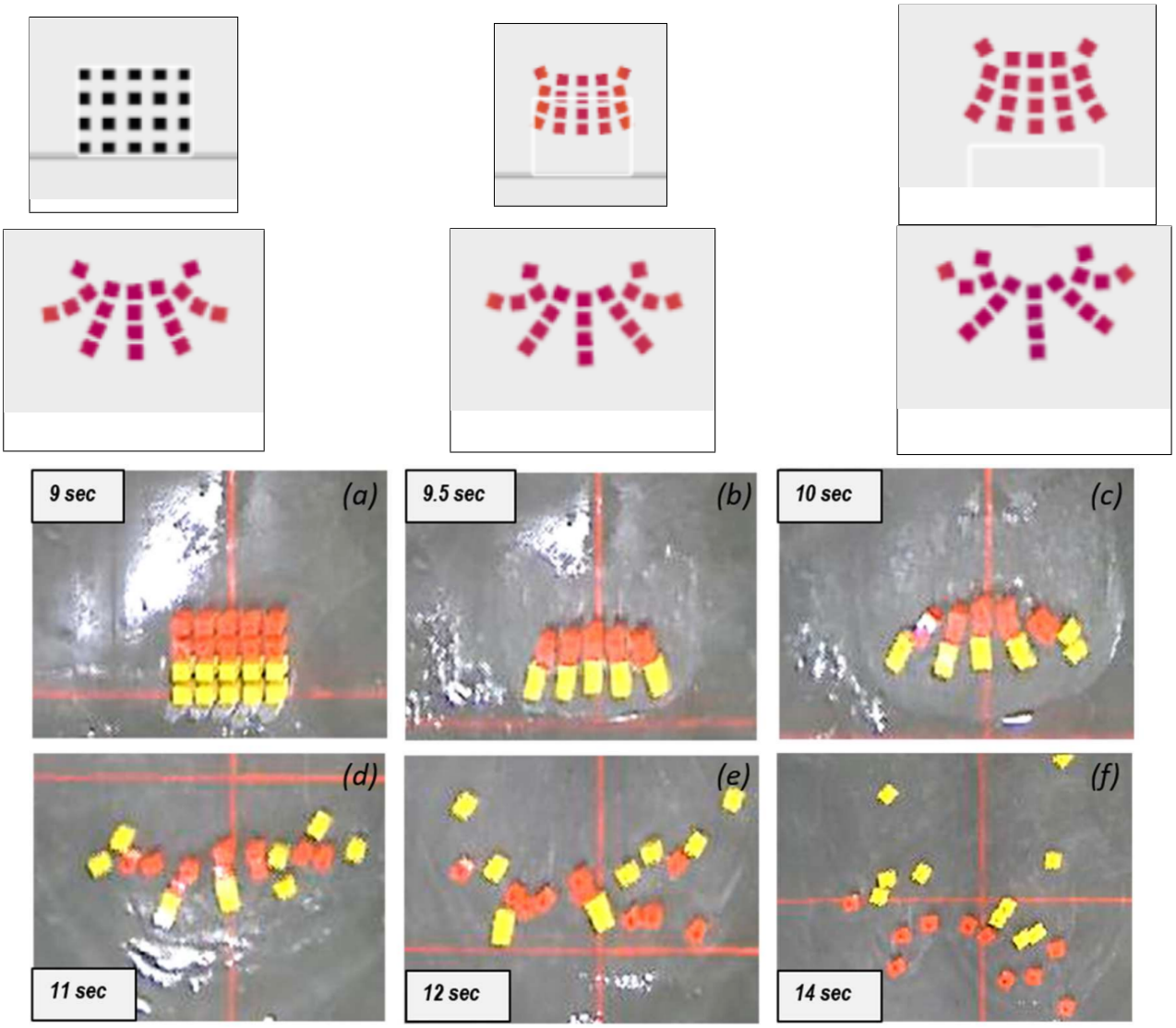

Fig. 9.7. Rotation of debris fields in a wave in OSU DWB experiments and simulations. Debris field external columns rotate outward characteristically as they are dragged by the fluid against a frictional floor. Experimental photographs from Park et al. 2021.

geometry, mobilizing flow, and friction. To summarize our simulations, we list the conclusions that Park et al. 2021 provided in their experimental paper with addendum's regarding our comparative experience with a digital twin in MPM.

1. **Experiments:** The less-dense debris group (SG = 0.65, wood) moved further and had less spread compared to a more dense debris group (SG = 0.99, HDPE). **Simulations:** Observed the greater forward motion with lower density, but neither material spread very well so it is difficult to draw a conclusion on debris material spreading in MPM.

2. **Experiments:** The mean longitudinal displacement of less dense debris decreased linearly as the amount of more dense debris increased in the debris group due to the interrupting influence by the more dense debris during the advection. However, the mean longitudinal displacement of the more dense debris was uniform. In summary, the more dense debris affected the mean longitudinal displacement of the less dense debris, but the converse was not true. **Simulations:**

3. **Experiments:** The spreading angle of less dense debris increase slightly (+3°) as the number of higher density elements were added. The spreading angle of the more dense debris decreased (-9.7°) as the less dense elements were added. **Simulations:**

4. **Experiments:** For groups with mixed debris, the initial configuration (e.g., wood debris on the seaward side or landward side of the HDPE debris; uniform, checker, random starting patterns) had little effect on the mean longitudinal displacement or spreading angle. **Simulations:**

5. **Experiments:** The cases with less dense debris (wood) only had a 30% higher probability of collision with the obstacles compared to the cases with the more dense (HDPE) debris

only. When the debris types were mixed, the less dense debris had a lower probability of collision with the obstacles. **Simulations:**

6. **Experiments:** Overall, the reflected wave and interaction among different debris play a role in the probability of collision. However, the density of each debris element was a dominant factor in determining the collision probability. **Simulations:**

7. **Experiments:** The leading-edge flow velocity is spatially uniform (1.4 m/s) and greater than the less dense debris (1 m/s) or more dense debris (0.7 m/s). The flow velocity of both debris types varied spatially and was sensitive to the flow depth, a draft of debris, and the existence of obstacles. In general, this paper highlights the importance of considering debris density in estimating the longitudinal distance and spreading angle. These variables were less dependent on the initial configuration of the debris field. Future studies should consider other aspects of the phenomena, including a better understanding of the potential impact by debris on obstacles, the role of the return flow in determining the debris trajectory, and investigations of the obstacles that more realistically reflect urban shorelines subjected to strong overland flow. **Simulations:**

9.3 *Debris Fields in Steady-State Flow Driven into a Raised Structure - UW WASIRF Wave-Flume*

The second study involves early efforts in designing a digital twin of a steady-state flowing wind-wave flume for debris impact studies.

The University of Washington's Washington Air-Sea Interaction Research Facility (UW WASIRF) flume is recreated herein, namely in its configuration during the debris impact tests by Dr. Nicolette Lewis in 2021 and 2023. The digital twin exists currently within our Material Point Method approach (i.e. ClaymoreUW, see Chapters 4 5), and may later be added to the NHERI SimCenter's open-source HydroUQ software. Some expository text and figures have been

provided by: Dr. Nicolette Lewis, Prof. Mike Motley, Prof. Pedro Arduino, Prof. Greg Miller, and Prof. Marc Eberhard. All numerical simulations presented originate from this dissertation.

The WASIRF debris impact studies, led by Nicolette Lewis with assistance from Abbey Serrone and multiple undergraduate students, are a 1:5 to 1:4 geometric scale of the OSU LWF debris load tests by Shekhar et al. 2020 and Mascarenas 2022 (i.e. 1:50 to 1:40 geometric scale of the prototype tsunami). Herein we only consider the latter scaling of 1:4, as our preliminary numerical down-scaling in MLS-MPM for the OSU LWF in Chapter 7, which explored hydrodynamics and a handful of debris field load tests, involved a set of scales which included 1:4 (i.e. 1:40 the prototype tsunami). This allows scaled results from the OSU LWF to potentially be applicable in bridging findings from the UW WASIRF and OSU LWF experiments, a step towards real-world tsunami debris-field impact models without bias towards experimental facility scales, though that is outside the current scope of this dissertation.

In a nutshell, the UW WASIRF flume experiments were intended as a test-bed for proving ourselves wrong. Its entire geometrically down-scaled set-up of the OSU LWF aimed to provide a scale invariant platform to interrogate any engineering conclusions, probabilistic frameworks, and numerical modeling approaches (e.g. MPM) that had been developed using the OSU LWF tests. Blind numerical-experimental comparisons involve simulation of an *untested* case, followed by an experimental recreation, and finally analysis of both results to assess accuracy of numerical loads given initial flow and debris-field conditions that were not complicit in the training and/or tuning of an assessed model. This is to prevent over-biasing of a modeling approach towards a set of experiments, as that would be dangerous to use in real-world design. If our aforementioned findings hold when scaled-down 1:4 from the OSU LWF model to the UW WASIRF model, it would lend credibility to them when scaled-up 1:10, i.e. from the OSU LWF model to the prototype tsunami event.

There is, at times, some hesitation to using results from numerical models in the development of

design codes and loading guidelines. It is imperative that researchers are able to show not only that their approach is well-validated against existing experimental data, but also that models can be robust enough to use across the entire parameter range of the physical problem. To that end, the final task of this tsunami debris-field load project was to apply the high-performance numerical modeling approach, which we hoped to have developed and validated over years of research and multiple doctoral students, is to model an untested facility's alternative geometric scale in debris tests to develop a series of experiments to be used for a blind prediction of the results and thus validation for general tsunami debris load capabilities.

It is only in the prior three to six months, and in-part due to advances in modern Multi-GPUs (see Chapter 3) and novel MPM literature, that we have reached a point of confidence in our numerical approach, which is now validated on four relevant tsunami debris experiments (Shekhar et al. 2020; Mascarenas 2022; Park et al. 2021; Stolle et al. 2016), this section being a potential fifth with future effort, and a plethora of debris-fluid-structure interaction benchmark cases in Chapter 6. Because of this, we have high-hopes in our schemes recreation of the chaotic behavior of debris loads in blind comparative trials. Ironically, we reach this point as our project comes to its scheduled conclusion. Because of this, the planned blind experiments had to be performed in the Spring/Summer of 2023 and are no longer blind as consequence.

However, we put reasonable effort herein in the development of a digital twin for the UW WASIRF experiments. Though not exhaustively quantitative or blind, we show examples of debris-fluid-structure interaction in pilot simulations that resemble both the WASIRF experiments and the OSU LWF experiments they are meant to resemble in a scaled-down capacity. It is found that our approach is an effective means to model the WASIRF flume's debris test's primary aspects, even in comparatively low resolution simulation. This also shows ClaymoreUW's ability to model near steady-state pump driven flow, with added linear pistons for introducing waves if desired. The WASIRF wind-makers are not currently modeled, but wind-wave interaction is a

growing topic of research and may be pursued in an update to the digital twin for HydroUQ. We will now summarize key aspects of the flume experiment's digital twin in MPM, noting that further detail is to be provided by Nicolette Lewis' documentation.

The **UW WASIRF wave-flume**, see Table 9.4, is small to medium scale (18 x 1.2 x 0.9 m, max flow-depth of 0.75 m, and max flow-speed of 0.6 m/s in our configuration). It is suited for geometric scales between 1:20 to 1:50 when studying tsunamis, corresponding to time-scales of 1:4.5 to 1:7.1 for Froude similitude. These time-scales are inadequate for tsunamis, but note that UW WASIRF is a closed-loop with pump-driven, stochastic steady-state water flow, and it also supports adjustable wind-flow. The latter was not used (though may be in the future for HydroUQ), but the former allows us to generalize conclusions drawn from non steady-state forms including: solitary waves (OSU LWF, Chapter 7), solitary-like waves (OSU DWB, Section 9.2), and dam-break waves (Waseda TWB, Chapter 8) from respective flume experiments. This provides useful validation for modeling tsunamis, which are often characterized by dam-break waves (though flumes struggle with very large dam-break time-scales) and only partially represented by solitary and solitary-like waves (as time and length scales can be off by orders of magnitude, Madsen et al. 2008). UW WASIRF's provides similar capabilities to the pumped-flow time-scales of Goseberg et al. 2013. Ideal prototype scales range from, loosely, half a minute to fifteen minutes, and in some cases, up to hours. UW WASIRF can, in theory, support steady-state flow over these time-scales (i.e. divided by the square-root of 40), but we note that one must continually add debris to the flume to stretch debris impacts to said period as the flume has limited length relative to stable flow-speed, and also that the inlet and outlet of the flume, where the pumps connect, absorb or add in pressure and surface waves that may or may not be important to some applications. Broad simulation settings applied in replication of the model and eventual prototype digital twin of the UW WASIRF flume are tabulated in Table 9.5

Across hundreds of **inundating flow** tests, near steady-state flow is reached for many pairings of

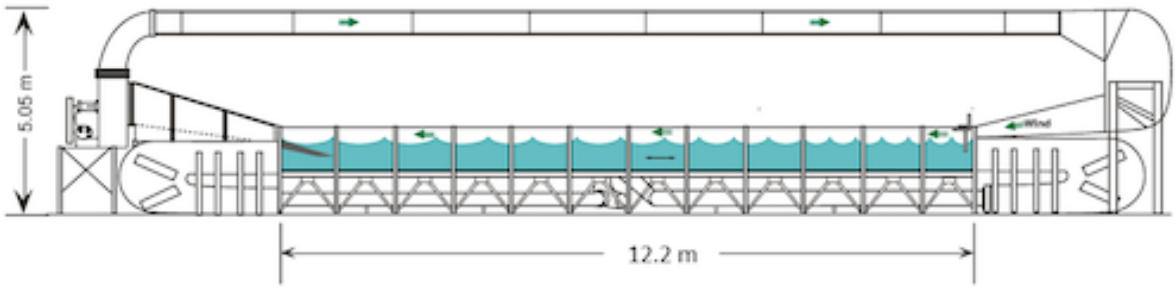

Fig. 9.8. Diagram of UW WASIRF Wave Flume. Side-view diagram of wave flume used to perform small-scale fluid-driven debris-field impact experiments. Facility located at the University of Washington.

flow-velocity and flow-depth, useful for studying probabilistic tsunami inundation behavior on near-shore structures. Our brief simulations efforts only consider a few, but future work may characterize all of them and then extrapolate to configurations not attainable within the facility's physical limits. Note that the first round of testing in UW WASIRF had inconsistencies in flow depth and speed related to a nonlinear pump response for thin flows. Efforts were made to control the depths and speed of flow, making them more deterministic, in the second round of tests via hydraulic jumps. Further information regarding depth determination along flume's debris field fetch length, near the box, upstream at the hydraulic jump and pump inlet, and downstream by the debris catch-net and pump outlet is beyond our current scope. In our MPM simulations we simply numerically extended the flume to hold the length of water required for the time-scale (e.g. 30 meters of water for a 50 second case at 0.6 m/s flow). We imposed velocity boundary conditions before nearing the debris-field placement range, i.e. we set streamwise velocity to 0.6 for all water 2 meters upstream of debris, and 3 meters downstream of the structure. Initial flow depth was set as an initial condition on the MPM particle geometry. We do no replicate the hydraulic jump explicitly, rather match the flow and depth it was observed to produce in our simulations for the entire length of the digital twin.

Table 9.2. UW WASIRF raised structure in the prototype and model. Based on experiments by Lewis 2023.

Parameter	Prototype	Model
Geometric Scale	1:1	1:40
Structure Length (m)	10.16	0.254
Structure Height (m)	6.15	0.152
Structure Width (m)	10.16	0.254
Dist. from Wave-Origin (m)	438.0	43.80
Elevation above Seabed (m)	20.0	2.00
Elevation above Ground (m)	2.50	0.25
Elevation above SWL (m)	0.8 - 4.4	0.02 - 0.12
Column Diameters (m)	0.4	0.01
Column Lengths (m)	4.0	0.1
Column Array (m)	5 x 4	5 x 4
Column Spacing (m)	4.0	0.1

In all our tests at UW WASIRF, an aluminum, non-rigid (but stiffened) **structural box** was mounted in a raised position in the flume to represent a scaled prototype raised building, e.g. a prototype residential or commercial structure, or potentially a bridge component. It is 1:4 the geometric scale of the "orange box" used for the OSU LWF debris tests in Mascarenas 2022 and Shekhar et al. 2020, so naive Froude similitude anticipates a 1:64 difference in force measurements. To ascertain this, it was equipped with force sensors, i.e. load-cells, operating at 1200 Hz to capture stiff debris impacts. Note that calibration did not remove all noise from the signal, post-processing outside the scope of this write-up was applied to discern debris signals from facility interference. The box is modeled as a separable, fully rigid boundary box in MLS-MPM with beveled edge node surface-normals at box edges and corners. Numerical load-cells were implemented on the box's front face as grid-node streamwise force sensors in ClaymoreUW. Columns under the structural box have yet to be added to our digital twin, but will be if observing damming in the WASIRF flume becomes a priority of future work.

Instrumentation of the flume with wave-gauges (WGs) was installed to measure the water's

free-surface at key locations and Acoustic Doppler Velocimetry (ADV) devices were used to measure fluid velocity. Gauges showed oscillation in flow depth for higher frequency pump settings but it was very consistent when time-averaged, allowing for stochastic use given many trials were performed at constant flow settings, as done in our experiments. ADVs showed mostly consistent flow-speeds to broadly support near steady-state flow assumptions, although some experimental velocity time-series were noisy. This applies to the first-round of WASIRF tests in August 2021.

Debris made of High Density Poly Ethylene (HDPE), same material as used in Park et al. 2021 at the OSU DWB and in Mascarenas 2022 for the OSU LWF, were cut into varying rectangular prisms. Debris used are shown in Figure 9.10 and Table 9.3 and were placed 0.8 m upstream of the structural box, half a grid-cell above the flow, centered transversely in the flume,a and with no initial velocity . Simulated debris is arranged according to the experimental debris-array configuration when it is a non-random case. We impose a three grid-cell, Δx, spacing between debris to improve debris-debris interaction. In some simulations debris are modeled as finite elements with an ASFLIP contact algorithm (Sec. 5.6.7), which improves their ability to pull apart but the element mesh of tetrahedrons can introduce impact damping at low resolutions (see debris-structure results in Section 6.2).

The full experimental itinerary for the UW WASIRF flume tests can be found in a planned publication by Nicolette Lewis, and all data is intended to be published as a DesignSafe DataDepot data-set in the coming year.

Prototype and model simulation parameters are stated in Table 9.5, with further specifications to be provided in a future update to our numerical digital twin of the wave flume. Said update may introduce wind-wave capabilities in ClaymoreUW and HydroUQ. All simulations shown here use our Multi-GPU ClaymoreUW code (see Chapter 4).

Load-cells are recreated numerically as an output of summed stream-wise force on the leading

Table 9.3. UW WASIRF debris in the prototype and model. Based on experiments by Lewis 2023. Dimensions set by Length x Width x Depth. Applies 1:40 geometric and Froude length scales.

ID	Debris Type	L_p	W_p	D_p	L_m	W_m	D_m
-	-	[m]	[m]	[m]	[m]	[m]	[m]
A	Motorcycle	2.03	1.16	0.51	0.0102	0.0254	0.0127
B	Small - Medium Car	4.06	1.16	1.02	0.0102	0.0254	0.0254
C	Small - Medium Car	4.06	4.06	1.02	0.0102	0.0102	0.0254
D	Truck / 20’ Crate	6.10	1.16	0.51	0.0152	0.0254	0.0127
E	Small Bus / Sail Boat	8.13	1.16	1.02	0.0203	0.0254	0.0254
F	Regular Bus	10.16	1.16	0.51	0.0254	0.0254	0.0127
G	Long Bus / 40’ Crate	12.19	1.16	0.51	0.0305	0.0254	0.0127
H	Yacht / 48-53’ Crate	15.24	4.64	1.02	0.0381	0.0102	0.0254
I	Yacht / 48-53’ Crate	15.24	4.64	2.03	0.0381	0.0102	0.0508

Table 9.4. UW WASIRF wave event in the prototype and model. Based on experiments by Lewis 2023.

Parameter	Prototype	Model
Geometric Scale	1:1	1:40
Flume Length (m)	740	18
Flume Width (m)	36.5	0.9
Flume Height (m)	-	1.2
Standing Water Level (m)	0.8 - 4.4	0.02 - 0.12
Max Flow Velocity (m/s)	0.32 - 3.80	0.05 - 0.60
Inundation Duration (s)	64 - 380	10 - 60
Debris Event Duration (s)	32 - 190	5 - 30
Flow-Speed at Structure (m/s)	0.32 - 3.80	0.05 - 0.60
Flow-Depth at Structure (m)	0.8 - 4.4	00.02 - 0.12

Table 9.5. UW WASIRF simulation (MLS-MPM) settings in the prototype and model. Replication of experiments by Lewis 2023.

Parameter	Prototype	Model
Geometric Scale	1:1	1:40
Fluid Bulk Modulus (GPa)	2.2	2.2
Debris Young's Modulus (GPa)	0.8	0.8
Grid-Cell Spacing (m)	1.0 - 2.0	0.025 - 0.05
Particles (Million)	0.75 - 6	0.75 - 6
Time-Step (s)	3e-4 - 6e-4	7.5e-6 - 1.5e-5

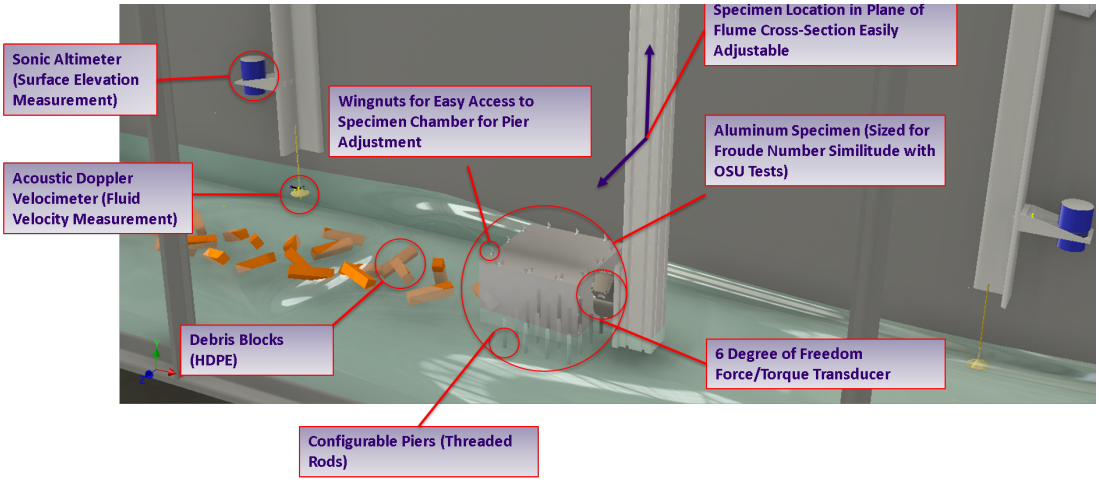

Fig. 9.9. UW WASIRF Digital Twin Visual Render. Visualization of the UW WASIRF flume for the experimental specification of Lewis 2023. Basic instrumentation, structural set-up, flow conditions, and debris transport is show. Image is not a numerical simulation.

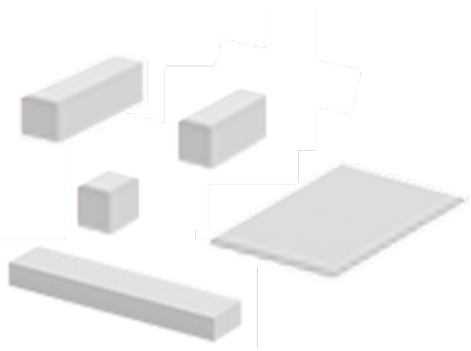

Fig. 9.10. UW WASIRF Debris.. Debris geometries from first round of experiments at the UW WASIRF flume.

face of the structural box. Force at each grid-node on the structural face is calculated through momentum change (assuming rigidity) and then added together. Output frequency is identical to experimental instruments.

Validation of the debris-field tests in the UW WASIRF flume are yet to be undertaken. Hydrodynamic cases will be used to check for consensus between the numerical and experimental results for staple variables such as water-level, water-velocity, and force on the structural box.

However, initial simulations using the digital twin were trialed for the purpose of evaluating whether UW WASIRF debris impact simulations would behave as a scaled-down version of the OSU LWF debris impact simulations. Figure **??** shows a low resolution pilot simulation with flow velocity visualized on fluid particles and force vectors superimposed to show impact and hydrodynamic reactions on the raised structural box. A transversely oriented debris field impacts the box and breaks apart in a way very similar to experimental and numerical observations in Chapter 7 at the OSU LWF. It does appear that it is somewhat more difficult for the WASIRF debris to displace the water buffer in front of the raised structure, but this is to be expected considering we have dropped the geometric length scale while maintaining identical material properties.

Overall, this is a positive result that our high-performance MLS-MPM methodology is applicable to smaller scale wave-flumes for debris impact studies. A full suite of simulations for the UW WASIRF flume may be revisited in 2024 during development of a potential open-source digital twin for the UW WASIRF flume inside of the NHERI SimCenter's HydroUQ software. Particularly, we are interested in modeling the wind-wave feature of the WASIRF flume in future efforts, as winds can amplify the hazard of waterborne events so there is some reason to believe winds may amplify the hazard of wave-driven debris field impacts.

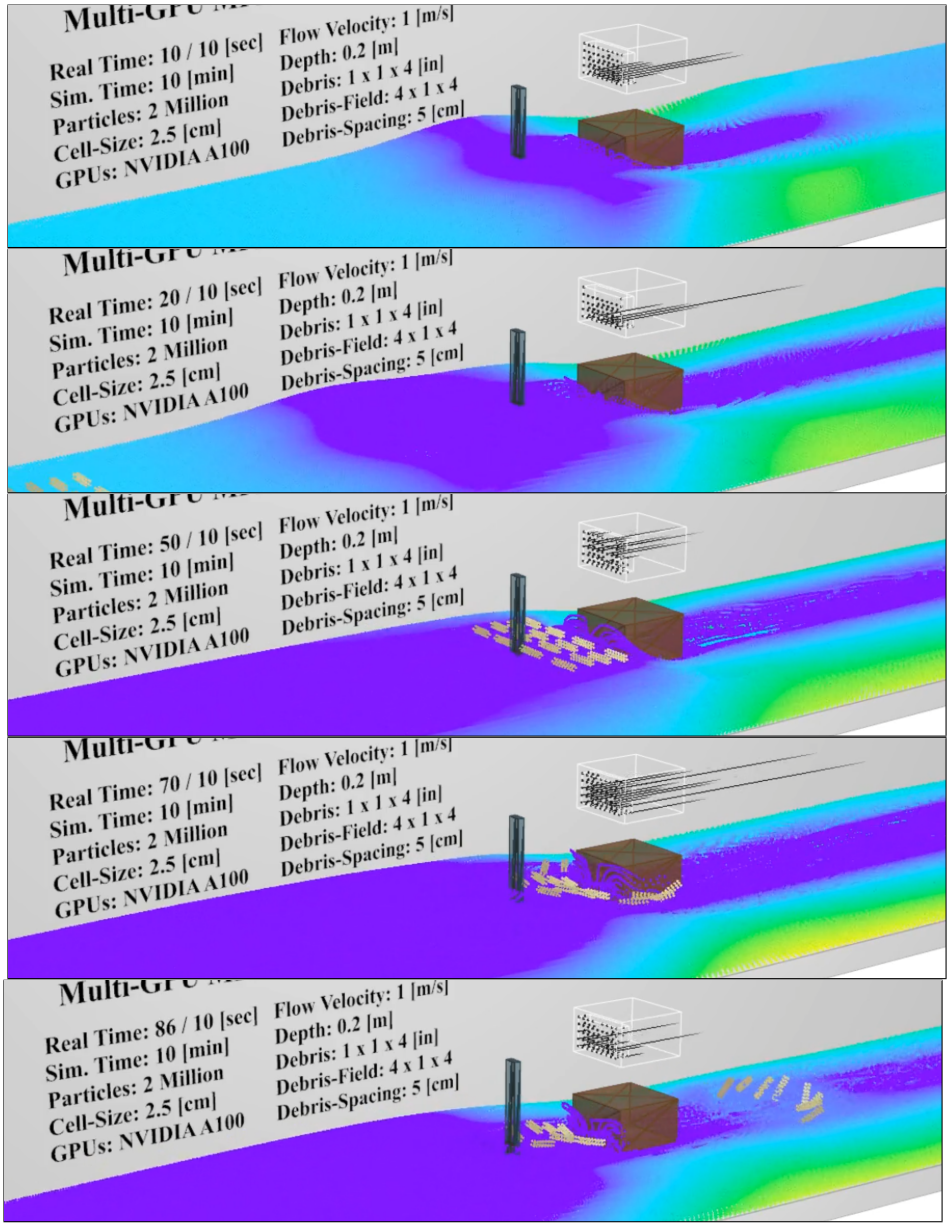

Fig. 9.11. UW WASIRF numerical debris field hits a structure in pump-driven flow. UW WASIRF debris field impact test replicated in our MLS-MPM. Load vectors visualized above box, instrument shown in front of box as a long prism. Experiments by Lewis 2023.

9.4 *Gravity-Generated Lahar Carrying Debris-Fields Down a High Friction Slope - USGS Volcanic Flume*

Problems governed by large deformation and displacement extend beyond coastal engineering. Therefore, in this section's final study we considered a natural hazard rooted in geotechnical engineering that may benefit from the MPM tool presented in this thesis. Lahars (i.e. debris-flows) are soil-rock-mud based flows down hill-slopes which attain significant momentum that, quite honestly, is almost impossible to design for typical structures to withstand. To improve this lack of mitigation for the built environment, Professors Michael Motley, Pedro Arduino, and Jeffery Berman with students Abbey Serrone and the author of this dissertation (Justin Bonus) proposed a research project to address said issue. Pilot simulations were completed in our high-performance MPM software, ClaymoreUW, to form the technical basis of our proposal. Note that our group, specifically Professor Arduino's lab, has studied debris-flow / landslide type problems with MPM for many years. In fact, we were one of the earliest groups to apply MPM to this hazard in a way appropiate to understanding it for structural loading. This credit goes to former students Woo-kuen Shin (Shin 2007; Shin 2009) and Carter Mast (Mast et al. 2014; Mast 2013), who performed proof-of-concept studies for landslides, debris-flow, and avalanche research during their graduate studies.

What follows in this section results of the pilot simulations done in pursuit of a open-source digital twin (Figure 9.12) for the USGS debris-flow flume near Blue River, Oregon. Some exposition and figures are adapted from the research proposal, written by aforementioned Professors and Abbey Serrone, and the experimental paper which we sought to replicate numerically by Iverson et al. 2010, which is by an author very well known for the study of debris-flows. We show simulations that capture the collapse, flow, and runout behavior of frictional, granular material at high velocity. Our pilot study results already give results, in 3D, that exceed what most numerical methods could hope to reach due to MPM's efficacy in modeling complicated materials in

Fig. 9.12. Digital twin of the USGS Volanic Hazard Program's lahar flume during a dry sand-gravel experiment. Digital twin implemented in ClaymoreUW for the Material Point Method.

dynamic, large-deformation scenes. Further, we show an initial simulation of a mitigation avenue for debris-flows, i.e. placing a large rigid box at the end of the hill-slope, and visualize its effect on flow, pressures in the 3D soil mass through time, and quantify the forces the box must resist to remain rigid. This simulation is violent, sand and gravel hit the box with incredibly high momentum so resolving without numerical instability is significant for future debris-flow studies. Forces quantified are also extreme, highlighting the threat posed by this hazard locally to the built environment.

Lahars result from the rapid release of volcanic debris and water, and the risk lahars pose in Washington is high. The most recent major event was associated with the 1980 eruption of Mount St. Helens, and the Cascade Range hosts a particular combination of threatening factors, making it one of the most high-risk areas in the world. The United States Geological Survey names lahars the "most threatening volcanic hazard in the Cascades", yet there is no guidance for engineers

looking to provide resilient infrastructure. Many communities are seeking out creative ways to improve their hazard resilience, but in some cases the lack of available information leaves problems without solutions. The objective of this research in the long term is to develop a comprehensive understanding of the interactions of a lahar with the natural and built environment as a first step toward quantification and codification of a set of loading guidelines for structural engineers. The immediate goals of this proposal are to characterize a series of typical lahar events in terms speed, depth, and composition of the flow; extend existing numerical models to develop a digital twin of an existing flow test facility to be used for experimental design; and to perform a comprehensive scan of the topology of the lahar risk zone in the vicinity of Fife, WA, a small community within Mount Rainier's lahar risk zone, to incorporate into a prototype numerical model of the city for real world, proof-of-concept studies.

Debris-flow experiments by Iverson et al. 2010 were performed at the USGS Volcanic Hazard Program's debris-flow flume near Blue River, Oregon, United States. A diagram showing the side-view of the flume, including the initial soil mass behind a gate and the plan-view of the runout pad topography, is on the left in Figure 9.13. On the right are the elevation contours of two sand-gravel (not saturated). By replicating the initial soil mass and flume in MPM we are attempting to replicate similar runout patterns on the left after soil-mass release.

The USGS flume is used to conduct large-scale experiments that test ideas about landslide formation, behavior and hazards. The flume is a reinforced concrete channel 95 m (310 ft) long, 2 m (6.6 ft) wide, and is on a 31 degree hill-slope. There is approximately 1-2 degrees of deviation in the channels horizontal bearing in a portion of the flume length, due to soil foundation changes and construction limitations, which causes moderate asymmetry in experimental flow. This isn't numerically replicated in our digital twin so far.

Macro-frictional pads (i.e. large, raised bumps to model geometric friction effects as opposed to a friction coefficient in Couloumb material models of friction) are placed along the flume's slope to

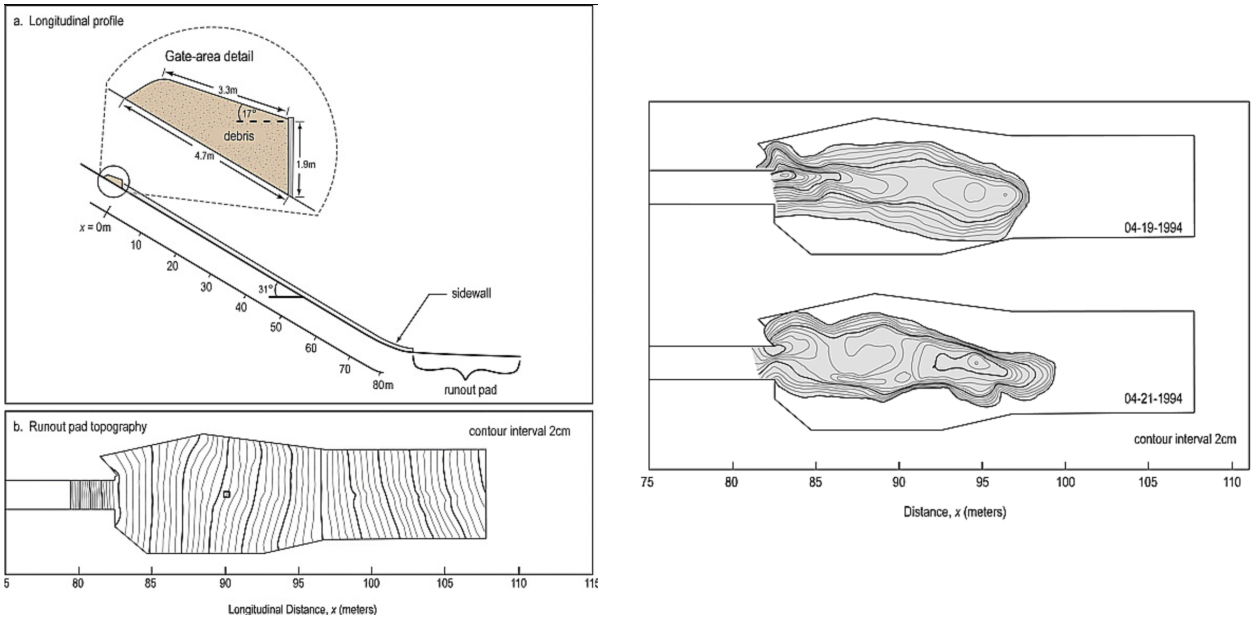

Fig. 9.13. Diagrams of Iverson 2010 debris-flow experiments. Soil mass consisting of sand, gravel, and mud (depending on the case), sketched for its initial state shown. Located at top of sloped flume behind a gate, when released will fall downslope onto a horizontal runout plane.

replicate hill-side roughness. In MPM, this was replicated as Couloumb friction with a decay layer (Yang 2016) to extend its influence vertically as substitute for the experimental bump heights. However, if we increased the grid-cell resolution to 1.25 cm or less (very viable) we may be able to model every bump on the flume's slope and runout pad as individual boundary conditions (i.e. separable velocity conditions on grid-nodes with surface normals set appropriately). This will certainly improve replication of the "bouncing" of gravel material off the flume's slope, which is partially involved in the sorting of aggregate during debris-flows and is not fully matched in Couloumb friction only models.

For the sand-gravel mixure, we applied the common Drucker-Prager material model. We neglect particle size distributions provided in Iverson et al. 2010, i.e. gradation from sand to gravel (and fines for some tests), for simplicity of the pilot study, though higher resolutions may be able to replicate it directly with mixed particle/particle densities, or in a two-phase formulation. Large

Fig. 9.14. Photographs of Iverson 2010 debris-flow experiments. Soil mass consisting of sand, gravel, and mud as it is released by swinging gates at the top of 31 degree sloped flume. Portions of flume have "bump pads" (grey) to replicate prototype slope roughness. Load-cell ports have sides of 34cm (black), load-cell faces are 25.2 cm (green).

gravel could also be modeled with DEM if needed to resolve grain redistribution. For our homogeneous material assumption we tested 1e7 and 1e8 Pa Young's modulus with Poisson ratio of 0.2. Density was set to 2400 kg/m^3. As the numerical MPM model did not use water, i.e. a purely dry simulation, it differs from the Lahar experiment which had noted liquefaction and separation of dry and saturated soil. However, future efforts in MPM could incorporate fluid effects and improved soil-rock mixture interaction with a variety of different approaches in the literature (Chandra et al. 2023; Zheng et al. 2021; Li et al. 2023; Tran et al. 2023; di et al. 2023). Replication of broad behavior of the Lahar experiments in our ClaymoreUW MPM numerical model can be seen in Figure 9.15, which is a ground-level front view of the flume as the debris-flow travels down the steep slope onto a runout plane.

Plan-view images of the Lahar runout on the horizontal bed at the flume's base are shown for the experiments and simulations in Figure 9.16. It shows simulated runout elevations at two points in time compared to the corresponding experimental runout images from Iverson et al. 2010. Note that experimental images were from tests which included saturation of part of the sand-gravel mass, so the images we chose are taken before the saturated, liquefied soil reaches the runout plane (roughly at 14 seconds) as this was not modeled in our MPM pilot simulations.

Full characterization of simulated runout flow across time is shown in Figure 9.17. Note that while there is deviation in some of the simulated runout shape, primarily due to an experimental case which included some saturated soil and varied grain-size distribution not modeled numerically, the overall runout length, width, geometry, elevations, and time taken to attain a static state are remarkably similar. Also, our simulation did so while carrying a debris field meant to represent cars and tree trunks, viewable in the elevation contours, showing stable entrainment interaction of the flow with debris.

Referring back to Figure 9.15 it is observed that there is erosion of entrained debris fields, i.e. not all debris stay at the front of the mass. This is clear by looking towards the top of the hill slope

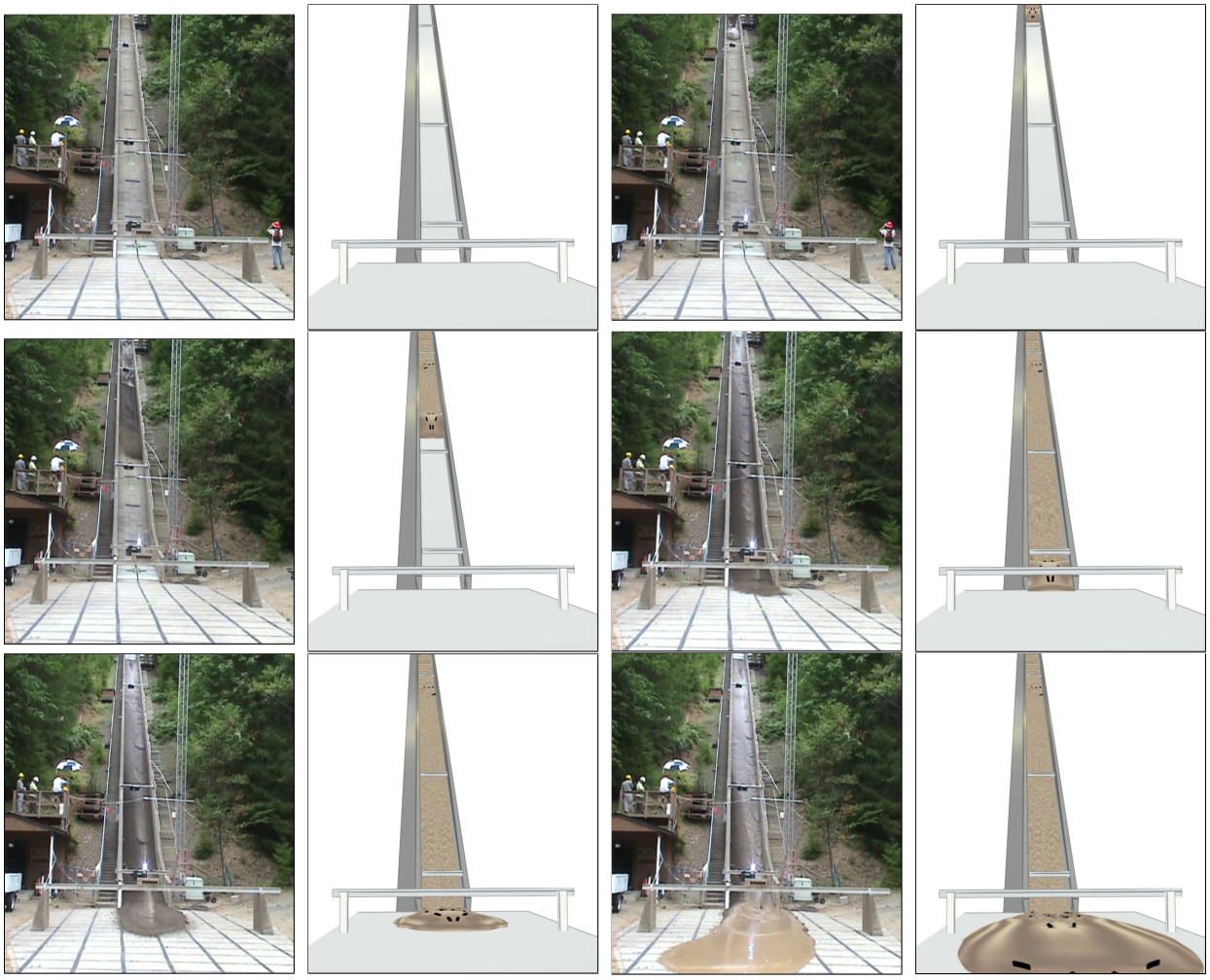

Fig. 9.15. Front-view of Lahar channelized flow experiments compared to MPM simulations. Experiments included saturated soil, while simulations were dry. Performed at the USGS debris-flow flume near Blue River, Oregon, USA.

where a few pieces of debris are seen to rest. Replicating debris field erosion-entrainment phenomena in our MPM approach is significant to the study of lahars.

Figure 9.16 shows simulated runout elevations at two points in time compared to the corresponding runout images from Iverson et al. 2010. Note that experimental images were from tests which included saturation of part of the sand-gravel mass, so the image we chose are taken before the saturated, liquefied soil reaches the runout plane (roughly at 14 seconds) as this was not modeled in our MPM pilot simulations. Of course, such behavior is well studied and viable to replicate in MPM for future study of Lahars in our group (Zheng et al. 2021; Mei et al. 2020; Chandra et al. 2023).

Figure 9.18 characterizes the loading through time (as vectors) of a high-velocity debris flow similar to Iverson et al. 2010 as it collides with a rigid box with blockage ratio of 0.66, at the end of the flume channel. Further, the pressure in the soil mass throughout impact is visible, and the motion past impact of the debris-flow is seen to have dissipated significant kinetic energy, meaning the runout on the flat-plane features less volume and significantly less hazard to any infrastructure that may be situated their in a prototype case.

9.5 *A Numerical Platform for Simulating Nearly Any Multi-Hazard Scenario*

Our approach to Debris-Fluid-Structure Interaction within high-performance Material Point Method is applicable to a swath of natural hazards. While tsunami hazards were numerically replicated in scaled experimental wave-flumes in Chapters 7 and 8 and Sections 9.3 and 9.2, this is not all that MPM is capable of and arguably this hazard is not even one of its primary strengths. With moderate effort, we took the exact methodology used in our tsunami-driving debris-fields into structures simulations and applied it to Lahars-driving debris-fields into structures simulations in Section 9.4. This unusual ease in application to a fundamentally different hazard mode is significant. While depth-integrated methods and other advanced tools, e.g. smoothed

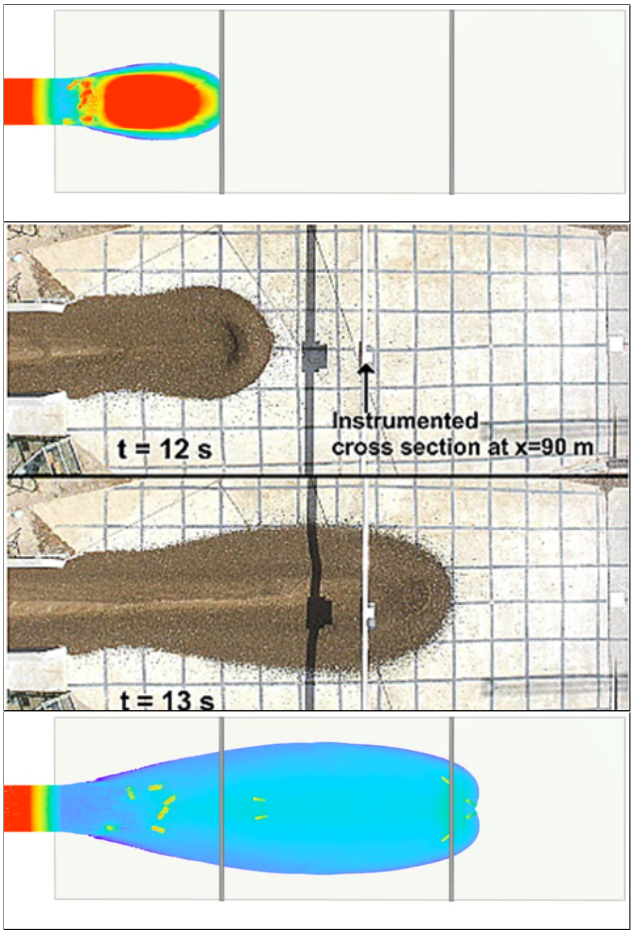

Fig. 9.16. Plan-view Lahar runout footprint experiments compared to MPM simulations. Elevation contours shown for simulations. Debris carried by simulation flow stand-out. Experiments included saturated soil, while simulations were dry. Performed at the USGS debris-flow flume near Blue River, Oregon, USA.

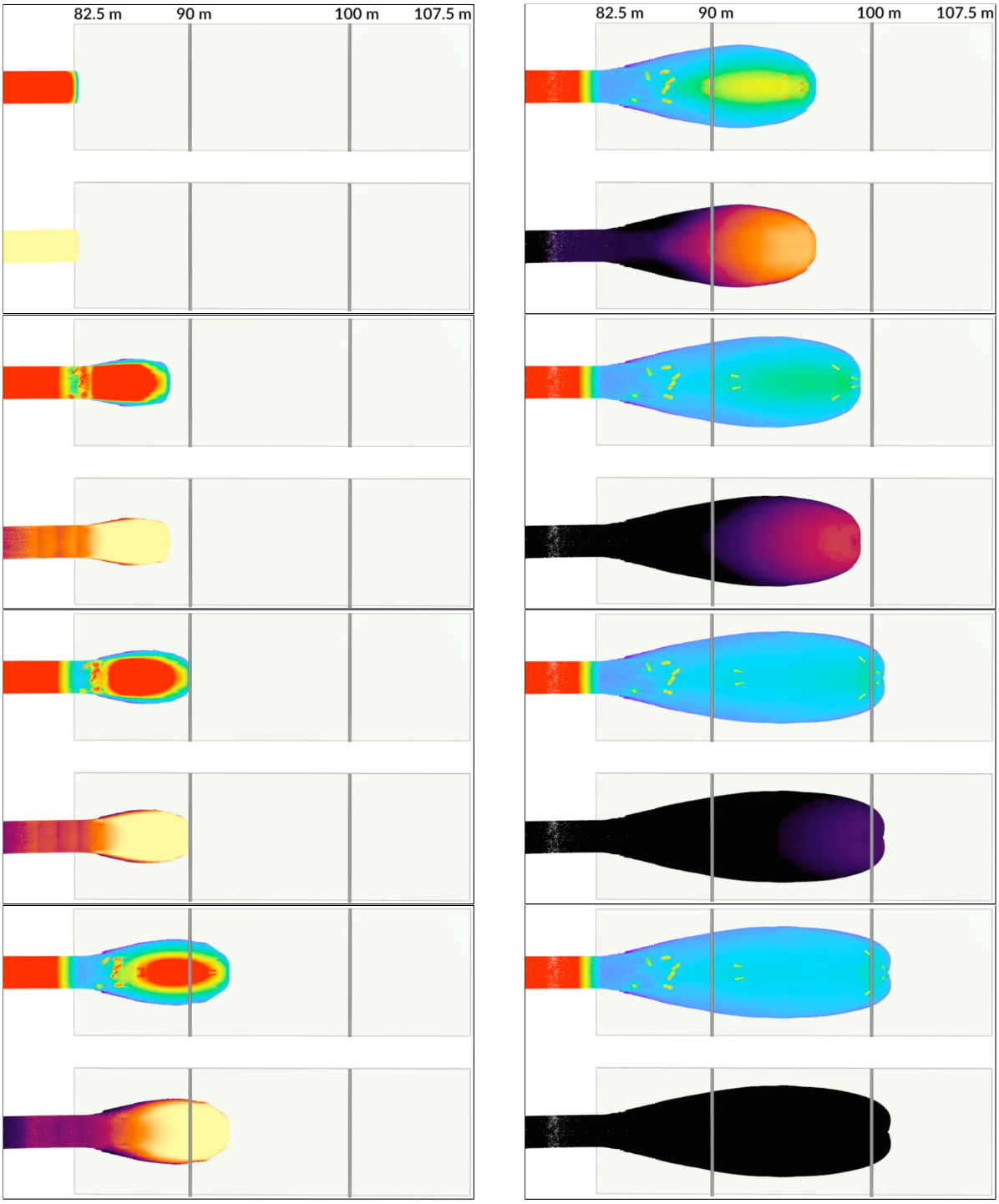

Fig. 9.17. Plan-view of channelized dry sand-gravel runout MPM simulation. Based on experiments in by Iverson et al. 2010. Simulations colored in each panel to show elevation contours on the top (with noted visibility of driven debris) and streamwise velocity on the bottom. Simulations do not model fluid phase. Drucker-Prager material used. Run-out plane is frictional and nearly plum (i.e. nearly horizontal). Performed at the USGS debris-flow flume near Blue River, Oregon, USA.

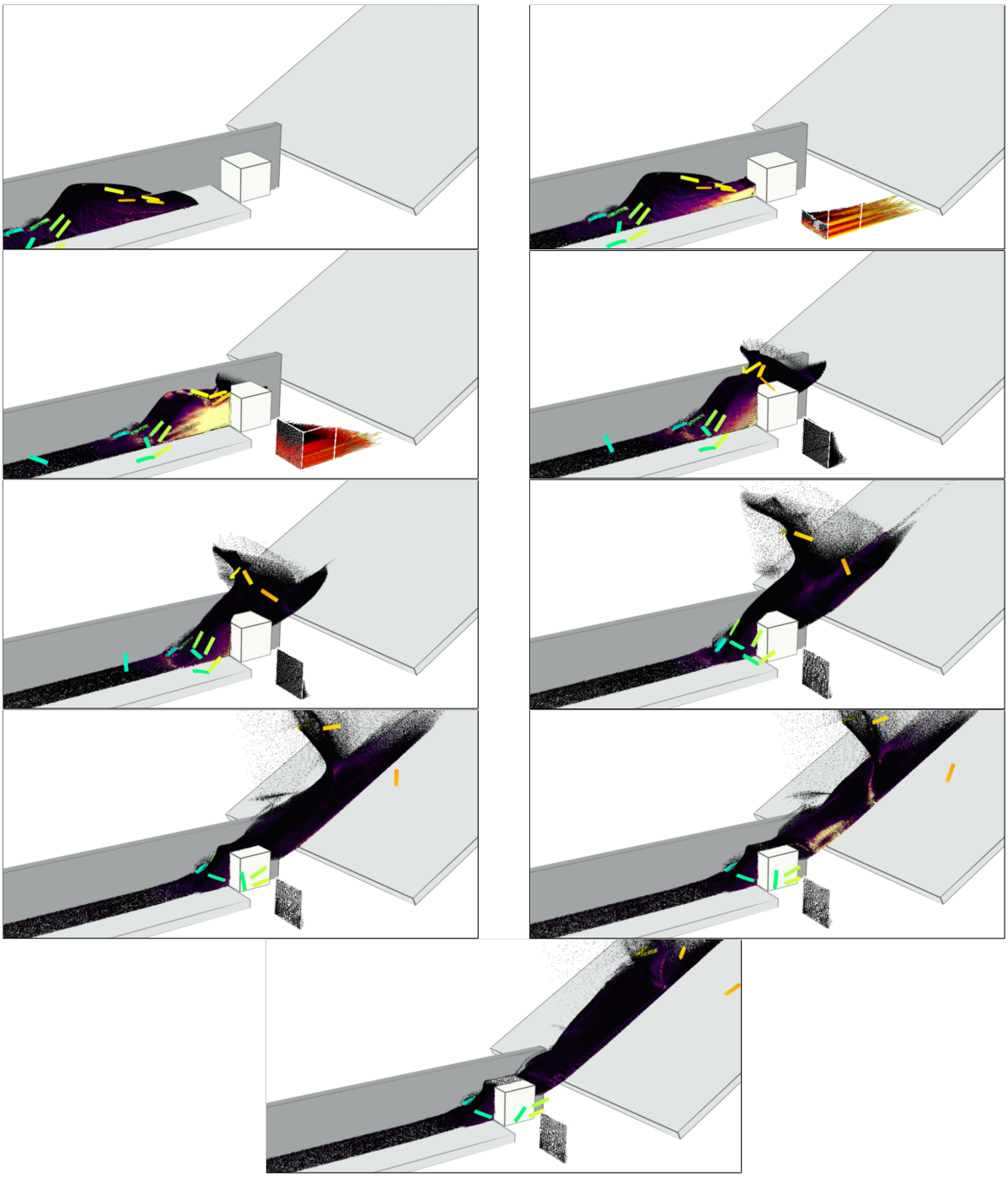

Fig. 9.18. Pressure in a sand-gravel mass as it collides with rigid mitigation structure. Based on experiments in by Iverson et al. 2010 with an added numerical box mitigating structure. Pressure visualized on sand-gravel particles with brighter colors being higher pressure. Debris are colorized according to which number debris to allow visual continuity between panels. Reaction force vectors on the upstream face of the rigid mitigation structure are visualized as vectors offset from the channel. Only half of the sand-gravel flow is shown (i.e. near-side of the channel's flow is invisible) to give a cross-sectional view of pressure development.

particle hydrodynamics, may also exhibit cross-applicability it is not with anywhere near the level of abstraction MPM exhibits: only requiring a change in material constitutive law, which is isolated on particles to allow elegant coupling with other material, e.g. other hazards, debris, and structures by default. Further, MPM excels at advanced material laws compared to idealized fluids oriented schemes, which excel much greater but constrained within their limited scope. The benefits MPM may possess, as a Jack of all trades, for the natural hazards modeling community could be immense. Namely, the ability to rapidly assess multi-hazard scenarios in complex environments may expedite the hazard identification, assessment, and mitigation process, allowing more time be devoted to tailored, specialist tools when the primary mode of concern is identified.

MPM is likely applicable to storm-surges and other waterborne hazards beyond tsunamis. This is an increasingly key hazard to quantify as sea-levels are anticipated to rise by 0.3 and 0.8 meters by the 2030s and 2080s, which may produce storm-surges inundation depths of 0.5 to 1.8 meters respectively Mousavi et al. 2011. While our simulations of wave-flumes primarily applies to tsunamis due to use of spilling solitary waves in almost all cases, we did observe fundamental effects of debris motion in a steady-state undulating flow in our UW WASIRF wind-wave-flume (see Section 9.3) and we replicated the reorganizing behavior of heterogeneous density (mix of wood and plastic) debris-field motion that was observed by Park et al. 2021 at the OSU DWB wave-flume. There are likely great applications in MPM for wind-water hazards with debris field motion and loading.

Avalanches have been a growing topic of study in MPM, though our group has investigated them back in Mast et al. 2014 along with landslides in the context of mitigation. As MPM can 3D model the entire event (static, triggering, collapse, flow, run-out, rest, etc.) (di et al. 2023) while interacting with debris and structures it shows unique promise for these hazards. Recent advances in this field are noted in Gaume et al. 2023, whom identified a shortlist of MPM's key applications including: **(i)** snow micro-structure deformation and failure, **(ii)** avalanche release (Guillet et al.

2023) (Gaume et al. 2019), **(iii)** avalanche dynamics over complex topography Li et al. 2020, **(iv)** interaction between avalanches and obstacles, forests or lakes (Védrine et al. 2022), and **(v)** erosion and entrainment (Védrine et al. 2022). The fourth point concerns structural impacts of debris carried by avalanches, much like our work but for tsunamis and debris. Depth-averaged MPM tools are also shown to be an interesting line of pursuit for slope-scale prototype modeling of avalanches in both Gaume et al. 2023 and Guillet et al. 2023.

Above ground and submarine landslides may trigger tsunamis of significant heights, e.g. mega-tsunamis, and run-out length. MPM is reputable in its capability for the 3D start-to-finish simulation of these events (Zhang et al. 2023).

9.6 *Remarks*

Our Multi-GPU MPM approach shows strong preliminary promise in application to alternative flumes than those that formed the bulk of our simulation efforts, i.e. the OSU LWF and Waseda University's flume. Wide wave-flumes, pump-based wave-flumes, and gravity-driven lahar-flumes all showed decent behavior in the first few-passes of simulations with fairly minor effort. The applicability to Lahars / debris-flows driving debris-fields is especially intriguing, as it suggests our MPM approach may apply to not just tsunamis, but Lahars as well. Further, there is reason to believe it can apply to avalanches (Stomakhin et al. 2013) and storm-surges, though these efforts are not fully fleshed out yet in our own work. This would constitute a minimum of four natural hazards of interest, with the secondary hazard of debris-fields appending to each. MPM may be the future direction for a wide variety of complex, large-deformation natural hazard numerical tools.

Chapter 10

DEBRIS-FIELD CHARACTERIZATION

10.1 *Introduction*

Coastal infrastructure is exposed to the barely quantified hazard of debris-field impacts and damming during inundation events. Faced with potentially significant design consequences from this secondary hazard, engineers may find themselves lightly equipped.

ASCE 7 and FEMA P-646 suggest limited design guidelines, only applicable to basic single-debris scenarios. Though consistent with the best available science, debris-field loading was identified as a subject for improvement in design guidelines (Nistor et al. 2017a). This chapter documents part of our effort to advance understanding of stochastic debris-field loading, primarily by evaluating how we should quantify debris-fields.

Current design guidelines prioritize single-debris impacts. Stiffness and velocity control load predictions via simple equations (Aghl et al. 2014b). They assume elastic collision of debris characterized as a 1D bar with full momentum transfer. Stiffness of debris (k), impact duration (t), and max force of impact (f) are found as

$$k = \frac{EA}{L} \quad \text{and} \quad t = 2\sqrt{\frac{m}{k}} \quad \text{and} \quad f = v\sqrt{mk}\ , \tag{10.1}$$

where E is the debris' Young's modulus, A is area of impact, L is length of debris, v is velocity of impact, and m is mass of debris. Complex debris require more information. One may phrase basic

debris design parameters as set $\mathbb{D}$:

$$\mathbb{D} = \{\rho, E, A, L\} \tag{10.2}$$

This can be used to quantify common debris hazards, but it does not capture debris-fields, complex debris, fluid conditions, or obliquity of impact. Proposed methods account for obliquity, water displacement, drag, and mobilization, however, no current approach extrapolates into debris-field hazards.

Debris-field hazards are short-duration, infrequent, and highly variant. Data is perishable from inundation drawback and post-disaster clean-up (Naito et al. 2014) so little is known about them. Further, loads are chaotic and so require large volumes of nonexistent observations to characterize stochastically. Empirical frameworks negated by minimal records, simulations opposed by computational and hazard complexity, and theory subverted by dynamic, nonlinear debris-fluid-structure interaction (DFSI), this chapter is a window into our active remedy.

We propose a means to quantify debris fields and their hazard to structures. The method is based of preliminary numerical results from our advanced numerical tool (Chapters 2 and 5) and our wave-flume experiments (Chapters 7, 8, and 9). We will use our numerical tool, ClaymoreUW (Chapter 4), to simulate various types of debris-field, demonstrating that we can define complex debris-fields according to our simple quantification scheme. Further, we create predictive models for structural demand resulting from numerical debris-field impacts.

10.1.1 Debris and Debris-Fields

Debris and debris-fields may be seen as an element and subsequent composition, or as two fundamentally unique entities. The literature usually falls into the former or latter view, but justification insofar for either has been weak, or at least short-sighted. Logically, a debris-field should be a simple summation of debris-structure and debris-debris collisions. Observably, this does not hold to be true because debris-fields of identical configuration produce different results

every time they are tested in experimental wave-flumes. Current models fail to capture the full behavior of debris-fields, which are highly nonlinear despite the reasonably linear behavior of constituent interactions. It is tempting to use chaos as a blanket justification, and in part we do so here, but research into debris-fields is still sparse enough to anticipate a physical explanation that does not abandon first-principals.

In any analysis of this type, we must take care to consider the scales at which debris and debris-fields manifest. In doing so we may be able to design a characterization scheme that is mostly invariant to scale and the blurry transition between a debris-field and a group of debris.

Microscopic. Collision between a piece of debris and a structure or collision between two pieces of debris. For the most part, well described by even simplistic analytical equations. **Mesoscopic.** Collection of collisions between a moderate number of debris (roughly 3 to 10) and a structure. As of now, primarily characterized by a handful of scaled flume experiments. **Macroscopic.** Interacting field of debris that may no longer be viewed simply as a summation of microscopic collision events. Currently, this scale is poorly characterized and exhibits behavior not fully explained by first-principals, usually due to stochastic effects. Other fields of study have a better handle on this scale but little work exists for tsunami-driven debris-fields of arbitrary composition.

10.2 *Independent Variables*

To better quantify debris-field impacts, we define independent variables for debris and debris-fields. In doing so, we choose variables that are: **(i)** measurable, **(ii)** distinct, **(iii)** applicable across debris-field scales, and **(iv)** predictive of dependent debris-field impact variables for engineering design.

A simple set of debris-field independent variables ($\mathbb{F}$) may concern **(i)** inheritance of debris sets, **(ii)** debris count, **(iii)** domain occupation, and **(iv)** relative orientation. The set of debris-field

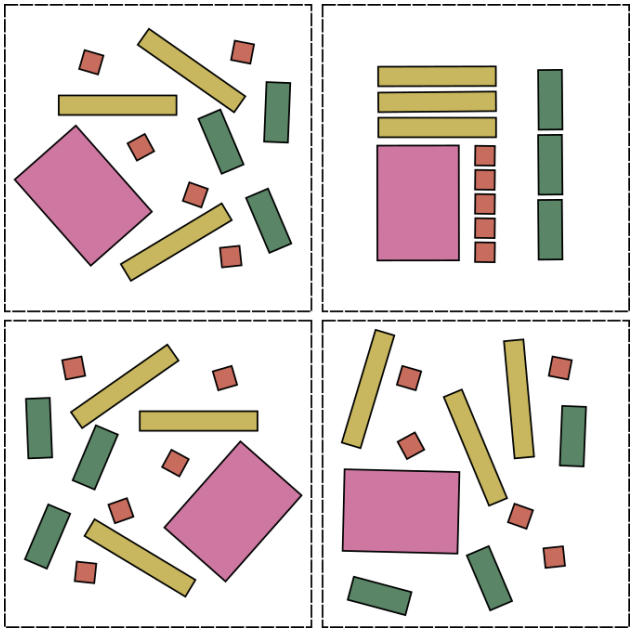

Fig. 10.1. Example Debris-Fields. The same set of debris compose each panel's debris-field. Differences derive from their arrangement.

independent variable ($\mathbb{F}$) is stated.

$$\mathbb{F} = \{\mathbb{D}, c, n, \theta\} \ , \tag{10.3}$$

where c, n, and θ are placeholders for general measures of debris count, occupation, and orientation and $\mathbb{D}$ is previously defined in Equation 10.2.

The categorization of debris-fields vs. debris ($\mathbb{F}$ vs. $\mathbb{D}$) is important. A schematic demonstrates how a heterogeneous debris-field can manifest in exceedingly unique ways (varying $\mathbb{F}$), even when debris parameters are held constant ($\mathbb{D}$) is shown in Figure 10.1. A more visual interpretation of these debris-field measures is presented by parking-lots, a common debris-field hazard, in Figure 10.2.

Fig. 10.2. Parking Lot Debris-Fields. (upper-left) Ordered parking lot. (upper-right) Angled parking lot. (lower-left) Sparse parking lot. (lower-right) Parking lot evolved by 2011 Tohoku tsunami. Courtesy of NHK World TV.

10.2.1 Count

The simplest measure of a debris-field is the number of debris within (c), For example, the number of cars in a parking lot. Most literature relies on this parameter, as strong trends are found with a number of demand parameters (usually momentum related variables, e.g. impulse on structure). Note that transformations of c (e.g. $\sqrt{c}$) are often more statistically significant for predicting structural quantities, like maximum impact force in both our numerical simulations and wave-flume experiments.

Despite the simplicity of debris-count and its relative robustness in ball-park predictions of structural demand, we can not always expect to know the number of debris in a debris-field, let alone a mean and variance for stochastic design. Even when we do, it is inadequate to explain much of the seemingly chaotic behavior in demand parameters (e.g. max force). Further quantification parameters for debris-fields exist and should be considered for improved stochastic design for tsunami-driven debris-field impacts.

10.2.2 Occupation

A debris-field's occupation of a given footprint, i.e. an area where debris can be "placed" by natural means (e.g. drift-wood on a strip of beach, a car in a parking-lot), can greatly influence the hazard it presents. Measures of occupation can be estimated with relative ease through video or images of debris sites.

We prefer to set the footprint of occupation as an L x L area, where L is the width of the structural face our debris-field is impacting against, as it helps to normalize all our calculations by the primary structural component (i.e. the resisting face) of design interest. To decouple debris-fields from structures, it may be preferable to define L as a side of the smallest square containing the debris-field, oriented along the stream-wise flow direction. A number of isotropic measures of occupation are available. Potential occupation measures (**n**) are porosity (n), void-ratio (e), and spacing (s):

$$n = \frac{A_W}{A_T} \quad \text{or} \quad e = \frac{n}{n-1} \quad \text{or} \quad s = \text{ Avg. debris spacing,} \tag{10.4}$$

where A_W is the water occupied area contained in A_T which is the total area, i.e. "foot-print", of a debris-field.

Void ratio (e) may be preferential to porosity (n) as a predictive meta-parameter for impact and damming forces. It is more sensitive to relative occupation of solids, so a physical phenomena like debris-field impacts that is influenced by small changes in debris packing will have a more natural

relationship with it than porosity. For instance, moving from 20% debris occupation to 10% will probably have a significant impact on impact and damming forces. Spacing (s) is not preferred because it is difficult to quantify from aerial photographs but it is an option. Transformations can be applied to occupation measures to improve predictive capabilities and distributions of data, e.g. using $\log n$, as they tend to be skewed and thus can damage statistical significance of linear regression models if not adjusted for.

An occupation measure of a debris-field can improve quantification of its hazard to structures. However, it is still not enough to fully quantify the field. Another variable that is known to influence structural demands, such as max force of impact, is the orientation of debris (Nistor et al. 2017a).

10.2.3 Orientation

Debris orientation (θ) relative to fluid flow and a structure's normal vector at point of impact is a commonly identified parameter of significance. Usually, a mean (μ_θ) and standard deviation (σ_θ) is found. This is seemingly stochastic and has utility, but it misses something essential. The actual distribution of the full debris-field, in its entirety, has profound implications for ultimate behavior. Rose diagrams characterize orientations of all debris-field debris in a binned manner to allow for a visual representation on polar coordinates. Instead of looking at a single rotation value for a debris-field we use the whole distribution from which it may be possible to observe finer trends. Phrased as a cumulative distribution function (CDF), a clearer view is gained, shown in Figure 10.3 by a rose plot (left) and empirical CDF and PDF for a randomly sampled debris-field. These CDFs are information dense, but impractical– They can't be easily "plugged into" a building code equation or presented as a table. Debris-field obliquity is a distribution ($P(\theta)$), not a single-value. Drawing from geotechnical soil sieve charts, obliquity distributions are characterized by simple meta-parameters to describe the continuous curve, not the discrete points.

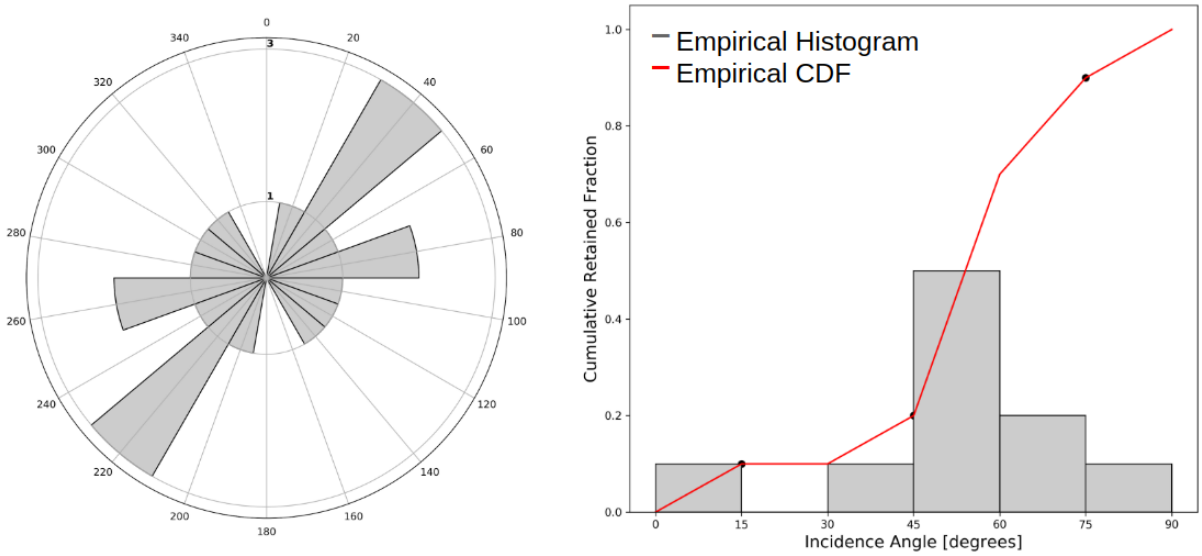

Fig. 10.3. Debris-Field Rose plot and Empirical PDF/CDF (Left) Rose plot of debris-field internal rotations. (Right) Empirical histogram and CDF of debris-field internal rotations.

Potential obliquity measures ($\boldsymbol{\theta}$) are coefficient of uniformity (C_u), coefficient of curvature (C_c), or simply percent more oblique than a critical angle (D_θ).

$$C_u = \frac{D_{60}}{D_{10}} \quad \text{or} \quad C_c = \frac{D_{30}^2}{D_{60}D_{10}} \quad \text{where} \quad D_\theta = \text{\% debris of obliquity } \geq \theta \ . \tag{10.5}$$

Above measure are common in geotechnical engineering to describe complex soil fields. These meta-parameters can potentially be applied to our debris-field internal rotation CDFs to form complex trends based on refined internal rotation distributions– not simplified external rotation measures or averages. We have yet to look into trends with these orientation meta-parameters in great detail, although preliminary numerical studies using this approach are taken later in this chapter.

10.2.4 Evolution

It is far easier to characterize debris-fields at rest for a site than during the event, and so we choose to study them this way. An underpinning of our methodology is the evolution of debris-field parameters ($\mathbb{F}$, e.g. measures of orientation and porosity) from a pre-event state to the moment of impact.

$$\mathbb{F} = f\left(\mathbb{F}_{\text{Prior}}\right) \tag{10.6}$$

Similarly, backwards evolution may be used to determine debris-fields at impact using data from disaster reconnaissance (i.e. after the event took place and debris-fields are at-rest).

$$\mathbb{F} = f\left(\mathbb{F}_{\text{Post}}\right) \tag{10.7}$$

The actual evolution of debris-field parameters is not the focus of our project (Naito et al. 2014), but it likely behaves in a stochastic manner. We must assume this relationship is knowable in order to use prior or post information of a debris-field to predict or reverse-engineer behavior during an inundation event (e.g. impact on a building).

10.3 *Dependent Variables*

To refine our analysis of fluid-driven debris-field impacts we must define the dependent variables we **(i)** can measure, **(ii)** want to know, and **(iii)** can discern trends in relative to independent debris-field variables.

A simple set of dependent variables may concern **(i)** an acceleration-based quantity, **(ii)** a velocity-based quantity, and **(iii)** a time-based quantity. Respective to a concerned structure possessing mass, we can rephrase as **(i)** a peak load, **(ii)** momentum transfer, and **(iii)** critical event duration.

10.3.1 Peak Load

For a peak load measure, we choose to evaluate maximum force. Max force ($f_{\max}$) is the maximum force value recorded on a structure of interest during debris-field impacts (or damming). It can be any direction. Stream-wise is our primary concern, but transverse and vertical peak loads are often important for bridges. Current methods for debris-field impacts poorly predict max forces, primarily because it is a second derivative quantity requiring refined analysis to retrieve. Note that filtering schemes applied to resultant force vs. time series greatly affect max force measures. Peak load is dynamically meaningful (giving strong insight into collision stiffness, etc.), but may not manifest in meaningful ways for design engineers. A peak load can last a millisecond in stiff collisions. It can endure a minute in damming scenarios. When engineers want to design for resultant building strains and displacements it may be more useful to have a first order derivative quantity (e.g. velocity-based, momentum transfer) as opposed to a second order derivative quantity (e.g. acceleration-based, peak load) without context of duration.

10.3.2 Momentum Transfer

Momentum transfer can be viewed in a couple of ways. Most simple is the collision's impulse. Impulse (I) is the time integral of force. It allows us to see the total momentum transfer of the collision.

$$I = \int_0^{T_D} f(t)dt \tag{10.8}$$

This measure is reasonably well predicted by current methods that rely on debris count (c), as the number of debris describes the total mass in the debris-field.

A similar measure, Arias intensity (I_A), can be found on a structure in impact scenarios.

$$I_A = \frac{\pi}{2g} \int_0^{T_D} a^2(t)dt \tag{10.9}$$

For a rigid structure this measures the energy resisted. Integrating over the duration of the event, we recieve a CDF of energy imparted to the structure from debris-field impacts.

10.3.3 Critical Event Duration

Arias duration (t_A) measures the time it takes for the normalized Arias intensity (I_a) to go from $I_{A,x}$ to $I_{A,y}$. We elect (x, y) to be $(10, 90)$ respectively, representing the interval of 10 to 90 percent of the arias intensity. Thus the Arias duration (t_A) captures the length of time for 80 percent of the debris-fields collision to impart itself on a structure.

10.4 *In-Air Study*

To expand empirical and theoretical understanding of debris-field impacts using our quantification scheme of independent and dependent variables, described above, we undertake a statistics-based numerical study using our advanced numerical tool (Chapter 5). The studies encompass purely in-air debris-field impacts to remove influence of driving fluids, which is too be included at a later date. Two data-sets are produced for this in-air study: **(i)** Ordered debris-field impacts and **(ii)** disordered debris-field impacts.

10.4.1 Ordered Debris-Fields

Ordered debris-fields (no porosity or variance in debris orientation, $n = 0$ and $\sigma_\theta = 0$, e.g. tightly packed adjacent debris) are evaluated. All debris-fields possess identical material properties and initial velocities. A rigid structure is located in the path of travel, force of impact is measured on its leading face. No gravity or fluid is present. Note that these simulations were performed in a very early state of the ClaymoreUW code, and thus they were done in single-precision.
Angle of impact (θ) is varied to evaluate the effect of rotation on ordered debris-fields without confluence from internal rotation or spacing. Results are intended to determine if rotation (θ) can

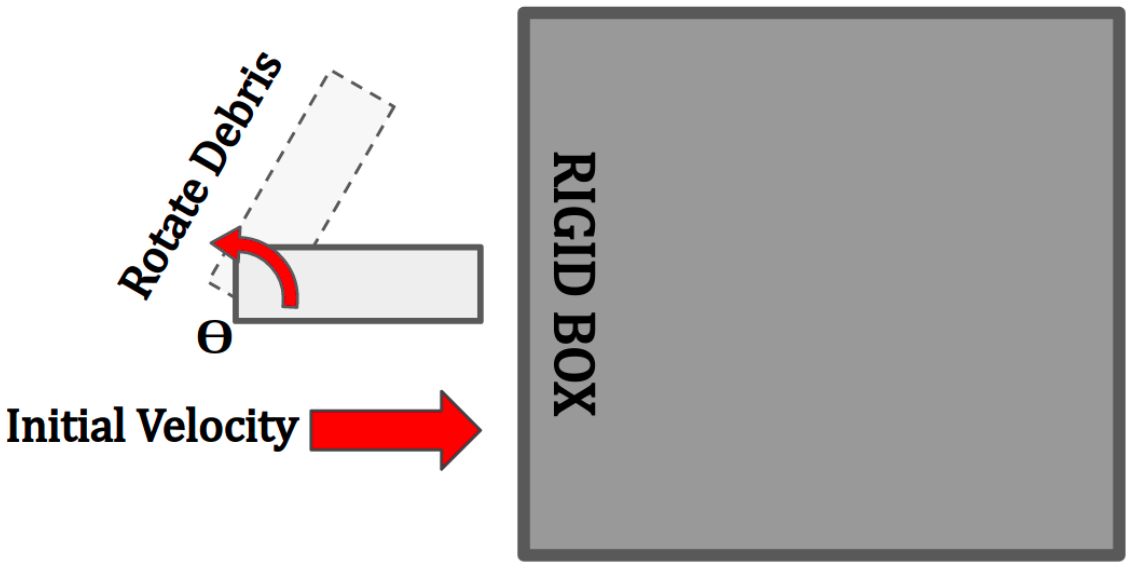

Fig. 10.4. In-Air Single Debris Impact Schematic.

explain heteroscedasticity in debris count (c) for an intensity measure (e.g. max force, $f_{\max}$).

To start we can consider the simplest case: A single piece of debris impacting a structure, with the angle of impact being varied across nine tests. See Figure 10.4 for a schematic. This will mitigate any behavior from multi-debris interaction, allowing a clean look at how debris rotation (θ) effects an intensity measure (IM) like force of impact.

Plotting time-series of ordered debris impacts in Figure 10.5 shows significant effects on both the peaks and shapes of the force-time profiles as angle of impact is varied. This preliminary result is encouraging because it shows a fundamental dependency of an intensity measure on debris rotation, something not commonly considered in modern literature.

Taking only the max force ($f_{\max}$) of each time-series as an intensity measure (IM) of interest we may then compare against debris count (c) as we increase the number of debris in the ordered debris-fields. A schematic for ordered debris-field impacts with varied field incidence angles is shown in Figure 10.6.

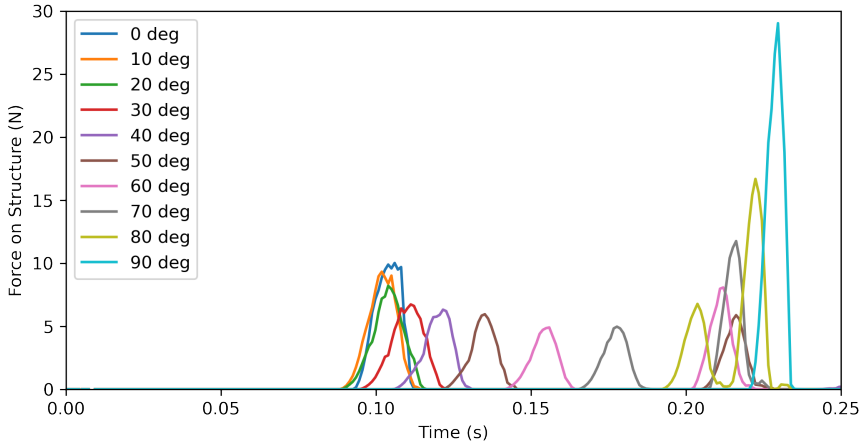

Fig. 10.5. Force of debris impact relative to impact angle. Force-time series on structural face when varying the angle of impact of a single piece of debris.

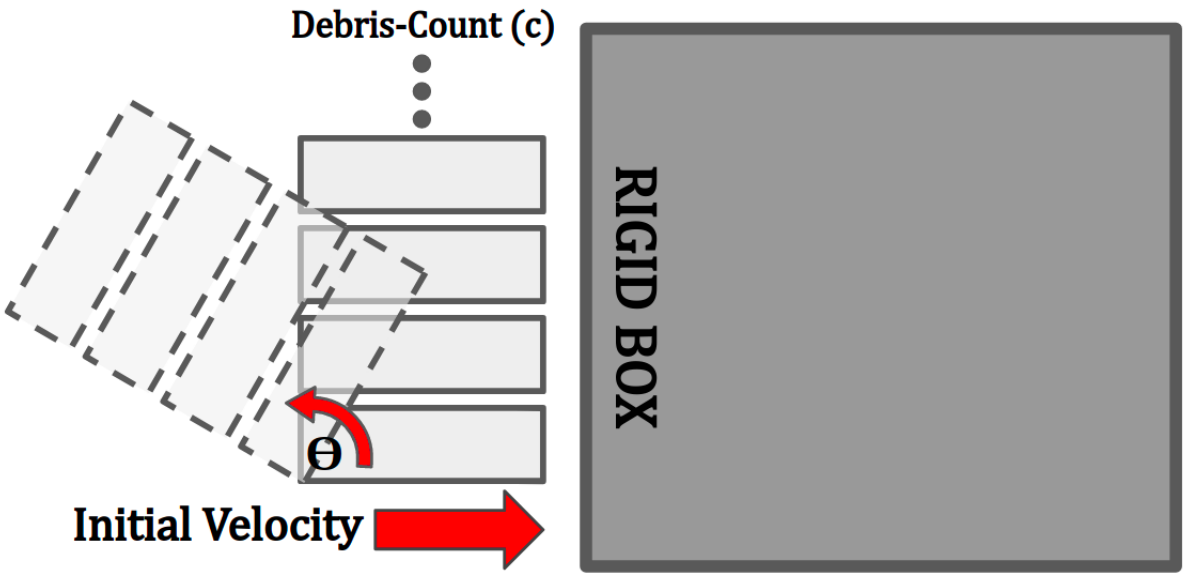

Fig. 10.6. In-Air Ordered Debris Impact Schematic.

In Figure 10.7 it appears that variance, grows as debris-count (c) increase. However, in Figure 10.8 it is seen that heteroscedasticity in debris-count is a trend from rotation of the ordered

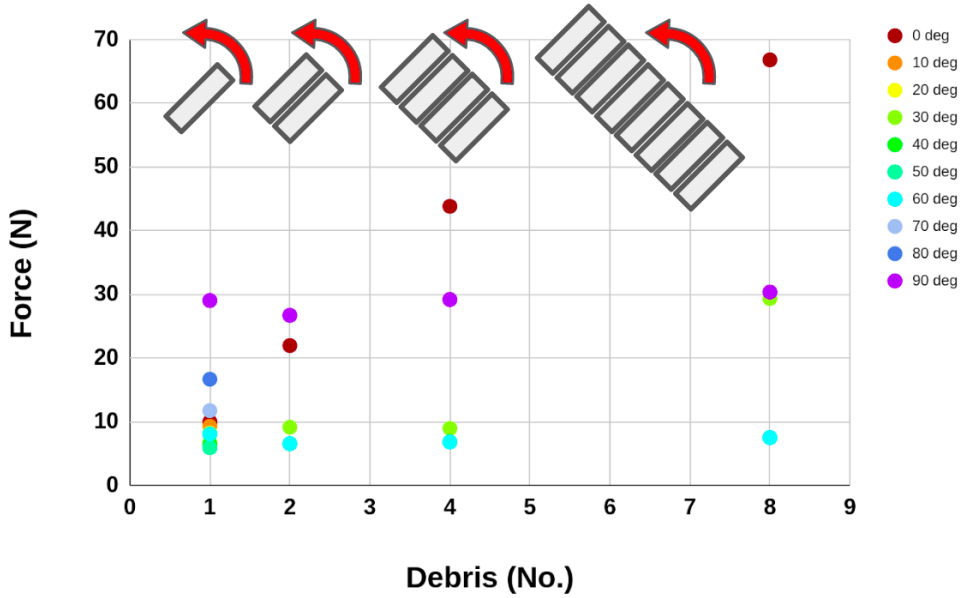

Fig. 10.7. Max scatter bar plot of in-air ordered debris-array impacts as count and rotation vary. Scatter plot with evident heteroscedasticity in max force over debris count (c) as rotation varies.

debris-fields. The trend appears to be defined by a 3-point curve with each end defined by a head-on collision mode and (e.g. 0 or 90 degrees) and a mid-point where max force is the smallest found to be a consistent value for all debris-counts around 45 degrees. This suggests that the two primary ordered modes, longitudinal and transverse, are the most structurally hazardous, while random configuration in between the two modes are less notable. This is supported by numerical and experimental results in Chapter 7.

10.4.2 Disordered Debris-Fields

Disordered debris-fields (varying c, n, θ, σ_θ, Figure 10.9) are created. Measures (c, n, θ, L, (x, y)) were randomly sampled from uniform distributions.

Sampled values were used to generate Finite Element meshes that are compatible with our numerical software, shown in Figure 10.10 where a high-level schematic of the debris-field

generation (left) is used to develop a set of finite element meshes for the debris-field (right).

Meta-data is also generated per randomly sampled debris-field. Rotation of each debris is tracked and can be arranged on a rose plot. Rose plots can be converted into empirical histograms and then empirical CDFs (10.3). From the CDFs we retrieve characteristic values that either define aspects of the curve at-large or the proportion of retained debris by some rotation (e.g. θ_{15}, the probability that a debris in the given field is at an angle less than 15 degrees to the impact structure). These scalar values allow for simplistic regression analysis over a measure of rotation without use of the full empirical curves.

The pass-fail nature of the debris-field generating script fundamentally alters what samples make it to the next stage, thus it is difficult to consider the samples random. Geometric constraints impose fundamental dependency between "randomly" sampled independent variables. Though not ideal for many statistical tests, this mirrors debris-fields in reality. Dependency in

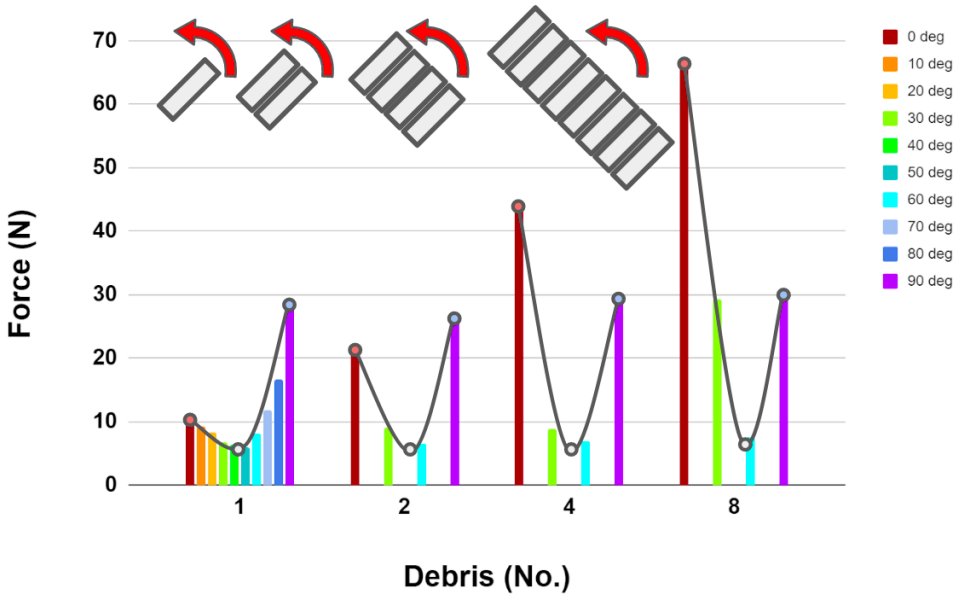

Fig. 10.8. Max force bar plot of in-air ordered debris-array impacts as count and rotation vary. Bar plot that suggests heteroscedasticity is a trend in debris-field rotation.

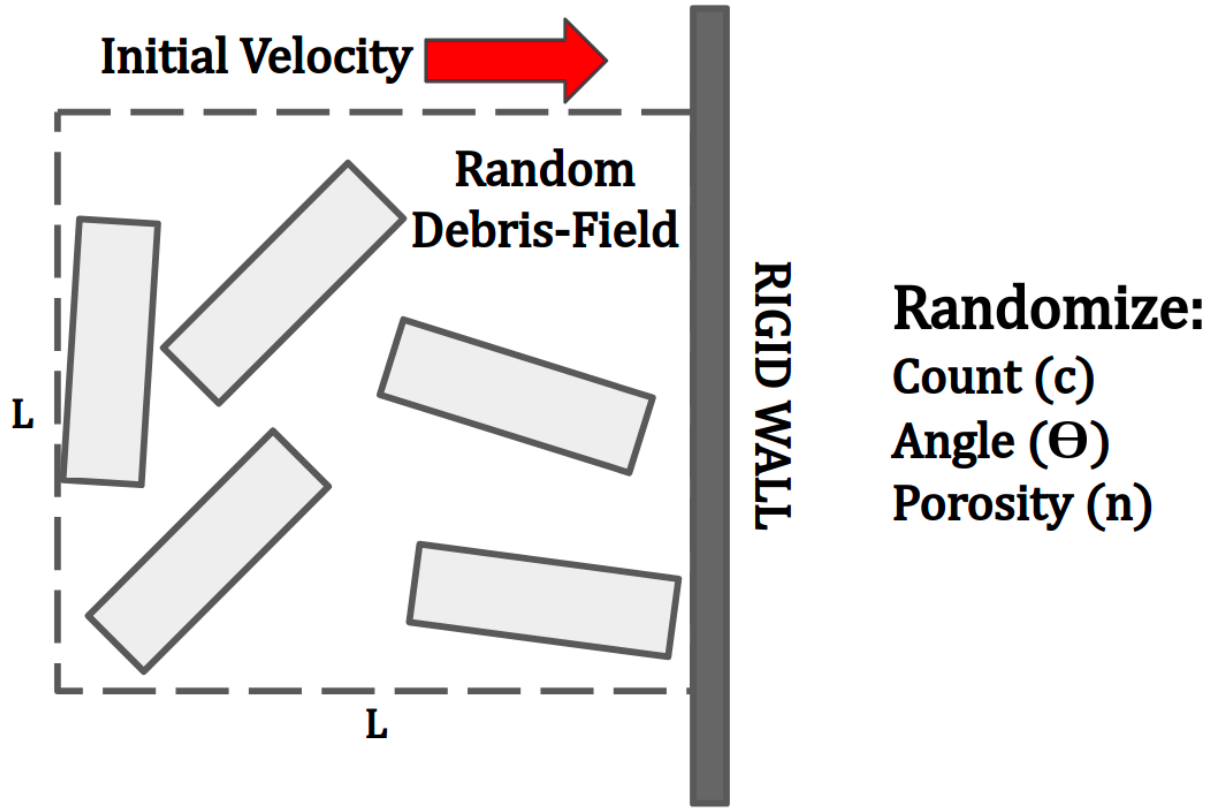

Fig. 10.9. In-Air Disordered Debris-Field Impacts Schematic.

"independent" variables also suggests that it may not be necessary to possess all independent variables for design, rather a "pick two out of three" principal could be adopted as done for basic regression models in Chapter 7.

The second data-set, disordered debris-fields, is visualized for three dependent variables of interest (max force, impulse, and arias time) across 100 simulations.

Simulation dependent variable results (Fig. 10.11) shown signs of potential causative trends, though it is littered with far more likely correlative features and thus not ideally posed for statistical analysis. In response, we pull three scalar intensity measures (IM, i.e. max force, impulse, and arias duration) from each simulation. All gathered independent and dependent variables may then be plotted against one another (Fig. 10.12) for a precursory look at data

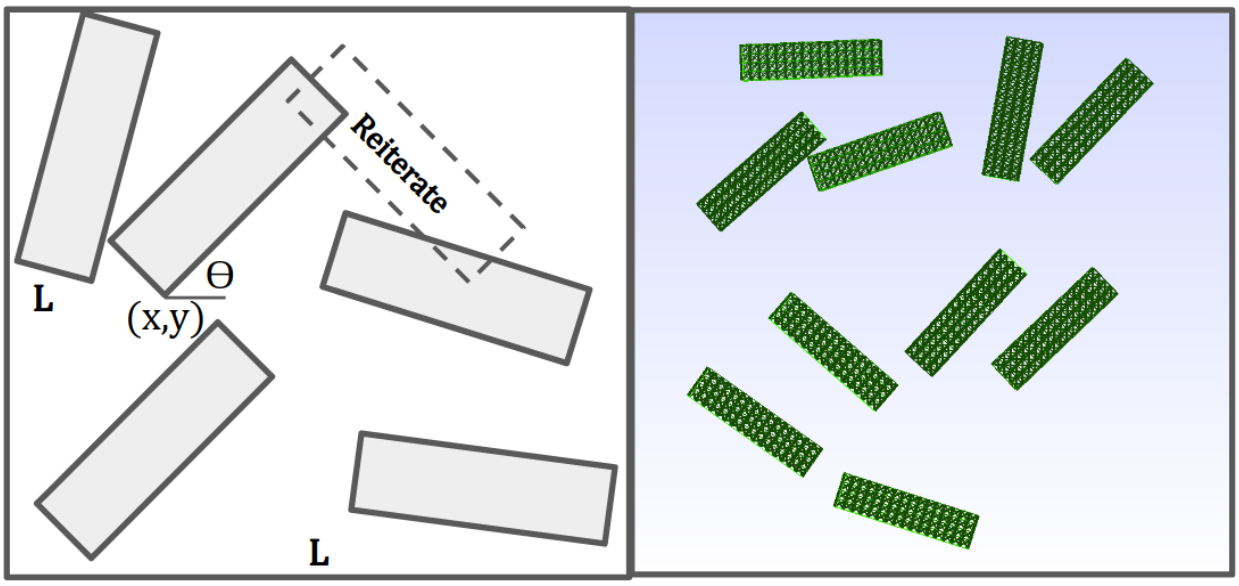

Fig. 10.10. Algorithm for creating stochastic debris-fields numerically. (Left) Diagram of debris-field generation algorithm. (Right) Example finite element mesh generated by algorithm.

distribution and correlation. Note that some variables are transformations of others for the purpose of improving distribution and correlation.

Histograms visually suggest lack of normality in many variables and correlation plots show, qualitatively, very strong and weak relationships between certain variables. Considering these points, the independents variables were pruned down to just three to represent debris count (c), occupation ($\log n$), and orientation ($\sqrt{\theta_{15}}$) in further analysis. Dependent variables were all maintained, though max force ($f_{\max}$) will be the primary focus.

Linear regression was performed on a correlation matrix plot (Figure 10.13). Many trends are observed, though some are seemingly influenced by outliers. Future work should seek to remove outlier points or choose variables that do not feature governing aberrations.

Three regression models are then developed for consideration. We aim to compare them to see what variables, and variables pairings, actually matter for force prediction. The first is a linear model considering only debris count (c, i.e. what current literature looks at). The second is a dual

linear model which considers a measure of rotation and spacing ($\sqrt{\theta_{15}}, \log n$), but not debris count (c). The third is a multi linear model which considers debris count, rotation, and spacing ($c, \sqrt{\theta_{15}}, \log n$).

Predictions are cast against simulation observations in Figures 10.14, 10.15, 10.16, 10.17, and 10.18. The linear model gives a decent prediction but does not feature any variance seen in observational data. The dual linear model doesn't give a very good prediction but does have the variance seen in the observational data. The multi linear model gives both good predictions and variance expressions. Qualitatively, the multi linear model is superior.

To quantitatively compare model variance, specifically for heteroscedasticity analysis (i.e. does variance of the dependent variables increase as the independent variable grows), we look at the residuals of each model in Figures 10.19, 10.20, 10.21, 10.22, and 10.23. It is difficult to interpret the scatter plot, but the histograms show seemingly normal data which has a tighter range of residuals for the multi linear model. Normality of residual distributions per regression model checked. This controls our subsequent choice of test for heteroscedasticity. Here we use the Shapiro test in Table 10.4.2.

Variable	Shapiro Stat.	Shapiro p-value	Normal?
Linear	0.952	0.141	YES
Dual	0.964	0.317	YES
Multi	0.934	0.04	NO

White's Test in Table 10.4.2 is used to evaluate heteroscedasticity in the regression models residuals. Breusch-Pagan's Test would be preferable but lack of residual normality (see prior Shapiro test) suggests use of the White's test. A visual test is not applicable as the data-set is too confusing for visual quantification.

Model	Exog. Variable	White's Stat.	White's p-value	Hetero.?	F-Test Stat.	F-Test p-value
Linear	c	1.4	0.497	NO	0.666	0.521
	log(n)	2.379	0.304	NO	1.166	0.686
	sqrt(D15)	4.081	0.13	YES	2.114	0.138
Dual	c	3.519	0.172	YES	1.79	0.184
	log(n)	0.816	0.665	NO	0.381	0.686
	sqrt(D15)	4.459	0.108	YES	2.34	0.113
Multi	c	0.266	0.876	NO	0.122	0.886
	log(n)	1.629	0.443	NO	0.78	0.467
	sqrt(D15)	0.577	0.75	NO	0.267	0.767

We find that while heteroscedasticity is detected by White's test for the linear and dual regression model for certain exogenous variables, no heteroscedasticity is found for the multi linear regression model. This can be interpreted as the introduction of curated measures serving to reduce heteroscedasticity in the regression model. More simply, the "chaos" of debris-field impacts is actually from variables not looked at in the current literature.

10.5 *Final Remarks*

This section reveals debris-fields as quantitative bodies. By exploring characteristics that describe debris-fields (e.g. debris count, occupation, orientation) we found independent variables for analysis. By considering basic needs of engineers designing for debris-field impacts we found dependent variables (e.g. impulse, max force, impact duration). Statistical analysis revealed relationships between knowable independent variables and desired dependent variables in the context of an in-air numerical study. In the next section we will mature these insights into a novel framework of probabilistic design.

Note that simulations undertaken in these in-air debris-field impact pilot studies were done in an earlier state of the ClaymoreUW code, we have intentions to repeat the studies for far more cases

(1000+) at higher-resolutions and with better accuracy owing to our improved code.

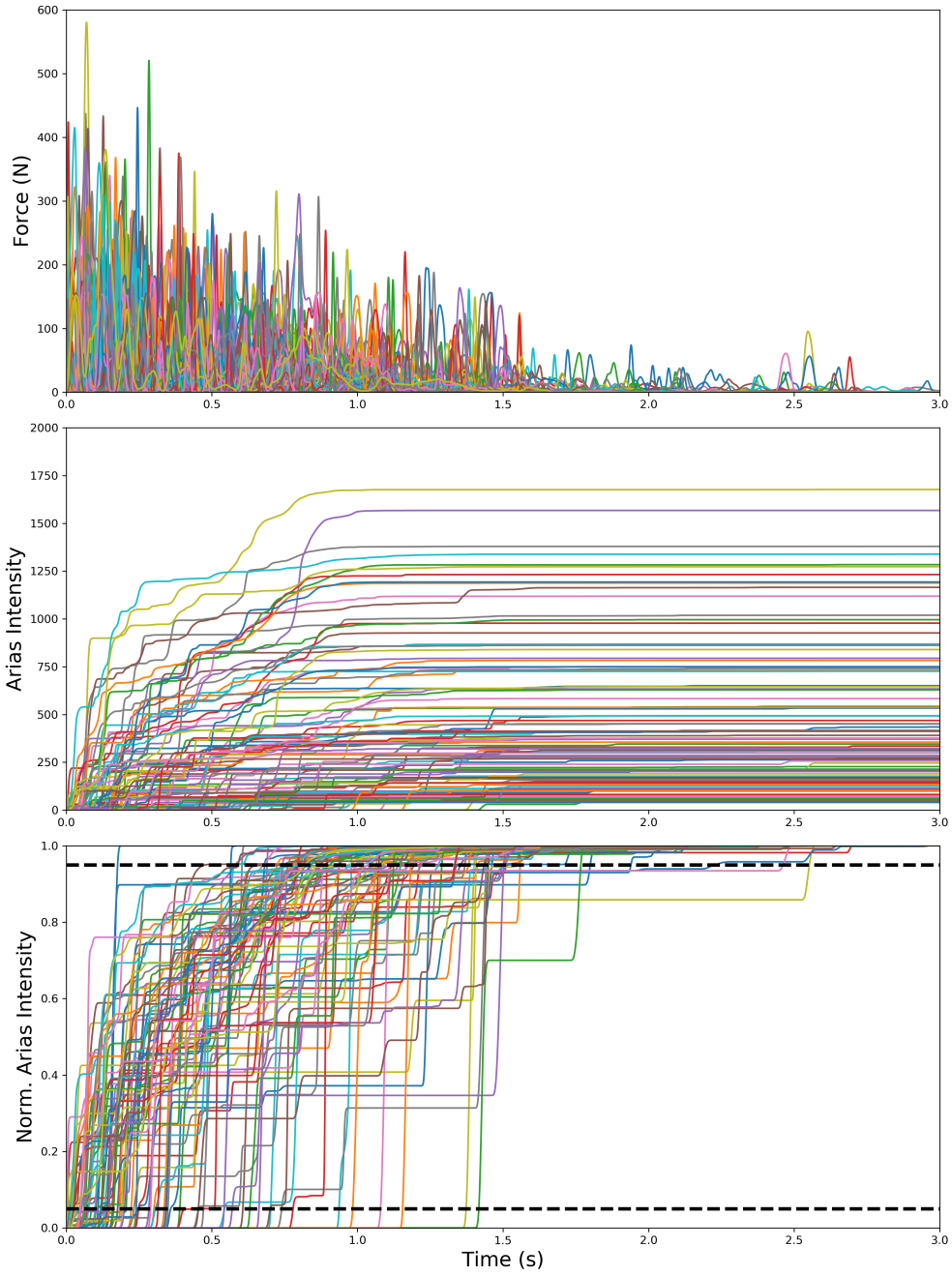

Fig. 10.11. Dependent variables from randomized in-air debris-field impacts. (Top) Force-time series from disordered debris-field simulations. (Middle) Cumulative impulse from disordered debris-field simulations. (Bottom) Arias intensity from disordered debris-field simulations, with dotted brackets defining the range from 10% to 90% intensity imparted.

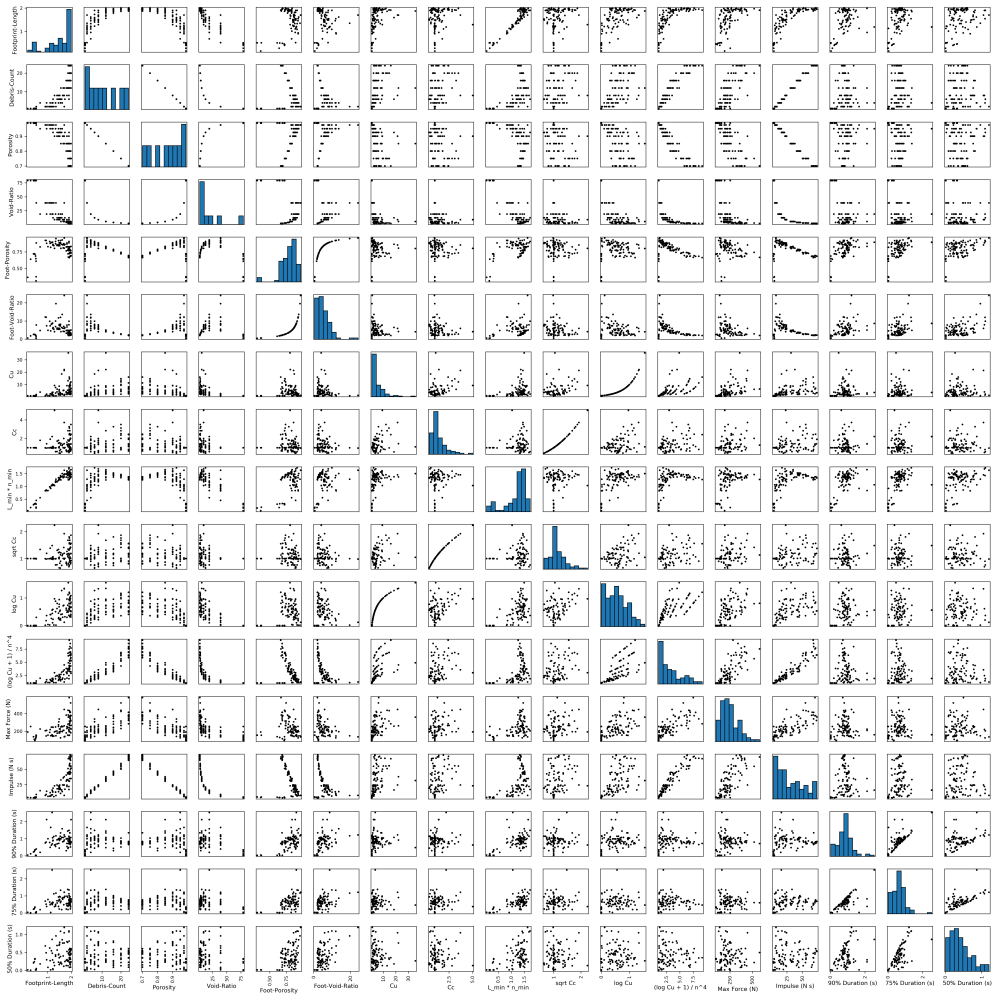

Fig. 10.12. In-air debris-field impact correlation plot. Correlation plot of all independent and dependent variables considered in disordered debris-field simulations.

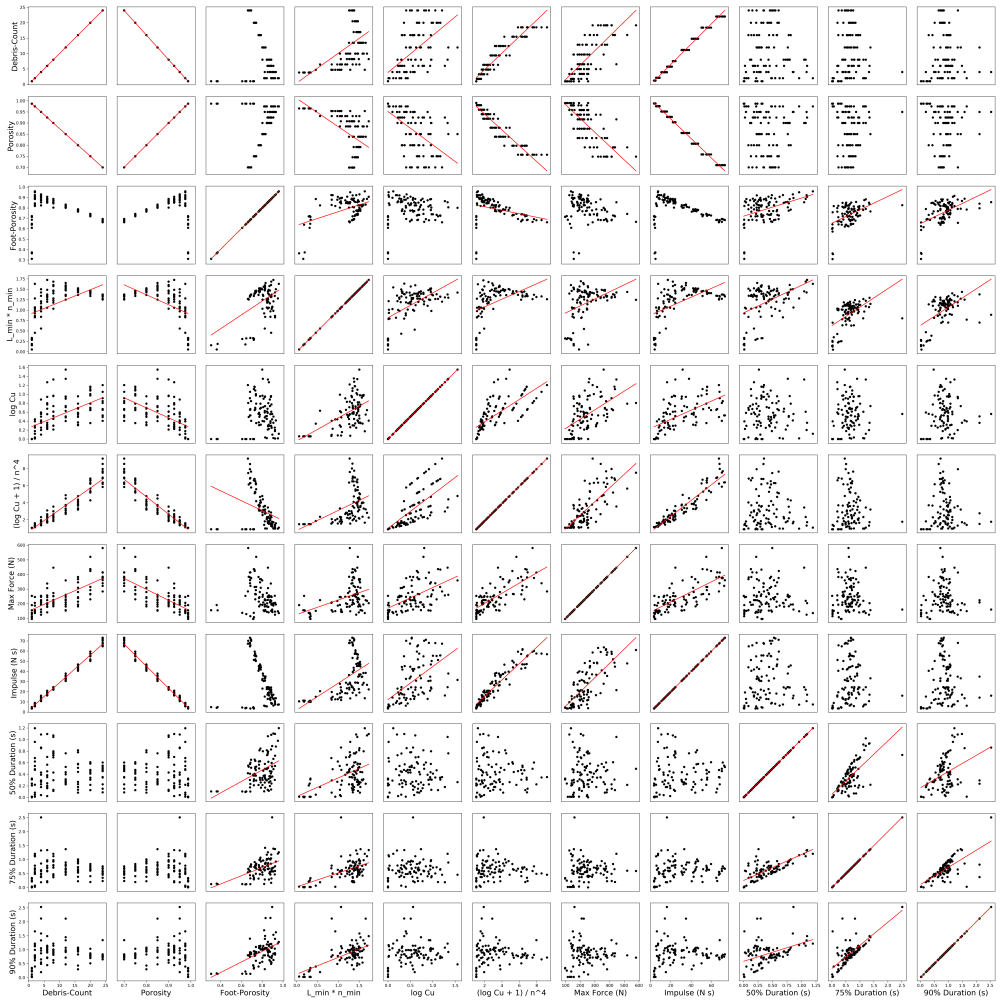

Fig. 10.13. In-air debris-field impact linear regression plot. Linear regression correlation plot of curated independent and dependent variables for disordered-debris-field impact simulations.

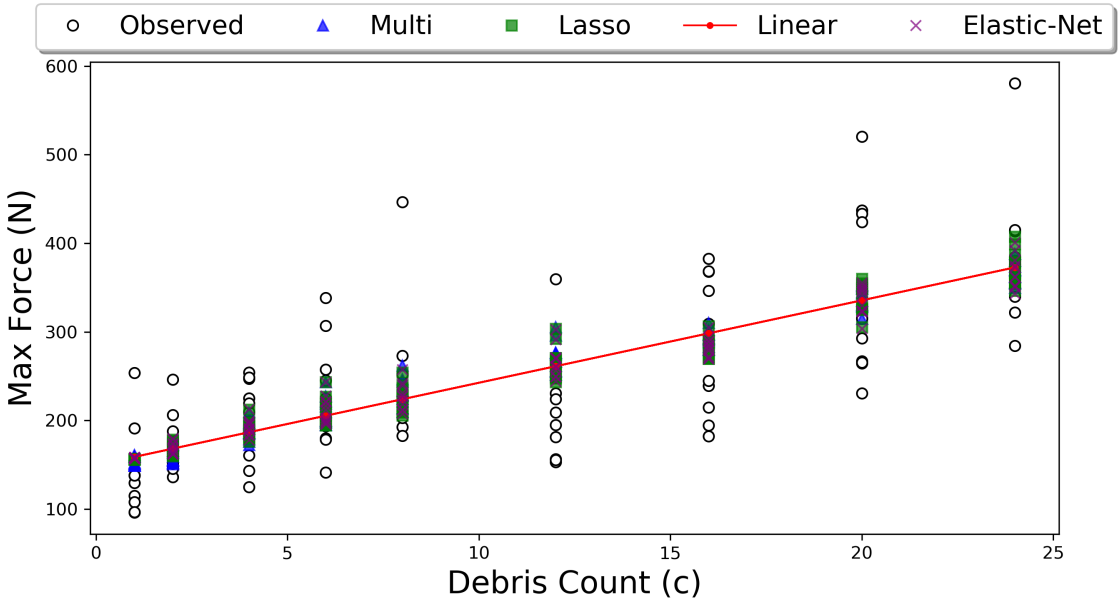

Fig. 10.14. Regression models and numerical observations for in-air debris-field impacts. Regression model predictions compared with simulation observations.

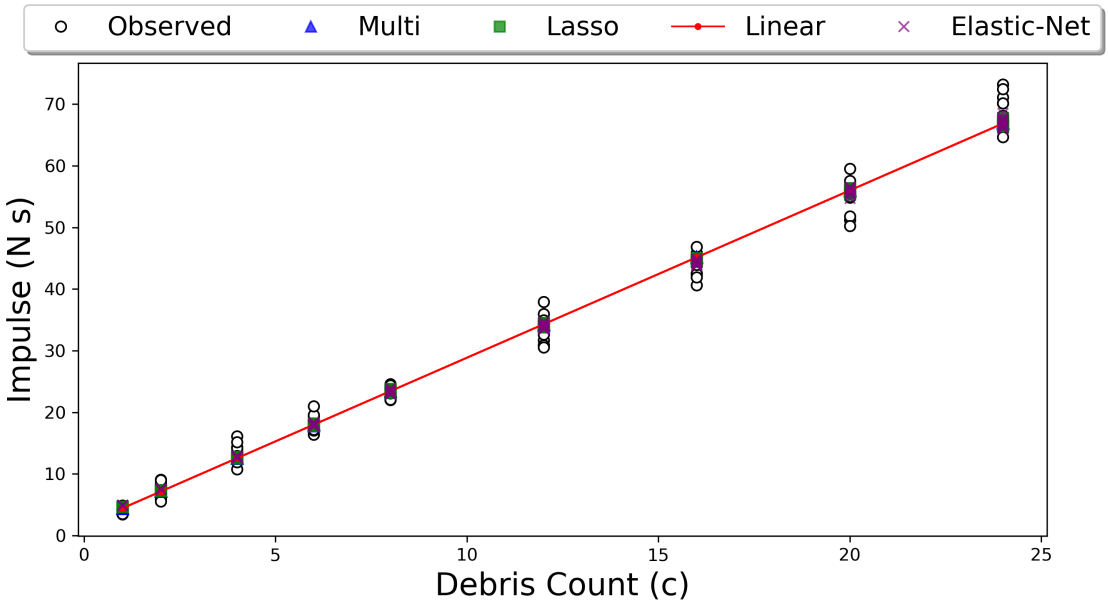

Fig. 10.15. Regression models and numerical observations for in-air debris-field impacts. Regression model predictions compared with simulation observations.

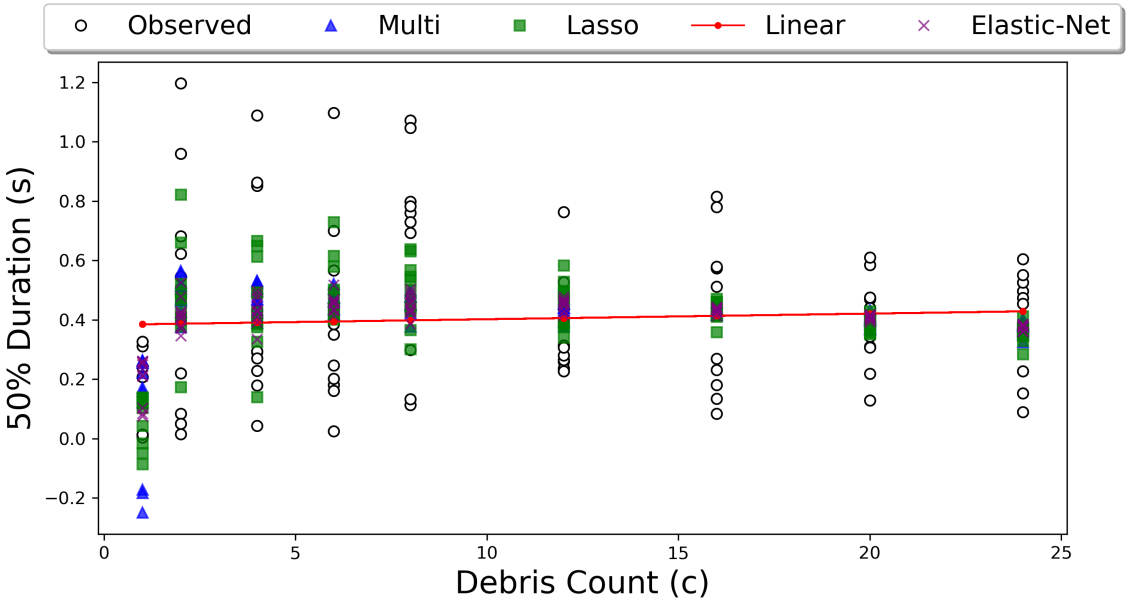

Fig. 10.16. Regression models and numerical observations for in-air debris-field impacts. Regression model predictions compared with simulation observations.

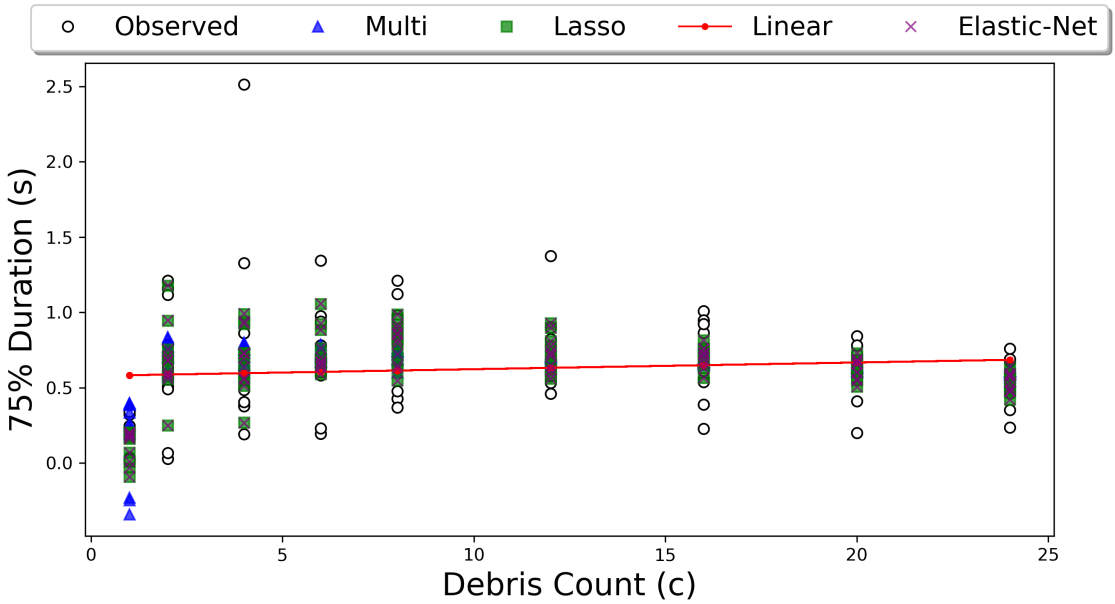

Fig. 10.17. Regression models and numerical observations for in-air debris-field impacts. Regression model predictions compared with simulation observations.

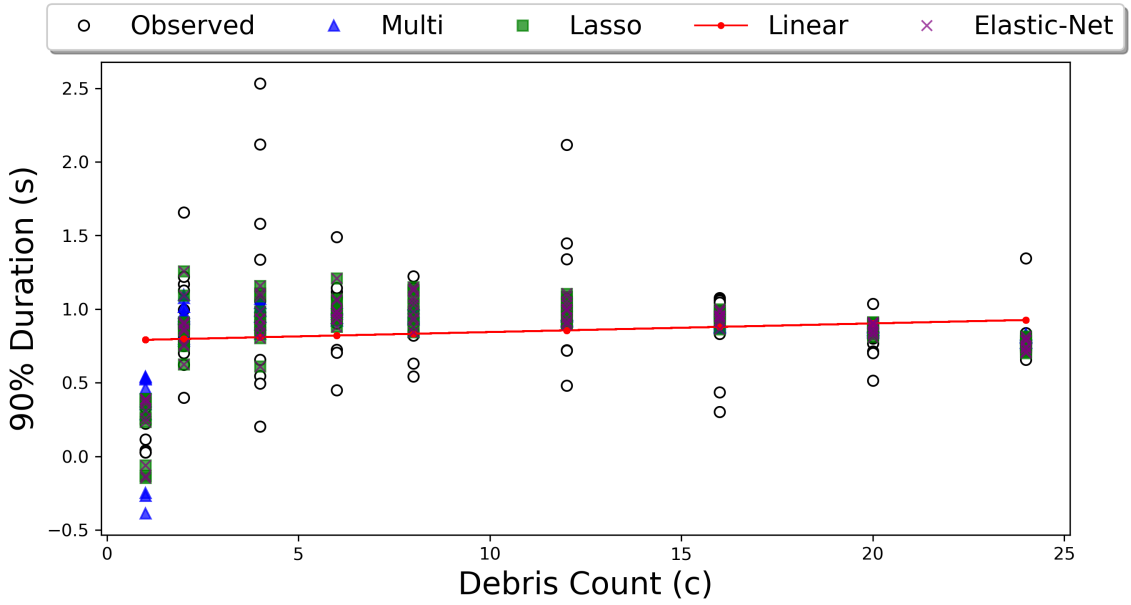

Fig. 10.18. Regression models and numerical observations for in-air debris-field impacts. Regression model predictions compared with simulation observations.

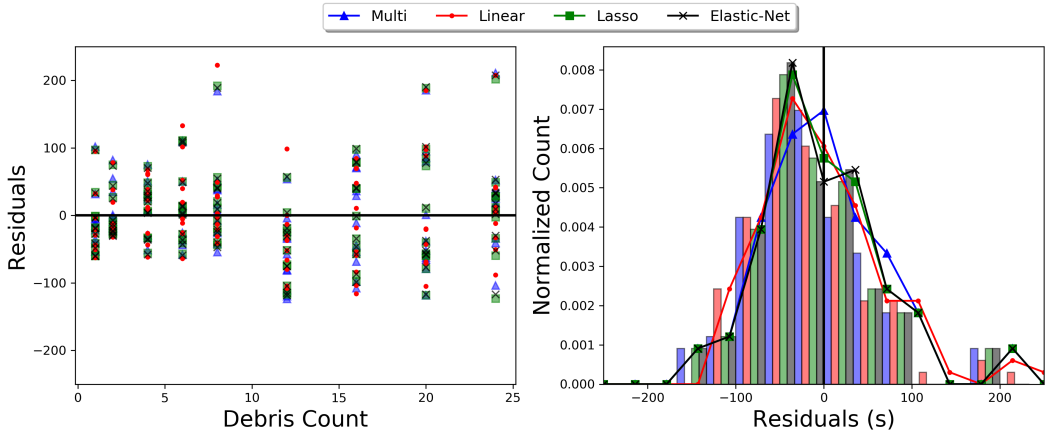

Fig. 10.19. Regression models and numerical observations for in-air debris-field impacts. Regression model predictions compared with simulation observations.

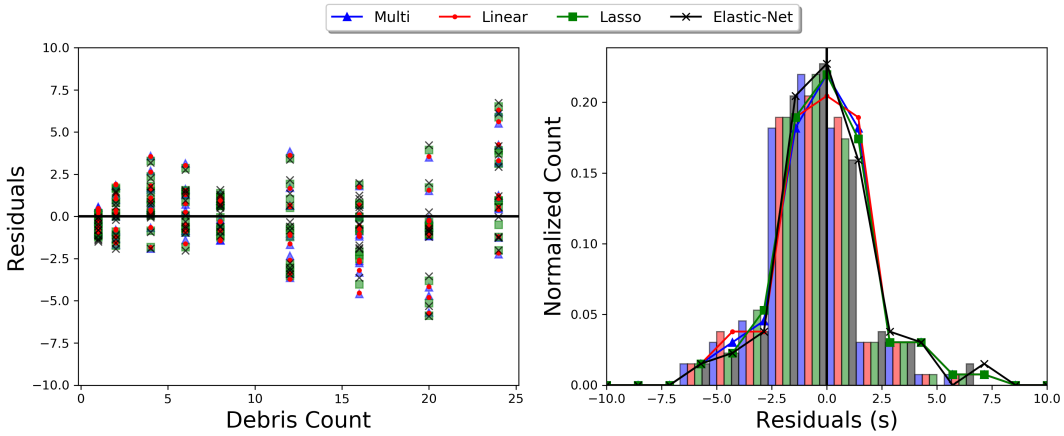

Fig. 10.20. Regression models and numerical observations for in-air debris-field impacts. Regression model predictions compared with simulation observations.

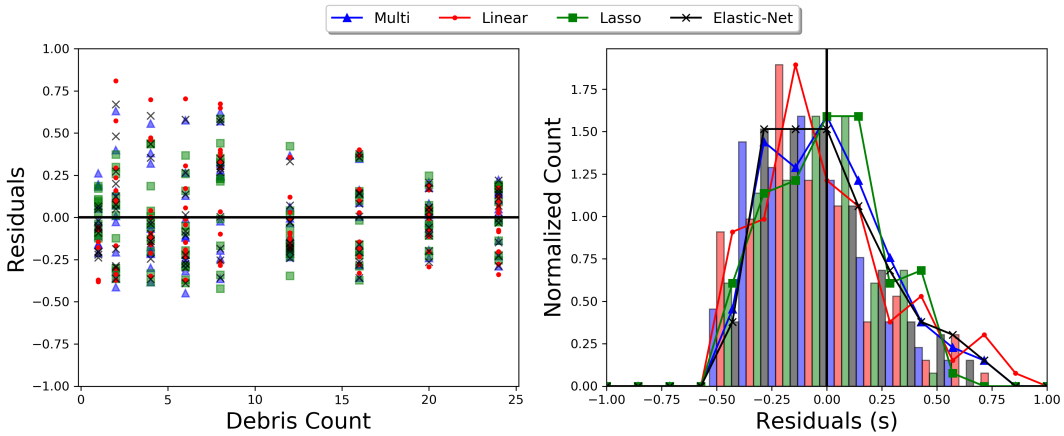

Fig. 10.21. Regression models and numerical observations for in-air debris-field impacts. Regression model predictions compared with simulation observations.

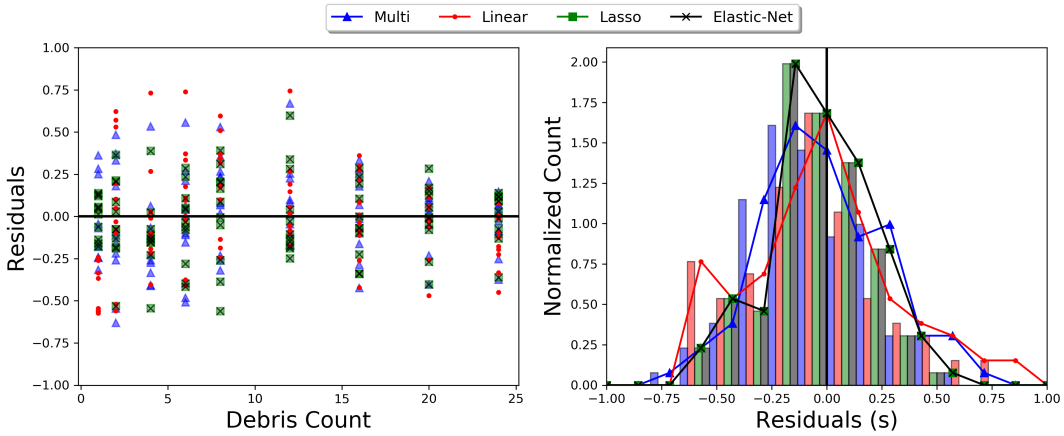

Fig. 10.22. Regression models and numerical observations for in-air debris-field impacts. Regression model predictions compared with simulation observations.

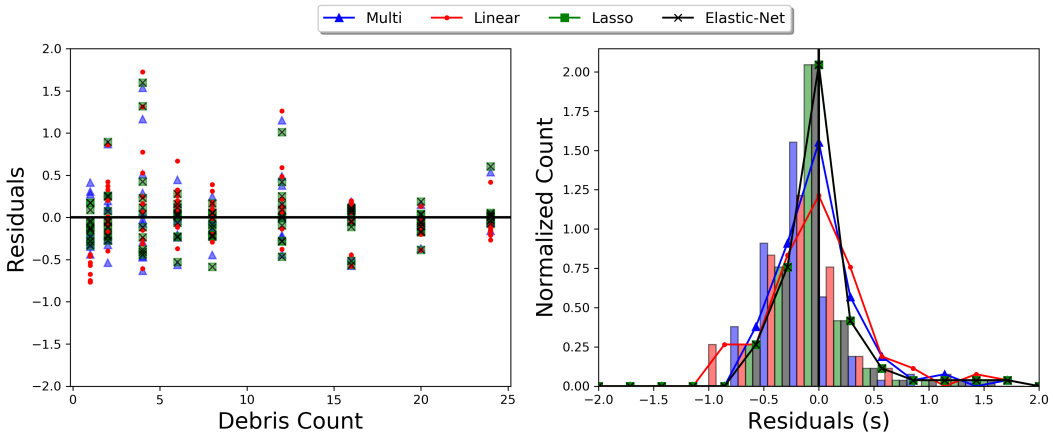

Fig. 10.23. Regression models and numerical observations for in-air debris-field impacts. Regression model predictions compared with simulation observations.

Chapter 11

PERFORMANCE-BASED FRAMEWORK

11.1 *Introduction*

The hazard of fluid-driven debris-field impacts is phrased within a Performance-Based Engineering (PBE) framework in this chapter. This allows it to be approach with probability and inline with modern coastal engineering. Further, steps to merge this framework with Performance-Based Tsunami Engineering (PBTE) are considered. Note that this chapter has been adapted and refined from work published in the COPRI PORTS' 22 conference proceedings.

11.2 *Performance-Based Debris-Field Engineering*

Great uncertainty exists in structural design for tsunami-driven debris-field hazards. In the light of hundreds of experimental and simulated tests, the authors believe a framework for probabilistic behavior is possible. Initial steps, limitations, and insights are discussed here.
Performance Based Engineering (PBE) is a clean, probabilistic hazard workflow (Moehle and Deierlein 2004). Our debris-field framework is an add-on to existing PBE frameworks, specifically Performance Based Tsunami Engineering (PBTE), and is coupled similar to other combined hazard studies (Attary et al. 2019). Guided by experimental and numerical results, we take the following steps:

1. Define debris-field parameters of novel significance, $\mathbb{F}$, $p(\mathbb{F})$.

2. Declare debris-field intensity measures, IM.

3. Determine debris-field intensity measures given prior conditions, $p(IM|O, D, \mathbb{F})$.

4. Connect engineering demand parameters to debris-field intensity measures, $p(EDP|IM)$.

To start, determine the initial conditions for analysis. This includes location (O) and design (D), as standard. Establish another parameter set ($\mathbb{F}$) for the debris-field. Parameters are measures of: **(i)** count, **(ii)** obliquity, **(iii)** occupation, and **(iv)** time-dependency. These independent variables must be measurable, minimal, scale-invariant, and predictive of engineering loads (i.e. dependent variables). A set of debris-field independent variables ($\mathbb{F}$) is stated.

$$\mathbb{F} = \{\mathbb{D}, \mathbf{c}, \mathbf{n}, \boldsymbol{\theta}\} \quad \text{and} \quad p(\mathbb{F}) = \{\mathbb{F}, T\} \tag{11.1}$$

Where $\mathbf{c}$, $\mathbf{n}$, and $\boldsymbol{\theta}$ are placeholder measures of debris count, occupation, and orientation. $\mathbb{D}$ is inherited from current single-debris design (Eq. 10.2) and T is time-periods that preceding variables act respective of (e.g. daily, monthly, seasonally). Distinction of debris-fields vs. debris ($\mathbb{F}$ vs. $\mathbb{D}$) is important because unique hazard loading results from different debris-field configurations (varied $\mathbb{F}$), even for consistent debris properties ($\mathbb{D}$) (Fig. **??**).

Obliquity (θ) of single-debris impacts affect structural demands, such as max force (Nistor et al. 2017a). However, debris-field obliquity is a distribution ($P(\theta)$), not a single-value. Drawing from geotechnical soil sieve charts, obliquity distributions are characterized by simple meta-parameters to describe the continuous curve, not the discrete points. Potential obliquity measures ($\boldsymbol{\theta}$) are coefficient of uniformity (C_u), coefficient of curvature (C_c), or simply percent more oblique than a critical angle (D_θ).

$$C_\mathrm{u} = \frac{D_{60}}{D_{10}} \quad \text{or} \quad C_\mathrm{c} = \frac{D_{30}^2}{D_{60} D_{10}} \quad \text{where} \quad D_\theta = \text{ \% debris of obliquity } \geq \theta \tag{11.2}$$

Occupation of a debris-field ($\mathbf{n}$) and spatial uncertainty in the measure ($P(\mathbf{n})$) influences impact

Fig. 11.1. Parking Lot Debris. (Upper-Left) Ordered parking lot. (Upper-Right) Angled parking lot. (Lower-Left) Sparse parking lot. (Lower-Right) Parking lot evolved by 2011 Tohoku tsunami. Courtesy of NHK World TV.

demands, i.e. for max force (less "dense" fields typically enact smaller max loads). Potential occupation measures (**n**) are porosity (n), void-ratio (e), and spacing (s), where A_W is the water occupied area contained in A_T which is the total area, i.e. "foot-print", of a debris-field.

$$n = \frac{A_W}{A_T} \quad \text{or} \quad e = \frac{n}{n-1} \quad \text{or} \quad s = \text{Avg. debris spacing.} \tag{11.3}$$

Note that **c**, **n**, and $\boldsymbol{\theta}$ for any fixed footprint (e.g. 10m x 25m) are correlated by geometric constraint (i.e. not independent). Multi-linear regression technically requires independent variables, so other regression types (e.g. RIDGE, LASSO, or Elastic Net) might be preferred for significant statistical models. Using just two variables could also be viable as a result of correlation constraining the third, greatly simplifying engineering design. In our studies we

observe that **c** and either **n** or $\boldsymbol{\theta}$ is often adequate for load predictions.

At-rest debris-field measures ($\mathbb{F}_{\text{Rest}}$) are definable with a site-survey, whereas at-impact values ($\mathbb{F}$) aren't known. We claim evolution of debris-field parameters at-rest to the moment of impact is predictable. Similarly, backwards evolution may apply to post event measures ($\mathbb{F}_{\text{Post}}$) from disaster reconnaissance. Field evolution equations are not our focus, see other studies for examples (Naito et al. 2014; Goseberg et al. 2016; Nistor et al. 2017b; Stolle et al. 2016; Stolle et al. 2020), but relationships must exist to use prior or post information to quantify the debris-field at impact:

$$\mathbb{F} = f\left(\mathbb{F}_{\text{Rest}}, O, D\right) \quad \text{and} \quad \mathbb{F} = f\left(\mathbb{F}_{\text{Post}}, O, D\right) \tag{11.4}$$

Time-dependency (T) phrases debris-field intensity measure (IM) as a probability of exceedence ($p(IM)$). Engineers often design by time-ranges (e.g. 100-year tsunamis) over the worst-case scenario, which can be expensive and over-conservative. 100-year tsunami design should not assume mobilization of the worst-case debris-field, but perhaps a 1-month debris-field. 100-years vs 1-month is arbitrary, but a debris-field's design time-period should be shorter because humans actively alter debris-fields (e.g. removing parking-lots, building ports, logging forests) so a long forecast is not viable, unlike tsunami prediction.

Now determine intensity measures (IM) relative to prior conditions, $p(IM|O, D, F)$. The IM for tsunami-driven debris-fields are partially shared with tsunamis (e.g. flow-velocity, flow-depth) but another measure differentiates. We postulate geometric "order" (from **n** and $\boldsymbol{\theta}$) of debris-fields as a time-dependent aspect of the hazard's IM. It is an IM unique to the debris-field, not the debris or tsunami.

Finally, recap preceding claims as axioms, which in turn give a framework to predict engineering demand parameters (EDP) from debris-field intensity measures ($p(EDP|IM)$). Axioms relate initial debris-field properties to probabilistic structural impact demands via a mix of

first-principals and observations in experiments and simulations. Axioms of debris-field impacts, not damming, are stated:

1. **Debris-fields are continuous, probabilistic distributions.**. Distributions of measures ($\mathbb{F}$)– No longer a summation of discrete debris. This view assists where discrete measures (e.g. debris count c) are uncertain, fields are heterogeneous (i.e. mixed debris), and spatial variance is significant.

2. **Geometric order of debris-fields is an intensity measure.** Using "order" (from occupation and orientation) to derive demands (i.e. $p(EDP|IM)$) parses geometric hazard contribution. Perfect order means zero porosity ($n = 0$, tightly packed) and no internal obliquity ($\theta = 0$, debris aligned), the most hazardous case of impact for a fixed number of debris (c). Ordered hazard is set by $C(\mathbf{c})$, experimentally studied in our ordered debris-array impacts.

3. **Deviation from order describes impact hazard reduction.** Allows existing research on ordered debris-arrays to be applied to disordered-fields. Disorder is geometric, i.e. occupation ($\mathbf{n}$) and orientation ($\boldsymbol{\theta}$) relative to perfect order. Worst-case is set by ordered behavior ($C(\mathbf{c})$) and reduced by equations that punish deviation ($R(\boldsymbol{\theta})$, $R(\mathbf{n})$). Intuitively, a lot sparsely occupied by cars, parked at random angles, is less impact hazardous than the same cars tightly-packed side-by-side.

4. **Debris-fields are time-dependent (hourly, daily, seasonally).** Considered for at least one parameter (e.g. occupation exceedance as $\lambda(\mathbf{n}) = f(\mathbf{n}, T)$), our intensity measure is a probability of exceedence ($p(IM)$) over periods (T). Take a near-shore parking-lot, where count ($\mathbf{c}$) and occupation ($\mathbf{n}$) of cars is time-dependent (e.g. more cars midday than midnight, in tourist season than not). Debris-field hazards vary predictably with time.

5. **Debris-field evolution, pre-event to impact, is definable.** Evolves at-rest debris-field measurements ($\mathbb{F}_{\text{Rest}}$) for the event ($\mathbb{F}$). A parking-lot can be initially surveyed (car count, occupation, and obliquity, i.e. semi-consistently spaced cars parked roughly parallel with lot lines) and then evolution over the distance to the structure gives the probabilistic configuration at impact ($\mathbb{F}$, $p(\mathbb{F})$).

Carrying axioms to conclusion, set a potential relationship between stochastic debris-fields and an engineering demand parameter ($\mathbf{f}_{\text{Field}}$, max impact force due to debris-field). Probability of exceedence is available ($p(\mathbf{f}_{\text{Field}})$) for performance-based design.

$$\mathbf{f}_{\text{Field}} = f(F, ...) = C(c) \int_0^{\frac{\pi}{2}} (R(\theta)P(\theta))d\theta \int_0^1 (R(n)P(n))dn \tag{11.5}$$

$$p(\mathbf{f}_{\text{Field}}) = f(F, ..., T) = C(c) \int_0^{\frac{\pi}{2}} (R(\theta)P(\theta))d\theta \int_0^1 (R(n)\lambda(n))dn \tag{11.6}$$

These axioms can give many approaches, this is one example written analytically. Ordered-field behavior ($C(\bullet)$) and disordered-field reductions ($R(\bullet)$) can be regression models of experimental, numerical, and case-study observations. Probability distributions ($P(\bullet)$) and distributions respective to time-periods ($\lambda(\bullet)$) could be from site-surveys or set by tables for common cases (e.g. typical parking lots for commercial businesses).

This framework is effective for simulated in-air debris-field impacts and its trends are reproduced in our fluid-driven experimental trials. For a real tsunami-driven event, a coupled approach is taken with PBTE. The IM should include debris-field "order", as above, along with tsunami measures (e.g. flow-depth and flow-velocity). As flow-depth and flow-velocity influence debris-field hazards (mobilization and debris-velocity respectively), coupling is natural. Our framework geometrically quantifies debris-fields on time-ranges, but it is just a small added step in a larger PBTE endeavor.

Many critical aspects will require intensive research (experimental, numerical, and via case-studies) to flesh-out and validate for peace-of-mind in design. Still, our step-by-step approach, starting at debris-field description and ending in a probabilistic and time-dependent way to quantify a coupled hazard, shows novel promise.

11.3 *Remarks*

Here we take the hazard of debris-field impacts into a form compatible with traditional Performance-Based Engineering. Trends between independent and dependent variables in the previous section are rephrased to fit PBE categorization. Theory for debris-field behavior is stated axiomatically to give a backbone to numerically produced and statistically derived trends. As debris-fields hazards coincide with flow-events (e.g. tsunamis), thought was given to the means of coupling with traditional Performance-Based Tsunami Engineering (PBTE) frameworks. This represents an initial step into a theory-backed, statistically derived, and performance based framework for the cascading hazard of tsunami-driven debris-field hazards.

Chapter 12

CONCLUSIONS AND FUTURE WORK

To tie the knot on a novel, self-consistent dissertation, this work's primary contributions are reiterated. Further, remaining lines of promising research are encapsulated to help stage future efforts in the study of tsunami debris and debris-fluid-structure interaction. A shortlist of novel contributions by this dissertation are as follow:

1. Development of a novel multi-physics numerical platform on the frontier of hardware infrastructure (i.e. Multi-GPU supercomputers) for tsunami-driven debris-fields. More broadly, this is a numerical approach to debris-fluid-structure interaction (DFSI) problems suited to modern computational demands.

2. Publishing our high-performance, Multi-GPU MPM-FEA code-base as an open-source project, ClaymoreUW, with user and developer documentation. This builds on highly optimized graphics software that has been rebuilt for engineers.

3. Validation of our novel numerical approach to DFSI in a range of demanding cases. Strengths and limitations of our approach are addressed.

4. Stochastic model of fluid-driven debris-field impacts based on experimental and numerical wave-flume tests at multiple wave-flumes.

5. Proposal of a novel characterization scheme for debris-fields at all scales. Establishment of preliminary regression models with statistical significance using simulated in-air and in-water cases.

6. Phrasing of debris-field impact hazards in a novel Performance-Based Engineering context.

These contributions were sourced throughout the document, below we summarize the respective contribution of each constituent chapter in this dissertation and emerging future directions stemming from this research.

12.1 *Fundamentals of the Material Point Method*

A derivation for the traditional Material Point Method (MPM, Section 2.3.1), Affine-Particle-In-Cell (APIC, Section 2.4), and Moving-Least-Squares Material Point Method (MLS-MPM, 2.5) was presented in Chapter 2. This was done in our lab's established nomenclature which first appeared in Mast 2013, bringing the derivations for graphics techniques to engineers. Efforts were made to formalize and contextualize APIC and MLS-MPM derivations, specifically giving a more accessible explanation of their value and better aligning them with a style recognizable to engineering researchers and practitioners.

12.2 *Accelerator Hardware for Engineering Simulations*

We provided a crash-course on GPU programming and have done so in part in this document (Chapter 3). The goal was to bring an engineer without exposure to GPUs to a point where they are comfortable with the basic structure of a GPU program, the theory behind it, and the common terminology surrounding it. Information on using multiple GPUs is provided

12.3 *Bringing the Material Point Method to Multi-GPU*

In Chapter 4 we show the basic steps to make the Material Point Method (MPM) algorithmically manifest across a single or multiple graphics processing units (GPUs). Each device holds thousands of parallel computational cores that may accelerate the numerical method by over 100x if properly employed.

An argument is made for engineers to leverage open-source software and resources from other fields instead of starting from scratch. Claymore (Wang et al. 2020b) is a graphics-oriented software for Multi-GPU MPM and our chose foundation for our open-source project, ClaymoreUW. Claymore is surveyed in Chapter 4 in terms of its design and usage under an engineering-centric lens. To increase accessibility and viability for engineers, we rebuild many aspects of the project to create our own rendition which we call ClaymoreUW.

We cover novel modifications our lab has made to the Claymore software, including the coupling of finite elements, implementation of various advection methods, enhancing computational precision, and a complete overhaul to the user-interface. A more exhaustive list and description of improvements and additions to Claymore made in development of ClaymoreUW are included in Chapter 4. Documentation, for both prospective users and developers, is provided for our ClaymoreUW software within Chapter 4 as well.

12.4 *Design and Implementation of Numerical Methods for Debris-Fluid-Structure Interaction in a Multi-GPU Material Point Method*

In Chapter 5 we develop a novel modelling approach to tackle fluid-driven debris-field hazards. Our numerical tool, a multi-physics coupling of material point method (MPM) (Sulsky et al. 1993) and finite elements (FEM, Section 5.13), scales across Multi-GPUs in high-performance computing centers (HPCs), diluting computational weight. Prioritizing flexibility, speed, and simplicity leads to the most basic algorithm approach for DFSI identified yet. We describe our novel coupling of Moving-Least-Squares Material Point Method, Finite Elements, Affine-Separable Fluid-Implicit Particles (ASFLIP, Section 5.6.7), and our novel, modified B-Spline F-Bar volumetric antilocking on Multi-GPU systems and the benefits provided. Further, we formulate a novel, high-performance material model called the PA-JB Fluid in Section 5.12 specifically optimizing for Multi-GPU usage

12.5 *Validating High-Performance Multi-GPU Material Point Method for Debris-Fluid-Structure Interaction*

Our novel numerical approach requires validation. We do so by isolating physical phenomena of concern into characteristic scenes for simulation. Results from our software are then compared against analytical results and alternative approaches. Numerical results are solutions to the underlying partial differential equations of study. Controlling for boundary and initial conditions, results should be equivalent to those found in reality and other validated numerical codes. Validation of debris-fluid-structure interaction (DFSI) requires **(i)** debris-structure interaction (DSI), **(ii)** debris-fluid interaction (DFI), **(iii)** fluid-structure interaction (FSI), and **(iv)** debris-debris interaction (DDI). Examples for each validation subset are presented in Chapter 6.

12.6 *Debris-Field Impact Experiments and Simulations - Oregon State University's Large Wave Flume*

Significant contributions to the quantification of both ordered and random debris-field configurations were achieved by statistically verified numerical results. In Chapter 7 we demonstrate the integration of advanced computational techniques with coastal engineering through the application of a Multi-GPU Material Point Method, implemented in the open-source ClaymoreUW software. We successfully simulate wave-driven debris field dynamics in a digital twin of the Oregon State University's Large Wave Flume (OSU LWF). The results presented are compelling and highlight the potential of high-performance computing in advancing the understanding of complex, real-world engineering problems.

The physical specification and numerical replication of experiments performed at Oregon State University Large Wave Flume (OSU LWF) are described. Facilities are recreated in our numerical open-source MPM software, at a technologically novel level using Multi-GPUs to simulate the flume with 75 - 150 Million particles 50 times. Further, exa-scale capabilities for wave-flume simulations are demonstrated by modeling the OSU LWF digital twin with 1 billion particles.

Numerical simulations are characterized and preliminary data-analysis of results are given with comparison to real-world results for validation of our approach in both ordered and randomized debris-fields. Preliminary extrapolation to the prototype scale through similitude scaling suggest a significant importance in the quantification of debris-field impact force relative to wave-only force. Our approach has shown an ability to replicate tests, predict tests not yet performed within expected results bounds, and may be of value in the prediction of prototype scale debris-field impacts in the real-world for tsunami events.

12.7 *Motion of Tsunami Debris Through a Model Port City - Three Numerical Methods with Experiments - Waseda University Flume*

In Chapter 8 a set of wave-flume experiments performed at Waseda University (Goseberg et al. 2016) are replicated with and compared between three advanced numerical methods: Multi-GPU Material Point Method (ClaymoreUW), GPU Smoothed Particle Hydrodynamics (DualSPHysics), and computational fluid dynamics (STAR-CCM+). This endeavor sheds light on the strengths and limitations of these numerical techniques in simulating debris motion in tsunami-like conditions. All three methodologies demonstrated their unique capacities to accurately replicate complex fluid-debris-obstacle interactions in a controlled, yet highly representative, harbor environment. Our findings suggest that while the Multi-GPU Material Point Method and GPU Smoothed Particle Hydrodynamics perform exceptionally well in capturing the debris longitudinal and lateral spreading, STAR-CCM+ has shown greater precision in estimating wave-gauge elevations respective to the pressure release system employed at Waseda University's tsunami wave basin. It is important to note, however, that the effectiveness of these methodologies can be scenario-dependent, thus emphasizing the importance of employing a combination of these techniques for comprehensive analysis.

While obstacles were found to have significant influence on the maximum longitudinal displacement of debris, the impact on the spreading angle was found to be less pronounced across

all three methods. This study not only offers a critical foundation for further research in optimizing numerical models for tsunami debris motion but also paves the way for the development of more robust, efficient, and reliable disaster management strategies in the face of increasing threats from extreme maritime conditions. As we continue refining and improving these numerical methodologies, it is our hope that they will enable us to better protect coastal communities worldwide.

12.8 *Multi-Hazard Simulation of Debris Driven by Flows of Water, Snow, Mud, and Soil Generated by Pumps, Pistons, or Gravity using the Material Point Method*

In addition to the validation studies and applications to debris fields presented in Chapters 6, 7, and 8, Chapter 9 presents three additional comparative studies using our numerical method versus experimental results across different debris field hazard manifestations.

Each section in Chapter 9 covers a pilot study investigating the capabilities of our numerical method to generalize across new axis of critical natural hazard behavior. While not developed enough to justify individual chapters, we produce digital twins of three experiment sets at three unique, scaled flume facilities, including: **(i)** Oregon State University's directional wave basin (OSU DWB), **(ii)** University of Washington's Wind-Air Sea Interaction Research Facility (UW WASIRF), and **(iii)** the U.S. Geological Survey's Volcanic Hazard Debris-Flow Flume (USGS Lahar flume).

12.9 *Debris-Field Characterization.*

Chapter 10 reveals debris-fields as quantitative bodies using a novel characterization scheme. By exploring characteristics that describe debris-fields (e.g. debris count, occupation, orientation) we find independent variables for analysis. By considering basic needs of engineers designing for debris-field impacts we find dependent variables (e.g. impulse, max force, impact duration). Statistical analysis of 100+ debris-field simulations reveal relationships between knowable

independent variables and desired dependent variables in the context of an in-air numerical study. These insights can be used to develop a novel but preliminary framework of probabilistic design, presented in Chapter 11.

12.10 *Performance-Based Framework.*

Informed by experimental and numerical results, we take initial steps into a novel performance-based engineering framework of design for tsunami-driven debris-field hazards where a new characterization scheme is proposed. This work has great potential to be developed into existing PBE pipelines, such as Performance-Based Tsunami Engineering (PBTE, Attary et al. 2019) and the overall computational pipeline of the Natural Hazard Engineering Research Institutes' SimCenter.

12.11 *A Numerical Platform for Simulating Nearly Any Multi-Hazard Scenario*

Our approach to Debris-Fluid-Structure Interaction within high-performance Material Point Method is applicable to a swath of natural hazards. While tsunami hazards were numerically replicated in scaled experimental wave-flumes in Chapters 7 and 8 and Sections 9.3 and 9.2, this is not all that MPM is capable of and arguably this hazard is not even one of its primary strengths. With moderate effort, we took the exact methodology used in our tsunami-driving debris-fields into structures simulations and applied it to lahars-driving debris-fields into structures simulations in Section 9.4. This unusual ease in application to a fundamentally different hazard mode is significant. While depth-integrated methods and other advanced tools, e.g. smoothed particle hydrodynamics, may also exhibit cross-applicability it is not with anywhere near the level of abstraction MPM exhibits: only requiring a change in material constitutive law, which is isolated on particles to allow elegant coupling with other material, e.g. other hazards, debris, and structures by default. Further, MPM excels at advanced material laws compared to idealized fluids

oriented schemes, which excel much greater but constrained within their limited scope. The benefits MPM may possess, as a Jack of all trades, for the natural hazards modeling community could be immense. Namely, the ability to rapidly assess multi-hazard scenarios in complex environments may expedite the hazard identification, assessment, and mitigation process, allowing more time be devoted to tailored, specialist tools when the primary mode of concern is identified.

MPM is likely applicable to storm-surges and other waterborne hazards beyond tsunamis. This is an increasingly key hazard to quantify as sea-levels are anticipated to rise by 0.3 and 0.8 meters by the 2030s and 2080s, which may produce storm-surges inundation depths of 0.5 to 1.8 meters respectively Mousavi et al. 2011. While our simulations of wave-flumes primarily applies to tsunamis due to use of spilling solitary waves in almost all cases, we did observe fundamental effects of debris motion in a steady-state undulating flow in our UW WASIRF wind-wave-flume (see Section 9.3) and we replicated the reorganizing behavior of heterogeneous density (mix of wood and plastic) debris-field motion that was observed by Park et al. 2021 at the OSU DWB wave-flume. There are likely great applications in MPM for wind-water hazards with debris field motion and loading.

Avalanches have been a growing topic of study in MPM, though our group has investigated them back in Mast et al. 2014 along with landslides in the context of mitigation. As MPM can 3D model the entire event (static, triggering, collapse, flow, run-out, rest, etc.) (di et al. 2023) while interacting with debris and structures it shows unique promise for these hazards. Recent advances in this field are noted in Gaume et al. 2023, whom identified a shortlist of MPM's key applications including: **(i)** snow micro-structure deformation and failure, **(ii)** avalanche release (Guillet et al. 2023) (Gaume et al. 2019), **(iii)** avalanche dynamics over complex topography Li et al. 2020, **(iv)** interaction between avalanches and obstacles, forests or lakes (Védrine et al. 2022), and **(v)** erosion and entrainment (Védrine et al. 2022). The fourth point concerns structural impacts of debris carried by avalanches, much like our work but for tsunamis and debris. Depth-averaged

MPM tools are also shown to be an interesting line of pursuit for slope-scale prototype modeling of avalanches in both Gaume et al. 2023 and Guillet et al. 2023.

Our Multi-GPU MPM approach shows strong preliminary promise in application to alternative flumes than those that formed the bulk of our simulation efforts, i.e. the OSU LWF and Waseda University's flume. Wide wave-flumes, pump-based wave-flumes, and gravity-driven lahar-flumes all showed decent behavior in the first few-passes of simulations with fairly minor effort. The applicability to debris-flows driving debris-fields is especially intriguing, as it suggests our MPM approach may apply to not just tsunamis, but lahars as well. Further, there is reason to believe it can apply to avalanches (Stomakhin et al. 2013) and storm-surges, though these efforts are not fully fleshed out yet in our own work. This would constitute a minimum of four natural hazards of interest, with the secondary hazard of debris-fields appending to each. MPM may be the future direction for a wide variety of complex, large-deformation natural hazard numerical tools.

BIBLIOGRAPHY

[Abdolali and Kirby 2017] Abdolali, A. and Kirby, J. T. (2017). "Role of compressibility on tsunami propagation." *Journal of Geophysical Research: Oceans*, 122, 9780–9794.

[Aghl et al. 2014a] Aghl, P. P., Naito, C. J., and Riggs, H. R. (2014a). "Full-scale experimental study of impact demands resulting from high mass, low velocity debris." *Journal of Structural Engineering*, 140, 04014006 doi: 10.1061/(ASCE)ST.1943-541X.0000948.

[Aghl et al. 2014b] Aghl, P. P., Naito, C. J., and Riggs, H. R. (2014b). "Investigating the effect of nonstructural mass on impact forces from elastic debris." *Proceedings of the 2014 Structures Congress*, American Society of Civil Engineers, 635–644.

[Aslami et al. 2023] Aslami, M. H., Rogers, B. D., Stansby, P. K., and Bottacin-Busolin, A. (2023). "Simulation of floating debris in SPH shallow water flow model with tsunami application." *Advances in Water Resources*, 171, 104363.

[Attary et al. 2019] Attary, N., Lindt, J. W. V. D., Barbosa, A. R., Cox, D. T., and Unnikrishnan, V. U. (2019). "Performance-based tsunami engineering for risk assessment of structures subjected to multi-hazards: Tsunami following earthquake." *https://doi.org/10.1080/13632469.2019.1616335*, 25, 2065–2084.

[Bai and Schroeder 2022] Bai, S. and Schroeder, C. (2022). "Stability analysis of explicit mpm." *Computer Graphics Forum*, 41, 19–30.

[Band et al. 2018] Band, S., Gissler, C., Ihmsen, M., Cornelis, J., Peer, A., and Teschner, M. (2018). "Pressure boundaries for implicit incompressible sph." *ACM Transactions on Graphics*, 37.

[Bathe 2001] Bathe, K.-J. (2001). "The inf–sup condition and its evaluation for mixed finite element methods." *Computers Structures*, 79, 243–252.

[Becker and Teschner 2007] Becker, M. and Teschner, M. (2007). "Weakly compressible SPH for free surface flows." *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Eurographics Association, 209–217.

[Belytschko et al. 1994] Belytschko, T., Lu, Y. Y., and Gu, L. (1994). "Element-free galerkin methods." *International Journal for Numerical Methods in Engineering*, 37, 229–256.

[Bonus 2023] Bonus, J. (2023). "Multi-scale simulation of tsunami-driven debris-field impacts on structures via a multi-GPU material point method." PhD thesis, University of Washington, Seattle, WA, United States.

[Bonus et al. 2022] Bonus, J., Arduino, P., Motley, M., and Eberhard, M. (2022). "Multi-scale numerical simulation of tsunami-driven debris-field impacts." *Proceeedings of PORTS '22*, American Society of Civil Engineers, 328–339 (9).

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

[Chandra et al. 2023] Chandra, B., Hashimoto, R., Pérez, M. M., and Soga, K. (2023). "High-performance, high-order implicit material point method for progressive levee failure simulations." 85–95 (11).

[Chen et al. 2020] Chen, X.-S., Li, C.-F., Cao, G.-C., Jiang, Y.-T., and Hu, S.-M. (2020). "A moving least square reproducing kernel particle method for unified multiphase continuum simulation." *ACM Trans. Graph.*, 39.

[Chen et al. 2018] Chen, Z.-P., Zhang, X., Sze, K. Y., Kan, L., and Qiu, X.-M. (2018). "v-p material point method for weakly compressible problems." *Computers Fluids*, 176, 170–181.

[Chock 2016] Chock, G. Y. (2016). "Design for tsunami loads and effects in the asce 7-16 standard." *J. Struct. Eng*, 142, 4016093.

[Cinar et al. 2022] Cinar, G. E., Keen, A., and Lynett, P. (2022). "Motion of a debris line source under currents and waves: Experimental study." *Journal of Waterway, Port, Coastal and Ocean Engineering*, 149.

[Dass and Kumari 1985] Dass, N. and Kumari, M. (1985). "Derivation of some equations of state for solids. a new approach." *physica status solidi (b)*, 127, 103–108.

[de Vaucorbeil et al. 2020] de Vaucorbeil, A., Nguyen, V. P., and Hutchinson, C. R. (2020). "A total-lagrangian material point method for solid mechanics problems involving large deformations." *Computer Methods in Applied Mechanics and Engineering*, 360, 112783.

[di et al. 2023] di, W., Wang, B., Yuan, W.-H., and Liu, L. (2023). "Investigation of rainfall intensity on the slope failure process using gpu-accelerated coupled mpm." *Computers and Geotechnics*, 163, 105718.

[Domínguez et al. 2022] Domínguez, J. M., Fourtakas, G., Altomare, C., Canelas, R. B., Tafuni, A., García-Feal, O., Martínez-Estévez, I., Mokos, A., Vacondio, R., Crespo, A. J. C., Rogers, B. D., Stansby, P. K., and Gómez-Gesteira, M. (2022). "DualSPHysics: from fluid dynamics to multiphysics problems." *Computational Particle Mechanics*, 9, 867–895.

[Dong and Grabe 2018] Dong, Y. and Grabe, J. (2018). "Large scale parallelisation of the material point method with multiple GPUs." *Computers and Geotechnics*, 101, 149–158.

[Efthimiou and Johnson 2007] Efthimiou, C. and Johnson, M. (2007). "Domino Waves." *SIAM Review*, 49.

[Fei et al. 2021a] Fei, Y., Huang, L., Guo, Q., Wu, R., and Gao, M. (2021a). "Revisiting integration in the material point method: A scheme for easier separation and less dissipation." *ACM Trans. Graph*, 40, 1–16.

[Fei et al. 2021b] Fei, Y., Huang, Y., and Gao, M. (2021b). "Principles towards real-time simulation of material point method on modern gpus.

[Fu et al. 2017] Fu, C., Guo, Q., Gast, T., Jiang, C., and Teran, J. (2017). "A polynomial particle-in-cell method." *ACM Transactions on Graphics*, Vol. 36, Association for Computing Machinery (11).

[Gao et al. 2018a] Gao, M., Wang, X., Wu, K., Pradhana, A., Sifakis, E., Yuksel, C., and Jiang, C. (2018a). "Gpu optimization of material point methods." Association for Computing Machinery, Inc (12).

[Gao et al. 2018b] Gao, M., Wang, X., Wu, K., Pradhana, A., Sifakis, E., Yuksel, C., and Jiang, C. (2018b). "GPU optimization of material point methods." Association for Computing Machinery (12).

[Gaume et al. 2023] Gaume, J., Blatny, L., Bobillier, G., Guillet, L., Kohler, M., Kyburz, M., Li, X., Meloche, F., Sovilla, B., Trottet, B., and Louis, V. (2023). "Recent advances in modeling snow and avalanches with the material point method and practical implications (11).

[Gaume et al. 2019] Gaume, J., van Herwijnen, A., Gast, T., Teran, J., and Jiang, C. (2019). "Investigating the release and flow of snow avalanches at the slope-scale using a unified model based on the material point method." *Cold Regions Science and Technology*, 168, 102847.

[Goseberg et al. 2016] Goseberg, N., Stolle, J., Nistor, I., and Shibayama, T. (2016). "Experimental analysis of debris motion due the obstruction from fixed obstacles in tsunami-like flow conditions." *Coastal Engineering*, 118, 35–49.

[Goseberg et al. 2013] Goseberg, N., Wurpts, A., and Schlurmann, T. (2013). "Laboratory-scale generation of tsunami and long waves." *Coastal Engineering*, 79, 57–74.

[Guillet et al. 2023] Guillet, L., Blatny, L., Trottet, B., and Gaume, J. (2023). "A depth-averaged material point method for shallow landslides: Applications to snow slab avalanche release.

[H. 1948] H., C. R. (1948). *Underwater explosions*. Princeton, Princeton Univ. Press, 1948, <https://www.biodiversitylibrary.org/item/102818> https://www.biodiversitylibrary.org/bibliography/48411.

[Hammerquist and Nairn 2017] Hammerquist, C. C. and Nairn, J. A. (2017). "A new method for material point method particle updates that reduces noise and enhances stability." *Computer Methods in Applied Mechanics and Engineering*, 318, 724–738.

[Harlow 1962] Harlow, F. H. (1962). "The particle-in-cell method for numerical solution of problems in fluid dynamics." *Los Alamos National Lab Technical Report*.

[Hasanpour 2023] Hasanpour, A. (2023). "Numerical investigation of tsunami-borne debris interaction with coastal bridges." PhD thesis, University of Nevada, Reno, Nevada, United States (5).

[Hasanpour et al. 2021] Hasanpour, A., Istrati, D., and Buckle, I. (2021). "Coupled SPH–FEM modeling of tsunami-borne large debris flow and impact on coastal structures." *Journal of Marine Science and Engineering*, 9, 1068.

[Hasanpour et al. 2023] Hasanpour, A., Istrati, D., and Buckle, I. (2023). "Three-dimensional numerical investigation of floating debris effects on bridge superstructures during tsunamis." *Ocean Engineering*, 289, 116262.

[Housner 1963] Housner, G. (1963). "The dynamic behavior of water tanks." *Bulletin of the Seismological Society of America*, 53, 381–387.

[Hu et al. 2018] Hu, Y., Csail, M., Fang, Y. U., Ge, Z., Zhu, Y., Fang, Y., Qu, Z., Prad-Hana, A., and Jiang, C. (2018). "A moving least squares material point method with displacement discontinuity and two-way rigid body coupling method with displacement discontinuity and two-way rigid body coupling." *ACM Trans. Graph*, 37, 146.

[Hu et al. 2019a] Hu, Y., Li, T.-M., Anderson, L., Ragan-Kelley, J., and Durand, F. (2019a). "Taichi: a language for high-performance computation on spatially sparse data structures." *ACM Transactions on Graphics (TOG)*, 38, 201.

[Hu et al. 2019b] Hu, Y., Li, T.-M., Anderson, L., Ragan-Kelley, J., and Durand, F. (2019b). "Taichi: a language for high-performance computation on spatially sparse data structures." *ACM Transactions on Graphics (TOG)*, 38(6), 201.

[Huang et al. 2008] Huang, P., Zhang, X., Ma, S. W., and Wang, H. K. (2008). "Shared memory openmp parallelization of explicit mpm and its application to hypervelocity impact." *CMES - Computer Modeling in Engineering & Sciences*, 38, 119–148.

[Huerta et al. 2004] Huerta, A., Vidal, Y., and Villon, P. (2004). "Pseudo-divergence-free element free galerkin method for incompressible fluid flow.

[Idelsohn et al. 2008] Idelsohn, S. R., Marti, J., Souto-Iglesias, A., and Oñate, E. (2008). "Interaction between an elastic structure and free-surface flows: experimental versus numerical comparisons using the PFEM." *Computational Mechanics*, 43, 125–132.

[Irving et al. 2004] Irving, G., Teran, J., and Fedkiw, R. (2004). "Invertible Finite Elements for Robust Simulation of Large Deformation." *Symposium on Computer Animation*, R. Boulic and D. K. Pai, eds., The Eurographics Association.

[Irving et al. 2006] Irving, G., Teran, J., and Fedkiw, R. (2006). "Tetrahedral and hexahedral invertible finite elements." *Graphical Models*, 68, 66–89.

[Issa et al. 2010] Issa, R., Violeau, D., Lee, E.-S., and Flament, H. (2010). *Modelling nonlinear water waves with RANS and LES SPH models*. World Scientific, 497–537.

[Iverson et al. 2010] Iverson, R. M., Logan, M., LaHusen, R. G., and Berti, M. (2010). "The perfect debris flow? aggregated results from 28 large-scale experiments." *Journal of Geophysical Research F: Earth Surface*, 115, F03005–.

[Jaiswal et al. 2008] Jaiswal, O. R., Kulkarni, S., and Pathak, P. K. (2008). "A study on sloshing frequencies of fluid-tank system." *Proceedings of the 14th World Conference on Earthquake Engineering*, International Association for Earthquake Engineering (10).

[Jiang et al. 2015] Jiang, C., Schroeder, C., Selle, A., Teran, J., and Stomakhin, A. (2015). "The affine particle-in-cell method." *ACM Transactions on Graphics*, Vol. 34, Association for Computing Machinery, 1–10 (7).

[Jiang et al.] Jiang, C., Schroeder, C., Teran, J., Stomakhin, A., and Selle, A. "The material point method for simulating continuum materials.

[Jiang et al. 2016] Jiang, C., Schroeder, C., Teran, J., Stomakhin, A., and Selle, A. (2016). "The material point method for simulating continuum materials." *SIGGRAPH 2016 Courses*, Association for Computing Machinery.

[Jing et al. 2002] Jing, Q., Mukherjee, T., and Fedder, G. (2002). "Large-deflection beam model for schematic-based behavioral simulation in NODAS." *2002 International Conference on Modeling and Simulation of Microsystems* (4).

[Kang et al. 2021] Kang, J., Homel, M., and Herbold, E. (2021). "Beam elements with frictional contact in the material point method." *International Journal for Numerical Methods in Engineering*, 123.

[Kennedy et al. 2021] Kennedy, A., Barra, J. M., Blunk, A. V., Lynett, P., Keen, A., and Lomonaco, P. (2021). "Array and debris loading, <https://www.designsafe-ci.org/data/browser/public/designsafe.storage.published/PRJ-3157/details-4082949221061881365-242ac117-0001-012>.

[Kumar et al. 2019] Kumar, K., Salmond, J., Kularathna, S., Wilkes, C., Tjung, E., Biscontin, G., and Soga, K. (2019). "Scalable and modular material point method for large-scale simulations, <https://arxiv.org/abs/1909.13380>.

[Leeuwen 2010] Leeuwen, J. M. J. V. (2010). "The domino effect." *American Journal of Physics*, 78, 721–727.

[Lewis 2023] Lewis, N. (2023). "Development of an open-source methodology for simulation of civil engineering structures subject to multi-hazards." PhD thesis, University of Washington, Seattle, WA, United States (11).

[Lewis et al. 2023] Lewis, N. S., Winter, A. O., Bonus, J., Motley, M. R., Eberhard, M. O., Arduino, P., and Lehman, D. E. (2023). "Open-source simulation of strongly-coupled fluid-structure interaction between non-conformal interfaces." *Frontiers in Built Environment*, 9.

[Li et al. 2023] Li, J., Wang, B., di, W., Zhang, P., and Vardon, P. (2023). "A coupled mpm-dem method for modelling soil-rock mixtures." *Computers and Geotechnics*, 160, 105508.

[Li et al. 2020] Li, X., Sovilla, B., Jiang, C., and Gaume, J. (2020). "The mechanical origin of snow avalanche dynamics and flow regime transitions." *The Cryosphere*, 14, 3381–3398.

[Liu et al. 1995] Liu, W. K., Jun, S., and Zhang, Y. F. (1995). "Reproducing kernel particle methods." *International Journal for Numerical Methods in Fluids*, 20, 1081–1106.

[Lomonaco et al. 2018] Lomonaco, P., Alam, M. S., Arduino, P., Barbosa, A., Cox, D. T., Do, T., Eberhard, M., Motley, M., Shekhar, K., Tomiczek, T., Park, H., van de Lindt, J. W., and Winter, A. (2018). "Experimental modeling of wave forces and hydrodynamics on elevated coastal structures subject to waves, surge or tsunamis: The effect of breaking, shielding and debris." *Coastal Engineering Proceedings*, 53–53.

[Ma et al. 2009] Ma, S., Zhang, X., and Qiu, X. M. (2009). "Comparison study of mpm and sph in modeling hypervelocity impact problems." *International Journal of Impact Engineering*, 36, 272–282.

[MacDonald 1966] MacDonald, J. R. (1966). "Some simple isothermal equations of state." *Rev. Mod. Phys.*, 38, 669–679.

[MacDonald 1969] MacDonald, J. R. (1969). "Review of some experimental and analytical equations of state." *Rev. Mod. Phys.*, 41, 316–349.

[Madsen et al. 2008] Madsen, P. A., Fuhrman, D. R., and Schäffer, H. A. (2008). "On the solitary wave paradigm for tsunamis." *Journal of Geophysical Research: Oceans*, 113.

[Majtan et al. 2022] Majtan, E., Cunningham, L. S., and Rogers, B. D. (2022). "Experimental and numerical investigation of floating large woody debris impact on a masonry arch bridge." *Journal of Marine Science and Engineering*, 10.

[Mankowitz et al. 2023] Mankowitz, D. J., Michi, A., Zhernov, A., Gelmi, M., Selvi, M., Paduraru, C., Leurent, E., Iqbal, S., Lespiau, J.-B., Ahern, A., Köppe, T., Millikin, K., Gaffney, S., Elster, S., Broshear, J., Gamble, C., Milan, K., Tung, R., Hwang, M., Cemgil, T., Barekatain, M., Li, Y., Mandhane, A., Hubert, T., Schrittwieser, J., Hassabis, D., Kohli, P., Riedmiller, M., Vinyals, O., and Silver, D. (2023). "Faster sorting algorithms discovered using deep reinforcement learning." *Nature*, 618, 257–263.

[Mascarenas 2022] Mascarenas, D. (2022). "Experimental evaluation of loads from inundation-driven debris fields." Masters thesis, University of Washington, Seattle, WA.

[Mascarenas et al. 2022] Mascarenas, D., Motley, M. R., Eberhard, M. O., Arduino, P., and Serrone, A. (2022). "Quantifying and understanding structural loading from wave-driven debris fields." 523–534.

[Mast 2013] Mast, C. M. (2013). "Modeling landslide-induced flow interactions with structures using the material point method." PhD thesis, University of Washington, Seattle, WA.

[Mast et al. 2014] Mast, C. M., Arduino, P., Miller, G. R., and Mackenzie-Helnwein, P. (2014). "Avalanche and landslide simulation using the material point method: flow dynamics and force interaction with structures." *Computational Geosciences*, 18, 817–830.

[McKenna 2011] McKenna, F. (2011). "OpenSees: A framework for earthquake engineering simulation." *Computing in Science & Engineering*, 13, 58–66.

[McLachlan et al. 1983] McLachlan, B. G., Beaupre, G., Cox, A. B., and Gore, L. (1983). "Falling dominoes." *SIAM Review*, 25, 403–404 doi: 10.1137/1025085.

[Mei et al. 2020] Mei, X., Ma, G., Hu, X., Shi, B., and Huang, D. (2020). "Simulation of the failure process of submarine landslides on a seabed by coupled mpm." *Journal of Coastal Research*, 111, 231 – 237.

[Moehle and Deierlein 2004] Moehle, J. and Deierlein, G. (2004). "A framework methodology for performance-based earthquake engineering." Vol. 369 (8).

[Moris et al. 2023] Moris, J. P., Burke, O., Kennedy, A. B., and Westerink, J. J. (2023). "Wave–current impulsive debris loading on a coastal building array." *Journal of Waterway, Port, Coastal, and Ocean Engineering*, 149, 04022025 doi: 10.1061/(ASCE)WW.1943-5460.0000731.

[Mousavi et al. 2011] Mousavi, M. E., Irish, J. L., Frey, A. E., Olivera, F., and Edge, B. L. (2011). "Global warming and hurricanes: the potential impact of hurricane intensification and sea level rise on coastal flooding." *Climatic Change*, 104, 575–597.

[Murnaghan 1944] Murnaghan, F. D. (1944). "The compressibility of media under extreme pressures." *Proceedings of the National Academy of Sciences*, 30, 244–247.

[Nairn and Hammerquist 2021] Nairn, J. A. and Hammerquist, C. C. (2021). "Material point method simulations using an approximate full mass matrix inverse." *Computer Methods in Applied Mechanics and Engineering, in press*.

[Naito et al. 2014] Naito, C., Cercone, C., Riggs, H. R., and Cox, D. (2014). "Procedure for site assessment of the potential for tsunami debris impact." *Journal of Waterway, Port, Coastal, and Ocean Engineering*, 140, 223–232.

[Nakamura et al. 2021] Nakamura, K., Matsumura, S., and Mizutani, T. (2021). "Particle-to-surface frictional contact algorithm for material point method using weighted least squares." *Computers and Geotechnics*, 134, 104069.

[Neto et al. 2005] Neto, E. A. S., Pires, F. M. A., and Owen, D. R. J. (2005). "F-bar-based linear triangles and tetrahedra for finite strain analysis of nearly incompressible solids. part i: formulation and benchmarking." *International Journal for Numerical Methods in Engineering*, 62(3), 353–383.

[Nguyen et al. 2023] Nguyen, V. P., de Vaucorbeil, A., and Bordas, S. (2023). *Karamelo: A Multi-CPU/GPU C++ Parallel MPM Code*. Springer International Publishing, 205–225.

[Nistor et al. 2017a] Nistor, I., Goseberg, N., and Stolle, J. (2017a). "Tsunami-driven debris motion and loads: A critical review." *Frontiers in Built Environment*, 3, 2.

[Nistor et al. 2017b] Nistor, I., Goseberg, N., Stolle, J., Mikami, T., Shibayama, T., Nakamura, R., and Matsuba, S. (2017b). "Experimental investigations of debris dynamics over a horizontal plane." *Journal of Waterway, Port, Coastal, and Ocean Engineering*, 143, 04016022 doi: 10.1061/(ASCE)WW.1943-5460.0000371.

[NVIDIA 2022] NVIDIA (2022). "CUDA Math API Reference Manual (1).

[NVIDIA 2023] NVIDIA (2023). "CUDA C++ Programming Guide Release 12.1 NVIDIA.

[Paczkowski et al. 2012] Paczkowski, K., Riggs, H. R., Naito, C., and Lehmann, A. (2012). "A one-dimensional model for impact forces resulting from high mass, low velocity debris." *Structural Engineering and Structural Mechanics*, 42, 831–847.

[Park et al. 2021] Park, H., Koh, M.-J., Cox, D. T., Alam, M. S., and Shin, S. (2021). "Experimental study of debris transport driven by a tsunami-like wave: Application for non-uniform density groups and obstacles." *Coastal Engineering*, 166, 103867.

[Parker 2006] Parker, S. G. (2006). "A component-based architecture for parallel multi-physics pde simulation." *Future Generation Computer Systems*, 22, 204–216.

[Poirier 2000] Poirier, J.-P. (2000). *Introduction to the Physics of the Earth's Interior*. Cambridge University Press, 2 edition.

[Polukoshko et al. 2007] Polukoshko, S., Viba, J., Kononova, O., and Sokolova, S. (2007). "Rigid body impact models partially considering deformation." *Proc. Estonian Acad. Sci. Eng*, 13, 140–155.

[Pradhana et al. 2017] Pradhana, A., Teran, J., Tampubolon, A. P., Gast, T., Klár, G., Fu, C., Jiang, C., and Museth, K. (2017). "Multi-species simulation of porous sand and water mixtures." *ACM Trans. Graph*, 36.

[Qiao et al. 2023] Qiao, Z., Li, T., Simoni, A., Gregoretti, C., Bernard, M., Wu, S., Shen, W., and Berti, M. (2023). "Numerical modelling of an alpine debris flow by considering bed entrainment." *Frontiers in Earth Science*, 10.

[Qiu et al. 2023] Qiu, Y., Reeve, S. T., Li, M., Yang, Y., Slattery, S. R., and Jiang, C. (2023). "A sparse distributed gigascale resolution material point method." *ACM Trans. Graph.*, 42.

[Ruggirello and Schumacher 2013] Ruggirello, K. P. and Schumacher, S. C. (2013). "A comparison of parallelization strategies for the material point method." 2014 World Congress on Computational Mechanics.

[She and Leveque 1994] She, Z.-S. and Leveque, E. (1994). "Universal scaling laws in fully developed turbulence." *Phys. Rev. Lett.*, 72, 336–339.

[Shekhar et al. 2020] Shekhar, K., Winter, A. O., Alam, M. S., Arduino, P., Miller, G. R., Motley, M. R., Eberhard, M. O., Barbosa, A. R., Lomonaco, P., and Cox, D. T. (2020). "Conceptual evaluation of tsunami debris field damming and impact forces." *Journal of Waterway, Port, Coastal, and Ocean Engineering*, 146, 04020039.

[Shin 2007] Shin, H.-S. (2007). "Experimental and numerical study on seismic response of pile-supported structures.

[Shin 2009] Shin, W. (2009). "Modeling Mixing and Separation of Solid Matter and Fluid in Landslides and Debris Flows by Representing the Multiphase Material through Distinct Phases." PhD thesis, University of Washington, Seattle, WA, United States.

[Song et al. 2021] Song, G., Guo, X., and Sun, B.-H. (2021). "Scaling law for velocity of domino toppling motion in curved paths." *Open Physics*, 19, 426–433.

[Stolle et al. 2016] Stolle, J., Nistor, I., and Goseberg, N. (2016). "Optical tracking of floating shipping containers in a high-velocity flow." *Coastal Engineering Journal*, 58, 1650005.

[Stolle et al. 2020] Stolle, J., Nistor, I., Goseberg, N., and Petriu, E. (2020). "Development of a probabilistic framework for debris transport and hazard assessment in tsunami-like flow conditions." *Journal of Waterway, Port, Coastal and Ocean Engineering*, 146, 4020026.

[Stomakhin et al. 2013] Stomakhin, A., Schroeder, C., Chai, L., Teran, J., and Selle, A. (2013). "A material point method for snow simulation." *ACM Trans. Graph.*, 32.

[Stronge 1987] Stronge, W. J. (1987). "The domino effect: A wave of destabilizing collisions in a periodic array." *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 409, 199–208.

[Sulsky et al. 1993] Sulsky, D., Chen, Z., and Schreyer, H. L. (1993). "A particle method for history-dependent materials." *Other Information: PBD: Jun 1993*.

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

[Sun 2020] Sun, B.-H. (2020). “Scaling law for the propagation speed of domino toppling.” *AIP Advances*, 10, 095124.

[Sun et al. 2018] Sun, Z., Li, H., Gan, Y., Liu, H., Huang, Z., and He, L. (2018). “Material point method and smoothed particle hydrodynamics simulations of fluid flow problems: A comparative study.” *Progress in Computational Fluid Dynamics, An International Journal*, 18, 1.

[Tait 1888] Tait, P. G. (1888). *Report on some of the physical properties of fresh water and of sea water*, Vol. II, Part IV. Physics and Chemistry (5) . Sourced from the 1911 reprint of the "*Report on the scientific results of the voyage of H.M.S. Challenger during the years 1873-76.*", available digitally at https://escholarship.org/uc/item/7cs1q13k#main.

[Takabatake et al. 2021] Takabatake, T., Stolle, J., Hiraishi, K., Kihara, N., Nojima, K., Shigihara, Y., Arikawa, T., and Nistor, I. (2021). “Inter-model comparison for tsunami debris simulation.” *Journal of Disaster Research*, 16, 1030–1044.

[Tran et al. 2022] Tran, Q. A., Grimstad, G., and Amiri, S. A. G. (2022). “MPMICE: A hybrid MPM-CFD model for simulating coupled problems in porous media. Application to earthquake-induced submarine landslides.

[Tran and Sołowski 2019] Tran, Q. A. and Sołowski, W. (2019). “Temporal and null-space filter for the material point method.” *International Journal for Numerical Methods in Engineering*, 120, 328–360.

[Tran et al. 2023] Tran, Q.-A., Sørlie, E., Grimstad, G., and Eiksund, G. (2023). “Role of hydraulic conductivity on the mechanism of earthquake induced submarine landslides -a cfd-mpm analysis.” *10th European Conference on Numerical Methods in Geotechnical Engineering* (7).

[Vittoz et al. 2019] Vittoz, L., Oger, G., de Leffe, M., and Touzé, D. L. (2019). “Comparisons of weakly-compressible and truly incompressible approaches for viscous flow into a high-order cartesian-grid finite volume framework.” *Journal of Computational Physics: X*, 1, 100015.

[Védrine et al. 2022] Védrine, L., Li, X., and Gaume, J. (2022). “Detrainment and braking of snow avalanches interacting with forests.” *Natural Hazards and Earth System Sciences*, 22, 1015–1028.

[Walhorn et al. 2005] Walhorn, E., Kölke, A., Hübner, B., and Dinkler, D. (2005). “Fluid–structure coupling within a monolithic model involving free surface flows.” *Computers & Structures*, 83, 2100–2111.

[Wang et al. 2020a] Wang, X., Qiu, Y., Slattery, S. R., Fang, Y., Li, M., Zhu, S., Zhu, Y., Tang, M., Manocha, D., and Jiang, C. (2020a). "A massively parallel and scalable multi-GPU material point method supplementary technical document." *ACM Trans. Graph*, 39, 4, Article 1, 4.

[Wang et al. 2020b] Wang, X., Qiu, Y., Slattery, S. R., Fang, Y., Li, M., Zhu, S.-C., Zhu, Y., Tang, M., Manocha, D., and Jiang, C. (2020b). "A massively parallel and scalable multi-GPU material point method." *ACM Trans. Graph.*, 39.

[Winter 2019] Winter, A. (2019). "Effects of flow shielding and channeling on tsunami-induced loading of coastal structures." PhD thesis, University of Washington, Seattle, WA, United States (9).

[Winter et al. 2020] Winter, A. O., Alam, M. S., Shekhar, K., Motley, M. R., Eberhard, M. O., Barbosa, A. R., Lomonaco, P., Arduino, P., and Cox, D. T. (2020). "Tsunami-like wave forces on an elevated coastal structure: Effects of flow shielding and channeling." *Journal of Waterway, Port, Coastal, and Ocean Engineering*, 146, 04020021.

[Wu et al. 2014] Wu, T.-R., Chu, C.-R., Huang, C.-J., Wang, C.-Y., Chien, S.-Y., and Chen, M.-Z. (2014). "A two-way coupled simulation of moving solids in free-surface flows." *Computers & Fluids*, 100, 347–355.

[Xie et al. 2023] Xie, M., Navas, P., and López-Querol, S. (2023). "An implicit locking-free b-spline material point method for large strain geotechnical modelling." *International Journal for Numerical and Analytical Methods in Geomechanics*, 47, 2741–2761.

[Yang et al. 2019] Yang, B., Corse, W., Lu, J., Wolper, J., and Jiang, C.-F. (2019). "Real-time fluid simulation on the surface of a sphere." *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 2, 1–17.

[Yang 2016] Yang, W.-C. (2016). "Study of tsunami-induced fluid and debris load on bridges using the material point method." PhD thesis, University of Washington, Seattle, WA (5).

[Yang et al. 2017a] Yang, W. C., Shekhar, K., Arduino, P., Mackenzie-Helnwein, P., and Miller, G. (2017a). "Modeling tsunami induced debris impacts on bridge structures using the material point method." *Procedia Engineering*, 175, 175–181.

[Yang et al. 2017b] Yang, W.-C., Shekhar, K., Arduino, P., Mackenzie-Helnwein, P., and Miller, G. (2017b). "Modeling tsunami induced debris impacts on bridge structures using the material point method." *Procedia Engineering*, 175, 175–181.

[Yeh et al. 2013] Yeh, H., Sato, S., and Tajima, Y. (2013). "The 11 March 2011 East Japan earthquake and tsunami: Tsunami effects on coastal infrastructure and buildings." *Pure and Applied Geophysics*, 170, 1019–1031.

[Zhang et al. 2017] Zhang, F., Zhang, X., Sze, K. Y., Lian, Y., and Liu, Y. (2017). "Incompressible material point method for free surface flow." *Journal of Computational Physics*, 330, 92–110.

[Zhang et al. 2023] Zhang, W., Wu, Z., Peng, C., li, S., Dong, Y., and Yuan, W.-H. (2023). "Modelling large-scale landslide using a GPU-accelerated 3D MPM with an efficient terrain contact algorithm." *Computers and Geotechnics*, 158, 105411.

[Zhang et al. 2010] Zhang, Y., Zhang, X., and Liu, Y. (2010). "An alternated grid updating parallel algorithm for material point method using OpenMP." *Cmes-computer Modeling in Engineering & Sciences*, 69, 143–166.

[Zhao et al. 2023] Zhao, Y., Jiang, C., and Choo, J. (2023). "Circumventing volumetric locking in explicit material point methods: A simple, efficient, and general approach." *arXiv math.NA* (10). Preprint available at https://doi.org/10.48550/arXiv.2209.02466.

[Zheng et al. 2021] Zheng, X., Pisanò, F., Vardon, P. J., and Hicks, M. A. (2021). "An explicit stabilised material point method for coupled hydromechanical problems in two-phase porous media." *Computers and Geotechnics*, 135, 104112.

[Zhu and Bridson 2005] Zhu, Y. and Bridson, R. (2005). "Animating sand as a fluid." *ACM SIGGRAPH 2005 Papers*.